**UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA SUR**

**FACULTAD DE INGENIERÍA Y GESTIÓN**

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

# “PROPUESTA DE IMPLEMENTACIÓN PARA LA MEJORA DEL PROCESO DE MIGRACIÓN DE LÍNEAS POSTPAGO BASADO EN UNA ARQUITECTURA ORIENTADA A SERVICIOS EN LA EMPRESA CLARO PERÚ”

**TRABAJO DE SUFICIENCIA PROFESIONAL**

Para optar el Título Profesional de

**INGENIERO DE SISTEMAS**

**PRESENTADO POR EL BACHILLER**

CAMPOS DIAZ, ALEJANDRO JUAN

Villa El Salvador
2017

**DEDICATORIA**

Eres una mujer que simplemente me hace llenar de orgullo, te amo y no va haber manera de devolverte tanto que me has ofrecido desde que incluso no hubiera nacido. Esta tesis es un logro más que llevo a cabo, y sin lugar a dudas ha sido en gran parte gracias a ti; no sé dónde me encontraría de no ser por tus ayudas, tu compañía, tu amor.

Te doy mis sinceras gracias amada madre.

Alejandro.

## AGRADECIMIENTO

A Dios, creador de la vida, por permitirnos estar con nuestros seres queridos, por acompañarnos en todo momento, y porque será siempre parte de todo lo que busco.

A mi alma mater, la Universidad Nacional Tecnológica de Lima Sur, por brindarme la oportunidad de lograr uno de mis mayores objetivos profesionales.

A mi asesor “Antonio Arqque Pantigozo”, por sus conocimientos, su orientación, su paciencia para guiarme durante el tiempo del desarrollo de este proyecto.

A mi familia, por estar siempre conmigo, apoyándome, motivándome a pesar de las dificultades.

Por todo ello muchas gracias.

Alejandro.

INDICE

## LISTADO DE FIGURAS

## LISTADO DE TABLAS

## GLOSARIO

- **Aplicativo,** programa informático diseñado para facilitar al usuario la realización de un determinado tipo de trabajo.
- **API**, siglas en ingles de *Application Programming Interfaces*, son las interfaces de programación de aplicaciones. Es una especificación formal sobre cómo un módulo de un software se comunica o interactúa con otro.
- **Archivo,** conjunto de información que se almacena en forma virtual en algún medio y pueden ser usados por las aplicaciones.
- **CAC**, siglas del término Centro de Atención al Cliente.
- **FIFO**, del inglés "*first in – first out*" significa "primero en entrar, primero en salir".
- **Incidencia**, interrupción no planificada de un Servicio de TI o una reducción de la Calidad de un Servicio de TI.
- **IT,** sigla en inglés que es lo mismo que TI en español: Tecnología de información, se trata de la tecnología necesaria a adquirir, procesar información por medios diferentes.
- **Log**, archivo donde se pintan los resultados de una ejecución.
- **Migracion**, proceso que incluye dos procesos a la vez, cambio de plan y cambio de tope de consumo.
- **Notepad++**, herramienta para visualizar las trazas de los servicios ejecutados.
- **Properties**, Archivo que contiene las propiedades de los objetos.

- **Requerimiento, s**on todos los requisitos pedidos por el cliente.
- **RPC**, siglas en inglés *Remote Procedure Call*, Llamada a Procedimiento Remoto Protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo es un gran avance sobre los sockets usados hasta el momento. De esta manera el 161 programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.
- **SMTP,** protocolo de red utilizado para el intercambio de mensajes.
- **SOA,** siglas en ingles del término Arquitectura Orientado a Servicios.
- **Servicio**, funcionalidad que realiza una tarea determinada.
- **Servidor**, lugar donde se hace el despliegue de los servicios y corren las ejecuciones.
- **Traza**, evidencia, huella de lo ejecutado y se muestra en el log.
- **Usuario**, es el individuo que utiliza una computadora, sistema operativo, servicio o cualquier sistema informático.
- **Validación,** acción que permite la comprobación la veracidad de algo.
- **XML**, Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

# INTRODUCCIÓN

El presente proyecto describe la problemática del proceso de Migración de líneas postpago, que a su vez consta de dos procesos internos, uno el Cambio de Plan y segundo el cambio de tope de consumo, todo ello para clientes móviles postpago afiliados a la empresa Claro, el cual brinda servicios de venta de paquetes de internet, minutos, etc.

Día a día los usuarios de telefonía móvil son cada vez mayores, a medida que las promociones ofrecidas por las empresas son más atractivas, ofreciendo ciertos beneficios según el tipo de plan.

La situación problemática radica en lo siguiente, cuando un cliente de una línea postpago solicita a Claro que le realice un Cambio de Plan, este no puede procesar su solicitud el mismo día, por lo tanto se le es programado para una posterior atención.

Actualmente este proceso está teniendo problemas de ejecución que implica realizar mal el proceso, genera descontento, incremento en el pago por parte de los usuarios y pérdidas económicas para la misma empresa.

Existen otros problemas que se dan durante el proceso de Migración, hay reprocesos de cambio de tope que no están siendo controlados llevando consigo a sobrecargas las peticiones, descontrol en los reportes de acción, carga innecesaria al servidor, etc. Estos problemas serán mencionados conforme se vaya desarrollando el proyecto. Motivo por la cual para dar solución a este problema, se está planteando implementar un conjunto de servicios tales como una SHELL, la

creación de un MDB (*Message Driven Bean*) para procesos masivos y validaciones, crear dos Colas (almacén de peticiones, solicitudes; *queue* en inglés) para realizar tanto el Cambio de Plan y el Cambio del Tope de Consumo el mismo día, asimismo realizar actualizaciones, validaciones necesarias para evitar los reprocesos y cargas en las solicitudes.

Para el proceso de Migración primero se realizará el Cambio de Plan e inmediatamente después el Cambio del Tope de Consumo.

El proyecto en cada capítulo toca un tema en específico relacionada a lo propuesta; en el capítulo 1, el planteamiento del problema describe la realidad problemática del proceso que se está desarrollando de manera deficiente. En el capítulo 2, se hace mención de los conceptos que se usaran en el proyecto como la metodología, la arquitectura, el Framework a usar, etc. En el capítulo 3 la implementación de propuesta, se hace una comparativa del antes y el después de la mejora del proceso.

Por otro lado el proyecto busca la mejor manera de solucionar el problema en mención, pero está abierto a las sugerencias que puedan surgir más adelante.

# CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA

## 1.1. DESCRIPCIÓN DE LA REALIDAD PROBLEMÁTICA

El rápido crecimiento de la telefonía móvil implica tener un mayor control en cuanto a los servicios prestados por las empresas de este mismo sector.

Claro Perú es una empresa de telefonía que ofrece servicios móviles los cuales como sabemos es uno de los principales procesos que actualmente vemos en nuestro día a día. Como toda empresa Claro trata de dar siempre el mejor servicio para los usuarios con la finalidad de satisfacer sus necesidades, pero durante la ejecución del proceso de migración están saliendo problemas que conlleva a molestias por parte de los usuarios y por lo tanto pérdidas para la misma empresa.

El problema es que cuando un cliente postpago afiliado a Claro ve la necesidad de hacer una migración (incrementar o reducir su plan de consumo), el usuario se acerca a un centro de atención al cliente (CAC), solicita su cambio de plan (el cambio de plan incluye una activación/

desactivación de tope de consumo con el que el usuario puede contar realizando un pago extra) y éste se le es programado para una fecha determinada, por lo general son los últimos días de facturación, es decir cuando está a punto de vencer el contrato.

Durante el proceso de cambio de plan el cliente desconoce que se está quitando o desactivando el Tope de Consumo que tenía anteriormente y una vez empiece el cambio de plan ya no cuenta con un tope que le informe que se haya consumido la totalidad del paquete contratado durante el día, lo cual lo lleva a consumir más de lo que el contrato había establecido, una vez llegada la fecha de facturación se da con la sorpresa de que el monto a pagar es mayor. El cambio de plan era programado para un día antes de la fecha de facturación a las 6 a.m. todos los días y la programación para la activación del tope consumo a las 2 a.m. del día siguiente, esto debido a que el proceso de cambio de plan tardaba bastante tiempo en procesar todas solicitudes ya que era un proceso masivo (tratar de realizar todos los cambios de planes en una sola ejecución) y es por ello que se programaba el tope de consumo para el día siguiente, el proceso de activación de tope depende del cambio de plan, es decir solo se ejecuta una vez se haya ejecutado correctamente el cambio de plan. Entonces como podemos ver, hay cerca de un día de holgura aproximadamente que el cliente tiene desactivado el tope de consumo.

Esto implica reclamos, molestias por parte de los clientes afiliados, éstos reclamos son generados por los cargos adicionales que se generan en

las facturas de los clientes por sus consumos del día que no tienen activados el Tope de Consumo. Cada reclamo implica un costo para Claro que para mantener la fidelidad del cliente genera Notas de Crédito para reducir los costos adicionales generados por no tener activados los Topes de Consumo.

Actualmente cuando se desactiva un Tope de Consumo y se envía a una cola (almacén de peticiones y solicitudes) de activación del MDB (*Message Driven Bean*), ésta lo procesa antes de que finalice el proceso de aprovisionamiento en Janus realizado por otro componente, ocasionando que finalice el proceso sin realizar la activación del Tope de Consumo.

Por otro lado también la activación del Tope de Consumo tarda o se realiza un día en ser atendido, una vez el proceso de Cambio de Plan se haya ejecutado con éxito, por lo que se genera todo el problema en mención.

Es preciso señalar que uno de los procesos internos de la migración es la activación y desactivación de los topes de consumos, para el flujo de desactivación no hay error ya que en un 99% se ejecuta sin problemas, el error está cuando es un proceso de activación, en la base de datos de EAI se registran todas las programaciones de activación o desactivación, al momento de activar un tope de consumo un servicio web revisa en la base de datos BSCS que el estado del tope esté desactivado, pero no, por alguna rezón el tope se encuentra activo, entonces no se puede activar un tope que ya está activo, se procede a desactivarlo (se actualiza un campo en BSCS y

se envía a una cola de errores), un *trigger* se ejecuta producto de la actualización haciendo un registro de la operación en la base de datos BSCS, existe otro servicio que vuelve a procesar los errores, pero valida su respuesta con la de otro servicio de Aprovisionamiento Janus (plataforma de servicios donde se realización las aprovisiones) el cual posee la actualización realizada por el *trigger*, Janus por su parte demora en hacer las validaciones cerca de tres minutos mientras el servicio lo hace en milisegundos y como éste termina primero no puede realizar la validación correctamente llevando al error.

En términos generales cuando la ejecución del proceso de migración tarda demasiado (más de cinco minutos) el servidor *WebLogic* entiende que ha ocurrido error en la ejecución y las vuelve a procesar con mismo *request* (solicitud, petición a ejecutar) del servicio encargado de realizar la activación y desactivación de los Topes de Consumo, éste, al no tener ninguna validación que restrinja que ya no se pueda ejecutar la desactivación/activación de un registro ya procesado, permite que se vuelva a desactivar/activar el tope llevando generar reprocesos y a saturar el servidor.

Este inconveniente se ha tenido por cerca de un año y medio, y para poner un alto a las pérdidas se encargó al área de SOAP, realizar las configuraciones y dar solución al procesos de migración que tenían errores de manera manual, es decir que si había un error en el proceso de migración, ellos realizaban la modificación en las tablas involucradas de la base de datos

manualmente, a pesar que ésta se debía dar de manera automática a través de los aplicativos y componentes que participan en el procesos de migración.

## 1.2. JUSTIFICACIÓN DEL PROYECTO

Según datos estadísticos en un día hay un promedio de 284 reclamos y al mes 8520 aproximadamente, quejas por parte de los clientes postpago que son producidos por cargos adicionales que se generan en sus facturas en consecuencia del consumo del día que no tienen activados el Tope de Consumo.

En este proyecto se pretende resolver el problema del proceso de migración de clientes móviles con línea postpago con el fin de mantener la fidelidad de los clientes. Se propone que el cambio de plan y los cambios de los topes de consumo se den automáticamente el mismo día, es decir realizar el cambio de plan e inmediatamente después el cambio del tope de consumo con el fin de no generar quejas por parte de los usuarios ya que cada reclamo implica un costo para Claro.

Se realiza el proyecto con la finalidad de que la empresa Claro evite seguir teniendo problemas en el proceso que conllevan al error, y por lo tanto a la pérdida económica. Ahora, si bien es cierto cuando el cliente realiza un reclamo y éste es justificado Claro como empresa debe asumir esto y devolver el extra generado en las facturas con la finalidad de mantener la fidelidad del cliente. Con el proyecto se busca eliminar estos reclamos o por lo menos reducirlos y de este modo satisfacer al cliente y evitar seguir

teniendo pérdidas. Por lo tanto se beneficiaría tanto la empresa misma, primero por no generar notas crédito y segundo por mantener la fidelidad del cliente, mientras que beneficiaría también al mismo cliente ya que éste estaría recibiendo un servicio de calidad y de acuerdo a lo firmado en el contrato.

Para lograrlo se necesitará implementar componentes desarrollados bajo el lenguaje de programación Java, configuraciones en el servidor, una arquitectura orientada a servicios (SOA) ya que los componentes involucrados en el proyecto no está desarrollado en un solo lenguaje y ésta arquitectura hace posible la integración entre éstos componentes, el marco de trabajo es Spring Framework porque permite acoplarse a cualquier aplicación sin la necesidad de modificar código para realizar algunas funcionalidades. Y bajo una metodología del modelo cascada.

## 1.3. DELIMITACIÓN DEL PROYECTO

### 1.3.1. Delimitación espacial

El desarrollo del proyecto se realizará en la empresa Everis Perú, en el área de integración (EAI), Everis es una consultora que ofrece servicios de *outsourcing* y es proveedor directo de la empresa América Móvil (Claro), ambos con sedes localizados en Lima – Perú, La Victoria.

La migración consta de varios procesos internos de los cuales para este proyecto se buscará mejorar el desarrollo del proceso de

cambio de plan y activación de tope de consumo para que se den el mismo día, evitar los reprocesos innecesarios y la mejora del proceso de Aprovisionamiento.

### 1.3.2. Delimitación social

El proyecto impacta en todos los clientes postpago afiliados a Claro a nivel nacional.

### 1.3.3. Delimitación temporal

El proyecto tendrá una duración de tres meses y medio, comenzará el 19 de diciembre del 2016 y finalizará el 28 de marzo del 2017.

### 1.3.4. Delimitación conceptual

En este proyecto vamos a tratar exclusivamente el proceso de Migración que a su vez consta de dos procesos internos, primero el cambio de plan es un proceso en el que el cliente ve la necesidad de cambiar o modificar su plan actual por otro que encaje mejor según él lo disponga que puede ser mayor o menor, segundo la activación o desactivación de topes de consumo, que implica poner un límite según el monto contratado para que el usuario no exceda en el consumo, si un clientes no tiene un tope de consumo hay la posibilidad de que

sobrepase la cantidad establecida en su contrato y por lo tanto el monto a pagar al fin de mes sería mayor.

El proyecto está dirigido hacia la empresa Claro con la finalidad de solucionar el problema que actualmente tiene en el proceso de migración, se usará una arquitectura orientada a servicios (SOA) que usa la entidad actualmente, con la finalidad de la fácil integración de los componentes a desarrollar, cabe mencionar que la solución es solo para clientes postpago.

## 1.4. FORMULACIÓN DEL PROBLEMA

### 1.4.1. Problema General

¿De qué manera la implementación de una propuesta basada en una arquitectura orientada a servicios permitirá mejorar el proceso de migración de líneas postpago a nivel nacional en la empresa Claro Perú?

### 1.4.2. Problemas Específicos

a. ¿Cuál es el flujo actual del proceso de Migración?
b. ¿Cuál ha sido el impacto del inconveniente en el proceso de migración durante el tiempo que se ha estado ejecutando?
c. Actualmente, ¿Qué componentes se están ejecutando de forma deficiente en el proceso?
d. ¿Cuál sería el nuevo flujo del proceso de Migración con los cambios propuestos?

e. ¿Cuál sería el resultado de la mejora del proceso de Migración una vez implementados los cambios?

## 1.5. OBJETIVOS

### 1.5.1. Objetivo General

Implementar una propuesta basada en una arquitectura orientada a servicios para la mejora del proceso de migración (Cambio de plan y activación/desactivación de topes de consumo) de líneas postpago a nivel nacional en la empresa Claro Perú.

### 1.5.2. Objetivos Específicos

a. Describir el flujo actual del proceso Migración de líneas postpago afiliado a Claro.
b. Identificar las pérdidas (económicas y desafiliación de clientes) generadas durante el tiempo de ejecución con inconvenientes.
c. Identificar los componentes que estén haciendo mal algunas validaciones y realizando reprocesos.
d. Modelar, implementar e integrar los componentes requeridos para el nuevo flujo de migración.
e. Optimizar del proceso de migración de líneas postpago para controlar en lo posible las pérdidas económicas y desafiliaciones que se dan producto de los reclamos generados por los clientes.

# CAPÍTULO II: MARCO TEÓRICO

## 2.1. ANTECEDENTES DE LA INVESTIGACIÓN

### 2.1.1. Antecedentes Internacionales

En cuanto a trabajos, proyectos o tesis relacionadas al tema a nivel internacional, existen estudios en el que parcialmente se asemejan a nuestra propuesta:

En México, Prieto (2015) de la Universidad Tecnológica de la Mixteca en su proyecto de tesis titulado "Adaptación de las Metodologías Tradicionales Cascada y Espiral para la Inclusión de Evaluación Inicial de Usabilidad en el Desarrollo de Productos de Software en México", explica por qué es necesario desarrollar un producto (software) haciendo uso de una metodología, menciona que muchas veces se ha desarrollado software sin ninguna metodología y al final el usuario terminaba rechazando el producto por no cumplir con lo que había solicitado en los requerimientos. El autor realizó varias pruebas en diferentes empresas de tecnologías de información y se encontró que el uso de una metodología como apoyo, disminuye el

índice de errores. Por ello el autor recalca la importancia del uso de la metodología para lograr una mejor calidad, eficiencia y eficacia en el desarrollo del software en las empresas. Y la metodología Cascada que forma parte de las metodologías tradicionales es muchas veces la más conveniente según el tipo de negocio, por las etapas que sigue en el desarrollo del software. Para el proyecto este trabajo servirá de modelo y guía para seguir los pasos o etapas que sigue la metodología, ver qué actividades y cómo es que éstas se realizan para lograr el resultado esperado.

Otro caso en Estado Hidalgo, Mendoza (2012) ambos de la Universidad Autónoma del Estado Hidalgo en su proyecto de tesis "Sistema de control, secuencial, y término de los ingresados en centros de readaptación social del Estado de Hidalgo" utilizó la metodología cascada para el desarrollo de un sistema y dar solución a problemas penitenciarios, como tener conocimiento de personas que pueden tener el beneficio de libertad y pueda ser analizado jurídicamente cada caso, ya que en la penitenciaria de Hidalgo hay 21000 internos y no hay capacidad para más, sumado a esto no tienen un sistema de información que les permita tener un control eficiente de los procesos ejecutivos y administrativos. El autor propuso desarrollar un sistema que permita automatizar las operaciones de los internos, y procesos administrativos, todo automatizado y para ello optaron por usar la metodología cascada para el desarrollo del

sistema. Este proyecto sirvió para conocer detalles de la metodología, cómo, porqué y cuando es recomendable trabajar bajo este modelo.

En Chile, Salinas (2011) presentó una arquitectura de software que tiene como objetivo primordial, el satisfacer los requisitos de automatización y extensibilidad que una herramienta de software de apoyo a la implementación del Proceso Personal de Software debe poseer. Para cumplir con estas exigencias se optó por utilizar la denominada Arquitectura Orientada a Servicios, ya que ésta entrega las facilidades que permiten poder lograr interoperabilidad entre distintos Entornos de Desarrollo Integrado con un servidor central que administre y almacene la información.

El trabajo pretendía, por tanto, utilizar los beneficios que entrega la denominada Arquitectura Orientada a Servicios (SOA, *Service Oriented Architecture*) la que permite a distintos sistemas, escritos en diferentes lenguajes de programación y de distintas plataformas tecnológicas, transformarse en servicios débilmente acoplados y altamente interoperables.

La arquitectura que fue definida por el autor es la que se muestra en la figura y estaba dividida en tres capas: Capa Cliente, Servidor (Java EE) y la capa de almacenamiento.

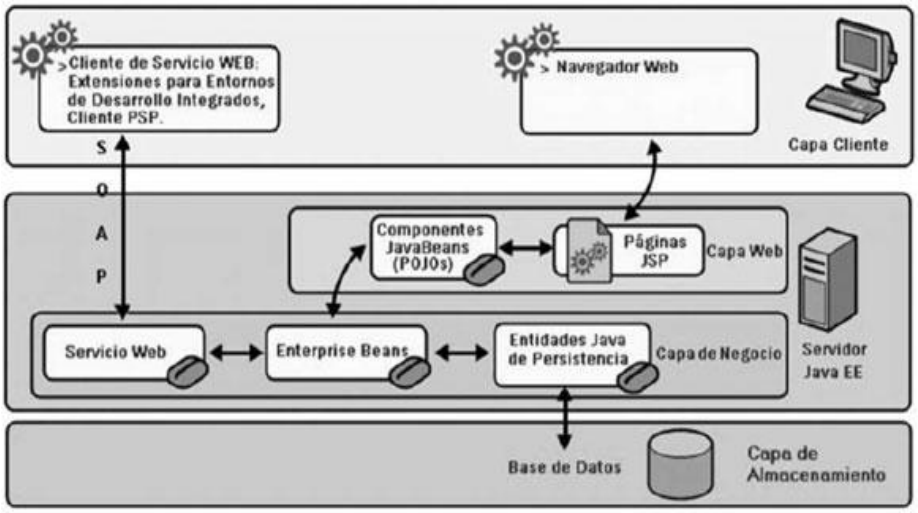



**Figura 1. Arquitectura Integral del Sistema**
Fuente: (Salinas, 2011)

Este proyecto tiene un aporte valido para la propuesta en mención, en el sentido que describe de qué manera una arquitectura de software orientada a servicios debe implementarse, permitir la comunicación con otros componentes sin importar que el lenguaje de programación de cada uno sea igual o diferente, el propósito es integrar los componentes que se incluirán en la propuesta con los componentes ya existentes, y realizar el proceso de migración de usuarios con líneas postpago de manera exitosa. Si bien en la empresa Claro se usa esa arquitectura (SOA) es necesario conocer la forma de su implementación y funcionamiento para el proyecto propuesto.

### 2.1.2. Antecedentes Nacionales

En cuanto a tesis o proyectos de investigación relacionadas al tema tratado a nivel nacional, podemos hacer mención del proyecto de investigación presentado por Santos (2012) de la Universidad Nacional Mayor de San Marcos, el proyecto enfocado a Banca describe uno de los procesos importantes para el pago de obligaciones y servicios públicos, es la “acreditación de pago” que se produce cuando una entidad bancaria envía un archivo plano con los datos realizados por las web y a través de ventanillas para que la entidad pública los registre como válidos. El autor propuso una implementación de un planificador de tareas y un aplicativo de registro de pagos desarrollados bajo el lenguaje Java y un marco de trabajo Spring Framework y otros componentes como JMS (Java *Message Service*) y un *Message Broker* (corredor de mensajes). Por lo tanto éste proyecto servirá de apoyo para ver de qué manera se hace la implementación de un MDB, y todo aquello que tenga que ver con el manejo de mensajes.

Pinedo (Pinedo et al, 2010) presentaron un proyecto que lleva por nombre “Implementación de un Sistema de Integración para las bibliotecas municipales de Lima y Callao utilizando SOA y J2ME”, el cual se basa en la arquitectura orientada a servicios (SOA). El proyecto propuso un enfoque de solución basado en una arquitectura orientada a servicios e integración de datos para las bibliotecas

municipales de Lima y Callao. En ese entonces las bibliotecas municipales de Lima y Callao no contaban con *web services*, al contrario todas ellas contaban con aplicaciones que solo proporcionan información de la misma biblioteca, lo cual no resultaba incómodo para los usuarios siendo necesario consultar en diferentes bibliotecas. El aporte para el proyecto es ver la forma de cómo ellos implementan SOA, que proceso siguieron y hacer una comparativa con el modelo de implementación de Claro y definir el mejor modelo a usarla.

Santos (2012) de la Universidad Nacional Mayor de San Marcos presento un proyecto para el diseño de un módulo de carga de pagos en entidades públicas mediante mensajería usando Spring Framework, el proyecto soluciona la problemática del proceso de acreditación de pago, el cual implica el envío de un archivo plano con todas las transacciones realizadas durante un día, tanto por medio del pago en línea por internet como también por el pago por ventanilla bancaria. Todo lo mencionado implica una carga masiva de la data hacia los sistemas gubernamentales. Y como solución propuso la implementación de un aplicativo automatizado de carga, el cual implica una lectura automática del archivo del día y su registro en la base de datos. Este proyecto tiene un gran aporte para la propuesta en mención ya que expone elementos similares (JMS (Java Message Service) y un corredor de mensajes (Message Broker)) al de la

propuesta y serviría de guía para poder implementarlas de forma correcta.

## 2.2. BASES TEÓRICAS

### 2.2.1. Metodología de desarrollo de software

Todo desarrollo de software es riesgoso y difícil de controlar, pero si se emplea una metodología se tiene una certeza de que se debe seguir procedimiento adecuado para para su implementación. Hay muchas metodologías para el desarrollo de software y cada autor define según sus perspectivas qué metodología es mejor que otra. Por otro lado otros autores señalan que la metodología a usar depende del tipo de proyecto que se quiera realizar.

Para desarrollar un producto software se tiene que definir un enfoque disciplinario y sistemático. Las metodologías de desarrollo influyen directamente en el proceso de construcción y se elaboran a partir del marco definido por uno o más ciclos de vida. (Piattini, 1996).

Piattini (1996) afirma que “no hay un consenso entre los autores sobre el concepto de metodología, y por lo tanto no existe una definición universalmente aceptada”. “Pero sí hay un acuerdo en considerar a la metodología como “un conjunto de pasos y procedimientos que deben seguirse para el desarrollo del software”.

Por ello es muy importante definir el tipo de proyecto y la metodología que más se adapte a su implementación para llegar cumplir a los objetivos.

#### 2.2.1.1. Ciclo de vida

Todo tipo de proyecto sin importar el tamaño, su complejidad, su desarrollo tienen pasos que deben seguir tales como: planificación, estimación de recursos, seguimiento, control y evaluación del proyecto (generalmente). Por lo tanto la selección de un ciclo de vida al orden de las actividades a desarrollar.

#### 2.2.1.2. Metodología Cascada

Según Martínez y González (2015) de acuerdo a los requerimientos, el equipo y las necesidades que el proyecto demanda, se opta por la utilización de una metodología tradicional para el desarrollo de los servicios, metodología cascada.

La metodología cascada divide las actividades en fases, es decir que para que una actividad inicie es imprescindible que la actividad anterior haya finalizado.

Igualmente Martínez (2015) especifica que “la principal estrategia es definir y seguir el progreso del desarrollo de software hacia puntos de revisión bien

definidos, es decir, se codifica y reparan los errores; es un proceso continuo de codificación y reparación".

Tomando como ejemplo el modelo seguido por el autor, para el proyecto se optó por seguir el modelo cascada, ya que es de fácil uso cuando se tienen los requerimientos claros, y por otro lado es el modelo que se ha venido usando por Claro durante algún tiempo.

La metodología cascada está compuesta por un ciclo de vida que abarca las actividades mostradas en la figura 2.

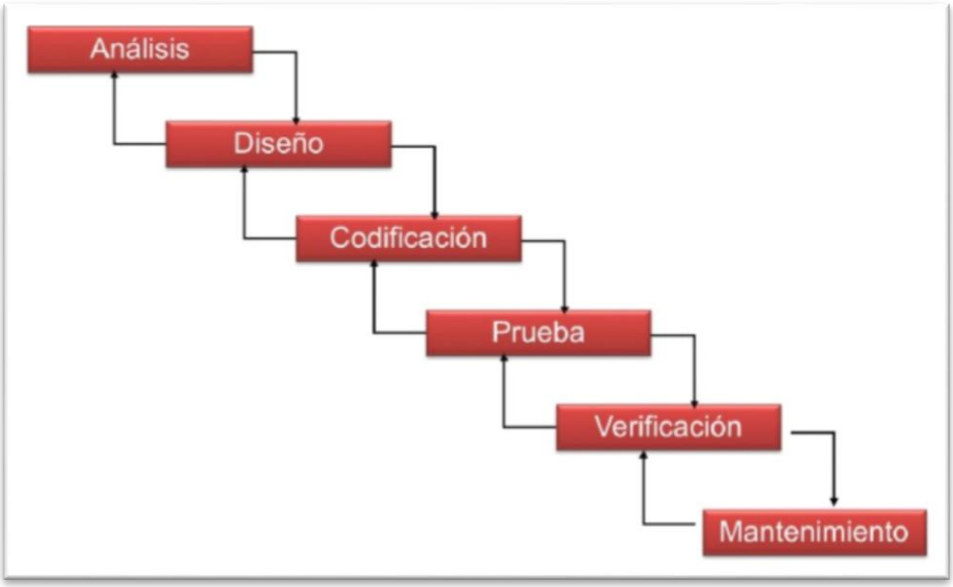



**Figura 2. Actividades Metodología Cascada**

Fuentes: (Martínez y González, 2015)

Este modelo es muy útil pues ayuda a los desarrolladores a comprender qué es lo que tienen que hacer en cada momento.

Su simplicidad hace que resulte sencillo explicárselo a los clientes que no están familiarizados con proceso del software. Además se muestran de forma explícita qué productos intermedios se deben obtener antes de abordar las siguientes tareas.

La Figura 3 muestra las diferentes fases que sigue el modelo Cascada durante el ciclo de vida del software.

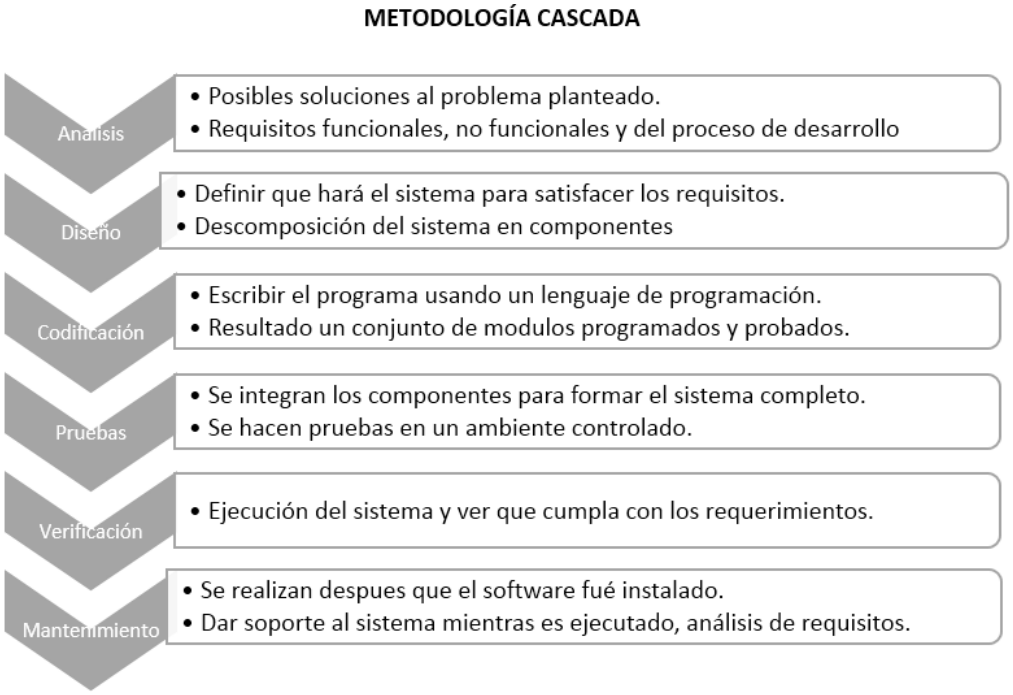



***Figura 3. Descripción del Modelo Cascada***

Fuentes: Elaboración propia

2.2.1.2.1. Implementación / Etapas

El método Cascada es considerado como el enfoque clásico para el ciclo de vida de desarrollo de software, se podría decir que es un método puro que implica un desarrollo rígido. Este modelo es caracterizado por ordenar de manera rigurosa las etapas del ciclo de vida de software, dado que el comienzo de cada etapa debe esperar a la finalización de la inmediata anterior.

Las fases desarrolladas para la solución del problema se estructuraron de la siguiente manera:

a. Análisis

El análisis, la definición de requisitos o especificaciones de características que ha de cumplir el software que vamos a desarrollar es la primera etapa del modelo cascada. Y probablemente sea la más importante. Al fin y al cabo, lo que sea o no el producto final depende de decisiones tomadas en esta etapa. Se trata fundamentalmente de estudiar las necesidades y preferencias del usuario. Es importante también dejar clara constancia de las decisiones tomadas en esta etapa, para ser tenidos en cuenta posteriormente. Por ello la documentación (contiene las necesidades del

usuario) producida en esta fase debe ser concreta y estar siempre disponible durante el resto del proceso.

Es la fase donde se inicia con la descripción del problema y el levantamiento de información.

b. Diseño

En esta fase se realiza un segmentado del sistema por módulos, es decir se descompone y organiza el sistema en partes para que puedan desarrollarse por separado. Como resultado surge el documento de Diseño del Software que contiene la descripción de la estructura relacional del sistema y de las especificaciones de lo que debe hacer cada una de sus partes, y cómo interactúan una con otra. Básicamente se enfoca en atributos distintos del software; la arquitectura lógica, estructura de datos, el detalle procedimental y la caracterización de la interfaz.

Se establecen los requerimientos hardware y/o software del sistema global, herramientas que se desarrollaran para el desarrollo de la codificación.

Se deben describir a detalle los componentes principales del software y las relaciones entre ellos, y

debe quedar plasmado en un documento del diseño del software.

c. Codificación

Es la fase de programación e implementación, en esta fase se implementa el código fuente según las especificaciones, haciendo uso de prototipos así como pruebas y ensayos para corregir errores.

Las pruebas consisten en verificar que cada componente cumpla con su especificación (Desarrollo en Cascada, s.f).

d. Pruebas

Los elementos que ya han sido programados se ensamblan, integran para componer el sistema, una vez completado la integración, se comprueba que se ejecute correctamente y que cumpla con los requisitos definidos previamente, antes de ser entregado al usuario final.

e. Verificación

Es la fase donde el usuario final ejecuta el sistema, para ello ya se han realizado las pruebas previamente

para comprobar que el sistema no tenga fallas (Desarrollo en Cascada, s.f).

f. Mantenimiento

Es una de las etapas más críticas ya que se destina el 75% de los recursos, en el mantenimiento del software ya que al utilizarlo como usuario final puede que no cumpla con todas nuestras expectativas (Desarrollo en Cascada, s.f).

Este alto porcentaje indica que el software sufrirá cambios después de su entrega al cliente. Los cambios ocurrirán debido a que se haya encontrado errores, o escenarios que no se planificaron previamente, a que el software deba adaptarse a cambios del entorno externo (sistemas operativos o dispositivos periféricos) o a que el cliente requiera ampliaciones funcionales o del requerimiento.

2.2.1.2.2. Características

Ledezma (2017) expone las siguientes características para el modelo cascada:

- Es la metodología más utilizada para el desarrollo de software.

- Es una visión del proceso de desarrollo de software como una sucesión de etapas que produce productos intermedios.
- Si se cambia el orden de las fases, el producto final será de inferior calidad.

2.2.1.2.3. Ventajas

El modelo Cascada al igual que cualquier metodología de desarrollo de software tiene ventajas y desventajas, en algunos puntos una es mejor que otra, pero cada una tiene una forma específica de desarrollo que la hace diferente a las demás.

Desarrollo en Cascada (s.f), ventajas del modelo:

- Realiza un buen funcionamiento en equipos débiles y productos maduros, por lo que se requiere de menor capital y herramientas para hacerlo funcionar de forma óptima.
- Es un modelo fácil de implementar y entender.
- Está orientado a documentos.
- Es un modelo conocido y utilizado con frecuencia.
- Promueve la metodología de trabajo efectivo: definir antes que diseñar, diseñar antes que codificar.

- Debido a que el proceso ya está planeado es más fácil estimar costos y plazos.
- Permite la departamentalización y control de la gestión.

#### 2.2.1.3. Justificación del Modelo en Cascada

Para el proyecto en mención se plantea usar el modelo cascada ya que es una metodología sencilla y adecuada cuando los requerimientos están claros y no hay posibilidad de retrasar las etapas que siguen.

### 2.2.2. Servicio Web

#### 2.2.2.1. Definición

Pinedo y Medina (2010) afirma que “un servicio web (en inglés, *Web Service*) es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de computadoras como Internet” (p.25)

2.2.2.2. Estándares empleados

Pinedo y Medina (2010) definen los siguientes estándares empleados para un servicio web:

- *Web Services Protocol Stack*: es definido como un conjunto de servicio y protocolos para los servicios web. Básicamente son utilizados para definir, localizar, implementar y hacer que los servicios web interactúen con otros.

  Este conjunto está conformado a su vez por cuatro subconjuntos:

a) Servicio de transporte

Encargado de transportar los mensajes entre las diferentes aplicaciones. Entre los más utilizados están HTTP, FTP, SMTP y JMS.

b) Mensajería XML

Es el encargado de la codificación de los mensajes en XML estándar y de éste manera

pueda ser interpretado en los distintos nodos de la red.

c) Descripción del servicio

Todo servicio web tiene que tenar una interfaz pública la cual esta descrita por un formato llamado WSDL (*Web Services Description Language*).

d) Descubrimiento de servicios

UDDI (*Universal Description Discovery and Integration*): UDDI es un marco independiente de la plataforma para describir servicios, negocios e integrar servicios de negocios. UDDI permite a los clientes buscar tal registro, encontrar el servicio deseado y extraer sus detalles.

- XML (*Extensible Markup Language*): Es el formato estándar para los datos que se vayan a intercambiar.
- SOAP (*Simple Object Access Protocol*) o XML-RPC (*XML Remote Procedure Call*): Protocolos sobre los que se establece el intercambio.

- Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP (*Hypertext Transfer Protocol*), FTP (*File Transfer Protocol*), o SMTP (*Simple Mail Transfer Protocol*).
- WSDL (*Web Services Description Language*): Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.

Quesada (2007) WSDL es un lenguaje de descripción de servicios Web que permite definir la funcionalidad abstracta y la forma de acceder a un servicio. Ya que los protocolos de comunicación y los formatos de los mensajes se están estandarizando en la comunidad web, se va haciendo cada vez más importante la descripción de las comunicaciones de forma estructurada. WDSL satisface esa necesidad al definir una gramática XML para describir servicios de red como colecciones de agentes de comunicación capaces de intercambiar mensajes.

#### 2.2.2.3. Ventajas de los servicios web

Las ventajas de los servicios web según Pinedo y Medina (2010) son los mencionados a continuación:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos. Las especificaciones son gestionadas por una organización abierta, la W3C, por tanto no hay secretismos por intereses particulares de fabricantes concretos y se garantiza la plena interoperabilidad entre aplicaciones.

### 2.2.3. Servicio de Mensajes con JMS

Tarde o temprano llegará el momento donde se tendrá la necesidad de intercomunicar aplicaciones. Para ello existen muchas soluciones pero dependiendo del tipo de problema y las necesidades es que se implementan las que mejor se adapten. Con ello llega el mundo de la mensajería donde se envían, reciben mensajes y cada mensaje contiene una solicitud de tarea determinada para ser atendida por las aplicaciones.

#### 2.2.3.1. Java Message Service (JMS)

Según el Departamento de Ciencia de la Computación e Inteligencia Artificial (2014) un JMS ofrece un API estándar (mediante conjunto de interfaces) para la mensajería empresarial, de modo que a través de Java se pueda enviar y recibir mensajes sin depender de ningún proveedor. Un JMS permite también la comunicación entre componentes ya sea débilmente acoplada, asíncrona (el proveedor JMS entrega los mensajes al destino conforme llegan y el cliente no tiene que solicitar dichos mensajes para poder recibirlos) y fiable (JMS asegura que cada mensaje se entrega una y solo una vez, y mediante inferiores niveles de fiabilidad permite la pérdida o el

duplicado de mensajes en aquellas aplicaciones que requieran menos control).

#### 2.2.3.2. Arquitectura JMS

JMS define varios elementos que forman parte de la mensajería:

a) Cliente JMS

Un cliente JMS es una aplicación en Java que envía y recibe mensajes.

b) Proveedor JMS

Escrito en Java y es el encargado de ofrecer prestaciones de administración como de control de los recursos JMS (toda implementación de la plataforma Java incluye un proveedor JMS).

c) Mensaje JMS

Es el elemento principal de un JMS y consta de tres partes cabecera, propiedades y cuerpo que contiene información que a su vez es enviada o recibida por un cliente JMS.

d) Dominio JMS

Está conformada por dos estilos de mensajes PTP y Pub/Sub.

e) Objetos administrados:

Los objetos JMS contienen los datos de configuración específicos del proveedor, usados por los clientes.

Los clientes pueden acceder e estos objetos a través de un JNDI.

En la figura 4 se aprecia el modo en el que interactúan dichos elementos.

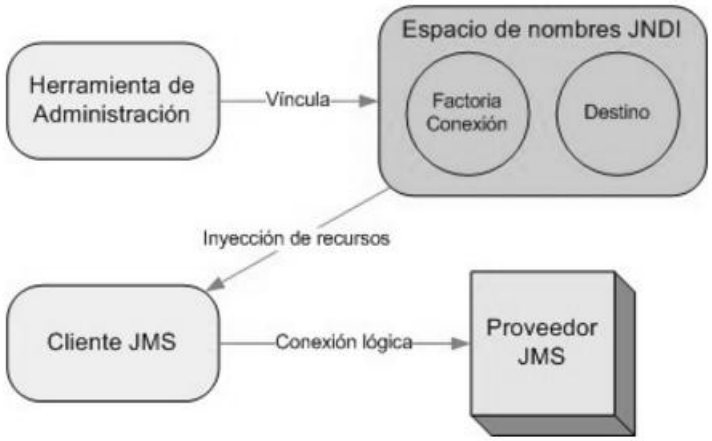



***Figura 4. Interacción de Elementos de Mensajería***

Fuente: (Dpto. de Ciencia de la Computación e Inteligencia Artificial, 2014)

Las herramientas de administración permiten vincular destinos y factorías de conexión a través de un espacio de nombres JNDI.

2.2.3.3. Mensajes

Es el elemento más importante de las especificaciones JMS. Todos los elementos de las especificaciones están relacionados con mensajes ya que éste es el medio por el cual los datos y eventos del negocio transmiten a través de un

proveedor JMS. (Departamento de Ciencia de la Computación e Inteligencia Artificial, 2014)

En la figura 5 se observa las partes por las que está compuesto un mensaje.


***Figura 5. Partes de un Mensaje***

Fuente: (Departamento de Ciencia de la Computación e Inteligencia Artificial, 2014)

Detalle de cada elemento del mensaje:

- Cabecera: todos los mensajes JMS soportan el mismo estándar de mensajes. Muchas de las cabeceras son asignadas automáticamente por el cliente o servidor.
- Propiedades: son un tipo de cabeceras adicionales pero opcionales a diferencia de las cabeceras que son obligatorias.

- Cuerpo: viene a ser el cuerpo del mensaje, conforma la recepción y envío de información en diferentes formatos.

#### 2.2.3.4. Message Driven Bean (MDB)

Según el Departamento de Ciencia de la Computación e Inteligencia Artificial (2014) un MDB es un oyente de mensajes que puede consumir mensajes de una cola. Estos mensajes pueden enviarse desde cualquier elemento JavaEE.

Conceptualmente se diseñaron para que el servidor de aplicaciones proporcione facilidades de *multi-threading*, esto es que múltiples consumidores procesen mensajes concurrentemente sin necesidad de desarrollar código adicional. Así los *MDB's* proporcionan dicha facilidad al manejar los mensajes entrantes mediante múltiples instancias de *beans* alojados en el pool del servidor de aplicaciones.

**Multihilo**

El Departamento de Ciencia de la Computación e Inteligencia Artificial (2014) afirma que las aplicaciones de negocio van a necesitar consumidores de mensajes multihilo que puedan procesar los mensajes de modo concurrente. Los MDB's evitan esta complejidad ya que soporta el multihilo sin necesidad de código adicional.

Tal y como lo muestra la figura 6 los MDB's gestionan los mensajes entrantes mediante múltiples instancias de beans (dentro de un pool), y tan pronto como un nuevo mensaje llega al destino, una instancia MDB sale del pool para manejar el mensaje.

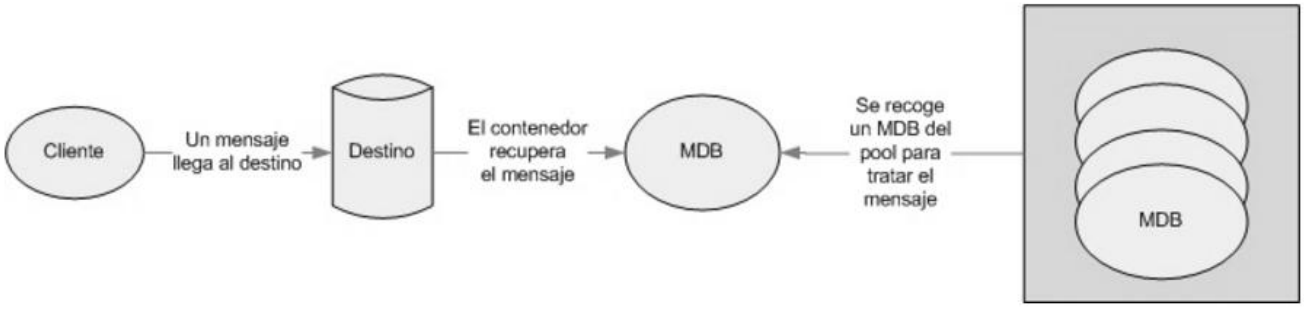



***Figura 6. Gestión de Mensajes entrantes***

Fuente: (Departamento de Ciencia de la Computación e Inteligencia Artificial, 2014)

### 2.2.4. Arquitectura Orientada a Servicios (SOA)

Marsili (2010) explica los alcances de ésta tendencia cuya adopción crece en las áreas de tecnología de la información de las grandes empresas.

La urgencia de datos, la exactitud y la seguridad a partir de un final de un proceso de negocio al otro son ahora un mandato de negocio. Las

organizaciones que pueden hacer esto tienen una distintiva ventaja competitiva.

Según López (2009) SOA es una arquitectura de software que permite la creación o cambios de los procesos de negocio desde la perspectiva de TI de forma ágil.

El autor define par SOA las siguientes capas de software:

- Aplicaciones básicas, sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad.
- De exposición de funcionalidades, las funcionalidades de la capa de aplicaciones son expuestas de forma de servicios.
- De integración de servicios, facilitan el intercambio de datos entre los elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración.
- De composición de procesos, que define e proceso en términos del negocio y sus necesidades y que varía en función del negocio.
- De entrega, los servicios son desplegados a los usuarios finales.

#### 2.2.4.1. ¿Qué es SOA?

Microsoft (2006) la arquitectura SOA establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda acceder a sus funcionalidades, las cuales se ofrecen como servicios. La forma más habitual de implementarla es mediante Servicios Web, una tecnología basada en estándares e independiente de la plataforma, con la que SOA puede descomponer aplicaciones monolíticas en un conjunto de servicio e implementar esta funcionalidad en forma modular. Si bien es cierto actualmente usamos SOA para el desarrollo de servicios, impulsamos su implementación porque ayuda a las aplicaciones en su performance ya que aporta flexibilidad, facilidad de manejo de datos.

¿Qué es un servicio exactamente?

IDS2015 (2015) define a servicio como una funcionalidad concreta que puede ser descubierta en la red y que describe tanto lo que puede hacer como el modo de interactuar con ella. Desde la perspectiva de la empresa, un servicio realiza una tarea concreta: puede corresponder a un proceso de negocio tan sencillo como introducir o extraer un dato como “Código de

Cliente". Pero también los servicios pueden acoplarse dentro de una aplicación completa que proporcione servicios de alto nivel, con un grado de complejidad muy superior, por ejemplo, "introducir datos de pedido", un proceso que, desde que comienza hasta que termina, puede involucrar varias aplicaciones de negocio.

La estrategia de orientación a servicios permite la creación de servicios y aplicaciones compuestas que pueden existir con independencia de las tecnologías subyacentes. En lugar de exigir que todos los datos y lógica de negocio residan de un mismo ordenador, el modelo de servicios facilita el acceso y consumo de recursos de tecnologías de información a través de la red.

En la figura 7 se puede apreciar cómo y que procedimiento realiza un servicio cuando es consumido por algun cliente.

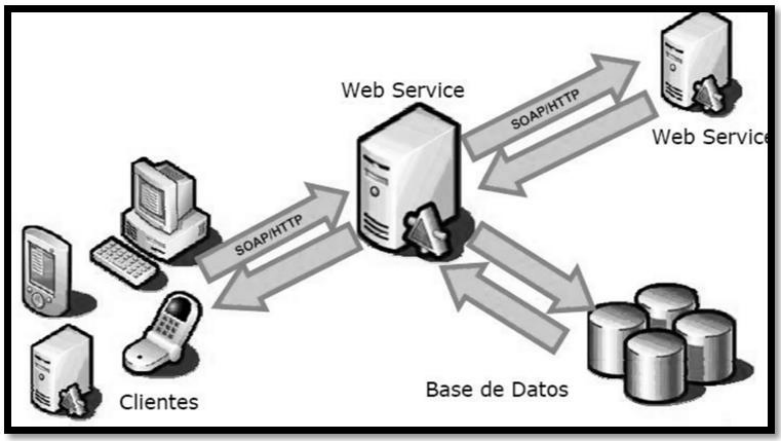



***Figura 7.Funcionamiento de un Servicio***

Fuentes: (Falagan, 2011)

2.2.4.2. Beneficios de SOA

Microsoft (2006) indica que los beneficios de SOA en las organizaciones se modelan a dos distintos niveles, la primera a nivel de usuario corporativo y el segundo a nivel de organización de IT.

Arquitectura Orientada a Servicios (2016) permite el desarrollo de una nueva generación de aplicaciones dinámicas que resuelven una gran cantidad de problemas de alto nivel, fundamentales para el crecimiento y la competitividad. Según la fuente citada las soluciones SOA permiten entre otras cosas:

- Mejora en los tiempos de realización de cambios de procesos.
- Facilidad para evolucionar a modelos de negocio en tercerización.

- Permite poder reemplazar elementos de la capa aplicativa SOA sin disrupción en el proceso de negocio.
- Facilidad para la integración de tecnologías disímiles.

Mientras el autor Vera (2014), expone los beneficios que otorga SOA de la siguiente manera:

- Mejorar la toma de decisiones:

  Al integrar el acceso a los servicios e información de negocio dentro de un conjunto de aplicaciones dinámicas compuestas, los directivos disponen de más información y de mejor calidad (más exacta y actualizada).

- Mejorar la productividad de los empleados:

  Un acceso óptimo a los sistemas, información y la posibilidad de mejorar los procesos permiten a las empresas la productividad individual de los empleados.

- Aplicaciones flexibles:

  La orientación dispone de mayor información y más actualizada, lo que le permite una respuesta rápida y eficaz cuando surgen problemas o cambios.

- Aplicaciones reutilizables y adaptables:

  Permite que las aplicaciones existentes puedan ser reutilizadas y adaptadas a nuevos entornos con

facilidad, así conseguimos optimizar los recursos utilizados en su desarrollo.

- Reducción de costos:

  El coste de ampliar o crear nuevos servicios se reduce considerablemente tanto en aplicaciones nuevas como ya existentes.

- Riesgo de migración:

  Al adaptar SOA a partir de una tecnología existente se sigue utilizando los componentes existentes, por lo que reduce el riesgo de introducir riesgos.

La arquitectura orientada a servicios (SOA) no se trata de software o de un lenguaje de programación, SOA es un marco de trabajo conceptual que permite a las organizaciones unir los objetivos de negocio con la infraestructura de TI integrando los datos y la lógica de negocio de sus sistemas separados.

Si se quiere definir qué es una arquitectura, de acuerdo con el estándar ANSI/IEEE Std 1471-2000, se tiene que es “la organización fundamental de un sistema, compuesta por sus componentes, las relaciones entre ellos, su ambiente y los principios que gobiernan su diseño y evolución”.

Por otra parte una arquitectura está compuesta por:

- Arquitectura de Procesos de Negocio.
- Arquitectura de Interfaces/Integración.
- Arquitectura Tecnológica.

Por lo tanto es importante conocer la necesidad de definir los diferentes elementos que conforman la arquitectura y diferenciar su alcance dentro del desarrollo. En la figura 8 se observa la interacción entre los compontes que intervienen en la arquitectura SOA.


***Figura 8. Modelo de Arquitectura Orientado a Servicios***

Fuente: (Herrera, et al, 2012)

2.2.4.3. Componentes del modelo conceptual de SOA

Herrera, Guillé y Javier (2012) en su proyecto “Estudio Comparativo de Servidores de Aplicaciones para Desarrollo de Software con SOA sobre Plataformas JavaEE. Caso Práctico:

Transportes Patria" describen a SOA en base a la estructura definida en la figura 9:

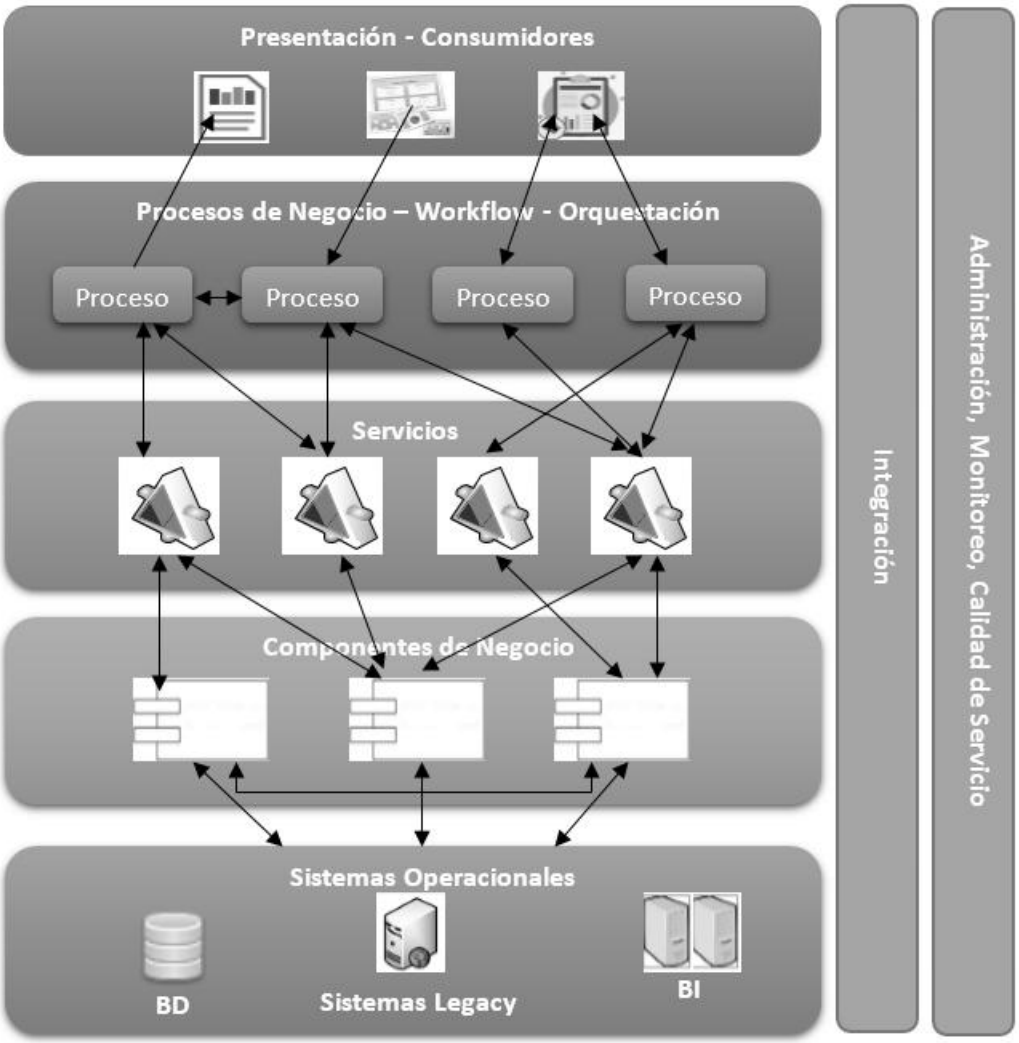



***Figura 9. Capas de una Arquitectura Orientada a Servicios (SOA)***

Fuente: (Herrera, Guillén y Javier, 2012)

a. Sistemas Operacionales:

Vienen a ser las aplicaciones existentes en la empresa, dentro de ésta capa podemos tener CRM's, ERP's, aplicaciones de BI, etc. Todas estas aplicaciones son integradas gracias a SOA.

b. Capa de Componentes:

Esta capa contiene los componentes que se encargan de brindar la funcionalidad que exponen los servicios. En ésta capa es posible usar otras tecnologías para ayudar a desarrollar las tareas de implementación, etc.

c. Capa de Servicios:

Contiene los servicios que la organización decide exponer, pueden ser descubiertos, referenciados directamente, o ser parte de una orquestación o de un servicio nuevo.

d. Capa de Procesos de Negocio

Esta capa contiene la orquestación de los servicios, los servicios están ligados a estos *workflows* por lo que actúan como una sola aplicación.

e. Capa de Presentación:

Normalmente ésta capa no forma parte de SOA, pero cada día se hace más relevante.

f. Integración (ESB – Enterprice Service Bus)

Esta capa facilita la integración de los servicios a través de la introducción de un conjunto de capacidades tales como ruteo, mediación de protocolos, mecanismo de transformación, etc.

g. Capa de Administración, Monitoreo y Calidad del Servicio:

Indica las características requeridas para monitorear, administrar y mantener la calidad del servicio en las distintas áreas.

El modelo o arquitectura SOA mostrada por el autor sirve para el propuesta porque permite identificar los distintos niveles que hay y qué procedimientos o actividades se realizan en cada una.

2.2.4.4. Estrategias para la implantación de SOA

Según la Arquitectura de Referencia de Claro (2016) para poder adoptar una arquitectura basada en servicios, aunque parezca obvio, lo primero que se debe hacer es determinar los sistemas que queremos exponer. Esto puede parecer sencillo, sin embargo, no lo es. Antes que nada tenemos que determinar qué es un servicio y luego debemos decidir cuales exponer y cuales desarrollar.

a. Adaptación de Metodología

   Tras analizar la problemática, la demanda generada en cuanto a los servicios se establece los objetivos y se adopta SOA como arquitectura.

b. Definición de la arquitectura SOA

   Una vez establecida los objetivos de la organización se procede a juntar los elementos necesarios para la gestión SOA a fin de garantizar el éxito y cumplimiento de los objetivos.

c. Definición de Fase del Proyecto

   Una vez establecida los objetivos de la organización se procede a juntar los elementos necesarios para la gestión SOA a fin de garantizar el éxito y cumplimiento de los objetivos.

d. Plan de Implantación

   Se procede a realizar la planificación y el dimensionamiento de la iteración a abordar dentro de la implementación SOA.

Según el documento de Arquitectura de Referencia de Claro (2016), propone adoptar una Arquitectura Empresarial que esté acorde a la necesidad del negocio, con el propósito de ayudar a que las tecnologías de la información (TI) se alineen con mayor eficacia a la

misión y objetivos para el área de TI que se ajuste a las necesidades de la empresa.

La razón principal para desarrollar una arquitectura empresarial es la de soportar el negocio mediante la provisión de una tecnología base y una estructura de procesos enmarcada dentro de una estrategia de tecnología de información.

Una arquitectura basada de capas significa la descomposición de los servicios de tal manera que la mayoría de las interacciones no se producen entre sus límites o fronteras. Sin embargo, no existe una regla estricta que las capas superiores no deben comunicarse directamente con las capas inferiores, esta regla se define con el área de arquitectura.

La figura 10 son las diferentes capas de la arquitectura orientada a servicios para el despliegue de componentes de nivel empresarial de las aplicaciones y sistemas propuesta para Claro.

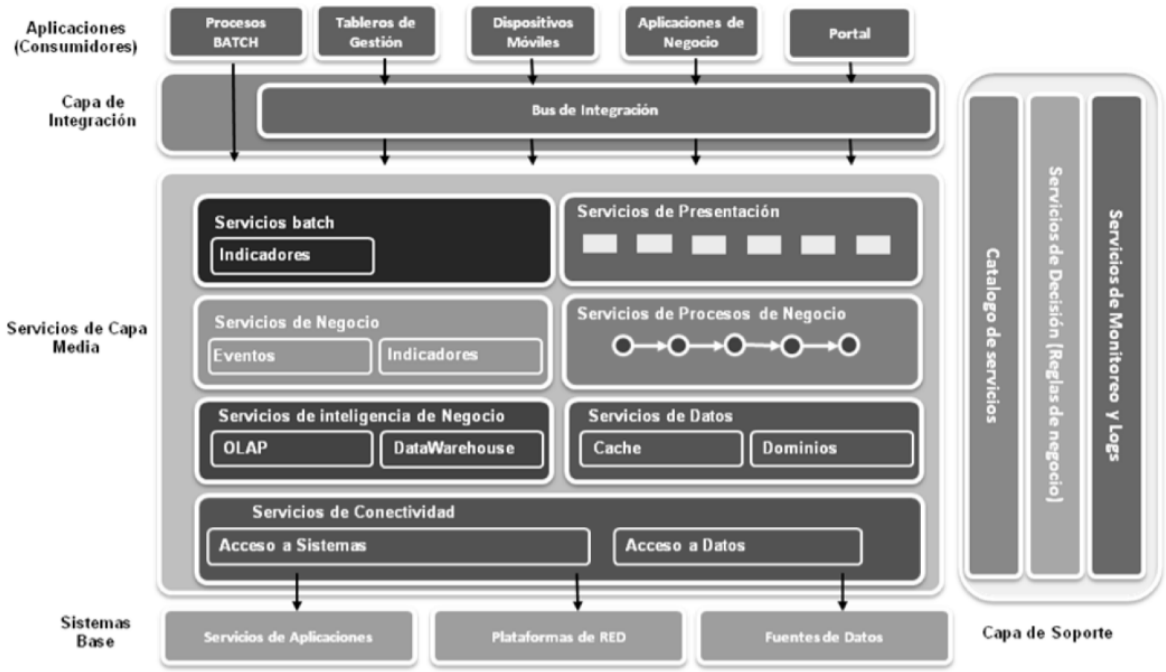



***Figura 10.*** ***Arquitectura de Referencia***

Fuente: (América Móvil, 2016)

### 2.2.5. Integración de Aplicaciones Empresariales (EAI)

#### 2.2.5.1. Definición

*Enterprise Application Integration* (EAI) o Integración de Aplicaciones de Empresa se define como el uso de software y principios de arquitectura de sistemas para integrar un conjunto de aplicaciones. (*Enterprise application integration*, s.f)

#### 2.2.5.2. Origen

Las empresas constantemente están implementando soluciones en algunos casos informales, tanto a nivel de

negocio como también a nivel técnico. Por consecuencia surgen sistemas aislados.

Debido alta necesidad de comunicación, intercambios de datos, información, entre otros se ha desarrollado una disciplina que tiene por objetivo principal lograr la comunicación de todos los sistemas existentes en la empresa, es lo que ahora comencemos como la integración de aplicaciones empresariales. (Pinedo y Medina, 2010)

#### 2.2.5.3. Justificación de EAI

Pinedo & Medina (2010) afirman que “EAI es el proceso de conectar las aplicaciones unas con otras para intercambiar información” (p.71)

Sucede que cuando las los sistemas no pueden compartir la información correctamente, generan problemas, se crean los llamados cuellos de botella, que lo único que hacen es retrasar el proceso o la ejecución del sistema y es necesario la intervención humana para tomar decisiones o en el mayor de los casos ingresar la información manualmente cuando se debió dar de forma automática.

Durante varias generaciones los sistemas empresariales han servido para un propósito específico a un único usuario o más aún a un grupo de usuarios, hoy en día la relaciones entre

los componentes que conforman un sistema son muchos incluso la transferencia de la información es más compleja ya que implica mayor precisión, mejor respuesta, conexiones con otros sistemas más amplios y modernos. Uno de los objetivos de las empresas actualmente es dar información completa, relevante en tiempo real y EAI busca resolver muchos de estos problemas. (*Enterprise application integration*, s.f).

#### 2.2.5.4. Objetivos de EAI

EAI puede ser usado para diferentes fines tales como los que se mencionan a continuación:

- Integración de datos (información): asegurando que la información en varios sistemas sea consistente. Esto también se conoce como EII (*Enterprice Information Integration*).
- Integración de procesos: enlace de los procesos de negocio entre diferentes aplicaciones.
- Independencia del proveedor: extrayendo las políticas o reglas del negocio de las aplicaciones e implementándolas en un sistema EAI, de forma que cualquiera de las aplicaciones usadas pueda ser cambiada sin que dichas reglas de negocio deban ser re implementadas.

- Facade común: Un sistema EAI puede actuar como el *front-end* de un cúmulo de aplicaciones.

(Pinedo & Medina, 2010).

### 2.2.6. Herramientas de desarrollo de software

Una herramienta de desarrollo de software es un programa informático que usa un especialista en programación para crear, depurar, gestionar o mantener programas, aplicativos y sistemas.

#### 2.2.6.1. Sistema de gestión de base de datos (SGBD)

Sistema de gestión de base de datos (s.f), el SGBD es un conjunto de programas que permite el almacenamiento, modificación y extracción de la información de la información de una base de datos, además proporciona también herramientas que permitan añadir, borrar, modificar y analizar los datos. Para poder acceder a la información los usuarios pueden hacer uso de herramientas específicas de consultas y generación de reportes.

a) Oracle Database:

Es un sistema de gestión de base de datos de tipo objeto-relacional creado por Oracle Corporation. Se considera a Oracle Database como uno de los sistemas

más completos, destacando el soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma.

Oracle Corporation es una de las mayores compañías de software del mundo. Sus productos desde bases de datos (Oracle) hasta sistemas de gestión (Oracle Database, s.f).

Se eligió usar la base de datos Oracle por ser muy robusto y con una gran cantidad de posibilidades como el soporte multiplataforma o de transacciones entre otras. Oracle posee muchas versiones de las cuales hay versiones de pago y versiones gratuitas, ya que actualmente Everis y Claro trabajan con esa base de datos, y por otro lado es lo que el cliente solicita.

b) PL SQL Developer Oracle

Andrade, Jarama y Morán (2007) afirman que PL SQL es un lenguaje procedimental desarrollado por Oracle como extensión al SQL estándar. El objetivo de éste lenguaje es proveer un entorno para la ejecución de la lógica procedimental en la base de datos. Esta aplicación es instalada como un cliente de la base de datos.

Mientras que para los autores Beltrán, Castro y Merchán (2009) PL SQL viene a ser una herramienta que incorpora opciones avanzadas para control y el tratamiento de errores denominados excepciones, manejo de cursores, una variedad de procedimientos y funciones empaquetadas incorporadas en el módulo SQL para la programación de disparadores (Trigger) y procedimientos del usuario (Procedure).

Para la implementación de la propuesta se hace uso de esta herramienta ya que por un lado se viene usando en la empresa y otro porque según lo mencionado por los autores mencionados líneas arriba otorga grandes beneficios a la organización ya que es una herramienta muy potente.

#### 2.2.6.2. JEE (J2EE) Java

Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Este modelo permite utilizar arquitecturas de N capas y se apoya ampliamente en componentes de software modulares ejecutándose sobre el un servidor de aplicaciones. (Java EE, s.f)

Es un lenguaje orientado a objetos, robusto, permite una localización temprana de errores, facilita la creación de sistemas interactivos y presenta alto rendimiento, dinámico (Beltrán et al, 2009).

#### 2.2.6.3. Servidor de aplicaciones

En informática, se denomina servidor de aplicaciones a un servidor en una red de computadores que ejecuta ciertas aplicaciones.

Usualmente se trata de un dispositivo de software que proporciona servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Los principales beneficios de la aplicación de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones. (Servidor de aplicaciones, s.f).

##### 2.2.6.3.1. Oracle WebLogic Server

Según Herrera, Guillén y Javier (2012) Oracle WebLogic es un servidor de aplicaciones Java EE (J2EE) y también un servidor web HTTP, desarrollado por BEA *Systems*, posteriormente adquirida por Oracle *Corporation*.

Se ejecuta en Unix, Linux, Microsoft Windows, y otras plataformas.

Mientras que para Ordóñez (2015) Oracle WebLogic Server es un servidor de aplicaciones que se ejecuta a nivel intermedio entre las bases de datos, aplicaciones relacionadas y clientes ligeros basados en navegación web. Basado en Java 2 *Platform*, *Entrerprice Edition* (J2EE) puede automatizar muchas tareas a nivel de sistema que de otro modo habrían exigido el tiempo de programación, es de fácil administración y gestiona eficientemente los recursos, especialmente con productos Oracle aunque es compatible con otros entornos y bases de datos a través de JDBC, trabaja con diferentes estándares de seguridad como *Secure Sockets Layer* (SSL) para el cifrado de las transmisiones de datos, así como otros mecanismos de autenticación y autorización, haciendo que las aplicaciones y transacciones sean más seguras.

Actualmente en Claro usa *WebLogic* como servidor de aplicaciones, contiene los despliegues de los servicios implementados, las configuraciones de las colas (almacenamiento de solicitudes, errores), JMS, origen de datos, etc.

Según esto y lo mencionado por los autores, para la implementación de la propuesta se usará WebLogic como servidor de aplicaciones.

La figura 11 mostrada a continuación muestra la interfaz principal del servidor WebLogic.


***Figura 11. Servidor Weblogic Oracle***

Fuente: Elaboración Propia

2.2.6.3.2. JNDI

Según Herrera, Guillén y Javier (2012) el JNDI es una interfaz de programación (API) que proporciona funcionalidades de nombrado y directorio a las aplicaciones desarrolladas bajo el lenguaje Java. Está definido para ser independiente de cualquier implementación de servicios de directorio. Así se puede acceder a una gran variedad de directorios.

Al momento de realizar un servicio nuevo se hace la configuración del JDNI en le properties externo el cual contiene las configuraciones de los servidores, bases de datos, etc.

#### 2.2.6.4. Spring (Framework)

Gil (2016) presentó un proyecto titulado “Servicio Web REST de gestión de eventos con Java Spring Framework” define a Spring como marco de trabajo que se utiliza para el desarrollo de aplicaciones y a su vez cómo un contenedor de inversión de control, es de código abierto y se utiliza como lenguaje de programación. La primera versión fue escrita por Rod Johnson, quien lo lanzó junto a la publicación de su libro Expert Oneon-One J2EE Design and Development (Wrox Press, octubre 2002).

Por otro lado Rivas, R. (2013) utilizó Spring como marco de trabajo para el desarrollo de un sistemas de préstamos bibliotecario, el autor indica que la gran mayoría de los Framework para el lenguaje de programación java tienen como función simplificar el desarrollo de aplicaciones java, pero en realidad existen muchos que no ayudan a que la programación sea más sencilla debido a que normalmente algunos Framework piden que las clases java hereden clases propias

del Framework. El problema es que si en algún momento se decide usar otro Framework se tiene que modificar todas las clases Java del proyecto, el cual implica costo de mantenimiento de la aplicación. En muchas ocasiones se usan varios Framework para la creación de aplicaciones y cada uno genera su propio conjunto de objetos; Gracias a esto surgió Spring.

Spring ayuda a solventar el problema ya que cambia las responsabilidades y en vez de que el propio desarrollador sea el encargado de generar los objetos de cada uno de los Framework es Spring basándose en ficheros XML a anotaciones el encargado de construir todos los objetos que la aplicación va a utilizar, Figura 12.

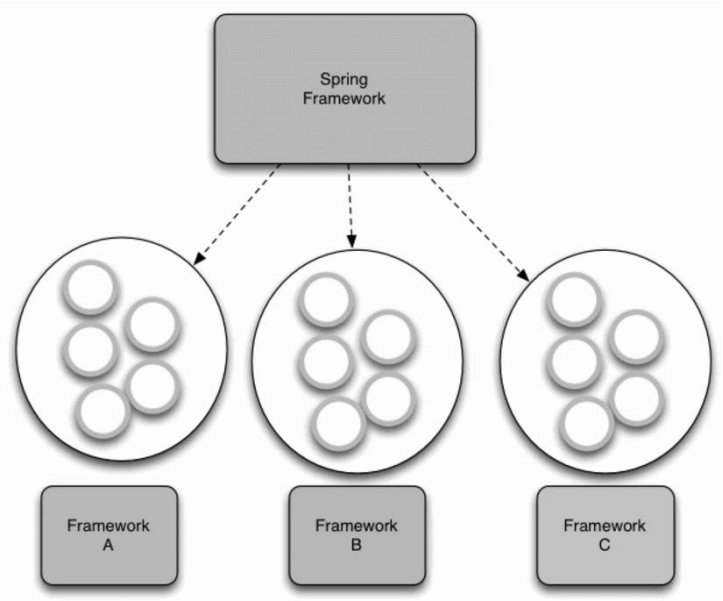



***Figura 12. Creación de objetos Spring***

Fuente: (Álvarez, 2014)

De esta manera al ser Spring el encargado de utilizar todos los objetos de los distintos Framework, es también el responsable de asegurarnos que se integran de la forma correcta, ver el detalle en la figura 13.

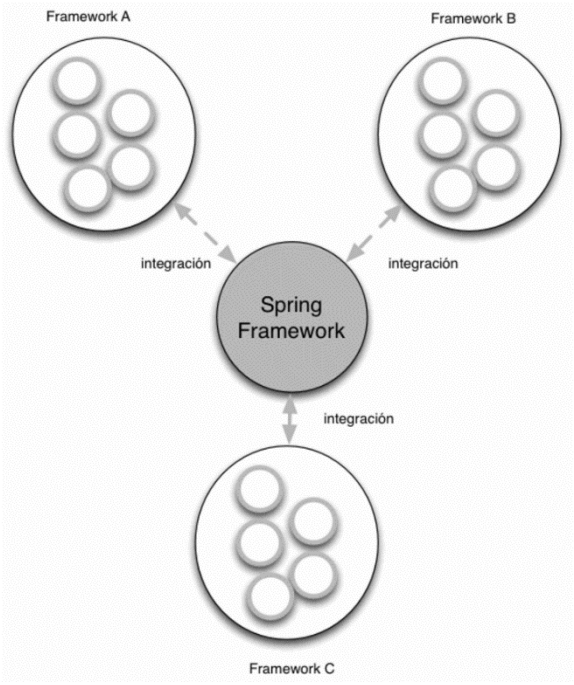



***Figura 13. Integración de Objetos Spring***

Fuente: (Álvarez, 2014)

Así pues en el proyecto usaremos este Framework que con amplios ficheros XML se encargan de inicializar los diferentes objetos como se muestra en la Figura 14.

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

   <bean id="ServicioA" class="com.genbetadev.ServicioA">
       <property name="mensaje" value="Hola GenBetaDev"/>
   </bean>

    <bean id="ServicioB" class="com.genbetadev.ServicioB">
       <property name="mensaje" value="Hola GenBetaDev 2"/>
   </bean>

</beans>
```

***Figura 14. Spring Código***

Fuente: (Álvarez, 2014)

Después de lo descritos sobre Spring vemos que es de gran utilidad para los proyectos que presenten estas características, por otro lado es un marco de trabajo que se está usando actualmente para el desarrollo componentes, a través de sus librerías y su forma de trabajar se hace más fácil la programación de los servicios requeridos para el proyecto.

### 2.2.6.5. SoapUI

Según el lanzamiento de SourceForge (2005) define a SoapUI como una herramienta, desarrollada en java, para la realización de pruebas a aplicaciones con arquitectura

orientada a servicio (SOA) y transferencia de estado representacional (REST). Soporta múltiples protocolos como SOAP, REST, HTTP, JMS, AMF y JDBC. Posee una versión de Código abierto y otra versión de pago realizada por la compañía *SmartBear*.

Para el proyecto necesitaremos de ésta herramienta para hacer las pruebas correspondientes a los servicios implementados para la mejora del proceso.

#### 2.2.6.6. Bizagi BPM Suite (BPMS)

Bizagi (2014) afirma que es una solución para el manejo de los procesos de negocio (*Business Process Management*) que le permite a la organización modelar, automatizar, ejecutar y mejorar los procesos de negocio a través de un entorno gráfico y con cantidades mínimas de programación lo cual permite alcanzar altos grados de productividad, eficiencia y un crecimiento rentable.

Para la propuesta es sumamente importante contar con esta herramienta ya que sirve para el modelamiento de los métodos de los servicios a implementar, impone reglas que se debe seguir para tener un proceso eficiente, se diagraman las actividades, permite conocer el flujo de los procesos de negocio.

Actualmente ya se cuenta con este solo sería necesario reutilizarlo.

### 2.2.7. SHELL

Una Shell de Unix o también Shell, término utilizado para referirse a un intérprete de comandos, el cual consiste en la interfaz de usuario tradicional de los sistemas operativos basados en Unix y similares como GNU/Linux.

Aguilera (2014) define a Shell como un programa utilitario del sistema, interprete de comandos es decir actúa como una interfaz alfanumérica y a su vez es un programa informático, hace posible que el sistema operativo realice órdenes deseadas por el usuario mediante una serie de comandos. Para la propuesta éste componente se encargará de enviar una cantidad de mensajes de una cola origen a una cola destino para su reproceso.

¿Cuándo es necesario utilizar una Shell?

Se usa mucho para realizar tareas repetitivas y automatizadas. Al ser un lenguaje que reside en el sistema operativo, el rendimiento es mucho mejor comparado con otro tipo de lenguaje de programación.

Los usos comunes que encontramos en Shell son los siguientes:

- Ejecución de programas
- Importaciones y Exportaciones de archivos

- Conexión a base de datos
- Transferencia de archivos por FTP
- Automatizar procesos

En la figura 15 se puede apreciar la utilidad de la Shell y cómo es que se realizan las llamadas

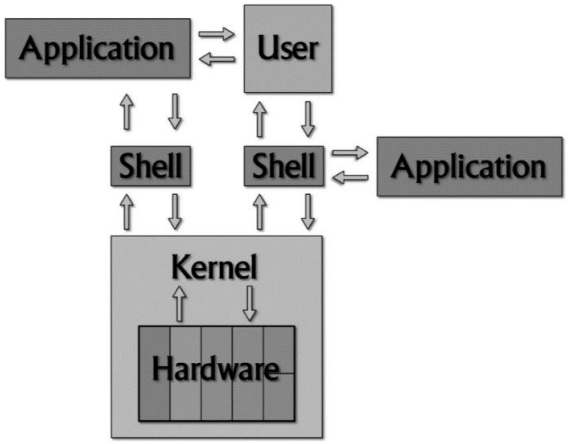



***Figura 15. Utilidad Shell***

Fuente: Elaboración Propia

**Estructura de carpetas SHELL**

Según el estándar de programación de Claro, menciona que se debe generar la siguiente estructura de carpetas:

- **/input:** Ficheros de entrada.
- **/output:** Ficheros de salida, resultados del proceso de la Shell.
- **/shell:** Se colocan TODOS los archivos .sh
- **/log:** Almacena los archivos .log que va dejando el programa.

- **/sql:** Ficheros .sql usados por las shells cuando se conectan a BD.
- **/ctl:** Ficheros .ctl. Solo cuando se utiliza la herramienta SQL Loader.
- **/sms:** Lista de números telefónicos, en caso de enviar SMS
- **/versión:** Se colocarán las versiones del programa Shell luego del pase.
- **/temp:** Directorio temporal para el almacenamiento de archivos.
- **/lib:** Archivos de librerías.

En el proyecto éste componente se encargará de pasar un lote (contiene 200 peticiones) de solicitudes de una cola a otra donde se pueda procesar y no genere saturación del servidor.

### 2.2.8. Queue (cola)

Según Leandro (2015) una *queue* “es una ubicación temporal usada para almacenar peticiones que no pueden ser realizadas inmediatamente”. Una *queue* o una cola es un tipo de estructura de datos caracterizada por ser una secuencia de elementos.

*Message queue* es un tipo de cola usado por servidores de e-mail SMTP donde el servidor procesa los mensajes entrantes y salientes desde un buffer FIFO.

Las colas realizan operaciones de inserción, eliminación, e inspecciones.

Blancarte (2014) en su intento por definir que es una queue (cola) lo refiere “como una estructura de datos muy utilizada en todos los sectores las cuales permiten manejar mejor todo tipo de soluciones”.

En la figura 16 se explica cómo un elemento es agregado a la cola y cómo queda luego de que el nuevo elemento entra a la cola. Cuando un elemento nuevo entra a una cola se posiciona siempre al final de la cola, así mismo éste será el último en salir.

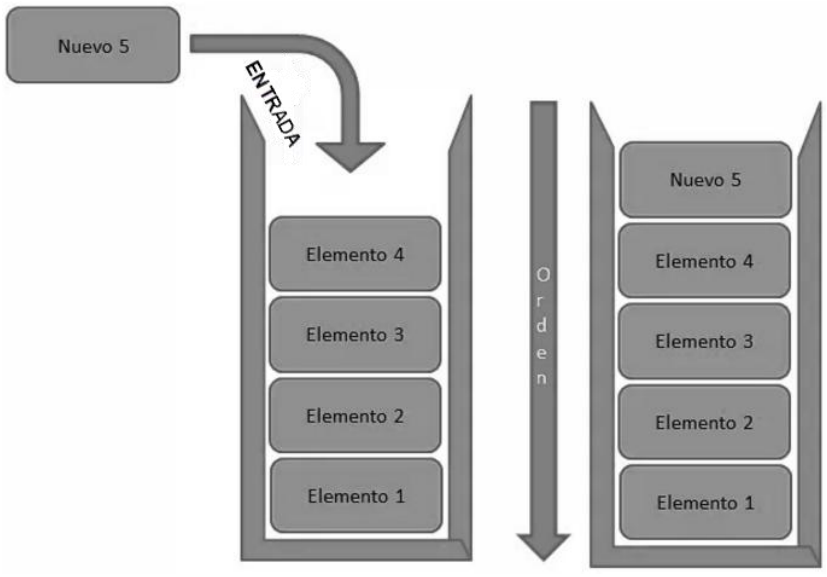



***Figura 16. Agregar elementos a una Cola***

Fuente: (Blancarte, 2014)

En la figura 17 se observa cómo es que un elemento es extraído, retirado de la cola (izquierda) y cómo queda la cola (derecha) después de retirar el elemento.

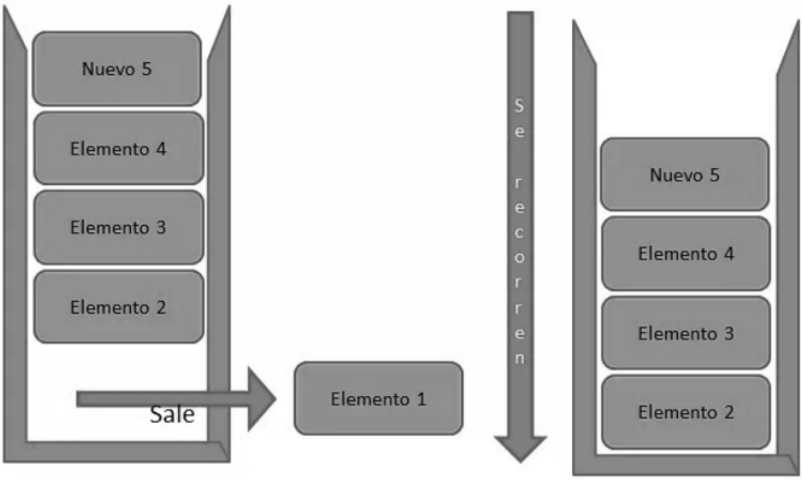



***Figura 17. Extraer elementos de la Cola***

Fuente: (Blancarte, 2014)

En ambos casos los autores señalan que las lecturas de las colas sigues la filosofía FIFO del inglés *First In – Firts Out* que traducido al español sería “Primero en entrar primero en salir”. Esto quiere decir que el primero elemento que ingresó a la cola será el primero que salga y el último que entre será último en salir.

Para el proyecto las colas servirán de almacenamiento de mensajes (solicitudes o peticiones), cada mensaje actualmente tiene alrededor de 200 registros o solicitudes y por ser estos de gran tamaño es necesario contar con colas que ayuden a que todos sean

procesados, claro no todos al mismo tempo pero sí siguiendo un orden.

### 2.2.9. SOAP

Emaildia (2007) define a SOAP como “un protocolo que da soporte a la interacción (datos + funcionalidad) entre aplicaciones en entornos distribuidos y heterogéneos, es interoperable (neutral a plataforma y lenguajes de programación, independiente del hardware y protocolos)”. Funciona sobre la infraestructura (estándares) existente en Internet. SOAP define cómo organizar información usando XML de forma estructurada y tipada para intercambiarla entre distintos sistemas.

SOAP (Simple Object Access Protocol), es un protocolo simple para el intercambio de información estructurada en un entorno distribuido y descentralizado. Utiliza XML para definir un framework extensible de mensajería proveyendo un formato de mensaje que puede ser intercambiado sobre una variedad de protocolos subyacentes. El Framework fue diseñado para ser independiente de cualquier modelo de programación o cualquier semántica específica de alguna implementación. SOAP define el formato XML para mensajes.

Pinedo y Medina (2010) existen otras partes en la especificación SOAP que describen cómo representar los datos de un programa como XML y cómo utilizar SOAP para realizar llamadas a procedimiento remoto (RPC). Estas partes opcionales de la especificación se utilizan para implementar aplicaciones estilo RPC en las que el cliente envía un mensaje SOAP que contiene una función a la que se puede llamar, además de los parámetros para pasar a la función. El servidor, por su parte, devuelve un mensaje con los resultados de la función ejecutada. Las aplicaciones SOAP con estilo de documento son muy flexibles. Muchos nuevos servicios XML Web Services aprovechan esta flexibilidad para diseñar servicios que sería difícil implementar mediante RPC.

La característica más notable de SOAP es que se ha implementado en diferentes plataformas de hardware y software, lo que significa que puede utilizarse para enlazar sistemas dispares dentro y fuera de una empresa. En el pasado se realizaron muchos intentos para conseguir un protocolo de comunicaciones común que pudiera usarse en la integración de sistemas. Sin embargo, ningún intento se ha generalizado tanto como SOAP. (Pinedo y Medina, 2010)

## 2.3. MARCO CONCEPTUAL

Es esta sección es preciso aclarar algunas definiciones relevantes para la comprensión del proyecto, en base a la información previamente revisada con la cual se explicará palabras claves como migración, mejora de procesos, cambio de plan, topes de consumo, metodología cascada, Arquitectura Orientada a Servicios (SOA), integración (EAI), *Message Driven Bean* (MDB), Shell, Java EE, *Queue* (colas), aplicación, *Oracle Weblogic* (servidor) y otras que sean necesario definir.

### 2.3.1. Proceso

Pérez (2010) define a proceso como: “Secuencia (ordenada) de actividades (repetitivas) cuyo producto tiene valor intrínseco para su usuario o cliente” (p.51).

La figura 18 muestra los elementos que conforman un proceso.

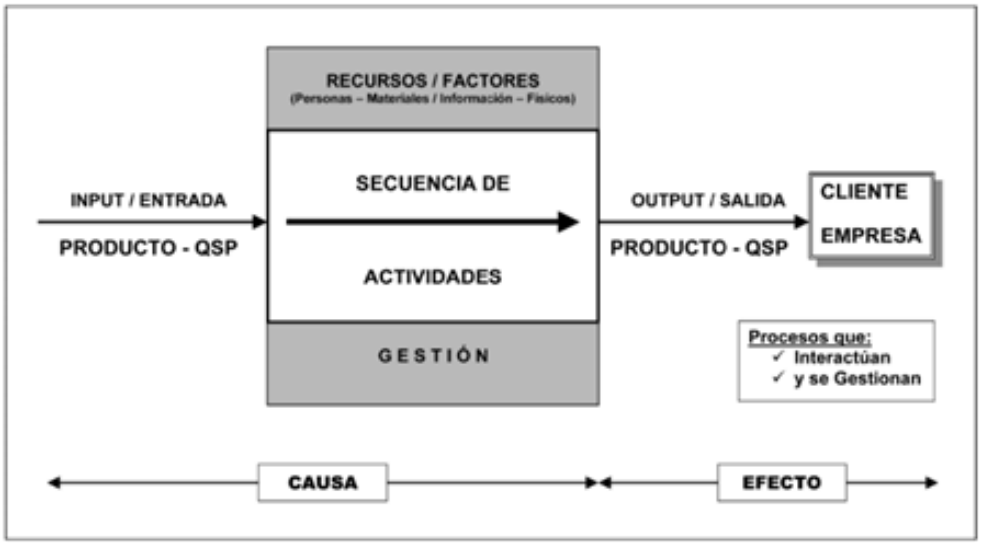



***Figura 18. Elementos de un proceso***

Fuente: (Pérez, 2010)

### 2.3.2. Mejora de procesos

La mejora de procesos, significa optimizar la efectividad y la eficiencia, mejorando también los controles, reforzando los mecanismos internos para responder a las contingencias y las demandas de nuevos futuros clientes. La mejora de procesos es un reto para toda empresa de estructura tradicional y para sistemas jerárquicos convencionales (Universidad de Champagnat, 2002).

El autor señala que para mejorar un proceso se debe considerar lo siguiente:

- Análisis de los flujos de trabajo
- Fijar objetivos de satisfacción, para conducir a la ejecución de los procesos.
- Desarrollar las actividades de mejora entre los protagonistas del proceso.
- Responsabilidad e involucramiento de los actores del proceso.

### 2.3.3. Migración

Proceso para realizar una modificación en el plan de consumo de un cliente que incluye procesos como el cambio de plan y activación del tope de consumo.

### 2.3.4. Cambio de Plan

Por solicitud de un cliente con línea postpago se modifica el plan contratado previamente por uno mayor o menor según crea conveniente el cliente.

### 2.3.5. Tope de Consumo

Es el límite que se pone al total del plan contratado, es decir es un tope para que el cliente no sobrepase o gaste más del paquete comprado. El tope de consumo implica un costo adicional de cinco nuevos soles, el cual se viene incluido en la factura mensual que emite Claro a las líneas postpago.

### 2.3.6. Metodología de desarrollo Cascada

Vera y Gonzalo (2009) en su proyecto con nombre "Análisis de métodos, técnicas y herramientas de verificación y validación de software usados por empresas ecuatorianas desarrolladoras de software" realizaron un estudio a las empresas ecuatorianas desarrolladores de software indicando que pueden existir muchos factores que pueden afectar la calidad de un producto y uno de estos sería la validación y verificación del software. Definen a la metodología cascada como un modelo que toma las actividades fundamentales del proceso de especificación, desarrollo, validación, evolución y las representa como fases separadas del proceso.

# CAPÍTULO III:
# DESARROLLO DE LA METODOLOGÍA PARA LA PROPUESTA DE IMPLEMENTACIÓN

## 3.1. ANÁLISIS DEL MODELO

Este capítulo comprende el análisis, diseño, desarrollo, y pruebas de los componentes propuestos a crear para el proceso de migración de líneas postpago para la empresa Claro. Por tal motivo nos basaremos en la metodología Cascada que fue descrita en el capítulo previo, la solución comprende desde la fase de Análisis de requerimientos hasta las pruebas finales.

### 3.1.1. Análisis

La información que incluimos a continuación es producto del análisis realizado a la problemática descrita anteriormente, los *stakeholders* dieron el visto bueno del proyecto por ser de gran aporte y porque permite la optimización del proceso, actualmente todavía se está tendiendo problemas con las migraciones, la cual son modificadas manualmente en la base de datos para que se pudieran dar por completado.

Para poder realizar el proceso de cambio de plan y de topes de consumo, decidieron programar ambas ejecuciones con una holgura de 1 día para poder procesar el cambio de plan correctamente, pero no se previó que al realizar el cambio de plan en su proceso automáticamente se desactivaba el tope de consumo originando excesos de consumo de una línea Postpago, por otro lado también se evidencio errores en los tiempos de ejecución de algunos componentes, pues el servidor interpretaba por defecto que una ejecución debería tomar como máximo cinco minutos pero al realizar estos procesos masivos los tiempo de ejecución superaban al de espera generando reprocesos de solicitudes o peticiones de migración que por consecuencia nunca se llega a terminar la solicitud con éxito, es decir las actualizaciones en las base de datos no se dan como deberían conllevando a no poder terminar el proceso.

Además se entiende que al ser componentes de integración este proyecto no solo funcionará para un aplicativo, sino que al estar basado en una arquitectura orientada a servicios estos pueden ser reutilizados por otros proyectos y restar tiempo de ejecución, ya que no serían necesario volverlos a desarrollar si no solo buscar el componente requerido y adaptarlo, claro si es que éste no existe es necesario su creación.

Por todo ello es necesario la implementación de los componentes propuestos que ayuden al proceso de migración y éste

se realice con éxito, siendo ello de mucha importancia ya que el no poder realizar el proceso de migración de forma óptima se generan reclamos, desafiliación de clientes y todo ello significa pérdidas económicas para Claro.

Por consiguiente éste proyecto se desarrolla como una propuesta de implementación siguiente los criterios establecidos por Claro, como el uso de una arquitectura orientada a servicios (SOA), y bajo una metodología de desarrollo Cascada. Para ello definimos el alcance del proyecto:

- Realizar el proceso de migración con éxito.
- Evitar reprocesos de solicitudes y/o peticiones de la migración de líneas.
- Reducir los reclamos por parte de los clientes, y por consiguiente evitar pérdidas económicas para Claro.
- Algo muy importante también la fidelización de los clientes ya que son la razón de ser de la empresa.

Es preciso mencionar algunos puntos necesarios para el proyecto tales como la estimación de costos y calendario, el impacto que tendría el proyecto sobre lo que actualmente existe y la factibilidad del mismo:

#### 3.1.1.1. Estimación de costos y calendario

Para el desarrollo de la estimación de costos es necesario primero desarrollar un plan de proyecto el cual será realizado en base a las fases que nos indica el Modelos Cascada, las mismas se realizaran con una o más iteraciones.

En la siguiente tabla1 se muestra el calendario de las actividades y tares del proyecto. Es necesario mencionar que está sujeto a refinamientos y cambios (nuevas versiones) según sea conveniente y con mutuo acuerdo con las partes interesadas.

Es preciso señalar que la estimación de costos se ha realizado según el método de Juicio de Expertos, el cual consiste en preguntar y guiarnos por el conocimiento y experiencias de las personas que han realizado un trabajo similar al cual le estamos determinando un coste.

**Tabla 1** *Cronograma de Actividades*

| Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|
| **PROYECTO DE MEJORA DE MIGRACIÓN DE LÍNEAS POSTPAGO** | **72 días** | **lun 19/12/2016** | **mar 28/03/2017** |
| **Fase Análisis** | **17 días** | **lun 19/12/2016** | **mar 10/01/17** |
| **Alcance del Proyecto** | **7 días** | **lun 19/12/2016** | **mar 27/12/16** |
| reunión de entendimiento | 2 días | lun 19/12/2016 | mar 20/12/16 |
| definir los alcances | 3 días | mié 21/12/16 | vie 23/12/16 |
| entrega de los alcances | 1 día | lun 26/12/16 | lun 26/12/16 |
| Actualización de los alcances | 1 día | mar 27/12/16 | mar 27/12/16 |
| **Plan de Ejecución del Proyecto** | **4 días** | **mié 28/12/16** | **lun 02/01/17** |
| definir el calendario de trabajo | 1 día | mié 28/12/16 | mié 28/12/16 |

| | | | |
|---|---|---|---|
| entrega de calendario | 1 día | jue 29/12/16 | jue 29/12/16 |
| actualización de calendario | 1 día | vie 30/12/16 | vie 30/12/16 |
| definición y corrección de riesgos | 1 día | lun 02/01/17 | lun 02/01/17 |
| **Modelado** | **6 días** | **mar 03/01/17** | **mar 10/01/17** |
| definición de casos de uso | 2 días | mar 03/01/17 | mié 04/01/17 |
| procesos de negocio | 2 días | jue 05/01/17 | vie 06/01/17 |
| reglas de negocio | 1 día | lun 09/01/17 | lun 09/01/17 |
| modelado de requerimientos | 1 día | mar 10/01/17 | mar 10/01/17 |
| **Fase de diseño** | **16 días** | **mié 11/01/17** | **jue 01/02/17** |
| análisis de propuesta de solución | 1 días | mié 11/01/17 | mie 11/01/17 |
| modelado de arquitecturas | 1 días | jue 12/01/17 | jue 12/01/17 |
| definición de requerimientos | 1 días | vie 13/01/17 | vie 13/01/17 |
| definir casos de uso del sistema | 1,5 días | lun 16/01/17 | mar 17/01/17 |
| análisis del sistema | 1,5 días | mar 17/01/17 | mie 18/01/17 |
| diseño del sistema | 2 días | jue 19/01/17 | vie 20/01/17 |
| elaboración de documento de diseño MDB | 1,5 días | lun 23/01/17 | mar 24/01/17 |
| elaboración de documento de contrato de servicio MDB | 1,5 días | mar 24/01/17 | mie 25/01/17 |
| validación de EDS MDB con arquitecto | 1 día | jue 26/01/17 | jue 26/01/17 |
| elaboración de documento de diseño SHELL | 1,5 días | vie 27/01/17 | lun 30/01/17 |
| elaboración de documento de contrato de servicio SHELL | 1,5 días | lun 30/01/17 | mar 31/01/17 |
| validación de EDS SHELL con arquitecto | 1 día | mie 01/02/17 | mie 01/02/17 |
| **Fase de construcción** | **19 días** | **jue 02/03/17** | **mar 28/02/17** |
| configuración de ambiente de desarrollo | 1 día | jue 02/03/17 | jue 02/03/17 |
| definición de arquitectura de desarrollo | 2 días | vie 03/02/17 | lun 06/02/17 |
| construcción de MDB de aprovision en Janus | 5 días | mar 07/02/17 | lun 13/02/17 |
| construcción Shell de manejo de errores | 5 días | mar 14/02/17 | lun 20/02/17 |
| configuración de Colas WebLogic | 3 días | mar 21/02/17 | jue 23/02/17 |
| creación de servicio JMS | 3 días | vie 24/02/17 | mar 28/02/17 |
| **Fase de pruebas** | **12 días** | **mié 01/03/17** | **jue 16/09/17** |
| pruebas unitarias MDB | 1 día | mié 01/03/17 | mié 01/03/17 |
| pruebas unitaria SHELL | 1 día | jue 02/03/17 | jue 02/03/17 |
| elaboración de plan de prueba MDB | 1,5 días | vie 03/03/17 | lun 06/03/17 |
| elaboración de plan de prueba SHELL | 1,5 días | lun 06/03/17 | mar 07/03/17 |
| creación de manual de instalación MDB | 2 días | mié 08/03/17 | jue 09/03/17 |
| creación de manual de operación MDB | 1,5 días | vie 10/03/17 | lun 13/03/17 |
| creación manual de instalación SHELL | 2 días | lun 13/03/17 | mié 15/03/17 |
| creación manual de operación SHELL | 1,5 días | mié 15/03/17 | jue 16/09/17 |
| **Fase de verificación** | **3 días** | **vie 17/03/17** | **mar 21/03/17** |

| | | | |
|---|---|---|---|
| soporte a certificaciones | 2 días | vie 17/03/17 | lun 20/03/17 |
| pase a producción | 1 día | mar 21/03/17 | mar 21/03/17 |
| **Fase de mantenimiento** | **5 días** | **mié 22/03/17** | **mar 28/03/2017** |
| soporte post producción | 5 días | mié 22/03/17 | mar 28/03/2017 |

Fuente: Elaboración Propia

En la siguiente tabla2 aprecia el recurso humano necesario para la implementación de la propuesta.

**Tabla 2** *Recurso humano del proyecto*

| EQUIPO DE TRABAJO | ACTIVIDADES |
|---|---|
| 1 Analista programador (AP) | Desarrollo y programación |
| 1 Analista funcional (AF) | Diseño y análisis |
| 1 Arquitecto de software (ARQ) | Arquitectura del proyecto y definir los componentes con los que interactúa el proyecto |
| 1 Líder del proyecto (LDP) | Llevar un control del proyecto |

Fuente: Elaboración Propia

Mientras que en la tabla3 se muestra la estimación del costo total del proyecto el cual tendrá un duración de tres meses y medio tal y como lo muestra la tabla1. En cuanto al costo de infraestructura, éste incluye los costos por cada puesto de trabajo (laptop, luz, agua, internet, etc.).

**Tabla 3** *Estimación de costos del proyecto*

| RECURSOS | COSTO DEL PROYECTO |
|---|---|
| humanos | S/. 50800 |
| infraestructura | S/. 9200 |
| **costo total** | **S/. 60000** |

Fuente: Elaboración Propia

3.1.1.2. Estimación del impacto

- Se realizarán operaciones de consulta, modificación sobre las tablas de la base de datos de EAI para actualizar el estado de las líneas que lograron ejecutar la activación del Tope de Consumo.
- Se incluirán validaciones para efectuar correctamente los reprocesos.
- Se incluirá una mejora al momento de realizar las validaciones de aprovisionamiento en Janus (plataforma que contiene varios servicios). El aprovisionamiento Janus valida que todas las desactivaciones de tope se encuentren alineados en todas las fuentes de datos correspondientes y necesarios para ejecutar el proceso sin problemas.
- Se registrará la Programación de la Activación del Tope de Consumo para que se ejecute el mismo día del Cambio de Plan.
- Se registrará la Programación de la Activación del Tope de Consumo para que se ejecute el mismo día del Cambio de Plan por Renovación.

#### 3.1.1.3. Factibilidad del proyecto

- Factibilidad técnica

  Con respecto a la factibilidad técnica del proyecto, no se tendría inconvenientes ya que Everis tienen todas las herramientas necesarias para su implementación, hardware y software.

- Factibilidad económica

  Everis ya cuenta con las herramientas necesarias para proceder con el desarrollo, desde licencias software de Spring, base de datos Oracle, herramientas de pruebas como el SOAPUI, etc. Lo cual solo serán necesario reutilizarlos. En cuanto al personal involucrado, se contará con un Analista Programador (AP), un Analista Funcional (AF), un Arquitecto de software, y el Líder del Proyecto (LDP) quien se encargará de gestionar todo lo necesario para llevar a cabo la ejecución del proyecto.

#### 3.1.1.4. Identificación de requerimientos

Se deben indicar requerimientos funcionales y no funcionales. Así mismo se debe señalar cómo dichos requerimientos resuelven el problema planteado.

#### 3.1.1.4.1. Requerimientos funcionales

A continuación en la tabla 4 se definen las funciones que los componentes a desarrollar serán capaz de realizar para resolver el problema planteado. Cada requerimiento resuelve una necesidad que los usuarios han podido plantear. Cabe mencionar que los requerimientos se han obtenido luego de una reunión de entendimiento con las personas involucradas en el proyecto (*stakeholders*).

**Tabla 4** *Lista de Requerimientos de Usuario*

| REQUERIMIENTOS DEL USUARIO | | |
|---|---|---|
| **CODIGO DE RU** | **REQUERIMIENTO DEL USUARIO** | **SITUACIÓN ACTUAL** |
| RU01 | Para un Cliente que ha realizado Cambio de Plan Postpago + Activación de Tope de Consumo, se requiere que la activación del Tope de Consumo se ejecute el mismo día de la Ejecución del Cambio de Plan(Por lo general es el último día del Ciclo de Facturación) | Actualmente, al realizarse la ejecución del Cambio de Plan, se desactivan los servicios del Plan anterior, incluyendo el servicio Tope de Consumo, para proseguir con la activación de los servicios (menos Tope de Consumo) correspondientes a su nuevo Plan. Por otro lado, la activación del servicio Tope de Consumo, se ejecuta un día después de la ejecución del Cambio de Plan, lo que genera que el Cliente esté un día sin su servicio de Tope de Consumo activo (en el caso que lo tenga activo desde su plan anterior) y como consecuencia el Cliente sobrepase su cargo fijo. |
| RU02 | Agregar validaciones para el Reproceso de las Activaciones/Desactivaciones de Tope de Consumo. | Actualmente, al darse el Reproceso de un mismo request en el método activarDesactivarTopesConsumo del servicio ControlConsumoPostpago, este, al no tener ninguna validación que restrinja que ya no se pueda ejecutar la desactivación/activación de un registro ya procesado, está permitiendo que se vuelva a desactivar/activar el tope. |
| RU03 | Crear Shell para validar el aprovisionamiento en Janus | Actualmente, cuando se desactiva un Tope de Consumo y se envía a la cola de activación *activarTopesConsumoMDB* del MDB *ControlConsumoPostpagoMDB*, esta lo procesa antes de que finalice el proceso de validación del aprovisionamiento en Janus (estado_prv!= 5), ocasionando que finalice el proceso sin realizar la activación del Tope de Consumo. |
| RU04 | Agregar validación en la Cola *ejecutarTopesConsumoMDB* para que valide el código de resultado = 3 y actualice el estado de la línea programada a Pendiente | Actualmente, las líneas se procesan en LOTES de 200 registros cada uno y son enviadas a la cola de ejecución *ejecutarTopesConsumoMDB* del MDB *ControlConsumoPostpagoMDB*, cada registro es procesado por el método *activarDesactivarTopesConsumo*, pero este no contempla el caso de que un cliente al momento de ejecutar la activación ya cuenta con tope activo y cuando estas líneas pasan por la desactivación el input(request inicial) es enviado a la cola de activación *activarTopesConsumoMDB* pero este no concluye el proceso correctamente y el estado del servicio se queda “EN PROCESO”, ya que por parte del *ejecutarTopesConsumoMDB* |

| | | |
|---|---|---|
| | | no presenta una validación para el código de resultado igual a 3 que retorna el método *activarDesactivarTopesConsumo*. Siendo este el motivo por el que las líneas queden en un estado “EN PROCESO”. |

Fuente: Elaboración Propia

**Tabla 5** *Lista de Requerimientos Funcionales*

| RU | RF | Descripción breve del RF | Complejidad | Orden de Atención | Dependencia entre RF’s |
|---|---|---|---|---|---|
| RU01 | RF01 | Registrar la Programación de la Activación del Tope de Consumo para que se ejecute el mismo día del Cambio de Plan | M | 1 | - |
| RU02 | RF02 | Agregar Validación en el SP PKG_CATALOGO_SERVICIOS.REGISTRA_CTRL_CONSUMO para que pueda manejar correctamente los reprocesos. | M | 3 | - |
| RU03 | RF03 | Crear Shell para validar el aprovisionamiento en Janus | M | 4 | - |
| RU04 | RF04 | Agregar validación de Código de Resultado en la Cola ejecutarTopesConsumoMDB | B | 5 | - |

Fuente: Elaboración propia

#### 3.1.1.4.2. Lista de Requerimientos no Funcionales

**Tabla 6** *Lista de Requerimientos no Funcionales*

| REQUERIMIENTOS NO FUNCIONALES | | |
|---|---|---|
| **CODIGO RNF** | **Descripción** | **Relacionado a RU/RF (Cuando es necesario)** |
| **RNF01** | **Reducir los Reclamos generados por Cobros adicionales:**<br>Estos Reclamos son generados por los cargos adicionales que se generan en las facturas de los clientes por sus consumos del día que no tienen activado el Tope de Consumo.<br><br>Datos estadísticos:<br>Promedio de reclamo por día : 284<br>Promedio de reclamo por mes: 8520<br><br>**Conclusión:**<br>Se evitaran **8520** posibles reclamos al mes. | **RF02** |
| **RNF02** | **Reducir las Notas de Crédito generadas para los realizar los ajustes por los Reclamos generados por Cobros adicionales:**<br><br>Cuando el Cliente ha presentado su Reclamo, América Móvil por mantener la fidelidad del cliente genera Notas de Crédito para reducir los cobros adicionales generados por no tener activado su Tope de Consumo.<br><br>Por ejemplo:<br>8520 Reclamos/mes = 8520 Notas de Crédito<br><br>**Conclusión:**<br>Se evitará la generación de **8520** Notas de Crédito por mes. | **RF02** |

| | | |
|---|---|---|
| **RNF03** | **Reducir pérdidas económicas para el Negocio, por las Notas de Crédito generadas por los Reclamos de los Clientes**<br><br>Cada Nota de Crédito se genera con un monto aproximado de 62 soles, lo cual representa pérdidas para el negocio ya que el consumo en realidad se hizo pero la empresa lo asume por el Reclamo generado por el Cliente.<br><br>En promedio:<br>8520 Notas de Crédito = 528,240.00 soles por mes<br><br>**Conclusión:**<br>Se evitará la pérdida de **528,240.00** soles por mes. | **RF02** |

Fuente: Elaboración Propia

### 3.1.2. Diseño

#### 3.1.2.1. Arquitectura

Actualmente Claro está haciendo uso de SOA como arquitectura para el desarrollo de software.

Claro en el documento de arquitectura de referencia indica que la intención de SOA es proporcionar servicios reutilizables comunes que puedan ser aprovechados por una variedad de consumidores. Estos servicios típicamente proceden de funcionalidades y datos ya existentes en la empresa. Por ello se ha visto necesario implementarlo ya que el proceso de migración existe y se reutilizaran componentes que implementados bajo esta arquitectura.

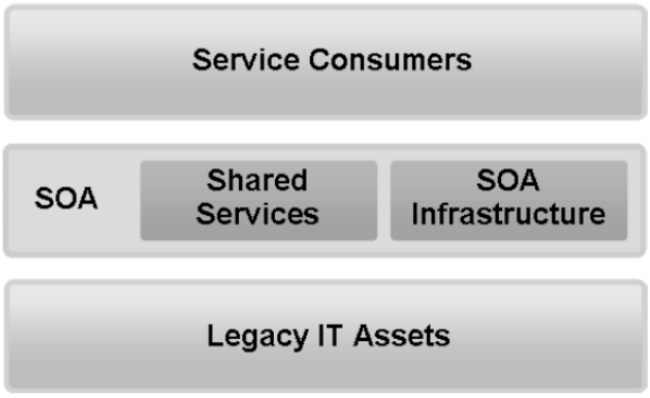



***Figura 19. Arquitectura SOA***

Fuente: (América Móvil, 2016)

Como se muestra en la figura 19, SOA actúa como una capa de valor añadido en la parte de los activos de TI existentes. Expone estos activos, así como las capacidades

personalizadas, como los servicios compartidos. SOA también incluye la infraestructura para ayudar en el descubrimiento, la gestión, la mediación, la vigilancia y seguridad de los servicios.

#### 3.1.2.2. Componentes involucrados en la propuesta de solución

En base a lo definido sobre SOA como arquitectura orientada a servicios los componentes que se muestra en la figura 20 son los requeridos o mejor dicho necesarios para dar la solución en gran parte del problema en mención. El proceso que se ha modificado es el de Aprovisionamiento en Janus (plataforma de componentes), dicho procedimiento se muestra de color verde y seleccionado dentro del recuadro.

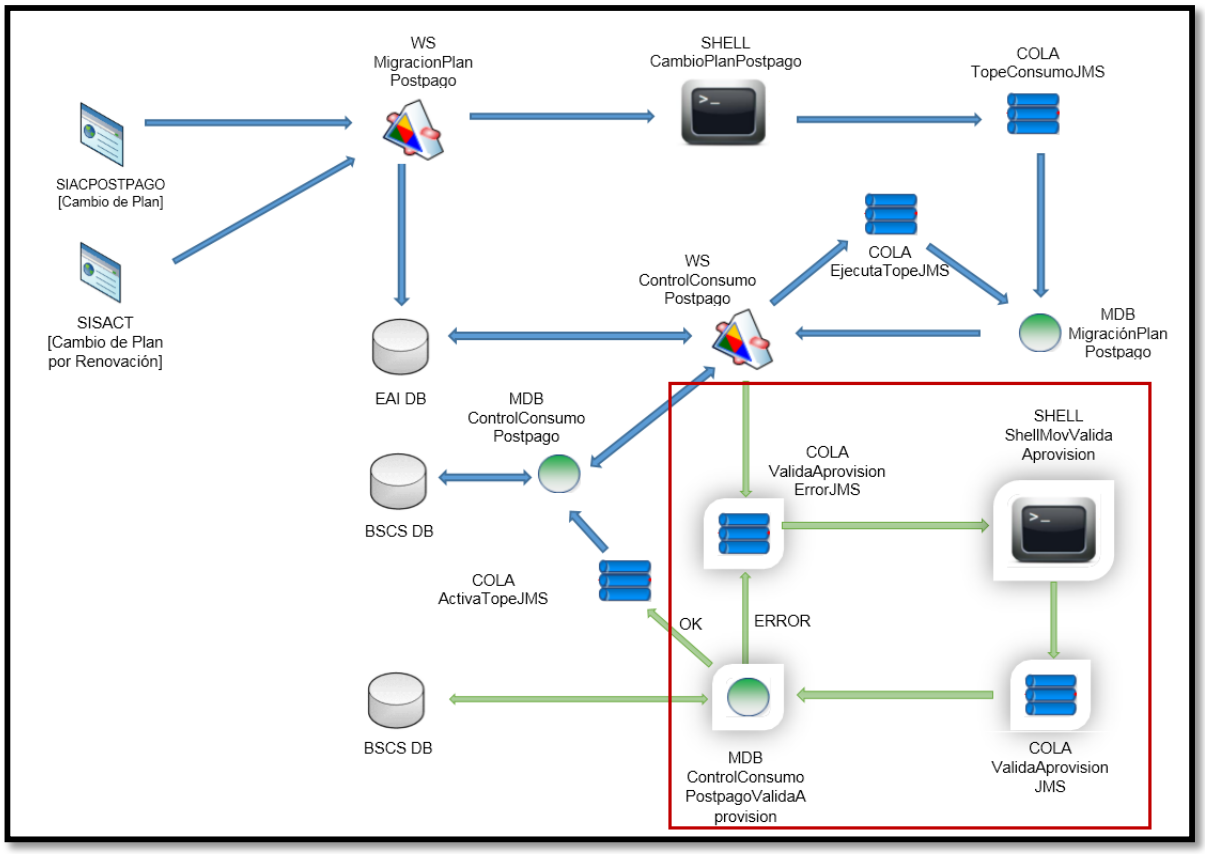



***Figura 20. Propuesta de solución para el proceso de Aprovisionamiento***

Fuente: Elaboración propia

### 3.1.2.3. Desarrollo de Shell ShellMovValidaAprovision

a) Interfaz de Shell ShellMovValidaAprovision

En la figura 21 se aprecia a la Shell encargado de enviar los mensajes de la cola de error (cola de origen) a la cola de reintentos (cola destino) para ser procesados nuevamente. El beneficio de implementar este componente es que permite manejar los mensajes de una forma más flexible, sin sobrecargar el servidor, evitar

reprocesos innecesarios, etc. La Shell se ejecutará automáticamente cada cierto tiempo enviando mensajes de un lado a otro. Cabe recalcar que cada mensajes que contienen cerca de 200 registros de peticiones.


***Figura 21. Interfaz de Shell***

Fuente: Elaboración propia

b) Especificación de cada método del servicio (Shell)

- Método Shell ShellMovValidaAprovision

En la tabla 7 colocamos los datos generales del servicio como la que realizará este componente, los tipos de métodos, detalles más técnicos que funcionales, mientras que en la tabla 8 los parámetros que debe recibir para ejecutar la tarea asignada.

**Tabla 7** *Plantilla de especificación del método del servicio*

| **Nombre del método:** | - | Nuevo X Modificado _ |
|---|---|---|
| **Método:** | Síncrono (Request/Response) __<br>Asíncrono(JMS) __X_<br>Asíncrono (ACK) ___ | |
| **Descripción del Método:** | | |

| | |
|---|---|
| | Envía los mensajes de la cola de error a la cola correcta. |
| **SLA permitido:** | 50 ms |
| **Tipo de notificación ante errores:** | SMS___ Mail___ Otros _logs_ |
| **Estimación de transacciones por segundo** | 10 tps |
| **Cliente que invocará el servicio** | Aplicación _X_ Servicio__ Shell__<br>Cliente Mail: _<br>Lista de clientes:<br>- Control-M |
| **Disponibilidad del servicio** | _24x7_ |

Fuente: Elaboración propia

- Datos entrada del Shell ShellMovValidaAprovision

**Tabla 8** *Datos de entrada para el servicio*

| Nombre | Tipo | Longitud | Descripción | Lista de valores |
|---|---|---|---|---|
| IDTRANSACCION | CHARACTER | - | Id Transacción | - |
| RUTAPROPERTIES | CHARACTER | - | Ruta de archivo properties | - |

Fuente: Elaboración propia

c) Diseño de Shell ShellMovValidaAprovision

- Modelamiento del proceso del método ShellMovValidaAprovisionProcess

La figura 22 muestra el proceso que seguirá la Shell luego de su implementación, las actividades que realizará, a que componentes se comunica, etc.

En la tabla 9 se muestra a detalle la tarea del componente.

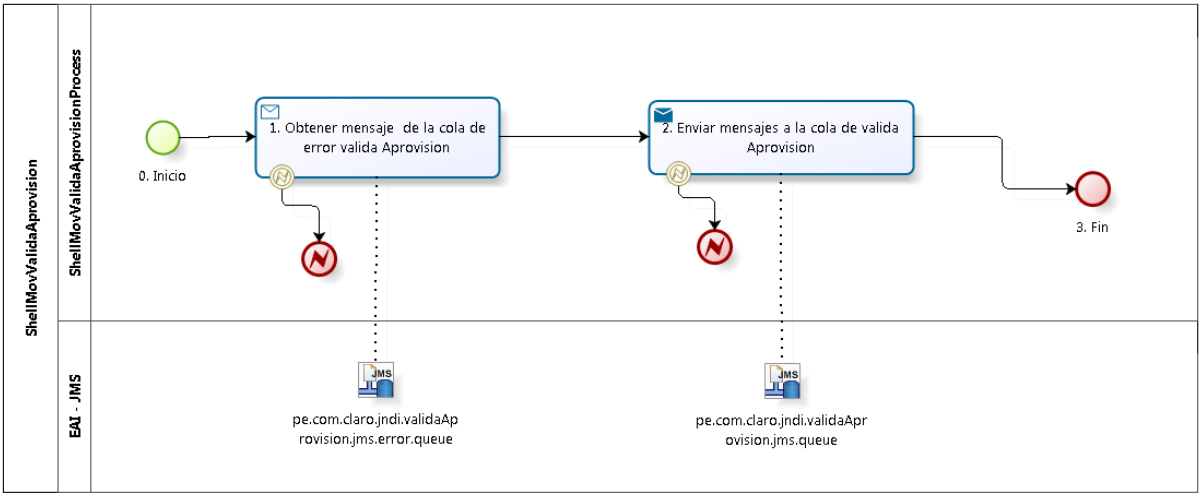



***Figura 22. Flujo de Shell ShellMovValidaAprovision***

Fuente: Elaboración propia

- Descripción de cada actividad del proceso

**Tabla 9** *Descripción de cada actividad de la Shell ShellMovValidaAprovision*

<table>
<tr><th>Actividad del proceso</th><th>Descripción del proceso</th></tr>
<tr><td>0. Inicio</td><td>Una vez ejecutado la Shell: ShellMovValidaAprovision.sh, se hará el siguiente proceso en paralelo Actividad 1 y Actividad 3.</td></tr>
<tr><td>1. Obtener mensajes de la Cola de ERROR Valida Aprovision</td><td>Procedemos a obtener el conjunto de mensajes (mensajeXML) de la cola de error: pe.com.claro.jndi.validaAprovision.jms.error.queue<br><br>ESTRUCTURA DEL MENSAJE:
<table>
<tr><th>NOMBRE</th><th>VALOR</th></tr>
<tr><td>mensajeXML</td><td>XML obtenido de la cola configurada.</td></tr>
</table>
ESPECIFICANDO MENSAJE XML:
<table>
<tr><th>NOMBRE</th><th>VALOR</th></tr>
</table></td></tr>
</table>

| | |
|---|---|
| idTransaccion | idTransaccion (Obtenido de la cola) |
| ipAplicacion | ipAplicacion (Obtenido de la cola) |
| aplicacion | idTransaccion (Obtenido de la cola) |
| msisdn | msisdn (Obtenido de la cola) |
| codigoContrato | codigoContrato (Obtenido de la cola) |
| codigoServPrograma mado | codigoServPrograma do (Obtenido de la cola) |
| codigoServComer cial | codigoServComercial (Obtenido de la cola) |
| descripcionServC omercial | descripcionServComercial (Obtenido de la cola) |
| tipoRegistro | tipoRegistro (Obtenido de la cola) |
| topeControlConsu mo | topeControlConsumo (Obtenido de la cola) |
| flagTopeMenor | flagTopeMenor (Obtenido de la cola) |
| flagLimiteCredito | falgLimiteCredito (Obtenido de la cola) |
| flagTipifica | flagTipifica (Obtenido de la cola) |
| flagTeleventas | flagTeleventas (Obtenido de la cola) |
| cicloFacturacion | cicloFacturacion (Obtenido de la cola) |
| idInteraccion | idIteraccion (Obtenido de la cola) |
| usuarioAplicacion | usuarioAplicacion (Obtenido de la cola) |
| usuarioSistema | usuarioSistema (Obtenido de la cola) |
| tipoServicio | tipoServicio (Obtenido de la cola) |
| fechaProgramacio n | fechaProgramacion (Obtenido de la cola) |
| fechaRegistro | fechaRegistro (Obtenido de la cola) |
| flagValidacion | flagValidacion (Obtenido de la cola) |

**Consideraciones:**

- Si la obtención de los mensajes (mensajeXML) es correcta, ir a la **Actividad 2**.

<table>
<tr><td></td><td>

- Si hay un error al obtener los mensajes del servidor por timeout o disponibilidad de servidor registrar la excepción en el Log y seguir con la **Actividad 3.**

</td></tr>
<tr><td>2. Enviar mensajes a la cola Valida Aprovision</td><td>

Procedemos a enviar el conjunto de mensajes (mensajeXML) a la cola de error a procesar: pe.com.claro.jndi.validaAprovision.jms.queue

**DATOS DE ENTRADA:**

| NOMBRE | VALOR |
|---|---|
| mensajeXML | XML (obtenido de la Actividad 1) |

**ESPECIFICANDO MENSAJE XML:**

| NOMBRE | VALOR |
|---|---|
| idTransaccion | idTransaccion (obtenido de la Actividad 1) |
| ipAplicacion | ipAplicacion (obtenido de la Actividad 1) |
| aplicacion | idTransaccion (obtenido de la Actividad 1) |
| msisdn | msisdn (obtenido de la Actividad 0) |
| codigoContrato | codigoContrato (obtenido de la Actividad 1) |
| codigoServPrograma mado | codigoServProgramado (obtenido de la Actividad 1) |
| codigoServComer cial | codigoServComercial (obtenido de la Actividad 1) |
| descripcionServC omercial | DescripcionServComercial (obtenido de la Actividad 1) |
| tipoRegistro | tipoRegistro (obtenido de la Actividad 1) |
| topeControlConsu mo | topeControlConsumo (obtenido de la Actividad 1) |
| flagTopeMenor | flagTopeMenor (obtenido de la Actividad 1) |
| flagLimiteCredito | falgLimiteCredito (obtenido de la Actividad 1) |
| flagTipifica | flagTipifica (obtenido de la Actividad 1) |
| flagTeleventas | flagTeleventas (obtenido de la Actividad 1) |
| cicloFacturacion | cicloFacturacion (obtenido de la Actividad 1) |
| idInteraccion | idIteraccion (obtenido de la Actividad 1) |

</td></tr>
</table>

| | |
|---|---|
| | <table><tr><td>usuarioAplicacion</td><td>usuarioAplicacion (obtenido de la Actividad 1)</td></tr><tr><td>usuarioSistema</td><td>usuarioSistema (obtenido de la Actividad 1)</td></tr><tr><td>tipoServicio</td><td>tipoServicio (obtenido de la Actividad 1)</td></tr><tr><td>fechaProgramacion</td><td>fechaProgramacion (obtenido de la Actividad 1)</td></tr><tr><td>fechaRegistro</td><td>fechaRegistro (obtenido de la Actividad 1)</td></tr><tr><td>flagValidacion</td><td>flagValidacion (obtenido de la Actividad 1)</td></tr></table><br>**DATOS DE SALIDA:**<br>No aplica.<br><br>**Consideraciones:**<br>• Si el envío de los de mensajes (mensajeXML) es correcta, ir a la **Actividad 3**.<br>• Si hay un error al obtener los mensajes del servidor por timeout o disponibilidad de servidor registrar la excepción en el Log y seguir con la **Actividad 3.** |
| 3. Fin | Finalizar la transacción. |

Fuente: Elaboración propia

Se deberá crear el shell con el nombre **ShellMovValidaAprovision** y la extensión **.sh**. Los valores de configuración para la ejecución del shell estarán definidos en los archivos **.varset**.

3.1.2.4. Desarrollo del MDB ControlConsumoPostpagoValidaAprovisionMDB

a) Diagrama de Interfaz del MDB ControlConsumoPostpagoValidaAprovisionMDB

Servicio que permite cargar todos los mensajes de la cola **validaAprovisionJMS** y validar que la provisión

en **Janus** es correcta. Evitar reclamos por parte del Cliente así como la pérdida de ingresos de los Clientes que realicen tráfico superior al Tope solicitado.

En la figura 23 se observa la interfaz del servicio, el flujo que sigue, las colas con las que se comunica, la base de datos BSCS que consulta para hacer la validación del aprovisionamiento

b) Plantilla de especificación de cada método del servicio

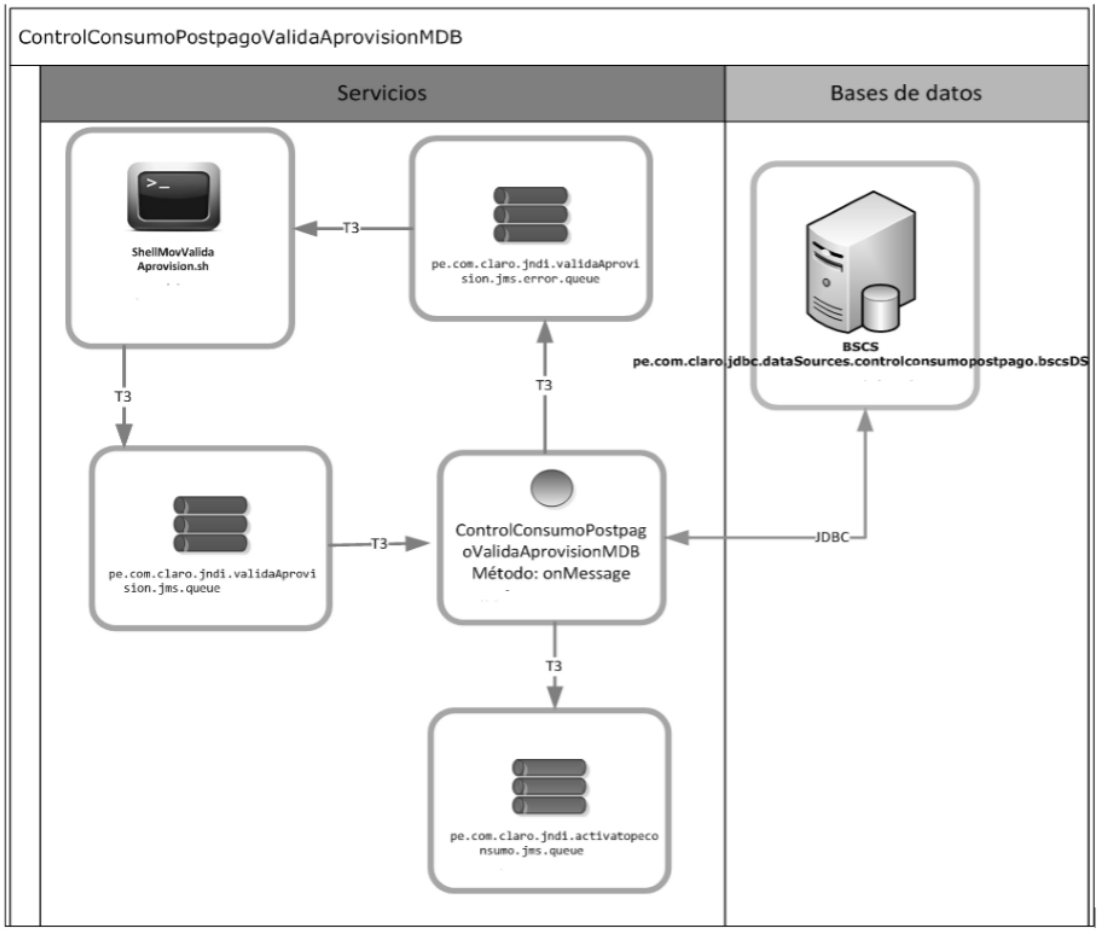



***Figura 23. Diagrama de interfaz del servicio ControlConsumoPostpagoValidaAprovisionMDB***

Fuente: Elaboración propia

En la tabla 10 y 11 respectivamente se muestra a detalle datos generales del servicio, como el tipo de servicio, la cantidad de métodos que utiliza, etc.

**Tabla 10** *Datos generales del servicio ControlConsumoPostpagoValidaAprovisionMDB*

| | |
|---|---|
| **Nombre del servicio** | ControlConsumoPostpagoValidaAprovisionMDB |
| **Descripción del servicio** | Servicio que permite cargar todos los mensajes de la cola validaAprovisionJMS y validar que la provisión en Janus es correcta. |
| **Beneficios del negocio** | Evitar reclamos por parte del Cliente así como la pérdida de ingresos de los Clientes que realicen tráfico superior al Tope solicitado. |
| **Objetivo estratégico relacionado** | Calidad de Atención al Cliente |
| **Número de métodos** | 1 |
| **Clasificación del servicio** | JAVA_X_ BPEL___ OSB Compuesto___ |

Fuente: Elaboración propia

***Tabla 11*** *Método onMessage del MDB ControlConsumoPostpagoValidaAprovisionMDB*

| | | |
|---|---|---|
| **Nombre del método:** | onMessage | Nuevo _X_<br>Modificado __ |
| **Método:** | Síncrono (Request/Response) __ Asíncrono(OneWay)_X_<br>Asíncrono (ACK) __ | |
| **Descripción del Método:** | Valida la aprovisión en Janus línea por línea. Si se aprovisionó entonces envía el mensaje a la cola de activación. | |
| **SLA permitido:** | _50_ ms | |
| **Tipo de notificación ante errores:** | SMS___ Mail___ Otros__ LOGS_X_ | |
| **Estimación de transacciones por segundo** | 10 tps | |
| **Cliente que invocará el servicio** | Aplicación __ Servicio__ Shell_X_<br>Lista de clientes:<br>- ShellMovValidaAprovision.sh | |
| **Disponibilidad del servicio** | _24x7_ | |
| **Palabras claves** | Valida, aprovision, control, consumo, postpago | |

Fuente: Elaboración propia

- Diseño del método onMessage del servicio ControlConsumoPostpagoValidaAprovisionMDB

En la figura 24 el detalle de cada actividad que realiza el MDB, qué base de datos consulta, los componentes que requiere para cumplir con el objetivo.

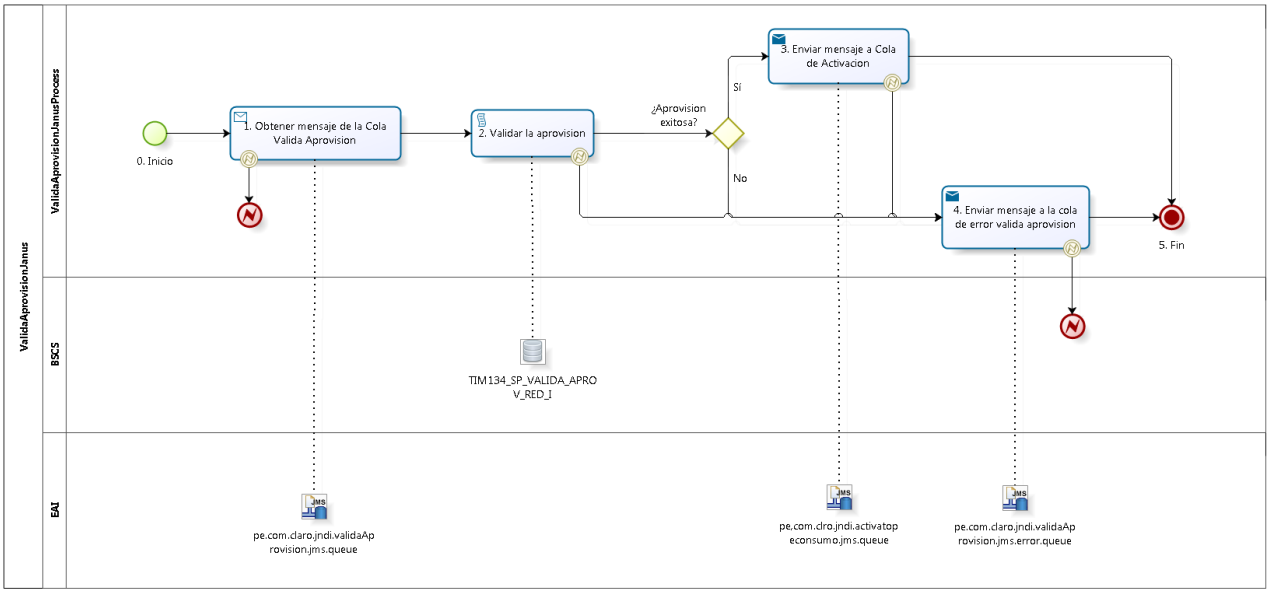



***Figura 24. Secuencia del método onMessage del servicio***

Fuente: Elaboración propia

- Diagrama de Componentes

El diagrama de componentes es para mostrar qué componentes forman parte del proceso, cómo es que éstos se comunican. En la figura 25 se puede apreciar la interacción entre el

servicio, el servidor y la base de datos a través del único método que posee *onMessage.*

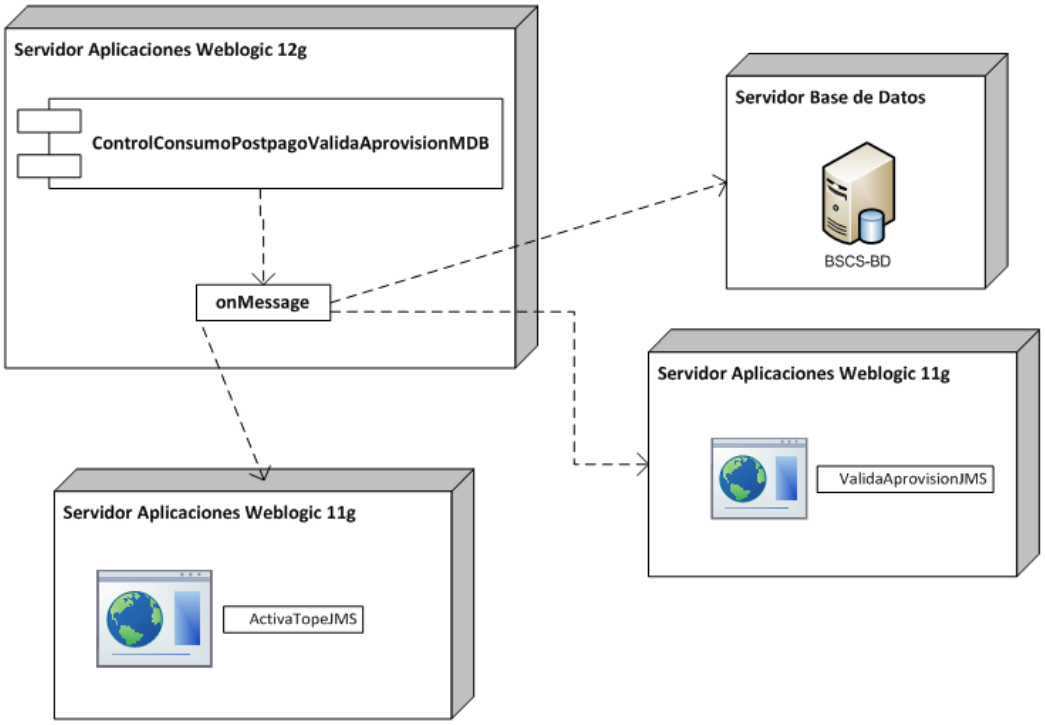



***Figura 25. Diagrama de componentes del servicio***

Fuente: Elaboración propia

- Descripción de cada actividad del proceso

  En este punto se detalla el proceso o la descripción de cada actividad involucrada en el componente (tabla 12).

**Tabla 12** *Descripción de las actividades que forman parte de la secuencia del método onMessage*

<table>
<tr><th>Actividad del proceso</th><th>Descripción del proceso</th></tr>
<tr><td>0. Inicio</td><td>Iniciar el proceso con la Actividad 1.</td></tr>
<tr><td>1. Obtener mensaje de la Cola Valida Aprovision</td><td>Para iniciar el proceso de validación contrato procedemos a obtener el mensaje de la cola: pe.com.claro.jndi.validaAprovision.jms.queue.<br><br>Colocar también la IP de la cola creada en el servidor<br><br>Quien presenta la siguiente estructura:<br><br>ESTRUCTURA DE LA COLA
<table>
<tr><th>NOMBRE</th><th>VALOR</th></tr>
<tr><td>xmlEntrada</td><td>XmlEntrada obtenido de la COLA</td></tr>
</table>
ESPECIFICANDO xmlEntrada:
<table>
<tr><th>NOMBRE</th><th>VALOR</th></tr>
<tr><td>idTransaccion</td><td>idTransaccion (Obtenido de la cola)</td></tr>
<tr><td>ipAplicacion</td><td>ipAplicacion (Obtenido de la cola)</td></tr>
<tr><td>aplicacion</td><td>idTransaccion (Obtenido de la cola)</td></tr>
<tr><td>msisdn</td><td>msisdn (Obtenido de la cola)</td></tr>
<tr><td>codigoContrato</td><td>codigoContrato (Obtenido de la cola)</td></tr>
<tr><td>codigoServProgramado</td><td>codigoServProgramado (Obtenido de la cola)</td></tr>
<tr><td>codigoServComercial</td><td>codigoServComercial (Obtenido de la cola)</td></tr>
<tr><td>descripcionServComercial</td><td>descripcionServComercial (Obtenido de la cola)</td></tr>
<tr><td>tipoRegistro</td><td>tipoRegistro (Obtenido del request)</td></tr>
<tr><td>topeControlConsumo</td><td>topeControlConsumo (Obtenido de la cola)</td></tr>
<tr><td>flagTopeMenor</td><td>flagTopeMenor (Obtenido de la cola)</td></tr>
<tr><td>flagLimiteCredito</td><td>falgLimiteCredito (Obtenido de la cola)</td></tr>
<tr><td>flagTipifica</td><td>flagTipifica (Obtenido de la cola)</td></tr>
<tr><td>flagTeleventas</td><td>flagTeleventas (Obtenido de la cola)</td></tr>
<tr><td>cicloFacturacion</td><td>cicloFacturacion (Obtenido de la cola)</td></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>idInteraccion</td><td>idIteraccion (Obtenido de la cola)</td></tr>
<tr><td>usuarioAplicacion</td><td>usuarioAplicacion (Obtenido de la cola)</td></tr>
<tr><td>usuarioSistema</td><td>usuarioSistema (Obtenido de la cola)</td></tr>
<tr><td>tipoServicio</td><td>tipoServicio (Obtenido de la cola)</td></tr>
<tr><td>fechaProgramacion</td><td>fechaProgramacion (Obtenido de la cola)</td></tr>
<tr><td>fechaRegistro</td><td>fechaRegistro (Obtenido de la cola)</td></tr>
<tr><td>flagValidacion</td><td>flagValidacion (Obtenido de la cola)</td></tr>
</table>
<br>Consideraciones:<br><br>
• Si se obtuvo correctamente el mensaje (xmlEntrada), ir a la Actividad 2.<br>
• Si hay un error al obtener el mensaje del servidor de colas, sea por causas de timeout o disponibilidad, detallar la excepción en el LOG y seguir con la Actividad 5.
</td>
</tr>
<tr>
<td>2. Validar la aprovision</td>
<td>
Para validar la aprovision se procederá a ingresar a la Base de datos BSCS y utilizar el SP:<br><br>
TIM134_SP_VALIDA_APROV_RED_I<br><br>
DATOS DE ENTRADA:
<table>
<tr><th>NOMBRE</th><th>VALOR</th></tr>
<tr><td>CONTRATO</td><td>Obtenido de la actividad 1</td></tr>
<tr><td>CODSER</td><td>Obtenido de la actividad 1</td></tr>
</table>
DATOS DE SALIDA:
<table>
<tr><th>NOMBRE</th><th>DESCRIPCIÓN</th></tr>
<tr><td>RESULTADO</td><td>Valor del resultado de la consulta</td></tr>
<tr><td>MENSAJE</td><td>Descripción del resultado de la consulta</td></tr>
</table>
Consideraciones:<br><br>
• Si la aprovision fue exitoso (RESULTADO = “0”), ir a la Actividad 3, caso contrario ir a la Actividad 4.
</td>
</tr>
</table>

<table>
<tr><td></td><td>

- Si ocurre algún error en la consulta, pintar el error en el LOG y continuar Actividad 4.

</td></tr>
<tr><td>3. Enviar mensaje a Cola de Activacion</td><td>

Para enviar el mensaje xmlEntrada a la cola pe.com.claro.jndi.activatopeconsumo.jms.queue

***Colocar también la IP de la cola creada en el servidor***

**DATOS DE ENTRADA:**

| NOMBRE | VALOR |
|---|---|
| xmlEntrada | Obtenido de la Actividad 1. |

**DATOS DE SALIDA:**
No Aplica

Consideraciones:

- Si el mensaje se envía correctamente, continuar con la Actividad 5.
- Si hay un error al enviar mensaje al servidor de colas por timeout o disponibilidad de servidor registrar la excepción en el Log, continuar con la Actividad 4.

</td></tr>
<tr><td>4. Enviar mensaje a la cola de error valida aprovision</td><td>

Para enviar el mensaje xmlEntrada a la cola pe.com.claro.jndi.validaAprovision.jms.error.queue

***Colocar también la IP de la cola creada en el servidor***

**DATOS DE ENTRADA:**

| NOMBRE | VALOR |
|---|---|
| xmlEntrada | Obtenido de la Actividad 1. |

**DATOS DE SALIDA:**
No Aplica

Consideraciones:

- Si el mensaje se envía correctamente, continuar con la Actividad 5.

</td></tr>
</table>

| | |
|---|---|
| | • Si hay un error al enviar mensaje al servidor de colas por *timeout* o disponibilidad de servidor registrar la excepción en el Log y continuar con la Actividad 5. |
| 5. Fin | Terminar el proceso. |

Fuente: Elaboración propia

3.1.2.5. Creación de la tabla temporal TBL_TMP_TRANS_TOPECONSUMOS

En la base de datos BSCS se crea una tabla temporal que contenga los parámetros mostrados en la tabla 13, para que pueda manejar correctamente los reprocesos.

**Tabla 13** *Parámetros de la tabla temporal*

| Nombre del Atributo | ¿Permite valor Nulo? | Tipo | Descripción |
|---|---|---|---|
| **ID_TRANSACCION** | SI | VARCHAR2(50) | Identificador de la transacción |
| **CO_ID** | NO | INTEGER | Código de contrato |
| **CREATION_DATE** | NO | DATETIME | Fecha de creación |
| **SESSIONRUN_ID** | NO | INTEGER | Usuario de ejecución |

Fuente: Elaboración propia

Asimismo se debe agregar un nuevo parámetro de entrada ID_TRANSACCION al SP PKG_CATALOGO_SERVICIOS.REGISTRA_CTRL_CONSUMO de la base de datos BSCS para hacer posible la transacción.

### 3.1.3. Codificación

#### 3.1.3.1. Creación de las colas (queue)

Los siguientes pasos son para la creación de las colas requeridas en la implementación de la propuesta, solo se estarán mostrando las páginas principales (inicial, medio y final) por seguir varios pasos, para más detalle de ello podemos visitar el Anexo.

Por otro lado es importante aclarar que solo se están creando los componentes propuestos, ya que hay algunos que no se han mencionado, también es necesario tomarlos en cuenta pero como ya están creados solo los reutilizaremos.

Para pasar a la creación de las colas propuestas primero debemos crear e instalar los módulos, servidor JMS el cual nos permitirá comunicar las colas con los servicios web. Para ver el detalle de la creación de estos compontes visitar en el Anexo A y Anexo B.

a) Creación de la cola de mensajería (pe.com.claro.jndi.validaAprovision.jms.queue)

En la figura 26 se procede a iniciar sesión en el servidor WebLogic donde se harán a las configuraciones de las colas a utilizar. Para mayor detalle de la creación de la cola de mensajería visitar el Anexo C.


***Figura 26. Inicio de Sesión del Servidor WebLogic***

Fuente: Elaboración propia

En la figura 27 a continuación ingresamos los datos o parámetros necesarios para la creación de la cola distribuida.

- Name: **validaAprovision.queue**
- JNDI Name: **pe.com.claro.jndi.validaAprovision.jms.queue**
- Destination Type: (por defecto)
- Template:(por defecto)

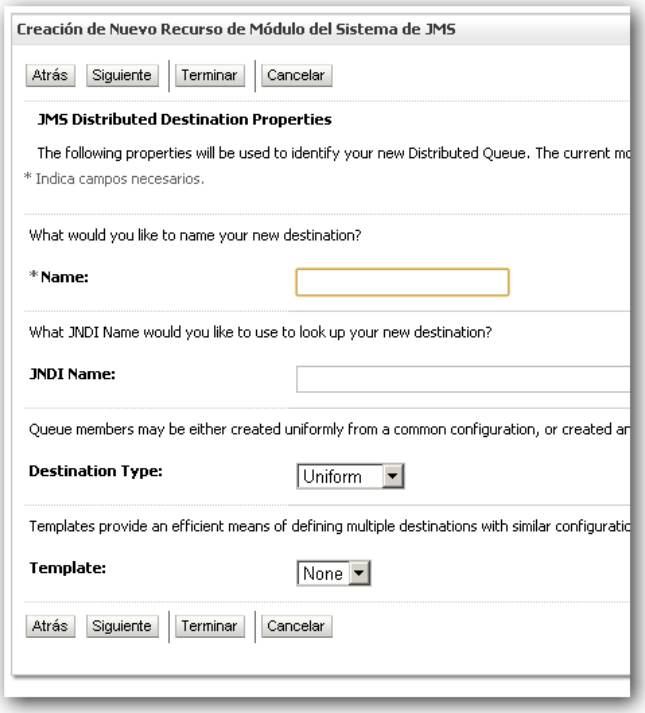



***Figura 27. Solicitud de datos requeridos para la creación de la Cola***

Fuente: Elaboración Propia

En la figura 28 seleccionamos el nodo en el que estará creado la cola de distribución.

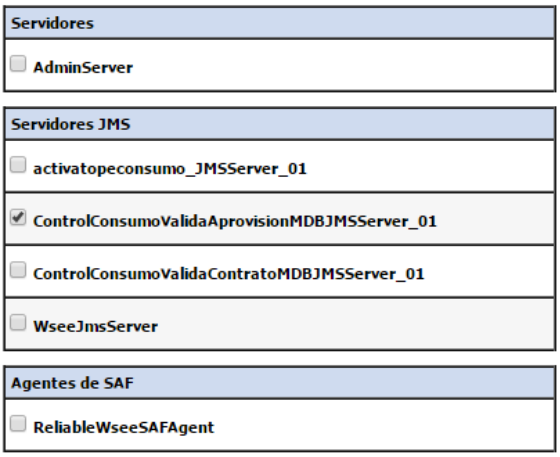



***Figura 28. Selección de los nodos en el servidor***

Fuente: Elaboración propia

La figura 29 es una ventana de confirmación una vez la cola se haya creado y configurado correctamente y es necesario activar los cambios para guarda.

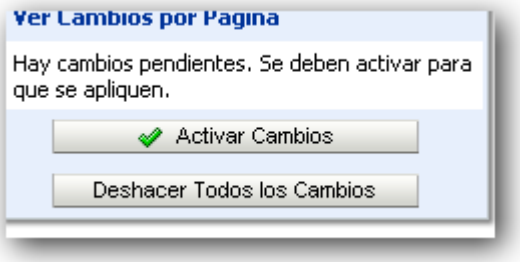



***Figura 29. Confirmar la creación de la Cola***

Fuente: Elaboración propia

b) Creación de la Cola de Error (pe.com.claro.jndi.validaAprovision.jms.error.queue)

Una vez iniciada la sesión al igual que la figura 26 en el servidor se procede crear de la cola destino donde almacenarán los mensajes con errores.

Para la creación de la cola de error debemos seleccionar la Cola distribuida creada un paso anterior tal y como se muestra en la figura 30 a continuación.

Para mayor detalle del procedimiento de la creación de la cola de error visitar el Anexo D.

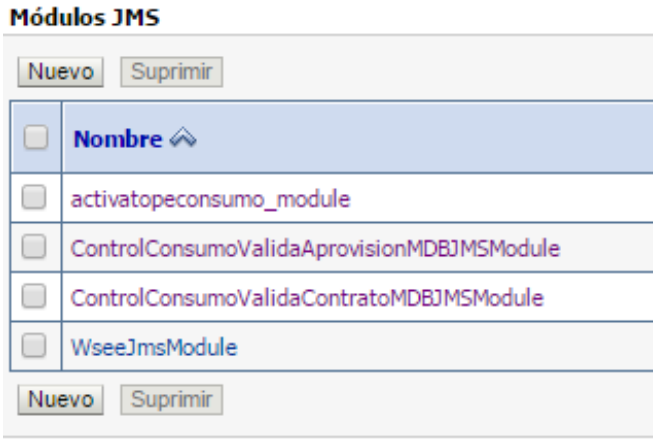



***Figura 30. Selección de Cola Distribuida***

Fuente: Elaboración propia

En la figura 31 ingresamos los siguientes parámetros en las cajas de texto:

- Name: **validaAprovision.error.queue**
- JNDI Name: **pe.com.claro.jndi.validaAprovision.jms.error.queue**
- Destination Type: (por defecto)
- Template:(por defecto)

Cada parámetro es necesario para la creación y configuración de la cola en el servidor.

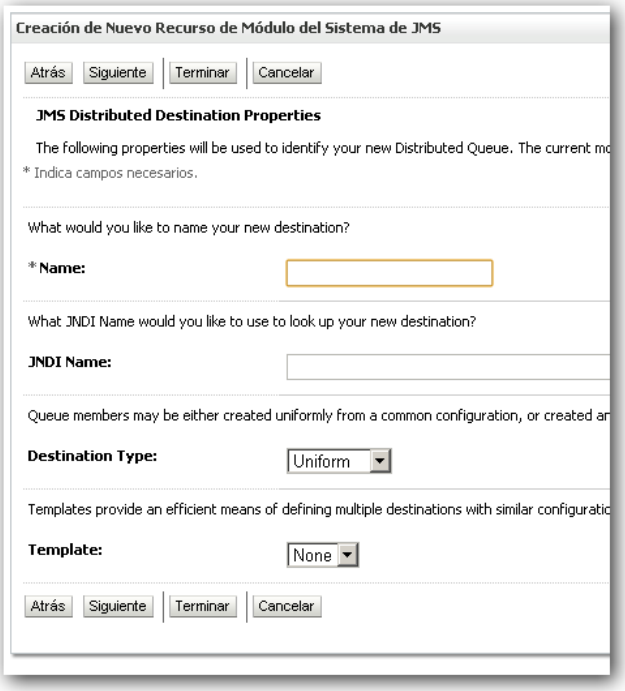



***Figura 31. Datos requeridos para la creación de la cola de error***

Fuente: Elaboración propia

En la figura 32 seleccionamos la opción “Direccionamiento avanzado”, dentro de la pestaña se encuentran los nodos del servidor, en este caso se puede apreciar que hay dos nodos disponibles donde se podrá revisar el tráfico de los mensajes.

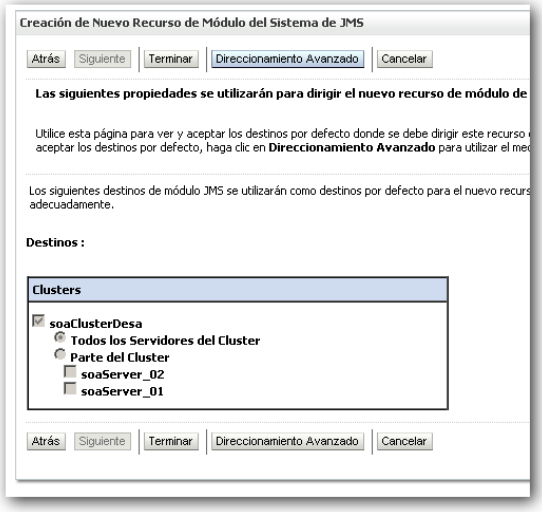



***Figura 32. Creación de la Cola de Error***

Fuente: Elaboración propia

La figura 33 es una ventana de confirmación una vez la cola se haya creado y configurado correctamente y es necesario activar los cambios para guarda.

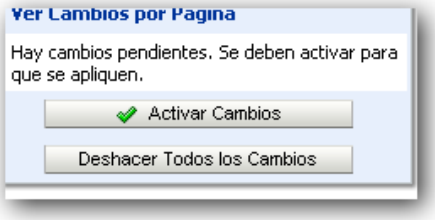



***Figura 33. Confirmación de la configuración***

Fuente: Elaboración propia

c) Creación de la cola de reintentos (pe.com.claro.jndi.validaAprovision.jms.queue)

La cola de reintentos contralará las veces que será necesario volver a procesar una solicitud. En la figura 34 se selecciona el Modulo JMS para proceder a configurar la cola de reintentos.

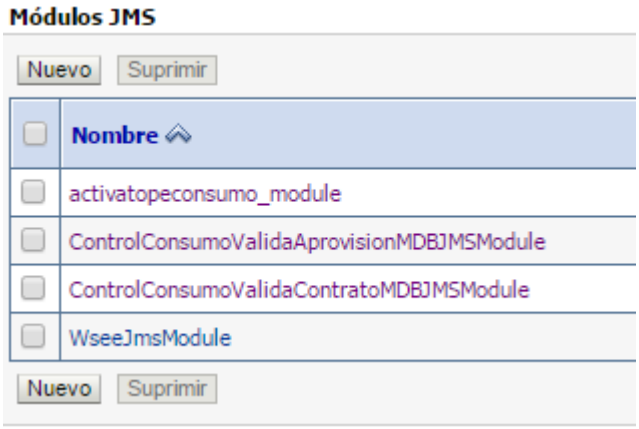



***Figura 34. Configuración de Reintentos de la cola***

Fuente: Elaboración propia

La figura 35 muestra los valores que serán necesario asignar para los reintentos, tales como la cantidad de reintentos, el tiempo que debe tardar la ejecución de uno de ellos, etc.

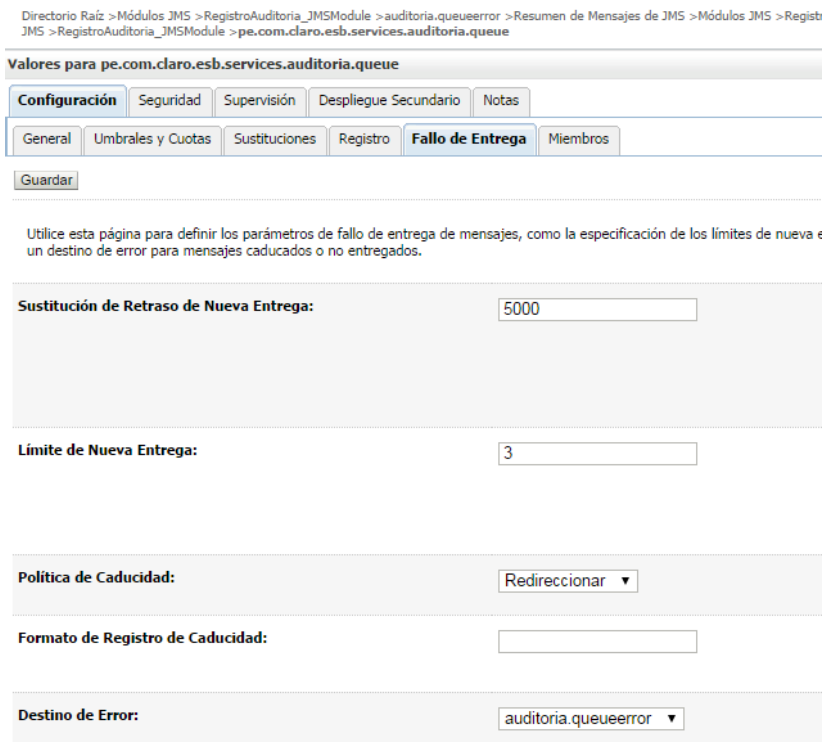



***Figura 35. Configuración de Reintentos***

Fuente: Elaboración propia

Estos datos deben ser coordinados con los interesados en el proyecto.

Finalizamos en la figura 36 la confirmación de los cambios realizados.

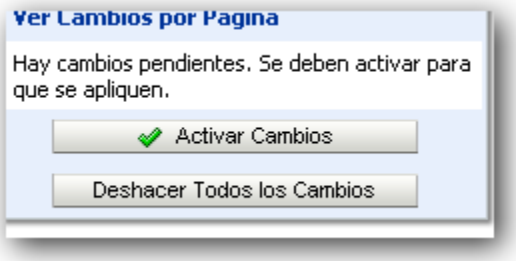



***Figura 36. Confirmación de configuración de reintentos***

Fuente: Elaboración propia

El detalle de la creación de la cola de reintentos se encuentra en el Anexo E.

### 3.1.3.2. Codificación del sistema

La codificación es el paso para generar código de los componentes propuestos que componen la mejora del proceso, se desarrolla procedimientos de operación y seguridad con el objetivo de asegurar el correcto funcionamiento del sistema. Este punto depende del diseño que se plantea y debe desarrollar tal y como éste lo indica.

La Figura 37 muestra la codificación de la clase java *ActivarDesactivarControlConsumoRequest* que contiene las variables o parámetros que son necesarios para ejecutar el servicio.

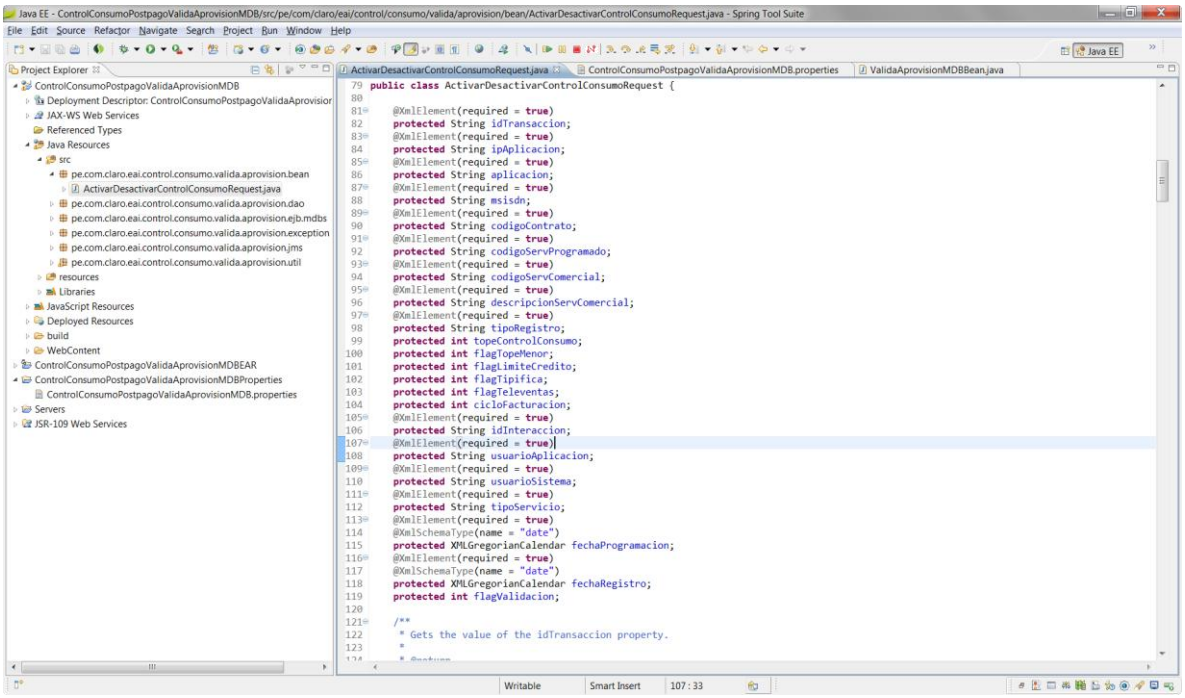



***Figura 37. Codificación de la clase ActivarDesactivarControlConsumoRequest***

Fuente: Elaboración propia

La figura 38 muestra la codificacion de la clase *BscsDAO* el cual contiene la declaración del método *validarDesactivacionTopeConsumo* y los parámetros que éste recibirá.

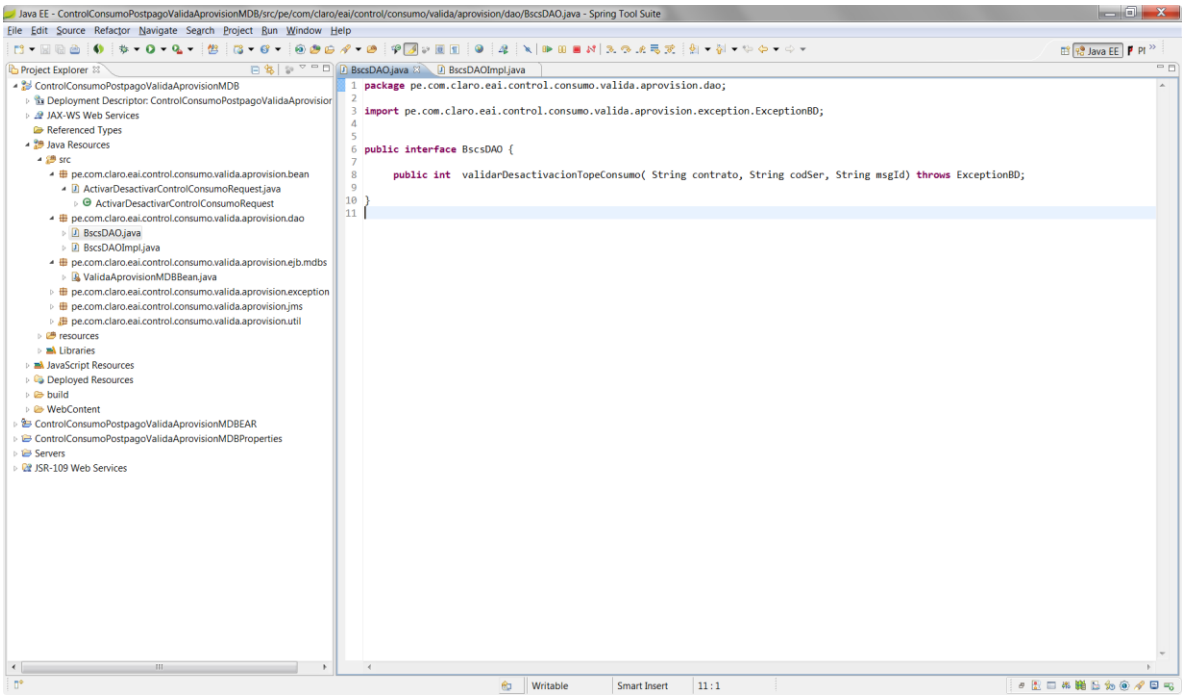



***Figura 38. Codificación de la clase BscsDao***

Fuente: Elaboración propia

A continuación la figura 39 muestra la implementación de la clase creada anteriormente, en ésta clase va la conexión a la base de datos y la llamada a los procedimientos almacenados que el servicio necesite consumir. Esta clase hereda el método *validarDesactivacionTopeConsumo* de la case *BscsDAO.*

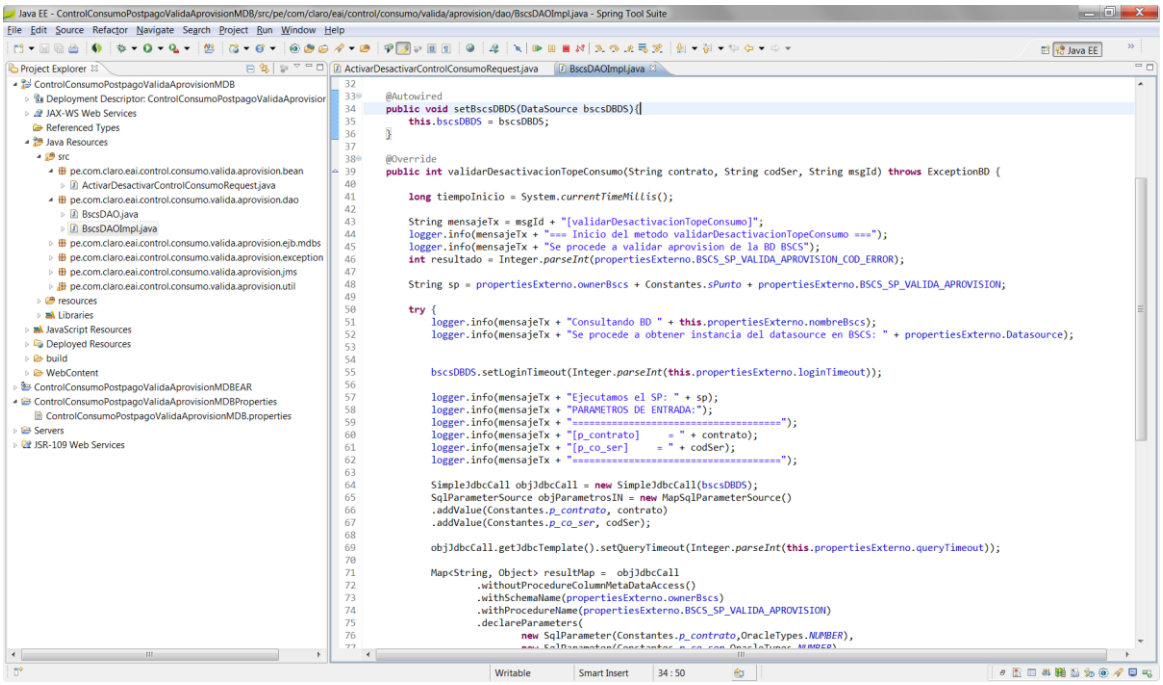



***Figura 39. Codificación de la clase BscsDAOImpl***

Fuente: Elaboración propia

La figura 40 y 41 contienen codificación de todos los valores de las constantes, variables, datos que son necesarios para conetarse al servidor de aplicaciones, base de datos, llamadas a otros servicios, etc y valores que deben ser creados en este archivo para no introducir codigo en duro en la programación de los componentes que formen parte de la lógica del negocio.

Es importante señalar que para proteger los datos de la empresa se están poniendo asteriscos en lugar de las direcciones IP y claves o usuarios, ya que estos son datos confidenciales.

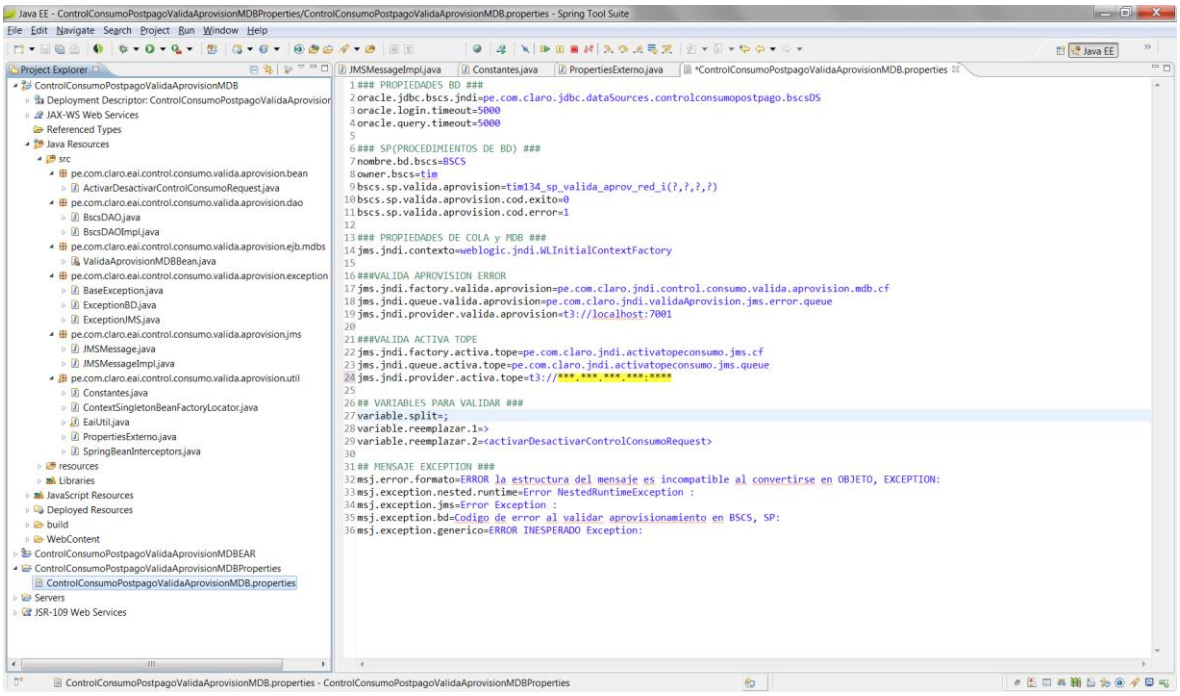



***Figura 40. Codificación del archivo Properties***

Fuente: Elaboración propia

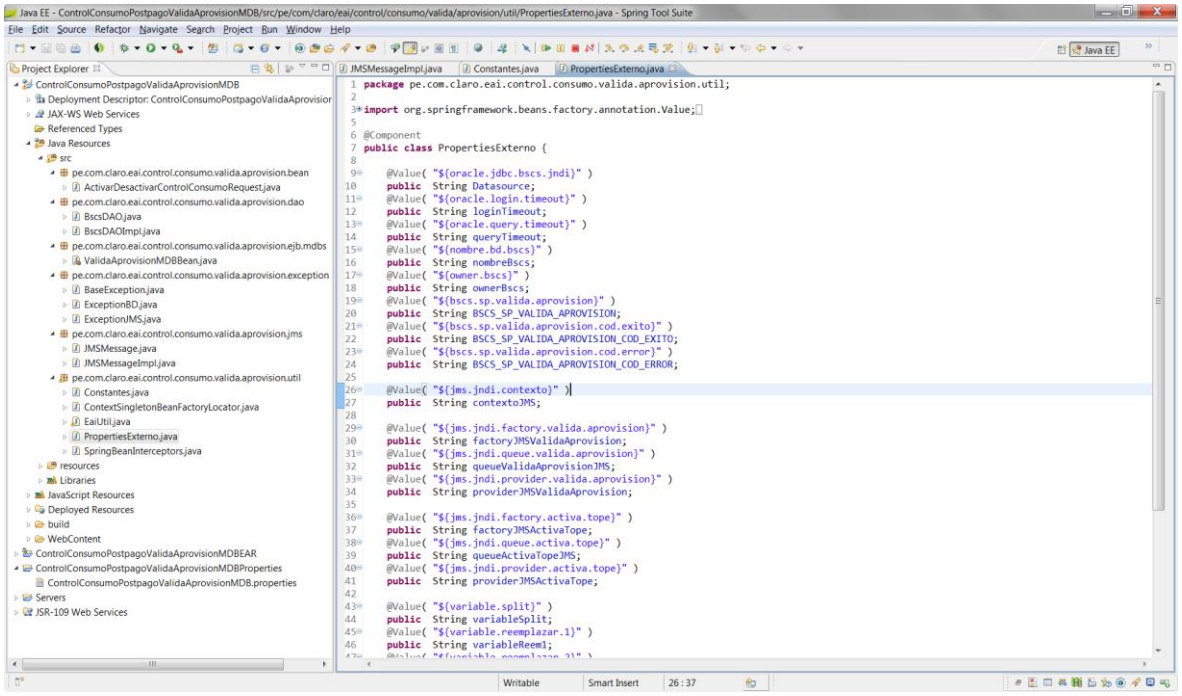



***Figura 41. Codificación del archivo propertiesExterno***

Fuente: Elaboración propia

La figura 42 muestra la codificación al servicio JMS quien es el encargado de comunicar el servicio principal con la cola (queue donde se guardarán todos los mensajes que tienen errores para volverse a procesar) que está creada y configurada en el servidor *WebLogic*.

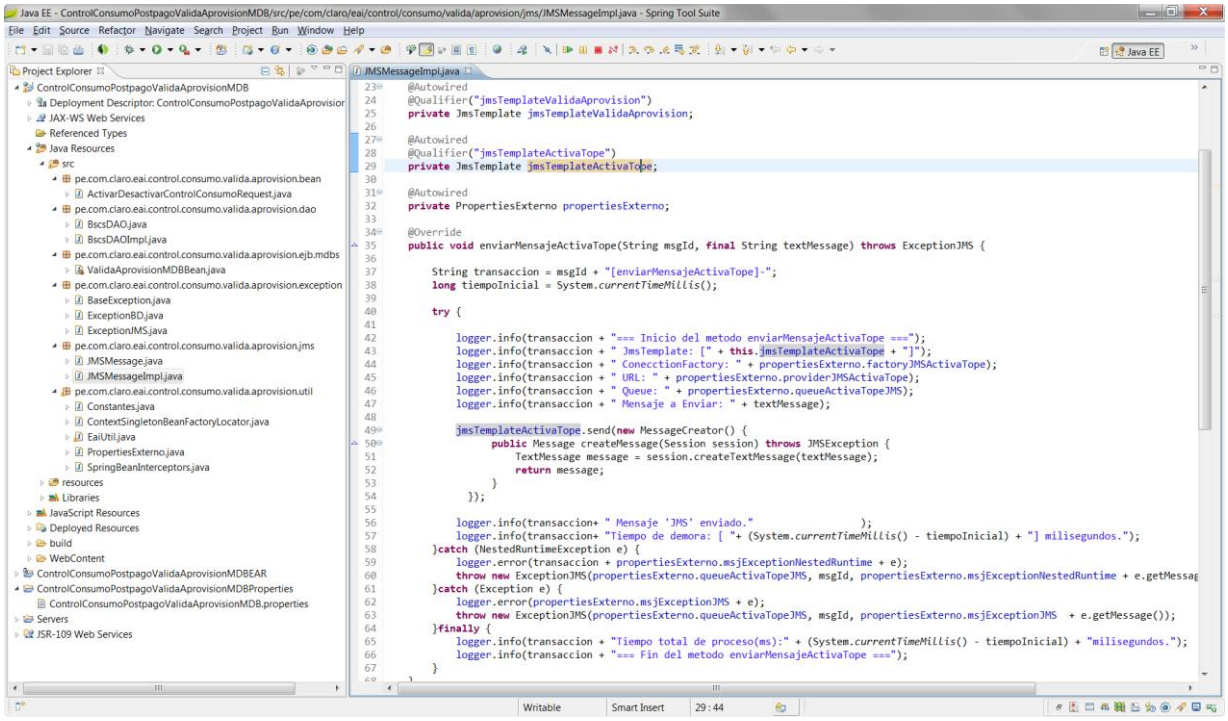



***Figura 42. Codificación de la clase JMSMessageImpl***

Fuente: Elaboración propia

La siguiente figura 43 muestra la clase principal ValidaAprovisionMDBBean que contiene el método *onMessage* del servicio, es la clase que contiene todo el proceso de negocio descrito en el cuadro de especificación líneas arriba. Para su correcto

funcionamiento éste servicio necesita de las clases y archivos creados previamente y que son las figuras anteriores.

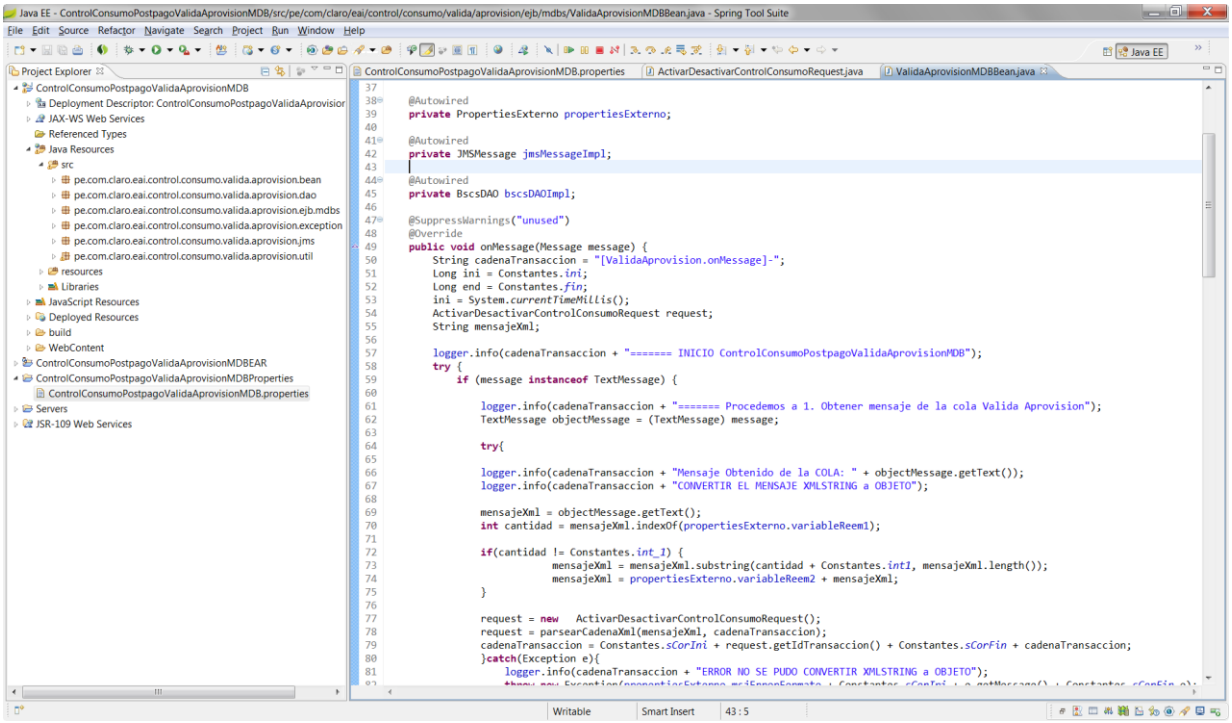



***Figura 43. Codificación de la clase ValidaAprovisionMDBBean***

Fuente: Elaboración propia

**Codificación de Shell**

Una vez se concluya con la codificación del servicio que se encargará de volver a procesar las peticiones o solicitudes que han tenido errores se procede a la creación de la Shell, que realizará la tarea de pasar automáticamente un lote de mensajes de una cola a otra, esta ejecución se da cada cierto tiempo, según hayan sido programadas por el usuario.

Para la creación de la Shell primero se deben crear las carpetas que contendrán los archivos que se generan cuando se empiece a desarrollarlo. Este procedimiento se observa en la figura 44.

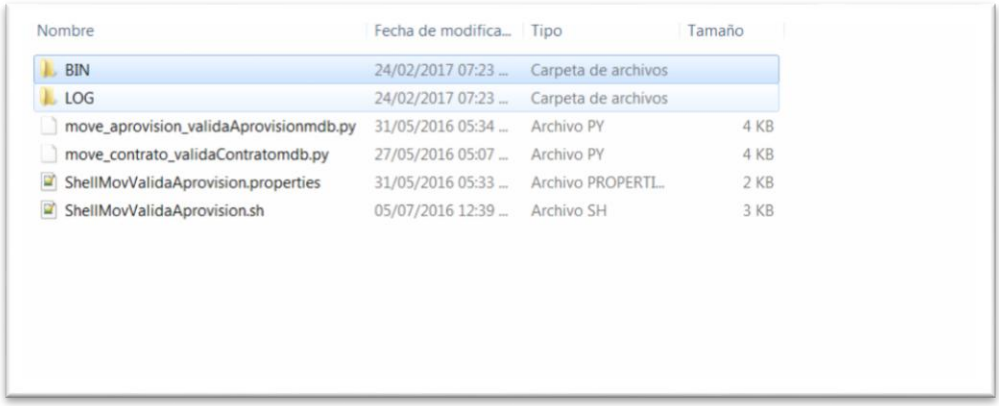



***Figura 44. Creación del Directorio necesario para la Shell***

Fuente: Elaboración propia

Como los procedimientos anteriores, la Shell también necesita de un archivo .properties que contenga los datos de las colas y los servicios con los que se comunicará. La figura 45 muestra la codificación del archivo properties. Por cuestiones de seguridad y

confidencialidad de datos se están bloqueando los usuarios y direcciones IP.

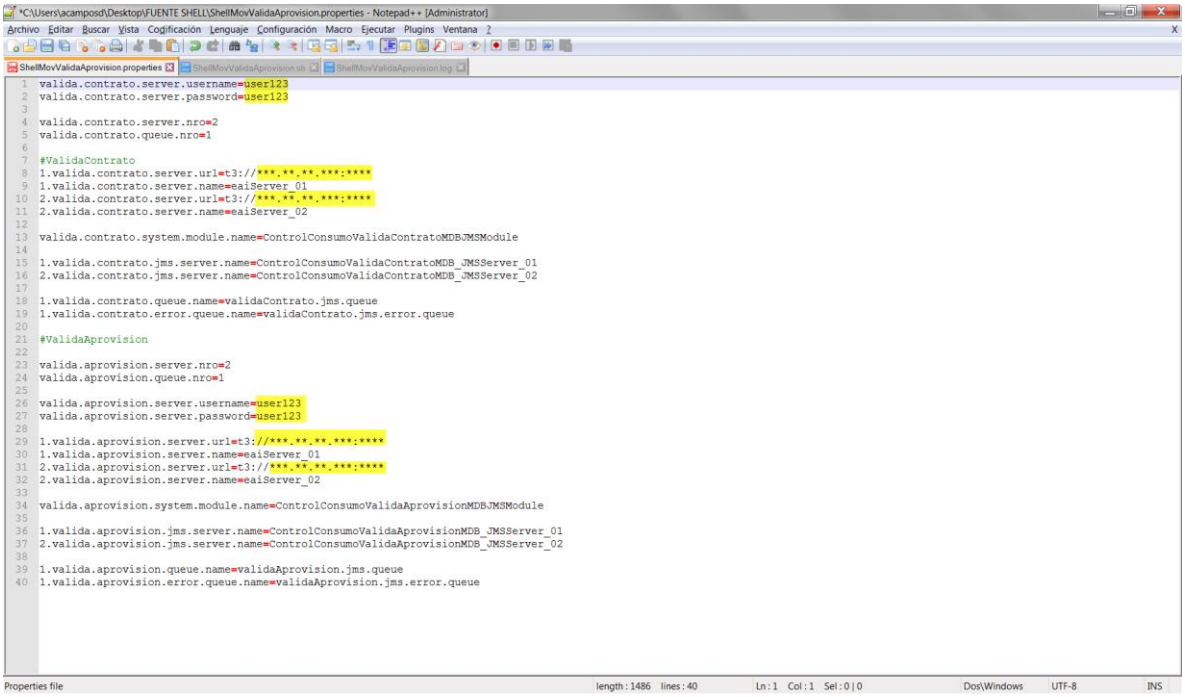



***Figura 45. Codificación de properties del servicio ShellMovValidaAprovisión***

Fuente: Elaboración propia

Una vez creado el archivo properties pasamos a crear el archivo que contendrá toda la lógica de la Shell, este archivo hará una invocación a otro archivo creado anteriormente *ShellMovValidaAprovision.properties***,** la figura 46 muestra el detalle de la codificación del archivo.

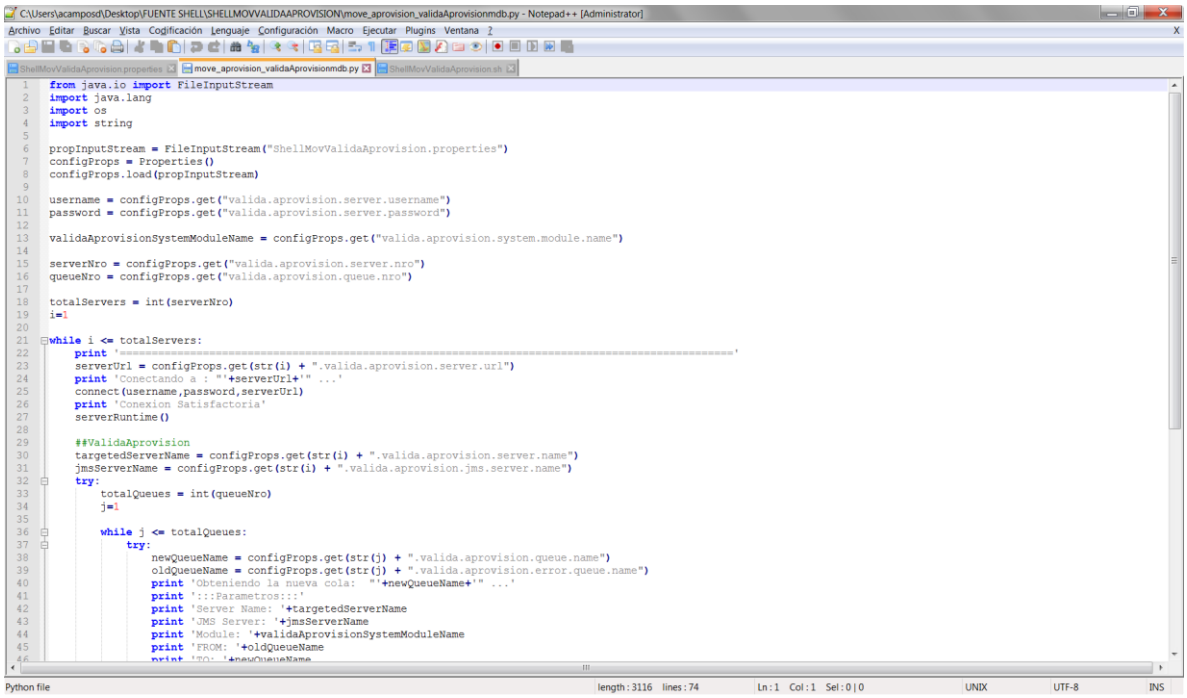



***Figura 46. Codificación del archivo move_aprovision_validaAprovisionmdb.py***

Fuente: Elaboración propia

Para finalizar con el proceso de la creación de la Shell, la figura 47 muestra la codificación del archivo ***ShellMovValidaAprovision.sh*** el cual se ejecutara cada cierto tiempo según sea configurado en el servidor para realizar alguna tarea en específico, y en este caso la tarea de mover los mensajes de la cola de errores a la cola de reprocesos.

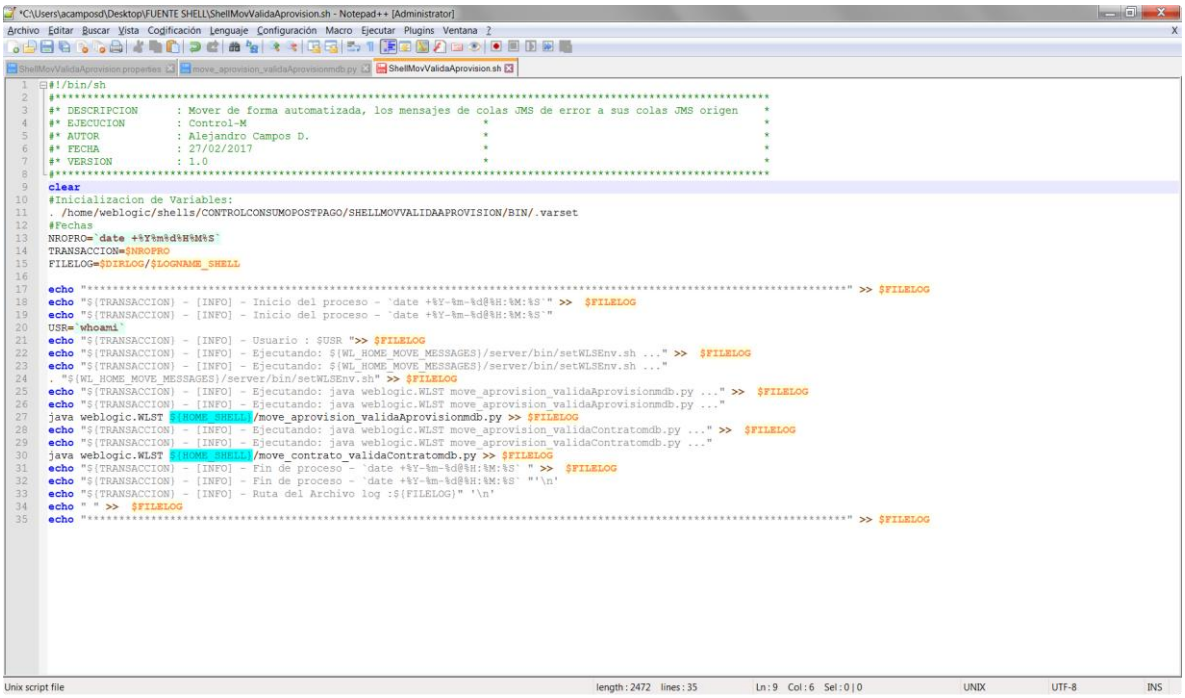



***Figura 47. Codificación del archivo ShellMovValidaAprovision.sh***

Fuente: Elaboración propia

## Codificación tabla temporal TBL_TMP_TRANS_TOPECONSUMOS

Creación de la tabla temporal TBL_TMP_TRANS_TOPECONSUMOS en la base de datos BSCS para lo cual usaremos el script que se muestra en la siguiente figura 48.

```
CREATE TABLE TBL_TMP_TRANS_TOPECONSUMOS
(
        ID_TRANSACCION VARCHAR2 (50) NULL,
        CO_ID INTEGER NOT NULL,
        CREATION_DATE DATE NOT NULL,
        SESSIONRUN_ID INTEGER NOT NULL
);
```

***Figura 48. Script para la creación de la Tabla Temporal***

Fuente: Elaboración propia

A continuación la figura 49 muestra la tabla temporal TBL_TMP_TRANS_TOPECONSUMOS creada y lista para poder realizar las transacciones con el fin de evitar los reprocesos innecesarios.

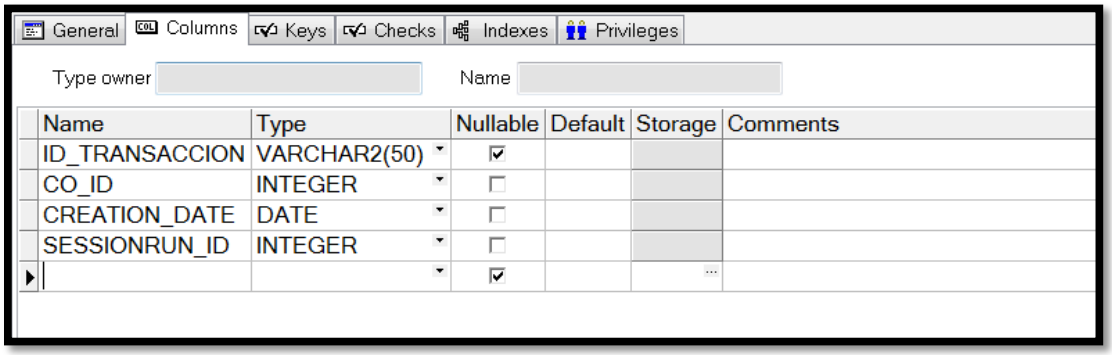



***Figura 49. Detalle de la creación de la tabla temporal***

Fuente: Elaboración propia

En este punto es necesario agregar la nueva variable P_ID_TRANSACCION al procedimiento almacenado REGISTRA_CTRL_CONSUMO para que se pueda realizar la transacción con la tabla nueva creada (TBL_TMP_TRANS_TOPECONSUMOS) y la tabla que contiene los registros a procesar. Las validaciones se realizaron en el SP como se había propuesto y de ésta manera controlaremos los reprocesos innecesarios.

En la figura 50 se observa el nuevo parámetro P_ID_TRANSACCION para el procedimiento almacenado.

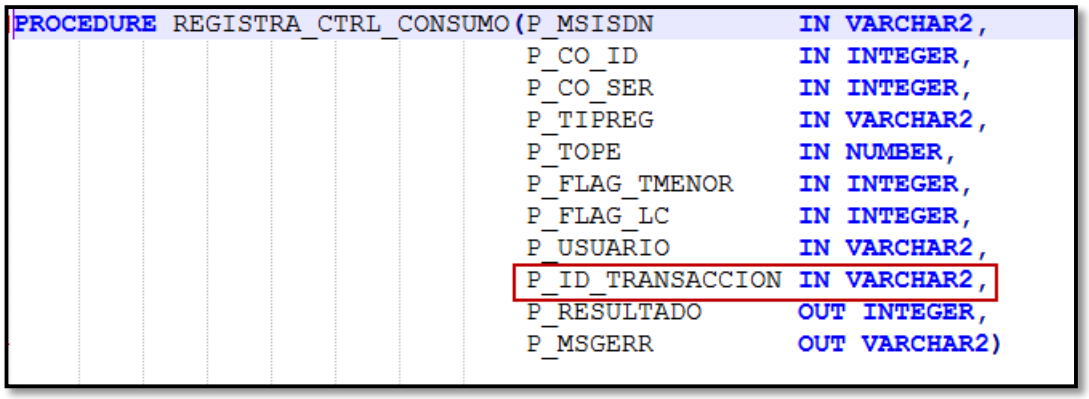



***Figura 50. Agregar parámetro al SP REGISTRA_CTRL_CONSUMO***

Fuente: Elaboración propia

### 3.1.4. Pruebas

#### 3.1.4.1. Pruebas del proceso de Cambio de Plan

Para que el proceso de Cambio de Plan se ejecute con correctamente según lo planificado, es decir se realice el mismo día que la activación del tope de consumo se requiere ingresar las mismas fechas en los campos *fechaProgramacionTope* y *fechaProgramacion* (ambos deben ser lo mismo) del *request*.

Este proceso incluye la ejecución de varios servicios, incluso de algunos que no están mencionados en el proyecto, pero de igual forma validamos que se realicen correctamente.

En la figura 51 se ingresa la solicitud del cambio de plan a procesar. Tanto *fechaProgramacionTope* y *fechaProgramacion* deben ser iguales.

La variable *ipAplicacion* no es una dirección real por ser datos confidenciales.

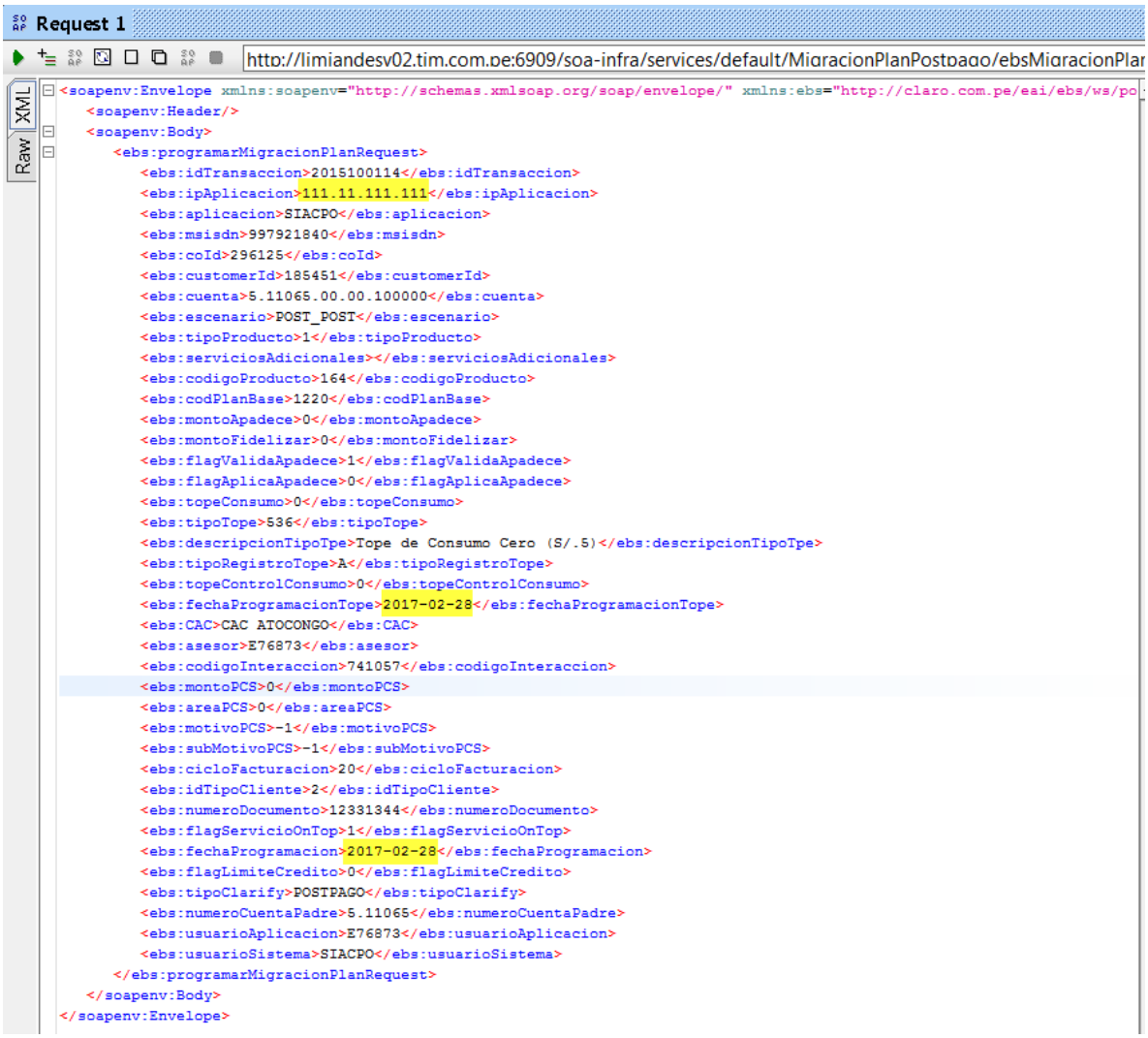



***Figura 51. Request del Servicio MigraciónPlanPostpago***

Fuente: Elaboración propia

En la figura 52 se observa que el registro de la migracion con la misma fecha se realizó de forma correcta.

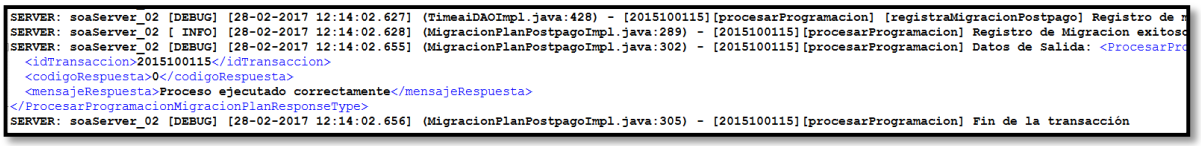



***Figura 52. Confirmación del registro de programación del Cambio de Plan exitoso***

Fuente: Elaboración propia

El proceso de cambio de plan involucra varios componentes que no se han visto en el proyecto, por lo tanto solo serán nombrados ya que al realizar las pruebas las peticiones pasan a traves de ellos.

Una ves registrada el Cambio de Plan para una fecha determinada, esa misma solicitud se envía para el proceso de Cambio de Tope de Consumo, este procedimiento lo realiza un componente externo *BPEL*. En la figura 53 se puede apreciar que la programación del tope de consumo también se registró y se envió a la cola para su ejecución correctamente en base a la programación del Cambio de Plan.

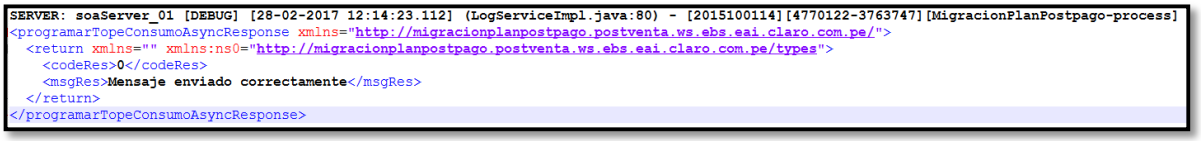



***Figura 53. Programación BPEL para el Cambio de Tope de Consumo enviado con éxito***

Fuente: Elaboración propia

Luego el componente que está a la escucha de lo que llegue a la cola, el MDB *migracionPlanPostpagoMDB* procesa la información

solicitada, la respuesta a dicho procedimiento se observa en la figura 54.

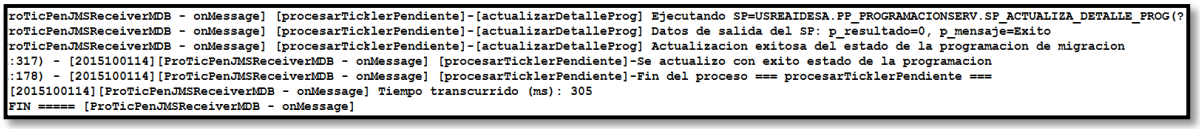



***Figura 54. Confirmación de la programación.***

Fuente: Elaboración propia

Una vez confirmada la programación de activación de tope de consumo, solo es necesario esperar a que llegue el día de la programación de Cambio de Plan, la Activación de Tope de Consumo se ejecutara inmediatamente después del cambio de plan. Estos procedimientos son realizados a nivel de aplicativos.

#### 3.1.4.2. Pruebas del servicio ControlConsumoPostpagoValidaAprovisionMDB

En este punto se procederá a revisar que los componentes implementados para la mejora del proceso de activación de topes de consumo responda correctamente a la peticiones del usuario según los parámetros ingresados.

En la figura 55 se muestra una captura del *request* (petición), donde se ingresan los datos necesarios para que el servicio valide el estado de la línea, que esté aprovisionado correctamente, que no haya errores en las validaciones, etc.

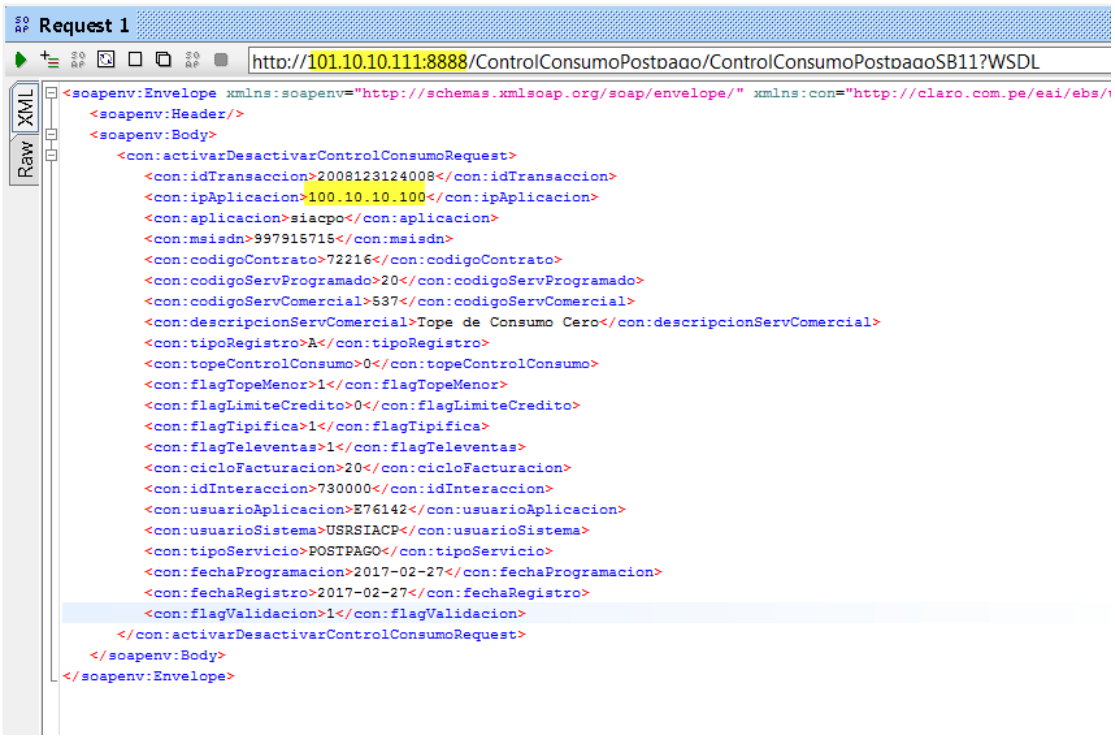



***Figura 55. Request del servicio ControlConsumoPostpagoValidaAprovisionMDB***

Fuente: Elaboración propia

En la figura 56 se muestra el *response* es decir la respuesta según el *request* ingresado previamente en la figura 55. En esta captura se muestra un codigo y mensaje donde se describe si el procedimiento se realizo correctamente.

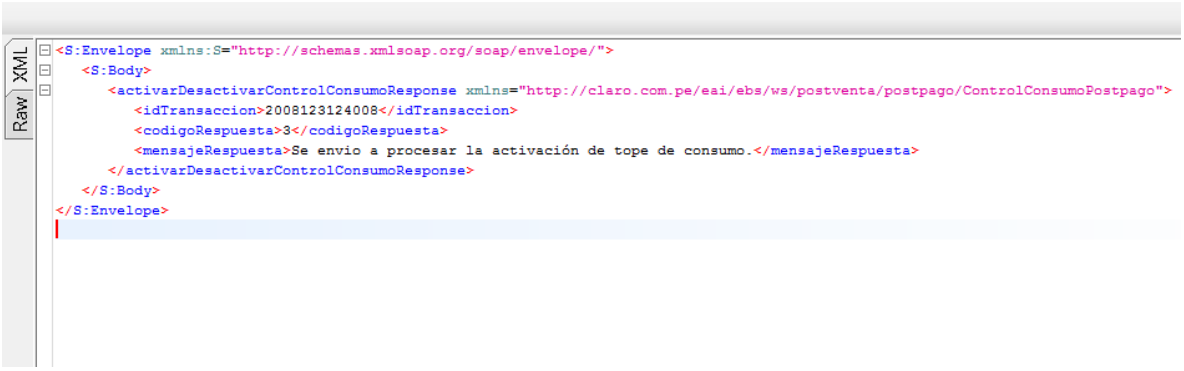



***Figura 56. Confirmación de que el envío se realizó correctamente***

Fuente: Elaboración propia

#### 3.1.4.3. Prueba del servicio ShellMovValidaAprovisión

La Shell ShellMovValidaAprovisión se ejecutará automáticamente, cuando llegue el tiempo de ejecución que le fue programado y pasará los mensajes de la cola de error a la cola de reintentos en lotes de 200.

En la figura 57 se aprecia los parámetros de entrada que recibe el servicio para hacer la migración de mensajes de la cola origen a la cola destino. El color amarillo es para indicar que son datos cualquiera no son datos verdaderos ya que por cuestiones de confidencialidad estos no deben ser expuestos.

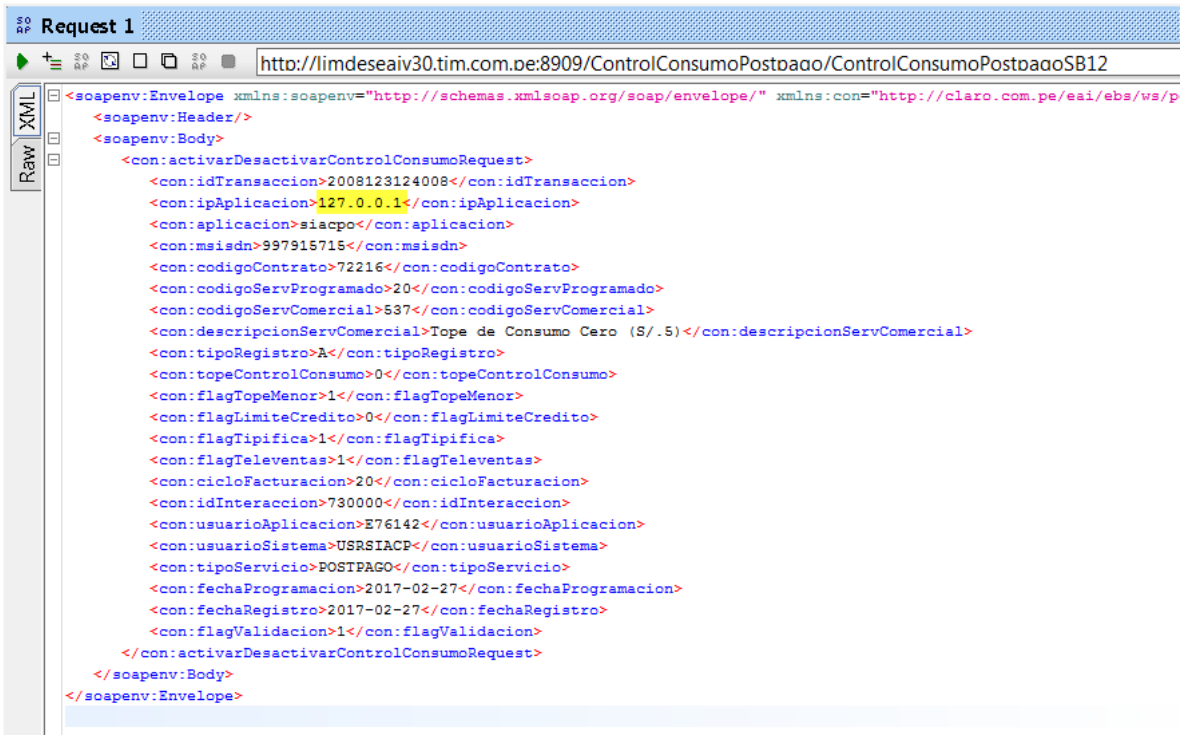



***Figura 57. Request del servicio ControlConsumoPostpagoWS para invocar a la Shell ShellMovValidaAprovision***

Fuente: Elaboración propia

En la figura 58 se muestra el resultado de la prueba, y como se puede apreciar responde correctamente, es decir que el proceso de paso de una cola a otra se dio sin ningún problema.

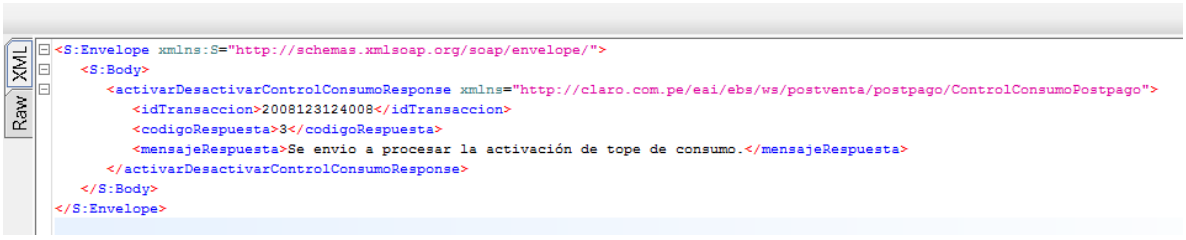



***Figura 58. Confirmación del traspaso de mensajes de cola de error a la cola de reintentos***

Fuente: Elaboración propia

En la figura 59 se puede observar el log de la Shell donde se aprecia que los mensajes de la cola de origen (**pe.com.claro.jndi.validaAprovision.jms.error.queue**) a la cola destino (**pe.com.claro.jndi.validaAprovision.jms.queue**) tal y como se había planeado.

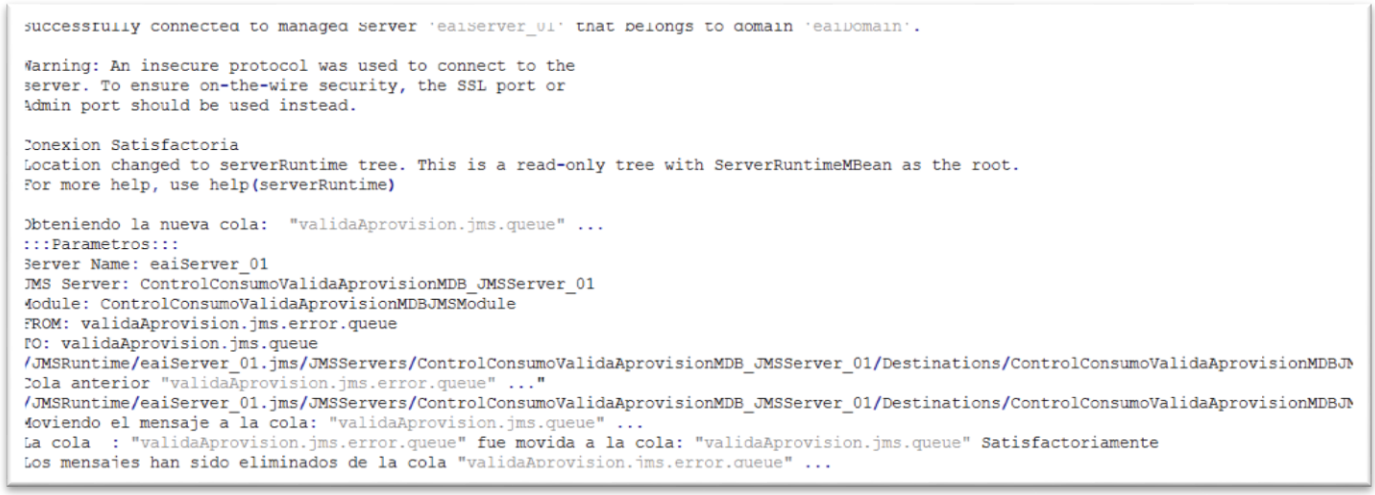



***Figura 59. Log de confirmación del traspaso de mensajes realizado por ShellMovValidaAprovisión***

Fuente: Elaboración propia

Tras las pruebas realizadas a los componentes propuestos en el proyecto podemos afirmar en líneas generales que se ha cumplido con lo planteado para dar solución al problema que aquejaba al proceso de Aprovisionamiento en Janus y por consecuencia al proceso principal (Migración), conllevando a que éste termine con errores y no se cumplan los objetivos.

### 3.1.5. Verificación

En este apartado verificamos que el sistema en general funcione correctamente con los cambios realizados con la propuesta. Validar esto incluye manejar los aplicativos SIACPOSTPAGO y el SISACT, ver que cumplan con los requerimientos, probar esto ello le corresponde a otra área ya que como se había indicado anteriormente en los capítulos previos las respuestas de la mejora son a nivel de servicios, es decir que éstos se ejecuten correctamente. Por otro lado si los servicios están funcionando sin problema el aplicativo funciona de la misma manera ya que estos solo hacen llamadas a los servicios para determinadas tareas y si los servicios funcionan bien no habría problemas al ejecutar el sistema a nivel de aplicativos.

### 3.1.6. Mantenimiento

En la etapa de prueba se validó que los servicios nuevos se ejecutaron correctamente en el ambiente de desarrollo en el cual casi siempre funcionan sin problemas, el punto más importante es que funcionen igual en un ambiente de producción y en algunos casos en éste ambiente surgen nuevas casuísticas, como caídas del servicio por algún problema técnico, que el ambiente de desarrollo y producción tengan configuraciones diferentes que implica que haya conflicto con algún otro componente, o que haya una funcionalidad extra que no se tomó en cuenta en la etapa de análisis. Por todo ello

es necesario estar al pendiente mientras se hacen las pruebas en el ambiente de producción con datos reales y por si fuera necesario dar soporte a alguna incidencia que aparezca.

## 3.5. REVISIÓN Y CONSOLIDACIÓN DE RESULTADOS

En el presente apartado se revisará todo lo relacionado al flujo antes y después de la mejora del proceso de Migración, en especial el proceso de validar Aprovisionamiento (proceso para la activación de los topes de consumo), los tiempos de respuesta luego de implementar la propuesta y el correcto funcionamiento de los componentes que forman parte del proceso para realizar el Cambio de Plan y la Activación de los Topes de Consumo sin problemas.

Como se mencionó en líneas anteriores la metodología empleada para el proyecto es el modelo Cascada, basado en una arquitectura orientada a servicios SOA, el cual con el apoyo de algunas herramientas de prueba como el SoapUI nos permitirá visualizar las respuestas de los servicio, también es necesario un editor de texto para poder ver el log (traza o evidencia) de la respuesta, es decir el detalle de cómo se ha ejecutado y si se cayó el servicio debido a qué fue, pinta todo lo que se ejecuta en background.

Si bien es cierto lo componentes se han desarrollado según lo propuesto, para la evaluación de los resultados se van a realizar las pruebas a nivel de servicio, es decir se harán pruebas tanto al MDB

como a la Shell componentes nuevos creados y no a las aplicaciones que consumen estos servicios. Nos basamos en la arquitectura SOA, así que si los servicios responden de forma óptima con tiempos de respuesta que estén dentro de lo permitido las aplicaciones también funcionaran de la misma manera. Por otro lado a modo de justificación, el proyecto se realiza en el área de integración, en esta área solo tenemos acceso a los servicios y no a los aplicativos que los centro de atención al cliente manejan, solo presentamos los resultados a nivel de servicios y la llamada a estos los realizan otras áreas que hacen la llamadas según corresponda.

A continuación en la figura 60 se muestra el flujo completo, mejorado del proceso migración, en el que se puede apreciar los componentes que interactúan en el proceso. En la figura hay varios procesos involucrados pero para el proyecto en especial se ha tomado el proceso de ***Validación de Aprovisión Janus*** que está de color celeste en la leyenda de la imagen.

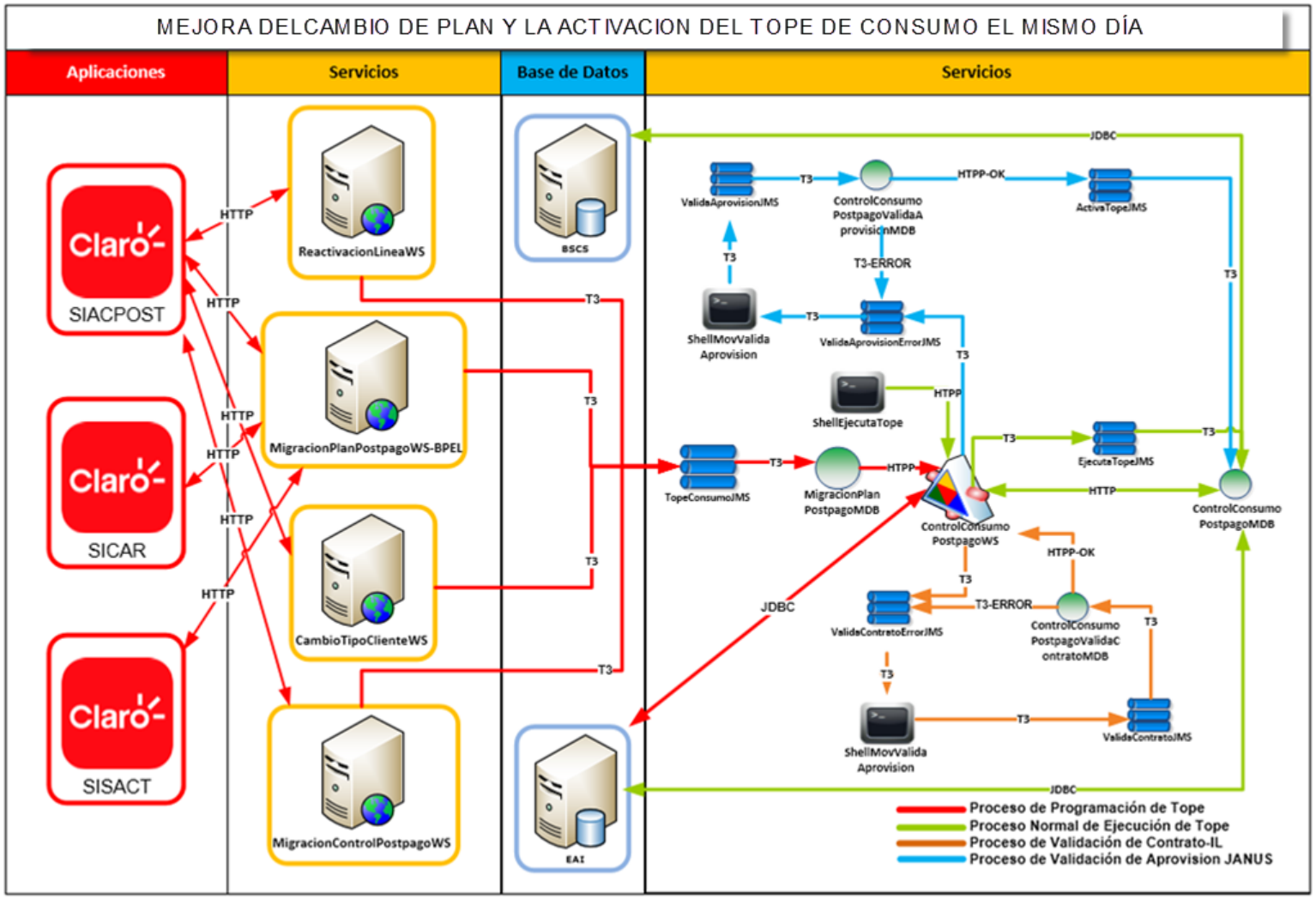



***Figura 60. Flujo completo del proceso de Migración***

Fuente: elaboración propia

### 3.5.1. Resultados de la mejora del proceso “Cambio de Plan”

#### 3.5.1.1. Antes de la mejora del proceso

Antes de la propuesta de mejora el Cambio de Plan tardaba aproximadamente un día en ser atendido, éste tiempo era tomado para que el proceso pueda terminar todo completamente, es decir si salía algún error había un día para corregirlo. Esta era la razón principal por la que las fechas de cambio de plan y activación de topes de consumo eran diferentes. Incluso si el cliente deseaba hacer el cambio de plan el mismo día se le indicaba que no era posible ya que antes debía terminar todas las solicitudes que pasaron antes lo cual tardaba bastante ya que se ejecutaban de forma masiva.

#### 3.5.1.2. Después de la mejora

La mejora del proceso no consiste en modificar el flujo actual, sino agregar un adicional a lo que ya existe y en esta mejora se propone realizar el Cambio de Plan y la Activación del tope de consumo el mismo día como también que el cambio de plan pueda darse el mismo día que se desea. Para ello las fechas de programación tanto del

Cambio de Plan y la Activación del Tope de Consumo deben ser las mismas. Como el proyecto es a nivel de servicio no se tiene acceso a los aplicativos usados en los centro de atención al cliente solo haremos una referencia de ello. En la figura 61 se observa que el registro de ambas fechas deben ser las mismas debe haber constancia de ello.

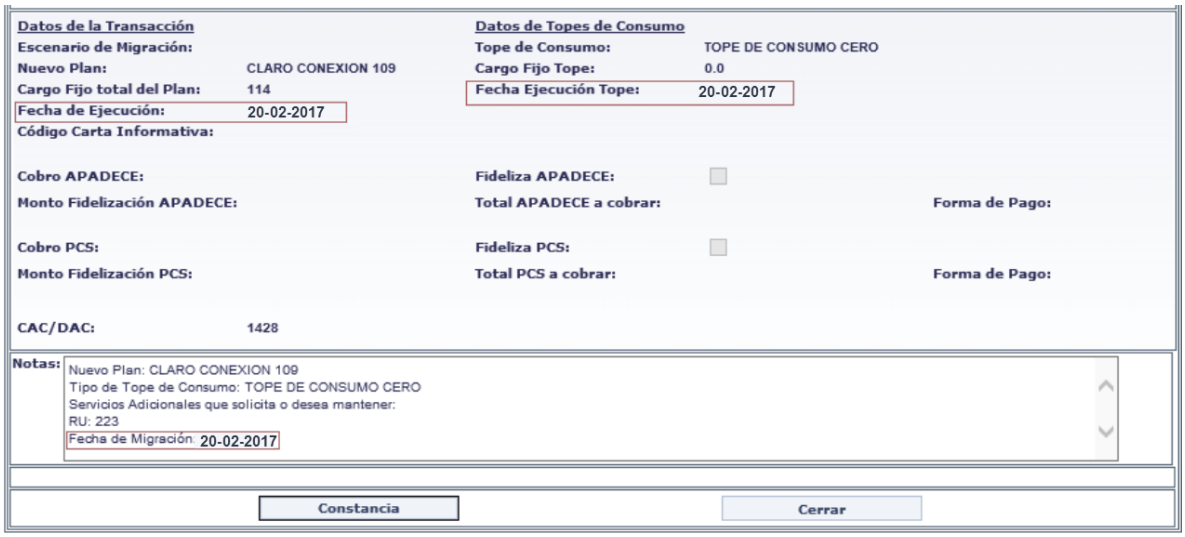



***Figura 61. Validación de fechas de Cambio de Tope y Activación de Topes de Consumo***

Fuente: Elaboración directa

### 3.5.2. Resultados de la mejora del proceso “Cambio del Tope de Consumo”

#### 3.5.2.1. Antes del proceso de mejora

Antes de la mejora del proceso que hacía la validación del Aprovisionamiento en Janus (plataforma de servicios con múltiples funciones) el proceso se desarrolla tal y como lo muestra la figura 62 donde se puede apreciar cómo era el flujo que se seguía y porqué salía error en la activación del Tope de Consumo.

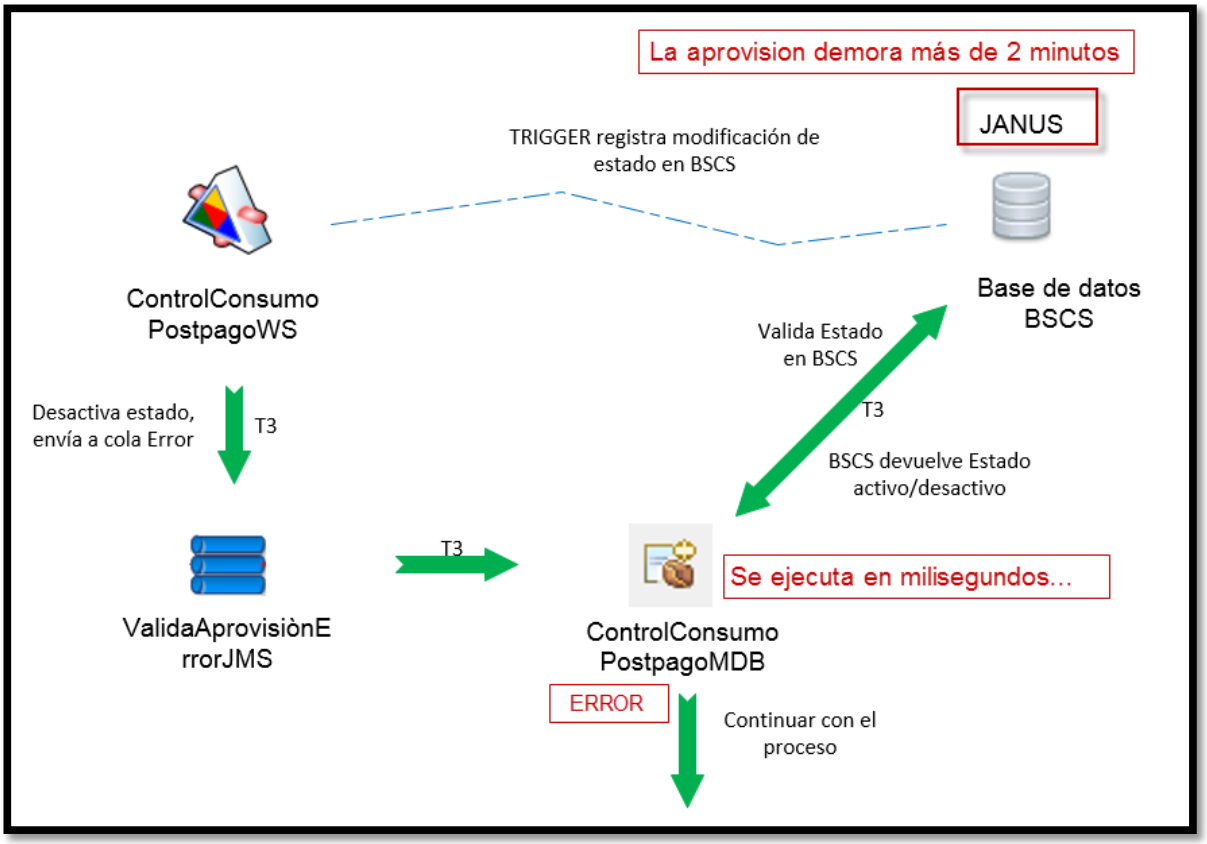



***Figura 62. Proceso de Aprovisionamiento antes de la mejora***

Fuente: Elaboración Propia

### 3.5.2.2. Después del proceso de mejora

La mejora del proceso de Migración (Figura 63) permite realizar el **Aprovisionamiento Janus** de manera más óptima, ya que el proceso se realiza por separado del flujo principal manejando los mensajes de una forma en particular por colas, es decir si hay errores en el aprovisionamiento lo que hace el servicio es volverlo a encolar en la cola de error para

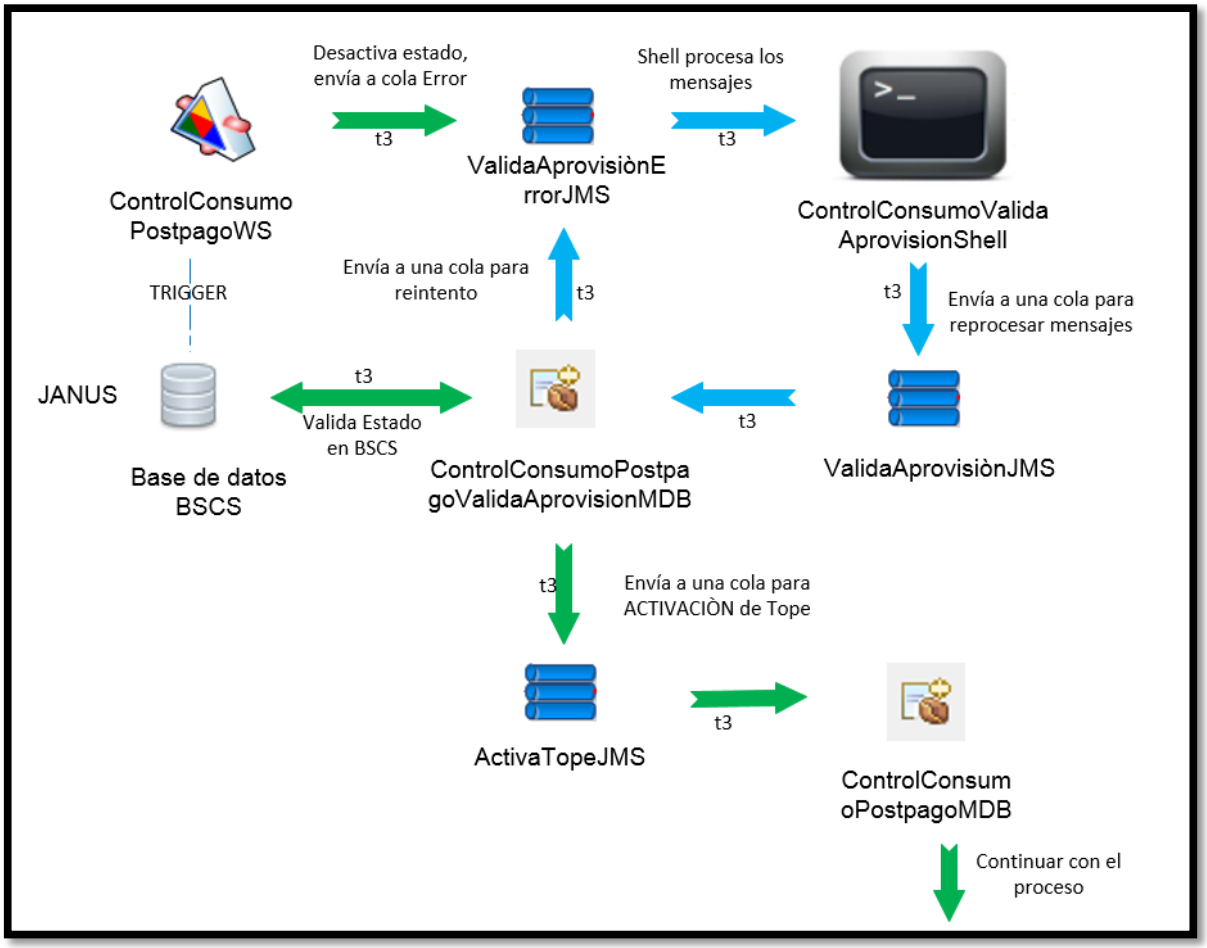



***Figura 63. Proceso de Aprovisionamiento después de la mejora***

Fuente: Elaboración propia

que posteriormente se mueva a la cola de reintentos y se vuelva a procesar.

3.5.2.3. Cuadro comparativo del antes y después de la mejora del proceso de aprovisionamiento

Luego de la implementación de la propuesta para la mejora del procedo activación de topes de consumo y en específico del proceso de aprovisionamiento se obtuvieron los siguientes resultados mostrados en la tabla 14.

**Tabla 14.** *Cuadro comparativo del antes y después del proceso de activación de tope de consumo.*

| ANTES DE LA MEJORA | DESPUÉS DE LA MEJORA |
|---|---|
| ✓ Activaciones de topes de consumo masivo una vez al día.<br>✓ Saturación del servidor por los reprocesos no controlados.<br>✓ Tiempos de respuesta fuera de lo permitido. | ✓ Activaciones de topes de consumo hasta tres veces al día. Los mensajes son atendidos por lotes.<br>✓ Ejecución exitosa al requerir reprocesos de solicitudes. |

| | |
|---|---|
| ✓ Para culminar una tarea con error era necesario realizarla de forma manual.<br>✓ Mal funcionamiento de algunos componentes producto de los errores de otros ejecutados previamente. | ✓ Tiempos de respuesta manejables por parte de los servicios.<br>✓ Los procesos se hacen en automático.<br>✓ El desarrollo del proceso es fluido, no hay estancamiento en el proceso por caídas de componentes. |

Fuente: Elaboración propia

### 3.5.3. Logros alcanzados con la propuesta

En la tabla 15 se observa lo que la propuesta permite alcanzar con su implementación.

**Tabla 15.** *Logros alcanzados con la propuesta*

| **LOGROS** |
|---|
| ✓ Registrar el cambio de plan y la activación del tope de consumo, para que ambos se ejecuten el mismo día.<br>✓ Obtener siempre una respuesta para realizar un mapeo y ver donde se está cayendo determinado proceso. |

- ✓ Los procesos se realizan de forma automática.
- ✓ El proceso de aprovisionamiento se realiza de una forma más óptima comparada con lo de antes de la mejora.
- ✓ Menos quejas por parte de los clientes afiliados a claro.
- ✓ Evitar pérdidas económicas que se genera por la creación de las notas de crédito.
- ✓ Reducir la tasa de migración de los clientes hacia otras empresas.
- ✓ Satisfacer los requerimientos del usuario.
- ✓ Satisfacer los requerimientos del cliente.

Fuente: Elaboración propia

RESULTADOS ESTADÍSTICOS

Con la implementación de la propuesta se ha conseguido solucionar el problema planteado, es decir se ha logrado solucionar la problemática del proceso de migración de líneas postpago en la empresa Claro. A continuación se muestra a grandes rasgos el logro conseguido con éste proyecto.

Cada vez que un cliente hace un reclamo o queja por sobrecargas de pagos justificadas Claro genera notas de crédito para solventar ello. Ahora, cada nota de crédito tiene un costo el cual es asumido por Claro en beneficio del cliente.

Valor en soles de cada Notas de Crédito: 62

Cantidad de reclamos (1 día): 284 (cantidad promedio)

Para la evaluación de los resultados tomaremos valores numéricos mensuales (30 días), por lo tanto en un mes tendríamos aproximadamente 8520 reclamos (284*30).

Tomando este dato como base se hace al análisis del antes y el después del proceso de mejora.

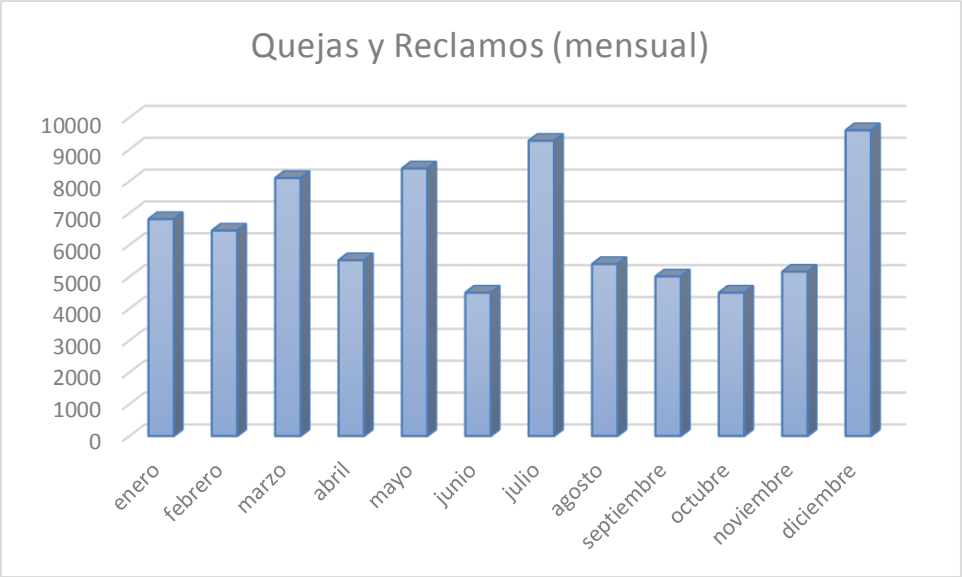
En la lista siguiente se observa la cantidad de reclamos aproximados al mes a lo largo del año 2016, para ello se han consultado a las personas interesadas y se ha podido establecer la siguiente lista. Cabe recalcar que son valores aproximados:

| Mes 2016 | Cantidad de reclamos (promedio) |
|---|---|
| Enero | 227 |
| Febrero | 215 |
| Marzo | 270 |
| Abril | 184 |
| Mayo | 280 |
| Junio | 150 |
| Julio | 309 |
| Agosto | 180 |
| Septiembre | 167 |
| Octubre | 150 |
| Noviembre | 172 |
| Diciembre | 320 |

En la tabla se observa la cantidad de reclamos mensuales respecto al año 2016.

| Mes | Reclamos |
|---|---|
| enero | 6810 |
| febrero | 6450 |
| marzo | 8100 |
| abril | 5520 |
| mayo | 8400 |
| junio | 4500 |
| julio | 9270 |
| agosto | 5400 |
| septiembre | 5010 |
| octubre | 4500 |
| noviembre | 5160 |
| diciembre | 9600 |

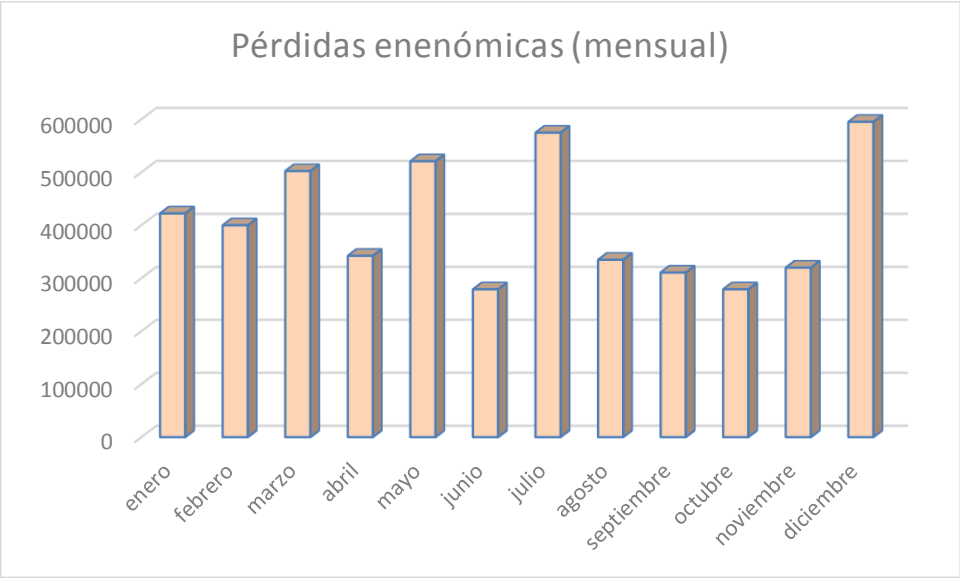
Los datos sombreados en la tabla informan los meses donde hubo más reclamos de los clientes, meses como mayo, julio y diciembre respectivamente. Las cantidades altas se deben a que esos meses Claro entra en el periodo de campaña, es decir es un periodo donde hay más solicitudes de cambio de plan, en la gráfica siguiente la explicación graficada.




En el siguiente cuadro se muestra las salidas (pérdidas) económicas mensuales de la empresa Claro producto del problema en el proceso de migración.

| Mes | Egresos (mes) |
|---|---|
| enero | 422220 |
| febrero | 399900 |
| marzo | 502200 |
| abril | 342240 |
| mayo | 520800 |
| junio | 279000 |
| julio | 574740 |
| agosto | 334800 |
| septiembre | 310620 |
| octubre | 279000 |
| noviembre | 319920 |
| diciembre | 595200 |

Del mismo modo se muestra el gráfico de barras donde se pueden apreciar los picos donde hubo más pérdidas según el mes que se haya evaluado.




INTERPRETACIÓN

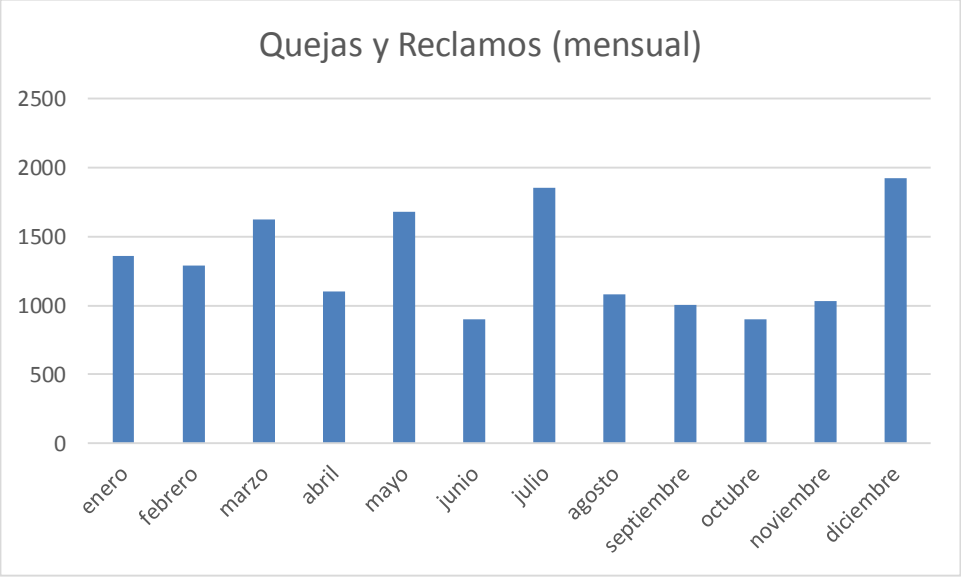
Según los números mostrados en los cuadros anteriores se puede observar que las pérdidas que tendría la empresa Claro dependiendo del mes son valores altos. Por ello se ha planteado esta propuesta, para reducir esos números en beneficio de la empresa mismo y del cliente, ya que son la razón de ser de la misma.

A continuación se muestra los valores o números después de la implementación del a propuesta. Si bien es cierto el proceso de migración todavía tendría problemas pero con la propuesta los reclamos se reducirían significativamente cerca de un 80%.

En el siguiente cuadro se muestra la cantidad de reclamos aproximados por mes con la reducción del 80% respecto a los datos antes de la mejora.

| Mes | Cantidad de reclamos |
|---|---|
| enero | 1362 |
| febrero | 1290 |
| marzo | 1620 |
| abril | 1104 |
| mayo | 1680 |
| junio | 900 |
| julio | 1854 |
| agosto | 1080 |
| septiembre | 1002 |
| octubre | 900 |
| noviembre | 1032 |
| diciembre | 1920 |

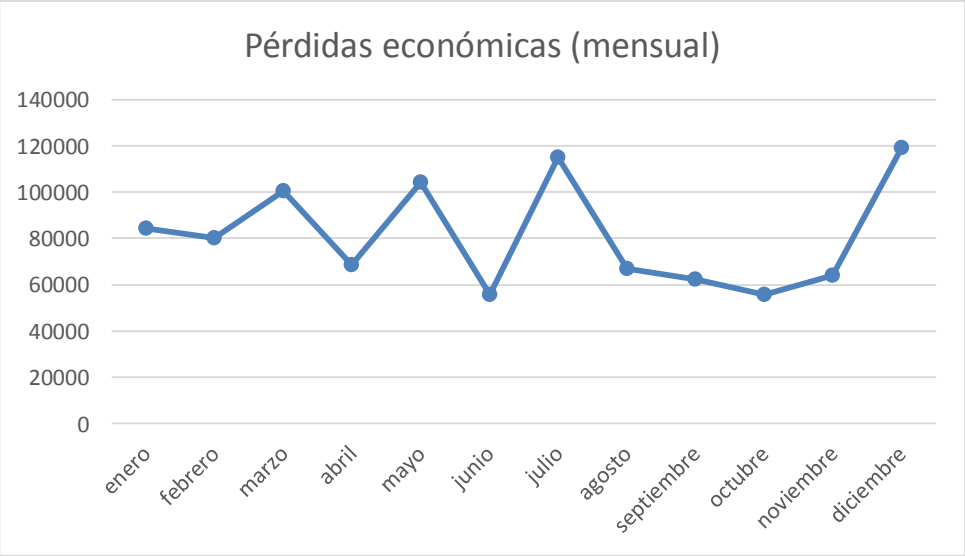
En la siguiente gráfica se muestra cómo se ha reducido los reclamos o quejas de los clientes. Es cierto que aún hay números altos de reclamos pero la propuesta ha conseguido reducirlos respecto a los números antes de la mejora.



Por lo tanto si los reclamos han disminuido, se generarían menos notas de crédito y con ello menos pérdidas para la empresa Claro. En el siguiente se muestra el valor aproximado en soles de lo mencionado con la reducción al 80%.

| Mes | Pérdidas económicas en soles |
|---|---|
| enero | 84444 |
| febrero | 79980 |
| marzo | 100440 |
| abril | 68448 |
| mayo | 104160 |
| junio | 55800 |
| julio | 114948 |
| agosto | 66960 |
| septiembre | 62124 |
| octubre | 55800 |
| noviembre | 63984 |
| diciembre | 119040 |

La gráfica siguiente muestra los datos después de la implementación de la propuesta, si bien es cierto el gráfico no ha cambiado mucho en forma, pero si se observa los números vamos en cuanto se han reducido.



INTERPRETACIÓN

Con la implementación de la propuesta se ha logrado reducir en gran parte las pérdidas económicas que pueda tener la empresa Claro con respecto al proceso de migración de líneas postpago. Ahora, si bien es cierto se ha reducido en gran parte el problema de generación de notas de crédito producto de los reclamos, éste aún persistiría, ya que el proceso de migración involucra varios componentes y alguno de ellos están en servidores con versiones anteriores a la actual, cuando alguno de ellos cae afectaría al proceso de migración en general y por lo tanto llegar de nuevo a lo mismo. Esta situación no pasa con frecuencia pero es importante mencionarlo para tener conocimiento del porqué y cómo es que se ve afectada el proceso de migración en general.

De esta manera damos credibilidad de que el proyecto aporta y mucho al proceso de migración y cumple con los objetivos planteados inicialmente.

# CONCLUSIONES

El análisis del proceso de migración de líneas postpago en la empresa Claro influyó positivamente ya que permitió definir los requerimientos del sistema, así como pautas a tomar en cuenta para adaptar el sistema a las necesidades del proceso de negocio.

El modelo de la arquitectura orientado a servicios (SOA) es un gran aporte ya que como se ha podido ver permite la comunicación entre la los aplicativos y las bases de datos, y lo más importante es que como es a nivel de servicios estos pueden estar desarrollados en distintos lenguajes de programación y no impedirán que el proceso se desarrolle de forma correcta.

El desarrollo del proyecto ayudará en gran medida al proceso de migración porque está creada en base a los requerimientos del mismo proceso, el proceso no se estaba desarrollando de forma correcta y con éste proyecto solucionamos ello. Esto es muy importante ya que evita riesgos de pérdidas económicas y algo que ha estado últimamente sucediendo es que los usuarios están migrando hacia otros operadores, si el problema se mantendría terminarían quejándose y migrando a otras compañías que brindan el mismo servicio.

La mejora del proceso ha permitido reducir los tiempos de respuesta de los servicios, evitar los reprocesos que sobrecargaban el servidor y en muchos casos la caída del mismo.

Con el desarrollo de la propuesta se ha permitido realizar el Cambio de Plan y la activación del tope de consumo el mismo día.

La propuesta busca solucionar el proceso de migración que se está desarrollando de manera deficiente, llevando a realizar trabajos manuales cuando estos deberían ser automáticos.

Son grandes las cantidades de reclamos generados en cada mes producto del mal funcionamiento del proceso de migración, el cual la propuesta busca solucionar.

La propuesta de mejora del proceso de migración, exclusivamente en el aprovisionamiento en Janus permitirá mejorar el proceso de la validación de contrato que son generados por cada migración, estos ocurren con otros componentes pero de similar forma, por lo tanto el desarrollo de la propuesta servirá como ejemplo para su implementación.

Sin duda la implementación de la propuesta reduce en gran parte el efecto del problema en mención, ya que se busca que la cantidad de reclamos sea mínima, esto es beneficioso tanto para Claro como para el cliente; para Claro para no generar notas de crédito y no tener pérdidas y para el cliente porque recibirá un mejor servicio según lo establecido en su contrato.

# RECOMENDACIONES

Se recomienda tomar como base los resultados del proyecto para la evaluación de la implementación de la mejora del proceso de Migración de líneas postpago.

Para que el sistema funcione correctamente y devuelva los resultados deseados, las personas encargadas de manejar los aplicativos deben estar capacitados para que tengan conocimiento del funcionamiento y secuencia del proceso.

Se debe garantizar la disponibilidad de las bases de datos y servidores activos para el correcto desempeño del sistema.

Con el objetivo de mejorar el sistema se deben tomar en cuenta las apreciaciones de los usuarios, en la medida que usen el sistema para lograr mejores versiones del mismo.

Se recomienda trabajar con la arquitectura orientada a servicios SOA ya que si en algún momento existe la posibilidad de que los mismos componentes planteados se requieran para otro proyecto solo se necesario reutilizarlos.

Se recomienda el uso del modelo en Cascada cuando los proyectos tienen los requisitos claros, y cuando haya una probabilidad nula de cambios en el sistema.

El proceso de migración engloba varios subprocesos, se recomienda revisar los procesos alternos al tomado para mejorar el proceso de forma más completa.

Se recomienda continuar con el proyecto porque da solución al problema expuesto, solución a problemas técnicos y mejora del proceso de migración de clientes con línea postpago, mientras que por otro lado se logra satisfacer las necesidades del cliente.

En próximas iteraciones sería conveniente seguir mejorando las partes del proceso, como también la migración de elementos que están en versiones antiguas a versiones nuevas que cuenten con los nuevos marcos de desarrollo y permitir de esta manera la comunicación e interacción más efectiva de los componentes requeridos para el correcto funcionamiento o ejecución del proceso de migración.

Estar continuamente aplicando la mejora continua para soluciones futuras y obtener resultados eficaces.

Replicar el modelo planteado para proyectos de similar desenvolvimiento previa evaluación a cumplir los objetivos.

## BIBLIOGRAFÍA


Álvarez, C. (2014). ¿Qué es Spring Framework? Figuras 4, 5 y 6. Recuperado de: http://genbetadev.com

Aguilera, S. (2014). Unix-Linux. Curso sobre Sistema Operativo Centralizado. Recuperado de: http://repositorio.ub.edu.ar/bitstream/handle/123456789/3047/2724-practica%20profesional%20II-1-Aguilera.pdf?sequence=1&isAllowed=y

América Móvil Perú (Claro, 2016). Arquitectura de Referencia

Andrade, D., Jarama, J., & Morán, F. (2007). Servicio al Cliente Analítico. Recuperado de: http://repositorio.ug.edu.ec/bitstream/redug/6964/1/Tesis%20Completa-133-2007.pdf

Arquitectura Orientado a Servicios, (s.f.). En Wikipedia. Recuperado el 25 de febrero del 2016 de: https://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

Avilés, J., & Peralta, M. (2016). Diseño e implementación de una solución de integración de autenticación entre plataformas windows y linux, utilizando el directorio activo de windows como controlador de dominio (Bachelor's thesis, Espol).

Recuperado de:

http://www.dspace.espol.edu.ec/xmlui/bitstream/handle/123456789/35610/D-103089.pdf?sequence=-1&isAllowed=y

Blancarte, O. (1 de agosto del 2014). Estructura de datos – Queue (Cola). Oscar Blancarte Blog.

Recuperado de:

https://www.oscarblancarteblog.com/2014/08/01/estructura-de-datos-queue-cola/

Beltrán, A., Castro, F., & Merchán, Y. (2009). Oracle Administration Manager Data Base MAO. DB.

Recuperado de:

http://repositorio.ug.edu.ec/handle/redug/6801

Carlos Ble. (2013).Modelo en Cascada. LIBROSWEB.

Recuperado de:

http://librosweb.es/libro/tdd/capitulo_1/modelo_en_cascada.html

Desarrollo en Cascada, (s.f.). En Wikipedia. Recuperado el 25 de febrero del 2016 de: https://es.wikipedia.org/wiki/Desarrollo_en_cascada

Dpto. de Ciencia de la Computación e Inteligencia Artificial. Universidad de Alicante. (06/26/2014). Servicios de Mensajes con JMS. Recuperado de: http://www.jtech.ua.es/j2ee/publico/mens-2010-11/index.html

Enterprise application integration. (s.f.). En Wikipedia. Recuperado el 12 de marzo del 2017 de: https://es.wikipedia.org/wiki/Enterprise_application_integration

Falagan, M. (2011).Los Web Services. AuraPortal. Recuperado de: https://falagan.wordpress.com

González, A., & a Objeto, O. (2007). Ingeniería de Software: Metodologías. Visitado el, 8. Recuperado de: http://profesores.elo.utfsm.cl/~agv/elo329/1s16/lectures/SoftwareEngineering/Metodologias_DesarrolloIncrementalIterativo.pdf

González, A., & a Objeto, O. (2007). Ingeniería de Software: Metodologías. Visitado el, 8.
Recuperado de:
http://profesores.elo.utfsm.cl/~agv/elo329/1s10/lectures/SoftwareEngineeringParte2.pdf

Gil, A. (2016). Servicio web REST de gestión de eventos con Java Spring Framework: Panel de gestión web.
Recuperado de:
https://idus.us.es/xmlui/handle/11441/45198

Hernández, A., Rodríguez A., Parra J., & Cachimaille Y. (2008). Metodología para la formación de recursos humanos del sistema nacional de salud en ensayos clínicos. Educación Médica Superior, 22(3), 0-0.
Recuperado de:
http://scielo.sld.cu/pdf/ems/v22n3/ems09308.pdf

Herrera A., Guillén, R., & Javier, W. (2012). Estudio Comparativo de Servidores de Aplicaciones para Desarrollo de Software con SOA sobre Plataformas Javaee. Caso Práctico: Transportes Patria.
Recuperado de:

http://dspace.espoch.edu.ec/bitstream/123456789/1528/1/18T00469.pdf

Hidalgo, A. (2015). Sistema informático para el registro y control de reclusos por contravenciones de tránsito del centro de rehabilitación de varones de Babahoyo (Bachelor's thesis). Recuperado de: http://dspace.uniandes.edu.ec/bitstream/123456789/1176/1/TUBSIS001-2015.pdf

IDS2015, (2015). Arquitectura Orientada a Servicios (SOA). Ingeniería de Software UAH. Recuperado de: https://ingenieriadelsoftwareuah2015.wordpress.com/2015/03/22/arquitectura-orientada-a-servicios-soa/

Java EE. (s.f). En Wikipedia. Recuperado el 12 de marzo del 2017 de: https://es.wikipedia.org/wiki/Java_EE

Labrador, M. CURSO 05-08 PROGRAMACIÓN AVANZADA EN SHELL. Recuperado de: http://www.informatica.us.es/~ramon/articulos/Programacion-BASH.pdf

Laura, D., & Hugo, V. (2015). Sistema Web de control de asistencia para optimizar la liquidación de servicios de una empresa de Outsourcing.
Recuperado de:
http://repositorio.cientifica.edu.pe/bitstream/handle/UCS/343/TLC-Duran_Laura.pdf?sequence=1&isAllowed=y

Leandro, A. (2015). Definición de Queue (informática). Alegsa, Diccionario de informática y tecnología.
Recuperado de:
http://alegsa.com.ar

Ledezma E. (2017). MODELO CARACTERISTICAS VENTAJAS DESVENTAJAS CASCADA.
Recuperado de:
https://www.academia.edu/5130339/MODELO_CARACTERISTICAS_VENTAJAS_DESVENTAJAS_CASCADA

Licenciatura en RR.HH. Universidad de Champagnat. (2002, Septiembre 11). Mejora e innovación de procesos.
Recuperado de:
https://www.gestiopolis.com/mejora-innovacion-procesos/

López, M. (2012). Diseño de un módulo de carga de pagos en entidades públicas mediante mensajería con spring framework. Industrial Data, 15(2), 073-079.
Recuperado de:
http://revistasinvestigacion.unmsm.edu.pe/index.php/idata/article/view/6374/5584

López, M., López, M., Luna, A., & Vásquez, L. (2011). Sistema de Información para el Control de Inventarios del Almacén del ITS. ConCiencia Tecnológica, (41), 41-46.

MARTINEZ, A. (2008). METODOLOGIA PARA EL DESARROLLO DE UNA INTERFAZE COMPUTACIONAL DE EMPRESA A EMPRESA PARA TRANSFERENCIA DE DATOS EN LINEA (Doctoral dissertation).
Recuperado de:
http://tesis.ipn.mx/bitstream/handle/123456789/376/Binder5.pdf?sequence=1&isAllowed=y

Microsoft Corporation (Microsoft, 2006). La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real.
Recuperado de:

https://www.microsoft.com/en-us/cloud-platform/biztalk

Martinez, S., & Gonzalez, W. (2015). Plataforma interactiva consultorio social-Uniminuto.
Recuperado de:
http://repository.uniminuto.edu:8080/jspui/bitstream/10656/3814/1/TTI_GonzalezMonjeWilmar_2015.pdf

Oracle Database. (s.f). En Wikipedia. Recuperado el 12 de marzo del 2017 de:
https://es.wikipedia.org/wiki/Oracle_Database

Orantes, S., Gutiérrez, A., & López, M. (2009). Arquitecturas empresariales: gestión de procesos de negocio vs. Arquitecturas orientadas a servicios ¿se relacionan?. Tecnura, 13(25), 136-144.

Ordóñez, P. (2015). Diseño e implementación del sistema de facturación electrónica para Diario El Mercurio en APEX, con almacenamiento en Oracle DBMS y publicado en WebLogic (Bachelor's thesis).
Recuperado de:
http://dspace.ups.edu.ec/handle/123456789/8948

Otón, S. (2006). Propuesta de una arquitectura software basada en servicios para la implementación de repositorios de objetos de aprendizaje distribuidos.
Recuperado de:
http://dspace.uah.es/dspace/handle/10017/472

Pinedo, L., & Medina, L. (2010). Implementación de un sistema de integración para las bibliotecas municipales de Lima y Callao utilizando SOA y J2ME.
Recuperado de:
http://cybertesis.unmsm.edu.pe/bitstream/cybertesis/2647/1/Medina_bl.pdf

Piattini M. (1996). Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión. Rama. Madrid.

Prieto, C. (2015). Adaptación de las Metodologías Tradicionales Cascada y Espiral para la Inclusión de Evaluación Inicial de Usabilidad en el Desarrollo de Productos de Software en México.
Recuperado de:
http://jupiter.utm.mx/~tesis_dig/12947.pdf

Puertas, G., & Maykol, A. (2016). Una evaluación experimental para comparar la calidad de un software aplicando o no TDD dentro del modelo cascada.
Recuperado de:
http://tesis.pucp.edu.pe:8080/repositorio/bitstream/handle/123456789/6523/GOICOCHEA_ANTONY_EVALUACI%c3%93N_EXPERIMENTAL_SOFTWARE.pdf?sequence=1&isAllowed=y

Quesada, E. (2007). ACTUALIDAD INVITADA. Innovación, Calidad e Ingeniería del Software, 3(2), 40.
Recuperado de:
http://www2.ati.es/IMG//pdf/Num2Vol3Oct07.pdf#page=40

Rivas, R. (2013). Desarrollo de sistemas de información con framework Spring-Hibernate. Prueba de concepto: desarrollo de un sistema de préstamo bibliotecario (Bachelor's thesis).

Salinas, E., Cerpa, N., & Rojas, P. (2011). Arquitectura orientada a servicios para software de apoyo para el proceso personal de software. *Ingeniare. Revista chilena de ingeniería*, *19*(1), 40-52.
Recuperado de:
http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-33052011000100005

Santos, F. (2014). Diseño de un módulo de carga de pagos en entidades públicas mediante mensajería con spring framework. Industrial Data, 15(2), 073-079.
Recuperado de:
http://revistasinvestigacion.unmsm.edu.pe/index.php/idata/article/view/6374/5584

Sistema de gestión de base de datos. (s.f).En Wikipedia. Recuperado el 12 de marzo del 2017 de:
https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos

Serna, D., Salazar, E., & Cortés, A. (2010). Arquitectura orientada a servicios en el contexto de la arquitectura empresarial. Avances en Sistemas e Informática, 7(2), 74-88.
Recuperado de:
http://www.revistas.unal.edu.co/index.php/avances/article/view/26600

Servidor de aplicaciones. (s.f.). En Wikipedia. Recuperado el 12 de marzo de: https://es.wikipedia.org/wiki/Servidor_de_aplicaciones

Urrutia, A., López, E., Martínez, F., & Corral, A. (2015). Procesos de desarrollo para videojuegos. CULCyT, (37).
Recuperado de:
http://openjournal.uacj.mx/ojs/index.php/culcyt/article/view/299/283

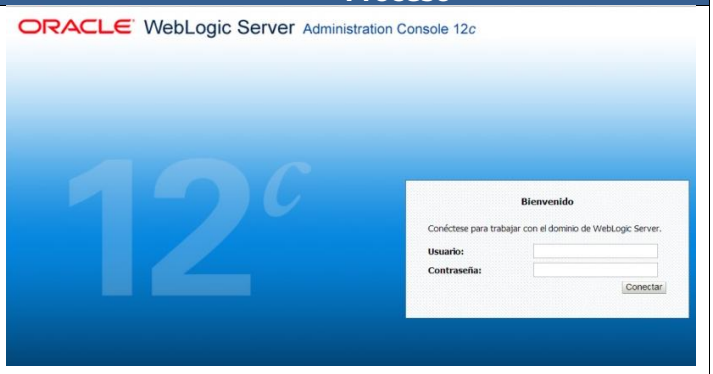
Vera, M. (2014). ¿Qué se entiende por SOA, y cuáles son sus beneficios? San Isidro, Lima, Perú. Intelligence to Business.
Recuperado de:
http://www.i2btech.com/blog-i2b/tech-deployment/que-se-entiende-por-soa-y-cuales-son-sus-beneficios/

Vera, T., & Gonzalo, F. (2009). Análisis de métodos, técnicas y herramientas de verificación y validación de software usados por empresas ecuatorianas desarrolladoras de software (Bachelor's thesis).
Recuperado de:
https://www.dspace.espol.edu.ec/bitstream/123456789/7735/1/D-39464.pdf

## ANEXO

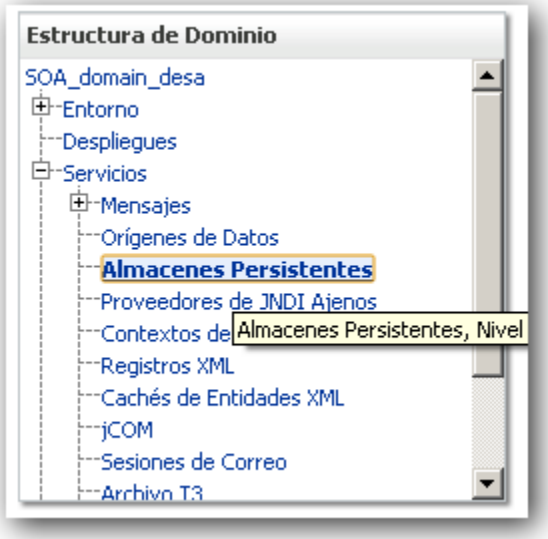
Por temas de confidencialidad de direcciones IP, usuarios y claves no son datos reales, solo son datos de referencia.

**ANEXO A**
CONFIGURACION DE JMS
(Los datos son solo de referencia)

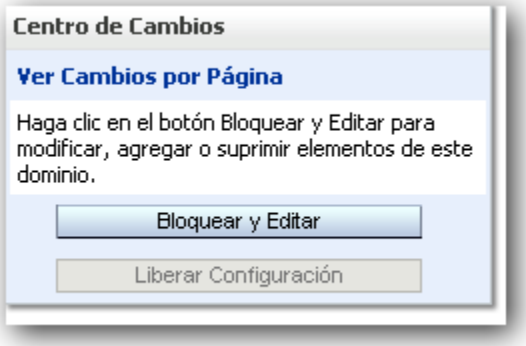
| Descripción | Proceso |
|---|---|
| 1. Ingresar a la consola:<br><br>http://1270.0.0.1:7901/console<br><br>2. Proporcionar el usuario y password:<br><br>**Username:** weblogic<br>**Password:** ***** |  |

El siguiente procedimiento solo debe ser aplicado una vez por cada **Managed Server** miembro del clúster. Ejecutar los siguientes pasos por cada nodo del **Cluster,** donde se instalara el componente **PersistenStores** nombre **ControlConsumoPostpagoValidaAprovisionMDBFileStore_auto_0X**

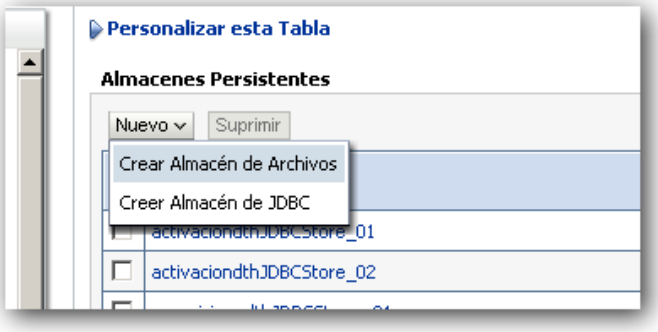
3. En el recuadro Estructura de Dominio elegir la opción "Almacenes Persistentes".

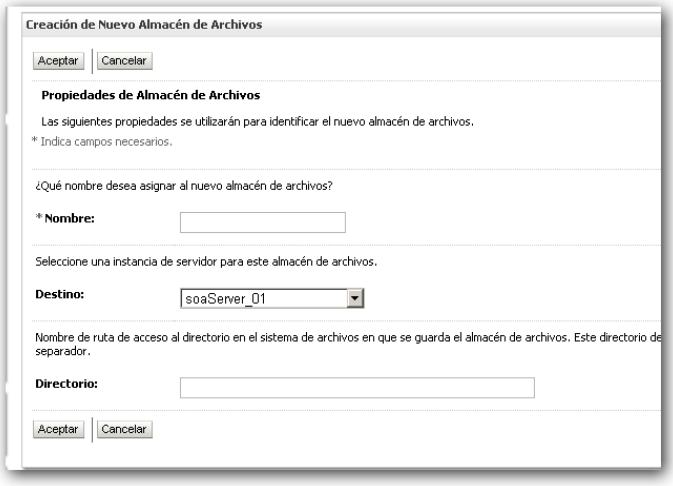
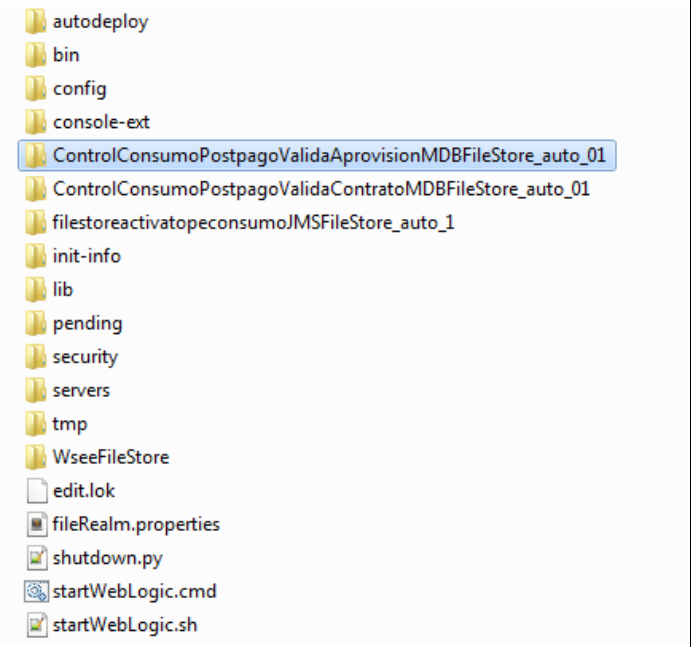
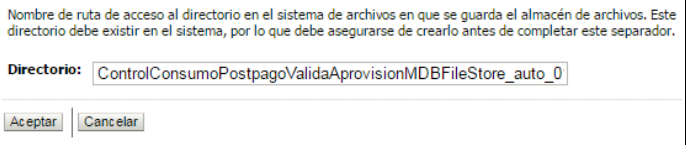
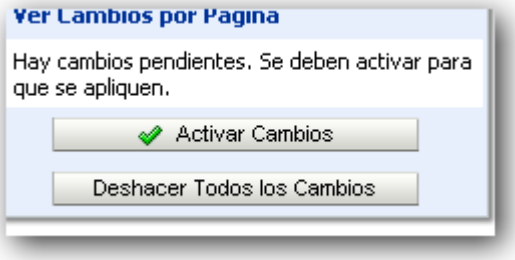



4. En el recuadro Centro de Cambios, hacer click al botón "Bloquear y Editar".




5. En el recuadro Resumen de Almacen Persistente hacer click al botón "Nuevo" – "Crear Almacen de Archivos".

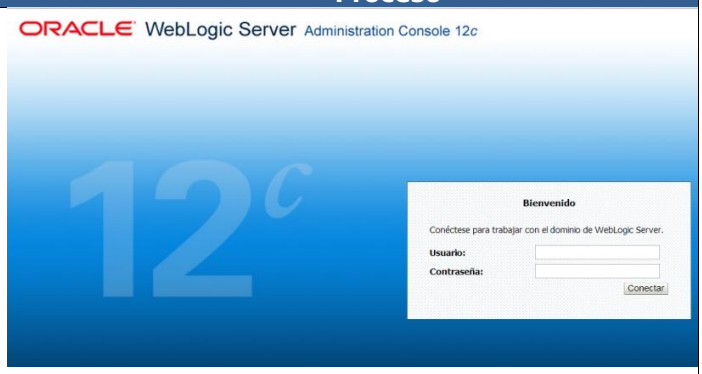
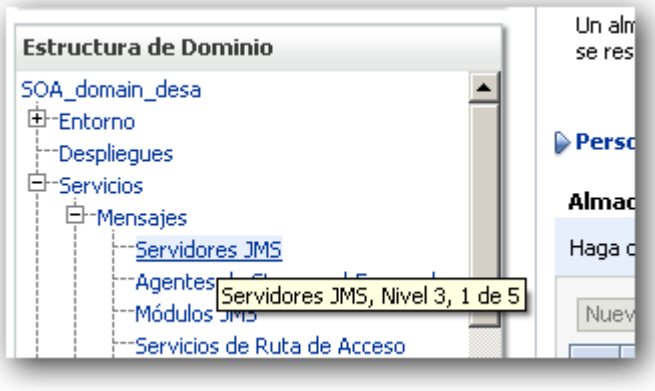


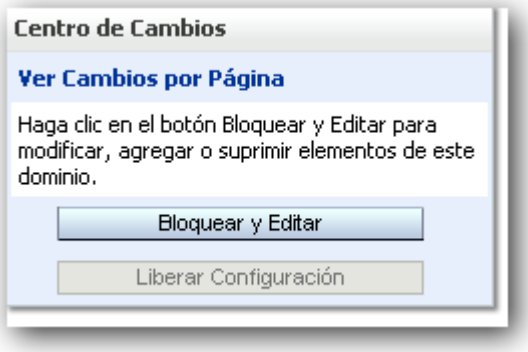
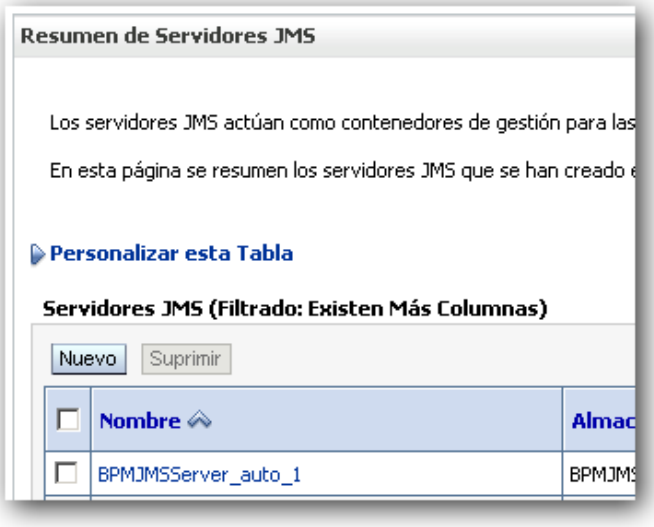
| | |
|---|---|
| 6. Ingresar los siguientes parámetros en las cajas te texto para los Labels:<br><br>Nombre**:** **ControlConsumoPostpagoValidaAprovisionMDBFileStore_auto_0X**<br><br>Destino**: soaServer_0X**<br><br>Directorio:<br>**ControlConsumoPostpagoValidaAprovisionMDBFileStore_auto_0X** |  |
| 7. Antes de darle Aceptar verificar que en el directorio del servidor:<br><br>**/oracle/weblogic/nodeManager/jmsserverMachine_XX/filestore/**<br><br>Se encuentre creado la carpeta:<br><br>**ControlConsumoPostpagoValidaAprovisionMDBFileStore_auto_XX**<br><br>Si no es asi, crearla. |  |
| 8. Seleccionar "OK". |  |
| 9. Hacer click en "Activar Cambios". |  |

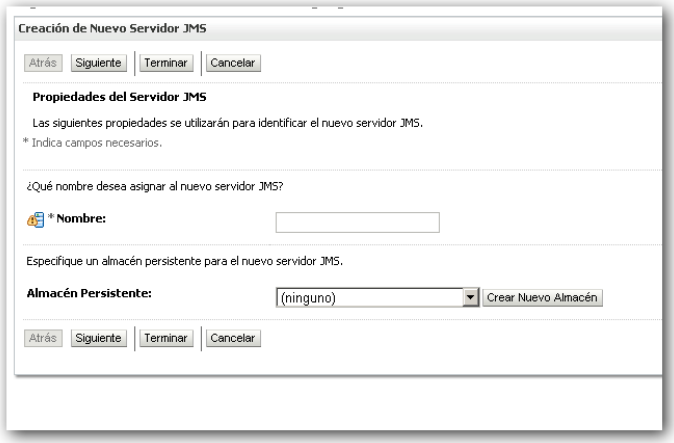
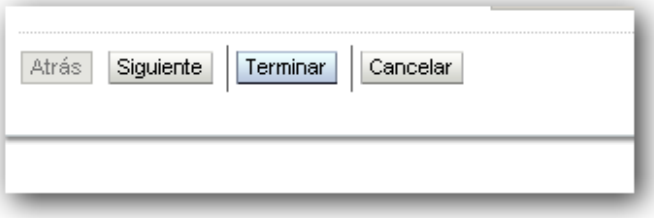
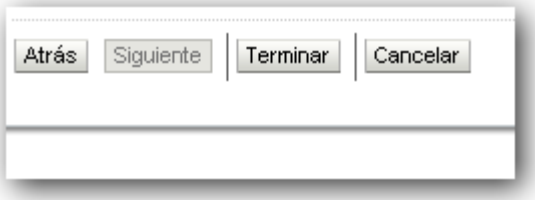
**ANEXO B**

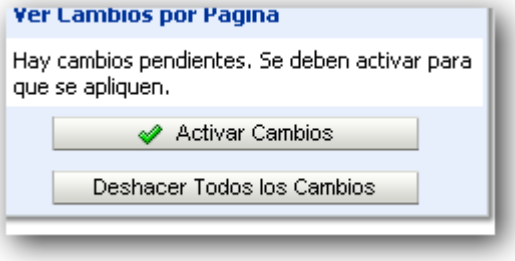
INSTALACION DE JMS SERVER
Los datos son solos de referencia

(Las imágenes, IP, usuarios y claves no son datos reales son solo de referencia)

| Descripción | Proceso |
|---|---|
| 1. Ingresar a la consola:<br><br>http://1270.0.0.1:7901/console<br><br>2. Proporcionar el usuario y password:<br><br>**Username:** weblogic<br>**Password:** ***** |  |
| El siguiente procedimiento solo debe ser aplicado una vez por cada **Managed Server** miembro del clúster. Ejecutar los siguientes pasos por cada **Cluster,** donde se instalara el componente **Servidores JMS** nombre **ControlConsumoValidaAprovisionMDBJMSServer_XX**<br><br>3. En el recuadro Estructura de Dominio elegir la opción "Servidores JMS". |  |

| | |
|---|---|
| 4. En el recuadro Centro de Cambios, hacer click al botón "Bloquear y Editar". |  |
| 5. En el recuadro Resumen de Servidores JM hacer click al botón "Nuevo". |  |

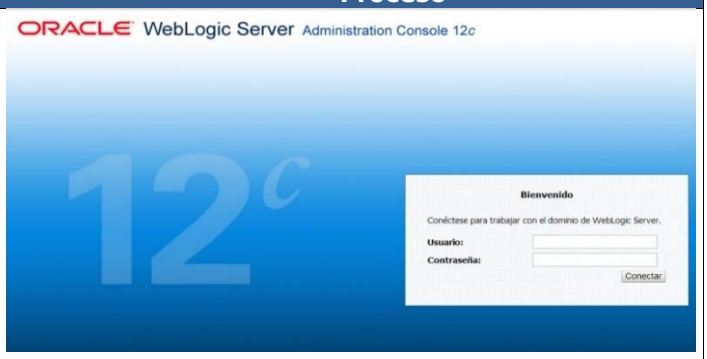
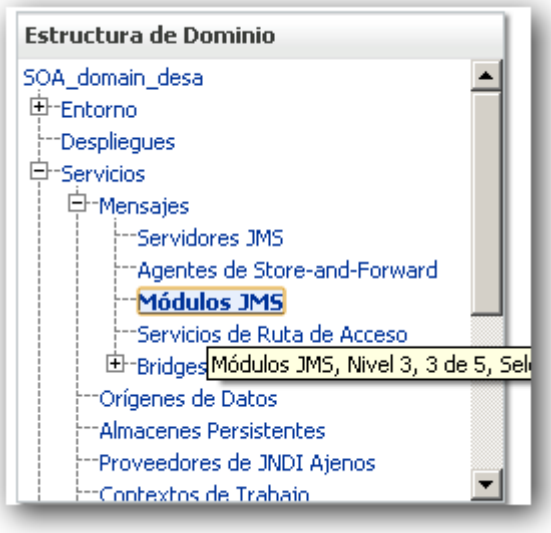
| | |
|---|---|
| 6. Verificar que el secuencial del JMSServer y Persistence Store coincidan<br><br>- JMSServer: **ControlConsumoValidaAprovisionMDBJMSServer_XX**<br><br>- PersistentStore: **ControlConsumoPostpagoValidaAprovisionMDBFileStore_auto_XX**<br><br>7. Ingresar los siguientes parámetros en las cajas te texto para los Labels:<br><br>- Nombre: **ControlConsumoValidaAprovisionMDBJMSServer_XX**<br><br>- Almacen Pesistente: **ControlConsumoPostpagoValidaAprovisionMDBFileStore_auto_XX** |  |
| 8. Seleccionar "Siguiente". |  |
| 9. Seleccionar target, debe de coincidir con el secuencial del Nombre del **ControlConsumoValidaAprovisionMDBJMSServer_XX** |  |
| 10. Seleccionar la opción "Terminar". |  |

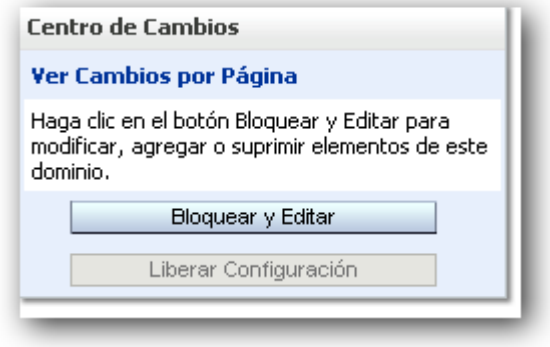
| | |
|---|---|
| 11. Hacer click en "Activar Cambios". |  |

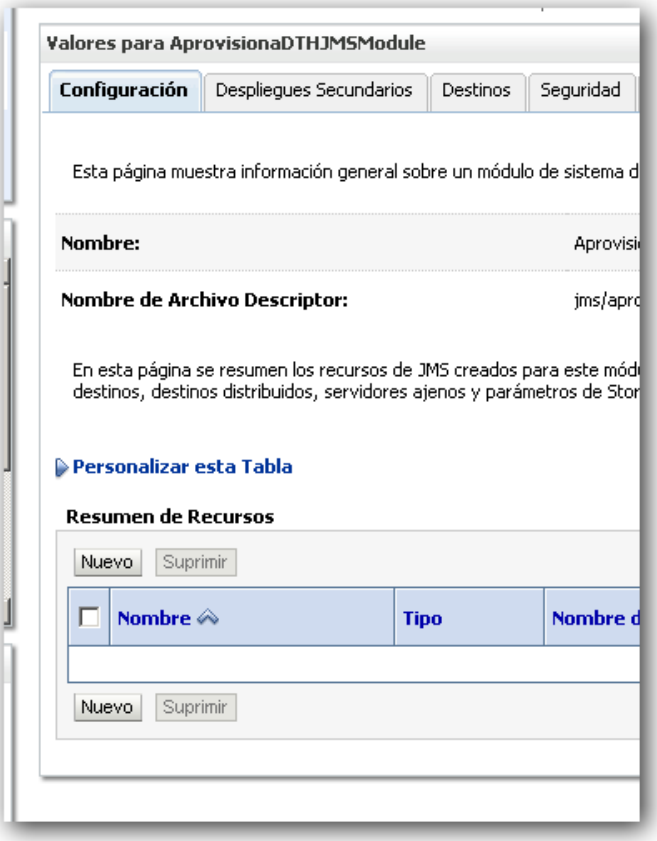
## ANEXO C

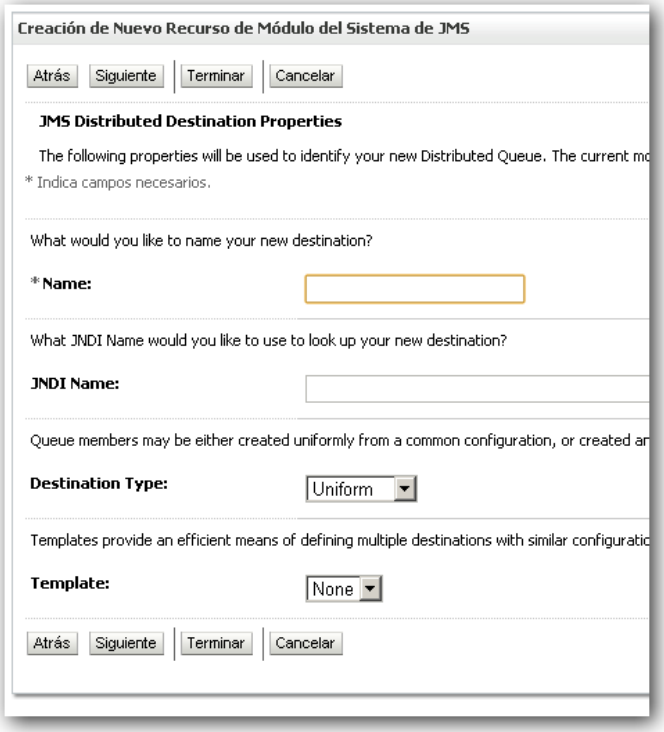
INSTALACION DE LA COLA DISTRIBUIDA
pe.com.claro.jndi.validaAprovision.jms.queue

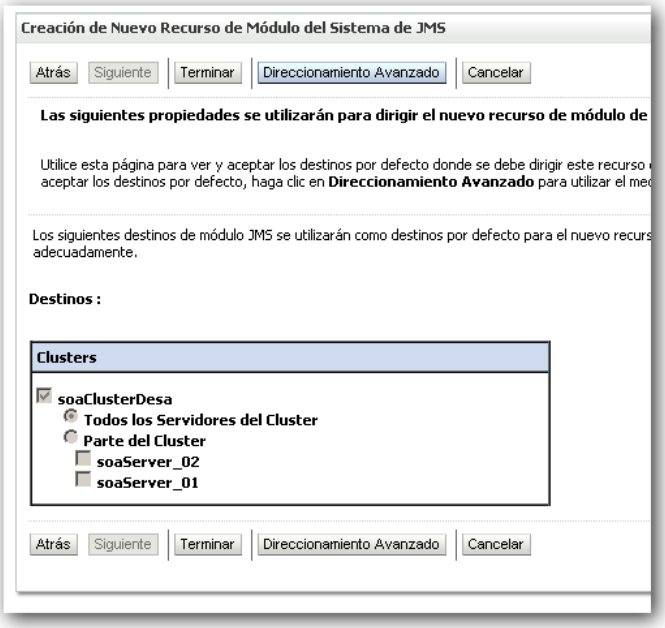
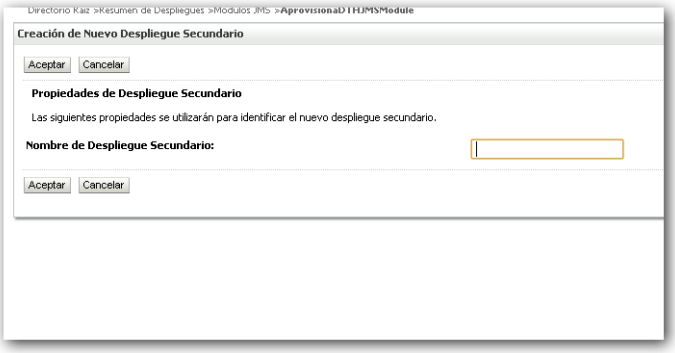
(Las imágenes, IP, usuarios y claves no son datos reales son solo de referencia)

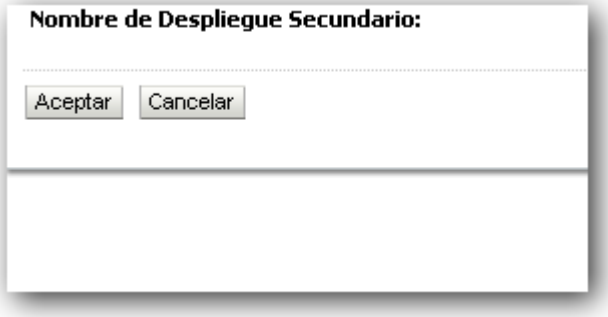
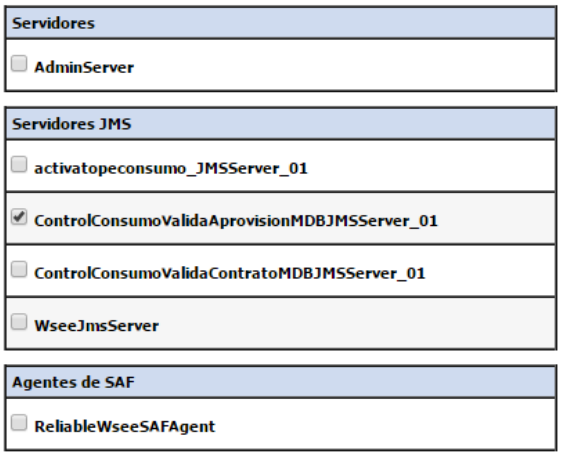
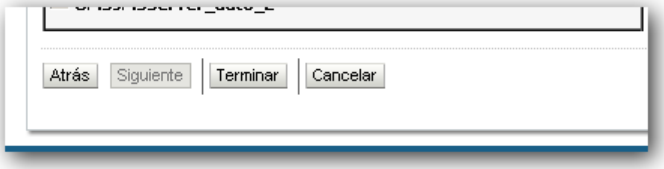
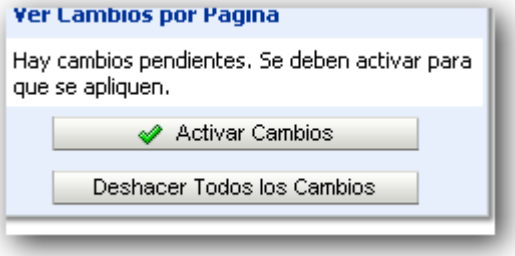
| Descripción | Proceso |
|---|---|
| 1. Ingresar a la consola:<br><br>http://127.0.0.1:7001/console<br><br>2. Proporcionar el usuario y password:<br><br>**Username:** weblogic<br>**Password:** ***** |  |
| 3. En el recuadro Estructura de Dominio elegir la opción "Módulos JMS": |  |

| | |
|---|---|
| 4. En el recuadro Centro de Cambios, hacer click al botón "Bloquear y Editar". |  |
| 5. En el recuadro Modulos JMS hacer click al componente creado **ControlConsumoValidaAprovisionMDBJMSModule** |  |

| | |
|---|---|
| 6. En el recuadro Valores para **ControlConsumoValidaAprovisionMDBJMSModule** en la pestaña Configuracion, hacer click al boton "Nuevo". |  |
| 7. Seleccionar la opción "Cola Distribuida", luego hacer click en la opción "Siguiente". |  |

| | |
|---|---|
| 8. Ingresar los siguientes parámetros en las cajas te texto para los Labels:<br>• Name: **validaAprovision.queue**<br>• JNDI Name: **pe.com.claro.jndi.validaAprovision.jms.queue**<br>• Destination Type: (por defecto)<br>• Template:(por defecto) |  |
| 9. Seleccionar "Siguiente". |  |

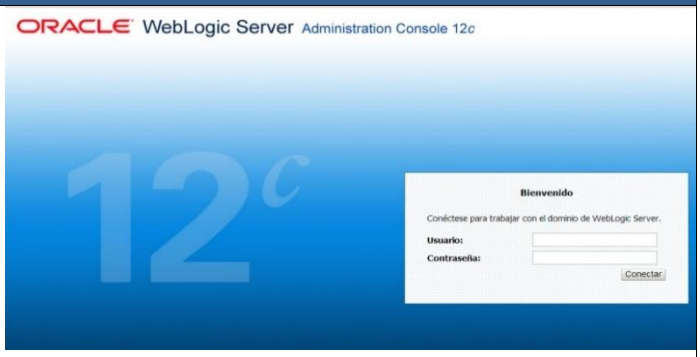
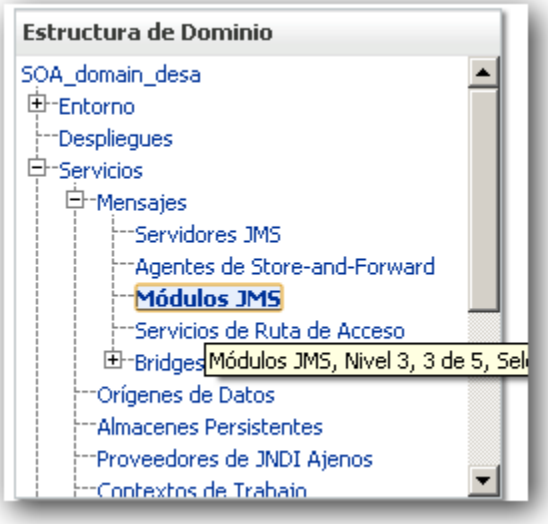
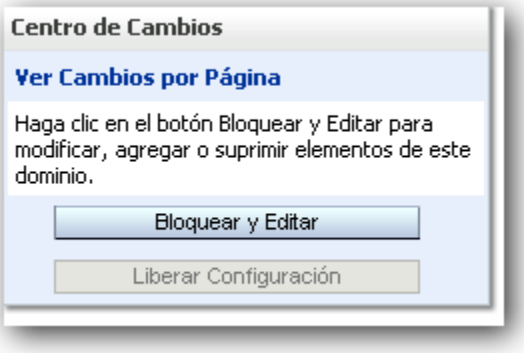
| | |
|---|---|
| 10. Hacer click al botón "Direccionamiento Avanzado". |  |
| 11. Hacer click al botón "Crear Nuevo Despliegue". |  |
| 12. Ingresar los siguientes parámetros en las cajas te texto para los Labels:<br>• Subdeployment Name: **validaAprovisionSubdeployment** |  |

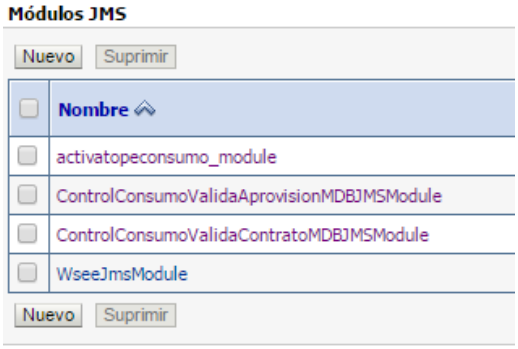
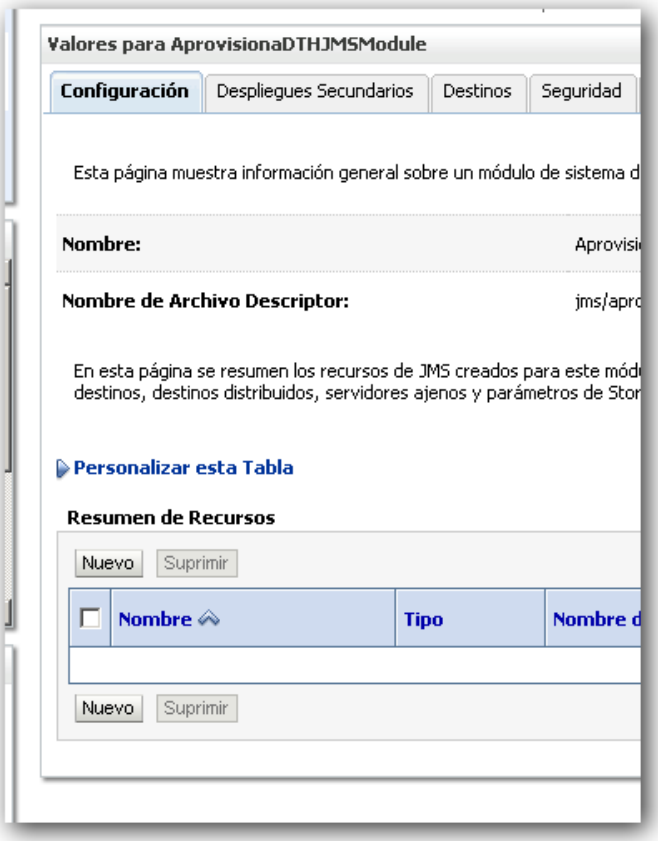
| | |
|---|---|
| 13. Seleccionar la opción "Aceptar". |  |
| 14. Seleccionar los Servidores JMS correspondientes creados en el procedimiento anterior.<br>- **ControlConsumoValidaAprovisionMDB JMSServer_01**<br>- **ControlConsumoValidaAprovisionMDB JMSServer_02**<br>- **ControlConsumoValidaAprovisionMDB JMSServer_03**<br>.<br>.<br>- .<br>**ControlConsumoValidaAprovisionMDB JMSServer_XX** |  |
| 15. Seccionar la opción "Terminar". |  |
| 16. Hacer click en "**Activar Cambios**". |  |

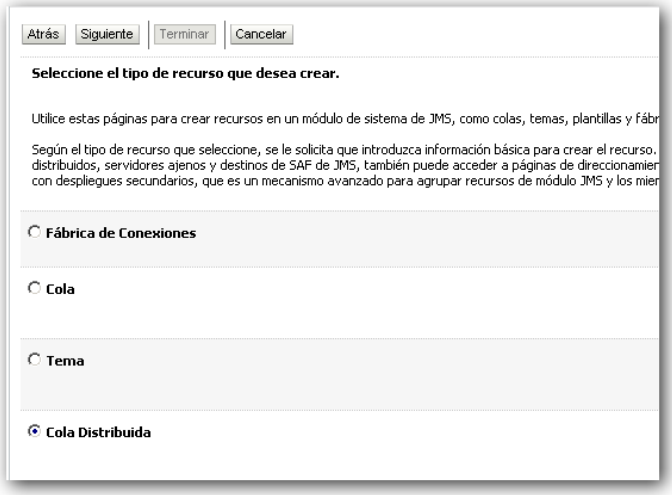
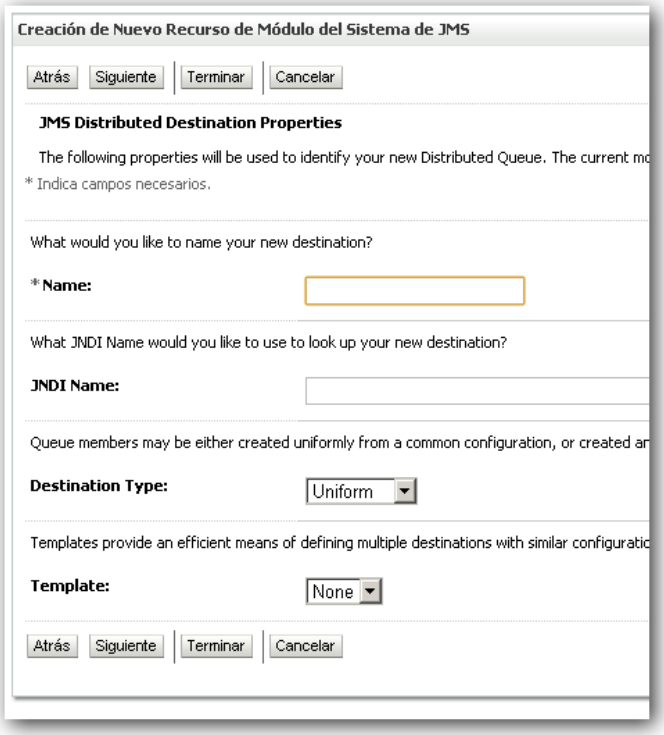
**ANEXO D**

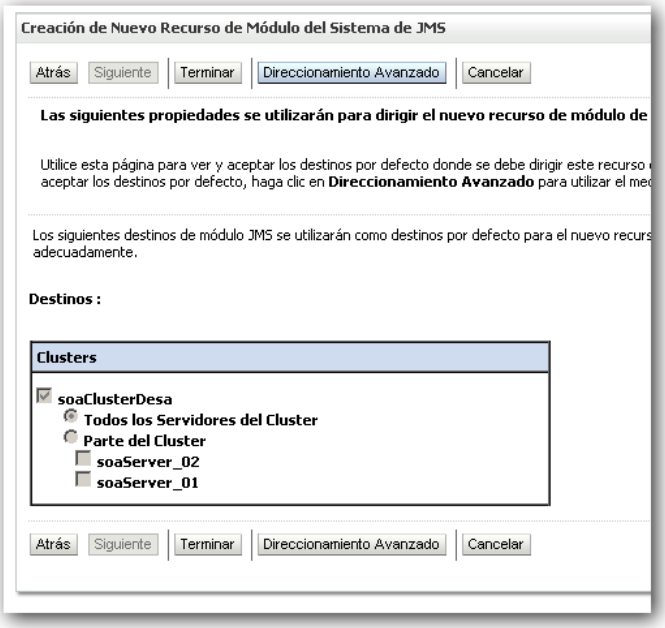
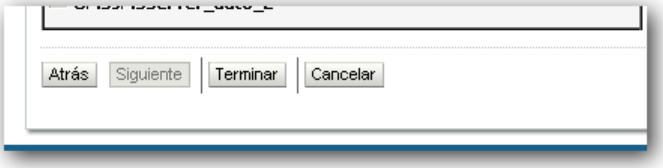
CREACION DE LA COLA DE ERROR
pe.com.claro.jndi.validaAprovision.jms.error.queue

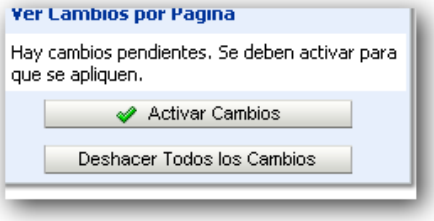
(Las imágenes, IP, usuarios y claves no son datos reales son solo de referencia)

| Descripción | Proceso |
|---|---|
| 1. Ingresar a la consola:<br><br>http://127.0.0.1:7001/console<br><br>2. Proporcionar el usuario y password:<br><br>**Username:** weblogic<br>**Password:** ***** |  |
| 3. En el recuadro Estructura de Dominio elegir la opción "Módulos JMS": |  |
| 4. En el recuadro Centro de Cambios, hacer click al botón "Bloquear y Editar". |  |

| | |
|---|---|
| 5. En el recuadro Modulos JMS hacer click al componenten creado **ControlConsumoValidaAprovisionMDBJMS Module** |  |
| 6. En el recuadro Valores para **ControlConsumoValidaAprovisionMDBJMS Module** en la pestaña Configuracion, hacer click al botono "Nuevo". |  |

| | |
|---|---|
| 7. Seleccionar la opción "Cola Distribuida", luego hacer click en la opción "Siguiente". |  |
| 8. Ingresar los siguientes parámetros en las cajas te texto para los Labels:<br>• Name : **validaAprovision.error.queue**<br>• JNDI Name: **pe.com.claro.jndi.validaAprovision.jms.error.queue**<br>• Destination Type: (por defecto)<br>• Template:(por defecto) |  |
| 9. Seleccionar "Siguiente". |  |

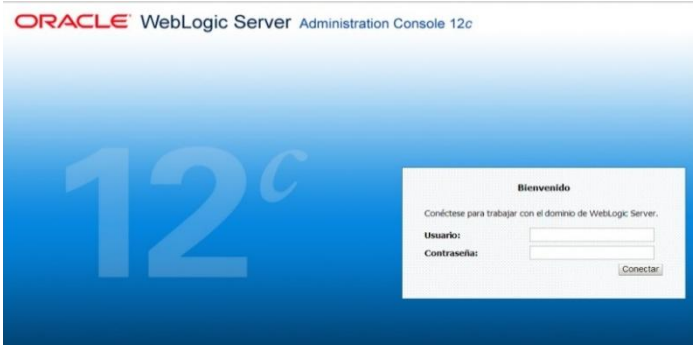
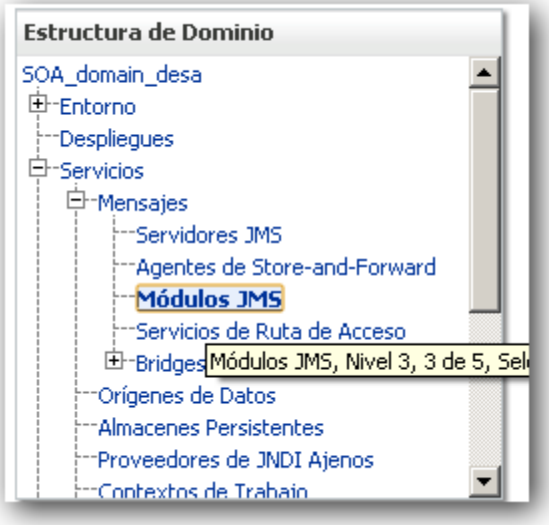
| | |
|---|---|
| 10. Hacer click al botón "Direccionamiento Avanzado". |  |
| 11. Seleccionamos los siguientes datos:<br><br>Despliegues Secundarios:<br>**validaAprovisionSubdeployment**<br><br>Destinos:<br>Seleccionar los Servidores JMS correspondientes creados en el procedimiento anterior.<br><br>- **ControlConsumoValidaAprovisionMDB JMSServer_01**<br>- **ControlConsumoValidaAprovisionMDB JMSServer_02**<br>- **ControlConsumoValidaAprovisionMDB JMSServer_03**<br>▪<br>▪<br>▪ **ControlConsumoValidaAprovisionMDB JMSServer_XX** |  |
| 12. Seccionar la opción "Terminar". |  |

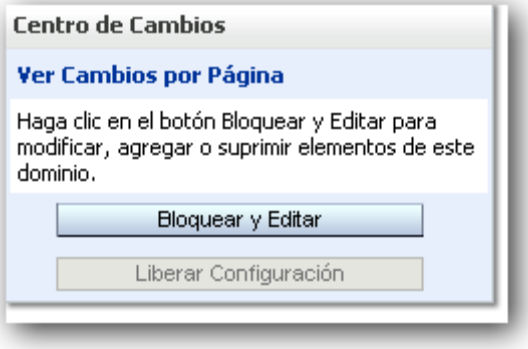
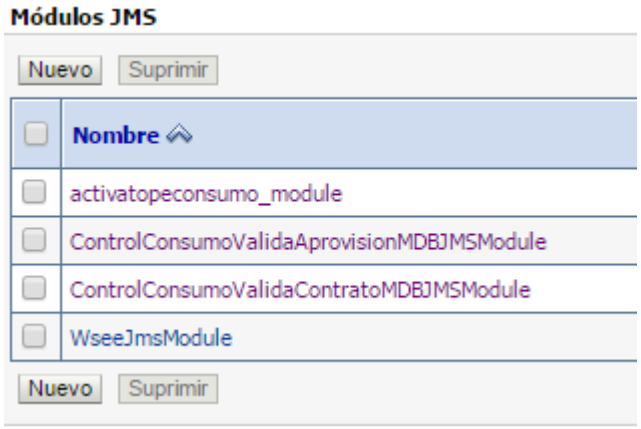
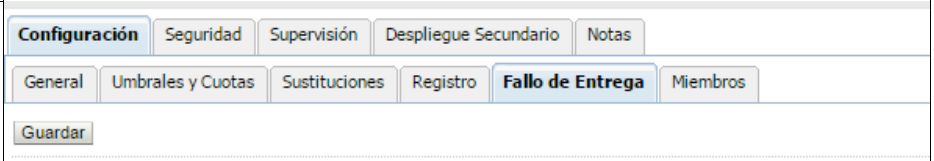
| | |
|---|---|
| 13. Hacer click en "Activar Cambios". |  |

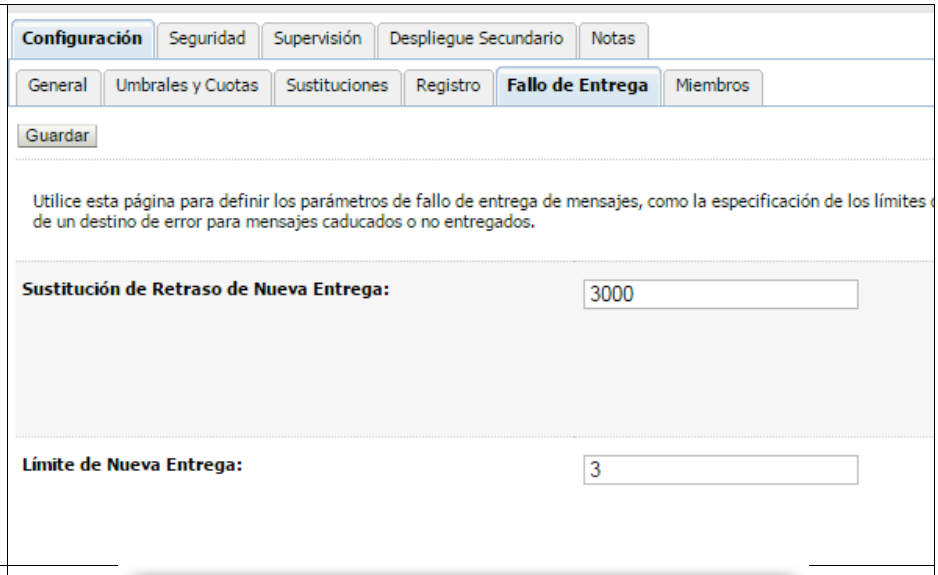
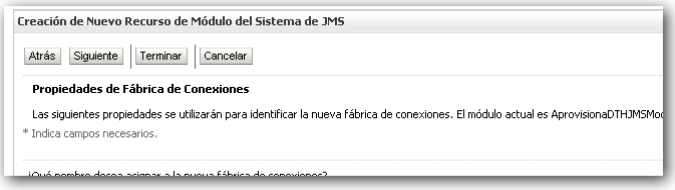
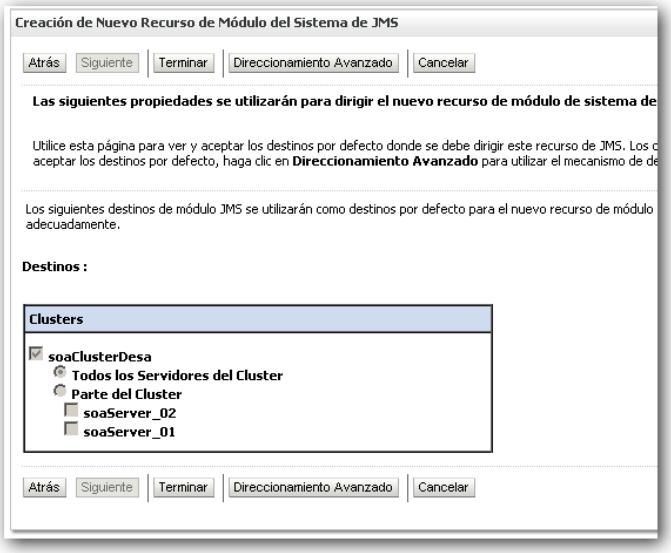
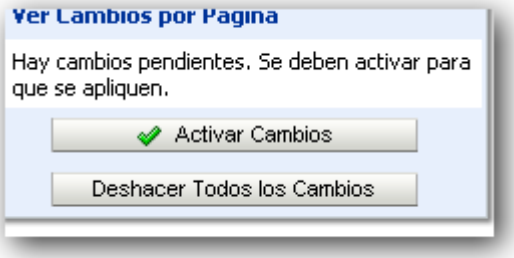
## ANEXO E

CONFIGURACIÓN DE REINTENTOS DE LA COLA
pe.com.claro.jndi.validaAprovision.jms.queue

(Las imágenes, IP, usuarios y claves no son datos reales son solo de referencia)

| Descripción | Proceso |
|---|---|
| 1. Ingresar a la consola:<br><br>http://127.0.0.1:7001/console<br><br>2. Proporcionar el usuario y password<br><br>**Username:** weblogic<br>**Password:** ***** |  |
| 3. En el recuadro Estructura de Dominio elegir la opción "Módulos JMS": |  |

| | |
|---|---|
| 4. En el recuadro Centro de Cambios, hacer click al botón "Bloquear y Editar". |  |
| 5. En el recuadro Modulos JMS hacer click al componente creado **ControlConsumoValidaAprovisionMDBJMSModule** |  |
| 6. Seleccionar la cola a modificar. |  |
| 7. Ingresar a la siguiente ruta:<br>Configuracion -> Fallo de entrega |  |

| | |
|---|---|
| 8. Realizar siguiente cambio : Límite de nueva entrega igual a 3 y seleccionar el botón guardar. |  |
| 9. Seleccionar "Siguiente". |  |
| 10. Hacer click al botón "Terminar". |  |
| 11. Hacer click en "Activar Cambios". |  |