Universidad de las Ciencias Informáticas

Facultad 1

Gestor de Tareas para Aprovisionamiento de Usuarios

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autores: Mario Azahares Mezquia.

Carlos Pujol Vargas.

Tutor: Ing. Yoandy Rodríguez Martínez.

Ciudad de La Habana, Junio de 2010

"Año 52 de la Revolución"

"...aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de sí mismos".

Che

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y otorgamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los _____ días del mes de _____ del año _____.

Mario Azahares Mezquia

_____

Firma del Autor

Carlos Pujol Vargas

_____

Firma del Autor

Ing. Yoandy Rodríguez Martínez

_____

Firma del Tutor

AGRADECIMIENTOS

Agradecerle a mi mamá por siempre estar a mi lado apoyándome en los buenos y malos momentos durante mis años de estudio, y por confiar en mí.

A mis hermanos Alejandro y Anamey que me han ayudado y apoyado siempre para que yo pudiese terminar mi carrera.

A mi papá, por brindarme su apoyo y su cariño.

A mi tía “Mulata”, por estar siempre pendiente de mí.

A mi tía María, por su amor y su preocupación.

A mi tío “Papito”, por su ayuda, su apoyo, su cariño durante todos estos años.

A mi novia Isabel, por su apoyo en cada momento desde que nos conocimos en Venezuela.

A mi prima Cristina y a su esposo Sadiely, por creer en mí, apoyarme y ayudarme cada vez que lo necesito.

A mis primas Tania y Yusmila, por su cariño.

A mi primo Robertico, que donde quiera que este se que está orgulloso de mí.

A Carmen, Luis y Carmen Luisa y la tía Ileana, por brindarme su cariño y dejarme formar parte de su familia.

A mi hermana Eliza, por su preocupación por mis estudios.

A mis primos Yusniel, Yunior, Yeni y Yordan, a los que quiero y trato de darles un buen ejemplo.

A mis primos que siempre han estado preocupados por mí y por mis resultados.

A mis vecinos Mabel, Iberis, Noel e Iliamne, por su ayuda y amistad.

A Odalys, por ser tan buena amiga y vecina.

A Yadira, Milvia y Alipio por su amistad y por estar ahí cada vez que necesito de ellos.

A mis amigos de antes y de ahora, Yoiner, Javier, Ariel, Yadira, Enrique, Uriser.

Y con los que compartí durante 7 meses como si fuéramos una familia y de donde ha surgido una verdadera amistad con Fuoman, JJon, Sariol y Angel.

A mi amigo Duany, que a pesar de vernos poco sé que siempre cuento con su apoyo y el de su familia.

A mi compañero de tesis, por ser un tipo “escapao”, por lucha y batallar hasta el final.

A los muchachos del Estado mayor de los cadetes con los que aprendí y me divertí mucho trabajando juntos.

A la Revolución Cubana y a Fidel por darme la oportunidad de estudiar y realizar mi sueño de estudiar para poder graduarme.

Mario

Agradecerles a mis padres por el amor, apoyo y confianza que me han brindado desde que nací y por compartir siempre mis sueños, sin ustedes ningunos de mis triunfos hubiesen sido posibles. A mi abuelita Somaida por aguantar mis malcriadeces, a mi abuelo Carlos por su apoyo y cariño, a mi abuela Mima y mi abuelo Pipo porque aunque nos veamos poco siempre están en mi corazón. A la mejor hermana que tengo Camildo por tener que soportarme todos estos años. A mis tías y tíos este triunfo también es de ustedes, a mis primas y primos que tanto nos hemos divertido juntos, a toda la familia en general. A mi hermosa novia por su apoyo en los momentos difíciles, por su amor y dedicación hacia mí. A mi compañero de tesis por su preocupación y tenacidad incansable durante esta batalla. A mis hermanas y hermanos del 7501 con los cuales he compartido momentos fabulosos, a todos mis amigos de la UCI, especialmente a Armando (mandinga) el otro lazo de la corbata, Alejandro (globito) y Alejandro Valdés Pérez (pl), a Evaristo (ecapó) y a Osniel (ogriñaaaannnn).

Carlos

DEDICATORIA

A mi madre, por ser lo más grande que existe.
A mi padre por su amor y cariño.
A mi hermano por tener que cargar conmigo y su cariño, aunque no hable mucho.
A mi hermana por su amor.
A mi primo Robertico.
A mis tías María y Mulata, y mi tío Papito por su apoyo de toda la vida.

Mario

A mi madre y padre por ser mis guías e inspirarme en la vida.
A mi hermana por todo su amor.
A mi abuela Oneida por su cariño y comprensión.
A mis demás abuelos por quererme tanto.
A mis tías, tíos y primos.

Carlos

RESUMEN

El ámbito del presente trabajo de diploma es la gestión del proceso de aprovisionamiento de usuarios, su título es “Gestor de Tareas para Aprovisionamiento de Usuarios”. Surge por la necesidad de facilitar y registrar todo el proceso referente a la creación, modificación, eliminación, habilitación y suspensión de las cuentas de usuarios. Lo novedoso del sistema es que automatiza los procesos anteriormente mencionados de manera centralizada mediante el uso de flujos de aprobación, disminuyendo así la probabilidad de ocurrencia de errores a causa de la mala manipulación del personal de administración, además de proteger la información sensible ante la vista del personal no autorizado resultado de la interacción con el componente de autorización del Sistema de Administración de Identidades.

La investigación estuvo enmarcada fundamentalmente en el estudio del proceso de aprovisionamiento de usuarios en sistemas de administración de identidades. El propósito que se persigue con este trabajo es el desarrollo del módulo Gestor de Tareas para Aprovisionamiento de Usuarios tomando como guía la metodología *Microsoft Solution Framework (MSF)* para el Desarrollo de *Software* Ágil.

Dentro de los resultados más relevantes se encuentra la creación de un sistema capaz de automatizar los procesos de creación, modificación y eliminación de cuentas de usuarios a partir de un mecanismo de solicitudes las cuales deben pasar por un flujo de aprobación antes de materializarse como una acción sobre una cuenta determinada.

Palabras claves: aprovisionamiento de usuarios, administración de identidades, metodologías, servicios, *workflow foundation*.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El acelerado desarrollo que ha experimentado las tecnologías de la información en el nuevo milenio ha cambiado la forma en que las empresas (en este documento, la palabra "empresa" se refiere simplemente a medianas o grandes organizaciones) gestionan sus negocios, con el objetivo de mantenerse a un nivel competitivo dentro del entorno donde se desenvuelven. Es por ello que el crecimiento de las organizaciones es directamente proporcional al buen uso que hagan estas de las tecnologías de la informática y las telecomunicaciones para automatizar procesos claves y costosos dentro de su funcionamiento.

Actualmente las organizaciones modernas ejecutan una compleja mezcla de infraestructura tecnológica para alcanzar el objetivo anteriormente planteado, dentro de la cual convergen un gran número de sistemas como son servidores de aplicaciones donde se encuentran hospedados servidores web y sistemas de base de datos, servidores de correo y mensajería instantánea, directorios de usuarios, así como un gran número de aplicaciones de gestión como son los sistemas para la gestión de nóminas, recursos humanos y para el comercio electrónico, entre otros (la diversidad de aplicaciones dentro de una organización varia en dependencia del negocio al que esta se dedica).

Por otro lado, existen en toda empresa diferente tipos de usuarios que hacen uso de estos sistemas en su trabajo diario, a gran escala estos se pueden dividir en cinco grandes grupos: empleados, contratistas, socios, vendedores y clientes. Para cada uno de los usuarios la mayoría de los sistemas mantienen un registro de cómo estos pueden acceder (usuario y contraseñas) y qué privilegios tienen. Esta información debe ser administrada cuando: un empleado es contratado, el rol que juega dentro del negocio o la información que lo identifica cambia (ejemplo mediante un ascenso o cambio de dirección particular) y cuando concluyen sus servicios de forma temporal (vacaciones) o permanente (retiro, cambio de empleo).

La diversidad de sistemas informáticos dentro de una organización resuelve el gran problema de la automatización pero crea otros no menos complejos ya que cada sistema es independiente en cuanto a interfaz de usuario, gestión de seguridad y procesos administrativos, esto trae como consecuencia que los encargados de administrar las aplicaciones dentro de la infraestructura tengan que gestionar un mismo usuario de diferentes maneras y en diferentes sistemas, fuerza a los usuarios a memorizar credenciales y procesos para autenticarse, además de tornar lento y engorroso el proceso de actualizar la información referente a los perfiles de usuarios y credenciales de autenticación. Todos estos contratiempos elevan los costos, disminuyen la productividad del usuario y crean brechas de seguridad.

La administración de identidades es la rama dentro de las tecnologías de la información encargada de solucionar los problemas expuestos anteriormente, entre las principales áreas que la componen se encuentra la autenticación, la autorización, la auditoría y el aprovisionamiento de usuarios. La autenticación es el proceso mediante el cual un usuario se identifica ante un sistema mediante la combinación de varios factores, los cuales pueden ser algo que solo él conoce (contraseña), algo que solo él posee (*smartcard*) o algo que solo él es (características físicas). La autorización es el proceso de otorgarle a un usuario ciertos privilegios con el objetivo de que pueda acceder a cierta información o realizar determinada función dentro de un sistema. La auditoría es el proceso encargado de revisar periódicamente los privilegios que poseen los usuarios en las diferentes aplicaciones a las que accede con el objetivo de retirarles los que ya no necesite. El aprovisionamiento de usuario es el área encargada de crear, modificar y desactivar cuentas y objetos de usuarios de manera automática.

Entre los principales retos a los que se enfrenta la administración de identidades se encuentran: reducir las demoras que se producen en el momento de otorgarle acceso y privilegios a los nuevos usuarios que se incorporan a las organizaciones o sea, prepararle el entorno propicio para que estos comiencen a producir lo más rápido posible, automatizar el proceso de aprobación para la creación y modificación de las cuentas, convenciendo a los administradores de que lo que se está haciendo es correcto, disminuir el número de usuarios que administran los distintos sistemas de la infraestructura, proveer a los usuarios de un mecanismo para la recuperación de contraseñas olvidadas y proveer mecanismos para la pronta desactivación de las cuentas y privilegios de usuarios cuando estos terminan su ciclo de vida dentro de la organización.

Se puede plantear que tanto en Cuba como en la Universidad de las Ciencias Informáticas el uso de sistemas de administración de identidades casi nulo, principalmente debido a que estos sistemas son complejas y costosas soluciones las cuales solo deben ser utilizadas en medianas y grandes empresas, no sería rentable utilizar estos sistemas en organizaciones que no superan el centenar de empleados debido a que los costos por compra e instalación de la infraestructura necesaria superarían los beneficios que aportan estos. Por la problemática planteada anteriormente y por la necesidad que presenta el país de contar con una solución que gestione las identidades de usuarios en sus organizaciones además de adecuarse a las necesidades que presentan estas, el Centro de Identificación y Seguridad Digital (CISED) se ha planteado el desarrollo del Sistema de Administración de Identidades, el cual estará integrado por cuatro módulos: Autenticación, Autorización, Auditoría y Aprovisionamiento. Debido a esto se plantea el siguiente problema: el módulo de Aprovisionamiento que pertenece al Sistema de Administración de Identidades del Centro de Identificación y Seguridad Digital (CISED) no cuenta con una aplicación que gestione de manera automática la creación, modificación y desactivación de cuentas de usuarios rigiéndose por los flujos de aprobación.

Tomando como base la situación problemática anterior, surge el siguiente **problema científico**: ¿Cómo gestionar de manera automática los procesos de aprovisionamiento de usuarios mediante flujos de aprobación en los sistemas y aplicaciones suscritos al Sistema de Administración de Identidades?

A partir del problema científico se puede determinar como **objeto de estudio** para la investigación: las herramientas de aprovisionamiento de usuarios.

Dicho objeto de estudio deriva en el **campo de acción** que consiste en el aprovisionamiento de los usuarios en las aplicaciones y sistemas de manera automática.

Se plantea como **idea a defender,** que la implementación de un componente que gestione de forma automática los procesos de aprovisionamiento de usuarios mediante flujos de aprobación posibilitará la creación, modificación y desactivación de cuentas de usuarios una vez que estos concluyan satisfactoriamente un proceso de revisión.

El **objetivo general** consiste en implementar una aplicación que gestione automáticamente las tareas de aprovisionamiento de usuarios mediante flujos de aprobación en los sistemas suscritos al Sistema de Administración de Identidades. Los **objetivos específicos** consisten en:

- Crear el gestor de cuentas que se encargará de revisar todos los servidores de cuentas de usuarios, para verificar las que deben ser suspendidas o eliminadas por desuso.
- Crear el gestor de operaciones para organizar en una cola de tareas las solicitudes que llegan al servidor de aprovisionamiento, de acuerdo con la prioridad de las tareas.
- Implementar las tareas que se encargarán de desactivar o eliminar las cuentas de usuarios que se encuentren en desuso.
- Crear un conjunto de pruebas unitarias que validen la implementación utilizando la *suite* incluida en el *Visual Studio. NET 2008 Team System.*
- Integrar el desarrollo al entorno *Team Foundation System*.

Para lograr los objetivos propuestos, se han organizado un grupo de **tareas a realizar** durante la investigación y el desarrollo del sistema:

- Estudio de los sistemas de aprovisionamiento y sus tendencias actuales, que permita fundamentar teóricamente la arquitectura del sistema desarrollar.
- Estudio de los directorios de servicios y sus estructuras, para lograr aprovisionar los usuarios de manera ordenada y distribuida dentro de los directorios de servicios.
- Estudio del protocolo de acceso a directorios (LDAP), que posibilite fundamentar teóricamente que el intercambio de información entre la aplicación en desarrollo y los directorios de servicios se realizará de manera satisfactoria.

- Estudio de las técnicas de envío de datos sobre canales seguros, que fundamente teóricamente que el envío de datos entre la aplicación y los directorios de servicios se realizará de forma segura.
- Implementar una aplicación que gestione de manera automática los servicios de aprovisionamiento de usuario.

Para desarrollar esta investigación y lograr los objetivos planteados se utiliza métodos teóricos y empíricos, mediante los cuales se obtiene una idea más detallada de lo que se quiere lograr. En la investigación se utilizan los siguientes:

Teóricos:

Histórico - Lógico: Se estudió la forma en que se realiza el proceso de aprovisionamiento de usuarios en diferentes sistemas desde sus inicios, obteniéndose los aspectos positivos y negativos del mismo, deduciéndose la necesidad de un sistema que facilitará este proceso.

Análisis - Síntesis: Se analiza la bibliografía encontrada referente al tema en cuestión y se sintetizan los aspectos más importantes para la investigación.

Empíricos:

Observación: Se realizó una exhaustiva observación para ver que sucedía en realidad en el proceso de aprovisionamiento de usuarios en diferentes sistemas y así identificar todos los problemas que se desean resolver.

Teniendo en cuenta el valor metodológico y práctico de este trabajo se obtienen como aportes: mejoras considerables en la Sistema de Aprovisionamiento de Usuarios del Sistema Administración de Identidades.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1

1.1. Introducción.

El aprovisionamiento de usuarios es una de las áreas que integran la administración de identidades. Su objetivo es crear, modificar y desactivar objetos y atributos de usuarios (cuenta de usuario, privilegios) los cuales se encuentran dispersos a través de múltiples sistemas, de forma rápida, barata y confiable. Esto es posible mediante la codificación de diferentes procesos de negocios tales como el proceso de entrada de nuevos empleados a la organización y la salida de estos de la misma.

Un sistema de aprovisionamiento de usuario funciona automatizando uno o más de los siguientes procesos: sincronización de identidades, este proceso detecta cambios en la información en el perfil de un usuario en un sistema y automáticamente lo replica hacia otros sistemas. Auto aprovisionamiento, detecta la entrada de nuevos usuarios en los sistemas de recursos humanos y automáticamente crea cuentas para esos usuarios con los privilegios adecuados en otros sistemas y aplicaciones. Auto desactivación, detecta la eliminación de usuarios en los sistemas de recursos humanos y automáticamente desactiva las cuentas de usuarios en todos los sistemas y aplicaciones. Servicios de solicitud para usuarios, permite a los usuarios actualizar la información de sus propios perfiles y solicitar nuevos privilegios. Delegar administración de usuarios, posibilita que los propietarios del negocio e interesados (*stake-holders*) modifiquen los privilegios de usuarios dentro del ámbito de autoridad que posean. Flujos de aprobación, validan todos los cambios propuestos por los usuarios mediante el servicio de solicitud y permite que los propietarios e interesados del negocio aprueben estos cambios antes de materializarse en los sistemas y aplicaciones. Reportes, provee información acerca de los privilegios que posee un usuario, además de información acerca de las cuentas inactivas o huérfanas.

Una de las técnicas que más se utiliza para modelar una estrategia de aprovisionamiento es la clásica basada en roles, con esta estrategia a cada usuario se le asigna un rol el cual posee determinados privilegios, lo que posibilita que varios usuarios que pertenezcan a un mismo rol compartan los mismos privilegios, esta estrategia funciona correctamente cuando se gestiona un único sistema. En organizaciones que posean una mayor infraestructura de software así como una mayor cantidad de empleados, esta estrategia traería como consecuencia que existieran más roles que usuarios debido a que existen usuarios en toda organización que poseen un rol único ya que realizan funciones que nadie más realiza, además un mismo usuario puede poseer privilegios diferentes en distintos sistemas, lo que trae como consecuencia que se tengan que refinar los roles que ya existen, duplicándose estos.

Para resolver el reto que impone la estrategia basada en roles se ha planteado el uso de una estrategia de aprovisionamiento basada en solicitudes, las cuales pueden ser creadas mediante aplicaciones web, correo electrónico o directamente cuando se produce un cambio en los sistemas de gestión de recursos

humanos. Esta estrategia posibilita asignarle determinados privilegios a un usuario para cierta aplicación en el momento que este lo necesite resolviéndose así los problemas que plantea el aprovisionamiento basado en roles. Una de las pocas deficiencias que presenta este modelo es la acumulación excesiva de privilegios por parte de un usuario ya que en toda organización hay un constante cambio en las actividades que realiza un empleado. Este problema es resuelto por el módulo de auditoría explicado con anterioridad en el presente trabajo.

1.2. Seguridad en sistemas informáticos.

La seguridad de la información es un requisito tanto ético como legal dentro de cualquier contrato entre clientes y grupos de desarrolladores. Consiste en lograr que la información sea confiable, íntegra y que esté disponible desde donde esté establecido consumirla y cuando esté establecido hacerlo. Para lograr este objetivo es necesario crear para los usuarios (sistemas informáticos o personas) que consumen dicha información una serie de políticas y privilegios. De estas tareas se encargan lo sistemas de administración de identidades, integrados estos por componentes análogos a los cortafuegos presentes en numerosos sistemas operativos, los cuales otorgan o deniegan el acceso a procesos que lo requieren en dependencia de reglas o privilegios que estos posean. Es por ello que a pesar de ser los sistemas de administración de identidades como su nombre lo indican sistemas de gestión, indudablemente son aplicaciones donde el principal objetivo que se persigue es incrementar extensiblemente la seguridad de la información.

1.3. Lenguaje de Marcado para Servicios de Aprovisionamiento (SPML v2).

Es un estándar basado en el lenguaje de marcado extensible (XML) desarrollado por el Comité Técnico de Servicios de Aprovisionamiento de la (OASIS), con el objetivo de intercambiar información sobre usuarios, recursos y servicios de aprovisionamiento a través de la puesta en marcha de mecanismos que posibilitan la creación de peticiones de aprovisionamiento a través de múltiples sistemas y organizaciones, permitiendo así la cooperación entre organizaciones en cuanto a servicios de aprovisionamiento se refiere. Entender los conceptos que propone este estándar ayuda en gran medida a comprender como funciona el proceso de aprovisionamiento.

SPML v2 propone cuatro roles básicamente: autoridad solicitante (AS), proveedor de servicios de aprovisionamiento (PSA), objetos del servicio de aprovisionamiento mayormente denominados objetos (O) y objetivos del servicio de aprovisionamiento (OSA). El siguiente diagrama muestra las relaciones básicas entre estos elementos.

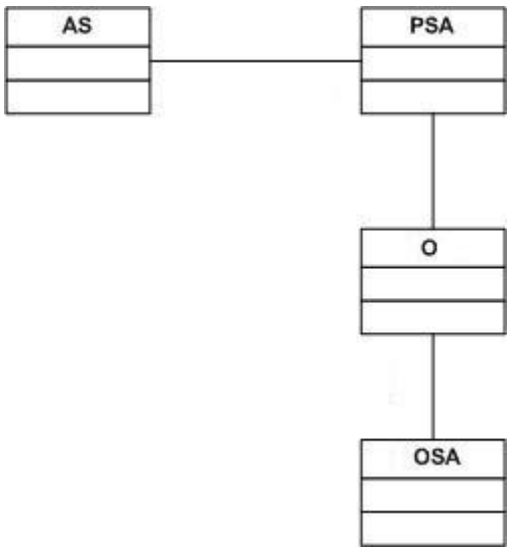


Figura 1. Relaciones entre los elementos que componen el modelo de dominio del estándar SPML v2.

El diagrama expuesto con anterioridad plantea las siguientes interacciones: Una autoridad solicitante (AS) envía una petición SPML bien formada hacia un proveedor de servicios de aprovisionamiento (PSA), esta petición es materializada en un objetivo del servicio de aprovisionamiento (OSA) mediante la acción sobre un objeto (O).

Adentrándonos en una definición más formal de los elementos que propone este modelo se entiende como autoridad solicitante cualquier sistema informático que envíe solicitudes SPML bien formadas hacia un proveedor de servicios de aprovisionamiento, ejemplo: aplicación web que permita la creación, modificación o eliminación de una cuenta de usuario. El proveedor de servicios de aprovisionamiento es un sistema informático que atiende solicitudes SPML enviadas por una autoridad solicitante conocida y envía una respuesta SPML hacia esta, ejemplo: un sistema de administración de identidades. Los sistemas objetivos del servicio de aprovisionamiento constituyen contenedores de objetos de aprovisionamiento además representan los destinos de las acciones de aprovisionamiento realizadas por un proveedor de servicios, ejemplo: directorio activo. Los objetos del servicio de aprovisionamiento representan una entidad de datos u objetos de información contenidos dentro de un sistema objetivo, ejemplo: una cuenta de usuario dentro de un directorio activo.

1.4. Historia de los sistemas de identidades.

Los sistemas de administración de identidad surgieron con el fin de satisfacer dos necesidades de primer impacto en el negocio de las organizaciones, el incremento de la seguridad y el ahorro de dinero. En cuanto las empresas empezaron a recurrir a la automatización en un esfuerzo por ir más de prisa, reducir los costos y tener mejores resultados, la necesidad de proteger las transacciones se hizo más importante. En un inicio estos sistemas comenzaron como herramientas muy sencillas, las cuales

resolvían solo el problema de la gestión de la identidad a niveles de intranet y aunque automatizaban algunos procesos vitales de aprovisionamiento requerían un constante chequeo por parte de los encargados de administrarlos debido a que los motores de control de políticas no habían evolucionado para convertirse en lo que son hoy: potentes herramientas que velan por que se cumplan las políticas de la organización. En la actualidad un administrador de identidad es una poderosa herramienta la cual cuenta con componentes que han evolucionado y que gestionan hasta los más mínimos detalles en el ciclo de vida de un usuario dentro de la organización. Han surgido estándares para el intercambio de información relacionada con la identidad del usuario entre sistemas independientes, lo que hace unas décadas era impensable. En la actualidad se pretende que estos sistemas manejen mecanismos biométricos para la autenticación de los usuarios característica esta que está siendo implementada por empresas destacadas en el sector como *Microsoft*.

1.5. Análisis comparativo de los Sistemas de Aprovisionamiento.

En la actualidad los sistemas de aprovisionamiento constituyen componentes integrados a sistemas de administración de identidades, los cuales continúan madurando en cuanto a funcionalidad y capacidad.

Funcionalidades básicas de provisión (motores de flujo de trabajo, los procesos de aprobación, la gestión de contraseñas y los conectores estándares), se pueden encontrar en la mayoría de las soluciones que propone el mercado. La competencia entre los proveedores de estos sistemas por encontrar nuevas características que los sitúe en una posición ventajosa con respecto a otros proveedores, es atroz. Aunque se evidencia un patrón en cuanto a las funcionalidades en estos sistemas, existen en ellos características que los posicionan a un mayor o menor nivel en cuanto a dominio del mercado se refiere. Dominando el entorno de los sistemas de aprovisionamiento se encuentran *Oracle* e *IBM*, dos gigantes del sector que comparten la cima, seguidos por *Novell* y *CA* las cuales también poseen una presencia significativa y perdiendo terreno en los últimos años, pero aún entre los líderes se encuentra *Sun Microsystems (*que paso a forma parte de *Oracle* hace menos de tres meses).

Siempre que se entra al mundo de las comparaciones, más allá de comparar productos se debe tener presente las necesidades inmediatas que se deseen solventar para así asegurarse de que estemos tomando el camino correcto. Es por ello que el producto más adecuado es el que resuelva nuestras insuficiencias y no precisamente aquel que posee una mejor posición en el mercado, aunque este factor es directamente proporcional con los factores: calidad y fiabilidad. Es por ello que antes de hacer un análisis comparativo de los sistemas de aprovisionamiento existentes se expondrá información de los sistemas punteros en esta rama.

- **IBM Tivoli Identity Manager (ITIM) v.5.1** (4)**:** Con este producto IBM ha afianzado su dominio del mercado, el cual se ha mantenido durante los últimos 10 años consolidándolo como uno de los proveedores de mayor experiencia en el área. ITIM es desplegable en la mayoría de los sistemas operativos modernos incluyendo z/OS[1]. Las tecnologías de flujos de trabajos relacionadas con el aprovisionamiento y la aprobación son relativamente completas y ofrece además una amplia gama de conectores así como una competitiva administración de funciones de *password*. Este sistema provee además un Simulador de Políticas el cual ayuda a los usuarios a simular roles y políticas de aprovisionamiento para analizar el comportamiento que estas pueden tener sobre el entorno de producción antes del despliegue del producto. Con respecto a la versión anterior de este producto IBM ha incrementado las facilidades de uso para los usuarios así como la integración de componentes a la plataforma, reflejándose estos últimos aspectos en la retroalimentación que le proveen los usuarios a la compañía.
- **Oracle Identity Manager v.9.1.0.2** (5)**:** Oracle ofrece gran consistencia y fiabilidad con cada versión que libera, características estas que colocan a sus productos en la cúspide del mercado. Esta última versión de la plataforma puede ser desplegada sobre dos bases de datos diferentes, sobre siete sistemas operativos, cuatro servidores de aplicación y múltiples plataformas de desarrollo de Java lo que evidencia la alta escalabilidad de sus productos. Su repositorio de identidad es altamente escalable, bien diseñado y probado. *Oracle Identity Manager* puede ser integrado a servicio de prueba lo que minimiza el riesgo basado en las propias decisiones que toma el usuario. Permite el acceso a los equipos de desarrollo de Oracle, para realizar cambios y configuraciones durante el proceso de despliegue. Las capacidades del flujo de trabajo de los motores de aprovisionamiento han sido recientemente incrementadas y nuevas librerías de conectores adicionadas.
- **Sun Identity Manager v.8.1** (6)**:** Esta versión provee nuevas características que mantienen a la compañía entre los líderes a pesar de haber perdido terreno en el último año, por la falta de confianza de los usuarios ante la compra de la empresa por Oracle. Algunas de las nuevas características de este producto son la Administración Externa de Recursos, la cual permite administrar las funcionalidades de aprovisionamiento y aprobación de aplicaciones que no están directamente conectadas al *Identity Manager*. La nueva línea de Conectores permite una nueva forma de conectar el *Identity Manager* con los recursos a los que administra; estos conectores son liberados de manera independiente al producto,

[1] **Z/OS:** Es el sistema operativo actual de los mainframes de IBM.

aumentado así la frecuencia con que surgen nuevas versiones. Se ha agregado el soporte de SPML 2.0 para realizar búsquedas y aprovisionar otros sistemas. Se ha adicionado además la Extensión de Administración de Java la cual provee al sistema de un mayor rendimiento a la hora de realizar operaciones de listado, creación, modificación, eliminación, obtención y autenticación de objetos. La integración con los administradores de roles de Sun permite al sistema invocar directamente servicios Web de administración de roles para notificar e invocar operaciones de roles realizadas sobre usuarios. En cuanto a la seguridad, el sistema brinda la posibilidad de usar el Estándar Avanzado de Encriptación (AES)[2] para el cifrado de contenidos en sustitución del Estándar de Encriptación de Datos (DES)[3]. Una de las nuevas características del sistema es el uso de las Firmas Digitales de XML, *Extensible Markup Language* (Lenguaje de Marcas Extensible), usando el mecanismo de no repudio utilizado por la W3C.

- **Novell Identity Manager Roles Based Provisioning Module v.3.6.1** (7)**:**
  Desarrollada por la empresa Novell la cual está posicionada entre los cinco líderes del mercado de la gestión de identidad, es una plataforma que permite a las organizaciones gestionar de forma completa el ciclo de vida de los usuarios desde que entran en la organización hasta que terminan de brindar sus servicios. Incluye capacidades para brindar servicios de aprovisionamiento, así como el servicio para deshabilitar dicho aprovisionamiento de las cuentas de usuario, flujos de trabajo de aprobación, la gestión de contraseñas y gestión de datos en los directorios de usuario de su organización, aplicaciones, bases de datos y plataformas de sistema operativo. A través de la administración de usuario simplificada y procesos, Identity Manager ayuda a las organizaciones a reducir los costes de gestión, aumentar la productividad y la seguridad, y cumplir con las regulaciones gubernamentales.
  Identity Manager incluye módulos de integración para varios sistemas comunes de los clientes: Novell eDirectory, *Microsoft Active Directory*, directorios LDAP, Novell GroupWise, y Lotus Notes. Esta plataforma proporciona a los usuarios los recursos validados en tiempo real para garantizar el cumplimiento de las políticas.
- **CA Identity Manager Release** (8)**:** Junto con las cuatro compañías antes mencionadas, CA conforma la lista de los cinco punteros en el mercado de la gestión de identidad. Esta empresa obtuvo sus mayores progresos en el 2009 y aunque no ha introducido cambios

[2] Advanced Encryption Standard (AES): también conocido como Rijndael es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos.
[3] Data Encryption Standard (DES): es un algoritmo de cifrado escogido como en los Estados Unidos.

significativos desde entonces sigue teniendo progresos en el sector. Esta última versión es una plataforma centralizada que provee de una automatización basada en flujos de trabajo para la administración de identidades de usuarios tales como identificadores y contraseñas a través de múltiples recursos. También permite la creación de cuentas de accesos o solicitudes de cambios de contraseña, a través de peticiones atomizadas, llevadas a cabo por personal administrativo o por el propio usuario. Estas peticiones disparan flujos de trabajos administrativos contenidos dentro del *Identity Manager* los cuales culminan el proceso automatizado de aprovisionamiento a través de múltiples recursos en la red. Esta plataforma es capaz de manejar la gestión la identidad sobre múltiples sistemas tales como *Windows* NT/2000/2003, Solaris[4], HP-UX[5], AIX[6] y Linux, directorios específicos como *Active Directory*, estándares como LDAP y ODBC y base de datos tales como IBM DB/2, Oracle y *MS SQL Server*. Dentro de las características que la propia compañía destaca, está la capacidad de delegar tareas administrativas a usuarios con privilegios especiales, el servicio de usuario para la administración de perfiles, contraseñas, incluyendo específicas habilidades para la administración de contraseñas, como son el soporte para el olvido de estas y la sincronización bidireccional de contraseñas.

Aunque todos los sistemas mencionados anteriormente son complejas plataformas con decenas de componentes integrados, las diferencias entre ellos indudablemente son menores que las características que los hacen similares. Claramente, todo objeto que pertenece a un grupo, a una categoría, presentan patrones que lo identifican como miembro de ese grupo o poseedor de esa categoría y los sistemas de aprovisionamiento no son la excepción.

Dentro de las características básicas de estos sistemas se encuentra:

- **El Servicio de Aprovisionamiento:** permite a los usuarios, aplicaciones, sistemas o dispositivos ser identificados y administrados automáticamente por la organización.
- **El Sistema de Autenticación:** es un servicio que verifica y rastrea la identidad del usuario y ocasionalmente de aplicaciones dentro de la organización.
- **El Sistema de Administración de Acceso:** es usado para controlar el acceso a los recursos dentro de la organización tales como archivos, aplicaciones, sistemas y dispositivos. Los servicios de administración de acceso se basan fundamentalmente en el

[4] Solaris es un sistema operativo compatible con el estándar POSIX y la Open System Specification.
[5] HP-UX es la versión de Unix desarrollada y mantenida por Hewlett-Packard ejecutable típicamente sobre procesadores HP PA RISC.
[6] AIX (Advanced Interactive eXecutive) es un sistema operativo UNIX System V propietario de IBM.

uso de roles y sistemas de evaluación de reglas para otorgar o denegar el acceso a los objetos en la organización.

- **El Sistema de Auditoria:** es usado para registrar la actividad que ocurre dentro de la infraestructura. La auditoría básica incluye el rastreo de los intentos de autenticación y acceso.
- **Los Servicios de Directorios:** servicios usados por las herramientas de autenticación, aprovisionamiento y acceso, así como las aplicaciones de la organización como un directorio que actúa como repositorio de información acerca de los objetos que la organización está gestionando.

1.6. Motor de Aprovisionamiento.

El Motor de Aprovisionamiento es uno de los componentes básicos de un sistema de aprovisionamiento de usuarios. Este sistema está destinado a ayudar a las organizaciones a agilizar los procesos del ciclo de vida de los usuarios, actualizando cada cambio que ocurre respecto a los objetos relacionados con los usuarios de forma automática. Para alcanzar los objetivos antes mencionados, el motor de aprovisionamiento debe realizar unos o más procesos, los cuales pueden ser:

- **Auto-Aprovisionamiento:** Es el proceso encargado de la creación automática de usuarios en sistemas y aplicaciones cuando estos hacen su entrada en la organización.
- **Auto-Desactivación:** Proceso que desactiva objetos de usuarios, cuando estos ya no brindan sus servicios a la organización o cuando son marcados como inactivos. Desactiva automáticamente usuarios u objetos de usuarios, cuando el tiempo de cumplimiento de una tarea ha concluido.
- **Sincronización de Identidad:** Cuando son realizados cambios en los atributos de un usuario, automáticamente este proceso replica dichos cambios en los sistemas o aplicaciones pertinentes.
- **Auto-Servicio de Cambio de Perfil:** Permite a los usuarios actualizar su información de perfil.
- **Auto-Servicio de Petición de Acceso:** Permite a los usuarios solicitar acceso a sistemas y aplicaciones.
- **Delegar petición de Acceso:** Permite a administradores realizar solicitudes de accesos a sistemas o aplicaciones en nombre de sus subordinados.
- **Flujo de Trabajo de Autorización:** Proceso que permite a los titulares del negocio revisar y aprobar o rechazar cambios propuestos a los perfiles de usuarios o a los derechos de acceso del mismo.

- **Certificación de Acceso:** Este proceso pide periódicamente a los administradores que chequeen la lista de sus subordinados inmediatos, para verificar que aún son empleados de la organización o que aún son sus subordinados. Pide periódicamente a los propietarios de datos o aplicaciones que chequeen la lista de acceso de usuarios.

Los componentes básicos para llevar a cabo estos procesos son:

- **Conectores:** Es el componente encargado de conectarse a los sistemas o aplicaciones para obtener o enviar información (crear usuarios, eliminar, modificar. Obtener información del usuario).
- **Base de Datos Interna:** Se encarga de persistir información relacionada con los objetos de usuario así como datos de aplicaciones o sistemas.
- **Interfaz de Usuario:** Permite a los usuarios revisar el contenido de las solicitudes realizadas así como realizar cambios, aprobar o rechazar propuestas de cambio.
- **Motor de Flujo de Trabajos:** Usado primariamente para que los usuarios revisen y aprueben o rechacen cambios propuestos.
- **Motor de Políticas:** Evalúa la información actual del usuario y los cambios propuestos, para chequear que estos cumplen con las políticas de la organización.
- **Motor de Reportes:** Provee a la organización de un extracto de la información proveniente de la Base de Datos Internas.

1.7. Motor de tareas.

El Motor de Tareas es el componente encargado de procesar una petición para convertirla en un procedimiento el cual invoca una funcionalidad en un proveedor. Las entidades claves de este componente son las peticiones, los procedimientos y los proveedores. Las peticiones son el punto de partida de todos los servicios de aprovisionamiento y son documentos XML que pueden ser enviadas al motor de tareas mediante servicios Web. Cada petición contiene la llamada a un procedimiento, el cual a su vez, debe invocar las funcionalidades correctas para satisfacer la petición. Los procedimientos son el núcleo de todas las transacciones de aprovisionamiento. Los proveedores son la fuente de todas las funcionalidades requeridas para soportar todos los servicios de aprovisionamiento. El motor de tareas usa los proveedores para ejecutar peticiones. Los proveedores son objetos que brindan las interfaces funcionales para la comunicación con componentes externos. A grandes rasgos sin entrar en detalles la arquitectura de un motor de tareas está basada en estas tres entidades.

Antes de abordar el tema acerca de las características que debe tener un motor de tareas se debe tener en cuenta que este componente es el encargado de cumplir las solicitudes realizadas por los usuarios

de la organización de forma transparente a los mismos. O sea, que es el último componente por el que fluyen las solicitudes antes de materializarse como un hecho en un sistema o aplicación externa.
Es por ello que cuando un usuario envía una petición al motor de tareas, la probabilidad de que esta no se gestione debe tender a cero y, en caso de no gestionarse, debe haber una respuesta indicando los problemas por los cuales no se pudo gestionar dicha solicitud, de manera que una de las características de un motor de tareas debe ser la capacidad del sistema de recuperarse ante fallos por lo que el uso de mecanismo de transacciones es una práctica recomendada en este entorno. Otra característica de importancia son las trazas que debe realizar el sistema ante cualquier acción que se lleve a cabo, las cuales permiten posteriormente tener información de las transacciones realizadas. La prioridad que se les debe dar a las peticiones es también una característica con que debe contar el motor de tareas, se deben procesar las peticiones de acuerdo con la importancia que poseen estas para la organización, por lo que debe haber cola para las peticiones con importancia baja y un mecanismo que procese en tiempo real las peticiones de alta importancia.

1.8. Tecnología y herramientas a utilizar.

A continuación se describen las herramientas de las que se harán uso en el desarrollo de la aplicación.

1.8.1. El Lenguaje Unificado de Modelado (UML) (9).

El Lenguaje Unificado de Modelado (UML) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema.
Los principales tipos de Diagramas UML existentes son:

- **Diagramas de estructura estática:** Describen las propiedades estructurales del sistema.
  Diagrama de clases: Conjunto de clases, interfaces y colaboraciones; así como sus colaboraciones.
  Diagrama de objetos: Conjunto de objetos y sus relaciones.
  Diagrama de casos de uso: Conjunto de casos de uso y actores y sus relaciones.
- **Diagramas de comportamiento:**
  Diagramas de interacción (secuencia y colaboración): Objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos.
  Diagrama de estados: Muestra una máquina de estado que consta de estados, transiciones, eventos y actividades.
  Diagrama de actividad: Es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.
- **Diagramas de Implementación:**
  Diagramas de despliegue.

Diagrama de componentes.

Se usa para entender, hojear, controlar, configurar y mantener información sobre el sistema en desarrollo.

UML proporciona la capacidad de modelar actividades de planificación de proyectos y de sus versiones, expresar requisitos y las pruebas sobre el sistema, representar todos sus detalles así como la propia arquitectura. Mediante estas capacidades se obtiene una documentación que es válida durante todo el ciclo de vida de un proyecto. Entre sus características destacan:

- ✓ Permite el modelado orientado a objetos.
- ✓ Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- ✓ Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- ✓ Es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.
- ✓ Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

1.8.2. Herramienta de modelado *Altova UModel* 2009.

Altova UModel 2009 (10) es utilizado como el punto de partida para el desarrollo de un software. Diseña visualmente modelos de aplicaciones en UML y genera código *Java, C sharp, o Visual Basic .NET* y documentación del proyecto. Es una herramienta UML que hace el diseño visual de software práctico para cualquier proyecto.

Las características de *UModel* 2009 para el desarrollo de software basado en las capacidades de modelado avanzado son:

- ➢ Soporte para los 14 tipos de diagramas UML.
- ➢ Modelado de esquemas XML en diagramas UML.
- ➢ Diagramas de proceso de negocio (BPMN)**.**
- ➢ Generación de código fuente en lenguajes Java, C sharp, y VB.NET.
- ➢ Ingeniería inversa de código fuente y ficheros binarios Java, C sharp y VB.NET.
- ➢ Sincronizado de modelo y código a través de ingeniería de ida y vuelta.
- ➢ Crea diagramas de secuencia desde el código fuente de la ingeniería inversa.
- ➢ Generación de documentación personalizable de proyecto.
- ➢ Compartir subproyectos para colaboración o reutilización.
- ➢ Capas de diagramas con visibilidad selectiva.

- Hyperlinks entre diagramas, documentos, o páginas web.
- Soporte para intercambio de modelos XMI 2.1.
- Integración con sistemas de control de versiones.
- API extendida para permitir manipulaciones externas.
- Estrecha integración con *Visual Studio* y Eclipse.

1.8.3. Metodología de desarrollo *MSF Agile* (11).

Las metodologías de desarrollo de software no son más que procedimientos o técnicas que te ayudan a documentar el proceso de desarrollo del software. No existe metodología mejor que otra, cada una tiene sus propias características y se enfocan en funciones específicas. Las metodologías se clasifican en ágiles y tradicionales. Las ágiles son aquellas donde el desarrollo de software se caracteriza por ser incremental, cooperativo, sencillo y adaptable, entre ellas se encuentran XP, SCRUM, MSF (*Microsoft Solution Framework*), entre otras. Las tradicionales son metodologías que se centran fundamentalmente en el control del proceso, además son las más efectivas para proyectos de gran tamaño. Las metodologías tradicionales más populares son OPEN, METRICA 3 y UP (*Unified Process*), pero la más utilizada por su eficiencia y beneficios es UP conocida como RUP. El trabajo estará centrado en la metodología MSF.

***MSF Agile*:** Es una metodología flexible y que está estrechamente ligada a un conjunto de modelos, conceptos y prácticas de uso que permiten gestionar, planificar y controlar los proyectos tecnológicos. MSF se centra en los modelos de equipo y de proceso y deja en un segundo plano las elecciones tecnológicas. Entre sus principales características se encuentran:

- ✓ Adaptable.
- ✓ Escalable: puede organizar equipos tan pequeños como de 3 ó 4 personas, así como también, proyectos que requieren 50 personas o más.
- ✓ Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.
- ✓ Tecnología agnóstica: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

Está compuesta por varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el Modelo de Aplicación.

- Modelo de arquitectura del proyecto: diseñado para acortar la planificación del ciclo de vida.
- Modelo de equipo: diseñado para mejorar el rendimiento del equipo de desarrollo.
- Modelo de proceso: diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega.

- Modelo de gestión del riesgo: diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir.
- Modelo de diseño del proceso: diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario.
- Modelo de aplicación: diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores.

1.8.4. Tecnología .NET.

Microsoft .NET es un conjunto de tecnologías sobre las cuales *Microsoft* ha venido trabajando en los últimos años, muchas veces de manera aislada y dispersa, a las cuales ha querido unir en un solo paquete. El objetivo de *Microsoft* con el desarrollo de .NET es el de brindar una plataforma de desarrollo sencilla de usar, potente en la distribución de servicios, que permita la comunicación entre aplicaciones independientemente del lenguaje en el que se han desarrollado.

.NET ofrece un entorno de desarrollo de aplicaciones llamado *Visual Studio .NET* que consta de varios lenguajes de programación como *Visual Basic .NET, Visual C sharp, Visual FoxPro y Visual C++ .NET*. Estos lenguajes combinan las características de los lenguajes existentes con nuevas posibilidades para proporcionar un potente sistema de desarrollo. A continuación, se detallan algunas de las características de la Arquitectura .NET (*Microsoft, Visual Studio Team System*).

1.8.5. Propiedades de la plataforma .NET.

Portabilidad: es la capacidad que posee un sistema de ser utilizado en varias plataformas. *Microsoft* continúa con la voluntad de apoyar a su sistema *Windows*, por lo que *Microsoft .NET* funciona solamente en plataformas basadas en *Win32*[7].

Escalabilidad: es la propiedad que posee un sistema de incrementar sus servicios en función del aumento del número de usuarios que lo utilicen. *Microsoft .Net* ofrece métodos de escalabilidad como la carga balanceada, que permite colaborar a varios servidores y satisfacer la petición de un servicio de manera simultánea. *Microsoft .NET* ofrece una solución más barata, con mayor rendimiento, escalabilidad y más fácil de implantar.

[7] **Win32*:*** Versión del API de Windows de 32 bits. Está compuesta por funciones en C almacenadas en librerías de enlace dinámico (DLL).

1.8.6. Framework .Net 3.5.

Es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables.

.NET Framework 3.5 engloba todas las características de las versiones anteriores, ya que la actualización de la versión se realiza de manera incremental, y además otras muchas nuevas como:

- ✓ Integración total de LINQ (*Language Integrated Query*) y del reconocimiento de los datos.
- ✓ Nueva compatibilidad con el protocolo Web para generar servicios WCF (*Windows Communication Foundation*).
- ✓ Aplicaciones web más interactivas, gracias a *ASP.NET AJAX*.
- ✓ Compatibilidad absoluta con las herramientas de *Visual Studio 2008* para WF[8], WCF[9] y WPF[10].
- ✓ Nuevas clases en la biblioteca de clases base.

1.8.7. Lenguaje de programación *C sharp*.

Los lenguajes de programación son herramientas que permiten crear programas y *software*. Los programas no son otra cosa que una colección de funciones que son llamadas sucesivamente por la función principal y única del programa; cada función tiene sus propias variables, es un módulo independiente. La Programación Orientada a Objetos es el próximo paso en la evolución de estos lenguajes, ya que hace combinaciones de funciones y datos, en una unidad auto consistente llamada clase, además de incorporar la herencia que permite organizar las clases jerárquicamente.

C sharp combina los mejores elementos de múltiples lenguajes de amplia difusión como C++, Java, *Visual Basic* o Delphi. Con ello quiere aprovechar el alto poder de abstracción de C++ y C con sus amplias posibilidades de aprovechamiento al más bajo nivel. Por otro lado, también pretende limar algunas deficiencias que posee Java en cuanto a portabilidad y eficiencia. *Microsoft* puso gran énfasis en que la migración de los programadores a este lenguaje, que forma parte de su plataforma .NET y que está ampliamente sustentado por el *Framework .NET*, además de ser considerado el lenguaje nativo de la plataforma .NET, fuera lo más fácil posible. Debido a ello la sintaxis y estructuración del código en C sharp es semejante al de C++, mientras que su sencillez y alto nivel de productividad se igualan al de *Visual Basic.*

[8] Windows Workflow Foundation: es una tecnología de Microsoft para definir, y ejecutar, flujos de trabajo de gestión.

[9] Windows Communication Foundation: Nueva plataforma de mensajería que forma parte de la API de la Plataforma .NET 3.0.

[10] Windows Presentation Foundation: Una de las nuevas tecnologías de Microsoft, y uno de los pilares de Windows Vista. Permite el desarrollo de interfaces de interacción en Windows tomando las mejores características de las aplicaciones Windows y de las aplicaciones web.

1.8.8. Características del lenguaje *C sharp*.

Eficiencia: Restringe el uso de punteros a la necesidad del programador e incluye un conjunto de restricciones que brinda mayor seguridad al código.

Modernidad: C sharp incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrando, que son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++, hay que simular.

Orientado a objetos: Como todo lenguaje de programación de propósito general actual, C sharp es un lenguaje orientado a objetos. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++, es que el de C sharp es más puro, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.

Orientado a componentes: La propia sintaxis de C sharp incluye elementos propios del diseño de componentes, que otros lenguajes tienen que simular mediante construcciones más o menos complejas.

Seguridad de tipos: C sharp incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente.

Gestión automática de memoria: Todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR[11]. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos.

1.8.9. Herramienta de desarrollo Visual Studio Team System 2008.

Las Herramientas de Desarrollo de Software o Entorno de Desarrollo Integrado (IDE) permiten que los programadores implementen de una forma concreta y organizada. Estas incluyen técnicas de soporte para detectar y mostrar errores en la programación y facilitan el completamiento de código, todo esto agiliza el desarrollo del *software*. Algunas traen ejemplos de códigos y documentación para lograr un mejor entendimiento en los programadores.

1.8.10. Visual Studio Team System 2008.

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE) de *Microsoft*. Soporta un diverso grupo de lenguajes de programación tales como *Visual C++.NET, Visual J#.NET y Visual Basic.NET y Visual C sharp.NET*. Provee amplias facilidades como la inclusión de un editor de código que soporta

[11] Common Language Runtime, componente de máquina virtual de la plataforma .Net de Microsoft.

resaltado de sintaxis y de código utilizando *IntelliSense*[12], que comprende variables, funciones y métodos, construcciones del lenguaje, como los bucles y las consultas brindando así la posibilidad de escribir código de alta eficiencia.

Microsoft Visual Studio Team System 2008 es una plataforma para herramientas del ciclo de vida del desarrollo de *software* extensible, integrado y productivo, que ayuda a los equipos de desarrollo de *software* mediante la mejora de las comunicaciones y la colaboración durante todo el proceso de desarrollo.

1.8.11. Sistema Gestor de Base de Datos *Oracle 11g*.

La Base de Datos *Oracle* es ampliamente utilizada en diversos tipos de grandes aplicaciones, y es mayormente popular porque las características que la definen, apenas dejan ver desventajas:

Versatilidad: la Corporación Oracle ofrece muchos productos de comercio electrónico que integran con su base de datos, lo que permite acelerar el proceso de diseño y construcción de la aplicación que utiliza esta base de datos.

Estabilidad: los administradores reportan que los servidores de Base de Datos de Oracle rara vez fallan, óptimo si se tiene una aplicación que requiere un funcionamiento a tiempo completo.

Disponibilidad de Interfaz de Usuario Gráfica (en lo adelante GUI): Oracle ofrece muchas herramientas con GUI para gestionar el servidor de base de datos.

Seguridad: las versiones actuales de *Oracle* incluyen un conjunto de herramientas que permiten la encriptación de datos sensibles dentro de la base de datos. También provee, al igual que otros gestores, seguridad a nivel de usuario, para proteger la información de ataques de usuarios malintencionados o errores de los administradores.

Soporte y mantenimiento: la compañía *Oracle* se destaca históricamente por responder a las solicitudes de usuarios de integrar nuevas características; también responde a las tendencias del mercado, manteniendo siempre la documentación disponible a su red de usuarios.

Multiplataforma: versiones populares incluyen *Oracle*, tanto para *Windows* como para Linux, aún para este último, las versiones son privativas.

Puede acotarse como desventaja que es un SGBD que para su correcto funcionamiento requiere del consumo de altos recursos de *hardware*, dígase velocidad del procesador y capacidad de RAM.

[12] Es la aplicación de autocompletar, mejor conocido por su utilización en Microsoft Visual Studio entorno de desarrollo integrado. Además de completar el símbolo de los nombres que el programador está escribiendo, IntelliSense sirve como documentación y desambiguación de los nombres de variables, funciones y métodos de utilización de metadatos basados en la reflexión.

1.8.12. *Windows Workflow Foundation (WWF).*

Es la tecnología de *Microsoft* para crear aplicaciones y soluciones que requieren contar con un flujo de trabajo coordinado y transparente. Con esta tecnología que Microsoft pone a disposición, se puede entender cómo utilizar la lógica de negocio en un componente que permite diseñar y controlar el flujo de una aplicación. Lo que *Windows Workflow Foundation*, entre otras cosas ofrece, es abstraer la lógica del proceso en un componente adicional que cuenta con una serie de servicios y está encargado de coordinar el flujo de la ejecución de todo un programa permitiendo tener bien definidos mecanismos que puedan ofrecer el estado apropiado y la transparencia de un proceso.

1.8.13. *XML*.

XML, siglas en inglés de *Extensible Markup Language* (Lenguaje de Marcas Extensible), es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). Es una simplificación y adaptación del SGML[13] (*Standard Generalized Markup Language* o Lenguaje Estándar Genérico por Etiquetas) y permite definir la gramática de lenguajes específicos como son XHTML[14], SVG[15] y MathML[16]. XML, proporciona un estándar de datos que puede codificar el contenido, la semántica y los esquemas de una gran variedad de casos, desde los más simples a los más complejos, sirve para marcar lo siguiente:

- ✓ Un documento normal.
- ✓ Un registro estructurado, como un registro de citas o un pedido de compra.
- ✓ Un objeto con datos y métodos, como el formulario permanente de un objeto Java o de un control ActiveX.
- ✓ Un registro de datos, como el conjunto de resultados de una consulta.
- ✓ Meta-contenido sobre un sitio web, como el Formato de Definición de Canal (CDF).
- ✓ Representaciones gráficas, como la interfaz de usuario de una aplicación.
- ✓ Entidades y tipos de esquema estándar.
- ✓ Todos los vínculos entre datos y personas que hay en la Web.

XML será el lenguaje que garantizará el intercambio de cualquier tipo de información, sin que ocasione problemas de tipo "contenido" o de tipo "presentación". Este garantiza que los datos estructurados sean uniformes e independientes de aplicaciones o fabricantes, lo que está originando una nueva generación de aplicaciones en la web.

---

[13] Consiste en un sistema para la organización y etiquetado de documentos.

[14] eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web.

[15] Scalable Vector Graphics (SVG) es una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados.

[16] Mathematical Markup Language es un lenguaje de marcado basado en XML, cuyo objetivo es expresar notación matemática de forma que distintas máquinas puedan entenderla.

1.8.14. Bison Framework.

Es un *framework* para la orquestación de procesos de negocio con *Windows Workflow Foundation*. Su principal objetivo es proporcionar un componente que permita gestionar las instancias de *Workflow* (WF). Además, encapsula un conjunto de actividades y servicios que le dan mayor dinamismo al desarrollo de sistemas centrado en la orquestación de procesos de negocio con WF. Entre las ventajas que brinda Bison *Framework* se pueden mencionar:

- ✓ Proporciona una mayor aproximación a los usuarios de negocio.
- ✓ Brinda rapidez y flexibilidad para modelar y cambiar los procesos según las necesidades.
- ✓ Aporta escalabilidad o capacidad de crecer.
- ✓ Fortifica el puente creado por el *Workflow* para la comunicación entre el analista y el desarrollador.
- ✓ Propone una arquitectura donde se encuentran bien definidas las capas de presentación y negocio.
- ✓ Posee actividades y servicios especializados en la orquestación de interfaces de usuario, que permiten definir su flujo de una manera gráfica dentro del *Workflow*.

1.8.15. RabbitMQ.

RabbitMQ es un software de negociación de mensajes de código abierto, el cual se encuentra dentro de la categoría de *middleware* de mensajería. Implementa el estándar *Advanced Message Queuing Protocol* (AMQP) que no es más que un protocolo de estándar abierto en la capa de aplicaciones de un sistema de comunicación. Las características que definen al protocolo AMQP son la orientación a mensajes, encolamiento, enrutamiento, exactitud y seguridad.

El servidor RabbitMQ está escrito en Erlang[17] y utiliza el *framework Open Telecom Platform* (OTP) para construir sus capacidades de ejecución distribuida y conmutación ante errores. RabbitMQ consta de diferentes partes:

- El servidor de intercambio RabbitMQ.
- Pasarelas para los protocolos HTTP, TCP/IP y STOMP[18].
- Librerías de clientes para Java y el *framework* .NET.

[17] Erlang es un lenguaje de programación concurrente y un sistema de ejecución que incluye una máquina virtual y bibliotecas.

[18] STOMP es un protocolo que soporta mensajes persistentes. Sirve para la comunicación con servidores de Base de Datos como Oracle y MySQL. La licencia es totalmente privativa.

1.9. Conclusiones.

Haciendo uso de los métodos de la investigación se realizó un estudio para sentar las bases para el desarrollo del sistema propuesto.

Se ofrece un estudio abarcador de las diferentes empresas y sistemas que lideran el mercado en cuanto a los sistemas de gestión de administración de usuario.

En este capítulo se realizó un estudio de las tecnologías y herramientas definidas por la arquitectura para el proyecto general que son:

- ✓ Metodología de desarrollo: *MSF*.
- ✓ Herramienta de modelado: *Altova UModel* 2009.
- ✓ Herramienta de desarrollo: *Visual Studio Team System* 2008.
- ✓ Sistema Gestor de Base de Datos: *Oracle 11g*.

CAPÍTULO 2: VISIÓN, PLANIFICACIÓN Y CONSTRUCCIÓN. 2

2.1. Introducción.

La metodología *Microsoft Solution Framework* aunque está netamente enfocada a la construcción de proyectos, plantea que se debe además dedicar un tiempo a capturar la visión de lo que se desea desarrollar, ya que esta actividad provee una noción bastante clara de hasta dónde se desea llegar, así como los objetivos que se proponen alcanzar con el desarrollo del producto. También plantea que se debe realizar una planificación eficiente para obtener una estimación lo más real posible del tiempo y los recursos necesarios para concluir el proyecto dentro del calendario y el presupuesto acordado.

2.2. Capturar la Visión del Proyecto.

Iniciar un proyecto requiere el establecimiento claro de la visión del proyecto. Capturar y trasladar esta visión central es el elemento más importante para mantener un proyecto enfocado. Durante el proyecto, esta visión puede cambiar como resultado de la interacción con los clientes. Si el cambio se produce, es necesario reajustar el proyecto a la nueva visión. Desde la visión, se comienza a entender cuál será la relación de los usuarios con el producto, se tendrá conocimiento si el proyecto es impulsado por fecha o por contenido. La visión y sus actividades están relacionadas con la creación de la base sólida sobre la cual puede ser un proyecto construido.

Para completar este flujo de trabajo *MSF for Agile* propone dividirlo en dos actividades:

Escribir Declaración de la Visión y Definir Personas.

2.2.1. Escribir Declaración de Visión.

El Sistema de Aprovisionamiento (*Provisioning*) se encarga de crear las cuentas de usuarios y los privilegios de dichas cuentas en las aplicaciones que hacen uso de sus servicios, además vela por que se cumplan estrictamente las políticas de seguridad en estas aplicaciones. Para crear una cuenta de usuario o pedir el acceso a una aplicación se debe primero: confirmar que la persona que está realizando dicha petición cumple una serie de requisitos necesarios para la organización. Para resolver este problema se ha creado un mecanismo basado en solicitudes, las cuales deben ser aprobadas o rechazadas por un grupo de aprobadores. El módulo Motor de Tareas se encarga de procesar de forma rápida y segura las solicitudes de usuarios disminuyendo el tiempo de espera de estos últimos.

Debido al flujo de salida de los trabajadores de una organización, ya sea de forma temporal o de forma permanente, se almacenan en los repositorios cuentas inactivas por desuso, estas cuentas se convierten en una potencial amenaza para la seguridad del sistema. El módulo Motor de Tareas se

encarga de revisar temporalmente estos repositorios en busca de cuentas en desuso para pasar a su suspensión, y en caso de que lleven el tiempo establecido suspendida se procederá a su eliminación.

2.2.2. Definir Personas.

El módulo Motor de Tareas es una aplicación que no presenta interacción directa con usuarios; sino que se vale de otros módulos para delegar estas funcionalidades. Por lo que se puede afirmar que el Motor de Tareas interactúa con sistemas externos los cuales le proveen los datos necesarios para el buen funcionamiento y configuración de sus procesos.

2.3. Planificación.

La Planificación es una fase que tiene su comienzo en una etapa temprana de la vida de un proyecto. En esta fase el equipo de trabajo analiza, identifica y prioriza los requerimientos que describen de forma total la solución y su comportamiento. Los principales artefactos que se generan en esta fase son la lista de escenarios y la lista de requerimientos de calidad de servicios.

2.3.1. Determinar la Longitud de las Iteraciones.

Una iteración es un conjunto de tareas programadas para que ocurran dentro de un período de tiempo determinado. La cantidad de tiempo necesaria para completar una iteración es conocida como longitud de la iteración. El cronograma del proyecto se divide en una serie de iteraciones y las tareas son programadas acorde a estas iteraciones. Determinar la longitud de la iteración incluye tener en cuenta factores claves incluyendo la fecha de entrega del proyecto, el tamaño de los escenarios y el tiempo de integración.

Teniendo en cuenta la fecha de entrega del módulo Motor de Tareas, se han programado para su desarrollo un total de tres iteraciones, las cuales abarcan un período de doce semanas aproximadamente.

Para la **primera iteración** se programarán los escenarios de mayor prioridad en el sistema tomando así esta iteración una longitud de seis semanas.

Para la **segunda iteración** se programarán los escenarios de prioridad media en el sistema tomando así esta iteración un total de cuatro semanas.

Para la **tercera iteración** se programarán los escenarios de prioridad baja en el sistema tomando esta iteración una longitud de dos semanas.

2.3.2. Escenarios del Sistema.

Los escenarios capturan los objetivos funcionales del sistema. Para encontrar estos objetivos, se deben analizar las necesidades únicas de cada personaje del sistema. Estos objetivos pueden ser enumerados inicialmente y luego descritos como escenarios. Para cada uno de estos objetivos, también se consideran los escenarios que pueden resultar de intentos fallidos o no óptimos para alcanzar la meta. Los escenarios se crean a través de intercambio de ideas y se añaden a la lista de escenarios. Estos son priorizados y descritos cuando se les cita para una próxima iteración. La creación de escenarios es completa cuando todos los escenarios previstos para la entrega en una iteración o un prototipo de arquitectura se escriben. Para alcanzar los objetivos del proyecto se identificaron un total de diez escenarios principales, los cuales están agrupados en tres áreas para su mejor compresión.

2.3.3. Lista de Escenarios.

Gestionar cuentas.

Desactivar cuentas por desuso.

Eliminar cuentas por desuso.

Gestionar solicitudes.

Solicitar creación de cuenta.

Solicitar modificación de cuenta.

Actualizar datos de la cuenta.

Suspender cuenta.

Bloquear cuenta.

Activar cuenta.

Cambiar contraseña.

Solicitar Eliminación de Cuenta.

Solicitar Asignación de Certificados.

Solicitar Aprovisionamiento.

Solicitar Desaprovisionamiento.

Obtener Solicitud.

2.3.4. Priorizar Escenarios del Sistema.

Los escenarios son priorizados en dependencia de la importancia que tienen estos para los usuarios, así como la necesidad de incluir dichos escenarios para la validez de la aplicación. El proceso de

priorizar la lista de escenarios ayuda a identificar los escenarios más importantes y valiosos para que sean implementados en las primeras iteraciones.

No	Escenarios	Prioridad
1	Obtener Solicitud.	4
2	Verificar Disponibilidad de Recursos.	4
3	Solicitar Creación de Cuenta.	4
4	Solicitar Modificación de Cuenta.	3
5	Solicitar Eliminación de Cuenta.	3
5	Desactivar Cuentas por Desuso.	3
6	Eliminar Cuentas por Desuso.	2
7	Solicitar Asignación de Certificados.	2
8	Solicitar Aprovisionamiento.	2
9	Solicitar Desaprovisionamiento.	2

Tabla 1. Listado de los Escenarios y sus prioridades.

2.3.5. Describir Escenarios de Mayor Prioridad.

La Descripciones de los escenarios de mayor prioridad deben contener los detalles suficientes para que los desarrolladores puedan una realizar la estimación.

Nombre	Descripción
Obtener Solicitud.	El escenario se ejecuta de manera automática. Su objetivo es buscar en un servidor de colas de mensajes la solicitud de mayor prioridad. Este escenario debe devolver la solicitud encontrada.
Solicitar Creación de Cuenta.	Este escenario se ejecuta cuando la solicitud devuelta por el escenario “Obtener Solicitud” es una solicitud de creación de cuenta, en este caso la solicitud debe pasar por un flujo de aprobación definido por los administradores.
Verificar Disponibilidad de Recursos	Este escenario se ejecuta cada vez que se quiera acceder a un recurso. Este escenario devolverá si el recurso se encuentra disponible o no.

Tabla 2. Descripción de los principales escenarios.

2.3.6. Estimar Escenarios de Mayor Prioridad.

La estimación de los escenarios se realiza para proveer un entendimiento de cuanto esfuerzo es necesario realizar para completar un escenario.

No	Escenarios	Prioridad	Riesgo	Esfuerzo(días)	(ROM)
1	Obtener Solicitud.	4	Alto	15	2
2	Verificar Disponibilidad de Recurso.	4	Alto	15	2
2	Solicitar Creación de Cuenta.	4	Alto	15	2

Tabla 3. Escenarios de mayor prioridad.

2.3.7. Cronometrar Escenarios.

Los escenarios son programados para el desarrollo y pruebas preliminares en una iteración específica. El plan de iteración refleja el entendimiento más actual de lo que debe llevarse a cabo dentro de una iteración. El plan de iteraciones debe completarse antes de que la iteración comience. Un plan inicial es creado en base a las estimaciones y refinado cuando los escenarios son divididos en tareas específicas.

Iteración 1: Se codificarán los escenarios que proveen las funcionalidades críticas del sistema:

- ✓ Obtener Solicitud.
- ✓ Verificar Disponibilidad de Recurso.
- ✓ Solicitar Creación de Cuenta.

Iteración 2: Se codificarán los escenarios de prioridad media para el sistema.

- ✓ Solicitar Modificación de Cuenta.
- ✓ Solicitar Eliminación de Cuenta.
- ✓ Desactivar Cuentas por Desuso.

Iteración 3: Se codificarán los escenarios de prioridad baja para en el sistema.

- ✓ Eliminar Cuentas por Desuso.
- ✓ Solicitar Asignación de Certificados.
- ✓ Solicitar Aprovisionamiento.
- ✓ Solicitar Desaprovisionamiento.

Una vez definidos los escenarios se gráfica la planificación realizada según las iteraciones necesarias para el desarrollo del sistema.

		Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	✓	**Primera Iteración**	**31 days**	**Mon 15/02/10**	**Mon 29/03/10**		
2	✓	Obtener Solicitud	15 days	Mon 15/02/10	Fri 05/03/10		Obtener Solicitud
3	✓	Verificar Disponibilidad de Recursos	15 days	Mon 15/02/10	Fri 05/03/10		Verificar Disponibilidad de Recursos
4	✓	Solicitar Creación de Cuentas	15 days	Tue 09/03/10	Mon 29/03/10	2;3	Solicitar Creación de Cuentas
5	✓	**Segunda Iteración**	**24 days**	**Tue 30/03/10**	**Fri 30/04/10**		
6	✓	Solicitar Modificación de Cuentas	12 days	Tue 30/03/10	Wed 14/04/10	4	Solicitar Modificación de Cuentas
7	✓	Solicitar Eliminación de Cuentas	12 days	Thu 15/04/10	Fri 30/04/10	6	Solicitar Eliminación de Cuentas
8	✓	Desactivar Cuenta por Desuso	12 days	Thu 15/04/10	Fri 30/04/10	6	Desactivar Cuenta por Desuso
9	✓	**Tercera Iteración**	**20 days**	**Mon 03/05/10**	**Fri 28/05/10**		
10	✓	Eliminar Cuentas por Desuso	10 days	Mon 03/05/10	Fri 14/05/10	7	Eliminar Cuentas por Desuso
11	✓	Solicitar Asignación de Certificado	10 days	Mon 03/05/10	Fri 14/05/10	7;8	Solicitar Asignación de Certificado
12	✓	Solicitar Aprovisionamiento	10 days	Mon 17/05/10	Fri 28/05/10	10;11	Solicitar Aprovisionamiento
13	✓	Solicitar Desaprovisionamiento	10 days	Mon 17/05/10	Fri 28/05/10	10;11	Solicitar Desaprovisionamiento

Gantt Chart

Figura 2. Plan de iteración.

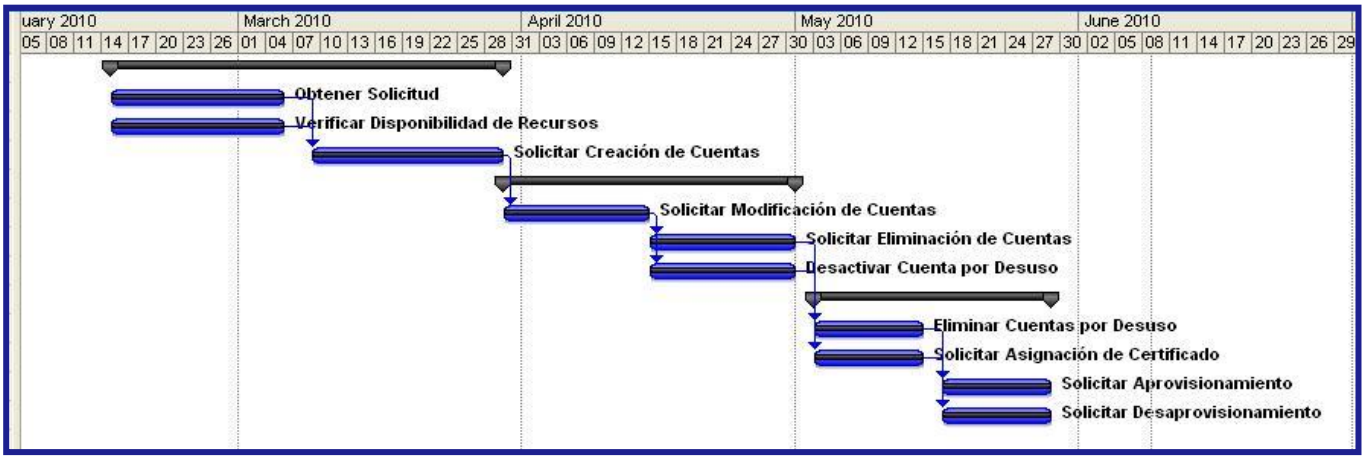


Figura 3. Diagrama de Gantt.

2.3.8. Descripción de Escenarios.

La descripción de los escenarios se realiza cuando estos son candidatos a participar en la siguiente iteración. Esta se realiza a un nivel que permita a los desarrolladores entender las necesidades de las personas que interactúan con dicho escenario y a un nivel que permita además realizar casos de pruebas.

Título	Obtener Solicitud.
Área	Gestión de Solicitudes.
Iteración	1
Estado	Activo.
Razón de estado	Proceso de Construcción.
Responsable	Carlos Pujol.

Descripción	Este escenario es el que inicia el flujo de trabajo de aprobación de solicitudes. Se ejecuta de manera automática y su objetivo es recorrer una cola de prioridad en busca de solicitudes. Al encontrar una solicitud, persiste el flujo de trabajo con el estado creado y chequea el tipo de solicitud, en dependencia del resultado ejecuta uno de los siguientes escenarios: - Solicitar Creación de Cuenta. - Solicitar Modificación de Cuenta. - Solicitar Eliminación de Cuenta. - Solicitar Asignación de Certificados. - Solicitar Aprovisionamiento. - Solicitar Desaprovisionamiento.
Historial	Inicio de descripción del escenario.
Prioridad	4
ID	101
Orden de Magnitud (ROM)	2

Tabla 4. Descripción del escenario “Obtener Solicitud”.

Título	Verificar Disponibilidad de Recursos.
Área	Gestión de Solicitudes.
Iteración	1
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Mario Azahares.
Descripción	Este escenario se ejecuta cuando se desea acceder a un recurso que fue creado anteriormente en el sistema. Este escenario devuelve si el recurso está disponible o no. En caso de que el recurso no se encuentre disponible se le debe comunicar o alertar al administrador a través de algún medio definido por este último.
Historial	Inicio de descripción del escenario.
Prioridad	4
ID	102
Orden de Magnitud (ROM)	2

Tabla 5. Descripción del escenario “Verificar Disponibilidad de Recursos”.

Título	Solicitar Creación de Cuenta.
Área	Gestión de Solicitudes.
Iteración	1
Estado	Activo.
Razón de estado	Proceso de Construcción.

Responsable	Carlos Pujol.
Descripción	Este escenario inicia cuando el escenario Obtener Solicitud adquiere una solicitud de tipo Creación de Cuenta. Una vez que el usuario ha realizado su solicitud, esta debe pasar por un flujo de aprobación definido por el administrador. Al finalizar este flujo de aprobación es que se crea la cuenta o no, notificándose a la persona que realizó la solicitud.
Historial	Inicio de descripción del escenario.
Prioridad	4
ID	103
Orden de Magnitud (ROM)	2

Tabla 6. Descripción del escenario “Solicitar Creación de Cuenta.

Título	Solicitar Modificación de Cuenta.
Área	Gestión de Solicitudes.
Iteración	2
Estado	Activo
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Este escenario comienza cuando el escenario Obtener Solicitud adquiere una solicitud de tipo Modificar Cuenta. Cuando esto ocurre, se inicia un flujo de aprobación para definir si se aceptan o no los cambios realizados por el usuario.
Historial	Inicio de descripción del escenario.
Prioridad	3
ID	104
Orden de Magnitud	1

Tabla 7. Descripción del escenario “Solicitar Modificación de Cuenta”.

Título	Solicitar Eliminación de Cuenta.
Área	Gestión de Solicitudes.
Iteración	2
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Mario Azahares.
Descripción	Este escenario comienza cuando el escenario “Obtener Solicitud” adquiere una solicitud de la cola y esta es de eliminación de cuenta. Una vez identificada el tipo de la solicitud se inicia un flujo de aprobación para eliminar dicha cuenta.
Historial	Inicio de descripción del escenario.
Prioridad	3

ID	105
Orden de Magnitud	1

Tabla 8. Descripción del escenario “Solicitar Eliminación de Cuenta”.

Título	Desactivar Cuentas por Desuso.
Área	Gestión de Cuentas.
Iteración	2
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Este escenario se ejecutará de manera automática cada cierto período de tiempo definido por el módulo de Políticas de Acceso. Consiste en inspeccionar los repositorios de cuentas de usuarios en busca de cuentas inactivas por un período de tiempo definido también por el módulo de Políticas de Acceso. En caso de encontrarse cuentas en desuso, estas pasaran a estar suspendidas notificándose de este evento a los administradores.
Historial	Inicio de descripción del escenario.
Prioridad	2
ID	201
Orden de Magnitud	1

Tabla 9. Descripción del escenario “Desactivar Cuentas por Desuso”.

Título	Solicitar Asignación de Certificados.
Área	Gestión de Solicitudes.
Iteración	3
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Este escenario comienza cuando ha llegado al escenario Obtener Solicitud una solicitud de tipo Solicitar Asignación de Certificados. Una vez ocurrido, se activa un flujo de aprobación definido por el administrador para definir si es necesario asignarle un certificado a dicha cuenta.
Historial	Inicio de descripción del escenario.
Prioridad	3
ID	203
Orden de Magnitud	1

Tabla 10. Descripción del escenario “Solicitar Asignación de Certificados”.

Título	Solicitar Aprovisionamiento.
Área	Gestión de Solicitudes.
Iteración	3
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Este escenario comienza cuando ha llegado al escenario Obtener Solicitud una solicitud de tipo Solicitar Aprovisionamiento. Cuando esto ocurre se activa un flujo de aprobación, definido por el administrador con anterioridad, en el cual se define si necesita aprovisionamiento o no. En caso de necesitar aprovisionamiento se debe verificar a qué tipo de recursos debe tener acceso y los permisos sobre ellos.
Historial	Inicio de descripción del escenario.
Prioridad	3
ID	106
Orden de Magnitud	1

Tabla 11. Descripción el escenario “Solicitar Aprovisionamiento”.

Título	Solicitar Desaprovisionamiento.
Área	Gestión de Solicitudes.
Iteración	3
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Mario Azahares.
Descripción	Este escenario es activado cuando al escenario Obtener Solicitud llega una solicitud de tipo Solicitar Desaprovisionamiento. Se activa el flujo de aprobación y se procede a eliminar los permisos que posee la cuenta sobre los recursos así como los privilegios que tiene sobre esos recursos.
Historial	Inicio de descripción del escenario.
Prioridad	3
ID	107
Orden de Magnitud	1

Tabla 12. Descripción el escenario “Solicitar Desaprovisionamiento”.

Título	Eliminar Cuenta por Desuso.
Área	Gestión de Cuentas.
Iteración	3

Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Mario Azahares.
Descripción	Este escenario se ejecutará de manera automática cada cierto período de tiempo definido por el módulo de Políticas de Acceso. Consiste en inspeccionar los repositorios de cuentas de usuarios en busca de cuentas suspendidas por un período de tiempo definido también por el módulo de Políticas de Acceso. En caso de encontrarse cuentas suspendidas, estas pasarán a ser marcadas como eliminadas notificándose de este evento a los administradores.
Historial	Inicio de descripción del escenario.
Prioridad	2
ID	202
Orden de Magnitud	1

Tabla 13. Descripción el escenario “Eliminar Cuentas por Desuso”.

2.3.9. Dividir los Escenarios en tareas.

Escenario Obtener Solicitud.

Título	Obtener una solicitud.
Tipo de Tarea	Desarrollo.
Iteración	1
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Se inicia al ejecutarse el escenario “Obtener Solicitud”. A continuación se selecciona la solicitud de mayor prioridad, en caso de que existan varias con igual prioridad se seleccionará la que mayor tiempo lleve en la cola en espera de ser atendida.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	101-101
Rango de prioridad	1

Tabla 14. Descripción de la tarea: Obtener Solicitud.

Título	Ejecutar Escenario.
Tipo de Tarea	Desarrollo.
Iteración	1
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Una vez seleccionada la solicitud se verifica de qué tipo es, y en dependencia del tipo se persiste el flujo y se activa un flujo de aprobación correspondiente al tipo de solicitud seleccionada.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	101-102
Rango de prioridad	1

Tabla 15. Descripción de la tarea: Ejecutar Escenario.

Escenario Verificar Disponibilidad de Recursos.

Título	Verificar Recurso.
Tipo de Tarea	Desarrollo.
Iteración	1
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Mario Azahares.
Descripción	Cada vez que se desee acceder a un determinado recurso, lo primero que se debe hacer es verificar que el mismo se encuentre disponible, y que el usuario que lo solicita tiene los permisos correspondientes para hacer uso del mismo.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	102-201
Rango de prioridad	1

Tabla 16. Descripción de la tarea: Verificar Recurso.

Escenario Solicitar Creación de cuenta.

Título	Crear Cuenta.
Tipo de Tarea	Desarrollo.
Iteración	1
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Cuando es atendida una solicitud de tipo Creación de Cuenta se inicia un flujo de aprobación. Cuando culmina el proceso se le notifica al usuario que su cuenta ha sido creada.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	103-301
Rango de prioridad	1

Tabla 17. Descripción de la tarea: Crear Cuenta.

Escenario Solicitar Modificación de cuenta.

Título	Modificar Cuenta.
Tipo de Tarea	Desarrollo.
Iteración	2
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Cuando la solicitud es de tipo “Solicitar Modificación de Cuenta”, se activa el flujo de aprobación para luego almacenar los nuevos datos si es aprobada la solicitud, en caso contrario notificar al usuario que no se autorizan los cambios solicitados.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	104-401
Rango de prioridad	1

Tabla 18. Descripción de la tarea: Modificar Cuenta.

Escenario Solicitar Eliminación de cuenta.

Título	Eliminar Cuenta.
Tipo de Tarea	Desarrollo.
Iteración	2
Estado	Activo
Razón de estado	Proceso de construcción.
Responsable	Mario Azahares.
Descripción	Cuando la solicitud es de tipo Solicitar Eliminación de Cuenta se activa el flujo de aprobación correspondiente y se elimina la cuenta si es aprobada la solicitud, de lo contrario la cuenta permanecerá activa.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	105-501
Rango de prioridad	1

Tabla 19. Descripción de la tarea: Eliminar Cuenta.

Escenario Desactivar Cuentas por Desuso.

Título	Buscar Cuentas Inactivas.
Tipo de Tarea	Desarrollo.
Iteración	2
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Cuando el sistema ejecute de manera automática el escenario “Desactivar Cuentas por Desuso” el sistema buscara en los repositorios de cuentas de usuarios todas las cuentas cuyo tiempo de inactividad sea mayor o igual al definido por el administrador.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	201-201
Rango de prioridad	1

Tabla 20. Descripción de la tarea: Buscar Cuentas Inactivas.

Título	Desactivar Cuentas Inactivas.
Tipo de Tarea	Desarrollo.
Iteración	2
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Todas las cuentas que fueron encontradas en los repositorios que se encontraban inactivas el sistema las pone en estado “Suspendida” y se le notifica al administrador del evento ocurrido.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	201-202
Rango de prioridad	1

Tabla 21. Descripción de la tarea: Desactivar Cuentas Inactivas.

Escenario Solicitar Asignación de Certificados.

Título	Activar flujo de aprobación para Asignación de Certificado.
Tipo de Tarea	Desarrollo.
Iteración	3
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Cuando llega la solicitud se activa un flujo de aprobación para determinar si es necesario asignarle un certificado digital. En caso afirmativo se le asigna dicho certificado a la cuenta de usuario.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	203-201
Rango de prioridad	1

Tabla 22. Descripción de la tarea: Activar Flujo de Aprobación para Asignación de Certificados.

Escenario Solicitar Aprovisionamiento.

Título	Activar Flujo de Aprobación para Aprovisionamiento.
Tipo de Tarea	Desarrollo.
Iteración	3
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Carlos Pujol.
Descripción	Cuando llega la solicitud se activa el flujo de aprobación para determinar si se aprovisiona o no la cuenta. Se le dan los permisos correspondientes en los recursos a los que tendrá acceso.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	106-101
Rango de prioridad	1

Tabla 23. Descripción de la tarea: Activar Flujo de Aprobación para Aprovisionamiento.

Escenario Solicitar Desaprovisionamiento.

Título	Activar flujo para aprobar el desaprovisionamiento.
Tipo de Tarea	Desarrollo.
Iteración	3
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Mario Azahares.
Descripción	Cuando la solicitud es recibida se activa el flujo de aprobación y en caso de que se apruebe el desaprovisionamiento se le quitan los permisos a la cuenta sobre los recursos a los cuales tenía acceso, así como los privilegios sobre cada recurso.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	107-101
Rango de prioridad	1

Tabla 24. Descripción de la tarea: Activar flujo para aprobar el desaprovisionamiento.

Escenario Eliminar Cuentas por Desuso.

Título	Buscar Cuentas Suspendidas.
Tipo de Tarea	Desarrollo.
Iteración	3
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Mario Azahares.
Descripción	Cuando el sistema ejecute el escenario “Eliminar Cuentas por Desuso” se hará una búsqueda en los repositorios de las cuentas de usuarios buscando las cuentas que estén marcadas como “Suspendidas” y que lleven un tiempo igual o mayor al definido por el administrador del sistema.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	202-201
Rango de prioridad	1

Tabla 25. Descripción de la tarea: Buscar Cuentas Suspendidas.

Título	Eliminar Cuentas Suspendidas.
Tipo de Tarea	Desarrollo.
Iteración	3
Estado	Activo.
Razón de estado	Proceso de construcción.
Responsable	Mario Azahares.
Descripción	Con los identificadores de todas las cuentas que se encuentran suspendidas el sistema marcará de manera automática todas estas cuentas como eliminadas, y se le notificará al administrador del evento ocurrido.
Historial	Inicio de descripción del escenario.
Tiempo de desarrollo (horas)	30
ID	202-202
Rango de prioridad	1

Tabla 26. Descripción de la tarea: Eliminar Cuentas Suspendidas.

2.3.10. Requerimientos de Calidad de Servicio.

Los requerimientos de calidad de servicios capturan los requerimientos no funcionales del sistema o las limitaciones en la funcionalidad del sistema. Estos deben ser precisos y no subjetivos en su definición. Los requisitos son identificados, priorizados, y si están programados para la iteración actual, deben ser descritos.

Fiabilidad

- ✓ El sistema estará disponible las 24 horas durante los 7 días de la semana.
- ✓ No se realizarán mantenimientos preventivos en horario laboral, deberán ejecutarse en un horario estipulado o los fines de semana, para no afectar la disponibilidad del sistema.
- ✓ Las fallas del software se dividirán en dos categorías:
  - **Simples:** la solución y la actualización se realizarán en línea en un período inferior a 4 horas.
  - **Complejas:** la solución y actualización se realizarán en un tiempo que se definirá posterior a una evaluación detallada.
- ✓ El sistema llevará un sistema de *tracking* de errores.
- ✓ Solo se accederá a la BD desde la aplicación, nunca directamente desde el gestor de Base de Datos.
- ✓ Se garantizará la consistencia de los datos, se realizarán comprobaciones y validaciones automáticas en todos los casos posibles.
- ✓ La información manejada por el sistema será eliminada una vez terminada de procesar.

Eficiencia

- ✓ El sistema debe ser capaz de gestionar una cantidad escalable de procesos concurrentes.
- ✓ El sistema debe codificarse siguiendo los estilos de código definidos por el proyecto.
  - Para la implementación se utilizó el estilo *Camel*, que plantea una serie de parámetros a seguir como:
    - Todas las clases deben encontrarse en archivos separados.
    - Tratar de evitar los comentarios de bloque, y si es necesario tratar de hacerlo usando el siguiente estilo.
      /*línea1
      línea2
      línea3*/
    - Realizar una declaración por línea, no poner variables de distintos tipos de datos en una misma línea.

- En cuanto a las convenciones de nomenclatura se utiliza *Cubierta Pascal* que pone en mayúscula la primera letra de cada palabra, por ejemplo ***W****orkflow****S****ervices*.

Restricciones de diseño

- ✓ El sistema debe implementarse usando el lenguaje *C sharp*, sobre la plataforma ASP.NET.
- ✓ El sistema gestor de bases de datos, será Oracle 11g.
- ✓ El sistema debe desarrollarse usando el IDE *Visual Studio Team System 2008*.
- ✓ Se utilizará el *Team Explorer*[19] como control de código fuente.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

- ✓ Se entregarán documentos técnicos.
- ✓ Se entregará el Manual de usuario.
- ✓ Se entregará Manual de administración y configuración del sistema.

Requisitos de Licencia

Para el desarrollo del sistema se necesitan un conjunto de aplicaciones, plataformas, sistemas operativos, gestores de bases de datos, herramientas, que son sistemas propietarios y necesitan de licencias para su buen desempeño y soporte, las cuales son:

- ✓ *Visual Studio Team System 2008*.
- ✓ *PL/SQL Developer.*
- ✓ *Oracle Database Enterprise Edition.*
- ✓ *Windows XP Professional SP 2.*
- ✓ *Windows Server 2003 Enterprise Edition.*

2.4. Construcción.

La Fase de Construcción plantea la codificación de cada uno de los componentes diseñados, así como las pruebas unitarias y de integración. Supone el desarrollo iterativo e incremental de la solución, por lo cual, debe existir una adecuada planificación de esta actividad.

2.4.1. Crear Arquitectura de la Solución.

Una buena arquitectura tiene una estructura interna limpia y simple para los principales elementos de la aplicación, lo cual reduce la complejidad de la misma. La arquitectura debe definir elementos estructurales que permita a la aplicación manipular mejor los cambios en los requerimientos y permitir

[19] Microsoft Visual Studio Team System 2008 Team Explorer es un entorno simplificado de Visual Studio Team System 2008 que se usa única y exclusivamente para obtener acceso a Team Foundation Server services.

a los módulos que lo requiera un desarrollo independiente. Una buena arquitectura toma además ventajas de las estructuras de capa para disminuir la dependencia entre los componentes y aumentar la reusabilidad.

La metodología plantea como primera actividad a realizar en la creación de la arquitectura de la solución la partición del sistema en componentes que estén relacionados lógicamente. Particionar el sistema produces beneficios, ya que reduce la complejidad, crea encapsulación, incrementa potencialmente la reutilización de los componentes y crea unidades lógicas para el despliegue. Esta actividad contiene un subconjunto de actividades entre la cuales se encuentra: seleccionar los patrones de arquitecturas adecuados.

Después de realizar un análisis del entorno donde se desenvolverá el módulo motor de tareas, así como de los componentes que lo integrarán y las funcionalidades que estos realizarán se decidió desarrollar una arquitectura dividida en capas, con el propósito de alcanzar un alto nivel de abstracción.

2.4.1.1. Vista Lógica.

Con el propósito de alcanzar un alto nivel de abstracción el módulo Motor de Tareas se encuentra divido en cuatro capas la cuales brindan un alto nivel de encapsulamiento de las responsabilidades, permitiendo reducir al máximo el acoplamiento y aumentar la reutilización entre las mismas. Con esta arquitectura las capas inferiores proveen servicios a las capas superiores y las capas superiores delegan responsabilidades a las capas inferiores.

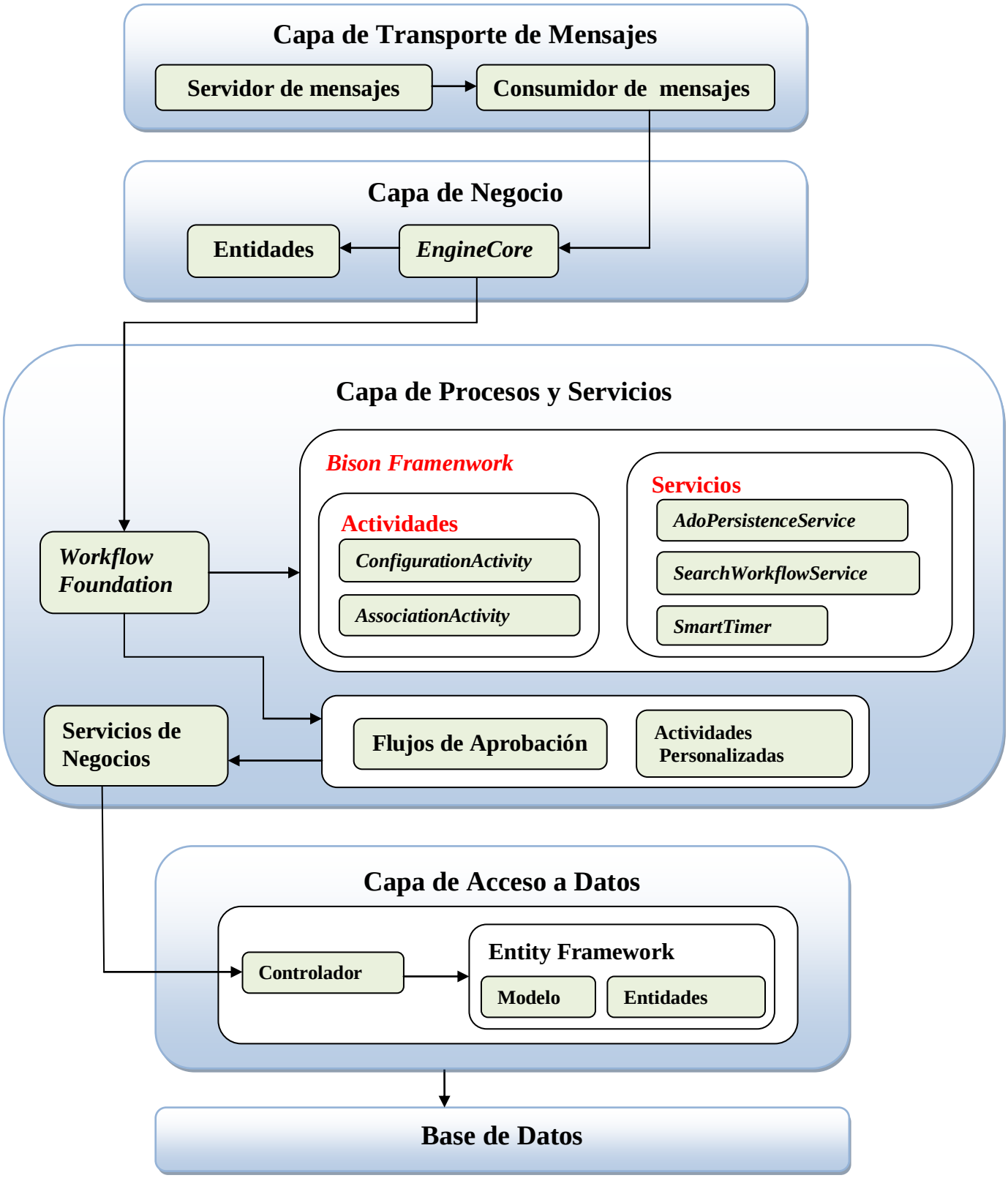


Figura 4. Vista lógica del módulo Motor de Tareas

2.4.1.2. Descripción de las Capas del Motor de Tareas.

Capa de Transporte de Mensajes: Provee la infraestructura necesaria para el intercambio de mensajes entre el servidor de mensajes y el motor de tareas.

Capa de Negocio: Está conformada por un conjunto de servicios de negocio que permite la ejecución de manera dinámica de los flujos de trabajos que se encuentran en la capa de procesos y servicios.

Capa de Procesos y Servicios: Esta capa permite el modelado del negocio mediante procesos, haciendo uso de *Windows Workflow Foundation*. Define además los servicios que darán cumplimiento a las actividades modeladas en el proceso.

Capa de Acceso a Datos: La capa de acceso a datos está directamente relacionada con las funcionalidades definidas en el negocio. Para establecer esta relación hace uso de las clases interfaces y controladoras que define la capa de negocio. De esta manera, es posible realizar cambios en esta capa sin que se vean afectadas las demás capas. Su principal función es realizar una implementación de las interfaces definidas en la capa de negocio y al mismo tiempo trabajar directamente con la fuente de datos establecida.

En esta capa se encuentra incluido el *Entity Framework*, herramienta utilizada para la generación del acceso a datos. *Entity Framework* es un nuevo *framework* de modelado que permite a los desarrolladores definir un modelo conceptual a partir de un esquema de base de datos que está alineado a una vista del mundo real de la información. Uno de sus beneficios es la facilidad de entendimiento y de mantenimiento del código de la aplicación, ya que, está preparado para los cambios en el esquema del modelo de datos que la soporta.

Capa de Base de Datos: Está constituida por todo el conjunto de tablas y procedimientos que permiten el almacenamiento de la información recolectada y procesada por los procesos.

2.4.1.3. Capa de Procesos y Servicios.

Esta capa es el núcleo del Motor de Tareas ya que es la encargada de dirigir los procesos de negocios mediante flujos de trabajo. Entre sus principales componentes se encuentran: la tecnología de *Microsoft Windows Workflow Foundation (WF)*, la cual permite la creación, ejecución y administración de flujos de trabajos, El *framework Bison* el cual extiende las capacidades de *Windows Workflow Foundation* en cuanto a la persistencia y el uso de actividades y servicios propios de este *framework*. Los flujos de trabajos y servicios que representan la lógica del negocio. Actividades personalizadas creadas para cubrir necesidades especificas del negocio.

2.4.1.3.1. Componentes de la Capa de Procesos y Servicios.

Workflow Foundation: Provee los componentes necesarios para la ejecución de instancias de *flujos de trabajo* mediante el uso de una arquitectura que ha sido diseñada para proveer una amplia extensibilidad y configuración de las funciones. Entre sus componentes se encuentran:

> ***WF Runtime Engine:*** Provee las bases fundamentales del conjunto de actividades requeridas para ejecutar y administrar el ciclo de vida de los flujos de trabajos.
>
> ***WF Runtime Services:*** Son instancias de clases que se crean y registran en el *WF Runtime* durante el inicio de la aplicación. Cada servicio presenta una función especial entre las cuales se encuentran: la persistencia de las instancias de flujos de trabajos, llevar las trazas correspondientes a estas instancias, así como la comunicación entre los flujos de trabajos y las aplicaciones externas a estos.

Bison Framework: *Framework* para la orquestación de procesos de negocio con *Workflow Foundation*. Cuenta con varios paquetes como son: *Activities, Runtime Services y Hosting*

De este *framework* el Motor de Tareas solo utiliza los siguientes servicios: *AdoPersisnteceService* el cual permite la persistencia de las instancias *de flujos de trabajo* en bases de datos *Oracle. SearchWorkflowService* permite la búsqueda de instancia de *workflow* mediante diferentes criterios de entrada. *SmartTimer* permite la ejecución de procesos cada un período de tiempo determinado. Además, se utilizaron las actividades *ConfigurationActivity y AssociationActivity* dicho *framework.*

Actividades Personalizadas: Conjunto de actividades que por su nivel de reutilización son definidas como actividades independientes para ser utilizadas por varios flujos de trabajo, o por uno más de una vez.

Flujos de Trabajo*:* Son todos aquellos procesos definidos dentro de la aplicación donde se define la lógica de negocio de la aplicación, estos a su vez interactúan con los servicios de negocio que son los que contienen la lógica de las funcionalidades.

Servicios de Negocio: Exponen servicios a los cuales los flujos de trabajo necesitan acceder para ejecutar cierta funcionalidad del negocio.

2.4.1.4. Patrones seleccionados.

2.4.1.4.1. Patrones *Grasp (General Responsibility Assignment Software Patterns).*

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. El nombre se eligió para indicar la importancia de captar (*grasping*) estos principios, si se quiere diseñar eficazmente el software orientado a objetos. A continuación, describiremos los patrones para asignar responsabilidades:

- **Patrón experto.**

Un modelo de clase puede definir docenas y hasta cientos de clases de *software*, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades. Si estas se asignan en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se nos presenta la oportunidad de reutilizar los componentes en futuras aplicaciones. A través del uso de este patrón se conserva el encapsulamiento, ya que, los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase "sencillas" y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

- **Patrón creador.**

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. El patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

- **Patrón bajo acoplamiento.**

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa que una clase no depende de muchas clases. Acoplamiento alto significa que una clase recurre a muchas otras clases. El grado de acoplamiento no puede considerarse aisladamente de otros principios como experto y alta cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño.

- **Patrón alta cohesión.**

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una baja cohesión hace muchas cosas no afines o realiza trabajo excesivo.

- **Patrón controlador.**

Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. Asignar la responsabilidad del manejo de un

mensaje de los eventos de un sistema a una clase. Un defecto frecuente al diseñar controladores consiste en asignarles demasiada responsabilidad. Normalmente, un controlador debería delegar a otros objetos el trabajo que ha de realizarse mientras coordina la actividad.

2.4.1.4.2. Patrones *Gof* (*Gang of four*).

- **Patrón *Singleton* (instancia única).**

Este patrón está diseñado para restringir la creación de objetos pertenecientes a una clase. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. El patrón *singleton* se implementa creando en nuestra clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado).

- **Patrón fachada (*facade*).**

Provee una interfaz unificada para un conjunto de interfaces de un subsistema. *Facade* define una interfaz de alto nivel que hace al sistema fácil de usar dividiendo el sistema en subsistemas, ayudando a reducir la complejidad del mismo, un diseño común disminuye las dependencias y comunicaciones entre subsistemas. Protege al cliente de los componentes del subsistema, así reduce el número de objetos con los que el cliente se relaciona. Promueve un débil acoplamiento entre el subsistema y los clientes. Esto permite modificar componentes en el subsistema sin modificar el cliente. No impide que las aplicaciones usen las clases del subsistema si es necesario.

2.4.1.4.3. Patrones de flujos de trabajo.

Los patrones para el diseño de *flujos de trabajo* van desde los más simples como el patrón secuencial hasta los más complejos, por ejemplo, el patrón de sincronización.

Varios de los patrones utilizados para el diseño de los *flujos de trabajo* se muestran a continuación.

- **Patrones de control de flujos básicos:** Estos patrones están presentes en la mayoría de los lenguajes de *Workflow*, y sirven para modelar procesos secuenciales, paralelos, o aquellos que incluyan alguna decisión.
- **Patrones de ramificación avanzada y sincronización:** Estos patrones superan a los patrones de control de flujo básico al permitir tipos avanzados de bifurcación y sincronización.
- **Patrones estructurales:** Estos patrones permiten terminar un subproceso cuando ya no haya nada que hacer, o permiten definir ciclos de forma arbitraria.
- **Patrones que manejan múltiples instancias:** Cuando se le da seguimiento a un caso, algunas veces es necesario que el proceso sea instanciado muchas veces.

2.4.1.5. Diagrama lógico de centro de datos.

El diagrama lógico de centro de datos define o documenta las configuraciones específicas de aplicaciones de tipo servidor, las cuales pueden ser IIS, SQL Server o BizTalk Server. Tiene el propósito específico de asegurar la conexión de entrada y salida del servidor web. (7)El diagrama muestra como estas aplicaciones configuradas lógicamente se interconectan. El diagrama de aplicaciones muestra los elementos de despliegue, como los servicios Web, aplicaciones Web, aplicaciones Windows, bases de datos externas y servicios web externos mostrando las conexiones entre estas aplicaciones, reflejando la configuración actual de la solución.

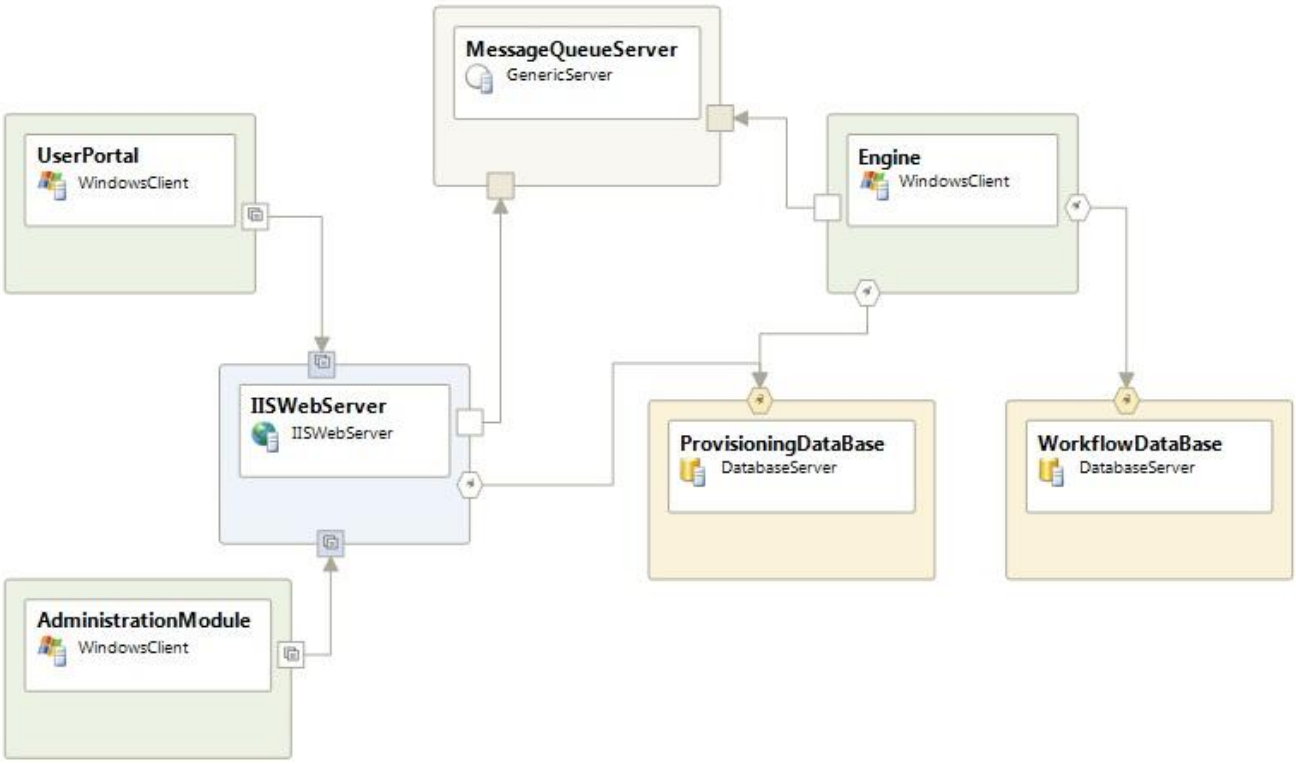


Figura 5. Diagrama lógico de centro de datos.

2.4.1.6. Diagrama de aplicación.

Otro de los aspectos fundamentales a la hora de definir la arquitectura del sistema, según la metodología utilizada, es realizar el diagrama de aplicación, el mismo se crea con el objetivo de dar una perspectiva de todos los componentes que se relacionen con la aplicación. Es el ámbito de la solución y muestra los elementos desplegables tales como servicios y aplicaciones web. Además, hace una referencia a las aplicaciones de bases de datos y servicios externos. El diagrama muestra la conexión entre las aplicaciones reflejando la actual configuración de la solución.

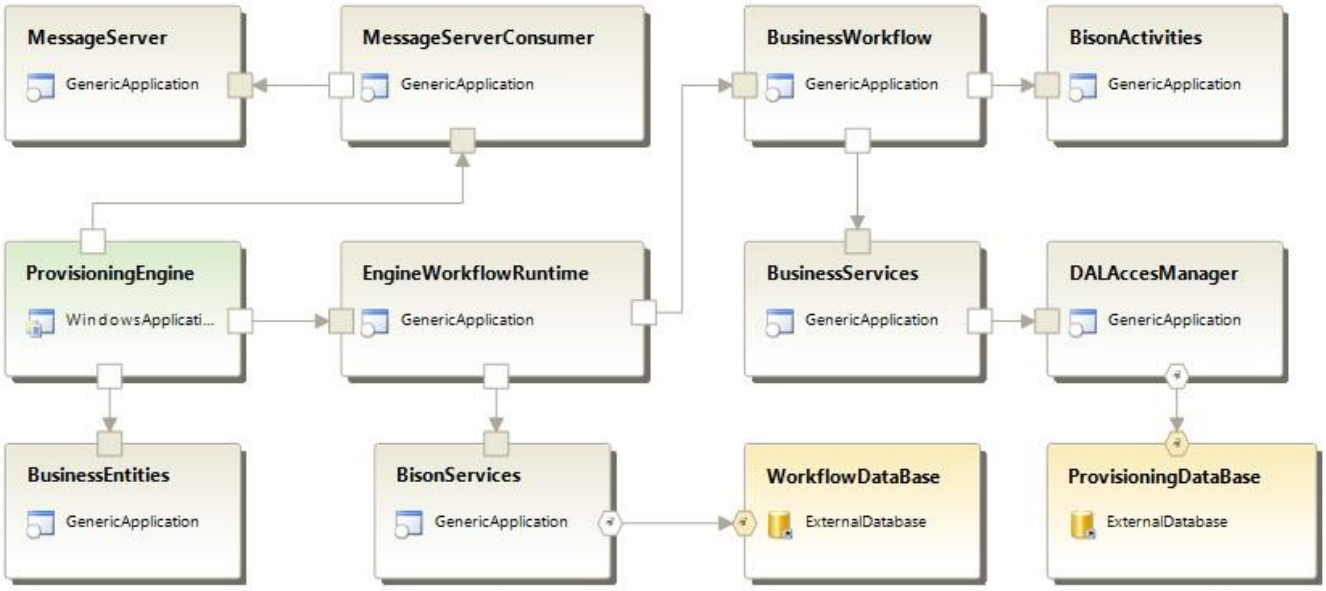


Figura 6. Diagrama de aplicación del sistema.

2.4.1.7. Flujo de funcionamiento del sistema.

El módulo Motor de Tareas está diseñado para que funcione como un servicio del sistema operativo *Windows* y se mantenga en ejecución a la espera de peticiones realizadas por los módulos Portal de Usuario y Administración, ambos pertenecientes al Sistema de Aprovisionamiento. Estas peticiones son enviadas hacia un servidor de colas de mensajes en espera de ser atendidas por el Motor de Tareas. Para realizar esta función está diseñada la capa de transporte de mensajes la cual está compuesta por el servidor de cola de mensajes y un componente cuya responsabilidad es entregar los mensajes recibidos a la capa de negocios del motor.

Una vez que los mensajes se encuentren en la capa de negocios estos son interpretados como solicitudes las cuales pueden ser de tipo creación, modificación o eliminación de cuenta. Para cada tipo de solicitud existe un flujo de aprobación, en dependencia del tipo de usuario que la envía y el tipo de cuenta que la solicitud maneja, estos flujos de aprobación son asignados a la solicitudes en tiempo de ejecución mediante información contenida en archivos *XML* los cuales son configurados por los administradores del sistema. Una vez seleccionado el flujo de aprobación correspondiente se delega la funcionalidad de crear una instancia de dicho flujo a la capa de procesos y servicios, estas instancias de flujo de trabajo tienen la responsabilidad de ejecutar la acción que se le solicita (creación, modificación, eliminación) sobre la cuenta en cuestión delegando esta función a la capa de acceso a datos del Motor de Tareas la cual accede a la base de datos para realizar el procedimiento requerido. Paralelo a todo

este proceso el Motor de Tareas ejecuta periódicamente una serie de tareas programadas (el periodo de tiempo con que se desee ejecutar las tareas es configurado en un fichero *XML*). Estas tareas se encargan de verificar los repositorios de usuario para suspender las cuentas que se encuentren inactivas hace un periodo de tiempo determinado, configurable este valor mediante un fichero *XML* y eliminar las cuentas que se encuentra suspendidas hace un determinado periodo de tiempo (igualmente configurable). Para esto el motor crea una serie de temporizadores en dependencia a la cantidad de tareas que tenga que realizar (un temporizador por cada tarea), la cantidad de tareas a realizar es configurable mediante ficheros XML. El motor inicia el conteo de los temporizadores, estos alertan al motor cuando el conteo termina y este inicia el flujo de trabajo correspondiente a cada tarea los cuales se definen en los ficheros XML.

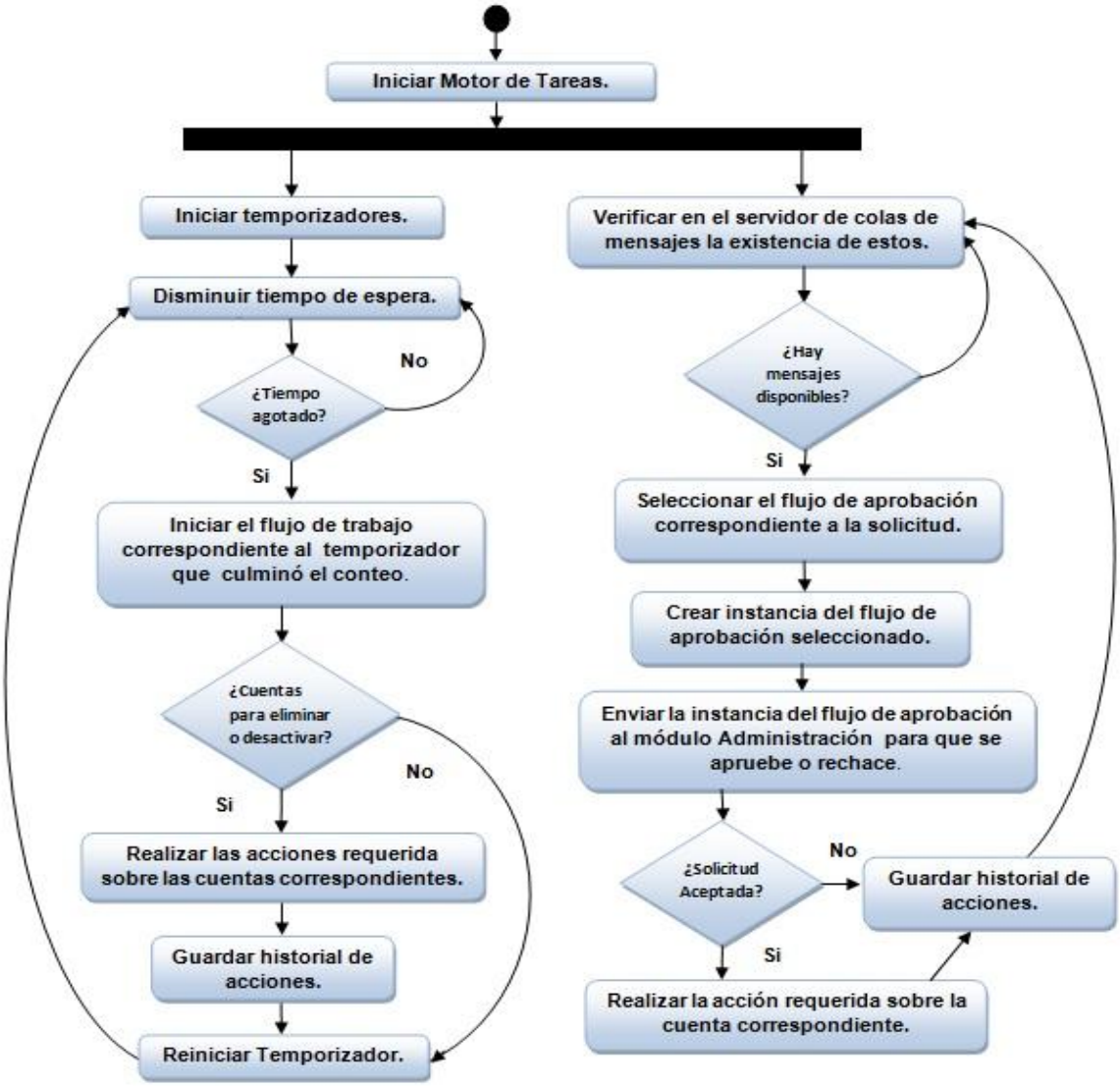
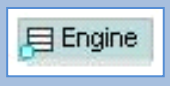


Figura 7. Diagrama de flujo de funcionamiento.

2.4.1.5. Diagrama de clases.

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro. *Ver* ***Anexo I***

2.4.1.5.1. Descripción de las clases.

Nombre:	*Engine*
Tipo de Clase:	Controladora
Descripción:	Esta clase es la encargada de implementar los métodos que permitirán administrar las cuentas y solicitudes de manera automática. 
Atributos	
Nombre	**Descripción**
accountManager:IAccountManager	Es un objeto de la clase interfaz *IAccountManager* que cuenta con los métodos para crear, modificar y eliminar cuentas de usuarios.
requestManager:IRequestManager	Es un objeto de la clase interfaz *IRequestManager* que cuenta con los métodos que permiten crear, rechazar y aprobar las solicitudes.
resourceManager:IResourceManager	Es un objeto de la clase interfaz *IResourceManager* la cual cuenta con los métodos crear, eliminar recursos, verificar disponibilidad de recursos y actualizar datos de los recursos.
Métodos	
Nombre	**Descripción**
<<GetAccessor, SetAccessor, property>> AccountManager():AccountManager	Permite obtener y modificar los valores de la clase *AccountManager* que es la que implementa los métodos de la clase *IAccountManager*.

<<GetAccessor, SetAccessor, property>> *RequestManager():RequestManager*	Permite obtener y modificar los valores de la clase *RequestManager* que es la que implementa los métodos de la clase *IRequestManager*.
Asociaciones	*Provisioning*

Tabla 27. Descripción de la clase Engine del diagrama de clases del sistema.

Ver el resto de las descripciones en el **Anexo II.**

2.4.1.5.2. Diagrama de clases persistentes.

Las clases persistentes son aquellas que almacenan alguna información más allá de la ejecución de la aplicación. En el siguiente diagrama se muestran las clases persistentes.

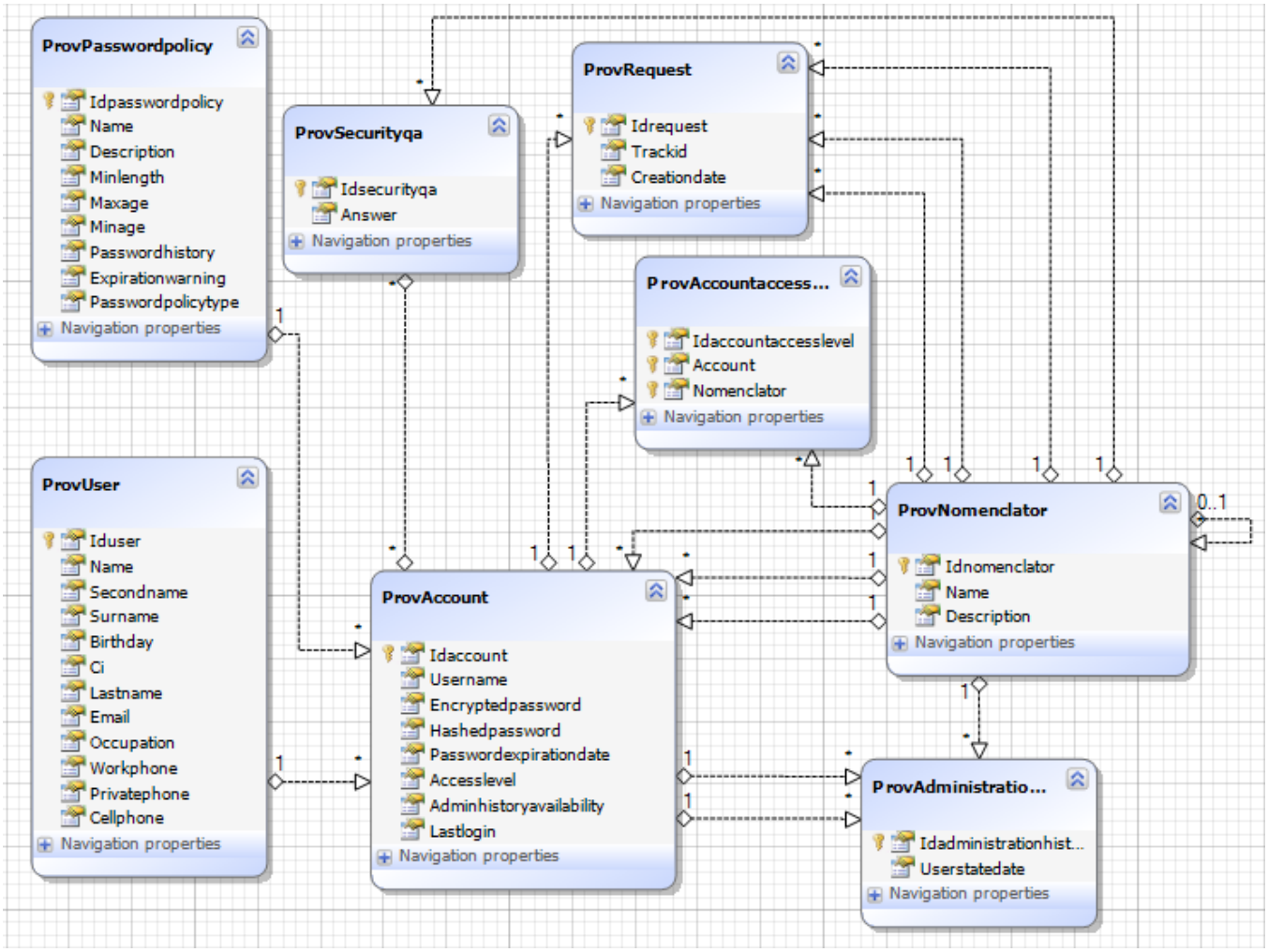


Figura 8. Diagrama de clases persistentes.

2.4.1.5.3. Descripción de las Clases Persistentes.

Nombre de la Clase:	*ProvPasswordpolicy*
Tipo de Clase:	Entidad
Atributo	**Tipo**
Idpasswordpolicy	*Guid*
name	*String*
Description	*String*
Minlength	*int*
Maxage	*int*
Minage	*int*
Passwordhistory	*int*
Expirationwarning	*int*
passwordpolicytype	*bool*
Responsabilidad	
Nombre	**Descripción**
ProvPasswordpolicy()	Constructor

Tabla 28. Descripción de la clase Persistente *ProvPasswordpolicy*.

Ver el resto de las descripciones en el **Anexo III**.

2.4.2. Descripción de los flujos de trabajo desarrollados.

El proceso principal que ejecuta el Motor de Tareas es crear, modificar y eliminar cuentas de usuarios, una vez que las solicitudes para realizar dichas acciones hayan pasado a través de un mecanismo que valide que estas peticiones son correctas. Este mecanismo es desarrollado mediante flujos de trabajo de máquina de estado. Un flujo de trabajo de máquina de estado está compuesto por tres elementos fundamentales, los cuales son: estados, eventos y transiciones. La relación que existe entre estos componentes es la siguiente: Un evento permite la transición de un estado a otro. Un estado consta de tres fases las cuales contienen un conjunto de actividades responsables de llevar a cabo la lógica del proceso, estas fases son: fase inicial, fase de espera y fase final.

La fase inicial contiene las actividades que se ejecutarán cuando se inicia un estado. Un estado se inicia cuando: ocurre una transición de otro estado hacia este o si este tiene la característica de ser el estado inicial del flujo de trabajo. La fase de espera se encarga como su nombre lo indica de esperar la ejecución de uno o varios eventos los cuales permiten la transición hacia otro estado, contiene además

las actividades que se ejecutarán una vez que el evento haya sido capturado. La fase final es última fase que se ejecuta en el estado actual antes de realizar la transición hacia otro estado.

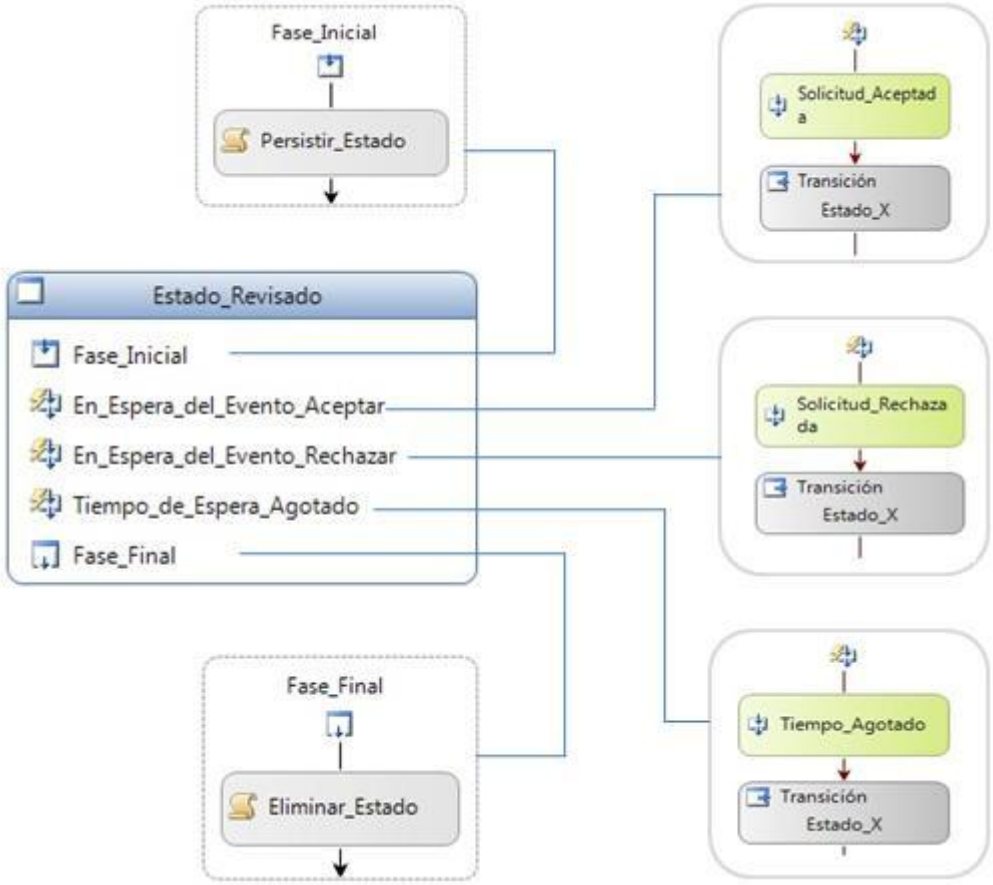


Figura 9. Vista de las fases del estado Revisado y sus componentes.

En la figura anterior se muestra en detalles una abstracción del estado Revisado, en su fase inicial se persiste el estado en que se encuentra la instancia de flujo de trabajo en una base de datos creada especialmente con este objetivo mediante la actividad “Persistir_Estado”. Como se observa en la figura la fase de espera se encuentra dividida, cada división representa los eventos por los que espera el flujo de trabajo, estos son: “Solicitud_ Aceptada”, “Solicitud_Rechazada” y “Tiempo_Agotado”, solo se ejecuta el primer evento que se reciba el cual permitirá la realización de las actividades correspondientes a este evento y la transición hacia el estado definido por la lógica del proceso.

El Motor de Tareas define cuatro flujos de trabajo los cuales se diferencian por la cantidad de niveles de aprobación que contienen .Para mejor comprensión se muestra una tabla con los estados por lo que puede pasar una solicitud que necesite del flujo de aprobación principal el cual cuenta con cuatro niveles de aprobación.

Estados	creado	revisado	aprobado	chequeado	rechazado	completado	finalizado
Creado	-	X	-	-	-	-	-
Revisado	-	-	X	-	X	-	-
Aprobado	-	X	-	X	X	-	-
Chequeado	-	-	X	-	X	X	-
Rechazado	-	X	X	X	-	-	X
Completado	-	-	-	-	-	-	X
Finalizado	-	-	-	-	-	-	-

Tabla 29. Posibles transiciones entre los estados por los que puede pasar una solicitud.

Por ejemplo, una solicitud en estado “Creado” puede pasar solamente al estado “revisado”. Por otra parte una solicitud en estado “Revisado” puede pasar al estado “aprobado” en caso de que se apruebe la solicitud, o puede pasar al estado rechazado en caso de que no se apruebe la solicitud.

Para el caso de una solicitud que se encuentre en el estado “Aprobado” puede ser enviada nuevamente al estado “revisado” para que se verifiquen los datos que contiene, o se puede aprobar para que pase al estado “chequeado” o simplemente es rechazada la solicitud.

Cuando la solicitud se encuentra en estado “Chequeado” solamente tiene posibilidad de pasar a tres estados, el primero es al estado “aprobado” para que se verifiquen nuevamente los datos introducidos. Puede ser rechazada pasando de esta manera al estado “rechazado” o puede ser chequeada satisfactoriamente pasando así al siguiente estado “completado”. Para una solicitud que se encuentre en estado “Completado” solamente existe un camino a seguir, y es pasar al estado “finalizado”.

Para todos los casos en que una solicitud fuese enviada al estado “rechazada” desde cualquier otro estado, esta tiene la posibilidad de pasar a ser revisada nuevamente o sencillamente pasa al estado finalizado.

A continuación se muestra en la **figura. 10** el flujo de trabajo diseñado para el flujo principal de aprobaciones.

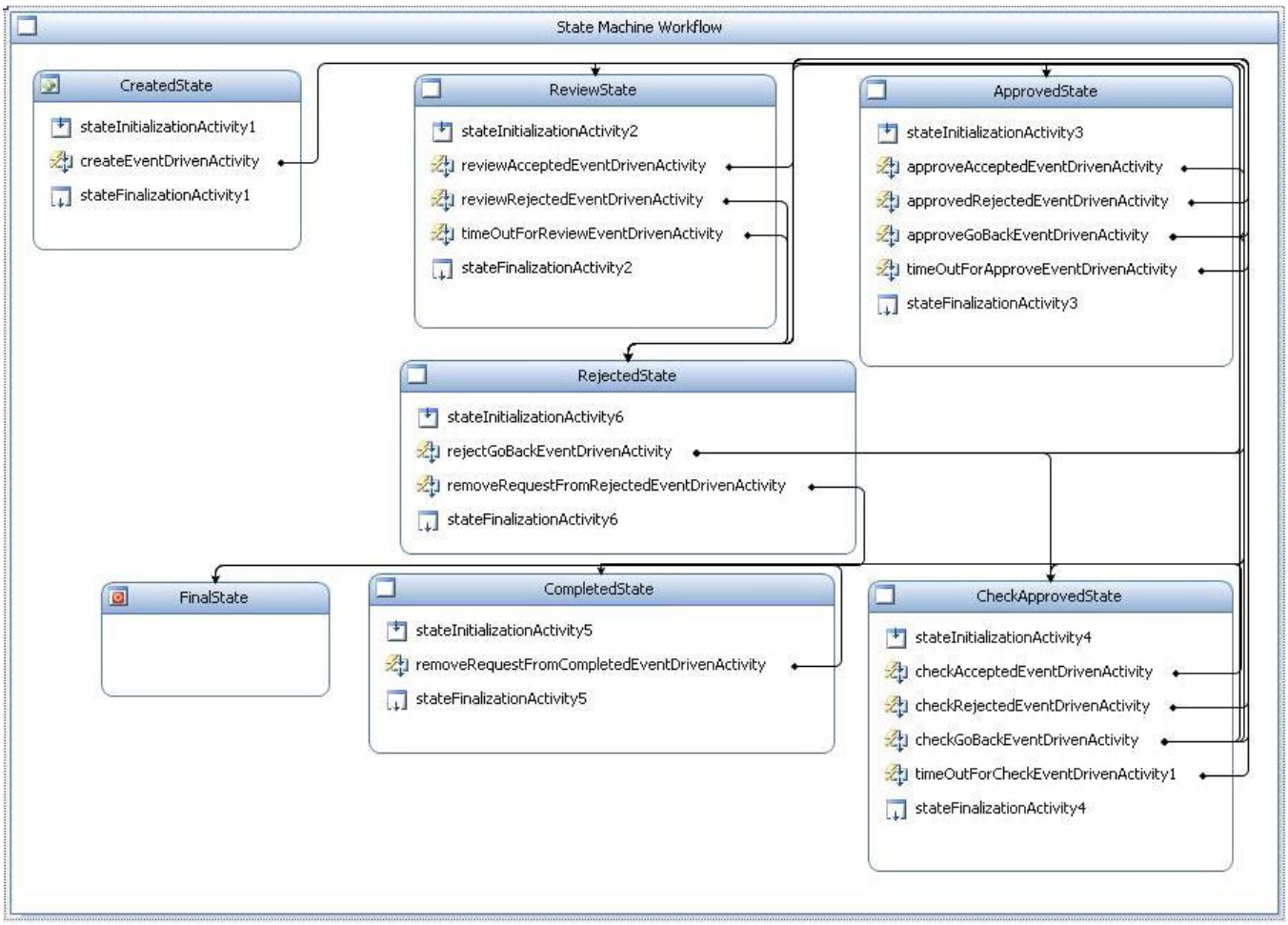


Figura 10. Diseño del Workflow desarrollado para el flujo de aprobación principal.

Para todas las solicitudes que se realizan existe un flujo de aprobación diferente en dependencia del tipo de cuenta y de cómo se haya configurado por parte del usuario que hará uso de la aplicación. Ejemplo de ello es:

Para una solicitud de creación de cuenta de cualquier usuario sobre cualquier rol necesita del flujo de aprobación.

Para una solicitud de eliminación de cuenta realizada por un administrador sobre cualquier tipo de cuenta no se necesita de ningún flujo de aprobación, por lo que el flujo correspondiente es de nivel cero.

En caso de que la solicitud no sea hecha por un administrador si deberá pasar por el flujo de aprobación principal.

Para las solicitudes de modificación de cuentas realizadas por un administrador deben ser revisadas antes de completar la acción, y necesitan del flujo de aprobación nivel uno.

Para las solicitudes de modificación de cuentas por parte de un usuario estándar el flujo de aprobación es diferente de los anteriores, ya que la solicitud debe ser revisada, luego debe ser aprobada y finalmente se completa el flujo de aprobación para que sea modificada, por lo que necesita del flujo de aprobación nivel dos. Ver **anexos IV, V** y **VI.**

2.5. Conclusiones.

En este capítulo se capturó la visión del proyecto y se realizó dentro de la fase de Planificación todo lo referente en cuanto a la obtención de los escenarios. Se determinó la longitud de las iteraciones así como los escenarios que son necesarios implementar. Se priorizó cada escenario y fueron divididos en tareas para su implementación. En la fase de Construcción se creó la arquitectura de solución y se realizó una descripción de las distintas capas del sistema y fue definido el diagrama de clases. Se logró el diseño del *Workflow* para la construcción del motor de tareas.

CAPÍTULO 3: ESTABILIZACIÓN.

3

3.1. Introducción.

En este capítulo se expone todo lo relacionado con el flujo de trabajo Estabilización, cerrando con su realización el desarrollo del Gestor de tareas para el Aprovisionamiento de usuarios. Se exponen las pruebas unitarias realizadas así como sus resultados.

3.2. Pruebas.

Las pruebas de software, son los procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. Constituyen un flujo dentro de una de las fases en el desarrollo de software consistente en probar las aplicaciones construidas. Con las pruebas se reducen drásticamente los problemas y tiempos dedicados a la integración. En las pruebas se simulan las dependencias, lo que permite que podemos probar el código sin disponer del resto de módulos.

Visual Studio Team System 2008 incluye la opción de crear proyectos de tipo *Test*, que permite crear, administrar, editar, ejecutar y almacenar los resultados obtenidos de las diferentes pruebas que permite realizar como son las pruebas manuales, unitarias, de *Web* y de carga así como permite medir la cobertura de código.

3.2.1. Pruebas Unitarias.

En programación, una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

El objetivo de las pruebas unitarias es el aislamiento de partes del código y la demostración de que estas partes no contienen errores.

Las pruebas fomentan el cambio y la refactorización. Si se considera que el código es mejorable se puede cambiar sin ningún problema. Si el cambio no estuviese realizado correctamente las pruebas avisarán de ello. Se reducen drásticamente los problemas y tiempos dedicados a la integración. En las pruebas se simulan las dependencias lo que permite que se pueda probar el código sin disponer del resto de módulos.

Las pruebas unitarias que se realizaron a los servicios del sistema para validar la salida de los datos le aseguran al programador que su solución no presenta errores lógicos de programación y ante una entrada de datos determinada por el probador los valores obtenidos son los esperados.

La realización de las pruebas estuvo muy ligada a la planificación de las distintas iteraciones de desarrollo que se planificaron en dependencia de la complejidad de los escenarios a implementar.

Las pruebas aplicadas a la versión funcional del sistema arrojaron una serie de errores o no conformidades que han sido representadas en la **Figura. 11** *Resultado de las pruebas unitarias* y donde se observa cómo se les dio solución en cada iteración hasta quedar el sistema funcionando sin errores en el código.

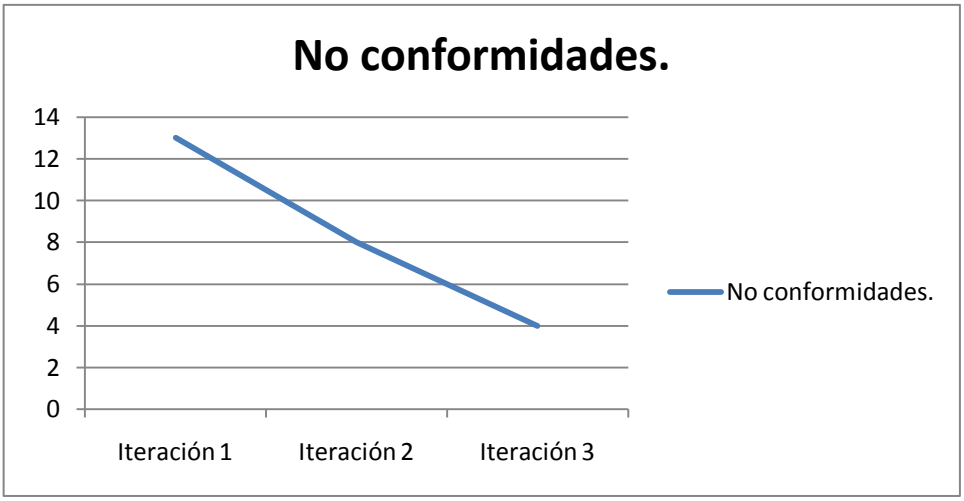


Figura 11. Resultado de las pruebas unitarias.

Durante la primera iteración se obtuvo 13 no conformidades, 8 en la segunda y 4 en la tercera iteración.

A continuación se muestran las pruebas unitarias realizadas al método *FindAllAccountsTest*. Arrojando como resultado la prueba realizada que el código no presenta problemas. Mostrándose los resultados en la Tabla 30. Resultado de la prueba Unitaria realizada al método *FinAllAccountsTest*.

Prueba de Unidad		
Nombre Prueba: *FindAllAccountsTest*		
Estado: Satisfactoria	**Tipo**: Caja Blanca	**Última ejecución**: 10/06/2010
Ejecutado por: Carlos Pujol Vargas	**Verificado por**: Yoandy Rodríguez Martínez	
Descripción: Para poder ejecutar la prueba se debe introducir una lista con todos los datos guardados en la tabla *Account* de la base de datos de la aplicación. Uno de los aspectos relevantes a la hora de realizar esta prueba es que solo se probó que la cantidad de elementos en la lista y el identificador de la cuenta fueran iguales, ya que el método presenta gran cantidad de variables de entrada.		
Entrada: *List*<*Account*>		
Criterio de aceptación: Se muestra el listado de las cuentas.		

Resultado:

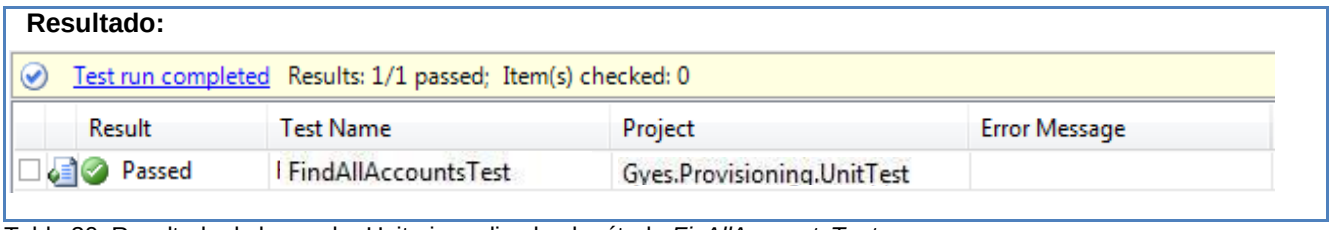


Tabla 30. Resultado de la prueba Unitaria realizada al método *FinAllAccountsTest*.

Ver otras pruebas en **Anexo VII.**

3.3. Conclusiones.

Con el objetivo de validar la solución propuesta se condujeron un conjunto de pruebas, enfocándose fundamentalmente en las pruebas unitarias, sin descartar las pruebas de validación que son necesarias como paso inicial para las pruebas de unidad debido a que verifican el flujo de la información. Se creó un proyecto de prueba y a cada clase a probar se le generaron sus funciones, se diseñaron los juegos de datos, se anotaron y evaluaron los resultados obtenidos. Es importante destacar que las pruebas unitarias no descubrirán todos los errores del código. Por definición, sólo prueban las unidades por sí solas. Por lo tanto, no descubrirán errores de integración, problemas de rendimiento y otros problemas que pudieran afectar a todo el sistema en su conjunto. Las pruebas que constan en el presente documento son las que se practicaron a las principales funcionalidades del módulo, aunque pruebas similares se aplicaron al sistema completo.

CONCLUSIONES GENERALES

El desarrollo de este trabajo permitió elaborar el módulo Gestor de Tareas para Aprovisionamiento de Usuarios del Sistema de Administración de Identidades del Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas, para ello se cumplió con los objetivos y tareas propuestas:

- Se realizó un estudio de las herramientas y tecnologías necesarias para realizar un correcto diseño e implementación del módulo, arrojando como resultados más relevantes la utilización de lenguajes, herramientas y metodologías de avanzada para la concepción de la aplicación.
- El sistema informático fue desarrollado utilizando C# como lenguaje de programación sobre la plataforma .NET propiciando seguridad y la utilización de *frameworks* que esta posee y Oracle como Sistema Gestor de Bases de Datos por la necesidad de un continuo crecimiento del trabajo y un rápido acceso a los registros almacenados.
- El módulo cuenta con un diseño orientado a objetos cumpliendo con los patrones seleccionados en su diseño. La arquitectura está representada por 5 capas lógicas que dan un alto nivel de encapsulamiento de las responsabilidades.
- El sistema desarrollado permite gestionar las cuentas de usuario aprovisionadas en repositorios suscritos al Sistema de Administración de Identidades.
- La aplicación se desarrolló en el período de tiempo establecido y cumple con todas las funcionalidades que se describieron.
- En los casos de prueba descritos y efectuados, correspondientes a la estabilización del sistema, se detallaron las pruebas a realizar sobre los escenarios seleccionados para comprobar y validar sus funcionalidades en el sistema. La gran parte de los resultados arrojados por estas pruebas fueron satisfactorios, siendo la base para conocer que el sistema está apto para ser liberado en un futuro.

RECOMENDACIONES

Luego de haber desarrollado una primera versión del Motor de Tareas para el Servidor de Aprovisionamiento se recomienda:

- Crear un componente integrado al módulo Motor de Tareas para el diseño y creación de flujos de trabajo en tiempo de ejecución interpretando los deseos del usuario y convirtiéndolos en flujos de aprobación.
- Extender el servicio de persistencia de los flujos de trabajos para que estos puedan ser almacenados en ficheros en el propio servidor donde se ejecutará el motor, posibilitándose así la independencia con respecto a la conexión a la base de datos.

REFERENCIAS BIBLIOGRÁFICAS

(1). *Scope of the Single Sign-On Standard.* [En línea] The Open Group. [Citado el: 23 de Enero de 2010.] http://www.opengroup.org/security/sso/sso_scope.htm#mgt_iface..

(2). Kioskea Network . *Kioskea .Net.* [En línea] [Citado el: 23 de Enero de 2010.] http://es.kioskea.net/contents/utile/fai.php3.

(3). MSDN. [En línea] Microsoft Corporation. [Citado el: 10 de Febrero de 2010.] http://msdn.microsoft.com/en-us/library/aa286483.aspx.

(4). IBM Corporation **.** *IBM Tivoli Identity Manager.* [En línea] [Citado el: 19 de Enero de 2010.] http://www.rational.com.ar/brochures/tivoliidentitymgr.pdf.

(5).Oracle. [En línea] 2010. [Citado el:16 de Febrero de 2010.] http://www.oracle.com/index.html.

(6). Sun Microsystems [En línea] [Citado el: 19 de Enero de 2010.] https://www.suntrainingcatalogue.com/eduserv/client/loadCourse.do?coId=es_ES_IDM-4485&coCourseCode=IDM-4485&l=es_ES

(7). Novell. *Novell Identity Manager.* [Online] 2010. [Cited: noviembre 23, 2009.] http://www.novell.com/products/identitymanager/.

(8). CA tecnologies. [En línea] 2010. [Citado el: 16 de febrero de 2010.] http://www.ca.com/es/default.aspx.

(9). UNIFIED MODELING LANGUAGE . *UML Resource Page.* [Online] 2010. [Cited: febrero 16, 2010.] http://www.uml.org/.

(10). Altova. [En línea] [Citado el: 24 de marzo de 2010.] http://www.altova.com/umodel.html.

(11). *MSF for Agile Software Development Process Guidance.* [Online] Microsoft Corporation, 2010.

(12). IIS. [En línea] [Citado el: 19 de Enero de 2010] http://www.desarrolloweb.com/manuales/36/

(13). POSIX. [En línea] [Citado el: 23 de Abril de 2010] http://www.lynuxworks.com/products/posix/posix.php3

BIBLIOGRAFÍA

1. ABE VICENTE, A.; V. F. A. SANTANDER, et al. JGOOSE: A REQUIREMENTS ENGINEERING TOOL TO INTEGRATE I* ORGANIZATIONAL MODELING WITH USE CASES IN UML., 2009.

2. CORPORATION, M. Configuración del motor de base de datos: aprovisionamiento de cuentas, 2010. [2010]. Disponible en: http://msdn.microsoft.com/es-es/library/cc281849.aspx

3. ---. IIS. Microsoft Corporation, 2010.

4. GRACIA, J. Manual de ASP.NET, 2004. [2009]. Disponible en: http://www.webestilo.com/aspnet/

5. MCGIBBON, M. K. A. B. UML Xtra-Light: How to Specify Your Hardware Requirements, PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE, 2007. [2010]. Disponible en: http://bibliodoc.uci.cu/pdf/0521892422.pdf

6. MYERS, B. R. Foundations of WF: An Introduction to Windows Workflow Foundation. 2007. p. ISBN-13 (pbk): 978-1-59059-718-7

7. READSHAW, N. Tivoli Identity Manager, IBM, 2010.

8. SUNMICROSYSTEMS. Sun IdentityManager 8.1 Release Notes. 2009. p.

9. TRILLES, P. Oracle Identity Manager, Financial Tech Megazine, 2010. [2009]. Disponible en: http://www.financialtech-mag.com/000_estructura/index.php?ntt=10007&vn=1&sec=25&idb=1

10. RabbitMQ. Trademark of Rabbit Technologies Ltd., 2010. Disponible en: http://www.rabbitmq.com/

11. Novell eDirectory, WorldLingo Translations LLC, 2010. Novell. Novell Making IT Work As One. Disponible en: http://www.novell.com

12. MICROSOFT. Windows Server 2003 R2, Microsoft Corporation, 2010.

13. Open System Specification. Echelon Corporation. Disponible en: http://www.echelon.com/solutions/building/papers/OpenSpecFrameworkv4.pdf

14. MSF for Agile Software Development. Disponible en: http://process.osellus.com/sites/wiki/MSF%20for%20Agile%20Software%20Development/Wiki%20Pages/Home.aspx

15. Extensible Markup Language (XML). W3C. Disponible en: http://www.w3.org/XML/

16. Allround Automations. 2010. Allround Automations, PO Box 40014, 7504 RA, Enschede, THE NETHERLANDS. Disponible en: http://www.allroundautomations.com/plsqldev.html

GLOSARIO DE TÉRMINOS

A

(ASP): Es un *framework* para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. ***7***

B

BPMN (*Business Process Management Notation*): Notación para el Modelado de Procesos de Negocio. Permite el modelado de asuntos de negocio donde se presentan gráficamente las diferentes etapas del proceso del mismo. Esta notación coordina la secuencia de eventos que fluyen entre los diferentes procesos y participantes.

C

C sharp: es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET,

F

Framework: Un *framework*, en el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. .....***IV***, ***20***, ***21***, ***22***, ***25***, ***28***, ***48***, ***49***

G

GRASP: Son patrones generales de software para asignación de responsabilidades, es el acrónimo de "*General Responsibility Assignment Software Patterns*". Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software…….**52**

I

IBM: Es una empresa multinacional que fabrica y comercializa herramientas, programas y servicios relacionados con la informática. ***2***, ***5***, ***6***, ***11***, ***12***, ***14***, ***65***

Identity Management, IdM: Es un sistema integrado de políticas y procesos organizacionales que pretende facilitar y controlar el acceso a los sistemas de información y a las instalaciones. ***1***

L

Language Integrated Query: Language-Integrated Query (LINQ) es un conjunto de características en Visual Studio 2008 que agrega eficaces capacidades de consulta a la sintaxis de los lenguajes *C sharp* y *Visual Basic*. ***21***

LDAP: Protocolo Ligero de Acceso a Directorios es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. ***4***, ***14***

M

MS SQL: Es un sistema para la gestión de bases de datos producido por *Microsoft* basado en el modelo relacional. ***14***

MSF: *Microsoft Solutions Framework* es un conjunto de ingeniería de *software* procesos, principios y prácticas probadas por objeto permitir a los desarrolladores para lograr el éxito en el desarrollo del ciclo de vida del *software*. ***III***, ***19***, ***20***, ***26***, ***28***

P

PST: Estándar que se ocupa de la semántica necesaria para la dotación de puntos de servicio a las solicitudes de cambio relativas a la gestión de provisión de determinados servicios ***8***, ***9***, ***10***

S

SAML: Es una versión de la norma OASIS SAML para la autenticación y el intercambio de datos de autorización entre dominios de seguridad. ***2***, ***5***

Smard card: es cualquier tarjeta del tamaño de un bolsillo con circuitos integrados que permiten la ejecución de cierta lógica programada **10**

Single Sign-On **(SSO):** Es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación. ***5***

SPML: La lengua del margen de beneficio del aprovisionamiento del servicio es el estándar abierto para la integración y el *interoperation* de las peticiones del aprovisionamiento del servicio. ***2***, ***13***

Streaming: (flujo de datos) permite reproducir contenidos multimedia en la red sin tener que descargar el archivo entero para posteriormente reproducir su contenido, ya que reproduce la secuencia de audio/vídeo mientras la descarga. En el *streaming* bajo demanda por Internet el contenido multimedia es reproducido a iniciativa del cliente, por lo que puede ser visualizado en cualquier instante. ***11***

U

UML: Lenguaje Unificado de Modelado o *Unified Modeling Language* (por sus siglas en inglés) es el lenguaje de modelado de sistemas de software que se utiliza para especificar, visualizar, modificar, construir y documentar los artefactos que se obtienen durante el desarrollo. ***18***, ***19***, ***65***

X

XML: *Extensible Markup Language* (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). ***13***, ***17***, ***19***, ***25***

Z

z/OS: Es el sistema operativo actual de los mainframes de IBM. ***12***