Universidad de las Ciencias Informáticas

Facultad 1

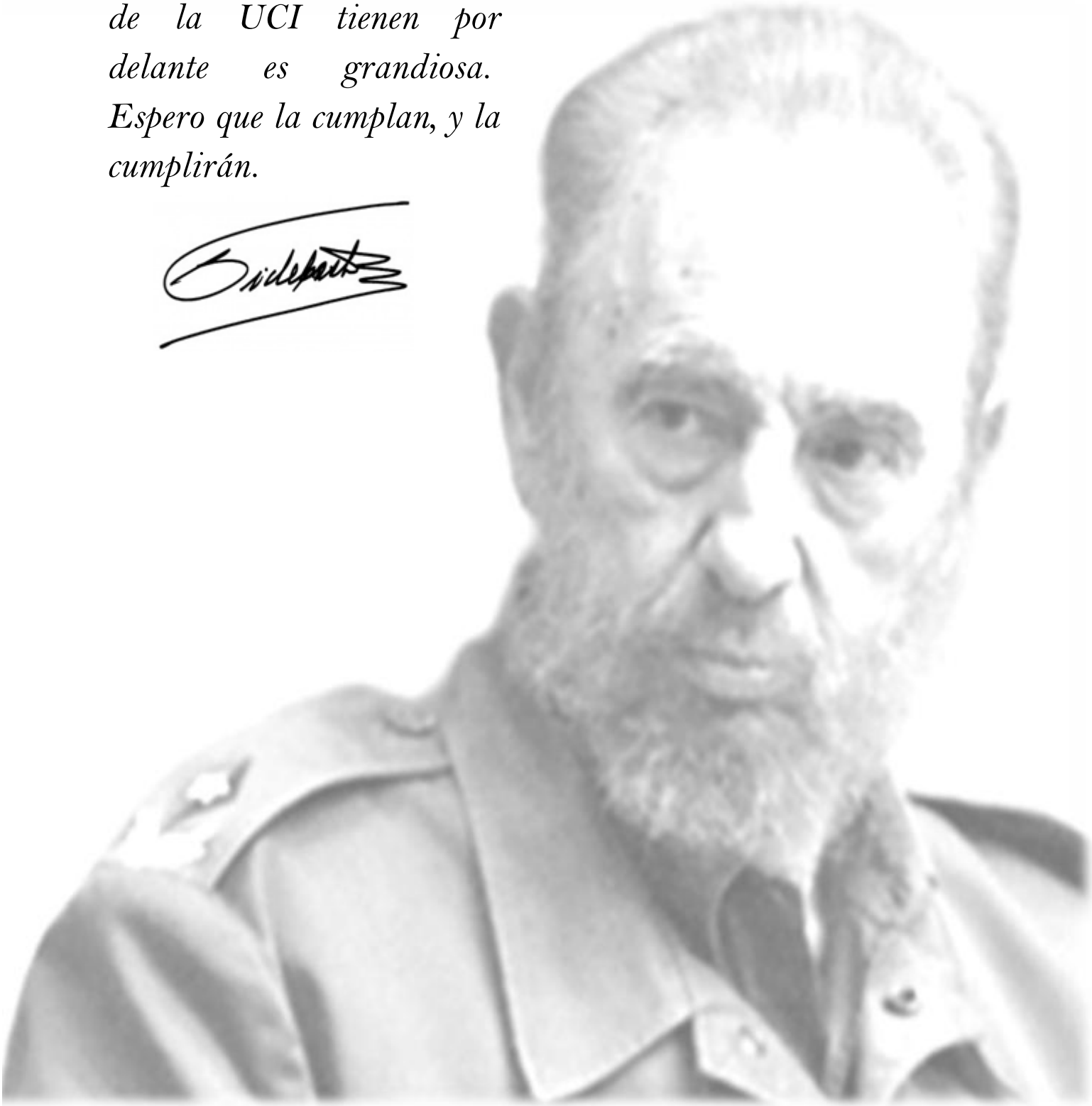
MÓDULO ADMINISTRACIÓN PARA LA APLICACIÓN DE APROVISIONAMIENTO DE USUARIOS DEL SISTEMA DE ADMINISTRACIÓN DE IDENTIDADES

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

Autores: Viviana Muñiz López.
Armando León Martínez.
Tutor: Ing. Diovis Proenza Labadie.

Ciudad de La Habana, Junio de 2010.
"Año 52 de la Revolución"

La tarea que los graduados de la UCI tienen por delante es grandiosa. Espero que la cumplan, y la cumplirán.



Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales del mismo, con carácter exclusivo.

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

Firma del autor: _____
Viviana Muñiz López

Firma del autor: _____
Armando León Martínez

Firma del tutor: _____
Diovis Proenza Labadie

Agradecimientos

«Autores»

A los profesores del proyecto, al tutor.

A todos aquellos que de una forma u otra estuvieron presentes en la realización de este trabajo. Muchas gracias.

«Viviana»

Llegó el momento de la verdad, la TESIS a la que se le dedica más tiempo que a la familia, pareja o cualquier hobby y pone en práctica todo lo aprendido durante estos 5 años de carrera. Para mí no hay mayor satisfacción que agradecerle a quien me brindó su mano cuando lo necesité y que representa mucho en mi vida aunque no lo crea así.

- ✓ En primer lugar a mis viejitos, mi mamá y mi papá, que bastante se han sacrificado por mí.
- ✓ A mi familia por pensar que todavía soy una niña y complacer todos mis caprichos, principalmente mi hermano a quien admiro y respeto mucho.
- ✓ A mi novio Carly por darme fuerzas para seguir, sin él no hubiera podido vencer esta batalla, agradecer también a su familia que desde que me conocieron fueron muy cariñosos conmigo.
- ✓ A mi compañero, el negrito cafetero le agradezco de todo corazón.
- ✓ A mis amigas del politécnico que siempre están para lo que necesite, Yado, Yiso e Isma que nunca intentaría reemplazarlas porque es imposible que lo logre.
- ✓ Muy importante agradecerle a mi seleccionado grupo de primer año en la universidad que aunque tomaremos distintos caminos siempre estarán en mi corazón, Ana Elda mi madrecita, Yilena Oviedo "mi niña" y Elien Juan Alarcón "el amor de mi vida" que forman parte de mi familia desde el primer día cuando me matriculé.
- ✓ A las "súper poderosas" Mily, Yeny Lauris, mi amore Chanel, Grethiña, Eli, Dole, Yoha mi profe, (…), en fin, menciono algunas porque son muchas, a todas muchas gracias por ganarse mi amistad.
- ✓ A mis chicos, Pinillo, Serge, William, Evaristiquin, Yurisbel, Dariel y Lorian que espero no se olviden nunca de mí por la cantidad de veces que los molesté.
- ✓ A todas mis amistades muchas gracias y los/las quiero mucho.

«Armando»

Son muchas las personas especiales a las que me gustaría agradecer por su amistad, apoyo, ánimo y compañía en las diferentes etapas de mi vida:

- ✓ A mi madre por ser la más especial de todas.
- ✓ A mi padre que siempre ha estado a mi lado brindándome su confianza y cariño.
- ✓ A mi abuela Anolan que siempre ha estado ahí para apoyarme y ayudarme a cumplir mis metas.
- ✓ A mi abuelo por haberme guiado siempre por los buenos caminos.
- ✓ A mi abuelita Amparo por quererme tanto.
- ✓ A mi hermanita Ady que la quiero mucho.
- ✓ A mi tía Alicia por el apoyo y la confianza que siempre ha depositado en mí.
- ✓ A mi Rosita que me ha dado tanto amor en tan poco tiempo.
- ✓ A mis primos Yosni, Heidy, Yuli y mis tíos Hano, Rey y Esty.
- ✓ A la familia Mederos por su ayuda siempre que la he necesitado.
- ✓ A todos mis amigos pero en especial a: mis hermanos Carlos, Alejandro T. y Alejandro V., Plinio, Raymundo, Adonis, Tony Moisés, David (el primo), Joe Luis, Jesús y Reinier. A mis hermanas: Thais, Yanet, Inelys, Yaricel, Anamaris y Eimy. Gracias por estar ahí siempre que he necesitado de ustedes.
- ✓ A mis amigos del proyecto: Evaristo (kikon), Osniel (el griña), Robertón (the boss) y Mario (macumbele).
- ✓ A mi compañera de tesis (la vivijagua), por las noches de trabajo que compartimos y por buscarme el café.

Dedicatoria

A la familia y amigos.

A Fidel por tener la idea de crear la universidad de "excelencia".

«Autores»

Resumen

El ámbito del presente trabajo de diploma es la gestión del proceso de aprovisionamiento de usuarios, su título es "Módulo Administración para la Aplicación de Aprovisionamiento de Usuarios del Sistema de Administración de Identidades". 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, 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, centrándose en el análisis del proceso de administración de los mismos y de los principales sistemas existentes a nivel mundial en esta esfera. El propósito que se persigue con este trabajo es el desarrollo del módulo Administración 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 llevar a cabo una serie de procesos complejos referente al aprovisionamiento, tales como administrar el flujo de aprobación vinculado a las cuentas de usuarios, gestionar las políticas de contraseña de estas cuentas en el sistema y proveer los métodos necesarios para configurar las acciones automáticas que se realizarán sobre las cuentas de usuario en una futura integración con los demás módulos del Sistema de Administración de Identidades.

Palabras claves

- Aprovisionamiento.
- *MSF* Ágil.
- *Gestión de Identidades (IdM).*

Índice

Introducción

El avance de la Informática y las Telecomunicaciones ha propiciado que el crecimiento de una institución o empresa sea directamente proporcional al uso que hacen de las tecnologías para automatizar sus procesos. Esta automatización ha conllevado a cambiar la forma en que se asegura la información, pasando de un contexto físico a un contexto digital.

La buena marcha de una empresa depende en gran medida del flujo de información y de la disponibilidad de los datos en el momento adecuado; pero también de la confidencialidad de los mismos, la seguridad de las transacciones de la información y la posibilidad de acceso desde cualquier punto. Siendo imprescindible mantener un control sobre qué usuarios pueden acceder a los datos, además de dónde, cómo y cuándo pueden hacerlo. Para ello se hace necesaria una ágil gestión de las identidades digitales. Una identidad digital constituye la representación de características fundamentales de una entidad o individuo dentro de un sistema digital.

La Gestión de Identidades, conocida por su acrónimo IdM, siglas en inglés que significan "Identity Management", es uno de los aspectos fundamentales en el futuro de la tramitación por medios electrónicos, define todo el entorno de problemas y soluciones relacionados con la gestión de la identificación de identidades digitales, permitiendo simplificar el proceso de gestión de usuarios mediante la creación y mantenimiento de perfiles y concediendo o denegando el acceso a los recursos apropiados (físicos o lógicos) por las personas adecuadas. La eficacia de los sistemas de gestión de identidades está dada principalmente por la centralización de los procesos de identificación de los usuarios.

En el actual entorno de negocios, la administración de los usuarios y de los accesos a los sistemas informáticos de la empresa no es una tarea sencilla. La información acerca de personas, aplicaciones y recursos está dispersa en la mayoría de las empresas y sigue proliferando. Existen innumerables riesgos que enfrenta una organización y que demuestran la necesidad de un proceso eficiente de gestión de identidades, entre los que se pueden mencionar: la dificultad en la generación de informes detallados sobre las cuentas de usuario existentes, un mayor nivel acceso que el requerido a los sistemas y aplicaciones fundamentales, el proceso de aprobación y solicitud de una cuenta es realizado en papel (nóminas de usuarios), la información del usuario es guardada en lugares de almacenamiento de datos redundantes e incoherentes y la existencia de cuentas de usuario creadas para el mismo usuario en numerosos sistemas. Como solución a muchos de estos problemas se encuentra el aprovisionamiento de usuarios, parte esencial de la gestión de identidades.

Un sistema de aprovisionamiento de usuarios (*provisioning*) constituye una solución para administrar de manera centralizada todo el ciclo de vida de las cuentas de usuarios en una organización, aportando ventajas sobre la información del usuario almacenada en los directorios de las organizaciones. Acelera la concesión y revocación de cuentas de usuario, acciones que pueden estar regidas por un flujo de aprobación definido mediante la administración del sistema.

La administración de un sistema de aprovisionamiento de usuarios es un proceso complejo que posibilita una configuración eficiente de las cuentas creadas así como la implantación de un conjunto de medidas o restricciones que el sistema y todos sus implicados deben cumplir.

Existen gran variedad de aplicaciones que automatizan los procesos de provisión de usuarios y roles, gestión de contraseñas y control de acceso. Actualmente, el uso de estas soluciones está restringido solo a las organizaciones que puedan acceder a los costosos precios que conlleva su adquisición y de la capacidad que tenga el sistema de adaptarse a las condiciones de una empresa con mayor o menor desarrollo tecnológico.

En Cuba las tareas de aprovisionamiento generalmente son lentas y propensas a errores humanos debido a los diseños pocos satisfactorios y descentralizados que presentan. Esta infraestructura que sustenta la provisión unida a la complejidad del proceso de negocio de la organización, provocan que resulte insostenible lograr coherencia, agilidad y calidad en los servicios.

Uno de los compromisos que tiene la Universidad de las Ciencias Informáticas (UCI) con la Revolución es desarrollar sistemas para su uso nacional e internacional. La nueva estrategia trazada por la universidad para alcanzar resultados relevantes en la producción ha traído consigo que los proyectos sean agrupados por centros o áreas de desarrollo, es entonces cuando surge el Centro de Identificación y Seguridad Digital (CISED) para la creación de soluciones vinculadas a las identidades de los usuarios. Un ejemplo de esto es el Sistema de Administración de Identidades, pensado para integrar componentes de autorización, autenticación y aprovisionamiento de usuarios. Para este último se necesita un módulo que administre el proceso de provisión de los usuarios.

Teniendo en cuenta lo anteriormente planteado se identifica el siguiente **problema a resolver**: ¿Cómo gestionar las cuentas de usuarios en la Aplicación de Aprovisionamiento del Sistema de Administración de Identidades?

Por tanto, el **objeto de estudio** que guiará la investigación se centrará en el proceso de aprovisionamiento de usuarios. El cual actúa directamente sobre el siguiente **campo de acción**: administración de cuentas de usuarios. Para dar cumplimiento al presente trabajo de diploma se plantea como **objetivo general**: Implementar el módulo Administración para la Aplicación de Aprovisionamiento de Usuarios del Sistema de Administración de Identidades.

Además, se trazaron los siguientes **objetivos específicos:**

- Realizar un estudio de las herramientas y tecnologías para diseñar e implementar el módulo.
- Diseñar el módulo Administración para la Aplicación de Aprovisionamiento de Usuarios.
- Implementar el módulo Administración para la Aplicación de Aprovisionamiento de Usuarios.
- Realizar las pruebas al funcionamiento del módulo implementado.

Dando cumplimiento a estos objetivos se espera materializar la **idea a defender** siguiente: La implementación del módulo Administración permitirá gestionar todo el ciclo de vida de las cuentas de usuarios de la Aplicación de Aprovisionamiento del Sistema de Administración de Identidades.

Tareas a cumplir:

- Realización de un estudio sobre los sistemas de aprovisionamiento.
- Identificación de las necesidades del módulo a desarrollar en el sistema.
- Revisión del proceso de administración en diferentes sistemas de aprovisionamiento existentes.
- Desarrollar el diseño del módulo Administración.
- Desarrollar la implementación del módulo Administración en la plataforma establecida.
- Desarrollar las pruebas para comprobar la calidad del funcionamiento del módulo implementado.

MÉTODOS CIENTÍFICOS DE LA INVESTIGACIÓN

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

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 Aplicación de Aprovisionamiento de Usuarios del Sistema de Administración de Identidades.

ESTRUCTURA DEL SISTEMA

Capítulo 1: Fundamentación teórica: Se efectuará un estudio del estado del arte analizando las principales aplicaciones dedicadas al aprovisionamiento de usuarios, exponiendo las características esenciales que sirvan de base para el sistema propuesto. Además, se realizará el análisis de las herramientas, metodologías, lenguajes que existen en la actualidad y que serán utilizados en el proceso de desarrollo de la aplicación.

Capítulo 2: Características del sistema: Se realizará la especificación de las funcionalidades del sistema y se determinarán las actividades a realizar por las personas que interactúan con el mismo. Además, se realizará el diseño de la aplicación donde se obtendrán los elementos necesarios para la futura implementación del módulo.

Capítulo 3: Desarrollo de la solución propuesta: En este capítulo se establecerá la arquitectura y los patrones de diseño a utilizar para realizar la implementación de la aplicación. Se obtendrá el diseño del modelo de base de datos y el diagrama de clases.

Capítulo 4: Estabilización del sistema: Se ejecutarán un conjunto de pruebas a la aplicación para verificar la calidad de la misma y se realizará un resumen de evaluación de estas pruebas.

Capítulo 1 Fundamentación teórica

En el presente capítulo se expone el fundamento teórico que apoya la solución dada al problema. Se darán a conocer los principales conceptos relacionados a la gestión de identidades enfatizando en el proceso de aprovisionamiento de usuarios. Se realizará un estudio del estado del arte revisando la documentación referente al tema para comprenderlo en su totalidad y se obtendrá un análisis de las herramientas, metodologías y lenguajes que serán usados en el desarrollo de la aplicación.

1.1. Conceptos asociados al dominio del problema.

Se muestran a continuación los principales conceptos relacionados con el dominio del problema y que aportarán un breve conocimiento de todos los aspectos que se tratarán posteriormente.

1.1.1. Seguridad.

La seguridad de sistemas o aplicaciones informáticas, son aquellas características que indiquen el índice de protección contra cualquier riesgo de agresión, resguardando la información y restringiendo el acceso al sistema de personal no autorizado.

La seguridad consta de tres aspectos fundamentales: la confidencialidad, que es la propiedad de prevenir la divulgación de información a personas o sistemas que no estén autorizados a poseerla, la integridad que es la que mantiene los datos libres de modificaciones no autorizadas y la disponibilidad que es la que permite que la información se mantenga a disposición de quienes deben acceder a ella, ya sean personas, procesos o aplicaciones.(1) Estos tres aspectos permiten resguardar eficientemente la identidad digital de los usuarios de cualquier sistema informático.

1.1.2. Identidad digital.

La identidad digital determina la identificación de un usuario, sea éste una persona o un sistema informático. No hay diferencia entre ambos dentro de un entorno de gestión de identidades. En resumen, una identidad sintetiza la información acerca de personas, aplicaciones y recursos que está distribuida en directorios y bases de datos en la mayoría de las empresas.(2)

1.1.3. Gestión de identidades.

La gestión de la identidad, es un área administrativa que se ocupa de la identificación de individuos en un sistema para cualquier organización y controla el acceso a los recursos al imponer restricciones a las identidades establecidas.(2)

Engloba áreas de negocio que se dedican a las siguientes tareas:

- Aprovisionamiento automatizado de cuentas de usuario y contraseñas.
- Implantación de sistemas de identificación y autenticación única (también denominado "*single sign-on*").
- Gestión centralizada de las atribuciones de los usuarios, basada en directorios de usuarios, habitualmente basados en LDAP(*Lightweight Directory Access Protocol*).
- Modelo de autorizaciones, que concentra en un solo punto las autorizaciones de acceso.

Ventajas fundamentales de implementar un sistema de gestión de identidad:

- Desde el punto de vista de la seguridad, proporciona un control notable y permite un incremento de confianza en las compañías. Ofrece información de quién tiene acceso a qué en la empresa, un elemento significativo dentro del control de acceso.
- Desde la perspectiva de la empresa, para las organizaciones que están encaminadas en un modelo de negocio en línea, la identidad es un componente esencial. Antes, los negocios solían cerrarse con un simple apretón de manos; en la actualidad, la gestión de identidad permite dar ese apretón de manos a millones de personas que se encuentren en lugares distantes con la mayor fiabilidad. Además, permiten tener un conocimiento de los usuarios, ya sean empleados, clientes, proveedores, los que intervengan en el sistema.(3)

1.1.4. Aprovisionamiento de usuarios.

El *aprovisionamiento* es un componente clave en las soluciones de gestión de identidades muy usado particularmente dentro de organizaciones donde los usuarios pueden estar representados por múltiples objetos en múltiples sistemas. El concepto "aprovisionamiento de usuarios" encierra el ciclo de vida de la gestión de las identidades digitales. Constituye la solución para administrar de manera centralizada el proceso de alta, mantenimiento y baja de cuentas de usuario existentes en varias aplicaciones y sistemas de computación, posibilitando la administración de identidades utilizadas para respaldar las decisiones de control de acceso y autenticación.

Elementos de un sistema de aprovisionamiento:

Según Ernst & Young, una de las mayores empresas de servicios profesionales del mundo enfocada a las necesidades empresariales, una solución de aprovisionamiento de usuarios debe estar integrada por los siguientes elementos:

- **Consola de administración:** se utiliza para interactuar con los componentes restantes del sistema y para configurarlos.
- **Interfaces de la red:** se utiliza para brindar una interfaz accesible al usuario del proceso de flujo de trabajo (*workflow*) para solicitar, revisar y actuar sobre las solicitudes de aprovisionamiento.
- **Motor de aprovisionamiento:** administra la creación, modificación, suspensión o eliminación de cuentas en los sistemas administrados.
- **Motor del *workflow*:** administra los procesos relacionados con la gestión de solicitudes (por ejemplo, para dar de alta, modificar o dar de baja el acceso de un usuario identificado).
- **Depósitos de datos:** almacenan los perfiles de usuarios, las plantillas de cuentas, las políticas de cuentas, las definiciones de roles, las definiciones del proceso de flujo de trabajo, el estado del flujo de trabajo, la información de auditoría, entre otros.
- **Vista de informes:** genera informes de datos de auditoría, flujo de trabajo y aprovisionamiento sobre la base de criterios de búsqueda, ingreso de datos en el campo deseado y tipo de informe. (2)

1.1.5. Administración.

Las empresas afrontan los desafíos de administrar la identidad de los usuarios de muchas formas diferentes. La administración es el proceso de planear, organizar, liderar y controlar los esfuerzos de los miembros de la organización, para lograr objetivos organizativos establecidos.(4) La administración en sistemas de aprovisionamiento de usuarios permite controlar y configurar todo el proceso de concesión y revocación de cuentas de usuario así como las restricciones que debe cumplir el usuario que interactúa con el sistema.

1.2. Sistemas estudiados.

El acceso a la información, el tráfico global que esto genera y la necesidad existente en el mundo en cuanto a la gestión de las identidades digitales, han traído consigo el desarrollo de soluciones y tecnologías que automatizan dicho proceso.

Hoy en día las tecnologías permiten casi todo y el aprovisionamiento de usuarios aparece cada vez con más intensidad en el mercado, aparecen productos de miles de compañías que se van fortaleciendo al competir con los sistemas existentes.

Mediante un estudio realizado se puede confirmar que las compañías líderes que producen sistemas de gestión de identidades constituyen los vendedores más conocidos (basado en las ventas, la presencia mundial y el crecimiento competitivo). (Ver Anexo 1). *Oracle* e *IBM Tivoli* dominan el mercado de la gestión de identidades, mientras que *Novell* y *CA* representan también una presencia significativa. Por su parte *SUN* presenta actualmente un reducido número de productos debido principalmente a que fue adquirida por *Oracle* aunque es todavía uno de los proveedores con más experiencia en la construcción de sistemas de aprovisionamiento de usuarios en el mercado. *Novell* sigue demostrando que es un competidor a respetar a través de acciones de *marketing* estratégico y las capacidades de sus productos. CA obtuvo sus mayores progresos en el año 2009 y aunque no ha introducido cambios significativos desde entonces sigue teniendo auge en el sector.

La realización de un minucioso estudio sobre las principales compañías que producen software de gestión de identidades y los principales sistemas a nivel mundial arrojó como resultado un conjunto de características que se exponen a continuación:

***IBM Tivoli Identity Manager (ITIM) v.5.1 (junio de 2009)*:** *ITIM* proporciona el *software* y los servicios para implementar soluciones basadas en la política de aprovisionamiento. Este *software* de gestión de identidades constituye una solución segura, automatizada y basada en políticas para gestionar los privilegios de usuario. Entre sus principales beneficios se encuentran:

- Posibilita la gestión de acceso de los usuarios basado en solicitudes para aprobar los accesos de los usuarios, para los roles, las cuentas y las autorizaciones de acceso detalladas.
- Presenta una interfaz mejorada para los usuarios que se puede adaptar con facilidad al aspecto de la empresa, personalizar para determinados tipos de usuario (auditores, directores), e integrar con los portales corporativos.
- Proporciona políticas fáciles de configurar mediante asistentes y plantillas que forman parte del paquete de *software* de gestión de identidades.
- Agiliza los despliegues y reduce la curva de aprendizaje de los nuevos usuarios.

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. 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 & Access Management Suite (enero de 2009): Permite a las empresas administrar todo el ciclo de vida de identidad de los usuarios en todos los recursos empresariales tanto dentro como fuera del *firewall*. Posibilita aplicar una protección minuciosa a los recursos empresariales y eliminar automáticamente los privilegios de acceso latente.
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 del producto. *Oracle Identity Manager* puede ser integrado a servicio de prueba lo que minimiza el riesgo basado en las propias decisiones que toma el usuario.

Sun Identity Manager V.8.1 (junio de 2009): *Sun Identity Manager* permite automatizar el proceso de creación, actualización y eliminación de cuentas de usuario en múltiples sistemas de Tecnología de la Información (TI). *Sun Identity Manager* puede realizar una exploración de auditoría para buscar diversas infracciones en los procesos de provisión y desprovisión de las cuentas de usuarios y, según la configuración, suprimir automáticamente el acceso o enviar una notificación a un administrador. 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*.

Novell Identity Manager v.3.6.1-Novell Sentinel v.6.1: Desarrollada por la empresa Novell, 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 en la misma. Incluye capacidades para el aprovisionamiento automatizado de cuentas de usuario, flujos de trabajo de aprobación, la gestión de contraseñas y gestión de datos en los directorios de usuario, aplicaciones, bases de datos y plataformas de sistema operativo. Incluye módulos de integración para varios sistemas comunes de los clientes: *Microsoft Active Directory* y directorios LDAP v3.

CA Identity Manager Release 12 (junio 2008): Esta última versión es una plataforma centralizada que provee de una automatización basada en flujos de trabajo para la administración de identidades de usuarios. También permite la creación de cuentas o solicitudes de cambios de contraseña a través de peticiones 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. La plataforma es capaz de manejar la gestión de la identidad sobre múltiples sistemas tales como, *Windows NT/2000/2003, Solaris, Linux,* directorios específicos como *Active Directory*, estándares como *LDAP* y

ODBC[1] y base de datos tales como *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, incluyendo habilidades específicas para la administración de contraseñas como son el soporte para el olvido y la sincronización de las mismas.

Como resultados principales del estudio realizado sobre estos sistemas de gestión de identidades se identificaron las características más importantes relacionadas con la administración de los sistemas de aprovisionamiento de usuarios y que podrían ser utilizadas como referencia para la implementación del módulo que se desea construir y su integración con los demás módulos de la Aplicación de Aprovisionamiento.

1.3. Tecnologías y herramientas a utilizar.

Las tecnologías han experimentado un vertiginoso desarrollo en los últimos años. Para la creación de cualquier *software* se hace imprescindible tener una visión de lo que desea construir, por lo que es de vital importancia contar con las herramientas y lenguajes necesarios para hacer un modelado o trazar una arquitectura a desarrollar.

Para la creación de la aplicación se deben tener en cuenta el lenguaje de programación con que se implementará, el servidor de aplicaciones *web* sobre el cual se publicará, el gestor de bases de datos que se utilizará para la persistencia de la información, así como el *IDE*[2], la plataforma sobre la que funcionará y otras tecnologías necesarias para su desarrollo.

La dirección del Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas seleccionó las tecnologías a utilizar para el modelado e implementación de la aplicación, utilizándose las mismas tecnologías definidas en la línea de la arquitectura del Sistema de Administración de Identidades, debido a que posteriormente esta solución será integrada a este sistema, no obstante, se efectúa una investigación para hacer constar las principales características de estas herramientas, que beneficiarán el desarrollo de la aplicación.

[1] ODBC: *Open DataBase Connectivity,* por sus siglas en inglés, significa abrir la conexión de la Base de Datos en español.

[2] *IDE: Integrated Development Environment,* por sus siglas en inglés, significa entorno de desarrollo integrado en español.

1.3.1. Lenguaje Unificado de Modelado.

El Lenguaje Unificado de Modelado (*UML, Unified Modeling Language* por sus siglas en inglés) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Permite la modelación de sistemas con tecnología orientada a objetos. (5)

Tiene propiedades, como su amplio vocabulario, que han sido realmente las que han contribuido a hacer de *UML* el estándar de la industria en el análisis y diseño de sistemas de cómputo en la actualidad. (Ver el Anexo 2 para conocer el vocabulario de *UML*)

Se utilizará como notación el Lenguaje Unificado de Modelado para lograr un mayor entendimiento del sistema ya que se pueden modelar y describir secuencialmente todos los procesos que se llevan a cabo según la problemática planteada.

1.3.2. Herramienta *CASE* de Desarrollo de *Software*.

Las herramientas *CASE*[3] son la base para el proceso de análisis y desarrollo de *software*, es habitual usarlas cada vez que se desarrolle un sistema por pequeño o grande que sea, con el objetivo de modelar los aspectos esenciales del mismo, desde sus inicios hasta su culminación, incrementando su calidad.

1.3.2.1. *Altova UModel* 2009.

Altova Umodel 2009 es una herramienta *UML* que permite modelar el desarrollo de *software*. Proporciona un diseño visual de *software* práctico para cualquier proyecto y genera la documentación del mismo. Combina la interfaz visual con funciones de usabilidad para ayudar a nivelar la curva de aprendizaje de *UML*, además de incluir funcionalidades para potenciar a los usuarios con las ventajas del desarrollo de *software UML*.

Las características de *Altova* para el desarrollo de *software que posibilitaron el modelado de la solución propuesta* son:

- Soporte para los 14 tipos de diagramas *UML*.
- Modelado de esquemas *XML (eXtensible Mark-up Language)* en diagramas *UML*.
- Diagramas de proceso de negocio *BPMN* (*Business Process Modeling Notation*).
- Generación de código fuente en el lenguaje *C#*.
- Ingeniería inversa de código fuente y ficheros binarios *C#*.
- Sincronizado de modelo y código a través de ingeniería inversa.

[3] CASE: *Computer Aided Software Engineering,* por sus siglas en inglés, significa ingeniería de *software* asistida por computadora en español.

- Compartir subproyectos para colaboración o reutilización.
- Hipervínculos entre diagramas, documentos y páginas *web*.
- Integración con sistemas de control de versiones.
- Estrecha integración con *Visual Studio*. (6)

1.3.3. Metodología de Desarrollo de *Software*.

Las metodologías son procedimientos o técnicas que ayudan a documentar el proceso de desarrollo del *software*. No existe metodología de *software* universal ni tampoco metodología mejor que otra, cada una tiene sus propias características y se enfocan en funciones específicas. (5)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*[4], *SCRUM y MSF* (*Microsoft Solution Framework*). Estas surgen como una extensión a las metodologías tradicionales para mejorar el desarrollo de sistemas, optimizando las prácticas de desarrollo de *software*. Las tradicionales son metodologías que se centran fundamentalmente en el control del proceso, además de ser muy efectivas para proyectos de gran tamaño. Dentro de las metodologías tradicionales están *OPEN*, *METRICA* 3 y *RUP*[5] que es una de las más utilizadas actualmente.

En un proyecto de desarrollo de *software* la metodología define quién debe hacer qué, cuándo y cómo debe hacerlo. Se desarrollan con el objetivo de dar solución a los problemas existentes en la producción de *software*, que cada vez son más complejos. (5)

1.3.3.1. MSF para el Desarrollo de *Software* Ágil.

Microsoft Solutions Framework es un marco de trabajo que se adapta de forma flexible a las características de cada proyecto. Con la aparición de la herramienta *Microsoft Visual Studio Team System*, se ha actualizado *MSF* a la versión 4.0, produciendo dos variantes: *MSF* para el Desarrollo de *Software* Ágil para el trabajo en entornos que emplean las metodologías ágiles y *MSF for CMMI para la mejora de procesos*. La base para desarrollar la aplicación propuesta será MSF Ágil, que brinda las siguientes fases: (Ver Anexo 3)

- **Visión y Alcance:** En esta fase se definen los objetivos que se persiguen y hasta dónde se quiere llegar con el proyecto. Este proceso se documenta en la declaración de visión que no es más que una descripción del valor del producto que será construido.

[4] XP: *Extreme Programming*, por sus siglas en inglés, significa programación extrema en español.

[5] RUP: *Rational Unified Process*, por sus siglas en inglés, significa proceso unificado del *Rational* en español.

- **Planificación:** Es en esta fase cuando se termina la planificación del proyecto, el equipo prepara las especificaciones funcionales, se realiza el proceso de diseño de la solución y se preparan los planes de trabajo, estimaciones de costo y cronogramas de los diferentes entregables del proyecto.
- **Desarrollo:** Durante esta fase el equipo realiza la mayor parte de la construcción de los componentes (tanto documentación como código) y se crea una solución de la arquitectura a establecer. Como resultado se obtiene una versión de la infraestructura del producto después de aplicarle revisiones de código que se va generando.
- **Estabilización:** En esta fase se llevan a cabo las pruebas sobre la solución, que enfatizan el uso y operación bajo condiciones realistas. El equipo se enfoca en detectar los errores, solucionarlos y preparar la solución para el lanzamiento.
- **Implantación o Despliegue:** En esta fase se decide la tecnología base y sus componentes, estabiliza la instalación y se traspasa el proyecto al cliente.
- **Soporte:** El objetivo de esta fase es refinar la versión del producto para darle continuidad con los nuevos cambios que se deseen. (7)

1.3.4. *Framework .Net* 3.5.

Un *framework* es una estructura de soporte definida en la cual otro proyecto de *software* puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre *software*, para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Proporciona además una estructura al código fuente, donde el desarrollador puede crear código legible y más fácil de mantener, lo cual permite la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

El *framework* de .Net 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. El framework .NET v3.5 engloba todas las características de las versiones anteriores y otras nuevas como:

- Integración total de *LINQ* [6]y del reconocimiento de los datos.
- Nueva compatibilidad con el protocolo *web* para generar servicios *WCF*[7].

[6] LINQ: *Language Integrated Query,* por sus siglas en inglés, significa lenguaje con consulta integrada en español.

- Aplicaciones web interactivas mediante la integración con AJAX[8].
- Compatibilidad absoluta con las herramientas de *Visual Studio* 2008 para *WF*[9], *WCF* y *WPF*[10].

Ventajas que proporciona .*Net Framework*:

- **Código administrado:** El *CLR*[11] realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.
- **Interoperabilidad multilenguaje:** El código puede ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio.
- **Compilación *just-in-time*:** El compilador *JIT* incluido en el *framework* compila el código intermedio generando el código máquina propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.
- ***Garbage collector*:** El *CLR* proporciona un sistema automático de administración de memoria denominado recolector de basura (*garbage collector*). El *CLR* detecta cuándo el programa deja de utilizar la memoria y la libera automáticamente. De esta forma, el programador no tiene que liberar la memoria de forma explícita aunque también es posible hacerlo manualmente.
- **Seguridad de acceso al código:** Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente de la web sin tener que preocuparse si esto va a afectar el sistema.
- **Despliegue:** Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El *framework* realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones. (8)

---

[7] WCF: *Windows Communication Foundation*, por sus siglas en inglés.

[8] AJAX: *Asynchronous JavaScript And XML,* por sus siglas en inglés.

[9] WF: *Windows Workflow Foundation,* por sus siglas en inglés.

[10] WPF: *Windows Presentation Foundation*, por sus siglas en inglés.

[11] CLR: *Common Language Runtime,* por sus siglas en inglés, significa lenguaje común de tiempo de ejecución en español.

1.3.5. *ADO.NET Entity Framework.*

Entity Framework es un conjunto de tecnologías de *ADO.NET*[12] que permiten el desarrollo de aplicaciones de *software* orientadas a datos y estos pueden abarcar varios sistemas de almacenamiento, cada uno con sus propios protocolos. Está diseñado para permitir a los programadores crear aplicaciones de acceso a datos programando con un modelo de la aplicación conceptual en lugar de programar directamente con un esquema de almacenamiento relacional. El objetivo es reducir la cantidad de código y mantenimiento que se necesita para las aplicaciones orientadas a datos.(9) Las aplicaciones de *Entity Framework* ofrecen las siguientes ventajas:

- Las aplicaciones están libres de dependencias de codificación rígida de un motor de datos o de un esquema de almacenamiento.
- Las asignaciones entre el modelo conceptual y el esquema específico de almacenamiento pueden cambiar sin tener que cambiar el código de la aplicación.
- Se pueden asignar varios modelos conceptuales a un único esquema de almacenamiento.
- La compatibilidad con *LINQ* proporciona validación de la sintaxis en el momento de la compilación para consultas en un modelo conceptual.

Al ser un componente de *.NET Framework*, sus aplicaciones se pueden ejecutar en cualquier equipo en que esté instalado *.NET Framework* 3.5 *Service Pack* 1 (SP1).

1.3.6. *Windows Workflow Foundation (WF).*

Es un *framework* para el desarrollo de aplicaciones *web* que permite guiar los procesos de negocio con *Windows Workflow Foundation*. Su principal objetivo es proporcionar un componente que permita gestionar las instancias de *workflow*. Además, encapsula un conjunto de actividades y servicios que le dan mayor dinamismo al desarrollo de sistemas. (10)

WF permite a los usuarios crear flujos de trabajo en sus aplicaciones. Un flujo de trabajo no es más que un conjunto de actividades que están organizadas jerárquicamente en una estructura de árbol. Las actividades proporcionan las funcionalidades para el flujo de control, las condiciones, el control de eventos, la administración de estados y la comunicación con aplicaciones y servicios. Al diseñar flujos de trabajo, se pueden utilizar actividades proporcionadas por *WF* o crear actividades personalizadas.

Workflow automatiza la secuencia de acciones, actividades o tareas utilizadas para la ejecución de un proceso, incluyendo el seguimiento del estado de cada una de las etapas y el aporte de las herramientas necesarias para gestionarlo.

[12] *ADO.NET:* Responde a las siglas de *Microsoft ActiveX Data Objects* de la plataforma *.NET*.

1.3.7. *Bison Framework.*

Es un *framework* para dirigir los procesos de negocio con *Windows Workflow Fundation*. Su principal objetivo es proporcionar un componente que permita gestionar las instancias de *workflow*, además de encapsular un conjunto de actividades y servicios que dan dinamismo al desarrollo de sistemas centrado en la orquestación de procesos de negocio con *WF*, específicamente para un ambiente *web*. *Bison Framework* está compuesto por:

- ***Activities*:** En este paquete se definen todas las actividades que por su comportamiento son necesarias en la arquitectura base.
- ***Runtime Services*:** En este paquete se encuentran todas las interfaces e implementaciones de servicios pertenecientes a la arquitectura base que propone el uso del *framework*. Estos servicios representan un factor clave dentro de la aplicación ya que son los que permiten la interacción con las instancias de procesos creadas.
- ***Hosting*:** En este paquete se encuentran las clases necesarias que permiten la gestión de las instancias de *WF*, además de contener otras clases útiles para el acceso a los servicios del *framework*. Contiene además una clase especializada en la creación de las instancias de los servicios a utilizar dentro del *Runtime*.

1.3.8. *ASP.NET.*

ASP.NET, una parte de la plataforma *.NET* de *Microsoft*, es una estructura de programación revolucionaria que permite el desarrollo de aplicaciones *web* dirigidas a corporaciones. Constituye una forma rápida y escalable de desarrollar, implementar y ejecutar aplicaciones en cualquier navegador o dispositivo.

Además, dispone de funciones de almacenamiento en caché y de administración de estado que potencian el rendimiento de aplicaciones al mismo tiempo que ofrece un alto rendimiento. Las aplicaciones *ASP.NET* son muy fiables y seguras, lo que contribuye a que los usuarios tengan una mayor confianza a la hora de utilizarlas.(11) Esta plataforma permite dotar de funciones adicionales a una aplicación *web*, escribir una menor cantidad de código y seleccionar cualquiera de los lenguajes de programación *.NET* disponibles, como C#, lenguaje utilizado para la implementación de la solución propuesta.

1.3.9. Lenguaje de programación *CSharp*.

CSharp o *C#,* como también se le conoce, es un lenguaje de programación orientado a objetos desarrollado y estandarizado por la compañía *Microsoft* como parte de la plataforma *.NET* y aprobado

como un estándar por la *ECMA*[13] e *ISO*[14]. Su primera versión fue publicada en 2001; su desarrollo fue guiado por Anders Hejlsberg[15] y está basado en los lenguajes *Delphi*, *Visual Basic*, *C++* y *Java*.

Este lenguaje cuenta con algunas características que hacen fácil y fiable su adopción, estas son:

- **Sencillez:** El código escrito en *C#* es auto-contenido. No incluye elementos poco útiles de lenguajes como *C++* tales como macros, herencia múltiple. El tamaño de los tipos de datos básicos es fijo e independiente del compilador.
- **Modernidad:** *C#* incorpora en el propio lenguaje elementos que han demostrado ser importantes como: la inclusión de una instrucción *foreach* y la inclusión de un tipo de dato básico *string*.
- **Orientación a objetos:** Soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo.
- **Orientación a componentes:** *C#* permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).
- **Seguridad:** Incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que evita que se produzcan errores difíciles de detectar por acceso a memoria.

1.3.10. Lenguaje Extensible de Marcado (*XML*).

XML, más que un lenguaje de marcado, es un meta-lenguaje que permite definir lenguajes de marcado adecuados a usos determinados. Fue desarrollado por el *W3 Consortium* para permitir la descripción de información contenida en la *web* a través de estándares y formatos comunes, de manera que los usuarios de Internet y programas puedan buscar, comparar y compartir información en la red. El formato de XML es muy parecido al del HTML aunque no es una extensión ni un componente de éste. Es un subconjunto de *SGML*[16] simplificado y adaptado a *Internet*. *XML* es un estándar internacionalmente reconocido que no pertenece a ninguna compañía y su uso es libre; permite el uso efectivo de *Internet* en diferentes alfabetos y en diferente *hardware*.

[13] ECMA: *European Computer Manufacturers Association,* por sus siglas en inglés.

[14] ISO: *International Organization for Standardization,* por sus siglas en inglés.

[15] Nacido en Copenhague, Dinamarca en el año 1961. Destacado ingeniero de software, actualmente trabaja para *Microsoft*, donde es el arquitecto del lenguaje de programación C#.

[16] SGML: *Standard Generalized Markup Language,* por sus siglas en inglés.

1.3.11. Entorno de Desarrollo Integrado (*IDE*).

Un *IDE* es un entorno de programación que está compuesto por un conjunto de herramientas, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (*GUI*). Los *IDE* proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación.

1.3.11.1. *Visual Studio Team System* 2008.

Visual Studio 2005 Team System es una plataforma para herramientas de ciclo de vida de desarrollo de software extensibles, integradas y productivas que ayudan a los equipos de desarrollo de software mejorando las comunicaciones y la colaboración durante todo el proceso de desarrollo.(12) *Microsoft Visual Studio* es el IDE de programación que se utilizará en el desarrollo de la aplicación, soporta un diverso grupo de lenguajes de programación tales como *Visual C++.NET, Visual J#.NET, Visual Basic.NET y Visual C#.NET.*

Entre los principales beneficios que aporta el uso del este IDE para la aplicación que se desea desarrollar se encuentra que provee amplias facilidades como la inclusión de un editor de código que soporta resaltado de sintaxis, utilizando *IntelliSense* para el autocompletamiento de código, 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. (13)

1.3.12. Servidor *Web*.

Un servidor *web* es un programa que se ejecuta continuamente en una computadora, manteniéndose a la espera de peticiones que le hará un cliente y dependiendo de ellas buscará una página *web* o ejecutará un programa en el servidor. El servidor responde al cliente enviándole el código *HTML* de la página, cuando éste lo recibe, lo interpreta y lo muestra en pantalla. El servidor permite alojar los sitios a los cuales se necesita acceder.

Permiten el intercambio de datos y funcionalidades entre aplicaciones sobre una red y la intercomunicación entre sistemas de cualquier plataforma. Está soportado por algunos estándares que garantizan la interoperabilidad de los servicios.

1.3.12.1. *Internet Information Server* (*IIS*).

Constituyen un conjunto de servicios para los ordenadores que funcionan con *Windows*. *IIS* convierte un ordenador en un servidor de *Internet* o *Intranet*, es decir, que en las computadoras que tienen este servicio instalado se pueden publicar páginas *web* tanto local como remotamente. Los servicios de *IIS* proporcionan las herramientas y funciones necesarias para administrar de forma sencilla un servidor *web* seguro.

1.3.13. Embarcadero ER/Studio.

ER/Studio es una herramienta *CASE* para el modelado y diseño de bases de datos que brinda productividad en el diseño, generación, y mantenimiento de aplicaciones. Ofrece capacidades de sincronización bidireccional de los diseños físicos y lógicos, construcción automática de base de datos, documentación y fácil creación de reportes. (14) Soporta el proceso de diseño iterativo inherente en el ciclo de vida de la aplicación. Las capacidades de diseño que contiene, ayudan a crear un diseño lógico que puede transformarse en múltiples diseños físicos.

1.3.14. Sistema Gestor de Base de Datos (*SGBD*).

Un SGBD es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. Constituyen un tipo de *software* muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Tiene como principales objetivos manejar un conjunto de datos que constituyen información relevante para una organización, evitar la redundancia de los datos, mejorar los mecanismos de seguridad de los mismos y su privacidad e integridad realizando las validaciones necesarias para mejorar la eficacia en el acceso a datos.

1.3.14.1. *Oracle Database 11g*.

Oracle es un Sistema de Gestión de Base de Datos Relacional (*RSGBD*), su primera versión, elaborada por la compañía *Oracle Corporation,* aparece en el mercado en el año 1979. En la actualidad el sistema se encuentra en la versión 11g, liberada bajo una licencia privativa en el año 2007. Entre sus características principales, resaltan las siguientes:

- **Soporte de transacciones:** Capacidad de ejecutar un conjunto de órdenes formando una unidad de trabajo, es decir, en forma indivisible o atómica.
- **Escalabilidad:** Habilidad para manejar el continuo crecimiento del trabajo y estar preparado para hacerse más grande sin afectar su funcionamiento.
- **Estabilidad:** Posee una reducida tasa de fallos en su funcionamiento.
- **Soporte multiplataforma:** Compatibilidad con varios *sistemas operativos* explotadas actualmente a nivel mundial (*Windows, Linux, Unix*).
- **Soporte de *Triggers* (Disparadores):** Es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (*INSERT*), actualización (*UPDATE*) o borrado (*DELETE*).

- **Soporte de procedimientos almacenados:** Programa (o procedimiento) el cual es almacenado físicamente en una base de datos. Al ser ejecutados directamente en el motor de base de datos, posee acceso directo a los datos que necesita manipular y solo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes.
- **Soporte de Índices:** Estructura de datos que mejora la velocidad de las operaciones, permitiendo un rápido acceso a los registros de una tabla. (15)

El software propuesto utilizará *Oracle Database* 11g para gestionar la persistencia de la información aprovechando el alto nivel de confiabilidad, escalabilidad y seguridad que aporta este gestor de base de datos.

1.4. Conclusiones.

Con el objetivo de lograr un módulo que posea un alto grado de eficiencia en cuanto al proceso de administración de sistemas de gestión de identidades de usuarios se analizaron las características fundamentales de los productos líderes en este campo. Se examinaron los elementos esenciales de las herramientas a utilizar para la concepción del módulo, reconociendo los beneficios que posibilita su uso en el desarrollo de la aplicación. Es importante considerar que la lucha por la supremacía del mercado entre las compañías líderes del sector de la gestión de identidades provoca que se liberen versiones de aplicaciones que gestionan este proceso con mayor frecuencia, calidad y fiabilidad; utilizando para ello lenguajes, herramientas y metodologías de avanzada.

Basándose en lo anteriormente planteado la metodología que guiará el proceso de desarrollo del módulo será *MSF* para el Desarrollo de Software Ágil, propiciando un marco de trabajo flexible adecuándose a las características del proyecto. La herramienta de modelado será *Altova UModel* 2009, provocando su uso importantes beneficios como la posibilidad de modelación de la aplicación y la generación de código fuente en el lenguaje *de programación a utilizar (C#)*. Para el desarrollo de la solución se hará uso de la plataforma .NET lo que posibilita la creación de una aplicación robusta y escalable. Todo esto gestionado mediante el IDE de desarrollo *Visual Studio Team System* 2008 que garantiza la comunicación con los demás miembros del equipo y la generación de código de alta eficiencia. Para gestionar la persistencia de la información resultante de los procesos de negocio realizados en la organización, se utilizará *Oracle Database* 11g. Estas herramientas tienen en común que son propietarias y muy utilizadas a nivel mundial. Al concluir este capítulo se cuenta con una base teórica que fundamenta la realización de la aplicación.

Capítulo 2 Características del sistema

Para comenzar el desarrollo de un sistema informático se hace necesario aclarar todos los conceptos y sus relaciones utilizando un lenguaje que sea comprendido por todos los involucrados. Se identifican los objetivos de la aplicación, los cuales constituyen el pilar fundamental para el posterior desarrollo de la misma.

Las fases de *MSF* para el Desarrollo de *Software* Ágil a las que se les prestará atención en el presente capítulo son Visión y Planificación. Al respecto la metodología plantea que se debe obtener una visión clara del lo que se desea desarrollar y propone realizar una planificación que guíe al equipo de trabajo hacia la construcción exitosa del sistema. Para ello se realizan las actividades contenidas dentro de los flujos de trabajos que sugiere la metodología para dar comienzo al desarrollo de un proyecto. Estos flujos de trabajos son: capturar la visión del proyecto, crear los escenarios y crear los requerimientos de calidad de servicios. Las fases Desarrollo y Estabilización serán analizadas en posteriores capítulos.

2.1. Fase Visión.

Es indispensable tener presente la visión y alcance del proyecto para mantenerlo enfocado a las necesidades que va a resolver. Esta visión puede cambiar como resultado de la interacción con los clientes, de ser así, es necesario reajustar el proyecto a la nueva visión, con la que se comienza a entender cuál será la relación de los usuarios con el producto.

2.1.1. Descripción del sistema.

Los procesos descritos en este epígrafe se basan en las investigaciones previas desarrolladas por los autores debido a que el sistema surge por la necesidad del Centro de Identificación y Seguridad Digital, al cual pertenecen. Por tanto, no cuenta con procesos de negocio definidos.

La **Aplicación de Aprovisionamiento** (*Provisioning*) del Sistema de Administración de Identidades, que actualmente está en desarrollo, se encargará de gestionar las cuentas de los usuarios en las distintas aplicaciones o servidores con los que tendrá conexión, contará con varios módulos para su funcionamiento. Ellos son: Portal del Usuario, **Administración**, Gestor de Conectores, Reportes, Motor de Tareas, Alertas y Notificaciones.

El desarrollo del módulo **Administración** surge por la necesidad de gestionar el ciclo de vida de las cuentas de usuarios creadas previamente a través de las solicitudes.

2.1.2. Propuesta de solución.

Teniendo en cuenta la problemática a resolver y a partir del análisis de las tendencias actuales se propone el desarrollo de una aplicación web que englobará todo lo referente a la gestión de las cuentas en la Aplicación de Aprovisionamiento de Usuarios del Sistema de Administración de Identidades y se determinaron las principales funcionalidades y características que poseerá.

Se diseñará el módulo para que funcione con la menor cantidad de privilegios posibles, estableciendo para ello que un usuario sólo debe tener los privilegios que necesita para completar las tareas asignadas. Teniendo en cuenta lo anteriormente planteado se asegurará que el módulo Administración estará completamente disponible sólo a los usuarios de tipo Administrador. Los usuarios de tipo Aprobador tendrán acceso sólo al sub-módulo Gestión de Solicitudes. (Ver flujo de funcionamiento en el Anexo 6)

2.1.3. Flujo de funcionamiento.

Existirán tres tipos de usuario en el sistema: estándar, aprobador y administrador.

El proceso de gestión de cuentas de usuarios en el Sistema de Administración de Identidades está controlado por la Aplicación de Aprovisionamiento y consta de varios pasos. Cuando un usuario desea crearse una cuenta en el sistema llena un formulario con los datos necesarios y se crea una solicitud de creación de cuenta. Las solicitudes son enviadas al módulo Motor de Tareas donde se rigen por diferentes flujos de trabajo que definen los niveles de aprobación de la misma (Ver *Utilización de Workflow e integración con el módulo Motor de Tareas* en el Capítulo 3, Epígrafe 3.6).

Los aprobadores, una vez que acceden al sistema, poseen acceso a una lista de solicitudes pendientes que esperan ser procesadas (aprobadas, revisadas o revocadas). Solo los aprobadores son los encargados de realizar las aprobaciones de todas las solicitudes. En caso que se apruebe la solicitud, el aprobador debe especificar los datos restantes correspondientes al usuario solicitante y una descripción de la acción realizada, la cuenta creada es persistida en la base de datos del sistema y en los recursos de la organización. Si la solicitud es revocada, esta se elimina del listado de solicitudes pendientes y pasa al historial de acciones realizadas. Además, se actualiza la información que se le brinda al usuario sobre el estado de su solicitud.

Cuando un administrador accede al sistema se le muestran todas las funcionalidades disponibles. Los administradores pueden gestionar las cuentas de usuario creadas previamente a través de las

solicitudes, realizando acciones como crear, habilitar, eliminar, bloquear y suspender cuentas. Estas acciones son consideradas solicitudes de modificación, eliminación o de creación de cuentas, por tanto se les aplica el proceso de aprobación anteriormente descrito.

Los administradores del sistema tienen la posibilidad de gestionar las políticas de contraseñas, mediante acciones de creación, modificación y eliminación de estas políticas. Se definirán políticas de contraseña identificadas por nombre, dividiéndose en dos tipos: por defecto del sistema y personalizada, esta última es la definida por el administrador. Las políticas de contraseñas una vez aplicadas sobre los usuarios no podrán eliminarse.

A través de las configuraciones generales que se le aplican al sistema, los administradores pueden establecer la cantidad de preguntas de seguridad, administrar la configuración de las cuentas de usuario y administrar la asignación de los aprobadores a los estados por los que transcurren las solicitudes. Así queda definido el flujo de funcionamiento de la Aplicación de Aprovisionamiento de Usuarios del Sistema de Administración de Identidades. (Ver flujo de funcionamiento en el Anexo 6)

2.1.4. Vista conceptual del sistema.

Como quedó expuesto en el epígrafe 2.1.1 los procesos de negocio no están bien estructurados por lo que se presenta una vista conceptual de forma general de la aplicación para capturar los objetos más importantes, los eventos que suceden en el entorno donde estará el sistema y la relación que existente entre ellos.

Descripción.

- ***Administración:*** Representa el módulo para la gestión de las cuentas, las solicitudes, las políticas de contraseñas y las configuraciones generales.
- **Solicitud:** Representa una petición realizada por un usuario en el sistema, debe ser aprobada o revocada por el aprobador.
- ***Provisioning*:** Representa la Aplicación de Aprovisionamiento del Sistema de Administración de Identidades.
- **Usuario:** Representa los usuarios que interactúan con el sistema.
- **Cuenta:** Representa las cuentas de usuario creadas en el sistema.
- **PolíticasContraseñas:** Representa las políticas de contraseñas definidas en el sistema.
- **GestionarCuentas**: Gestiona todas las acciones que se realizan sobre las cuentas.
- **GestionarSolicitudes**: Gestiona todas las acciones que se realizan sobre las solicitudes.

- **GestionarPolíticasContraseñas**: Gestiona todas las acciones que se realizan sobre las políticas de contraseñas.
- **ConfiguracionesGenerales**: Gestiona todas las acciones que se realizan sobre las configuraciones generales del sistema.

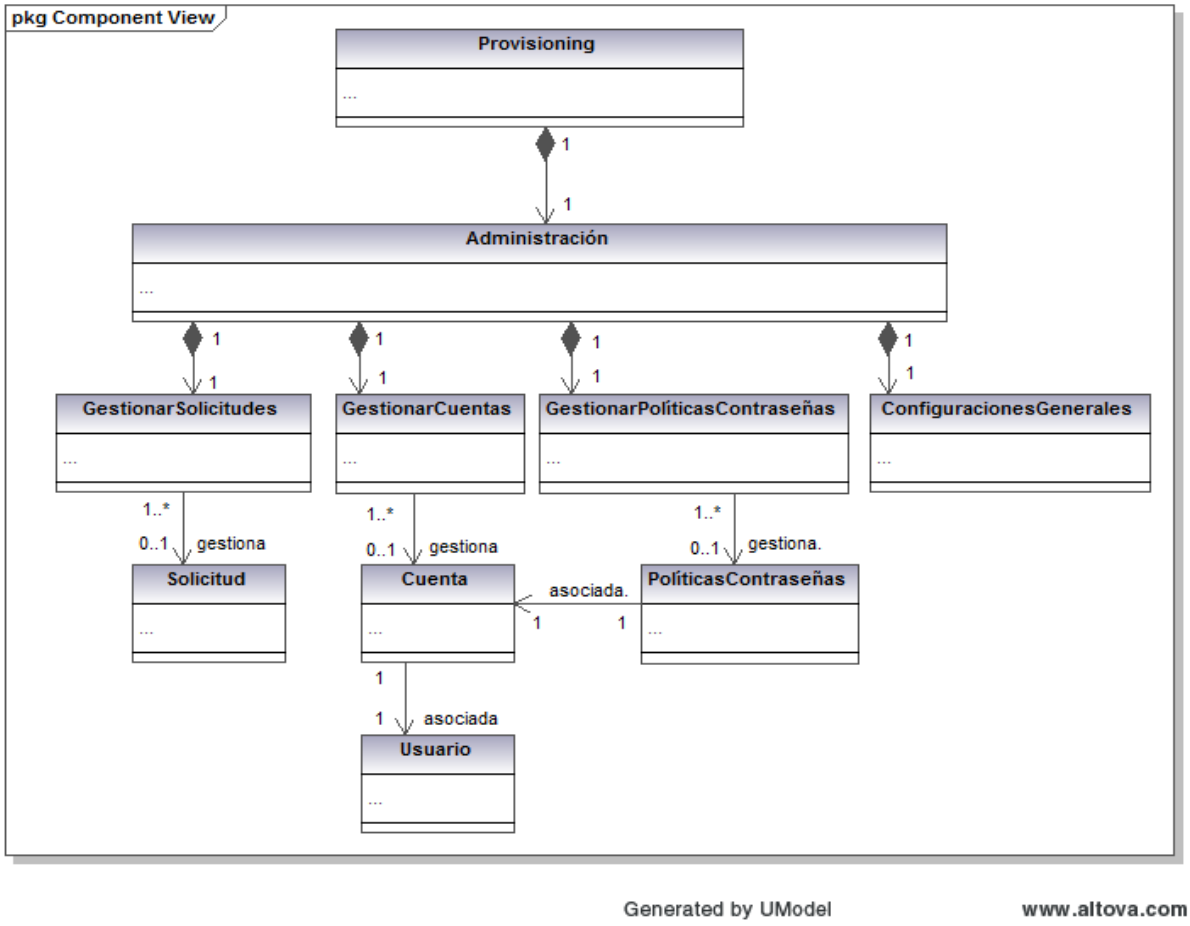


Figura 1: Modelo conceptual del Sistema.

2.1.5. Definición de las personas.

En esta actividad se definirán las personas o sistemas externos que interactúan con el sistema de aprovisionamiento.

Personas	Descripción
Administrador	Realizará operaciones de administración y tendrá todos los privilegios que existan en la aplicación de aprovisionamiento.
Aprobador	Podrá realizar la tarea de revocar o aprobar una solicitud, operación sobre los ámbitos del sistema que le sean asignados.

Usuario	Podrá interactuar con el sistema haciendo uso de los permisos que le sean asignados.

Tabla 1: Descripción de las personas.

2.2. Fase Planificación.

En esta fase el equipo prepara las especificaciones funcionales, se realiza el proceso de diseño de la solución y los planes de trabajo, estimaciones de costo y cronogramas de los diferentes entregables del proyecto. Los principales artefactos son la lista de los escenarios y los requerimientos de calidad de servicios, los cuales son utilizados para especificar los requisitos del *software* que sirven de guía para todo el proceso de desarrollo.

2.2.1. Lista de escenarios del sistema.

Los escenarios se utilizan para definir los servicios que proporcionará el *software* para el usuario y representa un camino único de la interacción del usuario a través del sistema que está construyendo. Dentro de las actividades en el ciclo de vida de un escenario (Ver Anexo 4) se encuentra listar los escenarios, para el desarrollo del módulo propuesto se identificaron los siguientes:

- **ES1. Gestionar cuentas.**
  - Habilitar cuenta.
  - Suspender cuenta.
  - Bloquear cuenta.
  - Eliminar cuenta.
  - Editar cuenta.
  - Listar cuentas.
- **ES2. Gestionar políticas de contraseñas.**
  - Crear política de contraseña.
  - Editar política de contraseña.
  - Listar políticas de contraseñas.
- **ES3. Gestionar solicitudes.**
  - Aprobar solicitud.
  - Revocar solicitud.
  - Editar solicitud.
  - Revisar solicitud.
  - Listar solicitudes.
- **ES4. Configuraciones generales.**
  - Mostrar historial de administración.

- Establecer preguntas de seguridad.
- Administrar cuentas de usuarios.
- Asignar aprobadores.

2.2.2. Priorizar la lista de escenarios.

Los escenarios se priorizan en dependencia de la importancia que tienen para los usuarios y para la validez de la aplicación, o sea cuán necesario es incluir el escenario. El proceso de priorizar la lista de escenarios trae consigo identificar los escenarios más importantes a la hora de implementar para que dicha implementación sea en las primeras iteraciones. Calificar al escenario con 5 puntos equivale a tener una prioridad "Alta", con 4 puntos "Media" y con 3 puntos "Baja".

No.	Escenarios	Prioridad
1	Gestionar solicitudes	5
2	Gestionar cuentas	4
3	Gestionar políticas de contraseñas	4
4	Configuraciones generales	3

Tabla 2: Priorizar la lista de escenarios.

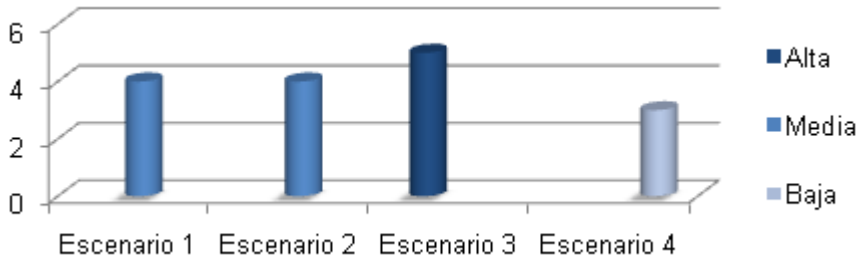


Figura 2: Gráfica evaluación de los escenarios (Escenarios-Prioridad).

2.2.3. Requerimientos de calidad de servicio.

Representan los requisitos no funcionales en las categorías de plataforma, rendimiento, seguridad y apariencia. A continuación se listan los requisitos de calidad de servicio que se identificaron para el desarrollo del módulo propuesto.

- Plataforma que soporta.
  - La aplicación debe estar disponible las 24 horas del día desde cualquier sistema operativo que acceda el cliente.
  - El sistema debe utilizar el *.NET framework* en su versión 3.5.
  - La base de datos que debe usarse será *Oracle 11g*.

- El servidor central debe tener sistema operativo *Windows Server 2000* o superior.

- Rendimiento.
  - Mostrar la información con el criterio de búsqueda seleccionado por el usuario en un tiempo máximo de 10 segundos.
- Seguridad.
  - Debe lograr la autenticación y autorización una vez integrado a los módulos que darán solución a estos procesos.
- Apariencia o interfaz externa.
  - Los campos obligatorios de cada ventana deben estar debidamente identificados y resaltados.
  - Permitir la paginación en los listados.
  - Contar con accesos directos a las funcionalidades más utilizadas.

2.2.4. Plan de iteraciones.

Se hace necesario estimar de forma adecuada el tiempo que les tomará a los programadores implementar cada uno de los escenarios que pertenecen a la aplicación, esto no es más que otra de las actividades de la fase de planificación. Dependiendo de la prioridad de los escenarios se decide cuáles de ellos se implementarán en las primeras iteraciones del ciclo.

Iteración #1: Se propone codificar las funcionalidades del módulo Administración, específicamente los escenarios que presentan una alta prioridad.

Iteración #2: Se propone codificar los escenarios que presentan una prioridad media.

Iteración #3: Última iteración, se implementarán los escenarios restantes del módulo Administración.

No.	Nombre del Escenario	Prioridad	Riesgo	Esfuerzo (días)	Iteración
1	Gestionar solicitudes	Alta	Alta	35	1
2	Gestionar cuentas	Media	Alta	35	2
3	Gestionar políticas de contraseñas	Media	Alta	10	2
4	Configuraciones generales	Baja	Alta	10	3

Tabla 3: Planificación de los escenarios.

Los escenarios son programados para el desarrollo y pruebas iniciales en una iteración específica. El plan de iteraciones se realiza mediante el diagrama de *Gantt*[17] (Ver Anexo 5) y debe completarse antes de que la iteración comience.

2.3. Descripción de los escenarios.

La arquitectura del sistema está condicionada por los escenarios. Estos proporcionan un medio para que los desarrolladores, usuarios finales y trabajadores lleguen a una comprensión común del sistema propuesto. Son utilizados para la validación de la arquitectura a lo largo del desarrollo. A continuación se presenta una breve descripción de los escenarios del sistema.

Título:	**Gestionar cuentas**
Área:	Módulo Administración
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Persona:	Administrador
Descripción:	Se encarga de gestionar las cuentas de usuario existentes en el sistema. Se tienen las funcionalidades para habilitar, suspender, bloquear, eliminar, modificar y listar las cuentas.
Historial:	1. Inicio de descripción del escenario
Prioridad:	Media
Identificador:	ES1
Orden de Magnitud:	2

Tabla 4: Descripción ES1 "Gestionar cuentas".

Título:	**Gestionar políticas de contraseñas**
Área:	Módulo Administración
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Persona:	Administrador
Descripción:	Se encarga de gestionar las políticas de contraseñas del sistema, permite crear, editar y listar las políticas existentes.
Historial:	1. Inicio de descripción del escenario

[17] Popular herramienta gráfica cuyo objetivo es mostrar el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. Desarrollada por el ingeniero industrial mecánico estadounidense Henry Laurence Gantt entre 1910 y 1915.

Prioridad:	Media
Identificador:	ES2
Orden de Magnitud:	1

Tabla 5: Descripción ES2 "Gestionar políticas de contraseñas".

Título:	**Gestionar solicitudes**
Área:	Módulo Administración
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción
Persona:	Administrador, Aprobador
Descripción:	Se encarga de gestionar el proceso de concesión y revocación de las solicitudes. Permite aprobar, revocar, revisar y listar las solicitudes.
Historial:	1. Inicio de descripción del escenario
Prioridad:	Alta
Identificador:	ES3
Orden de Magnitud:	1

Tabla 6: Descripción ES3 "Gestionar solicitudes".

Título:	**Configuraciones generales.**
Área:	Módulo Administración
Iteración:	3
Estado:	Activo
Razón de estado:	Proceso de construcción
Persona:	Administrador
Descripción:	Se encarga de establecer la cantidad de preguntas de seguridad que se les mostrará a los usuarios, de configurar los elementos bloqueo, suspensión y eliminación automática de las cuentas y asignar aprobadores a cada estado de las solicitudes.
Historial:	1. Inicio de descripción del escenario
Prioridad:	Baja
Identificador:	ES4
Orden de Magnitud:	1

Tabla 7: Descripción ES4 "Configuraciones generales".

2.3.1. Especificación de tareas por escenarios.

A continuación se describirán las tareas asociadas a cada escenario del sistema. El escenario **Administrar cuentas** se dividió en las tareas: Habilitar cuenta, Suspender cuenta, Bloquear cuenta, Eliminar cuenta, Editar cuenta y Listar cuentas.

Título:	**Habilitar cuenta**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Armando León Martínez
Descripción:	El escenario comienza a partir del listado de las cuentas existentes a través de la interfaz *Gestionar Cuentas*. El administrador selecciona la cuenta deseada y accede a la opción *Habilitar*. El escenario culmina cuando el módulo Administración envía la solicitud de habilitación de cuenta al módulo *Motor de Tareas*. Cuando una solicitud de habilitación es aprobada la cuenta pasa al estado "activa".
Historial:	1. Inicio de descripción del escenario
Prioridad:	Media
Identificador:	ES1.1
Tiempo de desarrollo:	80

Tabla 8: Descripción del ES1.1 "Habilitar cuenta".

Título:	**Suspender cuenta**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Armando León Martínez
Descripción:	El escenario comienza a partir del listado de las cuentas existentes a través de la interfaz *Gestionar Cuentas*. El administrador selecciona la cuenta deseada y accede a la opción *Suspender*. El escenario culmina cuando el módulo Administración envía la solicitud de suspensión de cuenta al módulo *Motor de Tareas*. Cuando una solicitud de suspensión es aprobada la cuenta pasa al estado "suspendida".
Historial:	1. Inicio de descripción del escenario
Prioridad:	Media
Identificador:	ES1.2
Tiempo de desarrollo:	80

Tabla 9: Descripción del ES1.2 "Suspender cuenta".

Título:	**Bloquear cuenta**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Armando León Martínez

Descripción:	El escenario comienza a partir del listado de las cuentas existentes a través de la interfaz *Gestionar Cuentas*. El administrador selecciona la cuenta deseada y accede a la opción *Bloquear*. El escenario culmina cuando el módulo Administración envía la solicitud de bloqueo de cuenta al módulo *Motor de Tareas*. Cuando una solicitud de bloqueo es aprobada la cuenta pasa al estado "bloqueada".
Historial:	1. Inicio de descripción del escenario
Prioridad:	Media
Identificador:	ES1.3
Tiempo de desarrollo:	80

Tabla 10: Descripción del ES1.3 "Bloquear cuenta".

Título:	**Eliminar cuenta**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Viviana Muñiz López
Descripción:	El escenario comienza a partir del listado de las cuentas existentes a través de la interfaz *Gestionar Cuentas*. El administrador selecciona la cuenta deseada y accede a la opción *Eliminar*. El escenario culmina cuando el módulo Administración envía la solicitud de eliminación de cuenta al módulo *Motor de Tareas*. Cuando una solicitud de eliminación es aprobada la cuenta pasa al estado "eliminada".
Historial:	1. Inicio de descripción del escenario
Prioridad:	Media
Identificador:	ES1.4
Tiempo de desarrollo:	80

Tabla 11: Descripción del ES1.4 "Eliminar cuenta".

Título:	**Editar cuenta**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Viviana Muñiz López
Descripción:	El escenario comienza a partir del listado de las cuentas existentes a través de la interfaz *Gestionar Cuentas*. El administrador selecciona la cuenta deseada y accede a la opción *Editar*. El sistema muestra un formulario para la inserción de los datos a actualizar. El administrador llena los campos solicitados. El escenario culmina cuando el módulo Administración envía la solicitud de modificación de cuenta al módulo *Motor de Tareas*. Cuando una solicitud de modificación es aprobada la cuenta mantiene su mismo estado y los datos se actualizan en la BD.
Historial:	• Inicio de descripción del escenario
Prioridad:	Media

Identificador:	ES1.5
Tiempo de desarrollo:	120

Tabla 12: Descripción del ES1.5 "Editar cuenta".

Título:	**Listar cuentas**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Viviana Muñiz López
Descripción:	Los administradores del sistema tienen acceso al listado de las cuentas a través de la interfaz *Gestionar Cuentas*.
Historial:	1. Inicio de descripción del escenario
Prioridad:	Media
Identificador:	ES1.6
Tiempo de desarrollo:	40

Tabla 13: Descripción del ES1.6 "Listar cuentas".

El escenario del sistema **Gestionar políticas de contraseñas** se dividió en las tareas: Crear política de contraseña, Editar política de contraseña y Listar políticas de contraseñas.

Título:	**Crear política de contraseña**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Viviana Muñiz López
Descripción:	El escenario comienza cuando el administrador selecciona la opción *Crear Política* a través de la interfaz *Gestionar Políticas de Contraseña*. El sistema muestra un formulario para la inserción de los nuevos datos. El administrador llena los campos solicitados. El escenario culmina cuando el módulo Administración crea la nueva política de contraseña en la base de datos del sistema. El nombre de la política deberá ser único.
Historial:	1. Inicio de descripción del escenario
Prioridad:	Media
Identificador:	ES2.1
Tiempo de desarrollo:	40

Tabla 14: Descripción del ES2.1 "Crear política de contraseña".

Título:	**Editar política de contraseña**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Viviana Muñiz López

Descripción:	El escenario comienza cuando el administrador selecciona la opción *Editar Política* a través de la interfaz *Gestionar Políticas de Contraseña*. El sistema muestra un formulario para la inserción de los datos actualizados. El administrador llena los campos solicitados. El escenario culmina cuando el módulo Administración actualiza la política de contraseña en la base de datos del sistema. El nombre de la política deberá ser el mismo.
Historial:	1. Inicio de descripción del escenario
Prioridad:	Media
Identificador:	ES2.2
Tiempo de desarrollo:	24

Tabla 15: Descripción del ES2.2 "Editar política de contraseña".

Título:	**Listar políticas de contraseñas**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Viviana Muñiz López
Descripción:	El escenario comienza cuando el administrador selecciona la opción *Listar Políticas* a través de la interfaz *Gestionar Políticas de Contraseña*. El sistema muestra un listado de las políticas existentes. El escenario culmina cuando el sistema muestra el listado de las políticas existentes.
Historial:	1. Inicio de descripción del escenario
Prioridad:	Media
Identificador:	ES2.3
Tiempo de desarrollo:	16

Tabla 16: Descripción del ES2.3 "Listar políticas de contraseñas".

El escenario del sistema **Administrar solicitudes** se dividió en las tareas: Aprobar solicitud, Revocar solicitud, Editar solicitud, Listar solicitudes y Revisar solicitud.

Título:	**Aprobar solicitud**
Tipo de Tarea:	Desarrollo
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Viviana Muñiz López
Descripción:	El escenario comienza a partir del listado de las solicitudes a través de la interfaz *Gestionar Solicitudes*. El administrador/aprobador selecciona la solicitud deseada y accede a la opción *Detalles*. El sistema muestra un formulario para la modificación opcional de los datos la solicitud. Es obligatorio especificar una descripción de la acción a realizar. A continuación se selecciona la opción *Aprobar Solicitud.* El escenario culmina cuando el módulo Administración aprueba la solicitud deseada.
Historial:	1. Inicio de descripción del escenario
Prioridad:	Alta

Identificador:	ES3.1
Tiempo de desarrollo:	80

Tabla 17: Descripción del ES3.1 "Aprobar solicitud".

Título:	**Revocar solicitud**
Tipo de Tarea:	Desarrollo
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Armando León Martínez
Descripción:	El escenario comienza a partir del listado de las solicitudes a través de la interfaz *Gestionar Solicitudes*. El administrador/aprobador selecciona la solicitud deseada y accede a la opción *Detalles*. El sistema muestra un formulario para la modificación opcional de los datos la solicitud. Es obligatorio especificar una descripción de la acción a realizar. A continuación se selecciona la opción *Revocar Solicitud.* El escenario culmina cuando el módulo Administración rechaza la solicitud deseada.
Historial:	1. Inicio de descripción del escenario
Prioridad:	Alta
Identificador:	ES3.2
Tiempo de desarrollo:	80

Tabla 18: Descripción del ES3.2 "Revocar solicitud".

Título:	**Editar solicitud**
Tipo de Tarea:	Desarrollo
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Armando León Martínez
Descripción:	El escenario comienza a partir del listado de las solicitudes a través de la interfaz *Gestionar Solicitudes*. El administrador/aprobador selecciona la solicitud deseada y accede a la opción *Detalles*. El sistema muestra un formulario para la modificación opcional de los datos la solicitud. Es obligatorio especificar una descripción de la acción a realizar. A continuación se selecciona la opción una de las opciones deseadas sobre la solicitud. El escenario culmina cuando el módulo Administración realiza dicha acción y modifica la solicitud con los nuevos datos insertados.
Historial:	2. Inicio de descripción del escenario
Prioridad:	Alta
Identificador:	ES3.3
Tiempo de desarrollo:	40

Tabla 19: Descripción del ES3.3 "Editar solicitud".

Título:	**Listar solicitudes**
Tipo de Tarea:	Desarrollo
Iteración:	1
Estado:	Activo

Razón de estado:	Proceso de construcción
Responsable:	Armando León Martínez
Descripción:	Los administradores y aprobadores del sistema tienen acceso al listado de las solicitudes a través de la interfaz *Gestionar Solicitudes*.
Historial:	2. Inicio de descripción del escenario
Prioridad:	Alta
Identificador:	ES3.4
Tiempo de desarrollo:	40

Tabla 20: Descripción del ES3.4 "Listar solicitudes".

Título:	**Revisar solicitud**
Tipo de Tarea:	Desarrollo
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Armando León Martínez
Descripción:	El escenario comienza a partir del listado de las solicitudes a través de la interfaz *Gestionar Solicitudes*. El administrador/aprobador selecciona la solicitud deseada y accede a la opción *Detalles*. El sistema muestra un formulario para la modificación opcional de los datos la solicitud. Es obligatorio especificar una descripción de la acción a realizar. A continuación se selecciona la opción *Revisar Solicitud.* El escenario culmina cuando el módulo Administración envía la solicitud al estado anterior para que sea revisada.
Historial:	3. Inicio de descripción del escenario
Prioridad:	Alta
Identificador:	ES3.5
Tiempo de desarrollo:	80

Tabla 21: Descripción del ES3.5 "Revisar solicitud".

El escenario del sistema **Administrar configuraciones generales** se dividió en las tareas**:** Mostrar historial de administración, Establecer preguntas de seguridad, Administrar cuentas de usuarios y Asignar aprobadores.

Título:	**Mostrar historial de administración**
Tipo de Tarea:	Desarrollo
Iteración:	3
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Viviana Muñiz López y Armando León Martínez

Descripción:	Los administradores del sistema tienen acceso al historial de administración a través de la interfaz *Historial de Administración*. Se muestra un listado donde se relacionan todas las cuentas del sistema con los estados por cuales ha transcurrido, además del administrador que realizó la acción y la fecha.
Historial:	1. Inicio de descripción del escenario
Prioridad:	Baja
Identificador:	ES4.1
Tiempo de desarrollo:	40

Tabla 22: Descripción del ES4.1 "Mostar historial de administración".

Título:	**Establecer preguntas de seguridad**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Viviana Muñiz López y Armando León Martínez
Descripción:	El escenario comienza cuando un administrador decide configurar la cantidad de preguntas de seguridad a realizar a los usuarios del sistema a través de la pestaña *Preguntas de Seguridad* de la interfaz *Configuración*. El sistema muestra un formulario para la inserción de la cantidad de preguntas y si requiere pregunta personalizada o no. A continuación se selecciona la opción *Salvar.* El escenario culmina cuando el módulo Administración guarda la nueva configuración.
Historial:	1. Inicio de descripción del escenario
Prioridad:	Baja
Identificador:	ES4.2
Tiempo de desarrollo:	16

Tabla 23: Descripción del ES4.2 "Establecer preguntas de seguridad".

Título:	**Administrar cuentas de usuarios**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Armando León Martínez
Descripción:	El escenario comienza cuando un administrador decide configurar elementos de las cuentas de usuario como el bloqueo, eliminación y suspensión automática de las cuentas del sistema a través de la pestaña *Cuentas de Usuario* de la interfaz *Configuración*. El sistema muestra un formulario para la inserción de los nuevos datos. A continuación se selecciona la opción *Salvar.* El escenario culmina cuando el módulo Administración guarda la nueva configuración.
Historial:	2. Inicio de descripción del escenario

Prioridad:	Baja
Identificador:	ES4.3
Tiempo de desarrollo:	24

Tabla 24: Descripción del ES4.3 "Administrar cuentas de usuarios".

Título:	**Asignar aprobadores**
Tipo de Tarea:	Desarrollo
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción
Responsable:	Viviana Muñiz López
Descripción:	El escenario comienza cuando un administrador decide asignar aprobadores a los distintos estados por los que transcurren las solicitudes a través de la pestaña *Cuentas de Usuario* de la interfaz *Configuración*. El sistema muestra un listado de los aprobadores y un listado de los ya asignados. A continuación se selecciona un aprobador del listado y se accede a la opción *Asignar Permiso* o *Eliminar Permiso.* El escenario culmina cuando el módulo Administración asigna o elimina un permiso de un aprobador sobre un estado.
Historial:	3. Inicio de descripción del escenario
Prioridad:	Baja
Identificador:	ES4.4
Tiempo de desarrollo:	24

Tabla 25: Descripción del ES4.4 "Asignar aprobadores".

2.4. Conclusiones.

En este capítulo se realizó la descripción de las características del sistema a través de la construcción de la vista global correspondiente a los conceptos fundamentales manejados en el mismo, para obtener una visión sobre lo que se desea desarrollar. Se plantearon los escenarios que resolverán las funcionalidades del módulo propuesto, de los cuales se realizó su descripción. Se identificaron las personas que intervendrán en el sistema y se realizó una estimación del tiempo que les tomará a los programadores implementar los escenarios, planificando dicha implementación en 3 iteraciones.

El estudio realizado en este capítulo ha facilitado un entendimiento de la dinámica y estructura de la aplicación, contribuyendo a una mejor comprensión del problema que el módulo de Administración debe resolver ganando claridad en cuanto a su concesión, además se sentaron las bases para las restantes fases del proceso.

Capítulo 3 Desarrollo de la solución propuesta

Durante el presente capítulo se tendrá en cuenta la fase Construcción que facilita *MSF* para el Desarrollo de *Software* Ágil. Se obtendrán los artefactos necesarios para la concepción de la aplicación a desarrollar, ya sea código o documentación. Se propone el desarrollo del sistema utilizando buenas prácticas y patrones de diseño bien definidos con el objetivo de crear un sistema escalable, flexible y confiable.

3.1. Especificación de la arquitectura a utilizar.

La **arquitectura de software** es la estructura o las estructuras del sistema, que incluyen los componentes del software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos. Permite analizar la efectividad del diseño para cumplir con los requisitos establecidos, considerar opciones arquitectónicas en una etapa en que aún resulte relativamente fácil hacer cambios al diseño, y reducir los riesgos asociados con la construcción del software.

El software que se construye cuenta con una arquitectura basada en un modelo de capas, muestra el estilo arquitectónico Modelo Vista Controlador (MVC) e implementa el patrón arquitectónico Fachada.

Los **estilos arquitectónicos** definen los patrones posibles de las aplicaciones, permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos. Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema estableciendo una estructura para todos los componentes en el mismo.

Los **patrones arquitectónicos** al igual que los estilos, imponen una transformación en el diseño de una arquitectura aunque el alcance del patrón es menor ya que se concentra en un aspecto en vez de hacerlo en toda la arquitectura. Un patrón impone una regla sobre la arquitectura y es usado junto a un estilo arquitectónico para determinar la forma de la estructura general de un sistema.

3.1.1. Modelo de capas.

El modelo de capas de una arquitectura organiza el sistema en capas, cada una de las cuales proporciona un conjunto de servicios. Este tipo de arquitectura soporta el desarrollo incremental de sistemas y evita que los cambios en una de las capas afecten directamente al resto.

La arquitectura se encuentra representada por 5 capas lógicas que dan un alto nivel de encapsulamiento de las responsabilidades, permitiendo reducir al máximo el acoplamiento y aumentar la reutilización entre las mismas.

Descripción de las capas:

- **Capa de presentación:**

En esta capa se encuentran todas las interfaces que le son mostradas al usuario haciendo uso del estilo *MVC* y los componentes necesarios para su correcto funcionamiento de las mismas, como son: los ficheros *Java Script, que* posibilita la integración con los componentes de *jquery* y define los componentes *web* a utilizar y *CSS* que contiene todos los ficheros de estilos para la aplicación. Esta capa se encuentra representada por el proyecto *web* de la aplicación, y tiene interacción directa con la capa de negocio.

- **Capa de negocio:**

Esta capa contiene todas las clases necesarias para realizar correctamente todas las funcionalidades del sistema. Contiene las clases controladoras, que son las que manejan todas las operaciones sobre las entidades de negocio definidas y las clases entidades que representan lo que se gestiona en la aplicación. Se encuentra constituida por dos proyectos que agrupan los diferentes componentes: ***Provisioning.CoreLib*** *y* ***Provisioning.Entities*** y tiene relación directa con las capas: Procesos y Servicios y Acceso a Datos.

- **Capa de procesos y servicios:**

En esta capa se encuentran los servicios que provee el módulo Motor de Tareas, el cual gestiona la lógica de los procesos de negocio representada por *Workflows* permitiendo la realización del proceso de aprobación de cuentas de usuario a través de un flujo definido por 6 estados por donde se rigen las solicitudes, estos son: creada, revisada, aprobada, confirmada, revocada y completada. El proyecto que se encuentra relacionado es: **Provisioning.EngineInterfaces** que representa el módulo Motor de Tareas.

- **Capa de acceso a datos:**

Es el componente que da soporte a las funcionalidades de la capa de negocio y se encuentra relacionado con una fuente de datos. 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. La principal función de

esta capa 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 a través del *Entity Framework*.

- **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 utilizando como Sistema Gestor de Base de Datos *Oracle 11g*.

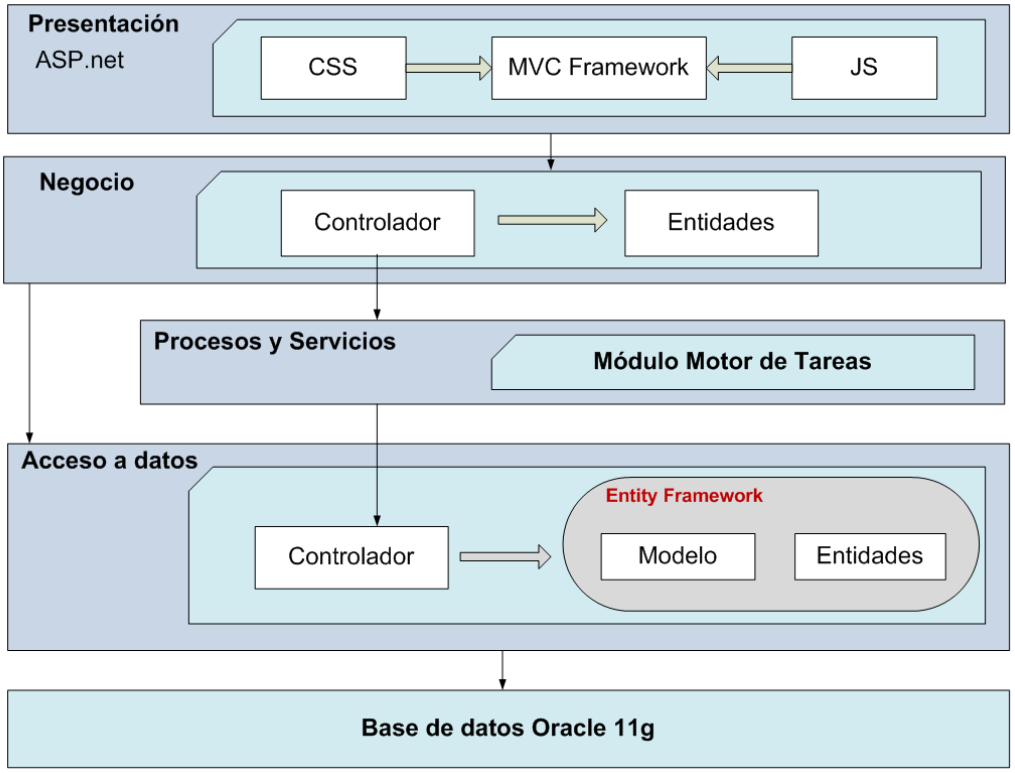


Figura 3: Vista lógica detallada de la arquitectura de software

3.1.2. Modelo Vista Controlador (*MVC*).

Es un estilo de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos y por tanto el Modelo, las Vistas y los Controladores se tratan como entidades independientes. (16)

MVC se manifiesta habitualmente en aplicaciones *web*, donde la **vista** es la página y el código que suministra de datos dinámicos a la página, el **modelo** es el Sistema Gestor de Base de Datos y la lógica de negocio y el **controlador** es el responsable de recibir los eventos de entrada desde la vista. En su forma más general está dado por un modelo, múltiples controladores que lo manipulan, y varias vistas de los datos del modelo, que cambian automáticamente cuando cambia el estado del mismo. A continuación se definen cada una de las partes que componen al *MVC*:

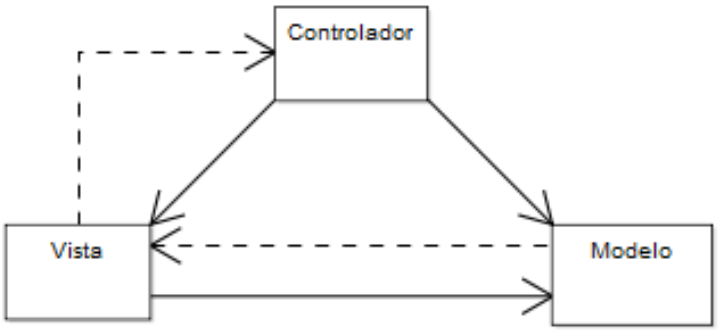


Figura 4: Relación Modelo-Vista-Controlador.

El *Modelo* (*Model*) es el objeto que representa el manejo de los datos y controla todas sus transformaciones. Este no tiene conocimiento específico de los controladores o de las vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas, y notificar a éstas cuando cambia el modelo.

La Vista (*View*) es el objeto que maneja la presentación visual de los datos representados por el modelo. Genera una representación visual del modelo y muestra los datos al usuario. Interactúa con él a través de una referencia al mismo.

El Controlador (*Controller*) es el objeto que suministra significado a las órdenes del usuario, operando sobre los datos contenidos por el modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del modelo o por alteraciones de la vista. (16)

En la solución propuesta se utiliza el estilo arquitectónico MVC permitiendo la definición de la estructura de la capa de presentación de la arquitectura utilizada y administrando el modo en que los datos son mostrados al usuario.

3.2. Patrones de diseño.

Los patrones de diseño tratan los problemas comunes que se presenten, con el fin de proponer soluciones a ellos. Existen diversas formas de implementar los patrones de diseño, los detalles para estas implementaciones son denominados estrategias. Existen 3 elementos que constituyen un patrón, el "nombre" que describe el problema de diseño, el "problema" que describe cuándo aplicar el patrón y la "solución" que describe los elementos que componen el diseño, sus relaciones, responsabilidades y colaboración. (16) Para darle solución a un problema ofrecen una descripción de los pasos a seguir. Los patrones de diseño se dividen en varias categorías:

- **Patrones creacionales:** Contienen aquellos patrones de *software* que se centran en la forma de crear las clases y sus instancias, así como el uso que recibirán una vez creadas.

- **Patrones estructurales:** Contienen aquellos patrones de *software* que se centran en la composición y estructura de las clases y en la herencia entre ellas.
- **Patrones de comportamiento:** Contienen aquellos patrones de *software* que se centran en la comunicación entre las distintas clases y objetos.

Para describir cómo los distintos tipos de objetos pueden ser organizados para trabajar y colaborar entre ellos, se utilizará el patrón de diseño *Facade* (fachada).

3.2.1. Fachada.

Es un patrón de diseño que sirve para proveer una interfaz unificada que funcione de intermediaria entre un cliente y la interfaz. Se clasifica entre los patrones estructurales, con la intención de proporcionar una interfaz para un grupo de subsistemas, conociendo que clases del subsistema son responsables de una petición y delega las peticiones de los clientes en los objetos del subsistema.

Se basa en una clase sencilla con un grupo de métodos, uno para cada operación permitida, de modo que sean estos métodos los que internamente hagan las operaciones, con el fin de llevar a cabo la correcta lógica de la aplicación. A esta clase se le conoce como fachada, de donde proviene el nombre del patrón. Se trata de hacer una estructura fácilmente entendible que proporcione una forma de trabajar ordenada.

Como se muestra en la siguiente figura se diseñaron las clases *DALAccountManager*, *DALRequestManager*, *DALPasswordPolicyManager* y *DALAdministrationHistoryManager* para la implementación de las reglas de negocio y para ocultar la complejidad de las clases inferiores a la clase controladora *AdministrationController* cuando las solicita.

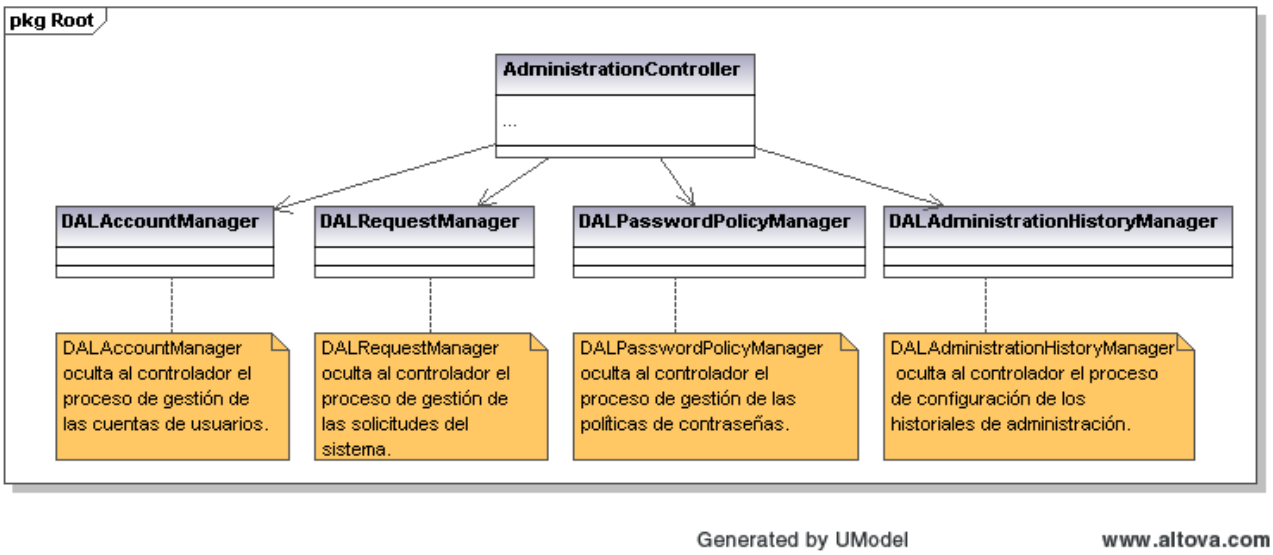


Figura 5: Uso del patrón fachada en el módulo.

3.3. 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.

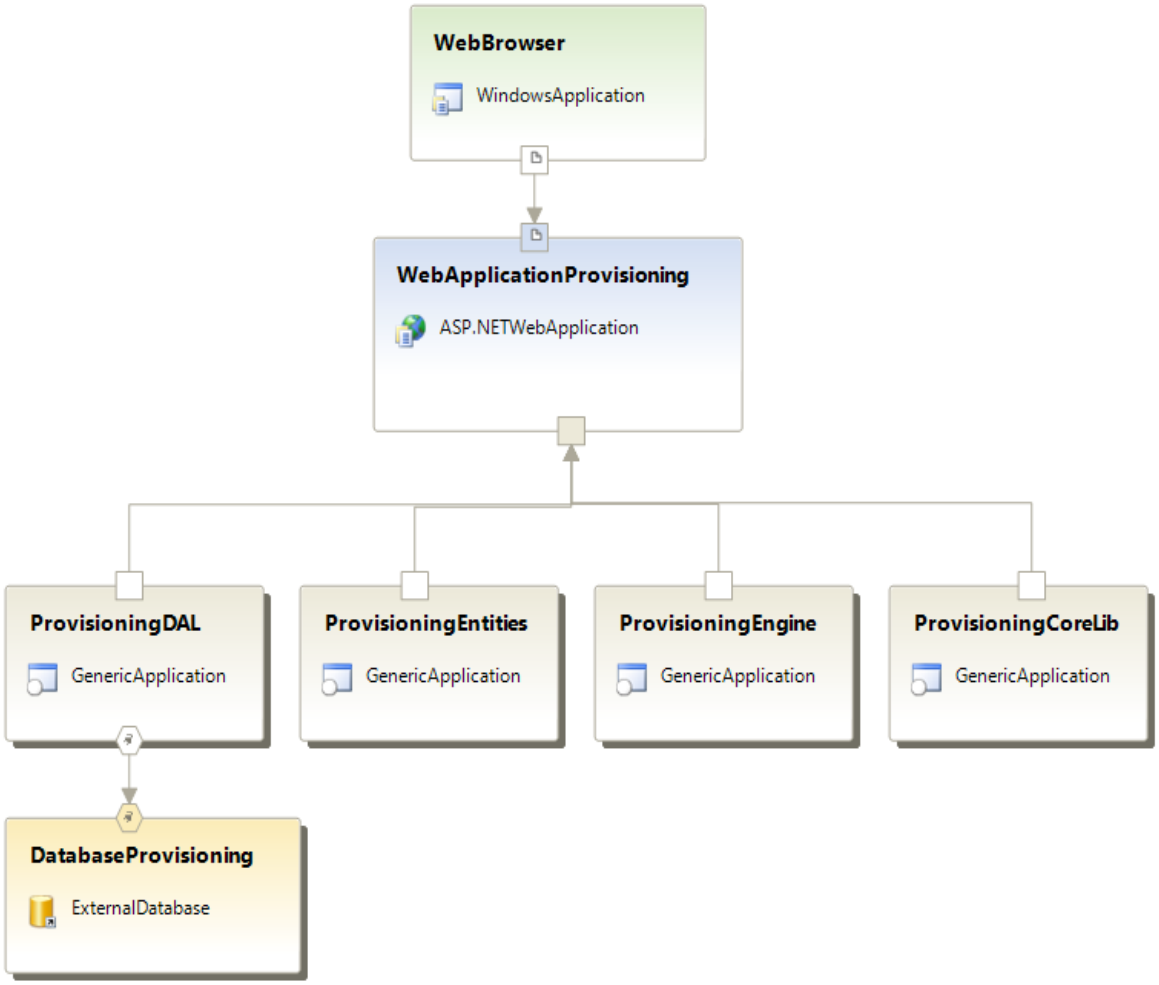


Figura 6: Diagrama de aplicación del Sistema de Aprovisionamiento.

3.4. 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. (Ver *Anexo 7:* "Interacción de la Aplicación de Aprovisionamiento en un entorno de trabajo real".)

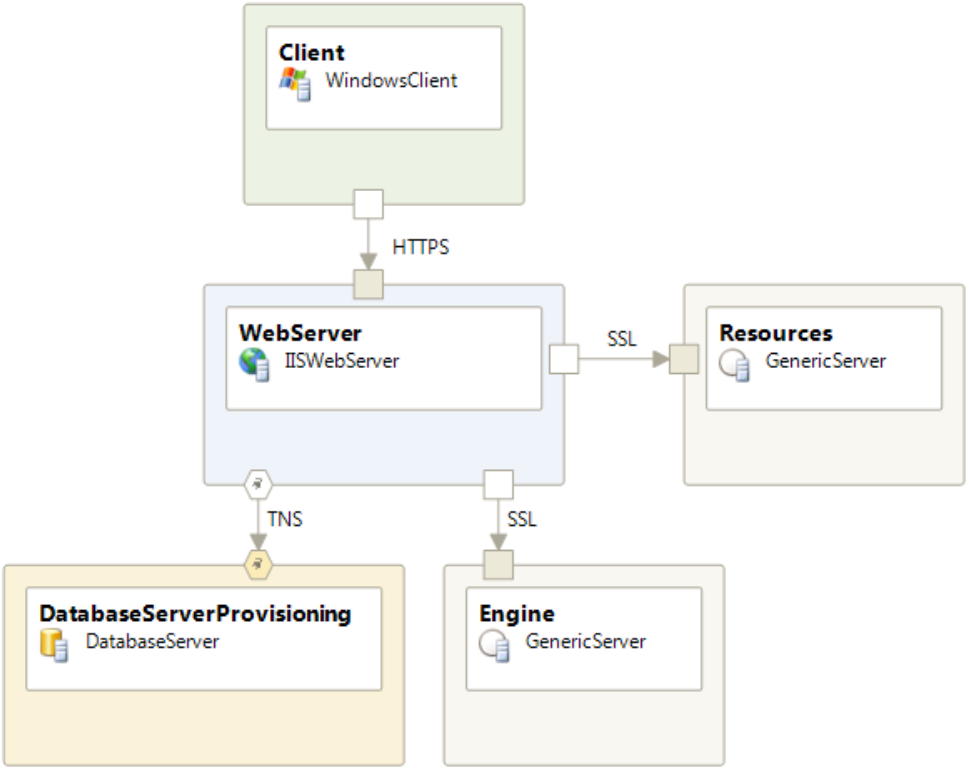
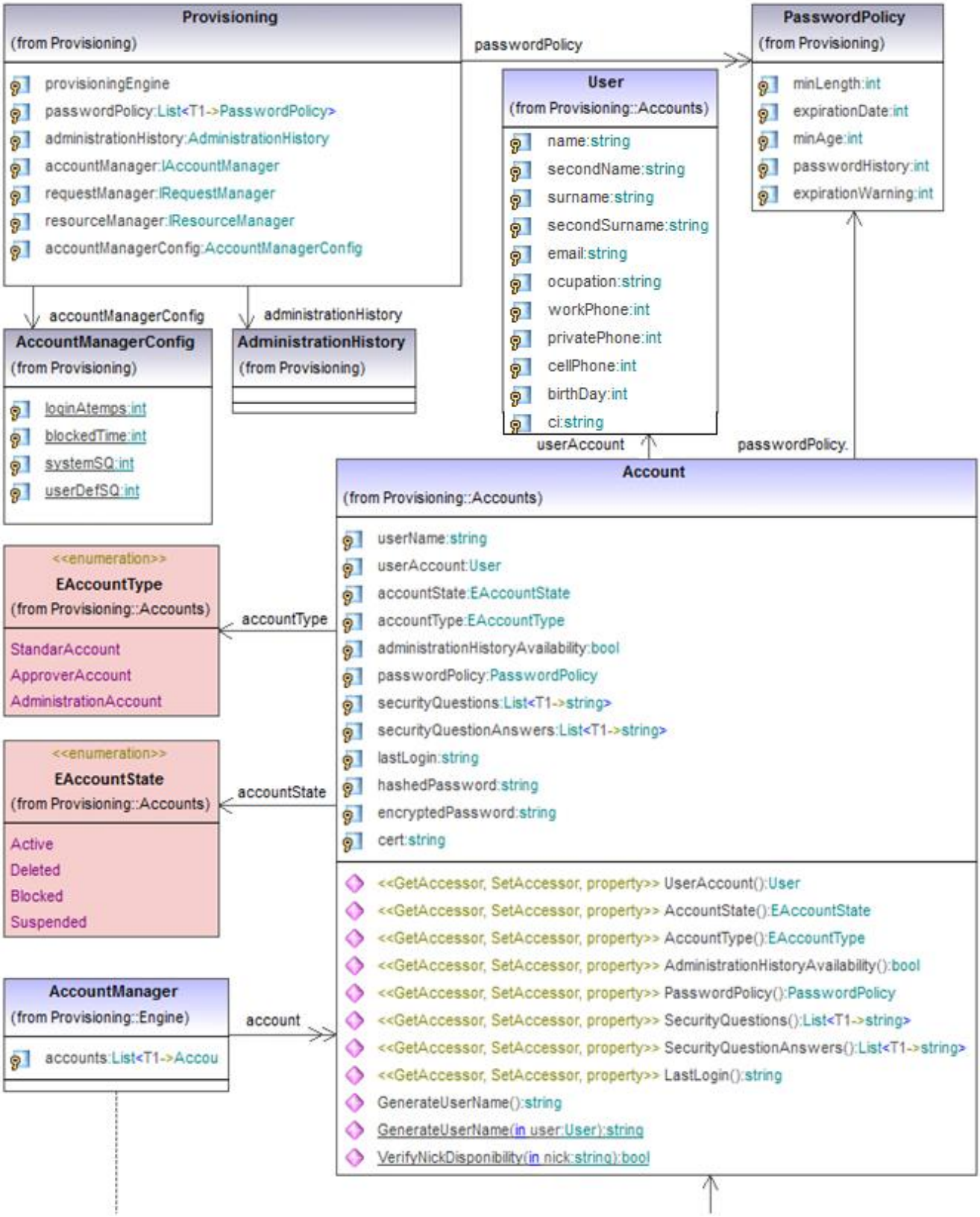


Figura 7: Diagrama lógico de centro de datos del Sistema de Aprovisionamiento.

3.5. 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. Se utilizan para modelar la vista de diseño, que incluye modelar el vocabulario del sistema, las colaboraciones o los esquemas.

En el siguiente diagrama se muestran los atributos y métodos de cada clase identificada en la solución propuesta y se representa de una forma sencilla la colaboración y las responsabilidades de las distintas clases que conforman el sistema.


Provisioning
(from Provisioning)
provisioningEngine
passwordPolicy:List<T1->PasswordPolicy>
administrationHistory:AdministrationHistory
accountManager:IAccountManager
requestManager:IRequestManager
resourceManager:IResourceManager
accountManagerConfig:AccountManagerConfig
passwordPolicy
PasswordPolicy
(from Provisioning)
minLength:int
expirationDate:int
minAge:int
passwordHistory:int
expirationWarning:int
User
(from Provisioning::Accounts)
name:string
secondName:string
surname:string
secondSurname:string
email:string
ocupation:string
workPhone:int
privatePhone:int
cellPhone:int
birthDay:int
ci:string
accountManagerConfig
administrationHistory
AccountManagerConfig
(from Provisioning)
loginAtemps:int
blockedTime:int
systemSQ:int
userDefSQ:int
AdministrationHistory
(from Provisioning)
userAccount
passwordPolicy.
Account
(from Provisioning::Accounts)
userName:string
userAccount:User
accountState:EAccountState
accountType:EAccountType
administrationHistoryAvailability:bool
passwordPolicy:PasswordPolicy
securityQuestions:List<T1->string>
securityQuestionAnswers:List<T1->string>
lastLogin:string
hashedPassword:string
encryptedPassword:string
cert:string
<<GetAccessor, SetAccessor, property>> UserAccount():User
<<GetAccessor, SetAccessor, property>> AccountState():EAccountState
<<GetAccessor, SetAccessor, property>> AccountType():EAccountType
<<GetAccessor, SetAccessor, property>> AdministrationHistoryAvailability():bool
<<GetAccessor, SetAccessor, property>> PasswordPolicy():PasswordPolicy
<<GetAccessor, SetAccessor, property>> SecurityQuestions():List<T1->string>
<<GetAccessor, SetAccessor, property>> SecurityQuestionAnswers():List<T1->string>
<<GetAccessor, SetAccessor, property>> LastLogin():string
GenerateUserName():string
GenerateUserName(in user:User):string
VerifyNickDisponibility(in nick:string):bool
<<enumeration>>
EAccountType
(from Provisioning::Accounts)
StandarAccount
ApproverAccount
AdministrationAccount
accountType
<<enumeration>>
EAccountState
(from Provisioning::Accounts)
Active
Deleted
Blocked
Suspended
accountState
AccountManager
(from Provisioning::Engine)
accounts:List<T1->Accou
account

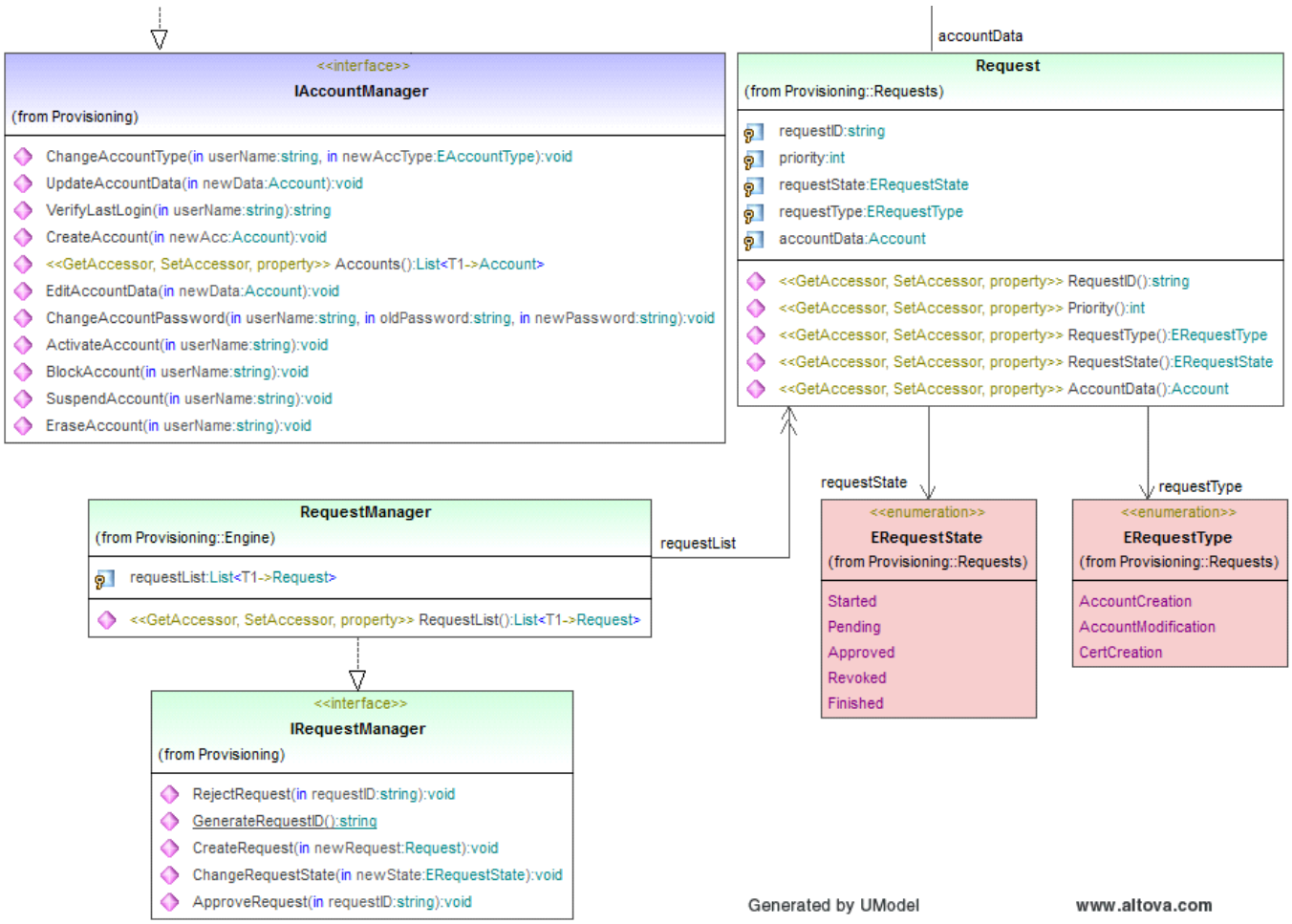


Figura 8: Diagrama de clases del Sistema de Aprovisionamiento.

3.5.1. Descripción de las clases controladoras.

Las clases controladoras tienen la responsabilidad de manejar las entidades del negocio y la información necesaria a persistir en la base de datos. En el diagrama propuesto estas clases implementan interfaces que exponen las funcionalidades del sistema, logrando independencia entre las diferentes capas y posibilitando la escalabilidad del sistema.

Nombre	*AccountManager*	
Descripción	Encargada de administrar las cuentas de usuarios.	
Atributo	**Tipo**	**Descripción**
account	*List <**Account**>*	Listado de las cuentas de usuarios.
Método	**Descripción**	
Constructor and Property: Get(), Set()	Constructor y propiedades de todos los atributos que se deseen.	
ChangeAccountType()	Cambiar el tipo de la cuenta.	
UpdateAccountData()	Actualizar los datos de la cuenta.	
VerifyLastLogin()	Verificar el último acceso al sistema.	

CreateAccount()	Crear una nueva cuenta.
EditAccountData()	Editar los datos de una cuenta.
ChangeAccountPassword()	Cambiar contraseña de la cuenta.
ActivateAccount()	Activar una cuenta.
BlockAccount()	Bloquear una cuenta.
SuspendAccount()	Suspender una cuenta.
EraseAccount()	Eliminar una cuenta.

Tabla 26: Descripción de la entidad *AccountManager*.

Nombre	*RequestManager*	
Descripción	Administra los datos de las solicitudes.	
Atributo	**Tipo**	**Descripción**
requestList	*List <**Request**>*	Listado de las solicitudes.
Método	**Descripción**	
Constructor and Property: Get(), Set()	Constructor y propiedades de todos los atributos que se deseen.	
RejectRequest()	Rechazar una solicitud.	
GenerateRequestID()	Generar el identificador de una solicitud.	
CreateRequest()	Crear una nueva solicitud.	
ChangeRequestState()	Cambiar el estado de una solicitud.	
ApproveRequest()	Aprobar una solicitud.	

Tabla 27: Descripción de la entidad *RequestManager*.

3.5.2. Descripción de las clases entidades.

Las clases entidades tienen como principal objetivo describir el modelo de objetos relacionados con el negocio.

Nombre	*Account*	
Descripción	Contiene los datos correspondientes a las cuentas de usuarios.	
Atributo	**Tipo**	**Descripción**
userName	*string*	Nombre del usuario.
userAccount	***User***	Datos del usuario.
accountState	***EAccountState***	Estado de la cuenta.
accountType	***EAccountType***	Tipo de cuenta.
administrationHistoryAvailability	*bool*	Disponibilidad de administración.
passwordPolicy	***PasswordPolicy***	Políticas de contraseña.
securityQuestions	*List <string>*	Listado de preguntas de seguridad.
securityQuestionAnswers	*List <string>*	Listado de respuestas a las preguntas.
lastLogin	*string*	Último acceso a la aplicación.
hashedPassword	*string*	Función *hash* aplicada a la contraseña.
encryptedPassword	*string*	Contraseña encriptada.
cert	*string*	Certificados requerido.
Método	**Descripción**	
Constructor and Property: Get(), Set()	Constructor y propiedades de todos los atributos que se deseen.	
GenerateUserName():string	Generar el nombre del usuario.	

*GenerateUserName(user: **User**):string*	Generar el nombre pasándole el usuario por parámetro.
VerifyNickDisponibility(nick: string):bool	Verifica si el *nick* está disponible para el usuario.

Tabla 28: Descripción de la entidad *Account*.

Nombre	*Request*	
Descripción	Contiene los datos correspondientes a las solicitudes.	
Atributo	**Tipo**	**Descripción**
requestID	*string*	Identificador de la solicitud.
priority	*int*	Prioridad de la solicitud.
requestState	***EAccountState***	Estado de la solicitud.
recuestType	***EAccountType***	Tipo de solicitud.
accountData	***Account***	Datos de la cuenta.
Método	**Descripción**	
Constructor and Property: Get(), Set()	Constructor y propiedades de todos los atributos que se deseen.	

Tabla 29: Descripción de la entidad *Request*.

Nombre	*PasswordPolicy*	
Descripción	Encargada de las políticas de contraseña.	
Atributo	**Tipo**	**Descripción**
minLength	*int*	Mínima longitud de la contraseña.
expirationDate	*int*	Día de expiración de la contraseña.
minAge	*int*	Mínima edad de la contraseña.
passwordHistory	*int*	Elemento historial de la contraseña.
expirationWarning	*int*	Advertencia de expiración de contraseña.
Método	**Descripción**	
Constructor and Property: Get(), Set()	Constructor y propiedades de todos los atributos que se deseen.	

Tabla 30: Descripción de la entidad *PasswordPolicy*.

Nombre	*User*	
Descripción	Contiene los datos correspondientes a los usuarios del sistema.	
Atributo	**Tipo**	**Descripción**
name	*string*	Nombre del usuario.
secoundName	*string*	Segundo nombre.
surname	*string*	Primer apellido
secoundSurname	*string*	Segundo apellido.
email	*string*	Dirección de correo electrónico.
ocupation	*string*	Ocupación del usuario.
workPhone	*int*	Teléfono del trabajo.
privatePhone	*int*	Teléfono particular.
cellPhone	*int*	Teléfono móvil.
birthDay	*int*	Fecha de cumpleaños del usuario.
ci	*string*	Número del carnet de identidad.
Método	**Descripción**	
Constructor and Property: Get(), Set()	Constructor y propiedades de todos los atributos que se deseen.	

Tabla 31: Descripción de la entidad *User*.

3.6. Utilización de *Workflow* e integración con el módulo Motor de Tareas.

Los flujos de trabajo se pueden personalizar aprovechando las funcionalidades que brinda la tecnología de *Windows Workflow Foundation* para cambiar el flujo de aprobación y asegurar que las acciones sobre las cuentas son correctas.

El módulo Motor de Tareas (*Engine*) es el que proporciona el flujo de aprobación definido para gestionar las solicitudes en la aplicación, el presentado a continuación representa el flujo principal. Las actividades dentro de cada estado están diseñadas con el flujo de trabajo correspondiente según la acción a desarrollar. Los estados definidos para las solicitudes son: creada, revisada, aprobada, rechazada, confirmada y completada.

Descripción del flujo de aprobación. (Ver flujo de funcionamiento en el Anexo 6)

El módulo Motor de Tareas define cuatro tipos de flujos de trabajos para atender las solicitudes los cuales se diferencian por la cantidad de niveles de aprobación que contienen.

A una solicitud enviada por un usuario estándar se le aplica el flujo de trabajo con el máximo nivel de aprobación mientras que para los administradores se define un flujo de trabajo sin niveles de aprobación para la creación, modificación y eliminación de las cuentas, de manera tal que en un futuro esta configuración de los niveles puede ser modificada sin generar cambios importantes en el sistema.

Al recibir una solicitud el módulo Motor de Tareas ejecuta el flujo de aprobación correspondiente y automáticamente se pone a disposición al módulo Administración que provee la interfaz para realizar todas las acciones sobre las cuentas. Si una solicitud aceptada por el aprobador pasa al estado "aprobada" de lo contrario pasa al estado "rechazada". Para el caso de las aprobadas se puede realizar las acciones para rechazarla, chequearla nuevamente o completarla.

En caso de que la solicitud lleve un tiempo de espera para ser revisada mayor al configurado por el administrador pasará la solicitud de manera automática al estado "rechazada". Para todos los casos en que la solicitud sea rechazada puede ser revisada nuevamente por el aprobador y de ser correcta enviarla al estado anterior.

La solicitud se completa al pasar por todos los niveles de aprobación dándole terminación al flujo de trabajo y logrando la creación, modificación o eliminación exitosa de la cuenta de usuario.

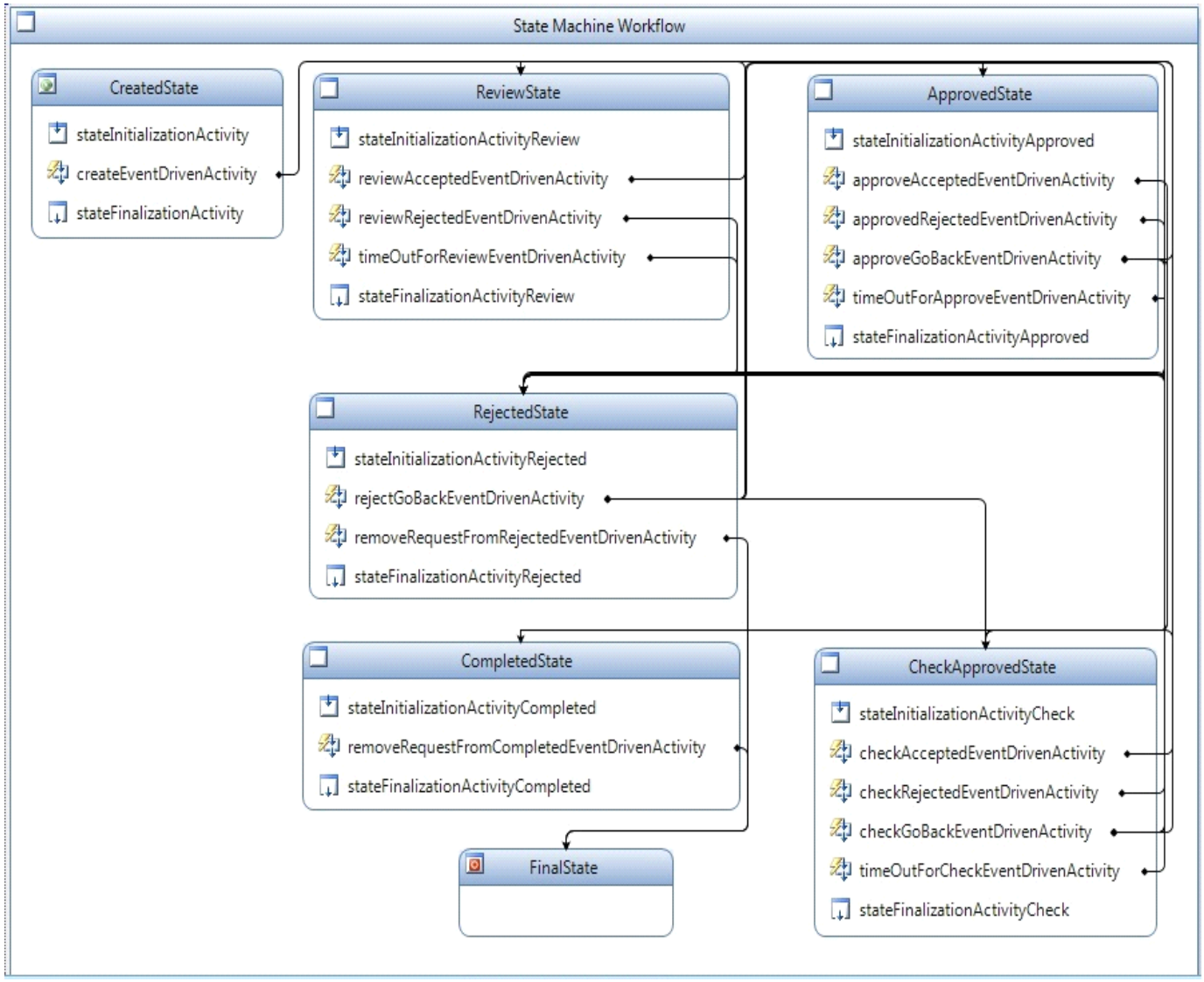


Figura 9: Utilización de *workflow* en la Aplicación de Aprovisionamiento.

3.7. Modelo de datos del sistema.

Un modelo de datos es una representación lógica y física de los datos persistentes usados por la aplicación, permite describir:

- Las estructuras de datos: El tipo de datos que hay en la base y la forma en que se relacionan.
- Las restricciones de integridad: Un conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada.
- Operaciones de manipulación de los datos: Operaciones de agregado, borrado, modificación y recuperación de los datos de la base.

(Ver modelo de la base de datos en el Anexo 8)

3.7.1. Descripción de las clases persistentes.

Nombre	*ProvAccount*	
Descripción	Contiene los datos correspondientes a las cuentas de usuarios.	
Atributo	**Tipo**	**Descripción**
Idaccount	*RAW(16)*	Identificador de la cuenta.
Username	*VARCHAR2(50)*	Nombre del usuario.
Encryptedpassword	*VARCHAR2(250)*	Contraseña encriptada.
Hashedpassword	*CHAR(20)*	Contraseña con algoritmo *hash*.
Passwordexpirationdate	*TIMESTAMP(6)*	Tiempo de expiración de la cuenta.
Accesslevel	*NUMBER(1)*	Nivel de acceso.
Adminhistoryavailability	*NUMBER(1)*	Historial de administración.
Lastlogin	*TIMESTAMP(6)*	Último acceso al sistema.

Tabla 32: Descripción de la clase persistente ProvAccount.

Nombre	*ProvPasswordPolicy*	
Descripción	Encargada de las políticas de contraseña.	
Atributo	**Tipo**	**Descripción**
Idpasswordpolicy	*RAW(16)*	Identificador de la política de contraseña.
Name	*VARCHAR2(50)*	Nombre de la política de contraseña.
Description	*VARCHAR2(250)*	Descripción de la política de contraseña.
Minlength	*NUMBER(2)*	Mínima longitud de la contraseña.
Maxage	*NUMBER(3)*	Máxima edad de la contraseña.
Minage	*NUMBER(3)*	Mínima edad de la contraseña.
Passwordhistory	*NUMBER(2)*	Elemento historial de la contraseña.
Expirationwarning	*NUMBER(3)*	Advertencia de expiración de contraseña.
Passwordpolicytype	*NUMBER(1)*	Tipo de política de contraseña.

Tabla 33: Descripción de la clase persistente ProvPasswordPolicy.

Nombre	*ProvRequest*	
Descripción	Contiene los datos correspondientes a las solicitudes.	
Atributo	**Tipo**	**Descripción**
Idrequest	*RAW(16)*	Identificador de la solicitud.
Trackid	*CHAR(6)*	Identificador de seguimiento de la solicitud.
Creationdate	*TIMESTAMP(6)*	Fecha de creación de la solicitud.

Tabla 34: Descripción de la clase persistente ProvRequest.

Nombre	*ProvUser*	
Descripción	Contiene los datos correspondientes a los usuarios.	
Atributo	**Tipo**	**Descripción**
Iduser	*RAW(16)*	Identificador del usuario.
Name	*VARCHAR2(50)*	Nombre del usuario.
Secoundname	*VARCHAR2(50)*	Segundo nombre.

Surname	*VARCHAR2(50)*	Segundo apellido.
Birthday	*TIMESTAMP(6)*	Fecha de cumpleaños.
Ci	*VARCHAR2(11)*	Número del carnet de identidad.
Lastname	*VARCHAR2(50)*	Primer apellido.
Email	*VARCHAR2(50)*	Dirección de correo electrónico.
Ocupation	*VARCHAR2(50)*	Ocupación.
Workphone	*VARCHAR2(15)*	Número de teléfono del trabajo.
Privatephone	*VARCHAR2(15)*	Número de teléfono particular.
Cellphone	*VARCHAR2(15)*	Número de teléfono móvil.

Tabla 35: Descripción de la clase persistente ProvUser.

Nombre	*AdministrationHistory*	
Descripción	Contiene los datos correspondientes al historial de administración.	
Atributo	**Tipo**	**Descripción**
Idadministrationhistory	*RAW(16)*	Identificador del usuario.
Userstatedate	*TIMESTAMP(6)*	Fecha de estados.

Tabla 36: Descripción de la clase persistente AdministrationHistory.

3.8. Interfaz gráfica.

- Interfaz *"Gestionar Cuentas":*

La interfaz "Gestionar Cuentas" permite administrar todas las acciones que se realizan sobre las cuentas de usuario existentes en el sistema. Cuenta con las opciones crear, habilitar, eliminar, suspender, bloquear y editar cuentas, además de permitir refrescar el listado y establecer filtros de búsqueda permitiendo facilidades de búsqueda en grandes cantidades de datos.

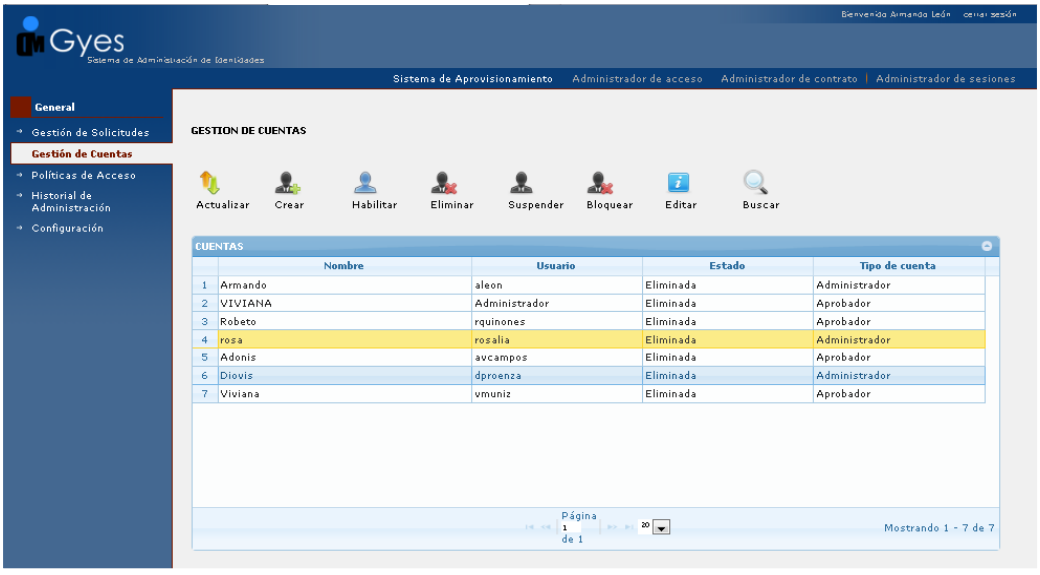


Figura 10: Interfaz "Gestionar Cuentas".

- Opción "Editar" de la interfaz "*Gestionar Cuentas":*

  La Opción "Editar" de la interfaz "*Gestionar Cuentas"* permite modificar los datos de la cuenta seleccionada en el listado, actualizando la información de la cuenta así como los datos personales del usuario.

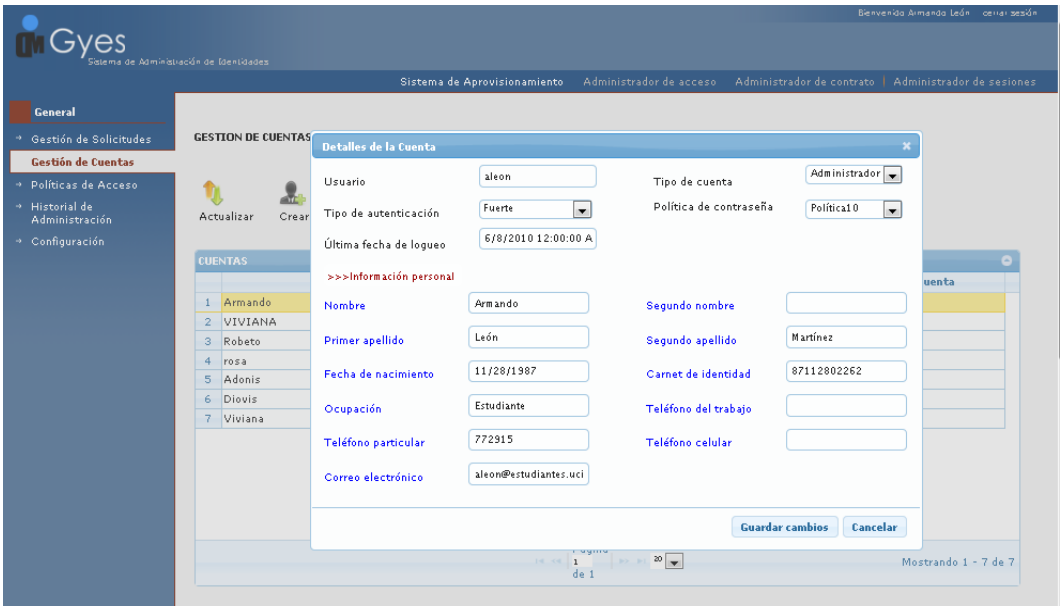


Figura 11: Opción Editar de la interfaz "Gestionar Cuentas".

3.9. Conclusiones.

El sistema desarrollado posibilita gestionar de manera centralizada todo el ciclo de vida de las cuentas de usuarios de la Aplicación de Aprovisionamiento del Sistema de Administración de Identidades, cumpliendo en el tiempo establecido con los objetivos trazados al inicio de la investigación. Las acciones a realizar sobre las cuentas de usuarios, como crear, habilitar, suspender, eliminar y bloquear, se rigen por un flujo de aprobación definido en el módulo Motor de Tareas y que cuenta con 5 estados: creada, revisada, aprobada, confirmada, revocada y completada. Se puede afirmar que la aplicación está lista para integrar con los demás módulos que forman parte de la Aplicación de Aprovisionamiento.

Capítulo 4 Estabilización del sistema

Después de obtenido el sistema se le realizarán las pruebas pertinentes, enfocadas a encontrar los errores existentes y darle solución. El primer paso que la metodología identifica es escribir los tipos de pruebas que se van a desarrollar, para la solución se propone realizar las pruebas unitarias mediante la técnica de caja blanca y para las funcionalidades de la aplicación mediante la técnica de caja negra y se describen las pruebas funcionales para cada escenario que deba validarse.

4.1. Pruebas de caja blanca.

Las pruebas de caja blanca del software se basan en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de pruebas que ejerciten conjuntos específicos de condiciones y/o bucles.(17) Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado.

4.2. Pruebas de caja negra.

Las pruebas de caja negra se refieren a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos de pruebas pretenden demostrar que las funciones del producto son operativas, que la entrada se realiza de manera adecuada, que se produce un resultado correcto y que la integridad de la información externa se mantiene. Las pruebas de caja negra examinan algunos aspectos del modelo fundamental del sistema sin valorar demasiado la estructura lógica interna del software.(17) Se centran principalmente en los requisitos funcionales del software y permiten obtener un conjunto de condiciones de entrada que ejerciten completamente estos requisitos ignorándose la estructura de control.

4.3. Pruebas unitarias.

Una prueba unitaria chequea el correcto funcionamiento de un código verificando los resultados que se esperan. Esto asegura que cada uno de los módulos funcione correctamente por separado con el objetivo de que no falle ningún elemento y que se complete el trabajo que se desea.(18)

Visual Studio Team System 2008 permite generar pruebas específicas para métodos, incluidos los métodos privados, facilitando los pasos necesarios para crear y personalizar pruebas unitarias,

además de ejecutarlas y examinar sus resultados. A continuación se describen las pruebas realizadas con la herramienta del VSTS a las funcionalidades principales del sub-módulo Administrar de Cuentas de la aplicación.

Prueba de Unidad		
Nombre Prueba: *FindAllAccountsTest*		
Estado: Satisfactoria	**Tipo**: Caja Blanca	**Última ejecución**: 02/06/2010
Ejecutado por: *Viviana Muñiz López*	**Verificado por**: *Diovis Proenza Labadie*	
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: Test run completed Results: 1/1 passed; Item(s) checked: 0 Result | Test Name | Project | Error Message Passed | FindAllAccountsTest | Gyes.Provisioning.WebApp.Tests		

Tabla 37: Descripción de la prueba unitaria *FindAllAccountsTest*.

Prueba de Unidad		
Nombre Prueba: *EditAccountDataTest*		
Estado: Satisfactoria	**Tipo**: Caja Blanca	**Última ejecución**: 02/06/2010
Ejecutado por: *Armando León Martínez*	**Verificado por**: *Diovis Proenza Labadie*	
Descripción: Para ejecutar la prueba se debe introducir el nombre de un usuario para compararlo con los guardados en la tabla *Account* de la base de datos de la aplicación, si se encuentra, la función del método es editar los datos existentes por los nuevos entrados.		
Entrada: userName		
Criterio de aceptación:		
Resultado: Test run completed Results: 1/1 passed; Item(s) checked: 0 Result | Test Name | Project | Error Message Passed | EditAccountDataTest | Gyes.Provisioning.WebApp.Tests		

Tabla 38: Descripción de la prueba unitaria *EditAccountDataTest*.

Prueba de Unidad		
Nombre Prueba: *ActivateAccountTest*		
Estado: Satisfactoria	**Tipo**: Caja Blanca	**Última ejecución**: 03/06/2010
Ejecutado por: *Viviana Muñiz López*	**Verificado por**: *Diovis Proenza Labadie*	
Descripción: Para ejecutar la prueba se debe introducir el nombre de un usuario para compararlo con los guardados en la tabla *Account* de la base de datos de la aplicación, si se encuentra, el método le cambia el estado de la cuenta para "Activado".		
Entrada: userName		
Criterio de aceptación: Cambia el estado de la cuenta.		
Resultado:		

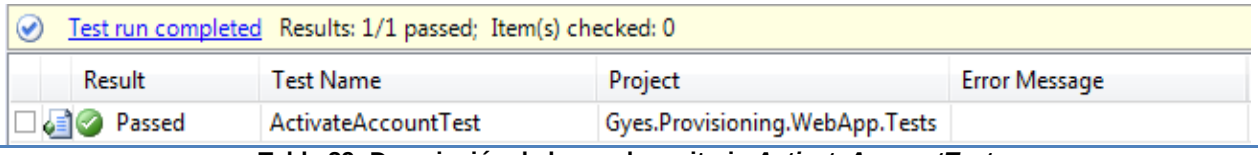


Tabla 39: Descripción de la prueba unitaria *ActivateAccountTest*.

Prueba de Unidad		
Nombre Prueba: *SuspendAccountTest*		
Estado: Satisfactoria	**Tipo**: Caja Blanca	**Última ejecución**: 03/06/2010
Ejecutado por: *Viviana Muñiz López*	**Verificado por**: *Diovis Proenza Labadie*	
Descripción: Para ejecutar la prueba se debe introducir el nombre de un usuario para compararlo con los guardados en la tabla *Account* de la base de datos de la aplicación, si se encuentra, el método le cambia el estado de la cuenta para "Suspendido".		
Entrada: userName		
Criterio de aceptación: Cambia el estado de la cuenta.		
Resultado:		

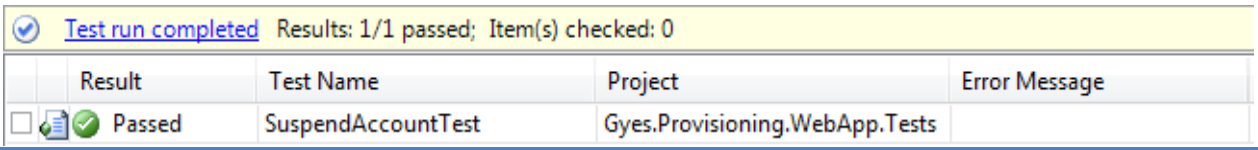


Tabla 40: Descripción de la prueba unitaria *SuspendAccountTest*.

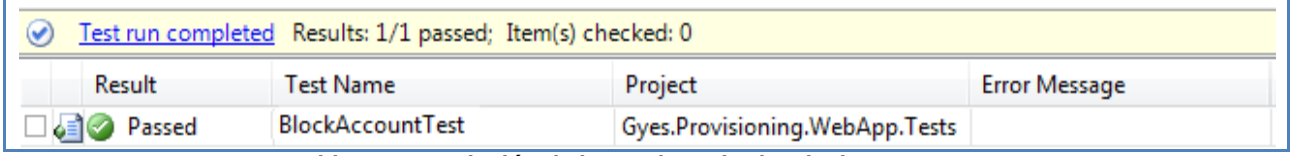
Prueba de Unidad		
Nombre Prueba: *BlockAccountTest*		
Estado: Satisfactoria	**Tipo**: Caja Blanca	**Última ejecución**: 03/06/2010
Ejecutado por: *Viviana Muñiz López*	**Verificado por**: *Diovis Proenza Labadie*	
Descripción: Para ejecutar la prueba se debe introducir el nombre de un usuario para compararlo con los guardados en la tabla *Account* de la base de datos de la aplicación, si se encuentra, el método le cambia el estado de la cuenta para "Bloqueado".		
Entrada: userName		
Criterio de aceptación: Cambia el estado de la cuenta.		
Resultado:  		

Tabla 41: Descripción de la prueba unitaria *BlockAccountTest*.

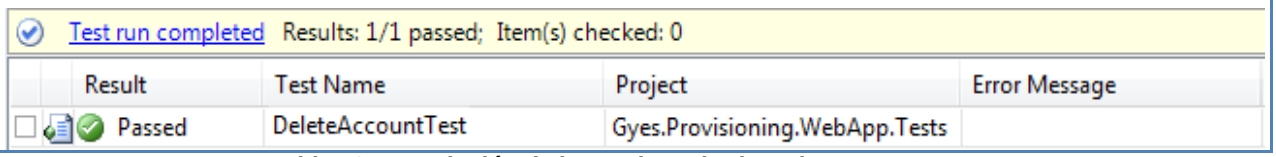
Prueba de Unidad		
Nombre Prueba: *DeleteAccountTest*		
Estado: Satisfactoria	**Tipo**: Caja Blanca	**Última ejecución**: 03/06/2010
Ejecutado por: *Armando León Martínez*	**Verificado por**: *Diovis Proenza Labadie*	
Descripción: Para ejecutar la prueba se debe introducir el nombre de un usuario para compararlo con los guardados en la tabla *Account* de la base de datos de la aplicación, si se encuentra, el método le cambia el estado de la cuenta para "Eliminado".		
Entrada: userName		
Criterio de aceptación: Elimina la cuenta.		
Resultado:  		

Tabla 42: Descripción de la prueba unitaria *DeleteAccountTest*.

4.4. Pruebas de validación del sistema.

A continuación se especificarán las secciones a probar en cada escenario, además se brindará una breve descripción de las variables de entrada que deben introducirse y las reglas que tiene que cumplir el campo a probar.

4.4.1. Secciones a probar en el escenario.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
ES 1: Gestionar cuentas	ES 1.1: Habilitar cuenta	A partir del listado de las cuentas existentes se selecciona la cuenta deseada, accede a la opción Habilitar e inmediatamente se le cambia el estado de la cuenta.
	ES 1.2: Suspender cuenta	A partir del listado de las cuentas existentes se selecciona la cuenta deseada, accede a la opción Suspender e inmediatamente se le cambia el estado de la cuenta.
	ES 1.3: Bloquear cuenta	A partir del listado de las cuentas existentes se selecciona la cuenta deseada, accede a la opción Bloquear e inmediatamente se le cambia el estado de la cuenta.
	ES 1.4: Eliminar cuenta	A partir del listado de las cuentas existentes se selecciona la cuenta deseada, accede a la opción Eliminar e inmediatamente se le cambia el estado de la cuenta.
	ES 1.5: Editar cuenta	A partir del listado de las cuentas existentes se selecciona la cuenta deseada, accede a la opción Editar y se actualizan los datos que se deseen cambiar, inmediatamente se actualiza el listado.
	ES 1.6: Listar cuentas	Se muestra el listado al acceder al sub-módulo Administrar Cuentas.

Tabla 43: Descripción de las secciones a probar en el escenario Gestionar cuentas.

4.4.2. Descripción de las variables de entrada.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Usuario	Campo de texto	No	No se introducen datos porque el campo está deshabilitado.
2	Tipo de cuenta	Lista desplegable	No	No se introducen datos, se seleccionan.
3	Tipo de Autenticación	Lista desplegable	No	No se introducen datos, se seleccionan.
4	Política de contraseña	Lista desplegable	No	No se introducen datos, se seleccionan.
5	Última fecha de *logueo*	Campo de texto	No	No se introducen datos porque el campo está deshabilitado.
6	Nombre	Campo de texto	No	El campo solo admite la entrada de letras.
7	Segundo nombre	Campo de texto	Si	El campo permite nulo y de entrarle valores deben ser solo letras.
8	Primer apellido	Campo de texto	No	El campo solo admite la entrada de letras.
9	Segundo apellido	Campo de texto	No	El campo solo admite la entrada de letras.
10	Fecha de nacimiento	Campo de texto	Si	No se introducen datos, se seleccionan del calendario.
11	Carnet de ID	Campo de texto	No	El campo solo admite la entrada de números.
12	Ocupación	Campo de texto	Si	El campo permite nulo y de entrarle valores deben ser solo letras.
13	Teléfono del trabajo	Campo de texto	Si	El campo permite nulo y de entrarle valores deben ser solo números.
14	Teléfono particular	Campo de texto	Si	El campo permite nulo y de entrarle valores deben ser solo números.
15	Teléfono celular	Campo de texto	Si	El campo permite nulo y de entrarle valores deben ser solo números.
16	Correo electrónico	Campo de texto	Si	Admite cualquier caracter solo especificar que debe escribirse en formato de correo.

Tabla 44: Descripción de las variables de entrada en el escenario Gestionar cuentas.

4.4.3. Matriz de Datos.

Para los escenarios Habilitar cuenta, Suspender cuenta, Bloquear cuenta y Eliminar cuenta no se entran datos por lo que no se representan en la matriz estás acciones ya que se realizan seleccionando la cuenta que se desee a partir de las mostradas en el listado. (Ver Anexo 9)

4.5. Resultados de las pruebas.

Buscar los elementos más relevantes de las pruebas es el mayor peso a la hora de analizar los resultados una vez ejecutadas. Después de aplicarle varias pruebas a la aplicación luego de terminada su primera versión se puede apreciar que de manera general los errores encontrados han sido solucionados en su totalidad y por tanto no afectan el desarrollo de la aplicación, por lo que se encuentra lista para ser integrada con los demás módulos.

4.6. 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

El desarrollo de este trabajo permitió elaborar el módulo Administración de la Aplicación de 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 el patrón Fachada. La arquitectura está representada por 5 capas lógicas que dan un alto nivel de encapsulamiento de las responsabilidades.
- El sistema desarrollado permite gestionar toda la información referente a las cuentas de usuario en el 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

Se recomienda a toda persona interesada en mejorar o continuar con este trabajo:

- Desarrollar otras versiones de este sistema incorporando nuevas funcionalidades que se crea puedan ser útiles para lograr un producto de mayor calidad.
- Desarrollar la integración con las demás aplicaciones que conformarán el Sistema de Administración de Identidades.
- Desarrollar el manual de usuario del módulo implementado para capacitar al personal que lo utilizará.

Referencias bibliográficas

1. UCI, D. S. D**. *Introducción a la Seguridad Informática.*** Disponible en: http://eva.uci.cu.

2. CORPORATION, M. ***Gestión de identidades.*** 2004.

3. JIMÉNEZ, G. ***Gestión de identidad: seguridad y libertad para crecer.*** Disponible en: http://www.borrmart.es/articulo_redseguridad.php?id=819&numero=20.

4. UCI, D. D. C. D. C. E. ***Introducción a la Administración de empresas.*** Disponible en: http://eva.uci.cu.

5. UCI, D. I. D. S. ***Introducción a la Ingeniería de Software.*** Disponible en: http://eva.uci.cu.

6. COMPANY, A. ***Altova Umodel 2009 User & Reference Manual.*** 2009.

7. CORPORATION, M. ***MSF for Agile Software Development Process Guidance.*** 2006.

8. CORPORATION, M .***NET Framework de Microsoft.*** Disponible en: http://msdn.microsoft.com/es-es/netframework/default.aspx.

9. CORPORATION, M. ***Introducción a Entity Framework.*** Disponible en: http://msdn.microsoft.com/es-es/library/bb399567%28VS.90%29.aspx.

10. BUKOVICS, B. ***Pro WF Windows Workflow in .NET 3.5.*** 2008. ISBN 9781430209768.

11. MULTIMEDIA, A. ***Biblia de ASP.NET.*** Madrid: ISBN 8441513856.

12. CORPORATION, M. ***Documentación de Visual Studio Team System*.** Disponible en: http://msdn.microsoft.com/es-es/library/fda2bad5%28VS.80%29.aspx.

13. CORPORATION, M. ***Visual Studio Team System*.** Disponible en: http://msdn2.microsoft.com/en-us/teamsystem/default.aspx.

14. EMPRESARIALES, C.-C. A. Y. S. ***Embarcadero ER/Studio*.** Disponible en: http://www.casenet.com.mx/index.php?option=com_content&view=article&id=58&Itemid=59.

15. ORACLE. ***Arquitectura Oracle Database 11g en Windows.*** 2007.

16. UCI, D. I. D. S. ***Arquitectura y Patrones de diseño.*** Disponible en: http://eva.uci.cu.

17. IN, B. S. ***White and Black Box Testing.*** Disponible en: https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/white-box/259-BSI.html .

18. CORPORATION, M. ***Trabajar con pruebas unitarias.*** Disponible en: http://msdn.microsoft.com/es-es/library/ms182515%28VS.80%29.aspx.

Bibliografía

1. ***Patterns Home Page***. Disponible en: http://hillside.net/patterns/.

2. CA. ***CA Identity Manager.*** Disponible en: http://www.ca.com/us/user-provisioning.aspx.

3. CORPORATION, M. ***What is Microsoft .NET Framework?*** Disponible en: http://www.microsoft.com/NET/.

4. CORPORATION, M. ***Visual C#.*** Disponible en: http://msdn.microsoft.com/es-es/vcsharp/default.aspx.

5. CORPORATION, O. ***Oracle Identity Management***. Disponible en: www.oracle.com.

6. **ERICH GAMMA, R. H., RALPH JOHNSON, JOHN VLISSIDES**. *Patrones del diseño: Elementos del software orientado al objeto reutilizable.* Addison Wesley ed. Madrid: 2002. ISBN 0201633612.

7. GARTNER. ***Magic Quadrant for User Provisioning***. 2008.

8. GRADY BOOCH, J. R., IVAR JACOBSON. ***UML - El Lenguaje Unificado de Modelado.*** Madrid: 1999. ISBN 8478290761.

9. HOLZNER, S. ***Design patterns for dummies***. Wiley Publishing Inc ed. EE.UU: 2006.

10. IBM. ***IBM Tivoli Identity Manager.*** Disponible en: http://www-01.ibm.com/software/tivoli/products/identity-mgr/.

11. KATRIB, M. ***Visual Studio 2008. Desafía todos los retos.*** 2008.

12. KITTA, T. ***Professional Windows Workflow Foundation.*** Wrox Press ed. 2007. ISBN 9780470053867.

13. LARMAN, C. ***UML y patrones.*** Prentice Hall ed. México: 2003. ISBN 0596001657.

14. MORAL, R. ***Gestión de identidad y acceso. La llave de la agilidad empresarial.*** 2005.

15. NOVELL. ***Novell Identity Manager.*** Disponible en: http://www.novell.com/products/identitymanager/.

16. ORACLE. ***Guía del administrador de negocio de Sun Identity Manager 8.1.*** Disponible en: http://docs.sun.com.

17. PRESSMAN, R. S. ***Ingeniería del Software. Un enfoque práctico. (Sexta edición)***. 2005. ISBN 9701054733.

18. SIA. ***Gestión Avanzada de Identidades IRENE - Enterprise Identity Management***. 2006.

19. SUN. ***Sun Identity Manager.*** Disponible en: http://www.sun.com/software/products/identity_mgr/faqs.xml.

20. URIOS, I. A. Y. X. ***La gestión de identidades y capacidades por las administraciones públicas.*** . 2006.

21. VILLAR, R. ***El Reto de la Seguridad en las Redes de Telecomunicaciones.*** 2006.

22. SOMMERVILLE, I. ***Ingeniería del Software (Séptima edición)***. 2005. ISBN 8478290745.

Glosario de términos

A

Aplicación*:** Cualquier programa que corra en un sistema operativo y que haga una función específica para un usuario. Por ejemplo, procesadores de palabras, bases de datos, agendas electrónicas, etc., ***VII, ***X***, ***2***, ***3***, ***4***, ***10***, ***21***, ***22***, ***23***, ***44***, ***50***, ***53***, ***61***, ***71***, ***72***

Autenticación: La autenticación es el acto de proveer una identidad a una red, aplicación o recurso. Las técnicas de autenticación van desde una simple entrada de identificador (ID) de usuario y contraseña hasta potentes mecanismos como los ficheros, certificados de clave pública y biométrica., ***6***

Autorización: Es el proceso de decidir si una identidad digital es permitida para ejecutar una acción de respuesta. La autorización sucede después de la autenticación y usa atributos o derechos, asociados con la identidad digital para determinar a qué recursos puede acceder dicha identidad digital., ***VII***, ***2***, ***27***

B

Base de Datos: Conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente. En una base de datos, la información se organiza en campos y registros., ***4***, ***10***, ***19***, ***20***, ***22***, ***26***, ***32***, ***33***, ***39***, ***40***, ***46***, ***50***, ***55***, ***56***, ***57***

C

Contraseñas: Una contraseña o clave (en inglés password) es una forma de autenticación que utiliza información secreta para controlar el acceso hacia algún recurso. Pueden contener caracteres alfanuméricos e incluso otros símbolos., ***2***, ***6***, ***9***, ***10***

E

Entidad: Representación de un objeto individual concreto del mundo real. Cualquier tipo de objeto o concepto sobre el que se recoge información puede ser una cosa, persona, concepto abstracto o suceso, ***1***, ***47***, ***48***

Escalabilidad: La capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes., ***9***, ***20***, ***46***

I

Interoperabilidad: Capacidad de comunicación entre diferentes programas y máquinas de distintos fabricantes., ***14***

L

LDAP: El Protocolo Ligero de Acceso a Directorios(Lightweight Directory Access Protocol por sus siglas en inglés) permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red., ***6***, ***9***

M

Microsoft: Compañía creadora de los sistemas operativos Windows 95, Windows NT, ***VII***, ***9***, ***12***, ***15***, ***16***, ***17***, ***18***, ***63***

Microsoft Active Directory (Directorio Activo de Microsoft): *Es un servicio de directorio que forma el núcleo de la solución de gestión de acceso e identidades de Microsoft.*, ***9***

Módulo: Componente autocontrolado de un sistema que posee una interfaz bien definida hacia otros componentes, ***VII***, ***2***, ***3***, ***4***, ***10***, ***20***, ***22***, ***23***, ***25***, ***26***, ***27***, ***30***, ***31***, ***32***, ***33***, ***34***, ***35***, ***36***, ***37***, ***42***, ***49***, ***53***, ***55***, ***58***, ***60***, ***61***, ***62***

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Una plataforma es una combinación de hardware y software usada para ejecutar aplicaciones, ***19***

S

Servidor: Un servidor es una computadora que maneja peticiones de data, email, servicios de redes y transferencia de archivos de otras computadoras (clientes). También puede referirse a un software específico. Una computadora puede tener distintos software de servidor, proporcionando muchos servidores a clientes en la red., ***10***, ***18***, ***27***, ***43***

T

Transacciones: Una transacción es un conjunto de procesos que se ejecutan uno después del otro. Si algún subproceso falla, lo realizado anteriormente debe reversarse para que los datos no se alteren, a este comportamiento se lo denomina todo o nada., ***1***, ***19***

U

Usuario: Persona que tiene una cuenta en una determinada computadora por medio de la cual puede acceder a los recursos y servicios que ofrece una red., ***VII***, ***1***, ***2***, ***5***, ***6***, ***7***, ***8***, ***9***, ***10***, ***19***, ***20***, ***22***, ***23***, ***25***, ***27***, ***40***, ***41***, ***47***, ***48***, ***49***, ***51***, ***52***, ***55***, ***56***, ***57***, ***61***, ***62***