Universidad de las Ciencias Informáticas

Facultad 1

SISTEMA DE APROVISIONAMIENTO DE CUENTAS DE USUARIO PARA LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores

Lianet Suárez Martínez

Pablo David Gago Ballester

Tutor

Ing. Javier Heredia Ruíz

Ing. Yendry Machado García

La Habana, Junio de 2014

“Año 56 de la Revolución”

"Un hombre que no arriesga nada por sus ideas, o no valen nada sus ideas, o no vale nada el hombre."

Platón

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo, y autorizamos al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas a hacer uso de forma exclusiva del mismo. Para que así conste, firmamos la presente tesis a los ____ días del mes de _____ del año _____.

_____ _____

Autor Lianet Suárez Martínez

Autor Pablo David Gago Ballester

_____ _____

Tutor Ing. Yendry Machado García

Tutor Ing. Javier Heredia Ruíz

AGRADECIMIENTOS

Agradezco con la vida a mi mamita que tanto se ha sacrificado, que siempre me apoya, me saca de mis tormentos, que me ha dado una excelente educación, me complace, me inspira, me da confianza, es mi gran amiga y por sobre todas las cosas me brinda muchísimo amor. A mi abuela, mi “mimita” linda. Al comienzo no tenía palabras para describir tanta maravilla pero alguien me ayudo, ella es una súper abuela, es especial. Le agradezco por haberme mimado, por hacer esas comidas tan ricas y por darme tanto amor. A mi papito, ese que me dormía en su espalda cuando chiquita, al que le debo gran parte de la cultura y conocimientos que hoy tengo, que también se ha sacrificado mucho por mí y que me ha complacido bastante en mis antojos. A mis primas, tías, a esos que ya no están físicamente, a toda mi familia por darme aliento.

A mi eterna e incondicional amiga, Elizabeth Vento (Tata), ella sabe la confianza y el cariño que le tengo y yo sé que siempre estará presente para lo que necesite, como mismo estaré yo para ella. Mis también eternas amigas del IPI, con las que conviví tres años muy importantes y siempre las tendré presente como un gran tesoro que la vida puso en mi camino.

A Yaiselis (Monga) por adorarme tanto, por ser una de las personas más sinceras que conozco, por demostrar ser una gran persona y muy buena amiga, y por soportarme durante los cinco años con mis estresantes “matraquillas”. A Yander por ser un fiel amigo que está ahí para lo que necesite, por estar conmigo en los momentos más difíciles, darme valiosos consejos y ayudarme, porque detrás de ese jodedor que habla un poquito de boberías hay una hermosa persona que casi nadie se imagina.

A Eliena que me obligaba a ir a almorzar a las once y media todos los días de este quinto año y por suerte no me pegó su locura. Arailis que es única y con su locura me ha hecho reír mucho. Misleydis que cada mañana esperaba por mí y me apuraba para ir al aula. A todos mis compañeros actuales y los anteriores.

A mi tutor Yendry Machado que se ha comportado excelentemente y no tengo como agradecerle. A Javier Heredia que a pesar de habernos tutorado un poquito más tarde, nos acogió con gran dedicación. A Yudenia Ramírez que también me ayudó inicialmente. A los tres, gracias por ser grandes profesionales.

A todos los profesores que han participado en mi formación, especialmente a tres maravillosos ejemplos que han marcado mi educación: Elsa Márquez, Alién García y Joel Arencibia.

A mi novio Reynaldo que tantas cosas de la tesis me ha criticado y tanto me ha fastidiado con el objetivo de que todo quede bien y nada malo suceda.

Agradecimientos especiales a mi compañero de tesis por ser tan responsable, por trabajar cada día sin descanso, por ser perfeccionista, por ser un excelente ingeniero y por ser el causante de que mi estrés con la tesis haya sido mucho menor de lo esperado.

DEDICATORIA

Dedico este trabajo a mi mamá, a mi abuela materna y mi papá por ser mis grandes maestros de la vida, por haber contribuido con su paciencia y apoyo a encaminarme como ser humano y por su lucha constante para hacer realidad mis sueños.

RESUMEN

La gestión de las cuentas de usuario en la Universidad de las Ciencias Informáticas, actualmente no cuenta con un sistema que la automatice desde el momento en que se solicitan dichas cuentas. Por esta razón se realiza todo el proceso de aprovisionamiento de cuentas de usuario de forma manual, ocasionando entre otros inconvenientes, que los usuarios no puedan desempeñar sus labores en un período considerable de tiempo, por lo que se afecta la productividad de la universidad. La presente investigación tiene como objetivo el desarrollo de un sistema que permita resolver esta problemática, gestionando el ciclo de vida de las cuentas de usuario. Para cumplir este objetivo, se estudiaron los sistemas de aprovisionamiento de cuentas de usuario más destacados a nivel mundial y se realizó un análisis la metodología, las tecnologías, lenguajes y herramientas necesarias para la construcción del sistema, destacándose el lenguaje Python con su marco de trabajo Django para la implementación del sistema del lado del servidor, XP como metodología que guía todo el proceso de desarrollo de software y PostgreSQL como sistema de base de datos para lograr la persistencia de la información.

Palabras clave: aprovisionamiento, cuentas de usuario, productividad, solicitud.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La subsistencia y competitividad de las entidades depende en gran medida de su capacidad para adaptarse a las versátiles exigencias del mercado. Del mismo modo en que estas entidades se transforman, muchas veces de forma acelerada, sus recursos humanos también varían. Estas transformaciones obedecen no sólo al buen uso que se haga de los recursos y procesos informáticos, es necesaria también la adecuada administración de los servicios con los que se cuenta. Para ello se hacen necesarias las identidades digitales, las cuales constituyen una representación de las características fundamentales de un individuo dentro de un entorno informático, y que se materializan por medio de cuentas de usuario como mecanismo de acceso a los servicios y sistemas de una entidad (1).

La gestión de las cuentas de usuario es una de las actividades más engorrosas de la administración de redes, implica que los administradores se esfuercen en cumplir con las peticiones que reciben para crear, modificar y/o eliminar cuentas de usuario y sus respectivos accesos a los distintos recursos de la red. Realizar este proceso de forma manual afecta en gran medida la productividad de las entidades. Es por esta razón que surgen los sistemas de administración de identidades que engloban áreas de negocio que se dedican a la implantación de sistemas de autenticación, autorización, auditoría y aprovisionamiento de cuentas de usuario (2). Esto último se refiere al proceso de proveer usuarios, grupos y otros objetos con acceso a aplicaciones y recursos que pueden estar disponibles en una entidad (3), y constituye el mecanismo encargado de la gestión del ciclo de vida de las cuentas de usuario.

Herramientas como Tivoli, Oracle Identity and Access Management Suite, WSO2 Identity Server, ForgeRock OpenIDM y Ping Federate, son algunas de las líderes a nivel mundial en el aprovisionamiento de cuentas de usuario, contando con numerosas funcionalidades, arquitecturas modulares y flexibles, tecnologías novedosas, alta calidad y seguridad. Aunque estas herramientas son potentes productos, la mayoría son privativas o no se ajustan a necesidades específicas de la Universidad de las Ciencias Informáticas (UCI).

Debido al crecimiento de la Infraestructura Tecnológica (IT) cubana, toma fuerza la necesidad de automatizar los procesos de aprovisionamiento de cuentas de usuario, y aunque aún su uso es insuficiente, desde 2009 la UCI se enfoca en la realización de proyectos de este tipo con el fin de satisfacer los requerimientos de la propia universidad. Precisamente, el Centro de Identificación y Seguridad Digital (CISED) es el encargado de esta tarea, y ha desarrollado herramientas que no logran ser completamente funcionales por problemas de flexibilidad; además, existe incompatibilidad con el proceso de migración a software libre en el que se encuentra inmerso todo el país. Por esta razón, se realizan intercambios con la Dirección de Redes y Servicios Telemáticos de la UCI, y se constata que desde su creación, la universidad realiza el proceso de solicitud de cuentas de usuario de forma manual, y el aprovisionamiento en los diferentes recursos se hace individualmente debido a que no existe un sistema que permita la centralización

de todo este proceso. Este mecanismo conduce a largos períodos de espera hasta que los solicitantes puedan desempeñar sus labores, y provoca que las cuentas creadas contengan configuraciones incorrectas en los accesos a los diferentes recursos. De forma general, se afectan a gran escala, la productividad y la seguridad de la propia universidad.

A partir de la situación antes planteada surge el siguiente **problema de investigación**: ¿Cómo agilizar el proceso de aprovisionamiento de cuentas de usuario en la Universidad de las Ciencias Informáticas?

Sobre la base de lo anterior se define como **objeto de estudio**: El proceso de administración de identidades digitales.

Para darle solución al problema planteado se precisa el **objetivo general** siguiente: Desarrollar una solución informática que permita el aprovisionamiento de cuentas de usuario para la gestión del ciclo de vida de las cuentas de usuario en la Universidad de las Ciencias Informáticas, mediante el uso de tecnologías libres.

Quedando enmarcado como **campo de acción**: El proceso de aprovisionamiento de cuentas de usuario.

Se plantea como **idea a defender**, que el desarrollo de un sistema de aprovisionamiento de cuentas de usuario, agilizará la gestión del ciclo de vida de las cuentas de usuario en la Universidad de las Ciencias Informáticas.

Para cumplir con el objetivo general propuesto se definen las siguientes **tareas de investigación**:

- Caracterización del proceso de aprovisionamiento de cuentas de usuario y de las soluciones informáticas que existen para soportarlo.
- Definición de metodología, tecnologías, lenguajes y herramientas adecuadas para la construcción de la solución informática.
- Identificación de las necesidades del sistema a desarrollar.
- Planificación y diseño del Sistema de Aprovisionamiento de Cuentas de Usuario (SACU).
- Implementación de la aplicación.
- Validación de la solución a través de pruebas.

El cumplimiento de las tareas de investigación se basó en el uso de **métodos** de investigación científica que permitieron estudiar y comprender a cabalidad el proceso de aprovisionamiento de cuentas de usuario, los utilizados en el desarrollo de la aplicación fueron los métodos teóricos y empíricos.

Teóricos:

- Analítico - sintético: Para analizar la información recopilada sobre el aprovisionamiento de cuentas de usuario y sintetizar los aspectos más importantes.

Empíricos:

- Observación: Para observar de forma exhaustiva el funcionamiento de los sistemas de aprovisionamiento.
- Análisis documental: Para extraer la información necesaria de literaturas especializadas, tanto académicas como empresariales, enriqueciendo de forma sustancial el proceso de investigación.
- Análisis comparativo: Para detectar diferentes características, ventajas y desventajas en herramientas de aprovisionamiento de cuentas de usuario actuales y representativas.

El trabajo de diploma se divide en tres capítulos, los cuales estarán estructurados de la siguiente forma:

- Capítulo 1: **"FUNDAMENTACIÓN TEÓRICA".** Se especifican conceptos de vital importancia para la comprensión del tema de la investigación. Se realiza un estudio del estado del arte de los sistemas de aprovisionamiento de cuentas de usuario existentes. Además, se analizan elementos como la metodología, herramientas, tecnologías y lenguajes de programación a utilizar en la construcción del software.

- Capítulo 2: **"PLANIFICACIÓN Y DISEÑO".** Se propone la solución al problema de investigación. Se define la arquitectura del sistema. Se detallan las historias de usuario (HU), el plan de iteraciones y el plan de entregas. Se definen los patrones de diseño a emplear y también se diseñan las tarjetas Clase-Responsabilidades-Colaboraciones (CRC).

- Capítulo 3: **"CONSTRUCCIÓN Y PRUEBAS".** Se establecen los estándares de codificación a utilizar en la implementación de la aplicación. Se describen las tareas de programación correspondientes a cada HU. Se obtienen elementos como el diagrama de despliegue y las interfaces gráficas de usuario. Por último, se muestran las pruebas realizadas al sistema y el resultado de las mismas.

Capítulo 1
FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan características y conceptos importantes relacionados con los sistemas de aprovisionamiento de cuentas de usuario. Se presenta una panorámica general de las tendencias y características actuales de dichos sistemas. Se describen los lenguajes y tecnologías a emplear para la implementación, así como la metodología y las herramientas para la modelación e implementación.

1.1. Identidad digital.

La identidad digital es un conjunto de elementos que una o varias personas tienen en un entorno informático y que los distinguen de los demás (4). De igual manera, puede ser utilizada para una gran variedad de funciones en dependencia de la combinación de IT con que cuente la organización. Algunas de estas funciones son: acceso a recursos de la red como correo electrónico, mensajería instantánea, directorios de usuarios y aplicaciones para gestionar los procesos de negocio de la entidad (5).

De manera general, una identidad digital le brinda al usuario la posibilidad de identificarse y formar parte de los recursos humanos de una entidad desde el punto de vista digital. Para mantener el control sobre estas identidades digitales, y con el fin de lograr el cumplimiento de los diferentes objetivos de una entidad, se hace necesaria su administración.

1.2. Administración de identidades digitales.

La administración de identidades es la combinación de procesos de negocio y tecnologías que se utiliza para administrar los datos en los sistemas informáticos y aplicaciones sobre los usuarios (5). Por otro lado, la administración de identidades engloba áreas de negocio que se dedican a las siguientes tareas (2):

- Aprovisionamiento automatizado de cuentas de usuario: Por medio del aprovisionamiento se gestiona el ciclo de vida de las cuentas de usuario.
- Sistemas de autenticación: Se comprueba que cada usuario sea quien dice ser. Este proceso se lleva a cabo por medio de la validación de las credenciales de usuario.
- Sistemas de autorización: Permiten controlar el acceso de los usuarios a los diferentes recursos organizacionales.
- Sistemas de auditoría: Se utiliza para registrar la actividad que ocurre dentro de la infraestructura de la organización y en ocasiones realizar acciones en caso de detectarse alguna anomalía.

Una vez explicadas brevemente estas áreas a las que se dedica la administración de identidades digitales, se elabora la siguiente imagen para una mejor comprensión (ver Figura 1).

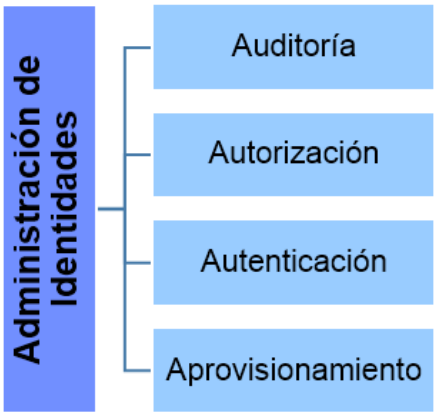


Figura 1: Áreas de la administración de identidades.

Por interés de la actual investigación se señala que los datos administrados incluyen objetos del usuario, atributos de identidad, los derechos y deberes de seguridad, y factores de autenticación. Además, es preciso hacer énfasis en una de las áreas mencionadas anteriormente: el aprovisionamiento de cuentas de usuario, puesto que el desarrollo de un sistema de este tipo constituye el objetivo general de la investigación.

1.3. **Aprovisionamiento de cuentas de usuario**.

El aprovisionamiento de cuentas de usuario garantiza la gestión del ciclo de vida de dichas cuentas, y comprende entre sus tareas fundamentales su creación, modificación y eliminación. Este proceso por lo general implica dos fases (6):

- Fase de aprobación: La creación, modificación o eliminación de cuentas de usuario deben ser autorizados por una o más personas que tienen responsabilidades de gestión (por ejemplo, directivos o supervisores), por lo que la solicitud de la acción a realizar sobre la cuenta debe pasar por un flujo de aprobación.

- Fase de consumación: En caso de aprobación con éxito, esta fase consiste en crear, modificar o eliminar realmente una cuenta de usuario en los recursos a los que éste tiene acceso.

Por medio del aprovisionamiento se le asigna a un usuario acceso a los recursos dentro de una entidad. Un recurso se puede entender como una cuenta de correo, las credenciales para una aplicación o un sistema, o la tarjeta de acceso a un edificio. El proceso de aprovisionamiento comienza cuando desde una aplicación se le brinda al motor de aprovisionamiento la información necesaria para la creación de una cuenta de usuario. Esta información o parte de ella, puede ser extraída de un repositorio de datos o introducida manualmente, el motor la procesa, y posteriormente, a través de conectores, se establecen los enlaces

hacia cada recurso habilitando el acceso de la nueva cuenta de usuario (2). La imagen de la Figura 2 fue elaborada para apreciar mejor lo antes explicado.

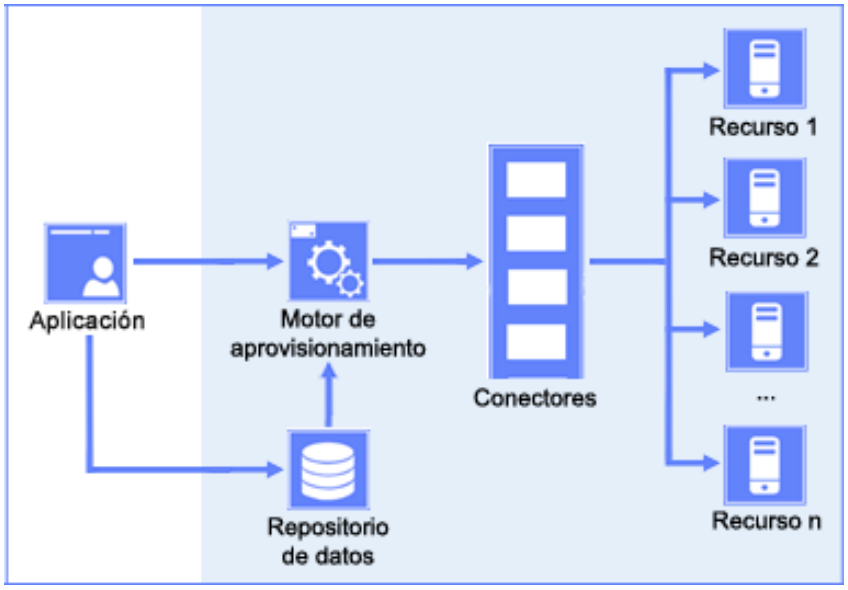


Figura 2: Aprovisionamiento de cuentas de usuario.

Partiendo de las definiciones analizadas anteriormente, se puede definir el aprovisionamiento de cuentas de usuario como la gestión del ciclo de vida de dichas cuentas y la administración de los recursos asociados a los usuarios. Este proceso se vale de componentes esenciales tales como el motor de aprovisionamiento, y los conectores para la interacción con los diferentes recursos de una entidad.

1.4. Soluciones similares.

A raíz de una búsqueda bibliográfica en los diferentes medios y en trabajos de diploma desarrollados en la UCI, se encuentran algunas soluciones informáticas para el aprovisionamiento que figuran entre las de mayor demanda a nivel internacional. Teniendo en cuenta el interés de la investigación en las aplicaciones no privativas, se seleccionan al menos dos herramientas de este tipo.

1.4.1. Gestor de Aprovisionamiento Tivoli de IBM 7.2.1.

El Gestor de Aprovisionamiento Tivoli forma parte del administrador de identidades Tivoli de IBM. Está formado por un servidor de suministro, una consola de operación y administración basada en la web, y un entorno de desarrollo de paquetes de automatización (7).

Este gestor de aprovisionamiento dispone de los siguientes componentes básicos (7):

- Servidor de suministro: Es el servidor en el que está instalado el Gestor de Aprovisionamiento Tivoli y que contiene la base de datos de suministro, el modelo de datos, los flujos de trabajo y los scripts; además, genera informes y permite configurar la infraestructura de despliegue.

- Interfaz de servicios web: Mediante los servicios web, incluidos los *Web Services Resource Framework* (WSRF), El Gestor de Aprovisionamiento Tivoli permite acceder, manipular o cambiar objetos del modelo de datos directamente en lugar de iniciar la interfaz web.

- Consola de operación y administración: Permite interactuar con el servidor de suministro del Gestor de Aprovisionamiento Tivoli. Proporciona una representación gráfica de los activos de IT. Incluye asistentes para simplificar la configuración y ofrece otras funciones como la creación de informes y el seguimiento del estado de las tareas, que no están disponibles desde la interfaz de línea de comandos.

- Entorno de desarrollo de paquetes de automatización: Es un *plug-in* para Eclipse que los desarrolladores pueden utilizar para crear o personalizar paquetes de automatización.

- Biblioteca abierta de IBM para el proceso de automatización: Es una biblioteca compartida para la automatización de procesos. Se trata de un completo catálogo en línea que contiene más de quinientas extensiones de productos de Tivoli y sus compañeros de negocio. Estas extensiones incluyen: Paquetes de automatización, adaptadores de integración, agentes, documentación e información de soporte.

- Directorio de usuarios: El Gestor de Aprovisionamiento Tivoli se integra con varios servidores de directorios, lo que le permite gestionar las cuentas de usuario y la autenticación.

- Aplicaciones de gestión externas: El Gestor de Aprovisionamiento Tivoli permite integrarse con aplicaciones externas.

1.4.2. Oracle Identity and Access Management Suite 11g.

La plataforma Oracle Identity Manager (OIM) automatiza la gestión de los derechos de acceso, seguridad y aprovisionamiento a los recursos de IT. Conecta las cuentas de usuario a los recursos, revoca y restringe el acceso no autorizado para proteger la información confidencial de la entidad (8).

OIM posee una arquitectura modular, flexible y escalable. Utiliza flujos de trabajo para definir el proceso de aprovisionamiento, empleando políticas de seguridad. Este sistema posee además, un amplio conjunto de conectores y proporciona tecnologías de conector genérico que permiten la fácil y rápida creación de un conector nuevo (8).

OIM está compuesto por cuatro componentes o módulos. A continuación se muestra una breve descripción de cada uno (9):

- Gestor de aprovisionamiento: Ensambla y modifica las transacciones de aprovisionamiento. Los perfiles de usuario, las políticas de acceso y los recursos, se definen a través de este componente, así como los flujos de trabajo de procesos y reglas de negocio.

- Servidor de aprovisionamiento: Es el motor de tiempo de ejecución de OIM que ejecuta las operaciones del proceso de aprovisionamiento.

- Adaptador de fábrica: Construye y mantiene las integraciones de OIM con los sistemas y aplicaciones administradas.

- Motor de reconciliación: Descubre cuentas ilegales creadas fuera de OIM. El motor de la reconciliación también sincroniza las reglas de negocio establecidas dentro y fuera del sistema de aprovisionamiento para garantizar la coherencia.

1.4.3. WSO2 Identity Server 4.5.0.

WSO2 Identity Server forma parte de una amplia gama de productos que brinda WSO2. Es un sistema creado para agilizar la gestión de cuentas de usuario y los accesos de estas últimas a los recursos de la entidad (10).

Entre las características fundamentales de WSO2 Identity Server se encuentran las siguientes (10):

- Sistemas y gestión de la identidad del usuario:

  - Implementa un almacén de usuarios flexible.

  - *Application Programming Interface* (API) para la integración de la gestión de identidad a cualquier aplicación.

  - Aprovisionamiento a través de *System for Cross-domain Identity Management* (SCIM) en lugar de *Service Provisioning Markup Language* (SPML) legado.

  - Asignación de credenciales a través de diferentes protocolos.

- Gestión de derechos:

  - Control de acceso basado en roles (RBAC).

  - Control de acceso basado en políticas a través de *eXtensible Access Control Markup Language* (XACML).

  - Gestión de derechos para todas las llamadas *Simple Object Access Protocol* (SOAP).

- Administración y control:

  - Consola web de gestión integral y monitoreo de la seguridad a nivel empresarial.

  - Soporte flexible de registro con la integración de sistemas empresariales.

Este servidor de identidades no sólo aprovisiona usuarios, sino que soporta los recursos y servicios a los que se conecta. Para ello emplea el estándar XACML. Cuando un usuario solicita acceso a un recurso, se genera una petición con la información de la solicitud. Con esta información se busca dentro del XACML la política que determinará si el usuario tiene o no los privilegios para acceder al recurso solicitado, devolviendo un *Permit* (solicitud aceptada) o *Deny* (solicitud denegada). Este modo de funcionamiento de Identity Server trae consigo desventajas tales como la actual inexistencia de una interfaz para definir y administrar políticas XACML, y la dificultad para lograr alta disponibilidad y confianza, pues en caso de corromperse el sistema, todos los recursos y servicios conectados a él serían inoperables (11).

1.4.4. ForgeRock OpenIDM 2.1.0.

ForgeRock OpenIDM permite la automatización de la gestión del ciclo de vida de las cuentas de usuario y de los privilegios de acceso a aplicaciones locales o basadas en la nube. Proporciona un marco flexible en el que componentes y servicios se pueden agregar o quitar a voluntad. Este sistema provee un conjunto de conectores para una serie de sistemas como bases de datos, directorios y sistemas operativos. El marco también se puede ampliar con conectores adicionales para apoyar integraciones personalizadas hacia otras aplicaciones (12).

A continuación se listan algunas de las prestaciones más recientes de ForgeRock (13):

- Plataforma genérica para exponer e invocar flujos de trabajo y métodos Java.

- Servicio de creación de reglas utilizando JavaScript.

- Servicio de notificación para las tareas.

- Servicio de programación de tareas configurable.

- Servicio de políticas de contraseñas configurable.

- Servicio de comprobación de correcto aprovisionado y correcto acceso a los recursos.

En estos momentos OpenIDM presenta algunos problemas que están pendientes a resolver, el sitio oficial de ForgeRock OpenIDM muestra un listado que asciende a catorce dificultades (13).

1.4.5. Aprovisionamiento automático de usuarios de Ping Federate 7.1.0.

Ping Federate es uno de los productos que ofrece Ping Identity. Tiene como propósito permitir y proteger la identidad, defender la privacidad y la seguridad de Internet (14).

Las principales características de Ping Federate se muestran a continuación (14):

- Gestión de identidad federada: Utiliza almacenes de identidades existentes en otras entidades para la autenticación.
- Acceso móvil seguro: Asegura el acceso de usuarios desde cualquier dispositivo móvil, utilizando estándares abiertos.
- Integración de la Identidad Social: Incrementa la captación de clientes, permitiendo a los consumidores iniciar sesión con sus credenciales de Google, Yahoo, Twitter, Facebook y Windows Live.
- Provisión automatizada de usuarios: Aprovisiona los usuarios de los servicios de la nube empresarial de forma automática. Utiliza el estándar SCIM para automatizar la entrada y salida de usuarios a los directorios corporativos y aplicaciones Software *as a Service* (SaaS).

1.4.6. Subsistema de Aprovisionamiento de cuentas de usuario para el Sistema de Administración de Identidades v1.0.

El Subsistema de Aprovisionamiento del Sistema de Administración de Identidades v1.0, surgió con el objetivo de gestionar las cuentas de los usuarios en el Ministerio del Interior (MININT); para lo cual se definieron los siguientes módulos: Portal del Usuario, Administración, Alertas y Notificaciones, Gestor de Conectores, Reportes y Motor de Tareas. De los módulos antes mencionados fueron desarrollados solamente tres: El módulo de Administración, que gestiona las cuentas de usuario en el Subsistema de Aprovisionamiento del Sistema de Administración de Identidades (15); el Motor de Tareas, que gestiona de manera automática los procesos de aprovisionamiento de cuentas de usuario mediante flujos de aprobación en los sistemas y aplicaciones suscritos al Sistema de Administración de Identidades (16); y el módulo de Conectores, que provee conectores para acceder a los recursos que utiliza el Subsistema de Aprovisionamiento de cuentas de usuario (17).

El módulo Motor de Tareas no cuenta con una interfaz gráfica para la administración de varias funciones como la revisión temporal de los repositorios. El motor aprovisiona usuarios solamente en OpenLDAP,

haciendo que el componente no satisfaga las necesidades debido a que se necesita aprovisionar usuarios en varios recursos y contar con la posibilidad de incorporar nuevos a la aplicación.

El módulo de Administración tiene una arquitectura rígida en cuanto al flujo de aprobación de las cuentas de usuario, ya que no dispone de una interfaz gráfica para su gestión; lo mismo sucede con la estructura administrativa de la organización, por lo que en caso de cambios en la misma, el funcionamiento del sistema falla. Además, su funcionamiento está basado en el módulo Motor de Tareas, lo que implica una desventaja adicional.

1.4.7. Subsistema de Aprovisionamiento de cuentas de usuario para el Sistema de Administración de Identidades v2.0.

El Subsistema de Aprovisionamiento de cuentas de usuario permite la gestión dinámica de las cuentas de usuario y los recursos asociados de forma genérica (18).

El sistema cuenta con un módulo llamado Portal de Usuario, que muestra todas las opciones a las que el usuario estándar tiene permisos; otro módulo llamado Motor de Tareas, basado en servicios web y del sistema operativo Windows, brinda una serie de funcionalidades para el aprovisionamiento de cuentas de usuario en los diferentes recursos; por último, el módulo Portal de Administración, que permite la configuración y gestión del sistema en general.

Este sistema posee algunos problemas de flexibilidad, ejemplo de ello es el hecho de que el flujo que deben seguir las solicitudes es estático, no se acopla a la estructura administrativa de la organización y no toma información de fuentes de datos externas.

1.4.8. Resumen de limitaciones.

Una vez analizados los principales sistemas de aprovisionamiento de cuentas de usuario a nivel mundial, es posible apreciar su elevada calidad, todos son parte de una suit más amplia que abarca la administración de identidades de forma compacta, es decir, en una sola solución. Es notable también que estos sistemas están destinados a lugares donde la forma de comunicación entre los diferentes elementos de la IT, es totalmente diferente a la existente en la UCI; ejemplo de lo antes mencionado es la utilización de servicios basados en la nube, tecnología que aún en Cuba, no ha tomado auge.

Por otra parte, los sistemas de aprovisionamiento desarrollados hasta el momento en la UCI tienen como inconveniente principal la falta de flexibilidad y adaptabilidad a condiciones y procesos cambiantes. También fueron desarrollados con herramientas, lenguajes y tecnologías que son privativas, lo que descarta la opción de realizarles mantenimiento para adaptarlos a las necesidades actuales de la universidad.

La siguiente tabla muestra una comparación de los sistemas analizados, permitiendo visualizar de forma unificada algunas desventajas o limitaciones que impiden la utilización o modificación de algún sistema de aprovisionamiento de cuentas de usuario existente. En el caso de la variable “Aprovisionamiento local”, hace referencia a si el sistema soporta aprovisionamiento local (que no depende de la nube) y la variable “Flujo gestionable” se refiere a si el sistema es capaz de gestionar los flujos a seguir por las solicitudes de cuentas de usuario. Existen campos en la Tabla 1 que contienen un guion, debido a que no han podido ser llenados por no contar con la infomación necesaria.

Aspecto / Producto	Software Libre	Gratis	Aprovisionamiento local	Flujo gestionable
Gestor de Aprovisionamiento Tivoli de IBM 7.2.1.	NO	NO	-	-
Oracle Identity and Access Management Suite 11g.	NO	NO	SI	SI
WSO2 Identity Server 4.5.0.	SI	SI	SI	-
ForgeRock OpenIDM 2.1.0.	SI	NO	SI	-
Aprovisionamiento automático de usuarios de Ping Federate 7.1.0.	NO	NO	NO	-
Subsistema de Aprovisionamiento de cuentas de usuario para el Sistema de Administración de Identidades v1.0.	NO	NO	SI	NO
Subsistema de Aprovisionamiento de cuentas de usuario para el Sistema de Administración de Identidades v2.0.	NO	NO	SI	NO

Tabla 1: Comparación de los sistemas de aprovisionamiento estudiados.

1.5. Metodología, tecnologías, lenguajes y herramientas.

1.5.1. Metodología XP.

XP (*Extreme Programming*, en español, Programación Extrema) es una metodología ágil centrada en potenciar el trabajo en equipo. XP propone relativa simplicidad en las soluciones implementadas y como ágil que es, plantea que el software que funciona es la medida principal de progreso, no la documentación generada (19).

Esta metodología es apropiada para equipos de desarrollo pequeños, donde el cliente es elemento clave e imprescindible dentro del mismo, debiendo estar presente en todo momento. El cliente incluso, genera gran parte de algunos artefactos, que unidos al diálogo constante entre él y los desarrolladores conforman todo lo que los programadores deben saber para escribir el código del software a desarrollar (19).

Es abierta a los cambios y destinada a proyectos con requerimientos cambiantes hasta último momento. Apuesta por una serie de prácticas que tienen como objetivos simplificar el trabajo y disminuir el costo derivado de los continuos cambios. Entre los artefactos fundamentales de XP se encuentran las HU, el plan de iteraciones, el plan de entregas, las tarjetas CRC, las tareas de programación, las pruebas unitarias y las pruebas de aceptación (19).

Por acoplarse a las características y necesidades del equipo de desarrollo y el cliente, se selecciona esta metodología como guía de todo el proceso de desarrollo del software.

1.5.2. Lenguaje de Programación Python 2.7.3.

Python es un lenguaje de programación creado por Guido Van Rossum a principios de los años noventa cuyo nombre está inspirado en el grupo de cómicos ingleses "*Monty Python*". Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de *script*, con tipado dinámico, fuertemente tipado, multiplataforma, orientado a objetos y que además, es de código abierto (20).

El lenguaje de programación Python es de alto nivel, aunque permite modificar bits y bytes. Permite dividir el programa en módulos reutilizables desde otros programas. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python) (21).

Python es el lenguaje seleccionado para la programación del lado del servidor, proporcionando la rapidez y robustez necesarias para un sistema como el que se desea desarrollar.

1.5.3. HTML 5.

HTML (*Hyper Text Markup Language,* en español, Lenguaje de Marcado de Hipertexto) es un lenguaje universal para el desarrollo de sitios web que permite describir hipertexto presentando el texto de forma estructurada y agradable. Es admitido por todos los navegadores, aunque el aspecto de las páginas depende del tipo de ordenador, el monitor, la velocidad de la conexión de Internet y, por último, del navegador (22).

Haciendo uso de este lenguaje se construirán todas las páginas de interfaz de usuario que serán mostradas al usuario final del sistema.

1.5.4. XML 1.0.

En 1996, el *World Wide Web Consortium* (W3C) comenzó a desarrollar un nuevo lenguaje de marcado estándar que fuera sencillo de utilizar, pero con una estructura más rígida que la de HTML, con lo cual se

logran remediar los defectos fundamentales de HTML. XML (*Extensible Markup Language*, en español, Lenguaje de Marcado Extensible) se encuentra conformado por un conjunto de reglas para definir etiquetas semánticas orientadas a organizar un documento en diferentes partes. Es un lenguaje de marcado ligero, flexible y que se puede utilizar para documentos internacionales, siendo ideal para almacenar datos y para enviar mensajes. Los documentos XML son extensibles y se pueden validar. Sus componentes más básicos son los elementos, atributos y comentarios (23).

Toda la información que será almacenada en la base de datos tendrá un formato XML definido previamente, a fin de facilitar en caso de que sea necesario, la alteración de la estructura que contiene esta información.

1.5.5. XPath 1.0.

XPath es la abreviación de lo que se conoce como XML *Path Language*. Constituye un mecanismo para la selección de información dentro de un XML. Con XPath es posible seleccionar y hacer referencia a textos, elementos, atributos y cualquier otra información contenida dentro de un fichero XML. La forma en que XPath selecciona partes del documento XML se basa en la representación arbórea que se genera de un documento (24).

Por medio de este lenguaje es posible la extracción rápida de la información almacenada en la base de datos del sistema a desarrollar.

1.5.6. JavaScript 1.2.

JavaScript es un lenguaje ligero, interpretado y del lado de cliente. Su propósito general es proporcionar funciones básicas que son soportadas por la mayoría de los navegadores. Permite dar dinamismo a páginas web con contenido HTML estático, pudiendo incluir programas que interactúan con el usuario, controlar el navegador, o dinámicamente crear contenido HTML (25). Se puede hacer cargo de gran parte de las funciones del cliente, de las cuales, antiguamente se encargaba el servidor. Una de sus funciones más empleadas es la validación (26).

JavaScript será el lenguaje encargado de otorgar dinamismo a las páginas HTML estáticas, brindando además la posibilidad de realizar peticiones al servidor de manera asincrónica.

1.5.7. CSS 3.0.

CSS (*Cascading Style Sheets*, en español, Hojas de Estilo en Cascada) se define como un conjunto de reglas en las que se asocian las propiedades estilísticas de un documento y sus respectivos valores, expresando así la forma de representarlo visualmente. Las hojas de estilo constituyen las herramientas de

los diseñadores con la que garantizan la coherencia de la presentación: los colores, las fuentes y el diseño de forma general (27).

A través de CSS será establecida la apariencia de las diferentes interfaces gráficas de usuario presentes en el sistema.

1.5.8. Marco de trabajo Django 1.6.

Django es un marco de trabajo de desarrollo para la web. Su implementación es totalmente sobre Python. Con este marco de trabajo se pueden crear y mantener aplicaciones de alta calidad. Incluye un servidor web ligero que se puede usar mientras se desarrolla. Al mismo tiempo, Django permite trabajar fuera su ámbito según sea necesario (28).

Django ofrece las siguientes facilidades (28):

- Sistema de plantillas para separar la presentación de un documento de sus datos.
- Construcción automática de interfaces de administración.
- Vistas genéricas que recogen ciertos estilos y patrones comunes en su desarrollo y los abstraen, de modo que se puede escribir rápidamente vistas comunes de datos sin tener que escribir mucho código.
- Sistema de caché robusto y con un nivel de granularidad ajustable, que permite guardar páginas dinámicas para que no tengan que ser recalculadas cada vez que se piden.
- Integración con bases de datos y aplicaciones existentes.
- Construcción de aplicaciones multilenguaje permitiendo especificar cadenas de traducción de más de cuarenta idiomas.

Basados en estas características, se escoge este marco de trabajo para la implementación del sistema, pues se centra en el desarrollo rápido, la reutilización y la seguridad.

1.5.9. Marco de trabajo JQuery 1.9.2.

JQuery es un marco de trabajo para el lenguaje JavaScript. Ofrece una infraestructura con la que se obtiene mayor facilidad para la creación de aplicaciones complejas del lado del cliente. La sencillez de su sintaxis y la poca extensión del código que se necesita escribir son sus características más notables. JQuery simplifica la manera de trabajar al crear interfaces de usuario, interactuar con documentos HTML, enriquecer aplicaciones web con efectos dinámicos, hacer uso de Ajax, manipular el árbol DOM y garantizar la correcta visualización de las interfaces web desde cualquier navegador (29).

Con la utilización de este marco de trabajo será posible realizar funcionalidades y validaciones necesarias para garantizar el funcionamiento correcto del sistema.

1.5.10. Entorno de Desarrollo Integrado Eclipse 4.2.1.

Eclipse es una plataforma que ha sido diseñada desde el principio para la construcción de aplicaciones integrando herramientas de desarrollo. Por su diseño, la plataforma no proporciona una gran cantidad de funcionalidades para el usuario final por sí misma, sin embargo define una arquitectura abierta para que infinitos módulos o *plug-ins* puedan instalarse permitiendo a los desarrolladores centrarse en su área de especialización. Se considera un Entorno de Desarrollo Integrado que proporciona herramientas que facilitan el desarrollo de aplicaciones, permitiendo compilarlas, correrlas y depurarlas. Provee además herramientas para administrar el espacio de trabajo y control de versiones sobre el código fuente (30).

Este entorno de desarrollo permitirá escribir el código fuente del sistema, al mismo tiempo que consume una mínima cantidad de recursos de hardware y software.

1.5.11. Sistema de bases de datos PostgreSQL 9.2.4.1.

PostgreSQL es un sistema de base de datos objeto-relacional. Cuenta con una arquitectura probada en cuanto a fiabilidad e integridad de la información. Es multiplataforma y totalmente compatible con ACID (*Atomicity, Consistency, Isolation and Durability*, en español Atomicidad, Consistencia, Aislamiento y Durabilidad). Tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios idiomas). Soporta el almacenamiento de grandes objetos binarios, como imágenes, sonidos o vídeos. Dispone de interfaces de programación nativas para C/C++, Java, .Net, Perl, Python, Ruby, ODBC, entre otros. Es altamente escalable tanto en la enorme cantidad de datos que puede manejar como en el número de usuarios concurrentes que puede soportar (31).

Esta herramienta constituirá el soporte para la base de datos que almacenará toda la información manejada por el sistema a desarrollar.

1.5.12. UML 2.1.

UML (*Unified Model Language*, en español, Lenguaje Unificado de Modelado) es un lenguaje para modelar sistemas de información, su objetivo es lograr modelos que, además de describir con cierto grado de formalismo dichos sistemas, puedan ser entendidos por los clientes o usuarios de aquello que se modela (32).

Este lenguaje permitirá la representación de los distintos modelos generados durante todo el proceso de desarrollo del sistema.

1.5.13. Herramienta de modelado Visual Paradigm for UML 8.0.

Visual Paradigm for UML es una herramienta CASE (Computer Aided Software Engineering, en español, Ingeniería de Software Asistida por Ordenador) que soporta el ciclo de vida completo de software: análisis y desarrollos orientados a objetos, construcción, prueba y despliegue. Permite diseñar diferentes diagramas de clases, realizar ingeniería inversa, generar código a partir de diagramas y generar documentación (33).

Haciendo uso de esta herramienta se obtendrán algunos de los artefactos propuestos por la metodología seleccionada.

1.6. Conclusiones.

La utilización de los métodos de investigación científica definidos en la introducción de la actual investigación, permitió estudiar y comprender a fondo el proceso de aprovisionamiento de cuentas de usuario. También permitió realizar un análisis de los sistemas de este tipo, de mayor calidad a nivel mundial, nacional y en el entorno de la UCI. Después de realizados estos estudios, se llegó a la conclusión que es preciso desarrollar un nuevo sistema.

Luego del estudio del estado del arte se decidió emplear como metodología para guiar el proceso de desarrollo del software: XP por ser ágil y a su vez sólida y flexible. Se estableció que la solución fuese un software libre, por lo que las herramientas y tecnologías empleadas en su construcción también deben serlo. Como lenguaje de programación se eligió Python por ser potente y posibilitar la creación de una aplicación robusta; auxiliándose de la valiosa ayuda que proporciona el marco de trabajo Django que agiliza el desarrollo de aplicaciones web en Python. El Entorno de Desarrollo Integrado que se utilizará para el desarrollo del sistema será Eclipse. Para lograr la persistencia de la información se seleccionó el sistema de base de datos PostgreSQL, que brinda una alta fiabilidad e integridad de los datos.

Capítulo 2
PLANIFICACIÓN Y DISEÑO

En este capítulo se presenta la propuesta de solución al problema de investigación planteado al inicio del trabajo. Se definen los requerimientos funcionales y no funcionales del sistema, que son descritos a través de las HU. A partir de la planificación de las iteraciones se confecciona el plan de entregas. Se muestra la selección de patrones de diseño que se deben emplear en la implementación de cada requerimiento y finalmente se diseñan las tarjetas CRC que exponen las responsabilidades y colaboraciones de las clases que componen el sistema.

2.1. Propuesta de solución.

Para darle solución al problema de investigación identificado se propone el desarrollo de una aplicación para el aprovisionamiento de cuentas de usuario que garantice la automatización del proceso de gestión del ciclo de vida de las cuentas de usuario. El SACU permitirá realizar solicitudes de cuentas de usuario desde cualquier lugar de la universidad. Las solicitudes contarán con una serie de parámetros gestionables que conformarán la información necesaria para la creación de las cuentas de usuario. Una vez creada la solicitud, estará lista para, de forma secuencial, ser aprobada por las personas autorizadas. Este proceso de aprobación estará regido por un flujo previamente elaborado que define los cargos de la estructura administrativa de la UCI que deben firmar las solicitudes de acuerdo a la plantilla por la que se rige cada solicitud. Los flujos de aprobación contienen además el orden en que cada aprobador debe firmar. Cuando una solicitud llega al final del flujo, significa que está lista para convertirse en una cuenta de usuario, por lo que los administradores podrán crearlas en los recursos definidos en la solicitud. La Figura 3 se ha elaborado para un mejor entendimiento.

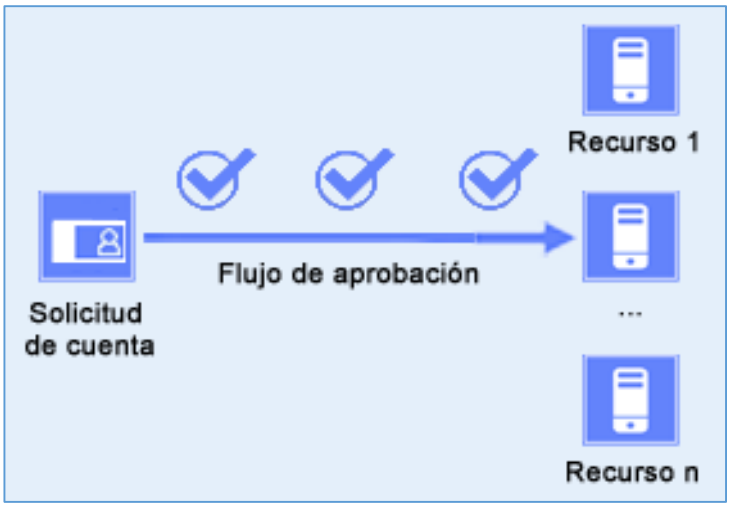


Figura 3: Propuesta de solución.

2.2. Modelo conceptual.

Aunque no es uno de los artefactos que debe generar la metodología elegida, se decidió su elaboración para obtener una representación de los conceptos del dominio del sistema. La elaboración de un modelo conceptual ofrece la ventaja de concentrarse en los conceptos del dominio, no en las entidades del software. Puede mostrar conceptos, sus asociaciones y atributos (34). La Figura 4 muestra el modelo conceptual del sistema objeto de la actual investigación.

Se definen como conceptos del dominio a:

- Plantilla: Elemento que define el contenido de las solicitudes de cuentas de usuario.
- Solicitud de cuenta de usuario: Documento que se crea para solicitar una cuenta de usuario.
- Flujo de aprobación: Define el flujo que deben seguir las solicitudes de cuentas de usuario. Define los aprobadores y el orden de aprobación de dichas solicitudes.
- Aprobador: Persona que firma o rechaza las solicitudes de cuentas de usuario.
- Área: Elemento que compone la estructura administrativa de la UCI.
- Estructura administrativa: Estructura de la UCI que contempla todas sus áreas.
- Cuenta de usuario: Elemento que identifica digitalmente a una persona y le brinda la posibilidad de acceder a los diferentes recursos de la UCI.
- Recurso: En él se aprovisionan las cuentas de usuario.

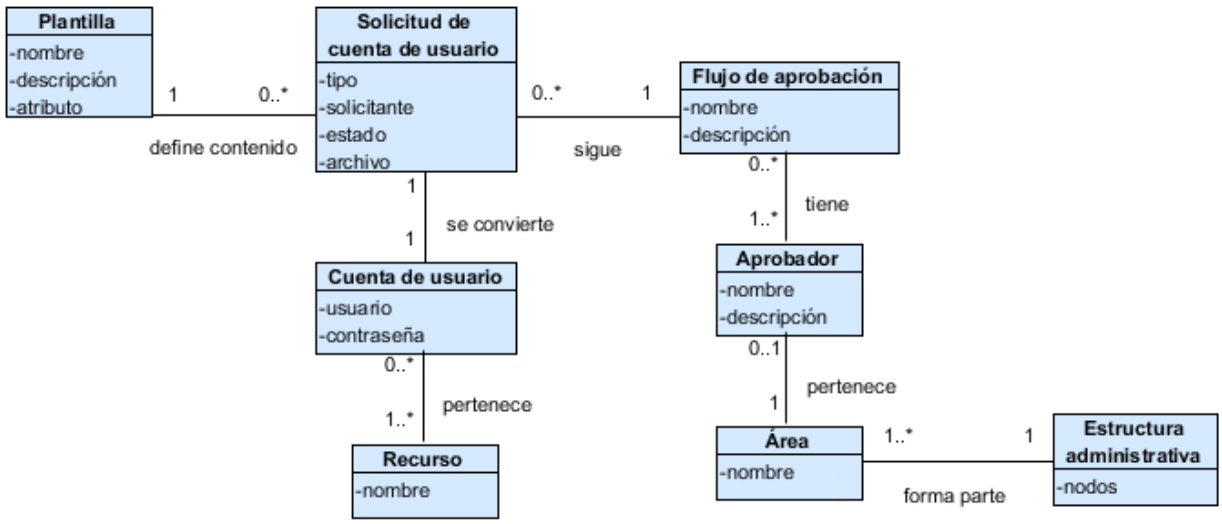


Figura 4: Modelo conceptual del SACU.

2.3. Requerimientos del sistema.

A pesar de que la metodología XP establece que tanto los requerimientos funcionales como los no funcionales deben quedar plasmados en las HU, el equipo de desarrollo ha decidido incluirlos en este apartado para mayor entendimiento y claridad.

2.3.1. Requerimientos funcionales.

Constituyen las funcionalidades que debe realizar el sistema (35). El siguiente listado muestra las correspondientes al SACU.

RF1. Gestionar recursos.

RF1.1. Crear recurso.

RF1.2. Listar recursos.

RF1.3. Editar recurso.

RF1.4. Eliminar recurso.

RF2. Gestionar plantillas.

RF2.1. Crear plantilla.

RF2.2. Listar plantillas.

RF2.3. Editar plantilla.

RF2.4. Eliminar plantilla.

RF3. Administrar estructura de la entidad.

RF3.1. Mostrar estructura de la entidad.

RF3.2. Actualizar estructura de la entidad.

RF4. Definir aprobadores.

RF5. Gestionar flujos de aprobación.

RF5.1. Crear flujo de aprobación.

RF5.2. Listar flujos de aprobación.

RF5.3. Editar flujo de aprobación.

RF5.4. Eliminar flujo de aprobación.

RF6. Administrar solicitudes de cuentas de usuario.

RF6.1. Crear solicitud de cuenta de usuario.

RF6.2. Listar solicitudes de cuentas de usuario.

RF6.3. Realizar seguimiento de solicitud de cuenta de usuario.

RF6.4. Rechazar solicitud de cuenta de usuario.

RF6.5. Aprobar solicitud de cuenta de usuario.

RF6.6. Editar solicitud de cuenta de usuario.

RF6.7. Eliminar solicitud de cuenta de usuario.

RF7. Administrar cuentas de usuario.

RF7.1. Crear cuenta de usuario.

RF7.2. Listar cuentas de usuario.

RF7.3. Listar últimas cuentas de usuario.

RF7.4. Eliminar cuenta de usuario.

RF8. Gestionar grupos de usuarios.

RF8.1. Crear grupo de usuarios.

RF8.2. Listar grupos de usuarios.

RF8.3. Editar grupo de usuarios.

RF8.4. Eliminar grupo de usuarios.

RF9. Autenticar usuario.

RF10. Administrar registros de sucesos del sistema.

RF10.1. Listar registros de sucesos del sistema.

RF10.2. Rotar registros de sucesos del sistema.

RF11. Buscar información en un listado de datos.

2.3.2. Requerimientos no funcionales.

Los requerimientos no funcionales son aquellos que no se refieren directamente a las funcionalidades específicas que proporciona el sistema, sino a sus propiedades emergentes. Pueden derivarse de las características requeridas del software (requerimientos del producto), de la organización que desarrolla (requerimientos organizacionales) o de fuentes externas (35). A continuación se muestra la selección de estos requerimientos para el SACU.

- Requerimientos del producto:
  - El sistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras.
  - El sitio contará con una estructura sencilla y uniforme.
  - El sistema debe estar disponible en idioma español e inglés.
  - Del lado del servidor la computadora empleada debe tener:
    - 2 Gb de RAM.
    - 20 Gb de Disco Duro.
    - Microprocesador a velocidad de 3.3 GHz.
  - Del lado del cliente la computadora empleada debe tener:
    - 256 de RAM.
    - Microprocesador de un núcleo a velocidad de 1.5 GHz.
  - Del lado del servidor debe instalarse:
    - Intérprete de Python versión 2.7.3 o superior, y anterior a la versión 3.3.
    - Marco de trabajo Django versión 1.6.

    - Biblioteca lxml versión 2.3.
    - Biblioteca reportlap 2.6.
    - Biblioteca auth-ldap versión 2.4.9.
    - Biblioteca gettext versión 0.17.
    - PostgreSQL versión 9.2.4.1.
  - Del lado del cliente debe instalarse:
    - Un navegador.

- Requerimientos organizacionales:
  - El sistema debe ser una aplicación web.
  - El sistema debe cumplir con un conjunto de patrones de diseño seleccionados por el equipo de desarrollo.
  - El código debe cumplir con los estándares de codificación definidos por el equipo de desarrollo.
  - Los usuarios deben autenticarse antes de comenzar a interactuar con el sistema.
  - Se deben definir distintos grupos de usuarios de acuerdo a los permisos que estos tengan en el sistema.
  - El SACU debe ser desarrollado empleando herramientas, lenguajes y tecnologías libres.

2.4. Arquitectura del sistema.

Django, como la mayoría de los marcos de trabajo, está diseñado para promover el acoplamiento débil y la estricta separación entre las capas de una aplicación. Está basado en la arquitectura Modelo Vista Controlador (MVC), que separa el acceso a datos (Modelo) de la presentación o interfaz de usuario (Vista), y pone la lógica del negocio como capa mediadora de las dos anteriores (Controlador). El controlador se encarga de aislar el modelo y la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, entre otros). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de bases de datos utilizado por la aplicación (36).

Aunque Django puede llamarse un marco de trabajo MVC, sus desarrolladores consideran que no hacen uso de esta arquitectura al pie de la letra. En su lugar, definen una propia: Modelo Vista Plantilla. Esta arquitectura constituye en realidad una adaptación de la mencionada anteriormente, que en esencia, realiza algunos cambios de conceptos en las responsabilidades de las distintas capas. La capa del modelo, sigue teniendo la misma función de acceso a datos; la capa Vista, a diferencia de su función en MVC, es responsable de aislar toda la lógica del negocio. Esta capa sirve de “puente” entre el modelo y la capa de presentación o interfaz de usuario llamada ahora Plantilla, que por lo general la constituyen ficheros de código HTML y que se conecta con la Vista siempre a través de una URL (28). La Figura 5 fue elaborada para facilitar la comprensión del patrón arquitectónico MVP.

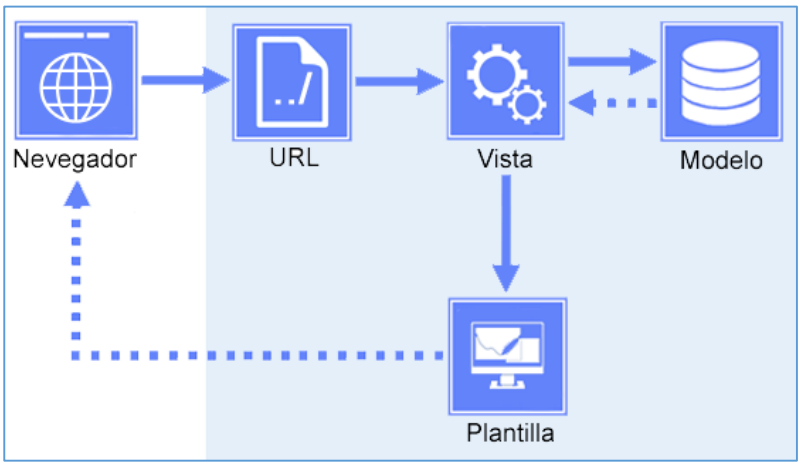


Figura 5: Patrón arquitectónico del SACU.

Es importante mencionar que el SACU posee una estructura de paquetes modular, que permite el bajo acoplamiento entre las aplicaciones que lo componen, respondiendo cada una de éstas al patrón arquitectónico antes descrito.

2.5. Historias de usuario.

Las HU constituyen la forma a través de la cual se especifican los requerimientos funcionales y no funcionales en XP. Su contenido y su nivel de detalle son flexibles, aunque deben ser lo mínimo posible como para que los programadores puedan hacer estimaciones de tiempo y complejidad de desarrollo. Constan también de una descripción escrita por el cliente en un lenguaje no técnico. Cuando llega el momento de la implementación, los desarrolladores dialogan con los clientes para obtener toda la información necesaria (37). En la actual investigación se definieron los siguientes parámetros para las HU en correspondencia con las necesidades del equipo de desarrollo y del cliente:

- Identificador (Id.): Id asignado a la HU.
- Nombre: Nombre de la HU.

- Prioridad técnica: Nivel de prioridad de la HU para los desarrolladores (Alta, Media, Baja).
- Complejidad de desarrollo: Nivel de complejidad técnica que supone desarrollar la HU (Alta, Media, Baja).
- Puntos estimados: Estimación hecha por el equipo de desarrollo del tiempo de duración de la HU (Una semana = 1 punto; dos semanas y cinco días = 2.5 puntos).
- Iteración asignada: Iteración a la que pertenece la HU en el plan de iteraciones.
- Descripción: Breve descripción de lo que realizará la HU.
- Observaciones: Aspectos importantes de interés para el cliente.

En los intercambios con los clientes se identificaron un total de doce HU. A continuación se muestran algunas de las más significativas, el resto se encuentran descritas en el Anexo 1.

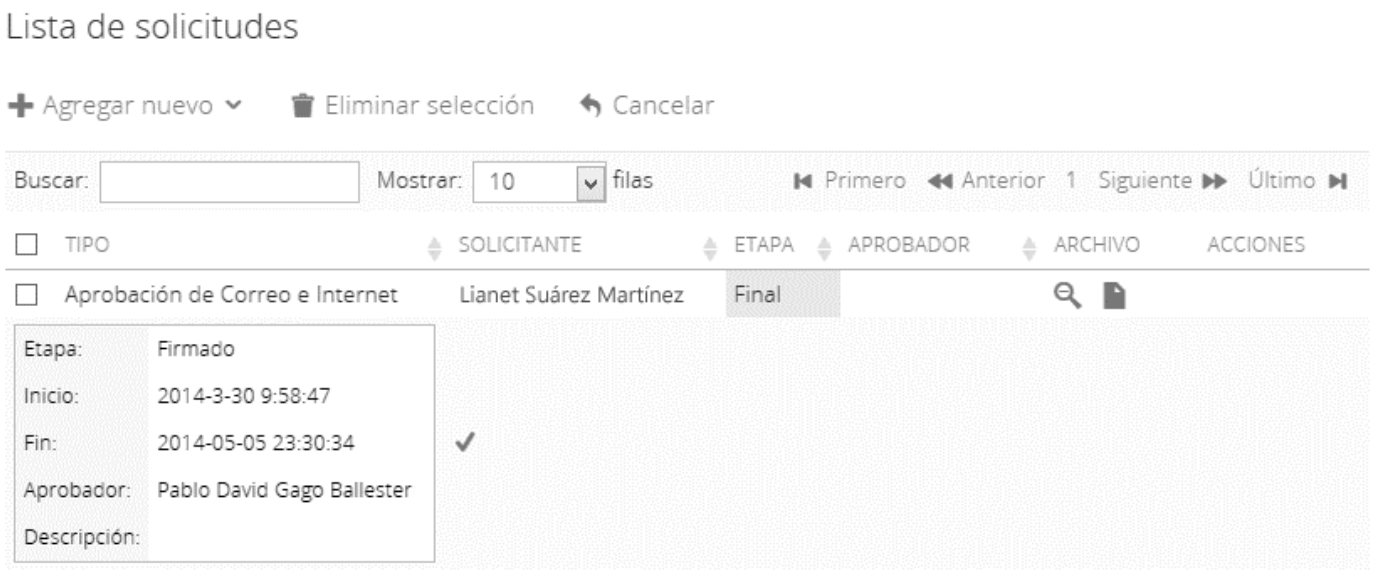
Historia de usuario	
Id.: 6.	Nombre: Administrar solicitudes de cuentas de usuario (1).
Prioridad técnica: Media.	Complejidad de desarrollo: Alta.
Puntos estimados: 3.	Iteración asignada: 5.
Descripción: Permite crear, listar y realizar seguimiento de solicitudes de cuentas de usuario.	
Observaciones: -	
Prototipo:  	

Tabla 2: Historia de usuario "Administrar solicitudes de cuentas de usuario (1)".

Historia de usuario	
Id.: 7.	Nombre: Administrar solicitudes de cuentas de usuario (2).
Prioridad técnica: Media.	Complejidad de desarrollo: Alta.
Puntos estimados: 3.	Iteración asignada: 6.
Descripción: Permite rechazar, aprobar, editar y eliminar solicitudes de cuentas de usuario.	
Observaciones: -	
Prototipo: 	

Tabla 3: Historia de usuario "Administrar solicitudes de cuentas de usuario (2)".

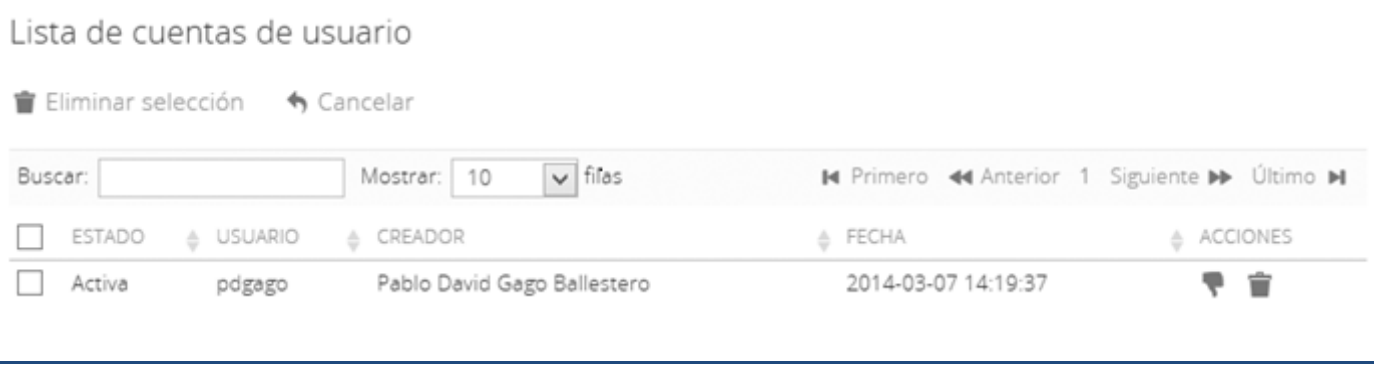
Historia de usuario	
Id.: 8.	Nombre: Administrar cuentas de usuario.
Prioridad técnica: Media.	Complejidad de desarrollo: Alta.
Puntos estimados: 3.	Iteración asignada: 7.
Descripción: Brinda la posibilidad de crear, listar y eliminar cuentas de usuario. También brinda la posibilidad de listar las últimas cuentas de usuario creadas.	
Observaciones: -	
Prototipo: 	

Tabla 4: Historia de usuario "Administrar cuentas de usuario".

2.6. Estimación de tiempo por HU.

Las estimaciones de tiempo asociado a la implementación de las HU las establecen los programadores utilizando como medida el punto. Un punto equivale a una semana ideal de programación. Las HU generalmente valen de uno a tres puntos (19).

La siguiente tabla muestra la estimación del esfuerzo medido en tiempo para cada HU.

Id. de HU	Historia de usuario	Estimación
1	Gestionar recursos.	3
2	Gestionar plantillas.	3
3	Administrar estructura de la entidad.	2
4	Definir aprobadores.	0.2
5	Gestionar flujos de aprobación.	3
6	Administrar solicitudes de cuentas de usuario (1).	3
7	Administrar solicitudes de cuentas de usuario (2).	3
8	Administrar cuentas de usuario.	3
9	Gestionar grupos de usuarios.	1
10	Autenticar usuario.	0.3
11	Administrar registros de sucesos del sistema.	0.4
12	Buscar información en un listado de datos.	0.1
Total		**22.3**

Tabla 5: Estimación de tiempo por HU.

2.7. Plan de iteraciones.

Las iteraciones agrupan varias HU, no debiendo exceder las tres semanas cada iteración. Mientas más iteraciones se realicen, más intercambios hay con el cliente, por lo que disminuye el número de cambios y se utiliza más eficientemente el tiempo. Las HU que abarca cada iteración y el orden en que se implementan son a propuesta del cliente y en dependencia de la prioridad para el negocio, aunque se debe llegar a acuerdos entre el cliente y los desarrolladores (19). Para el SACU, se definió el siguiente plan de iteraciones.

Iteración	Id. de HU	Nombre de HU	Duración
1	1	Gestionar recursos.	3 semanas.
2	2	Gestionar plantillas.	3 semanas.

3	3	Administrar estructura de la entidad.	2.2 semanas.
	4	Definir aprobadores.	
4	5	Gestionar flujos de aprobación.	3 semanas.
5	6	Administrar solicitudes de cuentas de usuario (1).	3 semanas.
6	7	Administrar solicitudes de cuentas de usuario (2).	3 semanas.
7	8	Administrar cuentas de usuario.	3 semanas.
8	9	Gestionar grupos de usuarios.	2.1 semanas.
	10	Autenticar usuario.	
	11	Administrar registros de sucesos del sistema.	
	12	Buscar información en un listado de datos.	

Tabla 6: Plan de iteraciones.

2.8. Plan de entregas.

El plan de entregas tiene como objetivo definir una estimación del contenido y la fecha de cada entrega que se le hará al cliente. Cada entrega no debe tardarse más de tres meses (19). El siguiente plan muestra las entregas agrupadas en iteraciones.

Iteraciones	Primera entrega (13-1-2014)	Segunda entrega (24-2-2014)	Tercera entrega (1-5-2014)	Entrega final (15-5-2014)
1	X	X	X	X
2	X	X	X	X
3	X	X	X	X
4		X	X	X
5		X	X	X
6			X	X
7			X	X
8			X	X

Tabla 7: Plan de entregas.

2.9. Modelo de datos.

El modelo de datos responde una serie de preguntas específicas como por ejemplo: ¿Cuáles son los objetos de datos primarios que va a procesar el sistema?, ¿Cuál es la composición de cada objeto de datos? ¿Cuál

es la relación entre los objetos? Para responder estas preguntas, los métodos de modelado de datos hacen uso del diagrama de entidad-relación (DER) (38).

Los DER tienen como propósito primario representar gráficamente la pareja objeto-relación que constituye la piedra angular del modelo de datos. Entre los componentes principales de un DER se encuentran: objetos de datos, atributos, relaciones y varios indicadores tipo (38).

El DER de la Figura 6 permite obtener una vista del modelo de datos del SACU. La entidad “auth_user” es generada por el marco de trabajo Django, el resto de ellas han sido definidas por el equipo de desarrollo.

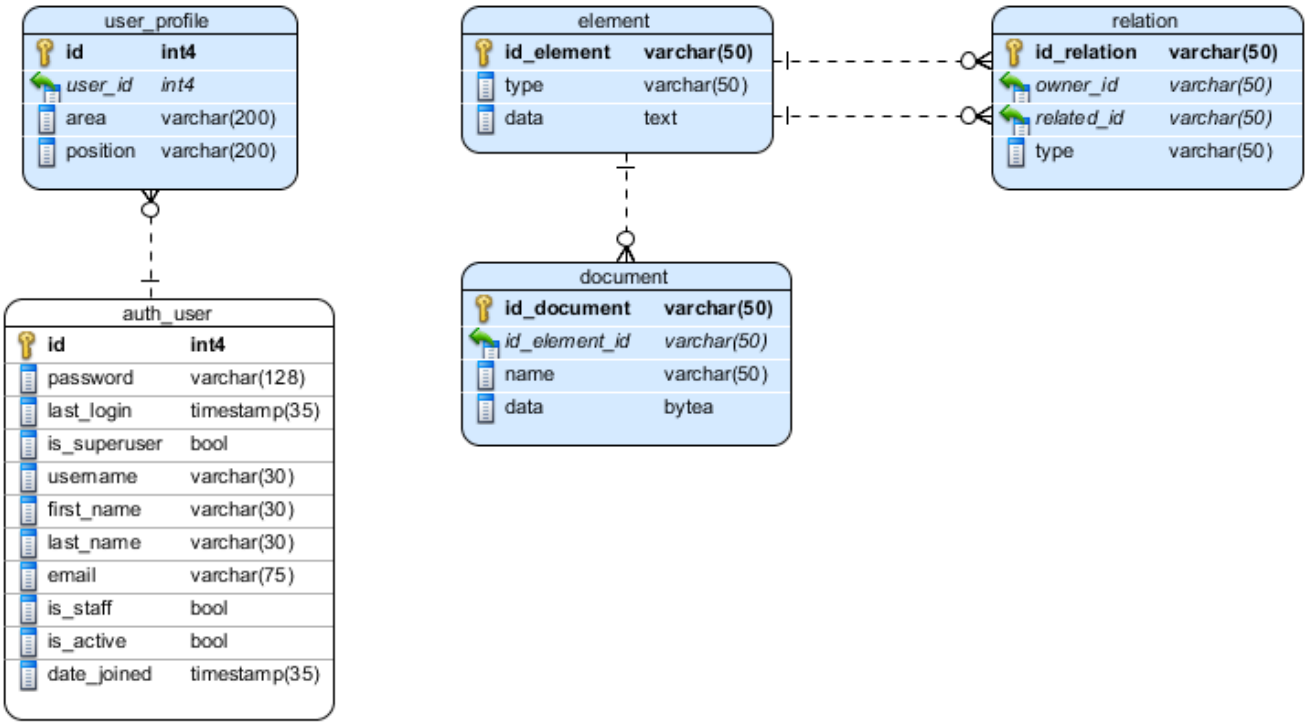


Figura 6: Modelo de datos físico del SACU.

Las entidades que conforman el modelo de datos son:

- auth_user: Constituye una de las entidades generadas por el marco de trabajo Django para garantizar la rápida y segura autenticación de los usuarios en el sistema.
- user_profile: Contiene información sobre los usuarios del sistema, que no es almacenada en la entidad auth_user, y que es necesaria para el correcto funcionamiento del mismo.
- element: Contiene todos los elementos del sistema: recursos, plantillas, estructura administrativa, aprobadores, flujos de aprobación, solicitudes de cuentas de usuario y las propias cuentas de usuario.
- document: Contiene los documentos referentes a las solicitudes de cuentas de usuario.

- relation: Contiene las relaciones entre los diferentes elementos.

2.10. Patrones de diseño.

En la ingeniería de software un patrón es una descripción de un problema y su solución, que recibe un nombre y que puede emplearse en otros contextos. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas. Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería del software; por el contrario, intentan codificar conocimientos, expresiones y principios ya existentes (34).

Entre los patrones a emplear en la construcción del actual sistema, se encuentran los GRASP (*General Responsibility Assignment* Software *Patterns*, en español, Patrones Generales de Asignación de Responsabilidades de Software), que son una mera codificación de los principios fundamentales de la asignación de responsabilidades a objetos (34), dichos patrones serán descritos a continuación.

- Experto: Consiste en asignar una responsabilidad al experto en información, es decir, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Si esto se hace de forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se presenta la oportunidad de reutilizar los componentes en futuras aplicaciones (34).

```
class ManagerElement():
    def Add(self, type, data):
        id = uuid.uuid4()
        element = Element(id_element=id, type=type, data=data)
        element.save()
        return id.__str__()
```

Figura 7: Ejemplo de aplicación del patrón Experto.

La clase *ManagerElement* (ver Figura 7), es la encargada de manejar objetos de tipo *Element*. El método *Add* adiciona un nuevo objeto de este tipo en la base de datos.

- Creador: La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. El diseño bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización (34).

```
class PostgresConnector:
    def _ConnectionArgs(self, args):
        try:
            self._connection = PostgresConnection(args['host'], args['user'],
                args['password'], args['data_base'], args['port'])
            return self._connection
        except Exception, ex:
            ThrowExceptionToLog(None, ex)
            raise Exception(ex)
        return False
```

Figura 8: Ejemplo de aplicación del patrón Creador.

El método *ConnectionArgs* (ver Figura 8) es accedido por varios métodos de la misma clase para crear objetos de tipo *PostgresConnection*. Sólo la clase *PostgresConnector* genera estos objetos.

- Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo acoplamiento no depende de muchas otras. El bajo acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios, que son más reutilizables y más fáciles de entender (34).

```
class XmlMapper():
    def TransformXmlFromXsl(self, xmlPath, xslPath):
        xml = etree.fromstring(xmlPath)
        xsl = etree.parse(xslPath)
        transform = etree.XSLT(xsl)
        return transform(xml)
```

Figura 9: Ejemplo de aplicación del patrón Bajo Acoplamiento.

La clase *XmlMapper* (ver Figura 9) no depende de otras clases en el sistema. Espera sólo valores nativos del lenguaje Python, lo que evita conocer las implementaciones de otras clases.

- Alta cohesión: En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que no realicen un trabajo enorme y que además colaboran con otras clases para llevar a cabo las tareas. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla, además a menudo genera un bajo acoplamiento (34).

```
class ManagerDocument():
    def Add(self, name, element, data):

    def Delete(self, id):
        Document.objects.get(pk=id).delete()

    def Update(self, id, data=None):
```

Figura 10: Ejemplo de aplicación del patrón Alta Cohesión.

En la clase *ManagerDocument* (ver Figura 10) se separan las responsabilidades, y cada una realiza tareas a fines con la utilidad de la clase.

- Controlador: Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. Un corolario importante del patrón controlador es que las operaciones del sistema deberían manejarse en la capa de dominio de los objetos y no en las de interfaz, presentación o aplicación. Esto soporta la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras (34).

```
def Delete(request, type):
    ids = request.GET.getlist('ids[]')
    ManagerElement().Delete(ids)
    return JsonResponse([])
```

Figura 11: Ejemplo de aplicación del patrón Controlador.

El método *Delete* (ver Figura 11) elimina un objeto de tipo *Element* de la base de datos al haberse ejecutado una acción invocada por la interfaz gráfica. Sirve de mediador entre el modelo y las plantillas.

- Indirección: Se asigna la responsabilidad a un objeto intermedio para que medie entre otros componentes o servicios, y éstos no terminen directamente acoplados (34).

```
def RenderFromXsl(request, template_id):
    transformation = xmlMapper.TransformXmlFromXsl('<xml>', xslPath)
    template = Template(transformation)
    return render_to_response('template/templates/editor.html',
                              {'template_elements': template.render(Context())},
                              context_instance=RequestContext(request))
```

Figura 12: Ejemplo de aplicación del patrón Indirección.

El objeto *xmlMapper* (ver Figura 12) implementa funciones para mediar entre cualquier biblioteca de tratamiento XML y otras clases que hacen uso de funciones con datos XML.

Los patrones GOF (*Gang of Four*, en español, Banda de los Cuatro), por su parte, describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos (39). Seguidamente se exponen los patrones GOF empleados en la construcción del SACU.

- Acción: Tiene como propósito definir una acción como un objeto para evitar la redundancia producida por diferentes elementos gráficos que realizan la misma función (40).

```
function InitialConfigs(){
    objectTable = DataTables('.table');
}
```

Figura 13: Ejemplo de aplicación del patrón Acción.

El objeto *objectTable* (ver Figura 13) es accedido por más de un elemento gráfico para realizar funciones como adición, eliminación o actualización de registros en cualquier tabla de las interfaces de usuario.

- Decorador: Tiene como propósito añadir responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia (40).

```
@login_required
@check_permission()
def List(request):
    return render_to_response('approval_flow/templates/list.html',
                              {'flows': views.GetList(ELEMENT_TYPE[0])},
                              context_instance=RequestContext(request))
```

Figura 14: Ejemplo de aplicación del patrón Decorador.

En el caso de la figura anterior se comprueba por medio de decoradores que un usuario que acceda al método *List* por cualquier vía, esté previamente autenticado y tenga los privilegios necesarios para ejecutarlo.

- Fábrica abstracta: Tiene como propósito proporcionar una interfaz para la creación de familias de objetos interdependientes o interrelacionados, sin especificar sus clases concretas (40).

```
def FactoryConnector(type):
    if type == 'MySql':
        from library.mysql.mysql_connector import MySQLConnector
        return MySQLConnector()
    elif type == 'Postgres':
        from library.postgressql.postgres_connector import PostgresConnector
        return PostgresConnector()
    else:
        raise Exception('The connector type is not valid.')
```

Figura 15: Ejemplo de aplicación del patrón Fábrica Abstracta.

La clase *FactoryConnector* (ver Figura 15) se encarga de crear objetos de conectores conociendo su tipo, así no es necesario en tiempo de ejecución conocer la clase que debe ser instanciada.

2.11. Tarjetas CRC.

Las tarjetas CRC permiten a los progamadores centrarse y apreciar mejor la programación orientada a objetos. Son usadas para representar clases con sus respectivas responsabilidades y colaboraciones (41). A continuación se muestran dos de las tarjetas CRC elaboradas durante el desarrollo del SACU, el resto se pueden encontrar en el Anexo 2.

Tarjeta CRC
Clase PostgresConnection.

Responsabilidades	**Colaboradores**
Abrir conexión.	
Cerrar conexión.	
Probar conexión.	

Tabla 8: Tarjeta CRC "PostgresConnection".

Tarjeta CRC	
Clase PostgresConnector.	
Responsabilidades	**Colaboradores**
Abrir conexión.	
Cerrar conexión.	
Probar conexión.	
Obtener el formulario correspondiente al conector de PostgreSQL.	
Definir parámetros de conexión.	PostgresConnection.
Recorrer resultados de las consultas.	
Obtener los usuarios del recurso.	
Crear usuario en el recurso.	
Eliminar usuario en el recurso.	
Habilitar usuario en el recurso.	
Deshabilitar usuario en el recurso.	
Dar privilegios a un usuario.	
Quitar privilegios a un usuario.	
Buscar usuarios por un criterio dado.	
Obtener la lista de tablas del recurso.	
Obtener las columnas de una tabla dada.	
Obtener todos los datos de una columna.	
Generar una consulta personalizada.	
Ejecutar consulta.	

Tabla 9: Tarjeta CRC "PostgresConnector".

2.12. Conclusiones.

La planificación y diseños realizados a lo largo del capítulo permitieron obtener una visión más clara y detallada del sistema que se desea desarrollar. En total se identificaron doce HU, distribuidas en ocho iteraciones que serán entregadas al cliente en cuatro etapas, de las cuales, la última debe ser el sistema completamente funcional. La definición de los patrones de diseño a emplear en la implementación del sistema constituye un elemento clave para garantizar la correcta solución a los problemas más comunes de la programación orientada a objetos. Por su parte, la especificación de las tarjetas CRC permitió representar las clases del sistema con sus respectivas responsabilidades y clases colaboradoras.

Capítulo 3
CONSTRUCCIÓN Y PRUEBAS

En este capítulo se definen los artefactos correspondientes a las etapas de implementación y pruebas, tales como los estándares de programación que debe seguir el equipo de desarrollo, las tareas de programación derivadas de cada HU, y las diferentes pruebas a partir del código fuente y el propio funcionamiento del SACU.

3.1. Estándares de codificación.

Un estándar de codificación está compuesto por un conjunto de pautas que llevan por objetivo uniformar la escritura del código fuente de un software para facilitar su legibilidad.

Las pautas fundamentales definidas para la implementación son las siguientes:

- El tamaño máximo de las líneas de código debe ser aproximadamente de ochenta a noventa caracteres, garantizando la completa visibilidad de las líneas de código sin necesidad de realizar desplazamiento horizontal.
- Los nombres de las clases, las funciones y las propiedades adoptarán la notación *UpperCamelCase* y no se utilizará el guion bajo como delimitador entre palabras.
- Los nombres de los atributos, las variables y los parámetros adoptarán la notación *lowerCamelCase*. Ésta define que todas las palabras que conformen el nombre, excepto la primera, comenzarán con la primera letra en mayúsculas y el resto de las letras en minúsculas.

3.2. Tareas de programación.

Las tareas de programación son una división más detallada de las HU que escriben los programadores. Pueden ser de tipo desarrollo, corrección o mejora (19). En la presente investigación todas la tareas son de tipo desarrollo, por lo que no se especifica este campo. Cada tarea tiene además, un identificador propio, el identificador de la HU a la que pertenece, un nombre, un programador responsable, un tiempo estimado, fecha de inicio, fecha de fin y una descripción. A continuación se exponen las correspondientes a la administración de solicitudes de cuentas de usuario, el resto de las tareas se encuentran en el Anexo 3.

Tarea de programación	
Número de HU: 6.	Id. de tarea: 6.1.

Nombre de la tarea: Crear solicitud de cuenta de usuario.	
Fecha de inicio: 4/2/2014.	Fecha de fin: 17/2/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 2.
Descripción: El sistema brinda la opción de seleccionar el flujo de aprobación que debe seguir la solicitud. Se muestran a partir de dicha selección varios campos para ser completados. El usuario introduce los datos correspondientes y finaliza la creación de la solicitud. Finalmente, el sistema la guarda, estableciéndola en la primera etapa del flujo de aprobación.	

Tabla 10: Tarea de programación "Crear solicitud de cuenta de usuario".

Tarea de programación	
Número de HU: 6.	Id. de tarea: 6.2.
Nombre de la tarea: Listar solicitudes de cuentas de usuario.	
Fecha de inicio: 18/2/2014.	Fecha de fin: 20/2/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.3.
Descripción: El sistema muestra un listado con todas las solicitudes de cuentas de usuario, a partir del cual se pueden llevar a cabo algunas acciones sobre las mismas.	

Tabla 11: Tarea de programación "Listar solicitudes de cuentas de usuario".

Tarea de programación	
Número de HU: 6.	Id. de tarea: 6.3.
Nombre de la tarea: Realizar seguimiento de solicitud de cuenta de usuario.	
Fecha de inicio: 21/2/2014.	Fecha de fin: 24//2/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.4.
Descripción: El sistema brinda la posibilidad de visualizar el historial de etapas por las que ha pasado la solicitud.	

Tabla 12: Tarea de programación "Realizar seguimiento de solicitud de cuenta de usuario".

Tarea de programación	
Número de HU: 7.	Id. de tarea: 7.1.
Nombre de la tarea: Rechazar solicitud de cuenta de usuario.	
Fecha de inicio: 25/2/2014.	Fecha de fin: 3/3/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 1.
Descripción: El sistema brinda la opción de rechazar una solicitud de cuenta de usuario, obligando a escribir los motivos. El usuario finaliza la operación y el sistema interrumpe el flujo de la solicitud, quedando ésta invalidada.	

Tabla 13: Tarea de programación "Rechazar solicitud de cuenta de usuario".

Tarea de programación	
Número de HU: 7.	Id. de tarea: 7.2.
Nombre de la tarea: Aprobar solicitud de cuenta de usuario.	
Fecha de inicio: 4/3/2014.	Fecha de fin: 10/3/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 1.
Descripción: El sistema brinda la opción de aprobar una solicitud de cuenta de usuario, una vez revisada, se procede a realizar la aprobación. El sistema le da seguimiento a la solicitud hacia su próxima etapa.	

Tabla 14: Tarea de programación "Aprobar solicitud de cuenta de usuario".

Tarea de programación	
Número de HU: 7.	Id. de tarea: 7.3.
Nombre de la tarea: Editar solicitud de cuenta de usuario.	
Fecha de inicio: 11/3/2014.	Fecha de fin: 14/3/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.4.
Descripción: El sistema brinda la opción de editar una solicitud de cuenta de usuario, el usuario realiza modificaciones pertinentes y el sistema reenvía la solicitud al inicio del flujo de aprobación.	

Tabla 15: Tarea de programación "Editar solicitud de cuenta de usuario".

Tarea de programación	
Número de HU: 7.	Id. de tarea: 7.4.
Nombre de la tarea: Eliminar solicitud de cuenta de usuario.	
Fecha de inicio: 15/3/2014.	Fecha de fin: 17/3/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.3.
Descripción: El sistema brinda la opción de eliminar una solicitud de cuenta de usuario mostrando un mensaje de confirmación. Si la respuesta del usuario es afirmativa, el sistema procede a eliminar dicha solicitud.	

Tabla 16: Tarea de programación "Eliminar solicitud de cuenta de usuario".

3.3. Diagrama de despliegue.

El diagrama de despliegue muestra una configuración estática de los elementos de procesamiento en tiempo de ejecución del software, así como los componentes que forman parte de dichos elementos. Constituye un gráfico de nodos conectados por asociaciones de comunicación. Los nodos pueden contener instancias de componentes que se ejecutan dentro de ellos (42). A continuación se muestra el diagrama de despliegue del sistema:

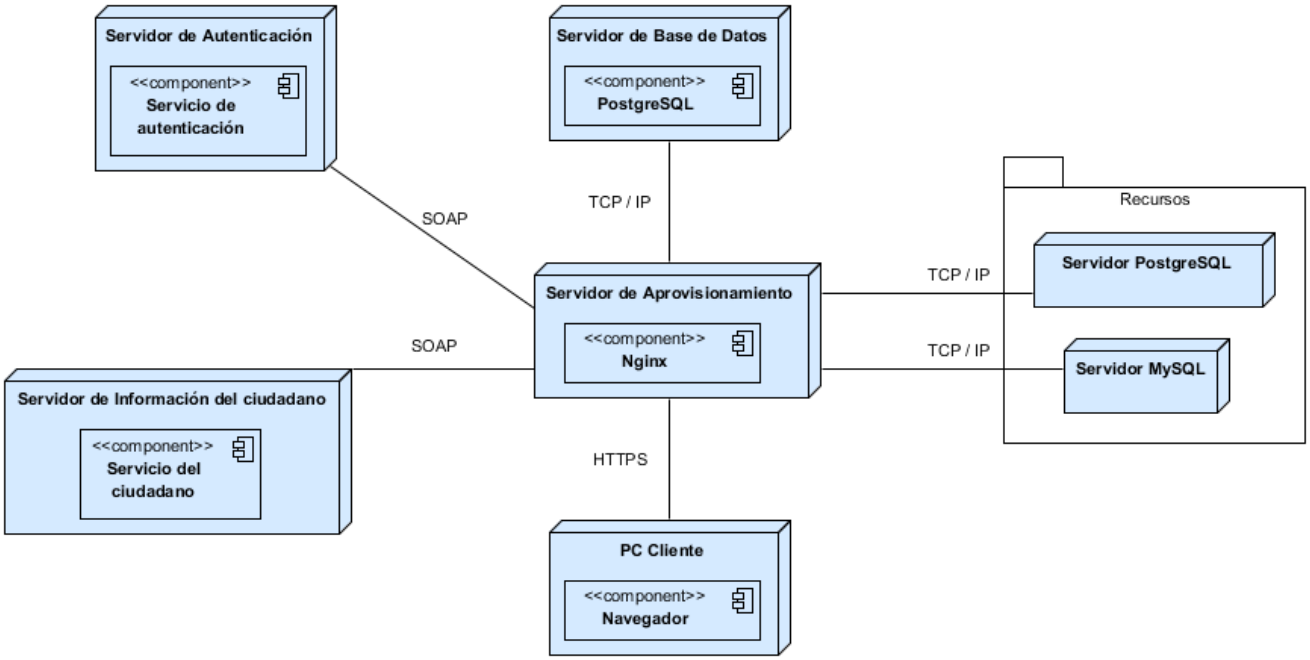


Figura 16: Diagrama de despliegue del SACU.

En el diagrama de despliegue anterior se pueden apreciar los siguientes nodos:

- Servidor de Aprovisionamiento: Es el servidor en el cual se encuentra instalado el SACU.
- Servidor de Base de Datos: En este servidor reside la base de datos del SACU.
- Servidor de autenticación: Es empleado para obtener la estructura administrativa de la UCI.
- Servidor de Información del ciudadano: Se utiliza para obtener los cargos de cada área de la estructura administrativa de la UCI.
- PC Cliente: Desde ella los usuarios acceden al SACU.
- Servidor PostgreSQL, Servidor MySQL: A estos servidores se accede desde el SACU para obtener información o para crear cuentas de usuario en ellos.

3.4. Pruebas.

Las pruebas del software son un elemento crítico para la garantía de calidad del software. El software debe probarse desde dos perspectivas diferentes: la lógica interna del programa, que se comprueba utilizando técnicas de diseño de casos de prueba de caja blanca, y los requerimientos del software, que se comprueban utilizando técnicas de diseño de casos de prueba de caja negra. En ambos casos, se intenta encontrar el mayor número de errores con la menor cantidad de esfuerzo y tiempo (38).

Las pruebas de caja blanca se basan en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles (38).

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software. Pretenden demostrar que sus funciones son operativas, que la entrada se acepta de forma adecuada, y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene (38).

3.4.1. Pruebas unitarias.

La prueba de unidad centra el proceso de verificación en la menor unidad del diseño del software: el componente o módulo (38). Se enfocan en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente (43). En XP la producción de código está dirigida por las pruebas unitarias. Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema (19).

Para la realización de las pruebas unitarias del SACU, se hizo uso de una biblioteca llamada “unittest”, desarrollada para automatizar esta tarea en Python. Seguidamente se muestran los casos de prueba aplicados a algunas de las funcionalidades más importantes del SACU, en el Anexo 4 se encuentran los restantes.

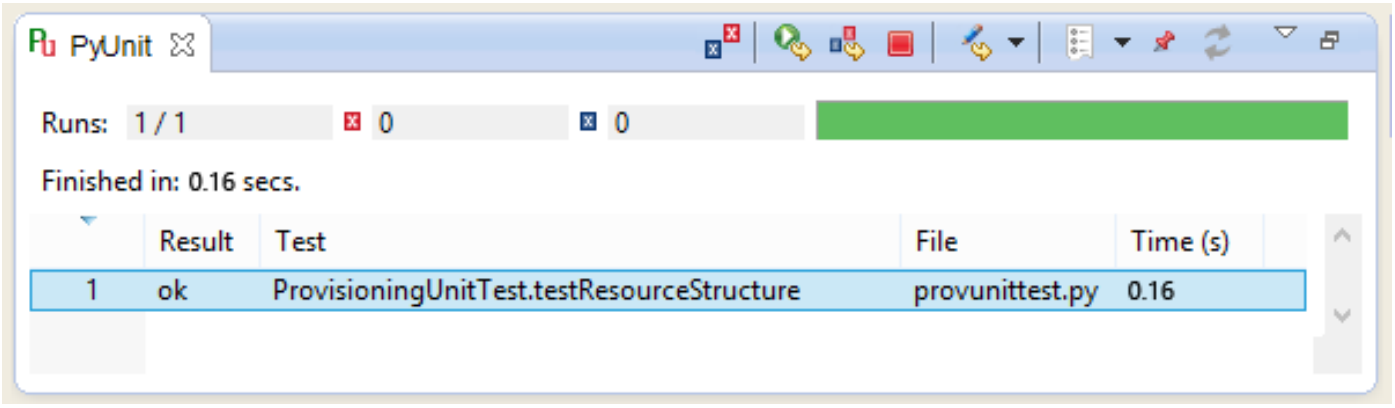
Caso de prueba unitaria	
Nombre de prueba: TestResourceStructure.	Id. de prueba: 3.
Descripción del caso de prueba: Permite obtener todos los elementos de tipo: recurso de lectura.	
Entradas: -	
Criterio de aceptación: El tamaño del arreglo de recursos devuelto es igual al esperado.	
Resultado: 	

Tabla 17: Caso de prueba unitaria "TestResuorceStructure".

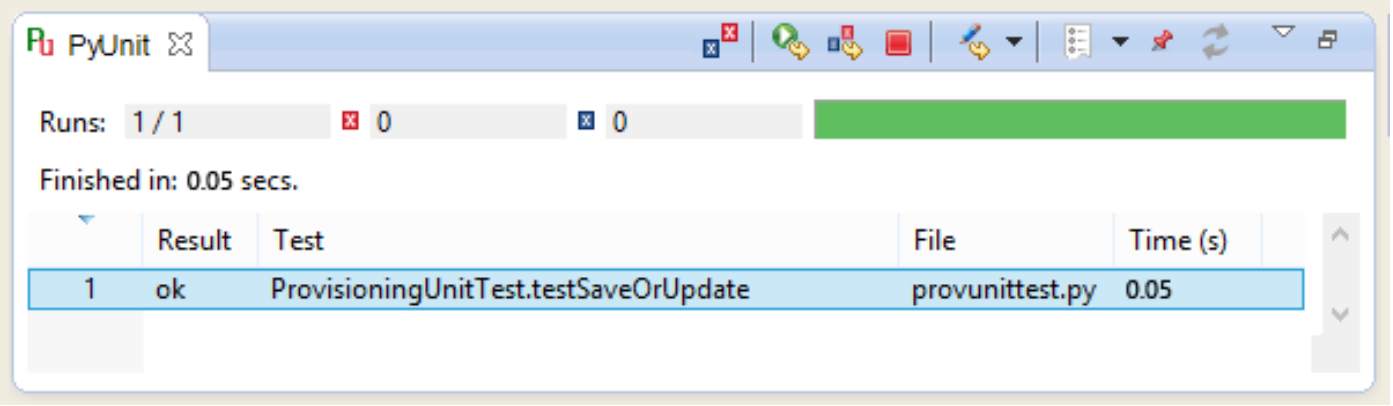
Caso de prueba unitaria	
Nombre de prueba: TestSaveOrUpdate.	Id. de prueba: 4.
Descripción del caso de prueba: Permite comprobar que el elemento pasado como parámetro se guarda correctamente en la base de datos.	
Entradas: Tipo del elemento, identificador del elemento e información del elemento.	
Criterio de aceptación: Se verifica directamente en la base de datos el cambio realizado.	
Resultado: 	

Tabla 18: Caso de prueba unitaria "TestSaveOrUpdate".

3.4.1.1. Resultados de las pruebas unitarias.

Las pruebas unitarias fueron aplicadas a cada método de considerable importancia una vez finalizada su implementación. Como consecuencia de esta estrategia de aplicación de las pruebas unitarias, no se tiene un control de las mismas en cuanto a cantidad de iteraciones. Cada uno de estos métodos fue probado al implementarse o modificarse. En total se detectaron veintidós no conformidades. Todas fueron corregidas en el momento de su detección.

3.4.2. Pruebas de integración.

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño (38).

Dado que es el SACU quien utiliza los dos servicios a los que se integra, el enfoque seleccionado para realizar las pruebas de integración es el ascendente, que empieza con la construcción y prueba de los módulos de los niveles más bajos de la estructura del programa. Las siguientes tablas muestran los casos de prueba de integración correspondientes al consumo de estos servicios.

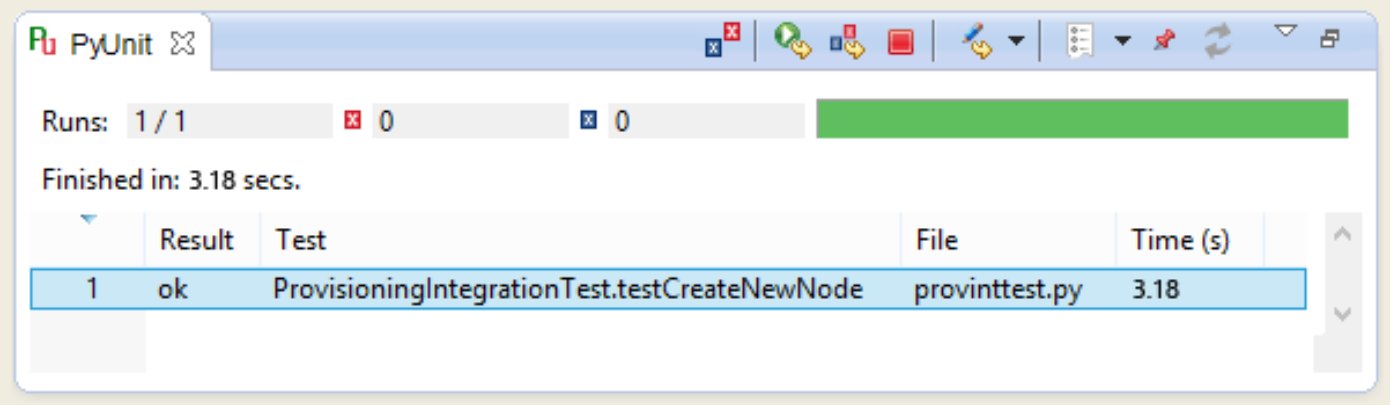
Caso de prueba de integración	
Nombre de prueba: TestCreateNewNode.	Id. de prueba: 1.
Descripción del caso de prueba: Permite verificar que la conexión al servicio funciona correctamente, se obtiene la estructura administrativa de la UCI, y se crea o actualiza la nueva información en la base de datos del sistema desarrollado.	
Entradas: -	
Criterio de aceptación: Se obtiene la estructura administrativa de la UCI.	
Resultado: 	

Tabla 19: Caso de prueba de integración "TestCreateNewNode".

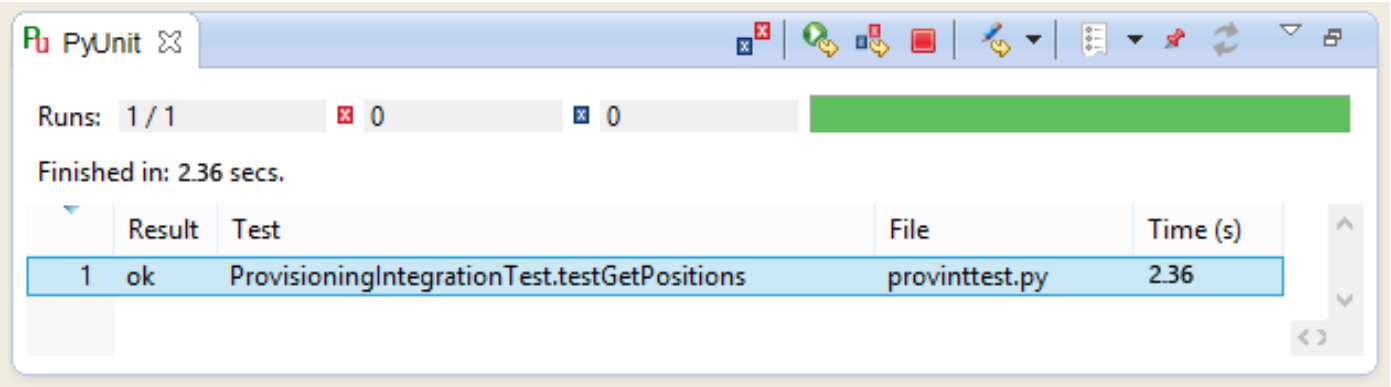
Caso de prueba de integración	
Nombre de prueba: TestGetPositions.	Id. de prueba: 2.
Descripción del caso de prueba: Permite la obtención de los cargos de un área en la estructura administrativa de la UCI.	
Entradas: Identificador de área en la estructura administrativa de la UCI.	
Criterio de aceptación: El tamaño del arreglo de cargos devuelto es igual al esperado.	
Resultado: 	

Tabla 20: Caso de prueba de integración "TestGetPositions".

3.4.2.1. Resultados de las pruebas de integración.

Las pruebas de integración realizadas al SACU como consecuencia del consumo de los dos servicios, requirieron de previas pruebas unitarias a los dos métodos que interactúan con estos servicios. Al efectuarse la integración con servicios y no con módulos del propio sistema, se hace imposible realizarle las pruebas unitarias a ambas partes. En total se necesitaron cuatro iteraciones de pruebas para cada uno de los métodos.

3.4.3. Pruebas de regresión.

Cada vez que se añade un nuevo módulo como parte de una prueba de integración, el software cambia. Se establecen nuevos caminos de flujo de datos, pueden ocurrir nuevas Entradas/Salidas y se invoca una nueva lógica de control. Estos cambios pueden causar problemas con funciones que antes trabajaban perfectamente. En el contexto de una estrategia de prueba de integración, la prueba de regresión es volver a ejecutar un subconjunto de pruebas que se han llevado a cabo anteriormente para asegurarse de que los cambios no han propagado efectos colaterales no deseados (38). En el caso del SACU se volvieron a ejecutar las pruebas unitarias a los métodos de mayor importancia. Posteriormente se seleccionó un subconjunto de pruebas de aceptación para ser realizadas nuevamente. De forma general se detectaron tres no conformidades que inmediatamente fueron corregidas.

3.4.4. Pruebas de aceptación.

Cuando se construye software a medida para un cliente, se llevan a cabo una serie de pruebas de aceptación para permitir que el cliente valide todos los requerimientos. Estas pruebas las realiza el usuario final (38). Las pruebas de aceptación se escriben en las fases tempranas del desarrollo, antes de que el sistema se implemente, con esto se consigue, además de validar las necesidades de los usuarios, dirigir la implementación (44). Los siguientes casos de prueba de aceptación son los correspondientes a la administración de solicitudes de cuentas de usuario, en el Anexo 5 se muestran los restantes.

Caso de prueba de aceptación	
Nombre de HU: Administrar solicitudes de cuentas de usuario.	Id. de prueba: 6.1.
Nombre del caso de prueba: Crear solicitud de cuenta de usuario.	
Descripción del caso de prueba: Permite verificar que las solicitudes de cuentas de usuario se crean correctamente.	
Condiciones de ejecución: Estar autenticado. Debe existir al menos un flujo de aprobación.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Solicitud de cuentas”. ▪ Pulsar la opción “Agregar nuevo” y seleccionar el flujo que debe seguir la solicitud que se está creando. ▪ Llenar los campos necesarios. ▪ Pulsar la opción “Guardar”.
Resultado esperado: El sistema muestra la nueva solicitud en el listado de solicitudes y en el listado de la sección “Pendientes de firma”.
Evaluación de la prueba: Satisfactoria.

Tabla 21: Caso de prueba de aceptación "Crear solicitud de cuenta de usuario".

Caso de prueba de aceptación	
Nombre de HU: Administrar solicitudes de cuentas de usuario.	Id. de prueba: 6.2.
Nombre del caso de prueba: Listar solicitudes de cuentas de usuario.	
Descripción del caso de prueba: Permite validar que el listado de solicitudes de cuentas de usuario se muestra correctamente.	
Condiciones de ejecución: Estar autenticado.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Solicitud de cuentas”.	
Resultado esperado: El sistema muestra el listado de solicitudes de cuenta de usuario.	
Evaluación de la prueba: Satisfactoria.	

Tabla 22: Caso de prueba de aceptación "Listar solicitudes de cuentas de usuario".

Caso de prueba de aceptación	
Nombre de HU: Administrar solicitudes de cuentas de usuario.	Id. de prueba: 6.3.
Nombre del caso de prueba: Realizar seguimiento de solicitud de cuenta de usuario.	
Descripción del caso de prueba: Permite validar que el seguimiento de solicitudes de cuentas de usuario funciona correctamente.	
Condiciones de ejecución: Estar autenticado. Debe existir al menos una solicitud.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Solicitud de cuentas”. ▪ Pulsar la opción “Detalles” correspondiente a la solicitud a la que se desea realizar seguimiento.
Resultado esperado: El sistema despliega un recuadro con los detalles de la solicitud.
Evaluación de la prueba: Satisfactoria.

Tabla 23: Caso de prueba de aceptación "Realizar seguimiento de solicitud de cuenta de usuario".

Caso de prueba de aceptación	
Nombre de HU: Administrar solicitudes de cuentas de usuario.	Id. de prueba: 7.1.
Nombre del caso de prueba: Rechazar solicitud de cuenta de usuario.	
Descripción del caso de prueba: Permite comprobar que la funcionalidad para rechazar las solicitudes de cuentas de usuario se desempeña correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Pendientes de firma”. ▪ Introducir en el campo “Descripción” los motivos por los que se rechaza la solicitud. ▪ Pulsar la opción “Rechazar” correspondiente a la solicitud que se desea rechazar. ▪ Pulsar la opción “Si” que aparece en la pregunta de confirmación.	
Resultado esperado: El sistema no muestra la solicitud rechazada en el listado de solicitudes pendientes de firma. En el listado de solicitudes de la sección “Solicitudes de cuenta”, el campo “Etapa” muestra la palabra “Rechazado”.	
Evaluación de la prueba: Satisfactoria.	

Tabla 24: Caso de prueba de aceptación "Rechazar solicitud de cuenta de usuario".

Caso de prueba de aceptación	
Nombre de HU: Administrar solicitudes de cuentas de usuario.	Id. de prueba: 7.2.
Nombre del caso de prueba: Aprobar solicitud de cuenta de usuario.	
Descripción del caso de prueba: Permite confirmar que la aprobación de solicitudes de cuentas de usuario funciona correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Pendientes de firma”. ▪ Pulsar la opción “Firmar” correspondiente a la solicitud que se desea firmar. ▪ Pulsar la opción “Si” que aparece en la pregunta de confirmación.
Resultado esperado: El sistema no muestra la solicitud en el listado de pendientes de firma, para el aprobador que está autenticado y aparece en el listado de pendientes de firma para el nuevo aprobador correspondiente en el flujo de aprobación.
Evaluación de la prueba: Satisfactoria.

Tabla 25: Caso de prueba de aceptación "Aprobar solicitud de cuenta de usuario".

Caso de prueba de aceptación	
Nombre de HU: Administrar solicitudes de cuentas de usuario.	Id. de prueba: 7.3.
Nombre del caso de prueba: Editar solicitud de cuenta de usuario.	
Descripción del caso de prueba: Permite confirmar que la edición de solicitudes de cuentas de usuario funciona correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Solicitud de cuentas”. ▪ Pulsar la opción “Editar” correspondiente a la solicitud que se desea editar. ▪ Realizar las modificaciones necesarias. ▪ Pulsar la opción “Guardar”.	
Resultado esperado: El sistema muestra el listado de solicitudes. Cuando se vuelve a acceder a la solicitud editada, se observan los cambios antes realizados. Se reinicia el flujo de aprobación.	
Evaluación de la prueba: Satisfactoria.	

Tabla 26: Caso de prueba de aceptación "Editar solicitud de cuenta de usuario".

Caso de prueba de aceptación	
Nombre de HU: Administrar solicitudes de cuentas de usuario.	Id. de prueba: 7.4.
Nombre del caso de prueba: Eliminar solicitud de cuenta de usuario.	
Descripción del caso de prueba: Permite comprobar que la eliminación de solicitudes de cuentas de usuario funciona correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios (ser la persona que creó la solicitud).	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Solicitudes de cuenta”. ▪ Eliminar solicitud: - Variante 1: Pulsar la opción “Eliminar” correspondiente a la solicitud que se desea eliminar. - Variante 2: Seleccionar las solicitudes que se desean eliminar y pulsar la opción “Eliminar selección”. ▪ Pulsar la opción “Si” que aparece en la pregunta de confirmación.
Resultado esperado: El sistema muestra el listado de solicitudes en el cual no aparecen las eliminadas.
Evaluación de la prueba: Satisfactoria.

Tabla 27: Caso de prueba de aceptación "Eliminar solicitud de cuenta de usuario".

3.4.4.1. Resultados de las pruebas de aceptación.

Con el objetivo de probar las funcionalidades del sistema se diseñaron treinta y cinco casos de prueba, los cuales arrojaron un total de veintiuna no conformidades. Las pruebas de aceptación fueron realizadas al finalizar cada una de las ocho iteraciones de implementación, cada iteración de implementación requirió tres de pruebas de aceptación. De forma general, en la primera iteración de pruebas de aceptación se detectaron dieciocho no conformidades, de las cuales fueron solucionadas dieciséis. En la segunda, se detectaron cinco no conformidades, incluyendo las dos no resueltas en la iteración anterior; en ésta se resolvieron todas las no conformidades. Finalmente, en la tercera iteración no se detectó ninguna no conformidad. El gráfico de la Figura 17 muestra lo antes explicado.

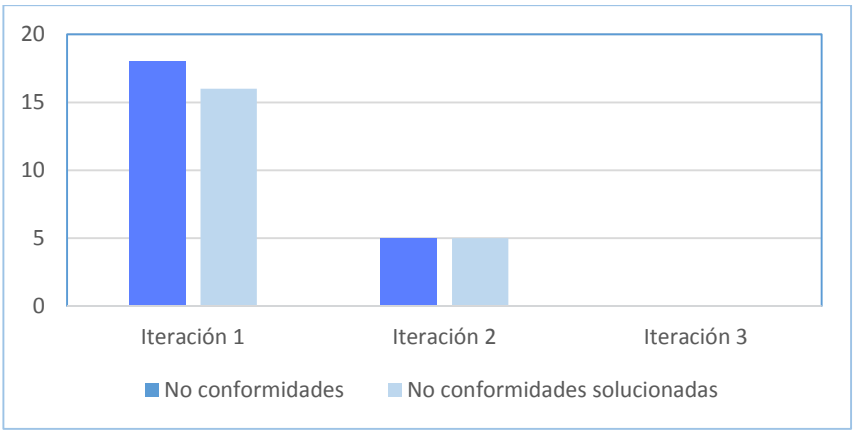


Figura 17: Resultados de las pruebas de aceptación.

3.4.5. Pruebas de rendimiento.

Las pruebas de rendimiento están diseñadas para comprobar este aspecto de calidad en tiempo de ejecución dentro del contexto de un sistema integrado. Se dan durante todos los pasos del proceso de pruebas y permiten además descubrir situaciones que lleven a degradaciones y posibles fallos del software (38).

3.4.5.1. Resultados de las pruebas de rendimiento.

Para llevar a cabo estas pruebas, se creó un entorno con las condiciones mínimas necesarias para la ejecución del sistema. Fueron ejecutadas sobre las funcionalidades que mayor demanda pueden recibir durante el funcionamiento del SACU. La Figura 18 muestra los resultados obtenidos al realizar las pruebas de rendimiento a la funcionalidad que muestra las últimas cuentas creadas.

Label	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Últimas cuentas creadas	40	377	413	618	92	691	0,00%	28,3/sec	179,4
TOTAL	40	377	413	618	92	691	0,00%	28,3/sec	179,4

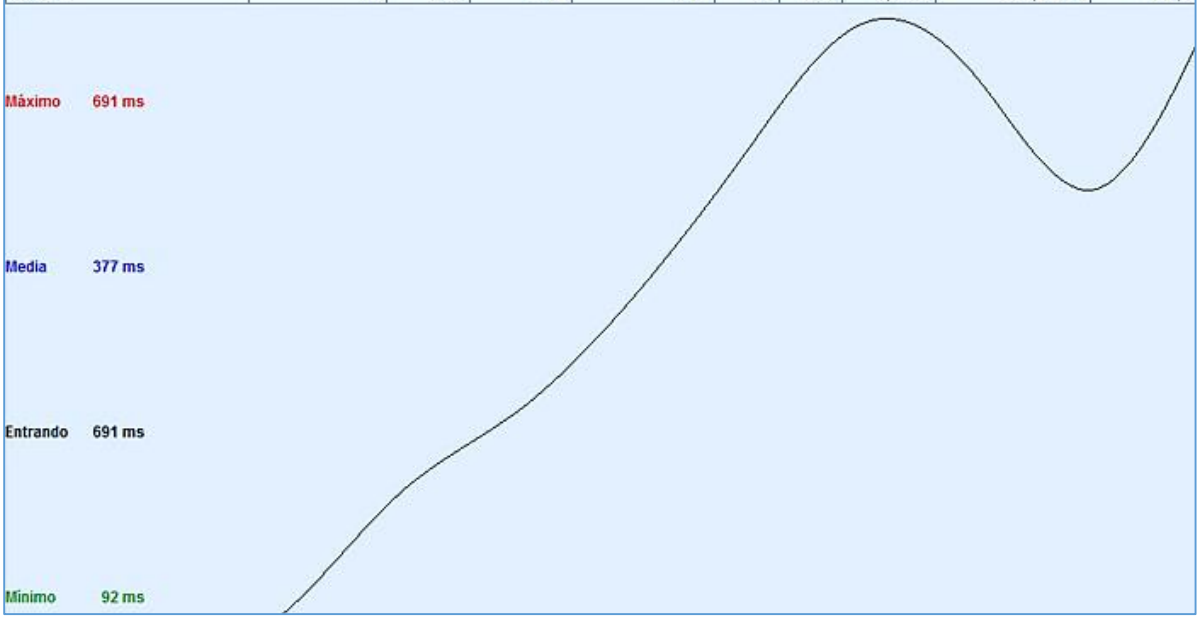


Figura 18: Resultados de las pruebas de rendimiento.

3.4.6. Pruebas de resistencia.

Las pruebas de resistencia están diseñadas para enfrentar a los programas con situaciones anormales, ejecutándolos de forma que demanden recursos en cantidad, frecuencia o volúmenes anormales. En esencia, el objetivo fundamental de estas pruebas es conocer a qué potencia se puede poner a funcionar el sistema antes de que falle (38).

3.4.6.1. Resultados de las pruebas de resistencia.

Al igual que en las pruebas de rendimiento, estas pruebas fueron ejecutadas en un entorno con las condiciones mínimas necesarias para la ejecución del sistema y se seleccionaron las funcionalidades que mayor demanda pueden recibir durante el funcionamiento del SACU. La Figura 19 muestra los resultados obtenidos al realizarle las pruebas de resistencia a la funcionalidad que muestra las últimas cuentas creadas.

Label	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Últimas cuentas creadas	152	4610	4067	9869	170	21011	0,66%	6,9/sec	43,4
TOTAL	152	4610	4067	9869	170	21011	0,66%	6,9/sec	43,4

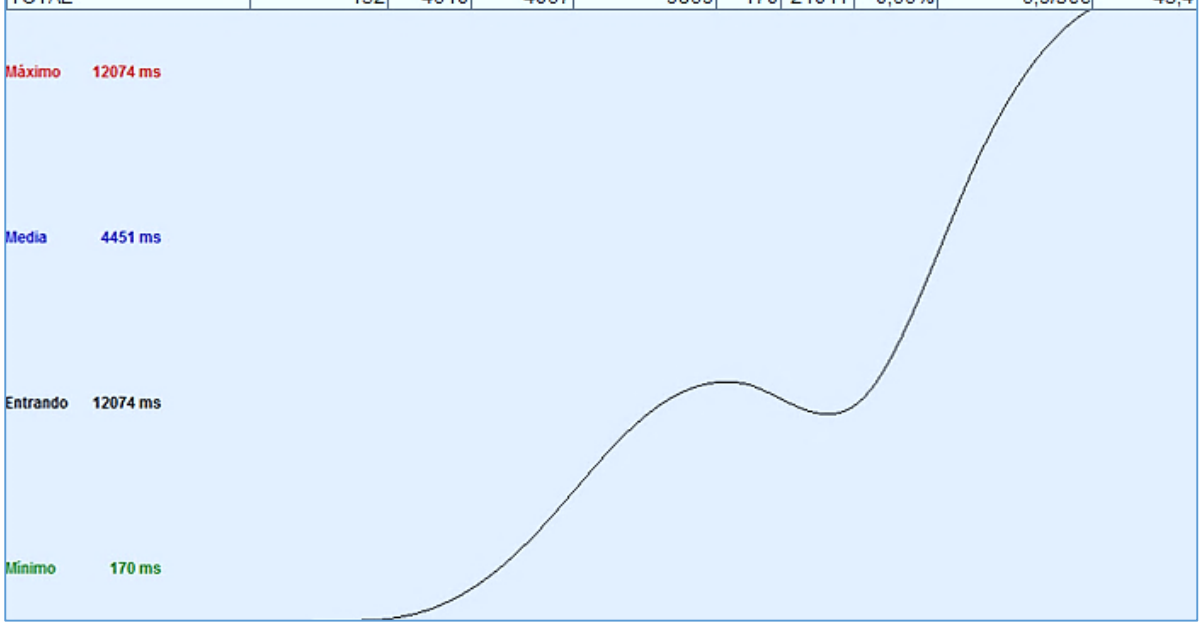


Figura 19: Resultados de las pruebas de resistencia.

3.5. Beneficios de la solución desarrollada.

Partiendo del valor que se le otorga a la solución desarrollada, se considera que la misma permite la centralización del proceso de aprovisionamiento de cuentas de usuario en una única herramienta que interactúa con múltiples recursos. Por otro lado, transforma los sistemas de trabajo de la UCI, incrementando la seguridad de todo el proceso de aprovisionamiento y aumentando la satisfacción y productividad tanto de usuarios como administradores. Se fomenta además el uso de las PKI (*Public Key Infrastructure*, en español, Infraestructura de Clave Pública).

3.6. Conclusiones.

A lo largo de la etapa de implementación se puso en práctica los estándares de codificación definidos, brindándole al código una mayor legibilidad y uniformidad. Las tareas de programación, por su parte, permitieron a los programadores obtener información más detallada sobre las funcionalidades que debían programar. La realización de las pruebas inmediatamente después de cada iteración dio como resultado que una vez finalizada la última iteración, el sistema quedó libre de algunos errores que se fueron detectando y corrigiendo en iteraciones anteriores.

CONCLUSIONES GENERALES

Una vez finalizada la investigación que sirvió de base para el desarrollo del Sistema de Aprovisionamiento de Cuentas de Usuario, se puede concluir que:

- A través del estudio de los principales sistemas de aprovisionamiento de cuentas de usuario existentes, se logró comprender las características del funcionamiento de este proceso.
- El diseño y planificación del Sistema de Aprovisionamiento de Cuentas de Usuario permitió describir los procesos que debió poner en ejecución el sistema.
- Mediante la implementación y validación de la solución desarrollada, se constató que el sistema permite gestionar el ciclo de vida de las cuentas de usuario en la Universidad de las Ciencias Informáticas.

RECOMENDACIONES

Se recomienda a los interesados en continuar el presente trabajo:

- Ampliar la gama de conectores embebidos en el sistema para permitirle crear cuentas de usuario en diferentes recursos.
- Implementar las funcionalidades necesarias para gestionar conectores.
- Añadir funcionalidades para realizar la conciliación de cuentas de usuario, y para habilitar o deshabilitar dichas cuentas.
- Validar la solución desarrollada teniendo en cuenta las siguientes categorías y subcategorías de calidad, definidas en la NC-ISO/IEC 9126: mantenibilidad, usabilidad, confiabilidad, portabilidad y seguridad.

REFERENCIAS BIBLIOGRÁFICAS

1. **Universidad de Salamanca.** Inicio. [En línea] 2013. [Citado el: 10 de 12 de 2013.] https://identidad.usal.es/web/altaybajaCorreo.

2. **Montoya S., José A. y Restrepo R., Zuleima.** *Gestión de identidades y control de acceso desde una perspectiva organizacional.* 1, Medellín : Ing. USBMed, 2012, Vol. III. ISSN: 2027-5846.

3. **affiliates., Oracle and/or its.** Oracle® Fusion Middleware Administrator's Guide for Oracle Directory Integration Platform. *Understanding the Oracle Directory Integration Platform for Provisioning.* [En línea] 2009. [Citado el: 2 de 10 de 203.] http://docs.oracle.com/cd/E15523_01/oid.1111/e10031/odip_provisioning.htm.

4. **Fraser-Thill, Rebecca .** Definition of Identity. [En línea] 30 de 4 de 2011. [Citado el: 5 de 10 de 2013.] http://tweenparenting.about.com/od/behaviordiscipline/a/Definition-of-Identity.htm.

5. **Hitachi ID Systems.** Defining Identity Management. [En línea] 2013. [Citado el: 9 de 10 de 2013.] http://hitachi-id.com/identity-manager/docs/identity-management-defined.html.

6. **Baldwin, Adrian, Mont, Marco Casassa y Shiu, Simon.** *Using Modelling and Simulation for Policy Decision Support.* Bristol : IEEE computer society, 2009. 978-0-7695-3742-9/09.

7. **IBM Tivoli Provisioning Manager.** Tivoli Provisioning Manager, Versión 5.1.1 . *Arquitectura de los componentes de Tivoli Provisioning Manager.* [En línea] [Citado el: 22 de 10 de 2013.] http://publib.boulder.ibm.com/infocenter/tivihelp/v20r1/index.jsp?topic=%2Fcom.ibm.tivoli.tpm.ovw.doc%2Foverview%2Fcovw_architect.html&lang=es.

8. **Oracle and/or its affiliates.** Oracle® Fusion Middleware Developer's Guide for Oracle Identity Manager. *Product Architecture.* [En línea] 2012. [Citado el: 9 de 10 de 2013.] http://docs.oracle.com/cd/E27559_01/dev.1112/e27150/oimarchtcture.htm.

9. **Oracle Identity Manager.** Arquitectura de Oracle Identity Manager. [En línea] 2009. [Citado el: 28 de 11 de 2013.] http://docs.oracle.com/cd/E10391_01/doc.910/e10374/oimarchtcture.htm.

10. **WSO2 Identity Server.** WSO2 Identity Server. [En línea] [Citado el: 15 de 10 de 2013.] http://wso2.com/products/identity-server/.

11. **Infante, Jorge.** La Arquitectura de Software desde adentro. *Mejora en el proceso de autorización. Uso de XACML con el Identity Server de WSO2.* [En línea] 10 de 5 de 2012. [Citado el: 11 de 12 de 2013.]

12. **ForgeRock OpenIDM.** ForgeRock OpenIDM 2.1. [En línea] 23 de 5 de 2013. [Citado el: 12 de 11 de 2013.] http://docs.forgerock.org/en/index.html?product=openidm&version=2.1.0.

13. **Craig, Marcos, Frost, Lana y Egloff, Andi.** OpenIDM 2.1.0-SNAPSHOT Notas de la versión. [En línea] 20 de 11 de 2013. [Citado el: 21 de 11 de 2013.] http://openidm.forgerock.org/doc/release-notes/index.html.

14. **Ping Identity Corporation.** PingFederate SSO and Identity Management for Customers, Partners and Employee. [En línea] [Citado el: 12 de 10 de 2013.] https://www.pingidentity.com/products/pingfederate/.

15. **Muñiz López., Viviana y León Martínez., Armando.** *Módulo administración para la aplicación de aprovisionamiento de usuarios del Sistema de Administración de Identidades.* La Habana : s.n., 2010.

16. **Azahares Mezquia, Mario y Pujol Vargas, Carlos.** *Gestor de Tareas para Aprovisionamiento de Usuarios.* La Habana : s.n., 2010.

17. **Verano Escalona, Wilhem Manuel y Martín Suárez, Ernesto Raúl.** *Módulo de Conectores para el Subsistema de Aprovisionamiento de Usuarios del Sistema de Administración de Identidades.* La Habana : s.n., 2011.

18. **Correa Bautista, José Carlos y Machado García, Yendry.** *SUBSISTEMA DE APROVISIONAMIENTO DE USUARIOS PARA EL SISTEMA DE ADMINISTRACIÓN DE IDENTIDADES.* Ciudad de la Habana : s.n., 2012.

19. **Letelier, Patricio y Penadés, María Carmen .** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : Universidad Politécnica de Valencia.

20. **González Duque, Raúl.** *Python para todos.*

21. **Martelli, Alex.** Python. Guía de referencia. [En línea] 2008. [Citado el: 3 de 11 de 2013.] http://dialnet.unirioja.es/servlet/libro?codigo=318613. ISBN: 978-84-415-2317-3 84-415-2317-7.

22. **Castro, Elizabeth.** *HTML con XHTML y CSS.* s.l. : ANAYA Multimedias.

23. **Sturm, Jack.** *Desarrollo de soluciones XML.* s.l. : Serie de programación Microsoft.

24. **Rivas Santos , Victor Manuel.** Tutorial de XPath. [En línea] Enero de 2001. [Citado el: 20 de 11 de 2013.] http://geneura.ugr.es/~victor/cursillos/xml/XPath/.

25. **Flanagan, David.** *JavaScript: The Definitive Guide.* s.l. : O'Reily, 2006. 978-0-596-10199-2.

26. **Bradenbaugh, Jerry.** *Aplicaciones JavaScript.* Madrid : ANAYA Multimedia, 2000. 84-415-1070-9.

27. **Wium Lie, Håkon.** *Cascading Style Sheets.* Oslo : University of Oslo, 2005. 1501-7710.

28. **Holovaty, Adrian y Kaplan-Moss, Jacob.** *The Definitive Guide to Django: Web Development Done Right.* Nueva York : Apress, 2009. 978-1-4302-1937-8.

29. **Alvarez, Migue Angel.** *Manual de jQuery.* 2012.

30. **Silva, Vladimir.** *Practical Eclipse Rich Client Platform Projects.* New York : Apress, 2009. 978-1-4302-18128-9.

31. **Comunidad PostgreSQL.** PostgreSQL: About. [En línea] [Citado el: 12 de 10 de 2013.] http://www.postgresql.org/about/.

32. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El Lenguaje Unificado de Modelado. Manual de referencia.* s.l. : Addison - Wesley, 2000. 8478290281.

33. **Rosales Morales, Yanet, Eduardo Marrero , Michael y Trujillo Oliva, Adrian.** *Extensión de la herramienta Visual Paradigm para la generación de clases de acceso a datos con Doctrine 2.0.* 10, La Habana : Ediciones Futuro, 2013, Vol. 6. 2306-2495.

34. **Larman, Craig.** *UML Y PATRONES INTRODUCCION AL ANALISIS Y DISEÑO ORIENTADO A OBJETOS.* s.l. : Prentice Hall, 1999. 970-1 7-0261-1.

35. **Sommerville, Ian.** *INGENIERIA DE SOFTWARE. Séptima edición.* Madrid : PEARSON EDUCACION, 2005. 84-7829-074-5.

36. **Betancourt Rodríguez, Iván.** *Serie Científica de la Universidad de las Ciencias Informáticas.* La Habana : Ediciones Futuro, 2013. 2306-2495.

37. **Joskowicz, José.** *Reglas y Prácticas en eXtreme Programming.* Vigo : s.n., 2008.

38. **Pressman, Roger.** *INGENIERIA DE SOFTWARE. Un enfoque práctico (5ta edición).* Madrid : The McGraw-Hill Companies, 2002. 0-07-709677-0.

39. **Guerrero, Carlos, Suárez, Johanna y Gutiérrez, Luz.** Información tecnológica. *Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web.* [En línea] 28 de 1 de 2013. [Citado el: 7 de 2 de 2014.] http://www.scielo.cl/scielo.php?pid=S0718-07642013000300012&script=sci_arttext. 0718-0764.

40. **Unidad Docente de Ingeniería del Software, Facultad de informática.** *Patrones del "Gang of Four".* Madrid : s.n.

41. **Beck, Kent.** *Una explicación de la programación extrema: aceptar el cambio.* Madrid : ADDISON-WESLEY, 2002. 9788478290550.

42. **OMG.** *OMG Unified Modeling Language.* s.l. : OMG, 1999.

43. **Pressman, Roger.** *INGENIERÍA DEL SOFTWARE un enfoque práctico. (6ta edición).* Madrid : The McGraw-Hill Companies, 2005. 9701054733.

44. **Yagüe, Agurtin y Garbajosa, Juan.** *Revista Española de Innovación, Calidad e Ingeniería del Software.* 4, Madrid : ATI, 2009, Vol. 5. para validar las necesidades de los usuarios y para dirigir la implementación.

GLOSARIO DE TÉRMINOS

A

ACID: (*Atomicity, Consistency, Isolation and Durability*, en español, Atomicidad, Consistencia, Aislamiento y Durabilidad). En bases de datos se denomina ACID a un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción.

Ajax: (*Asynchronous JavaScript And XML*, en español, JavaScript Asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

API: (*Application Programming Interface*, en español, Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Aplicación: Una aplicación, en términos informáticos, es un tipo de software que permite al usuario realizar uno o más tipos de trabajo.

Arquitectura: En términos informáticos, una arquitectura es una imagen que indica la estructura, funcionamiento e interacción entre las partes de un software.

B

Base de datos: Es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

C

Contraseña: Es una combinación de letras, símbolos y/o números que brinda, a quien lo conoce, la posibilidad de acceder a un recurso.

Correo electrónico: El correo electrónico (también conocido como *e-mail*, un término inglés derivado de *electronic mail*) es un servicio que permite el intercambio de mensajes a través de sistemas de comunicación electrónicos.

Cuenta de usuario: Una cuenta de usuario es una colección de información que indica los archivos y carpetas a los que se puede obtener acceso. En este sentido, tener una cuenta en una aplicación informática significa haberse dado de alta en su base de datos, formar parte de su cartera de usuarios.

D

Directorio de usuarios: Es una base de datos que almacena de forma organizada, información sobre los usuarios de una red de ordenadores y sobre este directorio, los administradores pueden gestionar el acceso de los usuarios a los recursos de la red.

DOM: (*Document Object Model*, en español, Modelo de Objetos del Documento) es esencialmente una API que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos.

H

Hipertexto: Es una herramienta de software que permite crear, agregar, enlazar y compartir información de diversas fuentes por medio de enlaces asociativos. La forma más habitual de hipertexto en informática es la de hipervínculos o referencias cruzadas automáticas que van a otros documentos.

I

Identidad federada: La identidad federada es una de las soluciones para abordar la gestión de identidad en los sistemas de información. El valor añadido adicional respecto a otras soluciones es la gestión de identidad interdependiente entre compañías.

Infraestructura Tecnológica: Conjunto de elementos tecnológicos *hardware* y software.

Interfaz: Es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora. Comprende todos los puntos de contacto entre el usuario y el equipo. Se trata de todos aquellos elementos presentes en pantalla que permitan la navegación por el programa.

Internet: Es una red de redes que permite la interconexión descentralizada de computadoras a través de un conjunto de protocolos denominado TCP/IP.

L

Lenguaje de Programación: Es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras.

M

Marco de trabajo: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos. Puede servir de base para la organización y desarrollo de software.

Mensajería instantánea: Comunicación escrita que se desarrolla en tiempo real a través de Internet. Intercambio de mensajes escritos de manera instantánea.

Motor de aprovisionamiento: Es uno de los componentes básicos existentes dentro de un sistema de aprovisionamiento de cuentas de usuario. Administra la creación, modificación y eliminación de cuentas de usuario en un sistema de aprovisionamiento.

N

Nube: Desde el punto de vista informático, es un paradigma que permite ofrecer servicios de computación a través de Internet.

O

OpenLDAP: Es una implementación libre y de código abierto del protocolo *Lightweight Directory Access Protocol* (LDAP).

P

PKI: (*Public Key Infrastructure*, en español, Infraestructura de Clave Pública) Es una tecnología que permite a los usuarios autenticarse frente a otros usuarios y usar la información de los certificados de identidad (por ejemplo, las claves públicas de otros usuarios) para cifrar y descifrar mensajes, firmar digitalmente información y garantizar el no repudio de un envío.

Protocolo: Normativas y criterios que fijan cómo deben comunicarse los diversos componentes de un cierto sistema de interconexión. Esto quiere decir que, a través de este protocolo, los dispositivos que se conectan en red pueden intercambiar datos.

R

Repositorio de datos: Es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos.

S

SaaS: (Software *as a Service,* en español, Software como Servicio) Modelo de distribución de software donde el soporte lógico y los datos que se manejan se alojan en servidores de una compañía de tecnologías de información y comunicación, a los que se accede con un navegador web desde un cliente, a través de Internet.

SCIM: (*System for Cross-domain Identity Management,* en español, Sistema de Gestión de Identidad entre Dominios) Especificación diseñada para hacer más fácil y barata la gestión de identidades basadas en la nube. Proporciona esquemas de usuario comunes y un modelo que provee pautas para el intercambio de estos esquemas mediante protocolos estándares.

Script: Es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. El uso habitual de los scripts es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario.

Servidor: En informática, un servidor es una computadora que, formando parte de una red, provee servicios a otros denominados clientes.

Sistema operativo: Conjunto de programas informáticos que permite la administración eficaz de los recursos de una computadora permitiendo la interacción entre los elementos de hardware y el resto de los programas.

SOAP: (*Simple Object Access Protocol*, en español, Protocolo de Acceso Simple a Objetos) Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Software: El software es un conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora. Equipamiento lógico e intangible de un ordenador.

Software libre: Un programa es libre si el usuario puede: ejecutarlo como desee, estudiar su código fuente y modificarlo, distribuir copias exactas del programa y distribuir copias de las versiones cambiadas.

SPML: (*Service Provisioning Markup Language*, en español, Lenguaje de Marcado de Servicio de Aprovisionamiento) Es un marco basado en XML para el intercambio de usuarios, recursos e información de servicios de aprovisionamiento entre organizaciones cooperantes.

T

Transacción: Es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica, manteniendo la integridad de los datos y haciendo que estas transacciones no puedan finalizar en un estado intermedio.

U

Usuario: En el contexto de la informática, un usuario es aquel que utiliza un sistema informático.

X

XACML: (*eXtensible Access Control Markup Language*, en español, Lenguaje de Marcado Extensible de Control de Acceso) Define un lenguaje de políticas de control de acceso declarativo implementado en XML y un modelo de procesamiento para evaluar las solicitudes de acceso de acuerdo con las reglas definidas en las políticas.

ANEXOS

4.1. Anexo 1: Historias de usuario

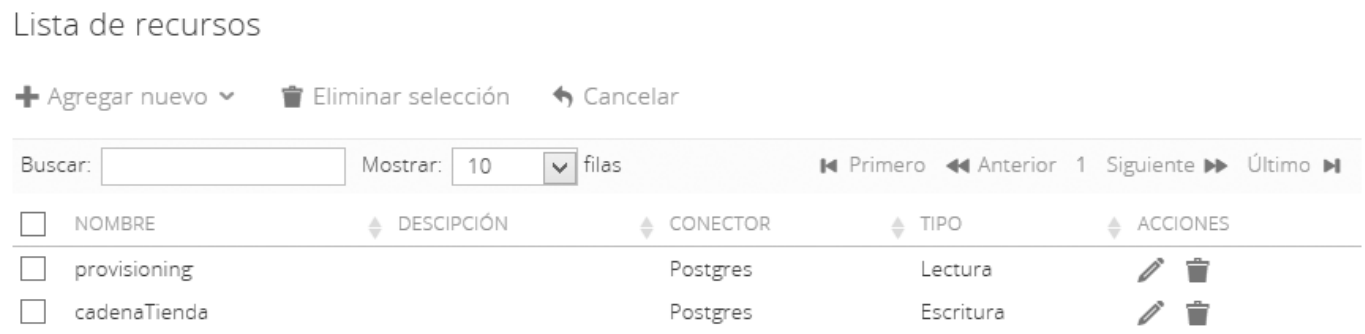
Historia de usuario	
Id.: 1.	Nombre: Gestionar recursos.
Prioridad técnica: Alta.	Complejidad de desarrollo: Alta.
Puntos estimados: 3.	Iteración asignada: 1.
Descripción: Cosiste en crear, editar, eliminar y listar las conexiones a los recursos.	
Observaciones: -	
Prototipo: 	

Tabla 28: Historia de usuario "Gestionar recursos".

Historia de usuario	
Id.: 2.	Nombre: Gestionar plantillas.
Prioridad técnica: Alta.	Complejidad de desarrollo: Alta.
Puntos estimados: 3.	Iteración asignada: 2.
Descripción: Se le brinda al usuario la posibilidad de crear, editar, eliminar y listar las plantillas empleadas para la creación de solicitudes de cuentas de usuario.	

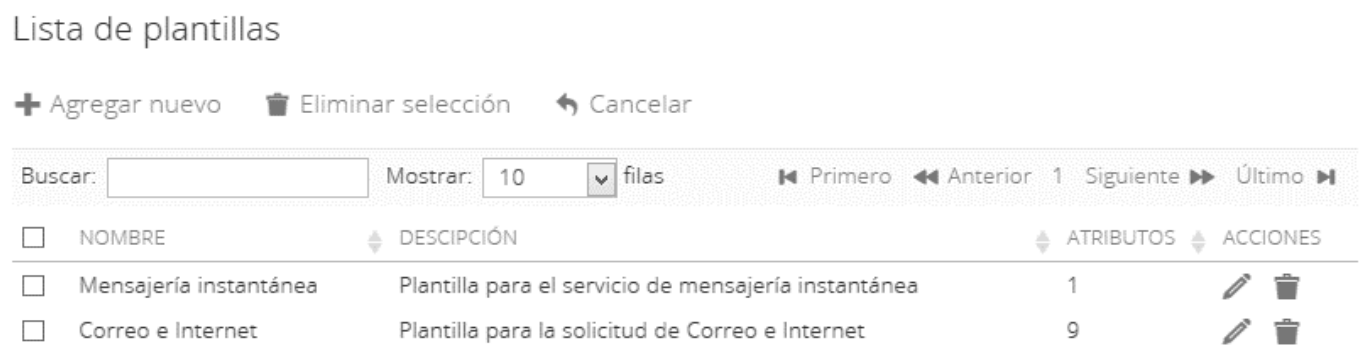
Observaciones: -
Prototipo: 

Tabla 29: Historia de usuario "Gestionar plantillas".

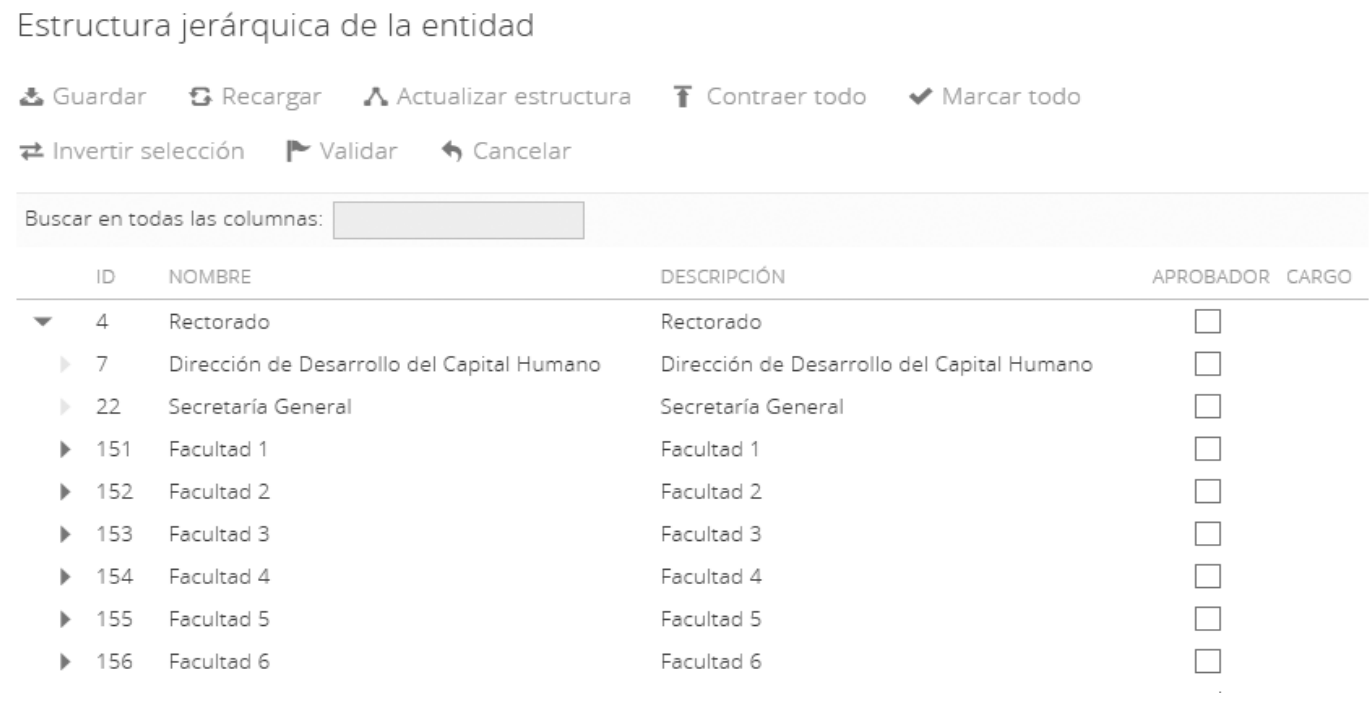
Historia de usuario	
Id.: 3.	Nombre: Administrar estructura de la entidad.
Prioridad técnica: Alta.	Complejidad de desarrollo: Alta.
Puntos estimados: 2.	Iteración asignada: 3.
Descripción: Brinda las posibilidades de mostrar y de actualizar la estructura de la entidad. La estructura es una representación jerárquica de las áreas de la entidad.	
Observaciones: Se debe visualizar a través de un árbol desplegable.	
Prototipo: 	

Tabla 30: Historia de usuario "Administrar estructura de la entidad".

Historia de usuario	
Id.: 4.	Nombre: Definir aprobadores.
Prioridad técnica: Alta.	Complejidad de desarrollo: Baja.
Puntos estimados: 0.2.	Iteración asignada: 3.
Descripción: Permite definir los aprobadores que deben revisar las solicitudes de cuentas de usuario.	
Observaciones: Se debe visualizar a través del árbol de la estructura de la entidad.	

Prototipo:

Tabla 31: Historia de usuario "Definir aprobadores".

Historia de usuario	
Id.: 5.	Nombre: Gestionar flujos de aprobación.
Prioridad técnica: Media.	Complejidad de desarrollo: Alta.
Puntos estimados: 3.	Iteración asignada: 4.
Descripción: Se podrá crear, editar, eliminar y listar flujos de aprobación.	

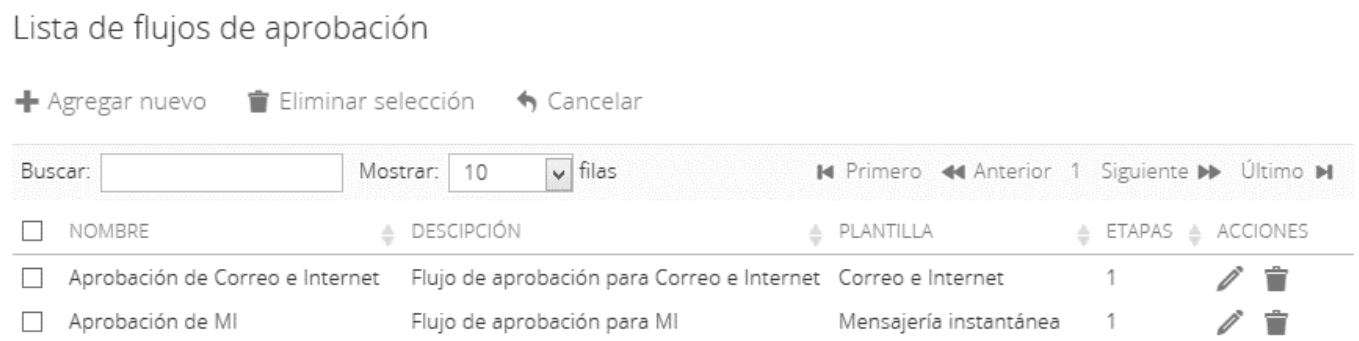
Observaciones: En el caso de la creación y edición se debe mostrar una interfaz que facilite el trabajo en la mayor medida posible.
Prototipo: 

Tabla 32: Historia de usuario "Gestionar flujos de aprobación".

Historia de usuario	
Id.: 9.	Nombre: Gestionar grupos de usuarios.
Prioridad técnica: Baja.	Complejidad de desarrollo: Media.
Puntos estimados: 1.	Iteración asignada: 8.
Descripción: Consiste en crear, editar, eliminar y listar grupos de usuarios en el sistema. Para los casos de la creación y edición permite además asociar usuarios y permisos.	
Observaciones: -	
Prototipo:  	

Tabla 33: Historia de usuario "Gestionar grupos de usuarios".

Historia de usuario	
Id.: 10.	Nombre: Autenticar usuario.
Prioridad técnica: Baja.	Complejidad de desarrollo: Baja.
Puntos estimados: 0.3.	Iteración asignada: 8.

Descripción: Cuando un usuario inicia sesión, el sistema le brinda las opciones correspondientes a su nivel de acceso. Los usuarios también pueden cerrar sesión.
Observaciones: -
Prototipo: 

Tabla 34: Historia de usuario "Autenticar usuario".

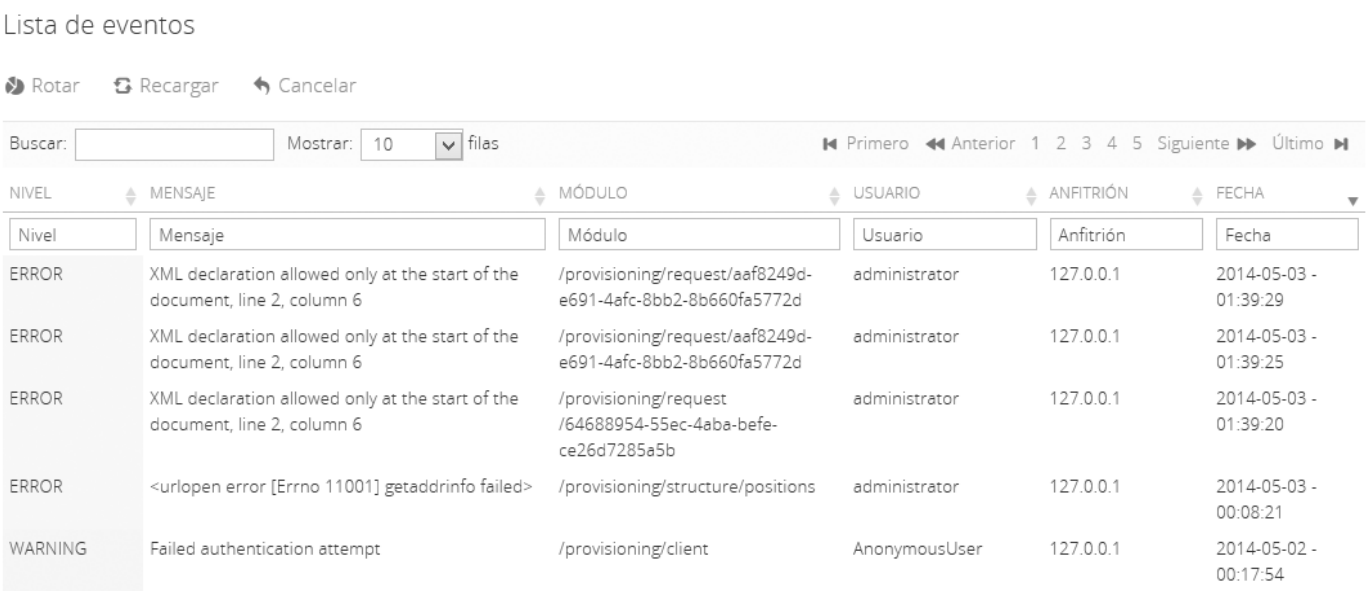
Historia de usuario	
Id.: 11.	Nombre: Administrar registros de sucesos del sistema.
Prioridad técnica: Baja.	Complejidad de desarrollo: Baja.
Puntos estimados: 0.4.	Iteración asignada: 8.
Descripción: Brinda la posibilidad de listar y rotar los registros de sucesos del sistema.	
Observaciones: -	
Prototipo: 

Tabla 35: Historia de usuario "Administrar registros de sucesos del sistema".

Historia de usuario	
Id.: 12.	Nombre: Buscar información en un listado de datos.
Prioridad técnica: Baja.	Complejidad de desarrollo: Baja.
Puntos estimados: 0.1.	Iteración asignada: 8.
Descripción: Permite realizar búsquedas en las listas que muestra el sistema.	
Observaciones: -	
Prototipo: 	

Tabla 36: Historia de usuario "Buscar información en un listado de datos".

4.2. Anexo 2: Tarjetas CRC.

Tarjeta CRC	
Clase Logger.	
Responsabilidades	**Colaboradores**
Generar información adicional a los registros de sucesos del sistema.	
Crear un registro de sucesos del sistema que sea genérico especificando el nivel de severidad.	
Crear registro de sucesos del sistema de nivel “Información”.	
Crear registro de sucesos del sistema de nivel “Advertencia”.	
Crear registro de sucesos del sistema de nivel “Error”.	
Crear registro de sucesos del sistema de nivel “Error”, con información acerca de la ejecución.	
Crear registro de sucesos del sistema de nivel “Critico”.	

Tabla 37: Tarjeta CRC "Logger".

Tarjeta CRC	
Clase XmlMapper.	
Responsabilidades	**Colaboradores**

Transformar un archivo XML a partir de un archivo XSL y devolver el resultado en un archivo HTML.	
Validar si un archivo XML corresponde a un XSD dado.	
Obtener un objeto XML a partir de una cadena de texto.	
Aplicar una sentencia Xpath a un archivo XML y devolver el resultado.	

Tabla 38: Tarjeta CRC "XmlMapper".

Tarjeta CRC	
Clase MySQLConnection.	
Responsabilidades	**Colaboradores**
Abrir conexión.	
Cerrar conexión.	
Probar conexión.	

Tabla 39: Tarjeta CRC "MySQLConnection".

Tarjeta CRC	
Clase MySQLConnector.	
Responsabilidades	**Colaboradores**
Abrir conexión.	
Cerrar conexión.	
Probar conexión.	
Obtener el formulario correspondiente al conector de MySQL.	
Definir parámetros de conexión.	MySQLConnection
Recorrer resultados de las consultas.	
Obtener los usuarios del recurso.	
Crear usuario en el recurso.	
Eliminar usuario en el recurso.	
Habilitar usuario en el recurso.	
Deshabilitar usuario en el recurso.	
Dar privilegios a un usuario.	

Quitar privilegios a un usuario.	
Buscar usuarios por un criterio dado.	
Obtener la lista de tablas del recurso.	
Obtener las columnas de una tabla dada.	
Obtener todos los datos de una columna.	
Generar una consulta personalizada.	
Ejecutar consulta.	

Tabla 40: Tarjeta CRC "MySQLConnector".

Tarjeta CRC	
Clase ManagerDocument.	
Responsabilidades	**Colaboradores**
Adicionar documento.	Document
Eliminar documento.	Document
Actualizar documento.	Document
Obtener documento por identificador.	Document
Obtener documentos por nombre.	Document
Obtener todos los documentos.	Document
Obtener documento por identificador de solicitud.	Document

Tabla 41: Tarjeta CRC "ManagerDocument".

Tarjeta CRC	
Clase ManagerElement.	
Responsabilidades	**Colaboradores**
Adicionar elemento.	Element.
Eliminar elemento.	Element.
Actualizar elemento.	Element.
Obtener elemento por identificador.	Element.
Obtener elementos por tipo.	Element.
Obtener todos los elementos.	Element.

Obtener etiquetas XML específicas por identificador de elemento.	

Tabla 42: Tarjeta CRC "ManagerElement".

Tarjeta CRC	
Clase ManagerRelation.	
Responsabilidades	**Colaboradores**
Adicionar relación.	Relation
Eliminar relación.	Relation.
Actualizar relación.	Relation.
Obtener relación por identificador.	Relation.
Obtener relaciones por tipo.	Relation.
Obtener todas las relaciones.	Relation.
Obtener relaciones por identificador de elemento principal.	Relation.
Obtener relaciones por identificador de elemento dependiente.	Relation.
Obtener relaciones por tipo de relación e identificador de elemento principal.	Relation.
Obtener relaciones por tipo de elemento principal.	Relation.
Obtener relaciones por tipo de elemento principal y tipo de elemento dependiente.	Relation.
Obtener relaciones por tipo de elemento dependiente.	Relation.
Obtener relaciones por identificador de elemento principal y tipo de elemento dependiente.	Relation.
Obtener relaciones por identificador de elemento principal y tipo de elemento dependiente.	Relation.
Obtener relaciones por identificador de elemento dependiente y tipo de elemento principal.	Relation.
Obtener relaciones por identificador de elemento principal e identificador de elemento dependiente.	Relation.

Tabla 43: Tarjeta CRC "ManagerRelation".

Tarjeta CRC
Clase ManagerUserProfile.

Responsabilidades	Colaboradores
Obtener área de un usuario por su nombre de usuario.	UserProfile.
Obtener área de un usuario dados sus datos registrados.	UserProfile.
Obtener cargo de un usuario por su nombre de usuario.	UserProfile.
Obtener cargo de un usuario dados sus datos registrados.	UserProfile.
Obtener área y cargo de un usuario por su nombre de usuario.	
Obtener área y cargo de un usuario dados sus datos registrados.	

Tabla 44: Tarjeta CRC "ManagerUserProfile".

4.3. Anexo 3: Tareas de programación.

Tarea de programación	
Número de HU: 1.	Id. de tarea: 1.1.
Nombre de la tarea: Crear conectores.	
Fecha de inicio: 25/10/2013.	Fecha de fin: 31/10/2013.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 1.
Descripción: Se crean un conjunto de conectores necesarios para acceder a los recursos.	

Tabla 45: Tarea de programación "Crear conectores".

Tarea de programación	
Número de HU: 1.	Id. de tarea: 1.2.
Nombre de la tarea: Crear recurso.	
Fecha de inicio: 1/11/2013.	Fecha de fin: 7/11/2013.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 1.
Descripción: El sistema muestra un formulario que solicita los datos necesarios para conectarse a un recurso, el usuario los introduce y finaliza la operación. El sistema guarda los cambios.	

Tabla 46: Tarea de programación "Crear recurso".

Tarea de programación	
Número de HU: 1.	Id. de tarea: 1.3.
Nombre de la tarea: Listar recursos.	

Fecha de inicio: 8/11/2013.	Fecha de fin: 8/11/2013.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.1.
Descripción: El sistema muestra un listado con todos los recursos a partir del cual se pueden llevar a cabo el resto de las acciones sobre los mismos.	

Tabla 47: Tarea de programación "Listar recursos".

Tarea de programación	
Número de HU: 1.	Id. de tarea: 1.4.
Nombre de la tarea: Editar recurso.	
Fecha de inicio: 9/11/2013.	Fecha de fin: 12/11/2013.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.4.
Descripción: El sistema muestra todos los datos del recurso que pueden ser modificados, el usuario realiza modificaciones y finaliza la edición. El sistema guarda los cambios.	

Tabla 48: Tarea de programación "Editar recurso".

Tarea de programación	
Número de HU: 1.	Id. de tarea: 1.5.
Nombre de la tarea: Eliminar recurso.	
Fecha de inicio: 13/11/2013.	Fecha de fin: 14/11/2013.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.2.
Descripción: El sistema brinda la opción de eliminar un recurso mostrando un mensaje de confirmación. Si la respuesta del usuario es afirmativa, el sistema procede a eliminar dicho recurso.	

Tabla 49: Tarea de programación "Eliminar recurso".

Tarea de programación	
Número de HU: 2.	Id. de tarea: 2.1.
Nombre de la tarea: Crear plantilla.	
Fecha de inicio: 15/11/2013.	Fecha de fin: 27/11/2013.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 1.6.
Descripción: El sistema muestra un formulario con los datos requeridos para la creación de una plantilla. Luego de que el usuario introduzca toda la información necesaria y finalice la creación de la plantilla, el sistema la guarda.	

Tabla 50: Tarea de programación "Crear plantilla".

Tarea de programación	
Número de HU: 2.	Id. de tarea: 2.2.
Nombre de la tarea: Listar plantillas.	
Fecha de inicio: 28/11/2013.	Fecha de fin: 29/11/2013.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.2.
Descripción: El sistema muestra un listado con todas las plantillas, a partir del cual se pueden llevar a cabo el resto de las acciones sobre las mismas.	

Tabla 51: Tarea de programación "Listar plantillas".

Tarea de programación	
Número de HU: 2.	Id. de tarea: 2.3.
Nombre de la tarea: Editar plantilla.	
Fecha de inicio: 30/11/2013.	Fecha de fin: 3/12/2013.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.4.
Descripción: El sistema muestra todos los datos de la plantilla que pueden ser modificados, el usuario realiza modificaciones y finaliza la edición. El sistema guarda los cambios.	

Tabla 52: Tarea de programación "Editar plantilla".

Tarea de programación	
Número de HU: 2.	Id. de tarea: 2.4.
Nombre de la tarea: Eliminar plantilla.	
Fecha de inicio: 4/12/2013.	Fecha de fin: 5/12/2013.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.2.
Descripción: El sistema brinda la opción de eliminar una plantilla mostrando un mensaje de confirmación. Si la respuesta del usuario es afirmativa, el sistema procede a eliminar dicha plantilla.	

Tabla 53: Tarea de programación "Eliminar plantilla".

Tarea de programación	
Número de HU: 3.	Id. de tarea: 3.1.
Nombre de la tarea: Mostrar estructura de la entidad.	
Fecha de inicio: 6/12/2013.	Fecha de fin: 16/12/2013.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 1.4.

Descripción: El sistema muestra una interfaz con la estructura de la entidad.

Tabla 54: Tarea de programación "Mostrar estructura de la entidad".

Tarea de programación	
Número de HU: 3.	Id. de tarea: 3.2.
Nombre de la tarea: Actualizar estructura de la entidad.	
Fecha de inicio: 17/12/2013.	Fecha de fin: 19/12/2013.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.3.
Descripción: El sistema muestra una opción para actualizar la estructura, el usuario la actualiza, luego el sistema la guarda.	

Tabla 55: Tarea de programación "Actualizar estructura de la entidad".

Tarea de programación	
Número de HU: 4.	Id. de tarea: 4.1.
Nombre de la tarea: Definir aprobadores.	
Fecha de inicio: 20/12/2013.	Fecha de fin: 13/1/2014
Programador responsable: Lianet Suárez Martínez.	Tiempo estimado: 0.2.
Descripción: El sistema muestra el árbol de la estructura de la entidad y el usuario marca las áreas a las que pertenecen los aprobadores que deben revisar las solicitudes. Luego el usuario selecciona el aprobador de cada área y finaliza la operación. El sistema guarda los cambios.	

Tabla 56: Tarea de programación "Definir aprobadores".

Tarea de programación	
Número de HU: 5.	Id. de tarea: 5.1.
Nombre de la tarea: Crear flujo de aprobación.	
Fecha de inicio: 14/1/2014.	Fecha de fin: 25/1/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 1.5.
Descripción: El sistema muestra las opciones necesarias para la creación de un flujo de aprobación, el usuario introduce todos los datos necesarios y finaliza su creación. El sistema guarda los cambios.	

Tabla 57: Tarea de programación "Crear flujo de aprobación".

Tarea de programación	
Número de HU: 5.	Id. de tarea: 5.2.

Nombre de la tarea: Listar flujos de aprobación.	
Fecha de inicio: 26/1/2014.	Fecha de fin: 27/1/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.2.
Descripción: El sistema muestra un listado con todos los flujos de aprobación, a partir del cual se pueden llevar a cabo el resto de las acciones sobre estos elementos.	

Tabla 58: Tarea de programación "Listar flujos de aprobación".

Tarea de programación	
Número de HU: 5.	Id. de tarea: 5.3.
Nombre de la tarea: Editar flujo de aprobación.	
Fecha de inicio: 28/1/2014.	Fecha de fin: 31/1/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.4.
Descripción: El sistema muestra todos los datos del flujo de aprobación que pueden ser modificados, el usuario realiza modificaciones y finaliza la edición. El sistema guarda los cambios.	

Tabla 59: Tarea de programación "Editar flujo de aprobación".

Tarea de programación	
Número de HU: 5.	Id. de tarea: 5.4.
Nombre de la tarea: Eliminar flujo de aprobación.	
Fecha de inicio: 1/2/2014.	Fecha de fin: 3/2/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.3.
Descripción: El sistema brinda la opción de eliminar un flujo de aprobación mostrando un mensaje de confirmación. Si la respuesta del usuario es afirmativa, el sistema procede a eliminar dicho flujo de aprobación.	

Tabla 60: Tarea de programación "Eliminar flujo de aprobación".

Tarea de programación	
Número de HU: 8.	Id. de tarea: 8.1.
Nombre de la tarea: Crear cuenta de usuario.	
Fecha de inicio: 18/3/2014.	Fecha de fin: 26/3/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 1.2.
Descripción: El sistema muestra el listado de solicitudes pendientes de creación, el usuario oprime la opción necesaria para crear la cuenta. Posteriormente el sistema crea la cuenta de usuario otorgándole acceso en los recursos definidos en la plantilla.	

Tabla 61: Tarea de programación "Crear cuenta de usuario".

Tarea de programación	
Número de HU: 8.	Id. de tarea: 8.2.
Nombre de la tarea: Listar cuentas de usuario.	
Fecha de inicio: 27/3/2014.	Fecha de fin: 29/3/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.3.
Descripción: El sistema muestra un listado con todas las cuentas de usuario, a partir del cual se pueden llevar a cabo el resto de las acciones sobre las mismas.	

Tabla 62: Tarea de programación "Listar cuentas de usuario".

Tarea de programación	
Número de HU: 8.	Id. de tarea: 8.3.
Nombre de la tarea: Listar últimas cuentas de usuario.	
Fecha de inicio: 30/3/2014.	Fecha de fin: 2/4/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.4.
Descripción: El sistema muestra un listado con las cuentas de usuario creadas desde una fecha seleccionada hasta la fecha actual.	

Tabla 63: Tarea de programación "Listar últimas cuentas de usuario".

Tarea de programación	
Número de HU: 8.	Id. de tarea: 8.4.
Nombre de la tarea: Eliminar cuenta de usuario.	
Fecha de inicio: 3/4/2014.	Fecha de fin: 7/4/2014.
Programador responsable: Pablo David Gago Ballester.	Tiempo estimado: 0.5.
Descripción: El sistema brinda la posibilidad de eliminar una cuenta de usuario mostrando un mensaje de confirmación. Si la respuesta del usuario es afirmativa, el sistema procede a eliminar dicha cuenta de usuario.	

Tabla 64: Tarea de programación "Eliminar cuenta de usuario".

Tarea de programación	
Número de HU: 9.	Id. de tarea: 9.1.
Nombre de la tarea: Crear grupo de usuarios.	
Fecha de inicio: 8/4/2014.	Fecha de fin: 10/4/2014.
Programador responsable: Lianet Suárez Martínez.	Tiempo estimado: 0.3.

Descripción: El sistema muestra el listado de usuarios y el listado de permisos que pueden conformar el nuevo grupo de usuarios. El usuario introduce los datos y finalmente el sistema guarda la información.

Tabla 65: Tarea de programación "Crear grupo de usuarios".

Tarea de programación	
Número de HU: 9.	Id. de tarea: 9.2.
Nombre de la tarea: Listar grupos de usuarios.	
Fecha de inicio: 11/4/2014.	Fecha de fin: 11/4/2014.
Programador responsable: Lianet Suárez Martínez.	Tiempo estimado: 0.1.
Descripción: El sistema muestra un listado con todos los grupos de usuarios, a partir del cual se pueden llevar a cabo el resto de las acciones sobre los mismos.	

Tabla 66: Tarea de programación "Listar grupos de usuarios".

Tarea de programación	
Número de HU: 9.	Id. de tarea: 9.3.
Nombre de la tarea: Editar grupo de usuarios.	
Fecha de inicio: 21/4/2014.	Fecha de fin: 22/4/2014.
Programador responsable: Lianet Suárez Martínez.	Tiempo estimado: 0.2.
Descripción: El sistema muestra una interfaz en la que el usuario puede agregar o eliminar usuarios y permisos del grupo que se está editando.	

Tabla 67: Tarea de programación "Editar grupo de usuarios".

Tarea de programación	
Número de HU: 9.	Id. de tarea: 9.4.
Nombre de la tarea: Eliminar grupo de usuarios.	
Fecha de inicio: 23/4/2014.	Fecha de fin: 23/4/2014.
Programador responsable: Lianet Suárez Martínez.	Tiempo estimado: 0.1.
Descripción: El sistema brinda la opción de eliminar un grupo mostrando un mensaje de confirmación. Si la respuesta del usuario es afirmativa, el sistema procede a eliminar dicho grupo.	

Tabla 68: Tarea de programación "Eliminar grupo de usuarios".

Tarea de programación	
Número de HU: 10.	Id. de tarea: 10.1.

Nombre de la tarea: Iniciar sesión en el sistema.	
Fecha de inicio: 24/4/2014.	Fecha de fin: 25/4/2014.
Programador responsable: Lianet Suárez Martínez.	Tiempo estimado: 0.2.
Descripción: El sistema brinda la posibilidad de iniciar sesión. El usuario se identifica, a continuación el sistema verifica la identidad del usuario y si es correcta inicia una sesión con las funcionalidades correspondientes a su nivel de acceso.	

Tabla 69: Tarea de programación "Iniciar sesión en el sistema".

Tarea de programación	
Número de HU: 10.	Id. de tarea: 10.2.
Nombre de la tarea: Cerrar sesión en el sistema.	
Fecha de inicio: 26/4/2014.	Fecha de fin: 26/4/2014.
Programador responsable: Lianet Suárez Martínez.	Tiempo estimado: 0.1.
Descripción: El sistema brinda la posibilidad de cerrar sesión, regresando a la página de inicio en la que se pude ver el listado de cuentas creadas recientemente e iniciar sesión.	

Tabla 70: Tarea de programación "Cerrar sesión en el sistema".

Tarea de programación	
Número de HU: 11.	Id. de tarea: 11.1.
Nombre de la tarea: Listar registros de sucesos del sistema.	
Fecha de inicio: 27/4/2014.	Fecha de fin: 28/4/2014.
Programador responsable: Lianet Suárez Martínez.	Tiempo estimado: 0.2.
Descripción: El sistema muestra un listado con los registros de sucesos del sistema.	

Tabla 71: Tarea de programación "Listar registros de sucesos del sistema".

Tarea de programación	
Número de HU: 11.	Id. de tarea: 11.2.
Nombre de la tarea: Rotar registros de sucesos del sistema.	
Fecha de inicio: 29/4/2014.	Fecha de fin: 30/4/2014.
Programador responsable: Lianet Suárez Martínez.	Tiempo estimado: 0.2.
Descripción: El sistema muestra el listado con los registros de sucesos del sistema, el usuario oprime la opción para rotarlos y el sistema elimina todos los registros que no corresponden al mes actual.	

Tabla 72: Tarea de programación "Rotar registros de sucesos del sistema".

Tarea de programación	
Número de HU: 12.	Id. de tarea: 12.1.
Nombre de la tarea: Buscar información en un listado de datos.	
Fecha de inicio: 1/5/2014.	Fecha de fin: 1/5/2014.
Programador responsable: Lianet Suárez Martínez.	Tiempo estimado: 0.1.
Descripción: El sistema muestra la opción para buscar en cada uno de los listados, el usuario introduce un dato correspondiente a cualquiera de los campos del listado, el sistema hace una búsqueda y muestra los resultados obtenidos.	

Tabla 73: Tarea de programación "Buscar información en un listado de datos".

4.4. Anexo 4: Pruebas unitarias.

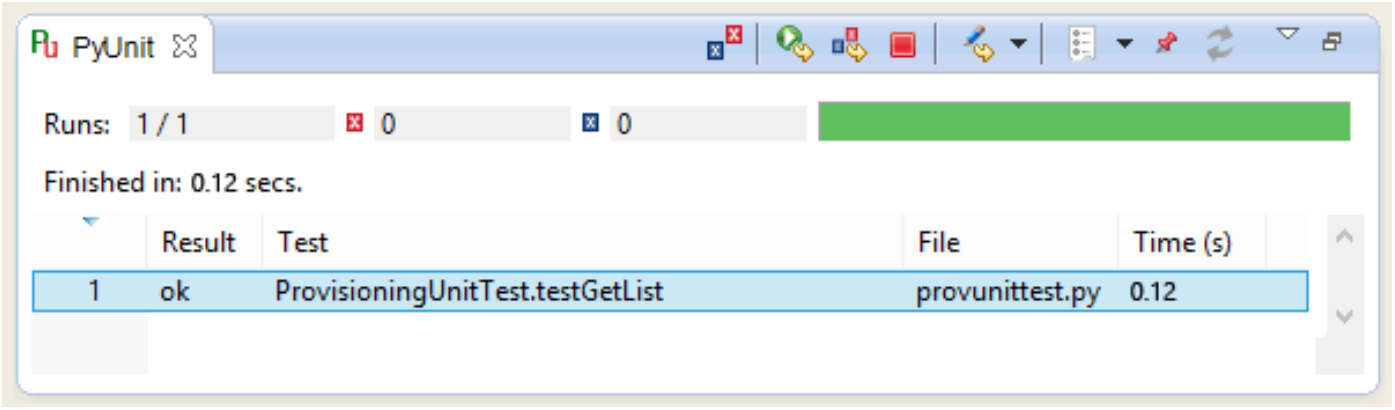
Caso de prueba unitaria	
Nombre de prueba: TestGetList.	Id. de prueba: 1.
Descripción del caso de prueba: Permite verificar que se genera correctamente un arreglo de diccionarios que contiene toda la información acerca de los elementos de la tabla “element” que corresponden a un tipo específico.	
Entradas: Tipo de elemento.	
Criterio de aceptación: El arreglo de diccionarios obtenido es igual al esperado.	
Resultado: 	

Tabla 74: Caso de prueba unitaria "TestGetList".

Caso de prueba unitaria	
Nombre de prueba: TestRequestRelations.	Id. de prueba: 2.
Descripción del caso de prueba: Permite validar que los elementos de tipo: flujo de aprobación, plantilla y recursos de escritura, asociados a una solicitud se obtienen correctamente.	
Entradas: Valor del identificador de una solicitud.	

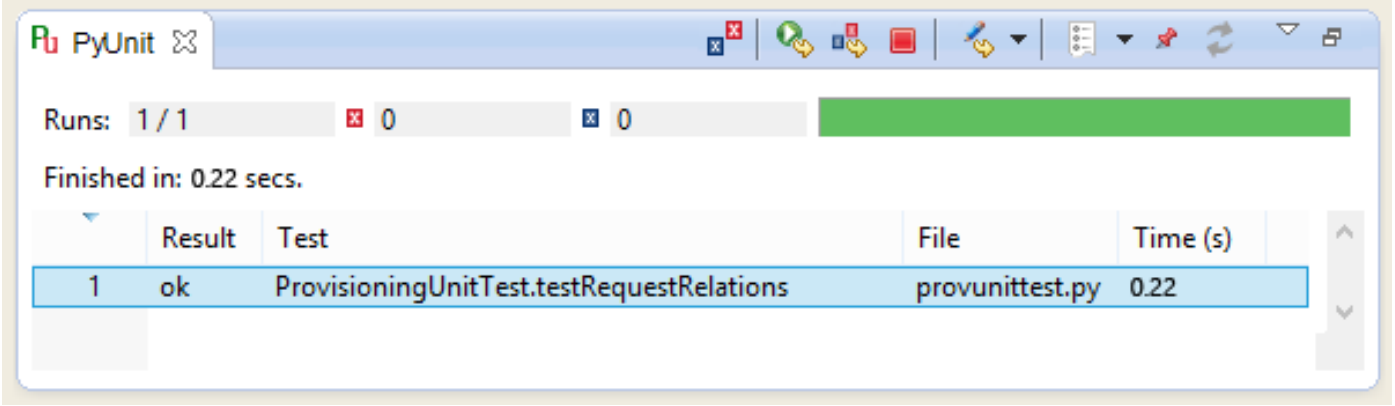
Criterio de aceptación: El diccionario obtenido es igual al esperado.
Resultado: 

Tabla 75: Caso de prueba unitaria "TestRequestRelations".

4.5. Anexo 5: Pruebas de aceptación.

Caso de prueba de aceptación	
Nombre de HU: Gestionar recursos.	Id. de prueba: 1.1.
Nombre del caso de prueba: Crear recurso.	
Descripción del caso de prueba: Permite verificar que los recursos se crean correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de recursos”. ▪ Pulsar la opción “Agregar nuevo”. ▪ Seleccionar el conector adecuado para el nuevo recurso. ▪ Llenar los campos necesarios. ▪ Pulsar la opción “Guardar”.	
Resultado esperado: El sistema muestra el nuevo recurso en el listado de recursos.	
Evaluación de la prueba: Satisfactoria.	

Tabla 76: Caso de prueba de aceptación "Crear recurso".

Caso de prueba de aceptación	
Nombre de HU: Gestionar recursos.	Id. de prueba: 1.2.
Nombre del caso de prueba: Listar recursos.	
Descripción del caso de prueba: Permite validar que el listado de recursos se muestra correctamente.	

Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de recursos”.
Resultado esperado: El sistema muestra el listado de recursos.
Evaluación de la prueba: Satisfactoria.

Tabla 77: Caso de prueba de aceptación "Listar recursos".

Caso de prueba de aceptación	
Nombre de HU: Gestionar recursos.	Id. de prueba: 1.3.
Nombre del caso de prueba: Editar recurso.	
Descripción del caso de prueba: Permite comprobar que la funcionalidad que permite editar los recursos se desempeña correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios. Debe existir al menos un recurso.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de recursos”. ▪ Pulsar la opción “Editar” correspondiente al recurso que se desea editar. ▪ Realizar las modificaciones necesarias. ▪ Pulsar la opción “Guardar”.	
Resultado esperado: El sistema muestra el listado de recursos. Cuando se vuelve a acceder al recurso editado, se observan los cambios antes realizados.	
Evaluación de la prueba: Satisfactoria.	

Tabla 78: Caso de prueba de aceptación "Editar recurso".

Caso de prueba de aceptación	
Nombre de HU: Gestionar recursos.	Id. de prueba: 1.4.
Nombre del caso de prueba: Eliminar recurso.	
Descripción del caso de prueba: Permite confirmar que los recursos se eliminan correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios. Debe existir al menos un recurso.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de recursos”. ▪ Eliminar recurso: - Variante 1: Pulsar la opción “Eliminar” correspondiente al recurso que se desea eliminar. - Variante 2: Seleccionar los recursos que se desean eliminar y pulsar la opción “Eliminar selección”. ▪ Pulsar la opción “Si” que aparece en la pregunta de confirmación.
Resultado esperado: El sistema muestra el listado de recursos en el cual no aparecen los eliminados.
Evaluación de la prueba: Satisfactoria.

Tabla 79: Caso de prueba de aceptación "Eliminar recurso".

Caso de prueba de aceptación	
Nombre de HU: Gestionar plantillas.	Id. de prueba: 2.1.
Nombre del caso de prueba: Crear plantilla.	
Descripción del caso de prueba: Permite verificar que la funcionalidad que crea las plantillas se desempeña correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios. Debe existir al menos un recurso de escritura en el sistema y en caso de agregarse referencias también debe existir al menos un recurso de lectura.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de plantillas”. ▪ Pulsar la opción “Agregar nuevo”. ▪ Llenar los campos correspondientes. ▪ Pulsar la opción “Guardar”.	
Resultado esperado: El sistema muestra la nueva plantilla en el listado de plantillas.	
Evaluación de la prueba: Satisfactoria.	

Tabla 80: Caso de prueba de aceptación "Crear plantilla".

Caso de prueba de aceptación	
Nombre de HU: Gestionar plantillas.	Id. de prueba: 2.2.
Nombre del caso de prueba: Listar plantillas.	
Descripción del caso de prueba: Permite validar que el listado de plantillas se muestra correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de plantillas”.
Resultado esperado: El sistema muestra el listado de plantillas.
Evaluación de la prueba: Satisfactoria.

Tabla 81: Caso de prueba de aceptación "Listar plantillas".

Caso de prueba de aceptación	
Nombre de HU: Gestionar plantillas.	Id. de prueba: 2.3.
Nombre del caso de prueba: Editar plantilla.	
Descripción del caso de prueba: Permite comprobar que la edición de plantillas funciona correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios. Debe existir al menos una plantilla.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de plantillas”. ▪ Realizar las modificaciones necesarias. ▪ Pulsar la opción “Guardar”.	
Resultado esperado: El sistema muestra el listado de plantillas. Cuando se vuelve a acceder a la plantilla editada, se observan los cambios antes realizados.	
Evaluación de la prueba: Satisfactoria.	

Tabla 82: Caso de prueba de aceptación "Editar plantilla".

Caso de prueba de aceptación	
Nombre de HU: Gestionar plantillas.	Id. de prueba: 2.4.
Nombre del caso de prueba: Eliminar plantilla.	
Descripción del caso de prueba: Permite confirmar que la funcionalidad de eliminar plantillas se desempeña correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios. Debe existir al menos una plantilla.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de plantillas”. ▪ Eliminar plantilla: - Variante 1: Pulsar la opción “Eliminar” correspondiente a la plantilla que se desea eliminar. - Variante 2: Seleccionar las plantillas que se desean eliminar y pulsar la opción “Eliminar selección”. ▪ Pulsar la opción “Si” que aparece en la pregunta de confirmación.
Resultado esperado: El sistema muestra el listado de plantillas, en la cual no se muestran las plantillas eliminadas.
Evaluación de la prueba: Satisfactoria.

Tabla 83: Caso de prueba de aceptación "Eliminar plantilla".

Caso de prueba de aceptación	
Nombre de HU: Administrar estructura de la entidad.	Id. de prueba: 3.1.
Nombre del caso de prueba: Mostrar estructura de la entidad.	
Descripción del caso de prueba: Permite verificar que la estructura de la entidad se visualiza correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Estructura administrativa”.	
Resultado esperado: El sistema muestra la estructura a través de un árbol desplegable.	
Evaluación de la prueba: Satisfactoria.	

Tabla 84: Caso de prueba de aceptación "Mostrar estructura de la entidad".

Caso de prueba de aceptación	
Nombre de HU: Administrar estructura de la entidad.	Id. de prueba: 3.2.
Nombre del caso de prueba: Actualizar estructura de la entidad.	
Descripción del caso de prueba: Permite verificar que la estructura de la entidad se actualiza correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Estructura administrativa”. ▪ Pulsar la opción “Actualizar estructura”. ▪ Pulsar la opción “Si” que aparece en la pregunta de confirmación.
Resultado esperado: El sistema muestra un mensaje informando que la estructura fue actualizada con éxito.
Evaluación de la prueba: Satisfactoria.

Tabla 85: Caso de prueba de aceptación "Actualizar estructura de la entidad".

Caso de prueba de aceptación	
Nombre de HU: Definir aprobadores.	Id. de prueba: 4.1
Nombre del caso de prueba: Definir aprobadores.	
Descripción del caso de prueba: Permite validar que la definición de aprobadores funciona correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Estructura administrativa”. ▪ Marcar la opción “Aprobador” correspondiente al área deseada. ▪ Seleccionar el cargo del aprobador correspondiente al área antes elegida. ▪ Pulsar la opción “Guardar”.	
Resultado esperado: El sistema muestra la página principal.	
Evaluación de la prueba: Satisfactoria.	

Tabla 86: Caso de prueba de aceptación "Definir aprobadores".

Caso de prueba de aceptación	
Nombre de HU: Gestionar flujos de aprobación.	Id. de prueba: 5.1.
Nombre del caso de prueba: Crear flujo de aprobación.	
Descripción del caso de prueba: Permite verificar que la creación de flujos de aprobación funciona correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios. Debe existir al menos una plantilla. Debe existir al menos un aprobador.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Flujo de aprobación”. ▪ Pulsar la opción “Agregar nuevo”. ▪ Construir el flujo de aprobación. ▪ Pulsar la opción “Guardar”.
Resultado esperado: El sistema muestra el nuevo flujo en el listado de flujos.
Evaluación de la prueba: Satisfactoria.

Tabla 87: Caso de prueba de aceptación "Crear flujo de aprobación".

Caso de prueba de aceptación	
Nombre de HU: Gestionar flujos de aprobación.	Id. de prueba: 5.2.
Nombre del caso de prueba: Listar flujos de aprobación.	
Descripción del caso de prueba: Permite validar que el listado de flujos de aprobación se muestra correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Flujo de aprobación”.	
Resultado esperado: El sistema muestra el listado de flujos de aprobación.	
Evaluación de la prueba: Satisfactoria.	

Tabla 88: Caso de prueba de aceptación "Listar flujos de aprobación".

Caso de prueba de aceptación	
Nombre de HU: Gestionar flujos de aprobación.	Id. de prueba: 5.3
Nombre del caso de prueba: Editar flujo de aprobación.	
Descripción del caso de prueba: Permite comprobar que la edición de flujos de aprobación funciona correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios. Debe existir al menos un flujo de aprobación.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Flujo de aprobación”. ▪ Pulsar la opción “Editar” correspondiente al flujo que se desea editar. ▪ Realizar las modificaciones necesarias. ▪ Pulsar la opción “Guardar”.
Resultado esperado: El sistema muestra el listado de flujos. Cuando se vuelve a acceder al flujo editado, se observan los cambios antes realizados.
Evaluación de la prueba: Satisfactoria.

Tabla 89: Caso de prueba de aceptación "Editar flujo de aprobación".

Caso de prueba de aceptación	
Nombre de HU: Gestionar flujos de aprobación.	Id. de prueba: 5.4.
Nombre del caso de prueba: Eliminar flujo de aprobación.	
Descripción del caso de prueba: Permite confirmar que los flujos de aprobación se eliminan correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios. Debe existir al menos un flujo de aprobación.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Flujo de aprobación”. ▪ Eliminar flujo: - Variante 1: Pulsar la opción “Eliminar” correspondiente al flujo de aprobación que se desea eliminar. - Variante 2: Seleccionar los flujos que se desean eliminar y pulsar la opción “Eliminar selección”. ▪ Pulsar la opción “Si” que aparece en la pregunta de confirmación.	
Resultado esperado: El sistema muestra el listado de flujos de aprobación en el cual no aparecen los eliminados.	
Evaluación de la prueba: Satisfactoria.	

Tabla 90: Caso de prueba de aceptación "Eliminar flujo de aprobación".

Caso de prueba de aceptación	
Nombre de HU: Administrar cuentas de usuario.	Id. de prueba: 8.1.
Nombre del caso de prueba: Crear cuenta de usuario.	
Descripción del caso de prueba: Permite verificar que la creación de cuentas de usuario funciona correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios. Debe existir al menos una cuenta pendiente de creación.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Pendientes de creación”. ▪ Crear cuenta: - Variante 1: Pulsar la opción “Crear” correspondiente a la solicitud que se desea convertir en cuenta de usuario. - Variante 2: Seleccionar las solicitudes que desea convertir en cuentas de usuario y pulsar la opción “Crear selección”. ▪ Pulsar la opción “Si” que aparece en la pregunta de confirmación.
Resultado esperado: Se muestra el listado de solicitudes pendientes de creación, en el cual no aparece la solicitud antes convertida en cuenta. Se muestra la cuenta creada en el listado de la sección “Últimas cuentas” y en el listado de la sección “Cuentas de usuario”.
Evaluación de la prueba: Satisfactoria.

Tabla 91: Caso de prueba de aceptación "Crear cuenta de usuario".

Caso de prueba de aceptación	
Nombre de HU: Administrar cuentas de usuario.	Id. de prueba: 8.2.
Nombre del caso de prueba: Listar cuentas de usuario.	
Descripción del caso de prueba: Permite validar que el listado de cuentas de usuario se muestra correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Cuentas de usuario”.	
Resultado esperado: Se muestra el listado de cuentas de usuario.	
Evaluación de la prueba: Satisfactoria.	

Tabla 92: Caso de prueba de aceptación "Listar cuentas de usuario".

Caso de prueba de aceptación	
Nombre de HU: Administrar cuentas de usuario.	Id. de prueba: 8.3.
Nombre del caso de prueba: Listar últimas cuentas de usuario.	
Descripción del caso de prueba: Permite comprobar que las últimas cuentas de usuario creadas se listan correctamente.	
Condiciones de ejecución: Para la variante dos es necesario estar autenticado.	

Entrada / Pasos de ejecución: ▪ Variante 1: Acceder al sistema sin estar autenticado. ▪ Variante 2: Entrar en la sección “Últimas cuentas”.
Resultado esperado: Se muestra el listado de cuentas de usuario.
Evaluación de la prueba: Satisfactoria.

Tabla 93: Caso de prueba de aceptación "Listar últimas cuentas de usuario".

Caso de prueba de aceptación	
Nombre de HU: Administrar cuentas de usuario.	Id. de prueba: 8.4.
Nombre del caso de prueba: Eliminar cuenta de usuario.	
Descripción del caso de prueba: Permite confirmar que las cuentas de usuario se eliminan correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Cuentas de usuario”. ▪ Eliminar cuenta: - Variante 1: Pulsar la opción “Eliminar” correspondiente a la cuenta que se desea eliminar. - Variante 2: Seleccionar las cuentas que se desean eliminar y pulsar la opción “Eliminar selección”. ▪ Pulsar la opción “Si” que aparece en la pregunta de confirmación.	
Resultado esperado: El sistema muestra el listado de cuentas en el cual no aparecen las eliminadas.	
Evaluación de la prueba: Satisfactoria.	

Tabla 94: Caso de prueba de aceptación "Eliminar cuenta de usuario".

Caso de prueba de aceptación	
Nombre de HU: Gestionar grupos de usuarios.	Id. de prueba: 9.1.
Nombre del caso de prueba: Crear grupo de usuarios.	
Descripción del caso de prueba: Permite verificar que la funcionalidad que crea grupos de usuario se desempeña correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de grupos”. ▪ Pulsar la opción “Agregar nuevo”. ▪ Conformar el grupo. ▪ Pulsar la opción “Guardar”.
Resultado esperado: Se crea un nuevo grupo de usuarios y el sistema lo muestra en el listado de grupos.
Evaluación de la prueba: Satisfactoria.

Tabla 95: Caso de prueba de aceptación "Crear grupo de usuarios".

Caso de prueba de aceptación	
Nombre de HU: Gestionar grupos de usuarios.	Id. de prueba: 9.2.
Nombre del caso de prueba: Listar grupos de usuarios.	
Descripción del caso de prueba: Permite validar que el listado de grupos de usuarios se muestra correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de grupos”.	
Resultado esperado: El sistema muestra el listado de grupos de usuarios.	
Evaluación de la prueba: Satisfactoria.	

Tabla 96: Caso de prueba de aceptación "Listar grupos de usuarios".

Caso de prueba de aceptación	
Nombre de HU: Gestionar grupos de usuarios.	Id. de prueba: 9.3.
Nombre del caso de prueba: Editar grupo de usuarios.	
Descripción del caso de prueba: Permite comprobar que la edición de grupos de usuarios funciona correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	

Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de grupos”. ▪ Pulsar la opción “Editar” correspondiente al grupo que se desea editar. ▪ Realizar las modificaciones necesarias. ▪ Pulsar la opción “Guardar”.
Resultado esperado: El sistema muestra el listado de recursos. Cuando se vuelve a acceder al recurso editado, se observan los cambios antes realizados.
Evaluación de la prueba: Satisfactoria.

Tabla 97: Caso de prueba de aceptación "Editar grupo de usuarios".

Caso de prueba de aceptación	
Nombre de HU: Gestionar grupos de usuarios.	Id. de prueba: 9.4.
Nombre del caso de prueba: Eliminar grupo de usuarios.	
Descripción del caso de prueba: Permite confirmar que los grupos de usuarios se eliminan correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de grupos”. ▪ Eliminar grupo: - Variante 1: Pulsar la opción “Eliminar” correspondiente al grupo que se desea eliminar. - Variante 2: Seleccionar los grupos que se desean eliminar y pulsar la opción “Eliminar selección”. ▪ Pulsar la opción “Si” que aparece en la pregunta de confirmación.	
Resultado esperado: El sistema muestra el listado de grupos de usuarios en el cual no aparecen los eliminados.	
Evaluación de la prueba: Satisfactoria.	

Tabla 98: Caso de prueba de aceptación "Eliminar grupo de usuarios".

Caso de prueba de aceptación	
Nombre de HU: Autenticar usuario.	Id. de prueba: 10.1.
Nombre del caso de prueba: Iniciar sesión en el sistema.	
Descripción del caso de prueba: Permite validar que el inicio de sesión funciona correctamente.	
Condiciones de ejecución: Tener cuenta de usuario en el servicio de autenticación configurado.	

Entrada / Pasos de ejecución: ▪ Pulsar la opción “Autenticación”. ▪ Introducir el usuario y la contraseña. ▪ Pulsar la opción “Entrar”.
Resultado esperado: El sistema inicia sesión y muestra las funcionalidades correspondientes al nivel de acceso de la persona autenticada.
Evaluación de la prueba: Satisfactoria.

Tabla 99: Caso de prueba de aceptación "Iniciar sesión en el sistema".

Caso de prueba de aceptación	
Nombre de HU: Autenticar usuario.	Id. de prueba: 10.2.
Nombre del caso de prueba: Cerrar sesión en el sistema.	
Descripción del caso de prueba: Permite comprobar que las sesiones del sistema se cierran correctamente.	
Condiciones de ejecución: Estar autenticado.	
Entrada / Pasos de ejecución: ▪ Pulsar la opción “Cerrar sesión”.	
Resultado esperado: El sistema regresa a la página inicial en la que se muestra el listado de las últimas cuentas creadas.	
Evaluación de la prueba: Satisfactoria.	

Tabla 100: Caso de prueba de aceptación "Cerrar sesión en el sistema".

Caso de prueba de aceptación	
Nombre de HU: Administrar registros de sucesos del sistema.	Id. de prueba: 11.1.
Nombre del caso de prueba: Listar registros de sucesos del sistema.	
Descripción del caso de prueba: Permite validar que el listado de registros de sucesos del sistema se muestra correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de eventos”.	
Resultado esperado: El sistema muestra el listado de registros de sucesos del sistema.	
Evaluación de la prueba: Satisfactoria.	

Tabla 101: Caso de prueba de aceptación "Listar registros de sucesos del sistema".

Caso de prueba de aceptación	
Nombre de HU: Administrar registros de sucesos del sistema.	Id. de prueba: 11.2.
Nombre del caso de prueba: Rotar registros de sucesos del sistema.	
Descripción del caso de prueba: Permite comprobar que los registros de sucesos del sistema se rotan correctamente.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada / Pasos de ejecución: ▪ Entrar en la sección “Gestión de eventos”. ▪ Pulsar la opción “Rotar”.	
Resultado esperado: El sistema muestra el listado de registros de sucesos del sistema, en el cual sólo se muestran los registros de sucesos del sistema del mes actual.	
Evaluación de la prueba: Satisfactoria.	

Tabla 102: Caso de prueba de aceptación "Rotar registros de sucesos del sistema".

Caso de prueba de aceptación	
Nombre de HU: Buscar información en un listado de datos.	Id. de prueba: 12.1.
Nombre del caso de prueba: Buscar información en un listado de datos.	
Descripción del caso de prueba: Permite validar que la funcionalidad “Buscar información en un listado de datos” se desempeña correctamente.	
Condiciones de ejecución: -	
Entrada / Pasos de ejecución: ▪ Acceder a cualquier listado en el sistema. ▪ Insertar en el campo “Buscar” una cadena de texto que corresponda a una parte o que sea igual a cualquier texto contenido en el listado.	
Resultado esperado: El sistema lista sólo los registros cuya información coincide con el criterio de búsqueda.	
Evaluación de la prueba: Satisfactoria.	

Tabla 103: Caso de prueba de aceptación "Buscar información en un listado de datos"