Universidad de las Ciencias Informáticas

Facultad#1

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título:

Seguridad de negocio en el Sistema de Gestión Académica de Pregrado

Autores:

Yadira Peña Hechavarría.

Lenier Garcia Vizcaino.

Tutores:

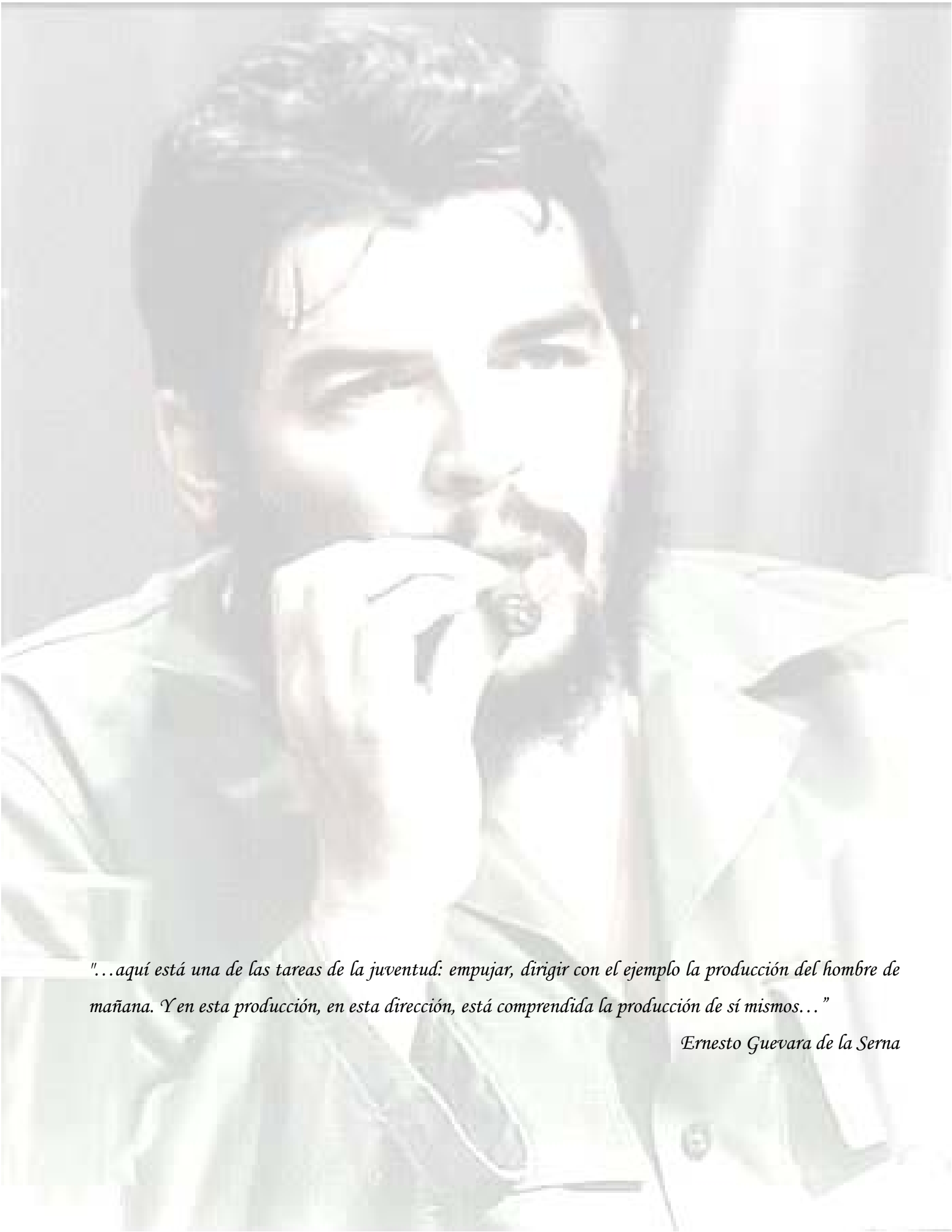
Ing. Norges Sánchez Tumbarell.

Ing. Yoan Carlos Machado Espinosa.

Ing. Yanio Garcia Vidal.

La Habana, Junio 2013

"Año 55 de la Revolución"



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

Ernesto Guevara de la Serna

Declaración de autoría

Declaramos que somos los únicos autores del trabajo de diploma: “Seguridad de negocio en el Sistema de Gestión Académica de Pregrado” y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales para que haga uso del mismo en su beneficio.

Para hacerlo constar firmamos el presente a los ____ días del mes de _____ del año _____.

_____
Yadira Peña Hechavarría
Firma del autor

_____
Lenier Garcia Vizcaino
Firma del autor

_____
Ing. Norges Sánchez Tumbarell
Firma del tutor

_____
Ing. Yoan Carlos Machado Espinosa
Firma del tutor

_____
Ing. Yanio García Vidal
Firma del tutor

Agradecimientos

Quiero darles gracias a Dios, por darme salud, voluntad, esperanza y las fuerzas necesarias para avanzar cuando las cosas parecían ir mal. Quiero agradecerle especialmente a mi familia. A mi papá Irán, por haber sido para mí un ejemplo de sacrificio, por ser una fuente de inspiración, la meta donde siempre deseé llegar, por no rendirse nunca, por vencer siempre las adversidades y no perder jamás la fe. Gracias papi, por tus consejos oportunos, por cada uno de tus correos, , tus chistes, por darme ánimos cuando decaía ,por todo tu amor y comprensión, por recordarme siempre lo orgulloso que te sientes de mí, de mis logros, por recordarme que nunca se puede perder la ecuanimidad, que las cosas, siempre tienes solución. A mi mamá, Joaquina, por ser una mujer incansable, luchadora, una madre ejemplar y dedicada, por todo su amor y comprensión, por respetar y apoyar mis decisiones aunque no siempre esté de acuerdo con ellas. Por corregirme y por ayudarme a levantar las veces que he caído. Gracias mami, por todo el apoyo que siempre me has brindado, por tu ejemplo y por tus constantes preocupaciones por mí. A mis hermanos, a los cuales adoro porque sin que lo supieran, me sentí responsable de darle el mejor ejemplo, para que luchen por lo que desean y salgan adelante con sus estudios.

A mis tíos, mi primito, que siempre me apoyaron y nunca tuvieron para mí un "no" las veces que los necesité. Ustedes han sido para mí una verdadera familia.

A mi novio (lo mejor que me ha pasado en la universidad), gracias Pelusso por hacerme sentir especial, por todo tu apoyo, tu amor, tu paciencia. Haber trabajado juntos para alcanzar lo que hoy tenemos ha sido para mí una verdadera experiencia.

A mis suegros, un millón de gracias a mis suegros, mi cuñada, gracias de todo corazón por sus preocupaciones, sus atenciones, por toda la confianza que depositaron en mí, por abrirme las puertas de su casa y de su corazón. Por apoyarme en todo momento, porque me hicieron sentir desde el primer día parte de su familia.

A Tata y Lore, nuca voy a olvidar todo lo que hicieron por Lenier y por mí, gracias por toda la ayuda que nos dieron.

A Osne (mi amiga del vicio), a Yadira, por toda la ayuda que me brindó con el documento, niñas a ustedes dos, muchas gracias por su amistad, por compartir conmigo sus experiencias, por escucharme, aconsejarme, gracias por todos los momentos que compartimos, si de algo estoy segura es que las voy a extrañar.

A mis tutores, Yoan, Yanio, Norges porque ustedes contribuyeron a que este sueño se hiciera realidad.

A mis compañeros de grupo, a las chicas del 146-102, y a todas esas personas con las que de una forma u otra he compartido, gracias a ustedes porque han formado parte de mi historia en la universidad.

Yadira

Agradecimientos

Primero que todo y todos a mi familia, a cada uno de sus miembros que cada día dieron su gota de sacrificio para que hoy pueda ser un ingeniero. A mi madre por su constante preocupación y sus oportunos consejos cuando los problemas me acongojaron, a mi padre por su incesante apoyo en todos los sentidos que alguien pueda llegar a imaginar, juntos logramos que esa herencia que me quieres dejar desde que comencé a estudiar hoy se haga realidad, gracias viejo.

A mi hermana que más que una hermana ha sido como otra madre para mí y siempre ha estado ahí para lo que sea siendo mi eterna amiga sin juzgarme ni criticar mis acciones, siendo capaz de escuchar cuando el mundo solo podía criticar.

A mi abuelita Aquilina que aunque ya no esté entre nosotros, esté donde esté sé que me está mirando y sabe que sus granitos de arena también están en mí, para ti este regalo que te llega desde mi corazón.

A mi novia, mi pelusita linda, quien además de enseñarme a amar y soportarme durante estos años, me ha inculcado a base de ejemplo como se es responsable y maduro sin dejar atrás la malicia y el jaraneo de la juventud, a ti mi bebé gracias por existir.

A mi familión de Remedios que a pesar de la distancia siempre han estado pendientes de mí, a mi abuela Amelia (Vieji), mi abuelo Pepe (Florindo) y mis tías y tíos gracias por todo.

A la tropa de Caibarién, mi primo Yovani y mi tío Pepe gracias por sus visitas cuando estoy en la casa y por el apoyo que le han dado a mi madre por sobre todas las cosas.

A mi familias postizas, la de Placetas y la de Moa que ya más que postizas son familia de la buena, a todos gracias. A mis amigos dentro de la universidad: Almarales, mi hermano desde el pre que él lo sabe; el JD, mi hermanito aquí en la UCI, gracias por tus consejos, al Irra (El inmortal), Julio y Yander (Tornado) por su ayuda incondicional a pesar de estar complicados con sus propios problemas, a Yoan por las consultas de php que me dio y lo mucho que me asesoró no solo durante el trabajo de diploma sino desde que entré a la producción en tercer año, mi tutor pero más que tutor, mi amigo, de veras gracias, a Yanio por sus clases de diseño, al piquete del 143 por los buenos momentos que pasamos dando chucho, a la tropa del 12 y el 123 con los que compartí residencia durante estos 5 años, a los tufados y los asados del dota y el xnova, a los invencibles del equipo de beisbol de la facultad, seremos campeones por lo menos dos años seguidos más (yo vengo con eso basta), en fin a todos los que de una manera u otra contribuyeron a que este sueño de convertirse en ingeniero de un guajirito de Jinaguayabo se hiciera realidad.

Lenier

Dedicatoria

Este Trabajo de Diploma, que es el resultado de mi esfuerzo, lo quiero dedicar a esas personas maravillosas y especiales que siempre creyeron y depositaron toda su confianza en mí. A mis padres, los mejores padres del mundo, porque juntos y con mucho amor, supieron hacer de mí una mujer de bien: a mi papá porque ha sido para mí un ejemplo de sacrificio, perseverancia y fortaleza, a mi mamá, por haberme educado con amor, dedicación y rectitud. A mis hermanos por su amor y apoyo incondicional, espero que este logro les sirva de impulso y le sirva de guía y ejemplo en el transcurso de sus estudios. A mis tíos, por haber sido para mí en estos 5 años una verdadera familia. A mi novio, por estar siempre a mi lado y apoyarme en todas mis decisiones sin cuestionarlas. A mis suegros, mi cuñada (mi otra familia), porque sé que mis logros y mis alegrías son también las suyas.

Yadira

A mis padres, a mi hermana, a mi novia y muy especial a mi abuela Aquilina que Dios la tenga a su lado.

Lenier

Resumen

El Sistema de Gestión Universitaria es una solución integral desarrollada en la Universidad de las Ciencias Informáticas para la automatización de procesos sustantivos. Actualmente a pesar del sistema tener implementada una fuerte política de control de acceso no reconoce bajo qué rol el usuario está accediendo a la información, violándose la seguridad lógica de los datos; esto se debe a que no se cuenta con un mecanismo que permita asignarles a los usuarios del sistema acceso únicamente a la información definida en su radio de acción. Con el objetivo de responder a esta situación se desarrolla una herramienta de configuración basada en el filtrado de datos, reduciendo así el cúmulo de información accesible por los usuarios definida por permisos. Esta solución permitirá al administrador del sistema definir una serie de permisos para controlar el acceso a la información. Además, se gestionarán las categorías de los filtros, permitiendo que la solución sea extensible a otros sub-sistemas que decidan controlar el acceso a sus datos.

En el desarrollo de la solución informática se utilizaron los lenguajes: *PHP, JavaScript, HTML, XML,* entre otros*;* como marco de trabajo GUUD, como herramienta para la administración de datos *PgAdminIII*, como servidor *web Apache*, como Sistema de Gestor de Base de Datos *PostgreSQL*, como Entorno de Desarrollo Integrado *NetBeans*, como herramienta de diseño *Evolus Pencil*, como herramienta de modelado *Visual Paradigm*, como lenguaje de modelado *UML*. Para guiar la investigación se utiliza un proceso de desarrollo con enfoque ágil basado en el nivel 2 de *CMMI*.

Palabras claves: “acceso”, “control”, “seguridad”, “usuarios”.

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

Con el surgimiento de las ciencias computacionales y hasta la actualidad, el campo de acción de la informática se ha amplificado vertiginosamente, con ello se ha logrado automatizar los más disímiles procesos de la actividad humana diaria. Producto de este auge y ante la latente necesidad de almacenar grandes volúmenes de información y gestionarla de manera eficiente surgen los sistemas de bases de datos automatizados, convirtiéndose así en una de las herramientas más ampliamente difundidas en la actual sociedad de la información, utilizadas como fuentes de recuperación y almacenamiento de información en campos como: ciencia, sociedad, economía, política y cultura. La aplicación de estos medios informáticos ha revolucionado la gestión de las empresas a nivel mundial surgiendo la necesidad de crear sistemas que permiten almacenar, actualizar y posteriormente acceder a los datos de forma rápida y estructurada, dichos sistemas son conocidos como Sistemas Gestores de Bases de Datos (SGBD).

Existen diversos SGBD como *Oracle*, *SQLServer*, *MySQL* y *PostgreSQL* por lo que resulta imprescindible seleccionar el más conveniente de acuerdo a las necesidades de las empresas y organismos, por ejemplo, lograr una disminución de los gastos por pago de licencias de *software* y soporte. Los SGBD en los últimos años producto del incremento de las prestaciones, así como del aumento de los ataques y violaciones a las barreras de seguridad impuestas para regular el acceso a los sistemas, han provocado que el tema de la seguridad tome una mayor importancia dentro del producto a desarrollar. Para ello es vital el cumplimiento de los principios básicos de la seguridad informática: confidencialidad, integridad y disponibilidad de la información.

En el caso de Cuba y fundamentalmente encaminado a lograr la soberanía tecnológica se utiliza generalmente *PostgreSQL*, producto de código abierto que es dirigido por una Comunidad de Desarrolladores (PGDG1) y comercializado por organizaciones que los representan en el mercado, este gestor cuenta con facilidades tales como: el soporte de tipos de datos únicos, la creación de disparadores y la gestión de usuarios a la hora de manejar permisos.

La Universidad de las Ciencias Informáticas (UCI) surgida en el año 2002 al calor de la Batalla de Ideas teniendo como principal impulsor del proyecto a nuestro Comandante en Jefe Fidel Castro es una de las organizaciones que se encuentra a la vanguardia del desarrollo de productos de código abierto en el país, dentro de sus áreas productivas se encuentra el Centro de Informatización Universitaria (CENIA) que a su vez cuenta con el Departamento de Gestión Universitaria encargado de automatizar varios de los procesos referentes a la vida en la universidad a través de un sistema informático publicado en una

plataforma *web* que garantiza en todo momento el acceso, actualización, fiabilidad y manejo de la información.

Todo producto publicado en la red corre el riesgo de recibir ataques de todo tipo, en lo fundamental dirigido a los datos guardados en su base de datos por lo que toda la seguridad que se implante es poca. En la actualidad el Sistema de Gestión Universitaria (SGU) tiene desplegada una fuerte política de control de acceso desde la *web* en todos sus subsistemas y módulos, a pesar de esto no se satisface la necesidad de otorgarle a cada usuario acceso a la información específica con la que debe trabajar independientemente de que todos tengan acceso a un proceso determinado. Esta situación puede llevar a que sea comprometida la consistencia de los datos y puede ocurrir el robo, transformación o destrucción de la información por parte de usuarios inexpertos o inescrupulosos. Ante este riesgo es de vital importancia controlar en todo momento quien manipula la información buscando obtener una robusta seguridad de negocio.

La Gestión Universitaria (GU) en la UCI, tiene una envergadura de nivel macro sistémico; en la misma fluyen procesos académicos, de cooperación, de residencia, entre otros; los cuales son identificados por su pertenencia a las diferentes áreas de la universidad, en un proceso se pueden ver inmersas diferentes áreas. Como ejemplo se puede citar cuando se define, como radio de acción de un profesor, aquel ámbito donde se encuentran elementos asociados a la asignatura que imparte y donde están involucrados sus estudiantes, de manera que el profesor se sienta cómodo y libre de responsabilidades ajenas en su labor sobre documentación importante. Esto sugiere que en las acciones sobre la información que se extrae en determinado momento en el SGU, se necesite de validaciones que estén enfocadas a evitar visualizar elementos que no pertenezcan al radio de acción definido para la persona autenticada, entiéndase esto como seguridad de negocio.

Teniendo en cuenta la problemática descrita anteriormente, se concibe la formulación del **problema de investigación** de la siguiente manera:

Limitación de la seguridad de negocio en el Sistema de Gestión Académica de Pregrado (SGAP) de la Universidad de las Ciencias Informáticas.

El **objeto de estudio** está constituido por: Seguridad de negocio en el SGU.

El **campo de acción** está enfocado a: Seguridad de negocio en el SGAP.

La investigación persigue como **objetivo general**:

Desarrollar una solución informática con herramientas libres, que permita extender la seguridad de negocio sobre el acceso a la información relacionada con los objetos de negocio, que se identificaron en los procesos de gestión académica de Pregrado de la Universidad de las Ciencias Informáticas.

Para obtener resultados satisfactorios en la investigación se trazan los siguientes **objetivos específicos:**

- Caracterizar los principales elementos de la fundamentación teórica de la investigación.
- Elaborar una propuesta de solución que sea viable, extensible y permita aplicar la seguridad de negocio sobre los objetos de negocio que se identificaron en el SGAP.
- Desarrollar la propuesta de solución utilizando las herramientas y técnicas definidas.
- Validar la propuesta de solución aplicando pruebas funcionales y de integración.

Se plantea la siguiente **Idea a defender**:

La realización de una solución informática para extender la seguridad de negocio sobre el acceso a los datos en el SGAP mejorará la confidencialidad, integridad y disponibilidad de la información.

Para dar cumplimiento a los objetivos de la investigación se definieron las siguientes **tareas investigativas**:

- Diagnóstico del estado del arte en los procesos de Gestión Académica de Pregrado en Cuba y el mundo.
- Identificación de los objetos de negocio de los procesos de Gestión Académica de Pregrado en la Universidad de las Ciencias Informáticas.
- Identificación de los trabajadores de negocio que interactúan con cada objeto de negocio identificado en los procesos de Gestión Académica de Pregrado en la Universidad de las Ciencias Informáticas.
- Análisis de la implementación de la seguridad de negocio en sistemas de gestión que la contengan.
- Análisis de técnicas de ingeniería de requisitos.
- Elaboración de las especificaciones de requisitos sobre la interacción entre trabajadores de negocio y objetos de negocio en la Gestión Académica de Pregrado en la Universidad de las Ciencias Informáticas.
- Elaboración del diseño de la estrategia de homologación de sucesos aplicable a la interacción entre trabajadores de negocio y objetos de negocio en la gestión académica de pregrado en la Universidad de las Ciencias Informáticas.
- Elaboración del modelo de la base de datos que sustente la estrategia de homologación diseñada.
- Caracterización de las herramientas que se utilizarán en el marco de trabajo para el desarrollo del SGU y que serán usadas en la implementación de la estrategia de homologación diseñada.
- Elaboración de los casos de prueba para la estrategia de homologación diseñada.

- Implementación de las funcionalidades descritas en las especificaciones de requisitos para la estrategia de homologación diseñada.
    - Funcionalidades de reglas de negocio.
    - Funcionalidades de flujos de control de la información.
    - Funcionalidades de presentación.
- Validación de la estrategia de homologación diseñada.
    - Realización de pruebas unitarias.
    - Realización de pruebas de integración con los módulos: Carreras, Personal y Secretaría, Control Docente y Reportes en el SGAP.

Con el satisfactorio cumplimiento de las tareas se espera obtener los siguientes resultados o aportes:

- Una solución informática que extienda la seguridad de negocio sobre la información gestionada en el SGAP.
- Solución extensible al SGU de la UCI.
- Documentación sobre la seguridad de negocio en los procesos de la Gestión Universitaria en la Universidad de las Ciencias Informáticas.

Para desarrollar las tareas investigativas antes mencionadas se trabaja con los siguientes **métodos de investigación:**

- **Método Teórico Histórico Lógico:**

Se logra una mayor comprensión del estado actual y las etapas por las que transitan los sistemas de gestión de la información. Muestra el desarrollo tecnológico y la evolución que tienen los principales Sistemas Gestores de la Información haciendo énfasis en el SGAP. Analizar las características comunes de los sistemas de gestión, su funcionamiento, cómo se trata el tema de la seguridad de la información y especialmente enfocar este estudio al SGAP para obtener una investigación fructífera para luego implementar la aplicación que se propone para mejorar la seguridad de negocio.

- **Método Teórico Analítico-Sintético:**

Se utiliza este método para analizar la estructura y almacenamiento de datos así como los servicios que propone *PostgreSQL*. Además se realiza un estudio de la documentación existente, tanto en el expediente de proyecto como en la *web*, enfocado a la obtención de seguridad en los Sistemas de Gestión de la Información y mediante ello se extraen los elementos fundamentales para sentar las bases teóricas que permitan darle solución al problema propuesto.

- **Método Empírico Entrevista:**

Se realiza a las personas involucradas en el proceso y que cuentan con amplios conocimientos del tema, con el objetivo de obtener la mayor cantidad de información referente a los procesos que forman parte del engranaje de seguridad de negocio.

El trabajo de diploma estará estructurado con los siguientes capítulos:

Capítulo 1. Fundamentación teórica: en el este capítulo se fundamentan las definiciones y tecnologías que serán utilizadas durante el proceso de desarrollo de la solución informática, así como un análisis del estado del arte de los procesos de seguridad de negocio enfocados al filtrado de información en sistemas que lo implementen tanto en el contexto nacional como internacional.

Capítulo 2. Diseño de la solución: en este capítulo se hará la especificación de los requisitos tanto funcionales como no funcionales a tener en cuenta durante el desarrollo de la solución, se hace un análisis de los componentes y funcionalidades que compondrán el producto final, es descrita la arquitectura a utilizar durante la implementación y se propone el diseño de la base de datos a desarrollar.

Capítulo 3. Implementación y pruebas: en este capítulo son descritas todas las pruebas y validaciones funcionales hechas al sistema.

Capítulo 1. Fundamentación teórica

1.1 Introducción

En el presente capítulo se realiza el análisis del arte de los procesos de seguridad de negocio enfocado al filtrado de información. Además se fundamentan conceptos, definiciones y se explican las tecnologías que serán utilizadas durante todo el proceso de desarrollo de la solución informática.

1.2 Conceptos relacionados con la investigación

Datos

Un dato es un documento o una información que permite llegar al conocimiento de algo o deducir las consecuencias legítimas de un hecho, no tiene sentido en sí mismo, sino que se utiliza en la toma de decisiones o en la realización de cálculos a partir de un procesamiento adecuado y teniendo en cuenta su contexto. En la informática, por lo general los datos son representaciones simbólicas o atributos de una entidad. También son considerados como expresiones generales que describen características de las entidades sobre las que operan los algoritmos (EcuRed 2013).

Seguridad

La seguridad es la protección ante peligros ya sean internos o externos que afecten negativamente la calidad de algo. Hace referencia al conjunto de medidas y políticas implementadas para establecer la protección contra aquello que ponga en riesgo la integridad física (Latham 1985).

Seguridad de los datos

La seguridad de los datos es su protección de operaciones indebidas que pongan en peligro su definición, existencia, consistencia e integridad, independientemente de la persona que los accede. Esto se logra mediante mecanismos que permiten estructurar el acceso y la actualización de los mismos sin necesidad de modificar o alterar el diseño del modelo definido de acuerdo a los requisitos del sistema o aplicación *software* (slideshare 2010).

Seguridad Informática

La seguridad informática se puede definir como cualquier medida que impida la ejecución de operaciones no autorizadas sobre un sistema o red informática, cuyos efectos pueden producir daños y comprometer la información. La seguridad informática tiene como objetivo mantener la confidencialidad, integridad y disponibilidad de la información (slideshare 2010).

- ✓ Confidencialidad: los objetos de un sistema han de ser accedidos únicamente por elementos autorizados a ello. Asegura el secreto de las comunicaciones contenidas en los mensajes.

- ✓ Integridad: los objetos sólo pueden ser modificados por elementos autorizados, y de una manera controlada.
- ✓ Disponibilidad: los objetos del sistema tienen que permanecer accesibles a elementos autorizados.

Políticas de seguridad

El término política de seguridad se define como un conjunto de requisitos especificados por los responsables de un sistema, que indica en términos generales que está y que no está permitido en el área de seguridad durante la operación general del sistema *web*. La política se refleja en una serie de reglamentos y protocolos a seguir donde se definen las medidas a tomar para proteger la seguridad del sistema; pero ante todo, una política de seguridad es una forma de comunicarse con los usuarios. Siempre hay que tener en cuenta que la seguridad comienza y termina con personas y debe:

- ➢ Ser holística: es decir cubrir todos los aspectos relacionados con la seguridad.
- ➢ Adecuarse a las necesidades y recursos.
- ➢ Ser atemporal: el tiempo en el que se aplica no debe influir en su eficacia y eficiencia.
- ➢ Definir estrategias y criterios generales a adoptar en distintas funciones y actividades, donde se conocen las alternativas ante circunstancias repetidas.

La política de seguridad de una organización son las normas, reglas o leyes que rigen la vida de la organización en cuanto a lo que se puede o no hacer. Toda política de seguridad debe tener normas sobre uso aceptable, que definan el uso apropiado de los recursos informáticos de la organización. Debe establecer claramente la responsabilidad de los usuarios con respecto a la protección de la información almacenada en sus cuentas. Debe señalar los permisos que pueden tener los usuarios sobre ficheros. Debe estipular el uso aceptable del acceso *web* y de todo tipo de accesos no relacionados con el objeto de la organización, de los recursos informáticos.

Otro punto importante dentro de las políticas de seguridad es el control de acceso. Esta medida determina los datos a los cuales el usuario tendrá acceso en dependencia de los que sean necesarios para desempeñar su trabajo. Para el control de acceso es necesario establecer un grupo de usuario con sus respectivos roles, especificar las tareas que van a realizar y documentar esta información (Latham 1985).

Niveles de Seguridad Informática

El estándar de niveles de seguridad más utilizado internacionalmente es el *TCSEC Orange Book*, desarrollado en 1983 de acuerdo a las normas de seguridad en computadoras del Departamento de Defensa de los Estados Unidos. Los niveles describen diferentes tipos de seguridad del Sistema Operativo y se enumeran desde el mínimo grado de seguridad al máximo. Estos niveles han sido la base de

desarrollo de estándares europeos y luego internacionales. Cabe aclarar que cada nivel requiere todos los niveles definidos anteriormente: así el subnivel B2 abarca los subniveles B1, C2, C1 y el D (Latham 1985).

- **Nivel D**

Este nivel contiene solo una división y está reservada para sistemas que han sido evaluados y no cumplen con ninguna especificación de seguridad. Sin sistemas no confiables, no hay protección para el *hardware*, el sistema operativo es inestable y no hay autentificación con respecto a los usuarios y sus derechos en el acceso a la información. Los sistemas operativos que responden a este nivel son *MS-DOS y System 7.0 de Macintosh*.

- **Nivel C1: Protección Discrecional**

Se requiere de usuarios que permiten el acceso a distinta información. Cada usuario puede manejar su información privada y se hace la distinción entre los usuarios y el administrador del sistema. Muchas de las tareas cotidianas de administración del sistema sólo pueden ser realizadas por este "súper usuario" quien tiene gran responsabilidad en la seguridad del mismo. Con la actual descentralización de los sistemas de cómputos, no es raro que en una organización encontremos dos o tres personas cumpliendo este rol. Esto es un problema, pues no hay forma de distinguir entre los cambios que hizo cada usuario.

A continuación se enumeran los requisitos mínimos que debe cumplir la clase C1:

- ✓ Acceso de control discrecional: distinción entre usuarios y recursos. Se podrán definir grupos de usuarios (con los mismos privilegios) y grupos de objetos (archivos, directorios, disco) sobre los cuales podrán actuar usuarios o grupos de ellos.
- ✓ Identificación y Autentificación: se requiere que un usuario se identifique antes de comenzar a ejecutar acciones sobre el sistema. El dato de un usuario no podrá ser accedido por un usuario sin autorización o identificación.

- **Nivel C2: Protección de Acceso Controlado**

Este subnivel fue diseñado para solucionar las debilidades del C1. Cuenta con características adicionales que crean un ambiente de acceso controlado. Se debe llevar una auditoria de accesos e intentos fallidos de acceso a objetos. Tiene la capacidad de restringir aún más el que los usuarios ejecuten ciertos comandos o tengan acceso a ciertos archivos, permitir o denegar datos a usuarios en concreto, con base no sólo en los permisos, sino también en los niveles de autorización.

Requiere que se audite el sistema. Esta auditoría es utilizada para llevar registros de todas las acciones relacionadas con la seguridad, como las actividades efectuadas por el administrador del sistema y sus usuarios. La auditoría requiere de autenticación adicional para estar seguros de que la persona que ejecuta el comando es quien dice ser. Su mayor desventaja reside en los recursos adicionales requeridos

por el procesador y el subsistema de discos. Los usuarios de un sistema C2 tienen la autorización para realizar algunas tareas de administración del sistema sin necesidad de ser administradores. Permite llevar mejor la cuenta de las tareas relacionadas con la administración del sistema, ya que es cada usuario quien ejecuta el trabajo y no el administrador del sistema.

- **Nivel B1: Seguridad Etiquetada**

Este subnivel, es el primero de los tres con que cuenta el nivel B. Soporta seguridad multinivel, como la secreta y ultra-secreta. Se establece que el dueño del archivo no puede modificar los permisos de un objeto que está bajo control de acceso obligatorio. A cada objeto del sistema (usuario, dato) se le asigna una etiqueta, con un nivel de seguridad jerárquico (alto secreto, secreto, reservado) y con unas categorías (contabilidad, nóminas, ventas). Cada usuario que accede a un objeto debe poseer un permiso expreso para hacerlo y viceversa. Es decir que cada usuario tiene sus objetos asociados. También se establecen controles para limitar la propagación de derecho de acceso a los distintos objetos.

- **Nivel B2: Protección Estructurada**

Requiere que se etiquete cada objeto de nivel superior por ser padre de un objeto inferior. La Protección Estructurada es la primera que empieza a referirse al problema de un objeto a un nivel más elevado de seguridad en comunicación con otros objetos a un nivel inferior. Así, un disco rígido será etiquetado por almacenar archivos que son accedidos por distintos usuarios. El sistema es capaz de alertar a los usuarios si sus condiciones de accesibilidad y seguridad son modificadas; y el administrador es el encargado de fijar los canales de almacenamiento y ancho de banda a utilizar por los demás usuarios.

- **Nivel B3: Dominios de Seguridad**

Refuerza los dominios con la instalación de *hardware*: por ejemplo el *hardware* de administración de memoria se usa para proteger el dominio de seguridad de acceso no autorizado a la modificación de objetos de diferentes dominios de seguridad. Existe un monitor de referencia que recibe las peticiones de acceso de cada usuario y las permite o las deniega según las políticas de acceso que se hayan definido. Todas las estructuras de seguridad deben ser lo suficientemente pequeñas como para permitir análisis y testeo ante posibles violaciones. Este nivel requiere que la terminal del usuario se conecte al sistema por medio de una conexión segura. Además, cada usuario tiene asignados los lugares y objetos a los que acceder.

- **Nivel A: Protección Verificada**

Es el nivel más elevado, incluye un proceso de diseño, control y verificación, mediante métodos formales (matemáticos) para asegurar todos los procesos que realiza un usuario sobre el sistema. Para llegar a este nivel de seguridad, todos los componentes de los niveles inferiores deben incluirse. El diseño

requiere ser verificado de forma matemática y también se deben realizar análisis de canales encubiertos y de distribución confiable. El *software* y *el hardware* son protegidos para evitar infiltraciones ante traslados o movimientos del equipamiento.

Negocio

Las definiciones que aparecen de negocio en la literatura están fundamentalmente enfocadas a sistemas, métodos o formas de obtener dinero, a cambio de ofrecer productos, bienes o servicios a otras personas. También puede verse como una actividad comercial o social que se ha pensado y que se desea desarrollar. Es una herramienta que nos permite organizar y planificar las actividades que se deben realizar para lograr las metas de una empresa cooperativa (Lengua Española 2010).

Para la presente investigación se define negocio como el arsenal de reglas que regulan el flujo de la información para la gestión académica y la influencia de la acción humana en la aplicación de estas reglas.

Seguridad de negocio

Lo que es conocido como seguridad lógica en la literatura, es presentado aquí como seguridad de negocio. De manera que, la seguridad de negocio, se define como la aplicación de barreras y procedimientos que resguarden el acceso a los datos y sólo se permita acceder a ellos a las personas o sistemas autorizados para hacerlo (Latham 1985).

Sistemas de gestión

Un sistema de gestión es una estructura probada para la gestión y mejora continua de las políticas, los procedimientos y procesos de la organización. Ayuda a lograr las metas y objetivos de una organización mediante una serie de estrategias, que incluyen la optimización de procesos y el enfoque centrado en la gestión. Por tanto, un sistema de gestión es un conjunto de etapas unidas en un proceso continuo, que deja trabajar ordenadamente una idea hasta lograr mejoras y su continuidad. La implementación de un sistema de gestión eficaz en una organización puede ayudarla a: gestionar riesgos, mejorar la efectividad y productividad, reducir costos, aumentar en la satisfacción de clientes y partes interesadas, potenciar la innovación, entre otras (SIG 2013).

Base de Datos

Entre las definiciones enunciadas por diferentes autores que hacen referencia a las bases de datos, se encuentran:

María Mercedes Marqués enuncia: *"Una base de datos es un conjunto de datos almacenados entre los que existen relaciones lógicas y ha sido diseñada para satisfacer los requerimientos de información de*

una empresa u organización. En una base de datos, además de los datos, también se almacena su descripción" (Marqués 2001).

Federico Benjamín Galacho define: *"Una base de datos es una colección o depósito de datos, donde los datos están lógicamente relacionados entre sí, tienen una definición y descripción común y están estructurados de una forma particular. Es también un modelo del mundo real y, como tal, debe poder servir para toda una gama de usos y aplicaciones"* (Benjamín 2009).

Resumiendo, las bases de datos son una colección de datos estructurados, no redundantes, interrelacionados entre sí de manera lógica, que pueden ser variables en el tiempo y se organizan independientemente de su utilización y su implementación en máquinas accesibles en tiempo real y compatible con usuarios concurrentes con necesidad de información diferente.

1.3 Análisis del estado del arte

1.3.1 Sistema de Autenticación de Aplicaciones de la Intranet

La investigación en la Universidad de las Ciencias Informáticas (UCI) sobre los *software* de gestión de seguridad arrojó como resultado la existencia del Sistema de Gestión de Sesiones basado en la arquitectura *SSO*[1] cuyo objetivo es gestionar las sesiones de los usuarios en un único proceso de autenticación invisible a los ojos de los mismos, a través de un sistema con una fachada de servicios *web* que es capaz de gestionar la apertura y cierre de sesiones por parte de las personas que trabajan en el dominio uci.cu. Este sistema no soluciona el problema existente, ya que sólo propone centralizar la autenticación de usuarios y mantiene la autorización como responsabilidad de cada aplicación (Ortega 2006).

1.3.2 Sistema de Gestión de Sesiones

En el 2008 se realizó el trabajo de diploma "Sistema de Gestión de Sesiones (SGS)" el cual se convirtió en una iniciativa viable de protección y control pues permitió ofrecer servicios de valor añadido con altos niveles de seguridad y confianza a los usuarios. La implantación de este sistema, logró mejorar la seguridad de la red en la universidad, permitiendo a su vez a los usuarios disminuir su trabajo de autenticación y el tiempo que se pierde en este proceso. Además en la universidad cada uno de los servicios o aplicaciones cuentan con su propio componente de seguridad, lo cual generalmente compromete la seguridad de todo el sistema, dado que el nivel de seguridad de un sistema es igual al

[1] *SSO: Single Sign On (SSO, por sus siglas en inglés) es una arquitectura de sistemas que le permite al usuario acceder a diferentes aplicaciones con una sola validación de acceso.*

nivel de seguridad del componente más inseguro que tenga. Esta solución de *software* no hace un control de roles ni de usuarios, pero si permite a los usuarios autenticarse una única vez y hacer un control de sesiones eficientes (Hurtado 2008).

1.3.3 Tivoli Identity Manager

IBM Tivoli Identity Manager es una solución de suministro a los usuarios automatizada y basada en políticas que gestiona roles de usuario, identidades y derechos de acceso en toda la infraestructura de TI[2]. Se trata de un *software* de gestión de identidades seguro, fácil de desplegar y utilizar. Ayuda a las empresas a cumplir con las normativas, gestionar riesgos y permitir una colaboración segura. Entre sus características se destacan las siguientes:

- ✓ Gestión de manera automática de roles, cuentas y derechos de acceso en todo el ciclo de vida del usuario, desde la incorporación hasta la finalización, esto reduce los costes generales y elimina los errores manuales.
- ✓ Establece una separación de tareas para reforzar la seguridad y el cumplimiento. Asocia los requisitos que evitan conflictos empresariales con los roles y políticas de suministro que rigen los derechos de acceso de usuario.
- ✓ Corrige y elimina los derechos de acceso que no cumplen con las normativas mediante flujos de trabajo de rectificación periódicos o de forma automática a través de políticas de control de acceso basado en roles. Esta potente característica proporciona más detalles útiles para el auditor a fin de demostrar la conformidad (Figueras 2012).

1.3.4 Sistema de Gestión Integral de Seguridad ACAXIA

Este sistema cuenta con un componente Compartimentación de la información con el que se controla de forma más estricta el acceso a la información que manejan los sistemas suscritos a ACAXIA. Teniendo en cuenta la Multi-entidad que gestiona el sistema, se decide establecer reglas a los permisos que tienen asignados los usuarios con un mismo rol en una entidad determinada. Estas reglas evitan que un usuario manipule información perteneciente a otro usuario con un mismo rol en una misma entidad. Dicha solución pretende delimitar y profundizar el nivel de autorización de los permisos otorgados a usuarios con un mismo rol en una entidad en dependencia de las funcionalidades que estos realizan, logrando con ello un mejor control de acceso a la información.

[2] *TI: Tecnología Informática.*

Este sistema presenta como principales inconvenientes:

- ✓ Desarrollado en un marco de trabajo diferente al empleado en el Sistema de Gestión Universitaria provocando que se dificulte su integración dado el marcado coste arquitectónico.
- ✓ No tiene en cuenta una arquitectura que contemple múltiples bases de datos imposibilitando el uso de este sistema, debido a que en el Sistema de Gestión Universitaria existe una arquitectura de bases de datos distribuidas que se comunican entre sí.

1.3.5 Sistema de Autenticación y Control de Acceso para la Aduana

Este sistema implementa un componente con estructura de *plugin*[3], usando la arquitectura de *Symfony*. El mismo está conformado por una estructura de directorio que cumple con las especificaciones requeridas por el *framework*[4] usado. El control de acceso al sistema, subsistemas, funcionalidades y acciones se maneja a nivel de usuarios y roles. Cada rol tiene un nivel de acceso al sistema descrito en las acciones que puede ejecutar en cada uno de los módulos y aplicaciones. El sistema tiene un registro de la relación tanto entre los usuarios y los roles, como entre los roles y las aplicaciones. La información referente a los usuarios se maneja mediante sesiones en tiempo de ejecución de la aplicación. El manejo de las sesiones de los usuarios se realiza a partir de la clase *sfUser* o una que extienda de ella. La utilización de la clase seleccionada para esto debe ser especificada en un sistema de seguridad como clase manejadora de sesiones por defecto. Para modelar el chequeo de la seguridad en las peticiones de los usuarios a las aplicaciones se comprueba el rol del usuario y el nivel de acceso sobre la funcionalidad que se está accediendo. Para manejar este mecanismo se usa la siguiente variante: El control de acceso a las aplicaciones se realiza en el controlador frontal de cada aplicación, apoyándose en los filtros de *Symfony* que ejecutan el mecanismo de autenticación en el Sistema de Seguridad. Los filtros se incluyen en la cadena de filtros del sistema y estos se encargan de chequear el acceso del usuario autenticado contra la aplicación/módulo/acción a la que se esté accediendo.

El sistema interactúa con los sistemas que estén instalados sobre el mismo proyecto, en el mismo servidor de aplicaciones de las funcionalidades a las que se controla la seguridad. Está almacenado en forma de *plugin* en el proyecto, permitiendo el acceso de sus funcionalidades a las aplicaciones que estén bajo el dominio del mismo proyecto. Para el registro de los subsistemas y sus funcionalidades, el sistema carga la estructura de cada aplicación con sus respectivos módulos y acciones, que se encuentran en un archivo

[3] *Un plugin es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.*

[4] *Framework: Marco de trabajo.*

de configuración *YAML*[5]. Posteriormente se guarda en la base de datos que está estructura en forma de árbol jerárquico (Naranjo 2010).

1.3.6 Subsistema de Gestión de Seguridad para el Sistema de Gestión Académica de Pregrado (AKADEMOS)

En el año 2005 se desarrolló el trabajo de diploma titulado "Subsistema Gestión de Seguridad" donde es expuesto un sistema que permite la gestión automatizada de los elementos que intervienen en la labor académica de la Universidad de las Ciencias Informáticas garantizando la confidencialidad, disponibilidad e integridad de la información manejada, así como un mayor control sobre las acciones de los usuarios en el sistema, basando su chequeo en un conjunto de roles definidos para los usuarios y un conjunto de acciones asociadas. Este sistema presenta como principales inconvenientes primeramente su reducido campo de acción dado que solo se enmarca en los procesos de gestión de pregrado y además está desarrollado sobre herramientas privativas como son: *Microsoft.NET* y el gestor de bases de datos *SQLServer* imposibilitando su integración al Sistema de Gestión Universitaria totalmente implementado sobre herramientas libres.

1.4 Valoración del análisis del estado del arte

Durante el análisis del estado del arte se realizó el estudio un conjunto de sistemas existentes en el mundo y en el país, los mismos cuentan con una variedad de funcionalidades en dependencia del área donde son aplicados, esto los vuelve sistemas competentes a la hora de gestionar un conjunto de actividades o servicios en una empresa o institución. Debido a que algunos de estos sistemas entran en la clasificación de *software* privativo, usan otra arquitectura o no implementan un sistema de seguridad para la protección de la información que se pueda aplicar para dar respuesta a las necesidades existentes, no pueden ser incluidos a la solución informática que se desea desarrollar. Es importante destacar que con este estudio se adquirieron un conjunto de conocimientos acerca de sus características, implementación y principios de funcionamiento que sirven como base para guiar el desarrollo de la solución.

[5] *YAML: formato de serialización de datos legibles por humanos inspirado en lenguajes como XML, C, Python y Perl.*

1.5 Lenguajes

1.5.1 Lenguajes de programación del lado del servidor

PHP (versión 5.3)

Es un lenguaje de programación reconocido, ejecutado e interpretado por el servidor. Fue diseñado originalmente para la creación de páginas *web* dinámicas. Es usado principalmente en interpretación del lado del servidor pero actualmente también puede ser usarse desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica. Generalmente se ejecuta en un servidor *web*, tomando el código *PHP* como entrada y creando páginas *web* como salida. Puede ser desplegado en la mayoría de los servidores *web* y en casi todos los sistemas operativos y plataformas sin costo alguno. Es un lenguaje de alta potencia, fácil de usar e incluye la programación orientada a objetos.

Ventajas de *PHP*:

- ✓ Es un lenguaje multi-plataforma.
- ✓ Capacidad de conexión con la mayoría de los manejadores de bases de datos que se utilizan en la actualidad, destaca su conectividad con *PostgreSQL y MySQL*.
- ✓ Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados *ext's* o extensiones).
- ✓ Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios *HTML*[6].
- ✓ Permite crear los formularios para la *web*.
- ✓ Permite las técnicas de programación orientada a objetos (Olson 2010).

1.5.2 Lenguajes de programación del lado del cliente

JavaScript (versión 1.2)

Es un lenguaje de programación interpretado totalmente independiente del servidor. Sencillo de aprender, su código es embebido en el *HTML* o llamado desde otro fichero haciendo fácil la creación de páginas *web* con contenido dinámico. Su mayor potencial está dado porque permite realizar validaciones de datos o elementos del lado del cliente reduciendo de esta manera la carga del servidor y las transacciones. Actualmente es soportado por la mayoría de los navegadores (Javascript 2012).

[6] *HTML: Lenguaje de Marcado de Hipertexto.*

Características:

- ✓ Lenguaje Interpretado: no requiere compilación, el navegador del usuario se encarga de interpretar las sentencias *JavaScript* contenidas en una página *HTML* y ejecutarlas adecuadamente.
- ✓ Orientado a Eventos: cuando un usuario acciona sobre un enlace o mueve el puntero sobre una imagen se produce un evento. Mediante *JavaScript* se pueden desarrollar *scripts* que ejecuten acciones en respuesta a estos eventos.
- ✓ Orientado a Objetos: el modelo de objetos de *JavaScript* está reducido y simplificado, pero incluye los elementos necesarios para que los *scripts* puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador.

1.5.3 Lenguaje de modelado

UML (versión 2.1)

UML es un estándar para describir un plano del sistema, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de *software* reutilizables. Es importante resaltar que *UML* no es para describir métodos o procesos sino que es una técnica de modelado de objetos y como tal supone una abstracción de un sistema para llegar a construirlo en términos concretos.
Se selecciona este lenguaje para el modelado ya que dentro de sus ventajas están:

- ✓ Mejores tiempos totales de desarrollo (de 50 % o más).
- ✓ Modelar sistemas (y no sólo de *software*) utilizando conceptos orientados a objetos.
- ✓ Establecer conceptos y artefactos ejecutables.
- ✓ Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- ✓ Crea un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- ✓ Mejor soporte a la planeación y control de proyectos.
- ✓ Alta reutilización y minimización de costos (Za 2012).

1.5.4 Lenguaje de Marcado de Hipertexto

HTML (versión 4)

HTML es uno de los lenguajes más utilizados para la elaboración de páginas *web*. Permite describir la estructura y el contenido en forma de texto de las páginas, así como para complementar el texto con objetos tales como imágenes. Se describe en forma de etiquetas (<…>), puede representar también hasta cierto punto la apariencia de un documento.

Características de *HTML*:

- ✓ Las etiquetas pueden tener atributos los cuales definen las propiedades del elemento.
- ✓ Los espacios, tabulaciones, líneas en blanco y retornos de carro del documento *HTML* se ignoran, tomándose como un único espacio en blanco. Esto permite añadir espacios para aumentar la claridad del documento.
- ✓ No distingue entre mayúsculas y minúsculas. Cuando es importante hacerlo, como al poner un título o un atributo, hay que ponerlo entre comillas.
- ✓ Tiene reglas estructurales que indican dónde pueden y dónde no pueden ir los elementos, esto obliga a tener un orden lógico en la escritura del código.
- ✓ Las etiquetas tienen que seguir un orden piramidal, las primeras que se abren son las últimas que se cierran (GALEANO 2009).

1.5.5 Lenguaje de Marcas Extensible

XML (versión 1)

XML es un metalenguaje que permite definir lenguajes de marcado adecuados a usos específicos. Aunque a primera vista un documento *XML* puede parecer similar a *HTML* hay una diferencia fundamental: un documento *XML* contiene datos que se auto-definen, o sea no posee etiquetas prefijadas con anterioridad, ya que es el propio diseñador el que las crea, dependiendo del contenido del documento.
Ventajas de *XML*:

- ✓ Es extensible: después de diseñado y puesto en producción, es posible extender *XML* con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin compilación alguna.
- ✓ El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión del lenguaje *XML*. Esto posibilita el empleo de cualquiera de los analizadores disponibles.
- ✓ Si un tercero decide usar un documento creado en *XML*, es sencillo entender su estructura y procesarla.
- ✓ Mejora la compatibilidad entre aplicaciones (Melián 2012).

1.5.6 Hojas de Estilos en Cascada

CSS (versión 3.0)

CSS es un lenguaje formal usado para definir la presentación de un documento estructurado, escrito en *HTML o XML*. Abarca cuestiones relativas a fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo, posicionamiento avanzado y muchos otros elementos.

Ventajas de *CSS*:

- ✓ Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio *web*, con lo que aumenta considerablemente la accesibilidad.
- ✓ Altera la presentación de cada elemento sin tocar el código *HTML*, ahorrando esfuerzo y tiempo de edición.
- ✓ Es relativamente sencillo y fácil de aprender.
- ✓ Los documentos que usan hojas de estilo generalmente resultan más compactos.
- ✓ Las hojas estilo pueden aplicarse de varias maneras y cambiarse formando una cascada con la información de cada una (Álvarez 2013).

1.6 Herramientas

1.6.1 Herramienta para la administración de datos

PgAdminIII (versión 1.14.0)

Es una herramienta de código abierto para la administración de bases de datos *PostgreSQL*, siendo la más completa y popular con licencia *Open Source*. Está escrita en *C++* usando la librería gráfica multiplataforma *wxWidgets*[7], lo que permite que se pueda usan en *Linux, FreeBSD, Solaris, Mac OSX* y *Windows*. Es capaz de gestionar versiones a partir de la *PostgreSQL 7.3* ejecutándose en cualquier plataforma, así como versiones comerciales de *PostgreSQL* como *Pervasive Postgres, EnterpriseDB, Mammoth Replicator* y *SRA PowerGres*.

Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas *SQL*[8] simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de *PostgreSQL* y facilita enormemente la administración. La aplicación también incluye un editor *SQL* con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar *scripts*[9] programados, soporte para el motor de replicación *Slony-I* y mucho más. La conexión al servidor puede hacerse mediante conexión *TCP/IP* y puede encriptarse mediante *SSL*[10] para mayor seguridad (Ubuntu 2008).

[7] *WxWidgets: son bibliotecas multiplataforma y libres para el desarrollo de interfaces gráficas programadas en lenguaje C++.*

[8] *SQL (en español, Lenguaje Estructurado de Consulta): Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.*

[9] *Script: es un programa que puede acompañar a un documento HTML o que puede estar incluido en él. El programa se ejecuta en la máquina del cliente cuando se carga el documento.*

[10] *SSL: Secure Sockets Layer (en español, capa de conexión segura) es un protocolo criptográfico que proporciona comunicación segura por una red, comúnmente Internet.*

1.6.2 Herramienta de diseño

Evolus Pencil (versión 2.0.3)

Es una herramienta gratuita y de código abierto que posibilita la creación de diagramas y prototipos de interfaz gráfica de usuario. Facilita la creación de ventanas de prototipos arrastrando los diferentes elementos ya sea como extensión de *Firefox* o como aplicación estándar para *Windows* o *Linux.* Además, permite hacer presentaciones y diseño de páginas *web* de forma rápida y facilita el diseño de prototipos de interfaz gráfica sin necesidad de tener conocimientos avanzados en el tema.

Características:

- ✓ Permite la edición en pantalla de los elementos de texto.
- ✓ Es multi-plataforma.
- ✓ Es multi-página, creación simultánea de varios documentos.
- ✓ Brinda un conjunto de componentes como: entradas de texto, íconos y botones.
- ✓ Posibilita a través de las propiedades de los componentes cambiar el estilo al diseño.
- ✓ Permite operaciones estándar de dibujo: alineado, escalado, rotación.
- ✓ Permite exportar imágenes al formato *png, html o pdf* (PROJECT 2010).

1.6.3 Herramienta de modelado

Las herramientas *CASE*[11], son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el costo de las mismas en términos de tiempo y dinero. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del *software* en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente a partir del diseño elaborado, compilación automática, documentación o detección de errores. Tienen como objetivos: aumentar la calidad del *software*, mejorar la planificación de un proyecto, mejorar la productividad en el desarrollo y mantenimiento de los sistemas informáticos, facilitar el uso de los distintos procesos de desarrollo o metodologías propias de la ingeniería del *software*, ayudar a la reutilización del *software*, portabilidad y estandarización de la documentación y garantizar una correcta documentación, generación de código, pruebas de errores y gestión del proyecto (ITESCAM 2012).

Visual Paradigm (Versión 8.0)

Es una herramienta *CASE* que utiliza *UML* como lenguaje de modelado. Soporta el ciclo de vida completo del desarrollo de *software*. Ayuda a una más rápida construcción de aplicaciones de calidad, y a un menor

[11] *CASE: Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés).*

coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Características:

- ✓ Entorno de creación de diagramas para *UML*.
- ✓ Diseño enfocado al negocio que genera un *software* de mayor calidad.
- ✓ Uso de un lenguaje estándar y común a todo el equipo de desarrollo facilitando la comunicación.
- ✓ Es amigable, multi-plataforma, posibilita la generación de documentos además de la integración con los distintos Ambientes de Desarrollo Integrados (García 2012).

1.6.4 Herramienta de desarrollo

NetBeans (versión 7.2.1)

Para la implementación de la aplicación es necesario además de un potente lenguaje de programación, un potente *IDE*[12] de desarrollo. El ideal para esto es el *NetBeans*, producto libre y gratuito, sin restricciones de uso, que soporta el desarrollo de todos los tipos de aplicaciones *Java*. La plataforma *NetBeans* permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de *software* llamados módulos. Un módulo es un archivo *Java* que contiene clases de *java* escritas para interactuar con las *Apis*[13] *de NetBeans* y un archivo especial (*manifest file*) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma *NetBeans* pueden ser extendidas fácilmente por otros desarrolladores de *software*.

Características:

- ✓ Administración de interfaces de usuario (ej. menús y barras de herramientas).
- ✓ Administración de las configuraciones del usuario.
- ✓ Administración del almacenamiento (guarda y carga cualquier tipo de dato).
- ✓ *Framework* basado en asistentes (diálogo paso a paso).
- ✓ Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas.
- ✓ Tiene un sistema de proyectos basados en el control de versiones.
- ✓ Contiene un número importante de módulos para extender el *NetBeans IDE (Informático 2012)*.

[12] *IDE: Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés).*

[13] *API: es una librería que agrupa gran cantidad de código de forma que sea fácil de usar para desarrollar programas.*

1.6.5 Servidor Web

Un servidor *web* es un programa que implementa el protocolo *HTTP*[14] y se encarga de contestar de forma adecuada las peticiones de ejecución que realiza un cliente, entregando como resultado una página *web* o información de todo tipo, de acuerdo a los comandos solicitados.

Servidor web Apache (versión 2.2.2)

Es el servidor *web* usado por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. La licencia *Apache* es una descendiente de la licencias *BSD*[15], no es *GPL* (Licencia Pública General). Esta licencia permite hacer lo que se desee con el código fuente (incluso productos propietarios).

Ventajas:

- ✓ Corre en una multitud de sistemas operativos, esto lo hace prácticamente universal.
- ✓ Es un servidor altamente configurable de diseño modular. Actualmente existen muchos módulos para *Apache* que son adaptables a este, y pueden ser instalados cuando sea necesario.
- ✓ Permite personalizar la respuesta ante los posibles errores que puedan ocurrir en el servidor.
- ✓ Es una tecnología gratuita de código fuente abierto (Castro 2012).

1.6.6 Sistemas Gestores de Bases de Datos (SGBD)

Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. Las características de un SGBD son: control de la concurrencia, independencia, consistencia, seguridad, integridad, manejo de transacciones, respaldo y tiempo de respuesta. (**Ver Anexo 1**)

PostgreSQL (versión 8.4.1)

Es un potente gestor de base de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de base de datos comerciales. Soporta gran parte del estándar *SQL* y en algunos aspectos, está diseñado para que sea extensible por los usuarios. Posee interfaces gráficas de usuario y enlazadores para algunos lenguajes de programación.

---

[14] *HTTP: Protocolo de transferencia de hipertextos. Pertenece a la capa de aplicación del modelo OSI y está diseñado para transferir lo que llamamos hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.*

[15] *BSD: es una licencia de software libre permisiva y tiene menos restricciones en comparación con otras, estando muy cercana al dominio público.*

Entre las ventajas PostgreSQL se encuentran:

- ✓ Alta concurrencia.
- ✓ Amplia variedad de tipos nativos.
- ✓ Multi-plataforma.
- ✓ Estabilidad y confiabilidad.
- ✓ Instalación ilimitada (Ubuntu 2011).

1.7 Proceso de desarrollo de software

Un proceso de desarrollo de *software* es un método de organización de las actividades relacionadas con la creación, presentación y mantenimiento de los sistemas de *software*. Es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto. Este proceso de desarrollo es extremadamente importante porque tiene como propósito la producción eficaz y eficiente de un producto que agrupe los requisitos del cliente. Así el resultado final del *software* será el más óptimo para los proyectos productivos (Pressman 2002).

1.7.1 Modelo de Capacidad y Madurez Integrado

En 1991 el Instituto de Ingeniería de *Software* crea el Modelo de Capacidad y Madurez *(CMM, por sus siglas en inglés)* y en el 2002 es actualizado como Integración de Modelos de Madurez de Capacidades (*CMMI*, por sus siglas en inglés). *CMMI* está orientado a garantizar la calidad del *software*, y a la acreditación de las empresas que se dedican al desarrollo del mismo en dependencia del nivel de madurez de sus procesos de producción. Representa la unión de un conjunto de modelos orientados a la mejora de procesos de ingeniería de *software*, ingeniería de sistemas, desarrollo de productos y adquisición de aplicaciones. Con su implementación se logra un aumento de la visibilidad de los procesos de producción y soporte, la reusabilidad de componentes, fiabilidad del *software* producido, obteniéndose como resultado de la combinación de este tipo de mejoras la disminución de los costes de producción y mantenimiento de las aplicaciones (ISE 2011).

Tiene dos representaciones:

- ✓ Continua: se centra en la mejora de un proceso o un conjunto de procesos, brindando un enfoque flexible para la mejora de los procesos.
- ✓ Escalonada: brinda una manera estructurada y sistemática de enfrentar la mejora de procesos, ejecutando un paso cada vez.

Estas dos representaciones son equivalentes y cada empresa puede adoptar la que mejor se adapte a sus prioridades de mejoras o a sus características. En el caso de la Universidad de las Ciencias Informáticas se utiliza la escalonada, ya que esta cuenta con cinco niveles de madurez:

1- Inicial o nivel 1.
2- Repetible o nivel 2.
3- Definido o nivel 3.
4- Gestionado Cuantitativamente o nivel 4.
5- Optimizado o nivel 5.

El nivel 2 está dividido en 7 áreas de procesos:

- ✓ Planificación de Proyectos.
- ✓ Seguimiento y Control del Proyecto.
- ✓ Gestión de Acuerdos con Proveedores.
- ✓ Medición y Análisis.
- ✓ Aseguramiento de la Calidad del Producto y el Proceso.
- ✓ Gestión de la Configuración.
- ✓ Administración de Requisitos.

1.7.2 Proceso de desarrollo con enfoque ágil basado en el nivel 2 de CMMI

La Universidad de las Ciencias Informáticas es considerada la mayor organización productora de *software* en el país. Como centro productivo tiene la misión de elaborar *software* y servicios informáticos a partir de la vinculación estudio-trabajo como modelo de formación. En la actualidad el centro está acometiendo un programa de mejora de sus procesos, con el objetivo de que la universidad alcance una certificación internacional del nivel 2 de *CMMI*. Este hecho la convertiría en la primera empresa cubana certificada con dicho modelo y una de las pocas en el área del Caribe y Centro América. Este proceso de desarrollo tiene como objetivo asegurar que la organización está basada en procesos y con un programa de mejora continua alineado con sus objetos de negocio. Este programa de mejoras cuenta con varias fases para el desarrollo del producto (**Ver Anexo 2**) y quedó determinado a partir de la utilización de prácticas bien definidas de las metodologías ágiles *XP* y *Scrum*.

XP (Programación Extrema)

Es una metodología ágil de desarrollo de *software* que posee cuatro tareas fundamentales: planificación, diseño, desarrollo y pruebas. Esta metodología está basada en la simplicidad durante el desarrollo, la comunicación entre las partes implicadas (clientes y desarrolladores) y la retroalimentación para poder reutilizar el código desarrollado. Establece entregas frecuentes con posibilidad de refactorización continua, permitiendo mejorar el diseño cada vez que se añade una funcionalidad. Para su implementación *XP*

establece las siguientes prácticas: el juego de la planificación, entregas pequeñas, metáforas, diseño simple, pruebas, programación en parejas, propiedad colectiva del código, cliente en su lugar, estándares de programación (Llano 2012).

Scrum

Es una metodología ágil enfocada a la gestión de proyectos. Sus principales características se pueden resumir en dos: el desarrollo de *sprint* o iteraciones y reuniones a lo largo del desarrollo. La evolución del proyecto por la metodología se define a través de reuniones diarias donde el trabajo del día anterior es revisado por el equipo, previendo además la labor a realizar el día siguiente. Dentro de las prácticas definidas por la metodología *Scrum* se encuentran: planificación de la iteración o *sprint*, revisión de la iteración, reunión diaria, pila del producto, incremento, propietario del producto.

Con el propósito de ayudar a la universidad a establecer las bases y los fundamentos para seguir mejorando sus procesos y fortaleciendo su cultura de calidad en el desarrollo de *software* se adopta el programa de mejora como proceso de desarrollo para guiar la solución informática (Llano 2012).

1.8 Marco de trabajo

Un marco de trabajo *(framework)* es un esquema para la implementación de una aplicación. Simplifica el desarrollo de la misma mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. En el **Anexo 3** se muestran algunas de las ventajas que tiene el uso de un marco de trabajo.

1.8.1 GUUD (versión 1.0)

GUUD es el marco de trabajo propuesto para el desarrollo de la solución informática, el mismo fue creado por el equipo de arquitectura del CENIA. Este consiste en la integración en una sola estructura de los *frameworks CodeIgniter (versión1.7.3) y JQuery (versión1.3.2)*. En la unión se incluyen además un conjunto de mejoras y algunas modificaciones realizadas específicamente a *CodeIgniter*.

CodeIgniter (versión 1.7.3)

CodeIgniter es un *framework* para el desarrollo de aplicaciones *web* usando *PHP*. Este permite el desarrollo de proyectos mucho más rápidos que si se escribiera código desde cero. Provee una rica colección de librerías para las tareas necesarias más comunes. Permite concentrase en el desarrollo del proyecto en cuestión, minimizando la cantidad de código necesaria para realizar las tareas. Usa el patrón de diseño arquitectónico Modelo-Vista-Controlador como paradigma de arquitectura de desarrollo y está creado para que sea fácil de instalar y usar en cualquier servidor (García 2012).

JQuery (versión 1.3.2)

JQuery es una librería de código *JavaScript* creada inicialmente por *John Resig*, que permite simplificar la manera de interactuar con los documentos *HTML*, manipular el árbol *DOM*[16], manejar eventos, desarrollar animaciones y agregar interacción con la técnica *AJAX*[17] a páginas *web*. *JQuery* es *software* libre y de código abierto, posee un doble licenciamiento bajo la licencia *MIT*[18] y la Licencia Pública General de *GNU* (versión 2), permitiendo su uso en proyectos libres y privativos. Ofrece una serie de funcionalidades basadas en *JavaScript* que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio (Rendón 2012).

Novedades que incorpora GUUD

GUUD en su infraestructura incorpora un conjunto de mejoras y modificaciones, a continuación se muestra una relación de las mismas.

Del lado del cliente:

- ✓ Se implementó un *plugin* para *JQuery* que permite el manejo de espacios de nombre e internacionalización.
- ✓ Se implementaron *widgets*[19] para utilizarlos de interfaz de algunos de los que ya posee *jquery-ui* como por ejemplo el *date*, el *tab* (ambos son interfaces de los *widgets* de mismo nombre de *jquery-ui*) y el *popup* (interfaz del *dialog* de *jquery-ui*). Además de los ya mencionados se implementaron otros nuevos entre los que se encuentran: *attach*, *menu*, *message*, *tooltip*, *form* (se construyó con la unión de los *plugins form* de *JQuery* el cual se utiliza para el envío de formularios *AJAX* y el *validate* utilizado para validar formularios), *grid* (utiliza como *plugin* el *jqgrid*), *multiselect* (para hacer selecciones múltiples), *navbar* (para la creación de barras de navegación), *tree* (para la creación de árboles) y el *graph* (utiliza la librería *Highchart*).
- ✓ Se implementaron funciones comunes para todo el sistema (contenidas en los archivos *core.js* y *common.js*) entre las que se destacan: *loadIn*, *getDataJson*, *fromJson*, *toJson*, *createSelect*, *isArray*, *isFunction* y *site_url*.

[16] *DOM: el Modelo de Objetos del Documento es una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML. Es un modelo estándar sobre cómo pueden combinarse dichos objetos y una interfaz estándar para acceder a ellos y manipularlos.*

[17] *AJAX: es una técnica de desarrollo web para crear aplicaciones interactivas.*

[18] *MIT: es una licencia de software, no tiene copyright, lo que permite su modificación y es muy parecida a la licencia BSD en cuanto a efectos.*

[19] *Widgets: pequeñas aplicaciones o programas, usualmente presentados en archivos o ficheros pequeños, cuyo principal objetivo es dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.*

Del lado del servidor:

- ✓ Se le agregó manejo de excepciones y mensajes.
- ✓ Se le implementó el *IOC* (Inversión de Control) para la interacción entre módulos.
- ✓ Se le añadió la característica de la modularidad o sea que una aplicación pueda dividirse en módulos. *CodeIgniter* no cuenta con esta posibilidad.
- ✓ Se añadieron los plugins export_pi (permite exportar a los formatos: *pdf, csv y xls*) e import_pi (permite importar desde archivos en formatos *csv o xls*).
- ✓ Se añadieron, modificaron y extendieron los *helpers*[20] *o* asistentes entre los que se encuentran como añadidos: *template* (brinda la posibilidad de usar plantillas, característica que no posee *CodeIgniter*, para esto se añadió también la librería *template*), *assets* (utilizado para la integración en las vistas de *javascript, css,* imágenes y el *template*), *grid y json* y como modificados: *form*, *array* y *security* (García 2012).

1.9 Conclusiones parciales

Con el estudio de los procesos de seguridad de negocio se obtuvo el conocimiento necesario acerca de las características fundamentales de dichos procesos relacionados con el filtrado de información. Al realizar el análisis de algunas soluciones existentes no se encontró ninguna que cumpliera con las condiciones necesarias, respondiera a todas las necesidades y exigencias del cliente y pudiera ser adoptada por el mismo, aunque se adquirió un conjunto de conocimientos sobre sus principales características, constituyendo una base para el desarrollo de la propuesta de solución. El estudio de las herramientas, lenguajes de programación y proceso de desarrollo a utilizar permitió la familiarización con los elementos del entorno de desarrollo y la obtención de conocimientos necesarios acerca de sus características generales. Teniendo en cuenta todo lo planteado anteriormente se toma la decisión de utilizarlas debido a que son en su mayoría multi-plataforma y se encuentran bajo licencia gratuita o libre. Por lo que se decide realizar una propuesta de solución usando el lenguaje *PHP*, GUUD como marco de trabajo, *UML* como herramienta de modelado, *Visual Paradigm* como herramienta *CASE*, *Apache* como servidor *web*, *PostgreSQL* como gestor de base de datos y se propone el proceso de desarrollo basado en el nivel 2 de *CMMI* para guiar la solución informática.

[20] *Los helpers o asistentes, como su nombre lo indica, son una colección de funciones de una categoría particular que ayudan en la realización de determinadas tareas.*

Capítulo 2. Diseño de la solución

2.1 Introducción

El diseño de un *software* es la descripción de la estructura del *software* que se va a implementar, los datos que son parte del sistema y las interfaces entre los componentes del sistema. En este capítulo se realiza una propuesta general de la solución a desarrollar para el Sistema de Gestión Universitaria (SGU) después de haber analizado detalladamente los procesos de seguridad de negocio enfocados hacia el filtrado de información. Se realiza el análisis de los componentes y de los requisitos tanto funcionales como no funcionales que formarán parte del producto final, además, es descrita la arquitectura a utilizar y se puntualiza el proceso de obtención de la base de datos que controla la seguridad de negocio dentro del sistema atendiendo a las regulaciones definidas.

2.2 Modelo de Dominio

Teniendo en cuenta que la solución informática formará parte del módulo Seguridad del SGU, y que básicamente su función será el filtrado de datos para reforzar la seguridad del negocio existente, no existe un negocio bien definido. Esto, unido a que el sistema se encuentra aún en desarrollo ha propiciado que en lugar de un modelado de negocio se haga un modelado de dominio.

El Modelo de Dominio es la representación de los conceptos más importantes y significativos en el desarrollo de un sistema. Este representa clases conceptuales del dominio del problema y conceptos del mundo real, no de los componentes del *software*. El objetivo fundamental del mismo es definir las interrelaciones de los objetos más importantes representados mediante clases. Desempeña un papel central en la comprensión del entorno actual y en la planificación futura de la posible aplicación. Además, contiene las principales clases identificadas, con el objetivo de proveer una representación gráfica del sistema más fácil de comprender para el cliente. A continuación se muestra la descripción de las clases representadas en el modelo de dominio.

Administrador: Clase encargada de definir permisos y grupos de permisos aplicables a la información accedida por el usuario. Encargada de la gestión de las categorías de los filtros.

Permiso: Clase encargada de guardar el filtro asociado y los objetos de negocio que restringen el acceso a la información.

Grupo de permisos: Clase encargada de almacenar un conjunto de permisos y los usuarios a los que les será restringido el acceso a la información por dichos permisos.

Filtro: Clase que almacena el módulo, el usuario y la información que serán restringidos.

Categoría: Clase encargada de almacenar las categorías de los filtros.

Usuario: Clase que almacena la información de los usuarios a los cuales se les va a asociar los permisos.
Información: Clase que tiene toda la información que se gestiona en el sistema.
Módulo: Clase que almacena la información relacionada con el módulo del sistema al que pertenecerán los permisos.
Teniendo en cuenta la identificación de las clases anteriormente explicadas se pudo estructurar el modelo de dominio que se presenta en la Figura 1:

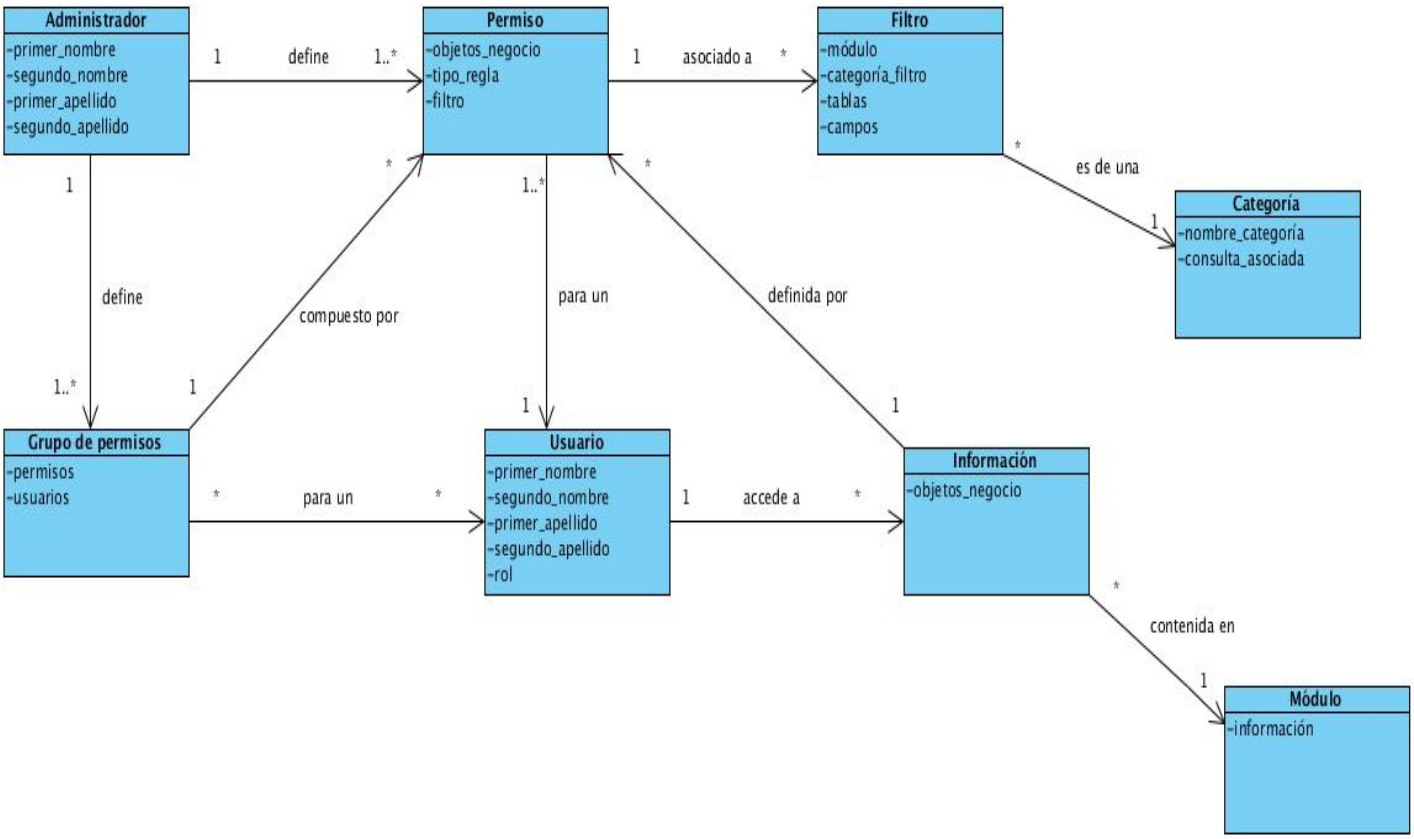


Figura 1. Modelo de Dominio

2.3 Descripción de la propuesta de solución

En busca de mejorar el control del acceso a la información en el Subsistema de Gestión Académica de Pregrado (SGAP) se decide crear una solución basada en el filtrado de datos que refuerce la seguridad de negocio existente. Como resultado se obtendrá una herramienta de configuración integrada al SGU que permitirá al administrador definir los permisos de acceso a cada usuario en particular, además gestionará las categorías de los filtros permitiendo hacer la solución extensible a otros sub-sistemas que decidan utilizar esta herramienta para controlar el acceso a sus datos. Se pretende con esta serie de acciones

reducir el cúmulo de información accesible por los usuarios solo al radio de acción definido por los permisos que le son asignados.

La propuesta parte del momento en que se creen las categorías de filtros, estas serán utilizadas posteriormente en la configuración de los filtros donde son asociadas a un módulo específico dentro del sistema y a un conjunto de tablas y procedimientos almacenados. Los filtros son utilizados a la hora de configurar los permisos donde son asignados los objetos de negocio a los que el usuario tendrá acceso. Los permisos a su vez pueden agruparse en grupos de permisos que son asignados a varios usuarios dentro del sistema, estos son inyectados a modo de condición en las consultas dentro de los modelos que llaman a la categoría a la que pertenecen los filtros asignados restringiendo así el acceso de los usuarios únicamente a la información previamente definida dentro de su radio de competencia.

2.4 Integración de la propuesta de solución al Sistema de Gestión Universitaria

El SGU es una solución integral que se dedica a gestionar los procesos de la universidad, teniendo en cuenta lo anterior se hace necesario crear una solución que pueda reforzar la seguridad existente en todo el sistema. Por ello la solución informática Seguridad de Negocio debe acoplarse a dicho sistema. Esta integración provee múltiples ventajas como la garantía de que a nivel de información se controle el acceso a los objetos del negocio del sistema, así como a los flujos de procesos activos evitando que se consulte información ajena al radio de acción del usuario o que este ejecute procesos indebidos sobre la información.

La solución informática se integra con el núcleo del SGU. El núcleo es el encargado de gestionar las funcionalidades horizontales del sistema, permitiendo estandarizar las mismas para ser usadas por cada uno de sus subsistemas, específicamente se integra con los siguientes módulos:

Seguridad: Es el encargado de garantizar la seguridad de todo el SGU, permite la autenticación de usuarios, gestión de roles, usuarios, dominios y modo de acceso. Brinda además un grupo de políticas de accesibilidad a las diferentes funcionalidades del sistema en dependencia del nivel de autorización que presente un usuario determinado.

Estructura y Composición: Es el encargado de gestionar la información referente a toda la estructura administrativa y la jerarquía de la misma, así como la asignación de responsabilidades a las estructuras.

Configuración: Es el encargado de gestionar la información necesaria para realizar las configuraciones de los procesos académicos.

Eventos: Es el encargado de gestionar los procesos de gestión de un evento.

Documentos Acreditativos: Es el encargado de gestionar la información de los documentos que se generan en el sistema como evaluaciones, certificados, actas, entre otros.

Inmuebles: Es el encargado de gestionar los inmuebles de la institución y sus características.

Además de esto, la solución informática se integra con el SGAP, este es el encargado de gestionar los procesos de formación de pregrado en la universidad, el cual incluye los procesos de diseño y gestión de carreras, actividades de secretarías, registro y control docente, la gestión de los trabajos de diploma y títulos y la planificación y control del proceso docente. Específicamente se integra con los siguientes módulos:

Carrera: Es el encargado de la gestión de los elementos que conforman los planes de estudios definidos por el MES[21], acorde a las necesidades y características de la entidad. De él se obtienen elementos importantes para el control docente como son: formas de evaluación, tipos de evaluación, tipos de bonificaciones, modalidades, tipos de cursos y las asignaturas con sus versiones.

Personal y Secretaría: Es el encargado de gestionar todo el personal vinculado con los procesos de Pregrado. Posibilita la realización de movimientos de un estudiante (traslados, bajas, licencias), así como acciones de secretaría (promoción de estudiantes, registro de datos docentes para profesores, solicitudes de movimiento de alumnos ayudantes).

Control Docente: Es el encargado de gestionar todo el registro de las evaluaciones docentes y asistencia de los estudiantes, asignar profesores a las diferentes estructuras docentes, establecer un balance de la carga docente de los profesores así como posibilitar un mejor control de los profesores por facultad y grupo.

Estudiante: Es el encargado de brindar información sobre las evaluaciones y promedios de los estudiantes en los diferentes años de la carrera y períodos lectivos, información no obtenida desde el módulo de Control Docente.

Reportes: Permite obtener información en forma de gráficas o listados sobre elementos de la gestión académica incluyendo los de control docente, como reportes sobre las evaluaciones de los estudiantes que obtiene de Control docente, y los muestra según los criterios especificados para estudiantes o grupos docentes.

Tesis y Título: Es el encargado de gestionar todo el proceso de desarrollo de una tesis de pregrado como asignación, seguimiento y control y el otorgamiento de los títulos.

[21] *Ministerio de Educación Superior*

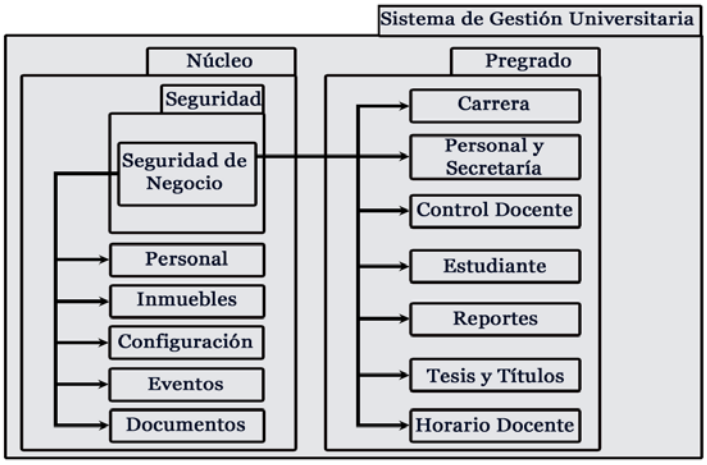


Figura 2. Mapa de integración

2.5 Definición de requisitos

Un requisito es una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este. Los requisitos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema (Sommerville 2005).

2.5.1 Ingeniería de Requisitos

La Ingeniería de Requisitos es el proceso de descubrir, recopilar, analizar, documentar y verificar las necesidades del cliente o usuario para un sistema. Facilita la comunicación entre los clientes y usuarios del *software* y los desarrolladores del mismo. Trata de establecer lo que el sistema debe hacer, sus propiedades emergentes deseadas y esenciales, y las restricciones en el funcionamiento del sistema y los procesos de desarrollo de *software*. Incluye un conjunto de tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un *software*, tomando en cuenta los diversos requisitos. Tiene como meta crear, mantener y entregar una especificación de requisitos de *software* de forma completa y correcta (Pressman 2002).

2.5.2 Técnicas de obtención de requisitos

La obtención de requisitos es el proceso mediante el cual los interesados en un sistema de *software* descubren, revelan, articulan y entienden sus requisitos. En muchos casos, se requiere tiempo para llegar a especificar claramente lo que el interesado espera de la aplicación de *software*, por lo que se hace necesario por parte de los analistas el empleo de técnicas que permitan establecer una buena comunicación con los interesados del producto y así lograr la satisfacción del cliente.

A continuación se enuncian las principales técnicas utilizadas para recopilar los requisitos.

- Observación:

Se utiliza para entender los requisitos sociales y organizacionales. Se centra fundamentalmente en la forma de trabajar de las personas y no en como el sistema los hace trabajar.

- Entrevista:

Se emplean para reunir información proveniente de personas o de grupos. Durante la entrevista el analista se reúne con el encuestado y ocurre el intercambio de preguntas y respuestas para extraer el dominio de la aplicación.

- Prototipos:

Permite al desarrollador crear un modelo del *software* que debe ser creado. En este método se obtienen los requisitos preliminares que pueden ser utilizados para construir una versión inicial. Este modelo es mostrado al cliente, quien proporciona los requisitos adicionales.

- Tormenta de Ideas:

Esta técnica se utiliza con el objetivo generar ideas en un ambiente libre de críticas o juicios. Ayuda a concebir una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de captura, cuando los requisitos son todavía muy difusos (Aspajo 2010).

2.5.3 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. Los requisitos funcionales describen explícitamente lo que el sistema debe hacer.

En la siguiente tabla se muestran los requisitos funcionales que se definieron para la realización de la solución informática.

Tabla 1. Requisitos funcionales

No	Requisito funcional	Complejidad
RF_1	Crear categoría de filtro.	Alta
RF_2	Listar categoría de filtro	Media
RF_3	Ver detalles de categoría de filtro.	Media
RF_4	Modificar categoría de filtro.	Alta
RF_5	Listar reglas de negocio.	Media
RF_6	Asociar filtros a módulo.	Alta
RF_7	Crear permiso.	Alta
RF_8	Listar permiso.	Media

RF_9	Ver detalles del permiso.	Media
RF_10	Modificar permiso.	Alta
RF_11	Eliminar permiso.	Media
RF_12	Crear grupo de permisos.	Alta
RF_13	Listar grupo de permisos.	Media
RF_14	Ver detalles del grupo de permisos.	Media
RF_15	Modificar grupo de permiso.	Alta
RF_16	Asociar grupo de permiso a usuario.	Alta
RF_17	Agregar usuario a un grupo de permisos.	Alta
RF_18	Desagregar usuario de un grupo de permisos.	Alta
RF_19	Asociar permiso a usuario.	Alta

Especificación de requisitos funcionales

La especificación es el producto del trabajo final que genera la ingeniería de requisitos, describe la función y el desempeño de un *software* y las restricciones que regirán su desarrollo. Además, sirve de base para las actividades de ingeniería subsecuentes, en este caso los requisitos quedan especificados a través de una plantilla donde se combinan descripciones en el lenguaje natural y modelos gráficos, quedando representados de una manera más consistente y entendible.

El **Anexo 5** muestra todas las especificaciones de requisitos. La siguiente tabla muestra un ejemplo de la especificación del requisito: Modificar categoría de filtro.

Tabla 2. Especificación de requisito. Modificar categoría de filtro

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_4	Modificar categoría de filtro.	El requisito permite modificar una categoría de filtro. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción Categorías de Filtros. En el listado de las categorías de filtros que muestra el sistema se selecciona la acción interna Modificar Categoría.	Alta	Alta

		Se muestran los datos almacenados de dicha categoría: Nombre, Nombre físico, Descripción, Estado (Habilitado/Deshabilitado), Tipo visual. Se modifican los datos.		

Prototipo

Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• Varchar	• Es un campo obligatorio. • No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Nombre físico	• Varchar	• Es un campo obligatorio. • No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Descripción	• Varchar	• Admite entre 0 y 100 caracteres válidos. • Admite cualquier tipo de carácter. • Solo admite 30 caracteres por palabra.

• Estado	• Boolean	• Selección. • Es un campo obligatorio.
• Tipo visual	• Varchar	• Selección. • Es un campo obligatorio.

2.5.4 Requisitos no funcionales

Los requisitos no funcionales, son aquellos requisitos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. Surgen debido a la necesidad de interoperabilidad con otros sistemas, *software* o *hardware*, o a factores externos como regulaciones de seguridad o legislaciones sobre privacidad. De manera general son las restricciones de los servicios o funciones que ofrece el sistema, incluyendo las restricciones sobre el proceso de desarrollo y estándares (Sommerville 2005). Los requisitos no funcionales definidos para la realización de la solución informática fueron clasificados, la siguiente tabla muestra dicha clasificación y el **Anexo 6** muestra la especificación de los requisitos no funcionales.

Tabla 3. Clasificación de los requisitos no funcionales

Clasificación	Cantidad
Usabilidad	5
Seguridad	3
Eficiencia	2
Soporte	2
Hardware	3
Documentación de usuarios en línea y ayuda del sistema	2
Interfaz	3
Restricciones de diseño	5
Total	25

2.5.5 Plan de iteración

Un plan de iteración tiene como objetivo definir detalladamente para cada una de las iteraciones a realizarse un conjunto de tareas o actividades. El siguiente plan de iteración tiene como entrada las especificaciones de requisitos y establece las iteraciones necesarias para desarrollar el producto.

Además, define los requisitos que se deben implementar en cada una de las iteraciones. El objetivo de cada iteración es obtener una versión funcional del sistema que, aunque no cuente con todos los requerimientos definidos, constituye un resultado valioso para el cliente.

Tabla 4. Plan de iteración

Iteración	Descripción	Requisitos a implementar	Duración total
Iteración 1	En esta iteración se va implementar una parte de los requisitos que tienen prioridad alta para el cliente.	RF_1, RF_4, RF_6, RF_7, RF_10, RF_12.	5/2/2013-15/3/2013 (1 mes, 10 días)
Iteración 2	En esta iteración se van a implementar el resto de los requisitos que tienen prioridad alta para el cliente.	RF_15, RF_16, RF_17, RF_18, RF_19.	20/3/2013-20/4/2013 (1 mes)
Iteración 3	En esta iteración se van a implementar los requisitos de prioridad media para el cliente.	RF_2, RF_3, RF_5, RF_8, RF_9, RF_11, RF_13, RF_14.	22/4/2013-11/5/2013 (20 días)

2.6 Arquitectura de software

La arquitectura de *software* es a grandes rasgos, una vista del sistema que incluye los principales componentes del mismo, la conducta de esos componentes y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema (Viera 2010).

Debido a que la solución informática formará parte del módulo Seguridad del SGU, además se integrará al SGAP y se espera que se extienda a todo el SGU se hace necesario que la misma cumpla con las pautas especificadas para el desarrollo del sistema. Teniendo en cuenta lo planteado se propone utilizar como estilo arquitectónico la Arquitectura Cliente-Servidor y como patrón arquitectónico el Modelo-Vista-Controlador (patrón arquitectónico usado por el marco de trabajo GUUD).

2.6.1 Estilo Arquitectónico

Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. En esta arquitectura las aplicaciones confluyen basándose en dos categorías las cuales son el cliente y el servidor.

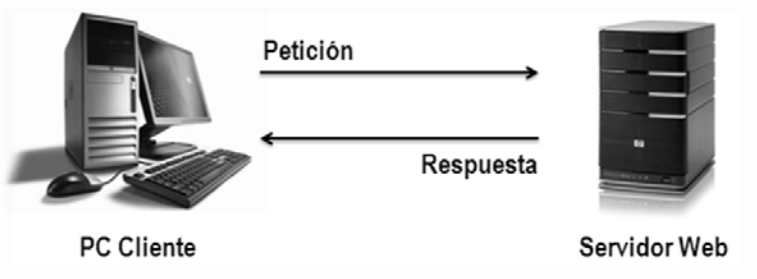


Figura 3. Arquitectura Cliente-Servidor

Características del Cliente:

- ✓ Es el proceso que inicia el diálogo o la solicitud de recursos.
- ✓ Puede conectarse a varios servidores a la vez.
- ✓ Espera y recibe las respuestas del servidor.
- ✓ Interactúa con los usuarios finales mediante una interfaz gráfica de usuario.
- ✓ No conoce la ubicación de los datos o de las aplicaciones.

Características del Servidor:

- ✓ Es cualquier recurso de cómputo dedicado a recepcionar y procesar una solicitud y luego enviar la respuesta al cliente.
- ✓ Al iniciarse espera a que lleguen las solicitudes de los clientes.
- ✓ Aceptan conexiones desde un gran número de clientes.
- ✓ Frecuentemente no interactúan con usuarios finales (Carbajal 2012).

2.6.2 Patrón Arquitectónico

Modelo-Vista-Controlador

El Modelo Vista Controlador (MVC, por sus siglas en inglés) es un patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos permitiendo analizar una programación multicapa. Generalmente es usado en aplicaciones *web*, dónde la vista es la página *HTML* y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio (García 2012).

MVC implementado por GUUD

El marco de trabajo GUUD, implementa el MVC con las siguientes particularidades: cuenta con un controlador frontal que inicializa los recursos básicos necesarios para correr el *CodeIgniter* (parte estructural del GUUD). Una vez realizada una petición *HTTP* por un cliente, el controlador frontal se encarga de analizar la *URI* y a partir de esta determina cuál controlador de aplicación (controlador de un

determinado módulo) debe ser cargado para atender la petición realizada. Cada controlador de aplicación tiene asociada una o varias librerías responsables de procesar los datos e implementar la lógica del negocio inherente a las acciones relacionadas con dicho controlador. De manera similar cada librería tiene asociado uno o varios modelos encargados del acceso a los datos.

Cuando un controlador de aplicación es cargado, este examina la petición para determinar si solo debe cargar una vista determinada o si es necesario interactuar con la base de datos. En este último caso el controlador de aplicación envía los datos recibidos a la o las librerías. Estas a su vez, cargan los modelos necesarios para obtener, registrar o actualizar en la base de datos la información solicitada o enviada. Para realizar esta tarea los modelos hacen uso de la clase *Active Record*. Esta clase permite consultar la base de datos con una reducida codificación y abstrayéndose del gestor que se use. La sintaxis de la consulta es generada por cada adaptador de base de datos. Cuando los datos son obtenidos, se retornan al controlador de aplicación en un proceso inverso al descrito anteriormente. Posteriormente, el controlador carga estos datos a archivos escritos en *HTML* los cuales pueden incluir llamadas a archivos escritos en *JavaScript* para manejar dinámicamente su contenido o hacer uso de asistentes (*helpers*) para la creación de forma simplificada de código *HTML*. Finalmente, el resultado obtenido de todo este proceso es enviado al navegador *web* como respuesta a la petición inicial.

La siguiente figura muestra el patrón Modelo-Vista-Controlador implementado por GUUD.

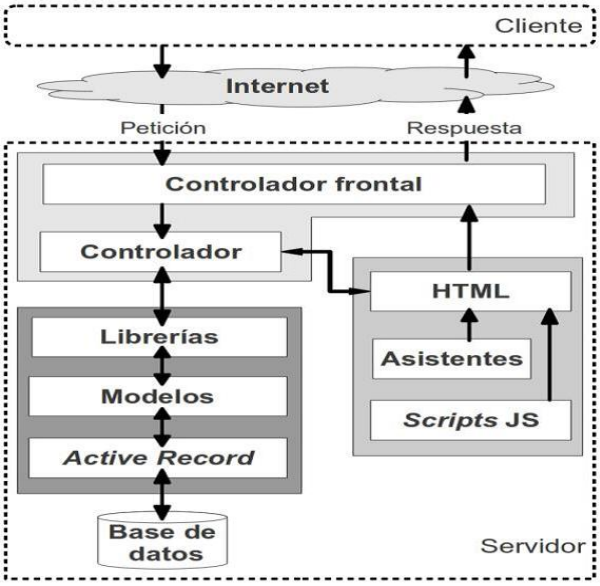


Figura 4. Patrón MVC implementado por GUUD

Entre las ventajas que presenta este modelo se encuentran: mayor escalabilidad, más claridad de diseño, facilita el mantenimiento, diseño modular y poco acoplado favoreciendo la reutilización, y mayor cohesión,

ya que cada elemento del patrón está altamente especializado en su tarea, la vista en mostrar datos al usuario, el controlador en las entradas, las librerías en su objetivo del negocio y los modelos en el acceso a los datos. Además, las modificaciones a las vistas no afectan los otros módulos de la aplicación. Si se desea hacer modificaciones en el modelo como agregar datos o métodos, solo debe modificarse el modelo y las interfaces del mismo, sin afectar la aplicación en su totalidad. Otra de las ventajas son: las vistas proveen mayor flexibilidad y agilidad pues se pueden crear múltiples vistas de un modelo, las vistas pueden anidarse, sincronizarse (García 2012).

2.7 Patrones de diseño

Un patrón de diseño es aquel que expresa un esquema para definir estructuras de diseño con las que construir sistemas de *software*, resulta ser una solución a un problema de diseño. Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces. Los patrones de diseño se caracterizan por: proporcionar catálogos de elementos reusables en el diseño de sistemas *software*, evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente, estandarizar el modo en que se realiza el diseño, facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando el conocimiento ya existente y no imponen alternativas de diseño frente a otras (García 2012).

- **Patrones para asignar responsabilidades**

Los Patrones Generales de *Software* para Asignar Responsabilidades (*GRASP, por sus siglas en inglés*) describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades, además dan la medida de un refinamiento del diseño.

GRASP contiene 5 patrones principales que son:

- ✓ Experto

Permite asignar una responsabilidad al experto en la información, en este caso sería a las clases que tienen la información necesaria para cumplir la responsabilidad. Con la utilización de este patrón se definió en qué clase colocar las funcionalidades de acuerdo con la información que estas necesitan. Se evidencia en las clases librerías, por ejemplo la librería filtros_lib que cuenta con la información necesaria para cumplir las responsabilidades sobre las categorías de filtros.

- ✓ Creador

Permite asignar a las clases la responsabilidad de crear instancias de otras clases. Este patrón se utilizó para identificar qué clase A debe crear elementos de una clase B, apoyándose en que la clase A debería: contener, agregar, registrar, utilizar y tener los datos de inicialización de la clase B. Se evidencia en la

clase *loader* que es el objeto *load* de las clases controladoras, encargada de cargar los elementos del marco de trabajo dígase librerías, ayudantes y modelos con los que necesita interactuar la controladora, por ejemplo, en la clase reglas_negocio solo son cargadas las librerías reglas_negocio_lib, filtros_lib.y el ayudante grid.

- ✓ Controlador

Permite asignar la responsabilidad de recibir o manejar un mensaje de los eventos del sistema a una clase. Las clases controladoras se encargan de obtener datos, y enviarlos a las librerías y las vistas. El patrón se evidencia en la aplicación de la siguiente manera: cuando llega una tarea a resolver al controlador este delega la responsabilidad a otras clases que son las que tienen implementado la lógica de negocio, por ejemplo, las librerías y los modelos que devuelven los datos al controlador y este los envía a las vistas que maneja.

- ✓ Bajo acoplamiento y Alta cohesión

Permiten asignar una responsabilidad de modo que se mantenga el bajo acoplamiento y la cohesión siga siendo alta. El grado de acoplamiento no puede considerarse aislado de otros principios como son Experto y Alta Cohesión. Los patrones se evidencian en la propia implementación de *CodeIgniter* que los contiene nivelados, pues permite el uso de los componentes de forma individual evidenciando el bajo acoplamiento, así como la dependencia entre ellos o alta cohesión.

- ➢ **Patrones GOF**

Los patrones GoF (*Gang of Four*), describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación clases y la formación de estructuras de mayor complejidad. Nos permiten crear grupos de objetos para ayudarnos a realizar tareas complejas. Se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

Los **patrones creacionales** tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es el de abstraer el proceso de creación de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

- ✓ Fábrica abstracta *(Abstract Factory)*

Permite trabajar con objetos de distintas familias de manera que estas no se mezclen entre sí, haciendo transparente el tipo de familia concreta que se esté usando. Se evidencia en el módulo Seguridad, en la librería fabrica_ma_lib, que se encarga de crear los objetos de los modos de autenticación por los que se puede acceder al Sistema de Gestión Universitaria.

- ✓ Instancia única *(Singleton)*

Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Se ve evidenciada en la aplicación debido a que todas las clases controladoras son instancias únicas, además de la clase *IOC (Inversion of Control)* que permite la integración entre los módulos.

Los **patrones de comportamiento** estudian las relaciones entre los diferentes objetos, normalmente ligados con la dimensión temporal. Ayudan a definir la comunicación e iteración entre los objetos de un sistema.

- ✓ Mediador *(Mediator)*

Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto. Se evidencia en las librerías que funcionan como mediadoras entre las clases controladoras y las modelos o acceso a datos, por ejemplo, la librería grupo_permisos_lib que media la interacción entre la controladora grupo_permisos y los modelos tb_grupo_permisos_mdl y tb_permiso_mdl.

2.8 Patrones de bases de datos

El diseño y construcción de una base de datos requiere del mayor esfuerzo y análisis posible ya que a partir de este diseño es que se crean las bases de datos. En la actualidad las bases de datos suelen ser muy grandes y el uso de patrones de diseño hace que el trabajo sea más fácil, además asegura un resultado correcto. Un patrón de base de datos es una plantilla que ya ha sido evaluada como la responsable de resolver un problema, es una guía de apoyo en la realización de un trabajo. Los patrones de diseño de bases de datos le permiten al usuario crear una base de datos más fortalecida a partir de una guía (EcuRed 2013).

- ➢ **Modelo entidad-atributo-valor**

El modelo entidad-atributo-valor es la representación de un modelo flexible donde se pueden representar los objetos con sus atributos, es un acercamiento al modelo orientado a objeto representado en el modelo relacional, donde la entidad *Class* representa las clases, la entidad *Attribute* representa los atributos de las clases, la entidad *Object* representa las instancias de las clases y la entidad *Value* representa los valores de cada atributo para cada objeto dado (EcuRed 2013). Un ejemplo de la aplicación de este patrón se evidencia en las tablas tb_dpermiso->tb_dpermiso_grupo_permiso->tb_dgrupo_permiso.

- ➢ **Llaves subrogadas**

Este patrón es muy utilizado pues se decide generar una llave primaria única que se le asigna a cada registro de una tabla. Normalmente se usan enteros en columnas *identity* o GUID *(Global Unique*

Identifier), está demostrado que no se repiten o solo lo hacen con una probabilidad extremadamente baja. Permite que las tablas sean más fáciles de consultar por el identificador (EcuRed 2013). Mediante este patrón fueron generados los identificadores de todas las tablas pertenecientes al modelo de datos.

2.9 Estándar de diseño

Para realizar el diseño de los prototipos se tuvo como guía las pautas del diseño visual establecidas en el Manual de Directrices del Sistema de Gestión Universitaria, que permite la uniformidad en todas las páginas *web* que lo componen.

Distribución de la información de la información en la solución informática

La vista de presentación es la primera vista que se le muestra a cualquier usuario, mediante la que podrá autenticarse y acceder al sistema. La siguiente figura muestra un ejemplo de la vista de presentación.

Figura 5. Áreas de la vista de presentación

1. Cabezal o área de identificación.
2. Área de datos de entrada.

La vista de escritorio se muestra luego de la autenticación del usuario, en la que podrá seleccionar el subproceso horizontal, módulo o servicio al que desee acceder y tenga los permisos requeridos. La siguiente figura muestra un ejemplo de la vista de escritorio.

Figura 6. Áreas de la vista de escritorio

1. Área de subprocesos horizontales.
2. Área del nombre de la aplicación.
3. Área de nombre de usuario.
4. Área de línea de procesos.
5. Área de servicios horizontales.
6. Área de pie de página.

La vista de gestión de procesos permite el acceso a los módulos del subsistema seleccionado por el usuario y las funcionalidades que poseen. La siguiente figura muestra un ejemplo de la vista de gestión de procesos mostrada por el sistema.

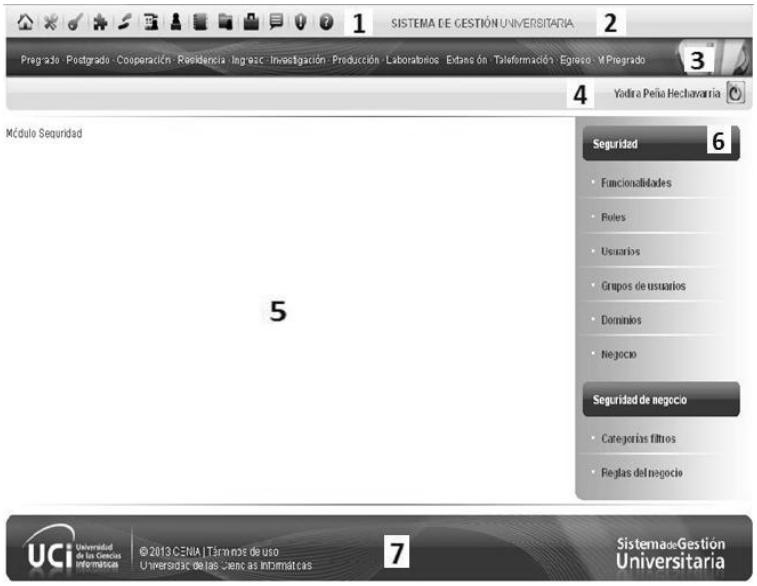


Figura 7. Áreas de la vista de gestión de procesos

1. Área de subprocesos horizontales.
2. Área del nombre de la aplicación.
3. Área de línea de procesos.
4. Área de nombre de usuario.
5. Área de contexto.
6. Área de menú de módulos.
7. Área de pie de página.

Tipos de mensajes

- **Mensajes de información**

Los mensajes de se utilizan para brindar información al usuario cuando se crea, se modifica, se elimina o se asocia un elemento. A continuación se enuncian algunos de los mensajes de información mostrados por el sistema y la Figura 8 muestra un ejemplo de los mismos.

- ✓ El elemento ha sido modificado satisfactoriamente.
- ✓ El elemento ha sido eliminado satisfactoriamente.

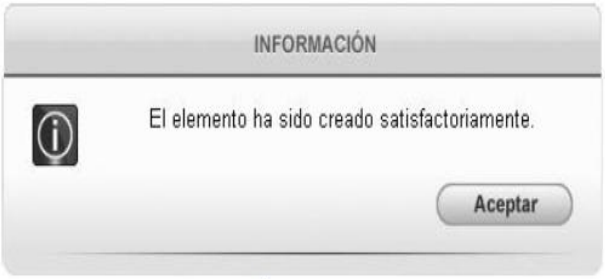


Figura 8. Mensaje de información

- **Mensajes de error**

Los mensajes de error se utilizan para mostrar al usuario cuando ha realizado una acción incorrecta. A continuación se enuncian algunos de los mensajes de error mostrados por el sistema y la Figura 9 muestra un ejemplo de los mismos.

- ✓ El elemento ya existe.
- ✓ Ha dejado campos obligatorios vacíos.

Figura 9. Mensaje de error

- **Mensaje de advertencia**

El mensaje de advertencia se utiliza cuando es necesario advertir al usuario de algún suceso si ejecuta la acción que está solicitando. A continuación se enuncia el mensaje de información mostrado por el sistema y la Figura 10 muestra un ejemplo del mismo.

- ✓ Perderá la información que no ha sido guardada.

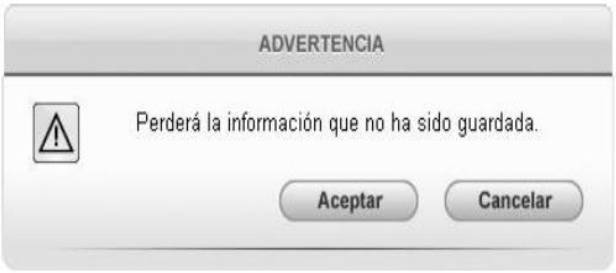


Figura 10. Mensaje de advertencia

- **Mensaje de confirmación**

El mensaje de confirmación se utiliza cuando es necesario asegurarse que el usuario desea realizar una acción determinada, por ejemplo cuando va a eliminar un elemento, es necesario asegurarse que eso es lo que desea. A continuación se enuncia el mensaje de confirmación mostrado por el sistema y la Figura 11 muestra un ejemplo.

- ✓ ¿Está seguro de realizar esta acción?

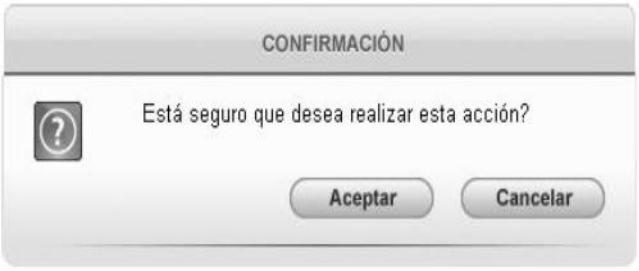


Figura 11. Mensaje de confirmación

2.10 Modelo de despliegue

El modelo de despliegue es utilizado para modelar el *hardware* empleado en las implementaciones de sistemas y las relaciones entre sus componentes. Es un modelo de objeto que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de *hardware* y otra información

relacionada al despliegue de la solución propuesta (Sparx Systems 2009). En la Figura 12 se muestra como quedaron distribuidos físicamente los componentes del sistema.

El modelo de despliegue elaborado está compuesto de la siguiente forma:

- PC Cliente: Estación de trabajo donde interactúa el cliente con la aplicación. Esta PC se conecta al servidor *web* a través del protocolo seguro *HTTPS*.
- Servidor *Web*: Contiene la aplicación *web*. Se conecta al servidor de base de datos mediante el protocolo *TCP/IP*.
- Servidor de base de datos: Servidor donde se almacenan los datos de la aplicación.

Figura 12. Modelo de Despliegue

2.11 Diseño de la de la base de datos

El diseño de bases de datos se utiliza para definir y especificar la estructura de los objetos de negocio que se emplean en el sistema cliente/servidor. Este proceso de diseño es guiado por algunos principios: el primero de ellos es que se debe evitar la información duplicada o datos redundantes, porque malgastan el espacio y aumentan la probabilidad de que se produzcan errores e incoherencias, el segundo principio es que es importante que la información sea correcta y completa; si la base de datos contiene información incorrecta, los informes que recogen información de la base de datos contendrán también información incorrecta y por tanto, las decisiones que se tomen a partir de esos informes estarán mal fundamentadas. Una base de datos correctamente diseñada permite obtener acceso a información exacta y actualizada (Office 2013).

2.11.1 Modelo de datos

El modelo de datos es el lenguaje mediante el cual queda descrita la base de datos especificando la estructura de los datos a almacenar (tipos, restricciones, operaciones de manipulación y relaciones entre estos). El modelo diseñado cuenta con un total de 9 tablas persistentes que almacenan toda la información del negocio. En la siguiente figura se muestra el modelo físico de la base de datos y el **Anexo 7** muestra el modelo lógico.

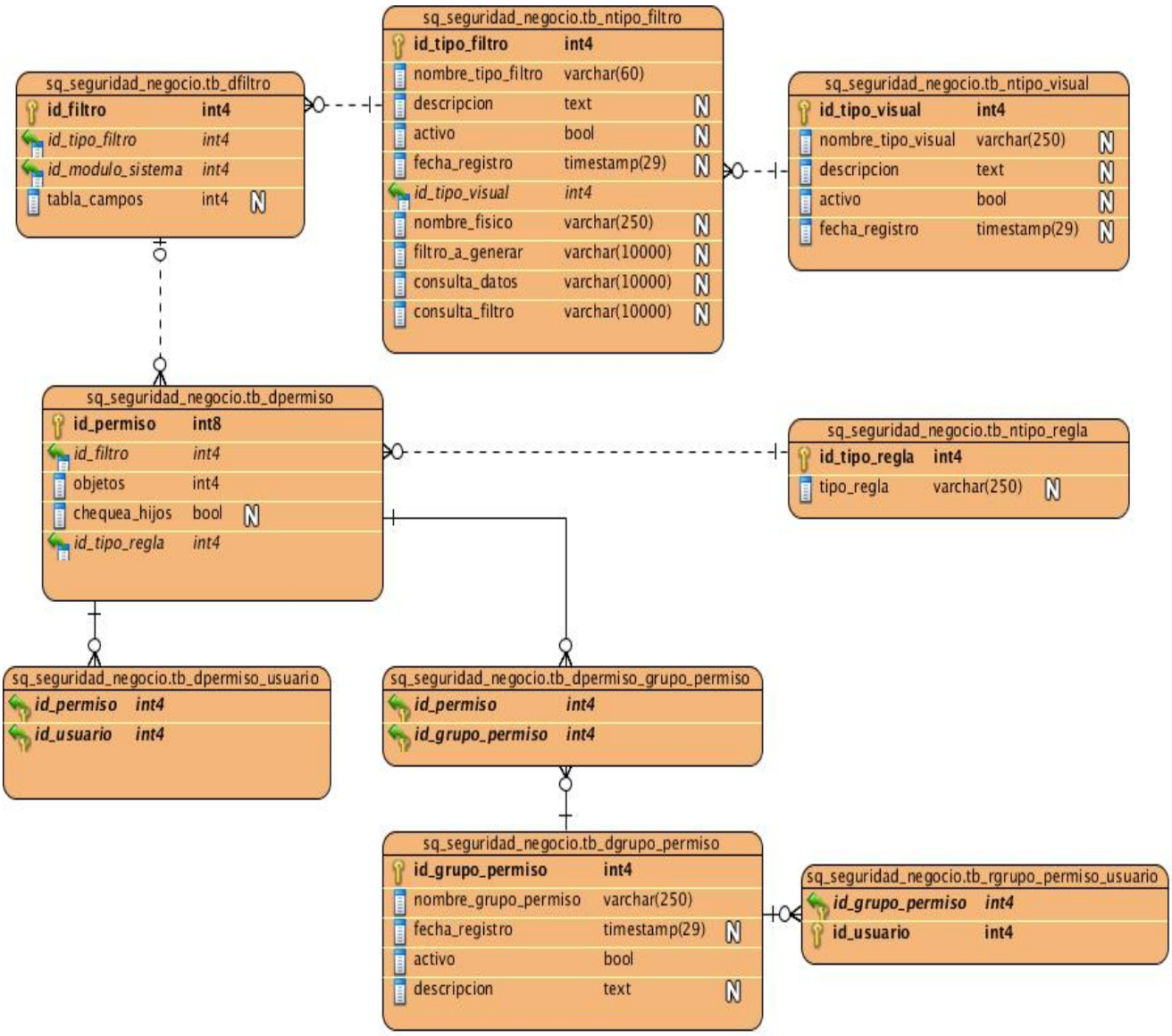


Figura 13. Modelo físico de la base de datos

2.12 Conclusiones parciales

Durante el desarrollo de este capítulo se logró la identificación, el refinamiento y la comprensión de los requisitos tanto funcionales como no funcionales, con los cuales debe cumplir la solución informática a desarrollar. Se realizó la selección de las herramientas a emplear y se obtuvo una propuesta de solución a nivel conceptual y con las primeras bases de desarrollo iniciada. Se analizaron los beneficios que traerá la integración de dicha solución al SGU. Además, se describió la arquitectura y el diseño propuesto para la

realización de la solución informática, teniendo como base los estándares establecidos en la Guía para el prototipado del *ERP*[22] Universitario. Con el objetivo de garantizar el almacenamiento de la información del sistema se realizó el diseño de la base de datos, a partir de una identificación de las clases persistentes y se generó el modelo de datos. Al concluir este capítulo han quedado creadas las condiciones necesarias para el desarrollo y validación de la solución informática.

[22] *ERP: Sistemas de planificación de recursos empresariales* (*Enterprise Resource Planning*, por sus siglas en inglés).

Capítulo 3. Implementación y pruebas

3.1 Introducción

La etapa de implementación del desarrollo de un *software* es el proceso de convertir una especificación del sistema en un sistema ejecutable. Siempre implica procesos de diseño y programación de *software*. En este capítulo se describe todo el proceso de implementación de la solución informática, además son descritas las pruebas y validaciones funcionales propuestas por el proceso de desarrollo utilizado en el Centro de Informatización Universitaria (CENIA).

3.2 Estándares de Codificación

Los estándares de codificación son un estilo de programación homogénea en un proyecto. Estos son de vital importancia durante la etapa de implementación, ya que permite que todo el personal del proyecto pueda entender de forma fácil el código, garantizando la organización y estructura del código fuente. Es decir, los estándares de codificación son un conjunto de reglas a seguir por los desarrolladores con el objetivo de establecer un orden y un formato común en el código fuente del *software* en desarrollo.

Para el desarrollo de la solución informática se utilizaron los estándares de codificación establecidos en el documento de Arquitectura de *Software* por CENIA, con el propósito de estandarizar las nomenclaturas en la implementación del sistema y obtener un producto estable y eficiente (García 2012).

- **Identación, llaves de aperturas y cierre, tamaño de líneas**

Se debe usar la identación sin tabulaciones, con un equivalente a 4 espacios, para mantener integridad en las revisiones *SVN*[23]. El uso de las llaves "{}" será en una nueva línea. Para mantener la legibilidad del código la longitud de las líneas de código es aproximadamente de 75-80 caracteres. La siguiente figura muestra un ejemplo del uso de la indentación y llaves.

```
public function index()
{
   echo $this->template->render('modulo_filtro/mostrar_view');
}
```

Figura 14. Identación y llaves

[23] *SVN: Subversion (SVN, por sus siglas en inglés) es un sistema de control de versiones.*

- **Conversión de nomenclatura**

Las variables se rigen por la nomenclatura *camelCase*[24]. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. En la Figura 15 se muestra un ejemplo del uso de variables.

```
$post = $this->input->all_post();
$permisosAsociados = $post['permisosAsociados'];
```

Figura 15. Variables

Los nombres de las clases siempre comienzan con mayúscula, en caso de nombre compuesto las palabras se separan con el carácter subrayado "_" y el resto en minúscula. La siguiente figura muestra un ejemplo de cómo deben nombrarse las clases.

```
class Grupo_permiso extends MY_Controller
{
    public function __construct()
    {
        parent::__construct();
        $this->load->library("modulo_filtro_lib");
        $this->load->library("filtros_lib");
    }
}
```

Figura 16. Clases

Las funciones se rigen por la nomenclatura *camelCase*. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa. La Figura 17 muestra un ejemplo del uso de esta nomenclatura.

```
public function asociarUsuario()
{
        $datos = $this->input->all_post(TRUE);
        $id_grupo_permiso = $datos['id_grupo_permiso'];
        $usuarios = $datos['usuarios'];
        echo json_encode($usuarios);
}
```

Figura 17. Funciones

[24] *CamelCase es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en CamelCase se asemejan a las jorobas de un camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula.*

Los ficheros siempre se escriben en minúsculas y en caso de nombres compuestos se usa el carácter subrayado"_".

Vistas: intuitivo y relacionado con el formulario y/o vista que representa.

Modelos: con el mismo nombre de la clase que representa que contiene en el nombre el sufijo_mdl o base en caso de ser modelos base.

Librerías: con el mismo nombre de la clase que representa y contiene en el nombre el sufijo_lib.

Controladoras: con el mismo nombre de la clase que representa.

Manager: con el mismo nombre de la clase que representa que contiene en el nombre el sufijo_mng.

- **Estructuras de control**

Se incluye *if, for, foreach, while, switch,* entre las estructuras de control y en los paréntesis debe de existir un espacio. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales. La siguiente figura muestra un ejemplo de esta organización.

```
public function registrarGrupoPermiso()
{
    if ($this->input->is_post_back('grupo_permiso'))
    {
        $grupo_permiso = $this->input->all_post(TRUE);
        if ($this->grupo_permiso_lib->registrarGrupoPermiso($grupo_permiso) != FALSE)
        {
           $this->message('SYS001');
        }
        else
            {
              throw new Exception_Error('SYS006');
            }
     }
     else
        {
           throw new Exception_Error('SYS007');
        }
}
```

Figura 18. Estructuras de control

Si las condiciones son muy largas y sobrepasan el tamaño de la línea, estas se dividen en varias líneas. Esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos. A continuación, en la Figura 19 se muestra un ejemplo de la separación en varias líneas de las condiciones.

```
private function detalles($id_grupo_permiso)
{
    $grupo_permiso = $this->grupo_permiso_lib->obtenerGrupoPermisoDadoIdGrupoPermiso($id_grupo_permiso);
    $usuarios = $this->grupo_permiso_lib->obtenerUsuarioDadoIdGrupoUsuarioDetalles($id_grupo_permiso);
    if($grupo_permiso !="" && $usuarios!="")
    {
       $this->template->set_data("grupo_permiso", $grupo_permiso);
       $this->template->set_data("usuarios", $usuarios);
    }
}
```

Figura 19. Condiciones en varias líneas

- **Documentación**

Todos los archivos cuentan con la documentación asociada al mismo. Para esto cumplen con un bloque de instrucciones al principio de cada clase, el cual se muestra en la siguiente figura:

```
/**
* Clase controladora del grupo de permisos
*
* Esta clase funciona como intermediaria entre la vista
* y las librerias
*
* @package Seguridad
* @subpackage  Controllers
* @category    Controllers
* @author  Lenier Garcia Vizcaino
*/
```

Figura 20. Documentación

- **Buenas prácticas**

Los valores booleanos y nulos siempre se escriben con mayúscula, para facilitar la legibilidad del código, se usa una línea en blanco antes de las estructuras de control y definición de las funciones. La siguiente figura muestra un ejemplo de la aplicación de las buenas prácticas.

```
$datos = $this->input->all_post(TRUE);
$id_grupo_permiso = $datos['id_grupo_permiso'];
```

Figura 21. Buenas prácticas

3.3 Técnicas de programación

Cuando se crea un producto el mismo debe tener la flexibilidad suficiente para ser modificable en el momento que se requiera. Estos deben ser claros, simples, con el fin de poder ser leídos e interpretados de forma fácil. Para lograr este objetivo se debe asumir en la programación técnicas que permitan la estandarización. Es en ese momento donde entran a desempeñar un papel muy importante las técnicas de programación, en las últimas décadas con el desarrollo de la Informática es muy común escuchar que los diversos lenguajes y programas que se crean están orientados a objetos y que soportan la programación modular (Pressman 2002). A continuación se explica la técnica de programación adoptada para la creación de la solución informática.

Programación orientada a objetos (POO)

En los últimos años la frase “orientado a objetos”, se ha vuelto muy popular, escuchándose a cada momento frases como: “sistemas operativos orientados a objetos”, “lenguajes orientado a objetos” y “programación orientado a objetos”. Dentro de los conceptos generales más utilizados en el modelo orientado a objeto se encuentran: la abstracción, la encapsulación y modularidad. Con respecto a la programación son: los objetos, las clases, los métodos, el envío y recepción de mensajes, la herencia y el polimorfismo. En la *POO* encapsular significa que se reúne y controla todo el grupo resultante en un conjunto y no de forma individual. La abstracción es un término externo al objeto, que controla la forma en que es visto por los demás. La herencia se define como una jerarquía de clases derivadas y la relación entre estas, donde se comparte la estructura y el comportamiento de una o más clases consideradas como clases padres. El polimorfismo constituye la definición de múltiples clases con funcionalidades diferentes, pero con métodos o propiedades denominados de forma idéntica (slideshare 2009).

3.4 Técnicas de validación de requisitos

Según *Sommerville*[25], la validación de requisitos examina las especificaciones para asegurar que todos los requisitos de *software* han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que

[25] *Sommerville: Profesor de ingeniería de Software de la Universidad de St Andrews en Escocia y el autor de un libro Ingeniería del Software. Profesor de Ciencias de la Computación de la Universidad Heriot-Watt en Edimburgo. Profesor de Ingeniería de Software en el Departamento de Informática de la Universidad de Lancaster. Destacado investigador en el campo de la ingeniería de sistemas, la fiabilidad del sistema y la informática sociales.*

errores detectados han sido corregidos, y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto. Para la presente propuesta de solución fueron puestas en prácticas las siguientes técnicas para la validación de los requisitos:

- ✓ Revisiones de requisitos: En esta técnica, los requisitos son analizados con el cliente para de este modo obtener los posibles errores que pueden existir en la especificación de los requisitos del *software* o validar la correcta interpretación de la información.
- ✓ Construcción de prototipos: En este enfoque de validación, se muestra un modelo del futuro sistema a los clientes, permite al usuario hacerse una idea de la estructura de la interfaz, además, el usuario tiene la posibilidad de corregir errores o añadir aspectos para su completitud.
- ✓ Generación de casos de pruebas: Los requisitos deben poder probarse. Si las pruebas para éstos se conciben como parte del proceso de validación, a menudo revelan los problemas en los requisitos. La realización de casos de pruebas posibilita la verificación del cumplimiento de los requisitos funcionales, que los mismos cumplan con la eficiencia requerida (Sommerville 2005).

3.4.1 Resultado de la aplicación de las técnicas de validación de los requisitos

Durante la revisión de requisitos se realizaron verificaciones en el documento “Especificaciones de requisitos”, este proceso comprendió las siguientes validaciones:

- ✓ Verificaciones de validez: los requisitos deben cumplir con las necesidades del cliente, luego de realizar algunos análisis y razonamientos surgieron cambios en las funciones que ya estaban identificadas.
- ✓ Verificaciones de consistencia: los requisitos no deben contradecirse en las especificaciones escritas, no debe haber restricciones o descripciones que estén opuestas a las reglas definidas.
- ✓ Verificaciones de completitud: los requisitos deben incluir todas las funcionalidades propuestas por el cliente y satisfacer de manera general todas las necesidades acordadas.
- ✓ Verificabilidad: para evitar posibles discusiones entre los miembros del equipo del proyecto y el cliente, se revisó que los requisitos estuvieran descritos de manera que se puedan diseñar casos de pruebas orientadas a estos y que demuestren que el sistema a entregar responde a las necesidades del cliente.

Al finalizar el proceso de revisión de los requisitos se detectaron algunas inconsistencias que fueron erradicadas de inmediato, entre ellas se encuentran:

- ✓ Descripción poco detallada de los requisitos.
- ✓ Interpretación incorrecta de algunas funcionalidades.

- ✓ Ausencia de opciones en el área de íconos flotantes.

3.5 Proceso de pruebas

Las pruebas son un elemento crítico para la garantía de calidad del *software* y representa una revisión final de las especificaciones, del diseño y de la codificación. Según *Pressman*[26] las pruebas de *software* pueden definirse como: *"el proceso de evaluación de un producto desde un punto de vista crítico, donde el probador somete al producto a una serie de acciones indagadoras, y el producto responde con su comportamiento como reacción".* Como ventaja secundaria, la prueba demuestra hasta qué punto las funciones del *software* parecen funcionar de acuerdo con las especificaciones y parecen alcanzarse los requisitos de rendimiento. Los objetivos de realizar una prueba son: descubrir un error, tener un buen caso de prueba y detectar un error no descubierto antes (éxito de la prueba).

Para los efectos de la investigación el proceso de pruebas es llevado a cabo en varios niveles de pruebas, definiendo así, los métodos y técnicas usadas. Se propone entonces realizar las pruebas en los siguientes niveles:

- ➢ Pruebas de unidad.
- ➢ Pruebas de integración.
- ➢ Pruebas del sistema.

3.5.1 Características de una buena prueba

Una buena prueba no debería ser ni demasiado sencilla ni demasiado compleja, pero si se quieren combinar varias pruebas se pueden enmascarar errores, por lo general, cada prueba debería realizarse separadamente. A continuación se presentan algunas características de una buena prueba:

- ✓ Una buena prueba ha de tener una alta probabilidad de encontrar un fallo. Para alcanzar este objetivo el responsable de la prueba debe entender el *software* e intentar desarrollar una imagen mental de cómo podría fallar.
- ✓ Una buena prueba debe centrarse en dos objetivos: probar si el *software* no hace lo que debe hacer y probar si el *software* hace lo que no debe hacer.
- ✓ Una buena prueba no debe ser redundante. El tiempo y los recursos son limitados, así que todas las pruebas deberían tener un propósito diferente.

[26] *Pressman: ingeniero de software estadounidense, autor y consultor. Presidente de la compañía Pressman & Associate. Reconocido autor de obras que se utilizan en la Ingeniería de Software y en la mejora de procesos.*

- ✓ Una buena prueba debería ser la “mejor de la cosecha”. Se debería emplear la prueba que tenga la más alta probabilidad de descubrir una clase entera de errores (Pressman 2002).

3.5.2 Métodos de prueba

Los métodos de prueba proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas. Los mismos se agrupan en: métodos de caja blanca o estructural y método de caja negra o funcional. En el proceso de pruebas de la solución informática propuesta se emplearán ambos métodos de prueba. La siguiente imagen muestra gráficamente la filosofía de las pruebas de caja blanca y caja negra. Se puede observar que las pruebas de caja blanca necesitan conocer los detalles procedimentales del código, en cambio las de caja negra únicamente necesitan saber el objetivo o funcionalidad que el código ha de proporcionar.

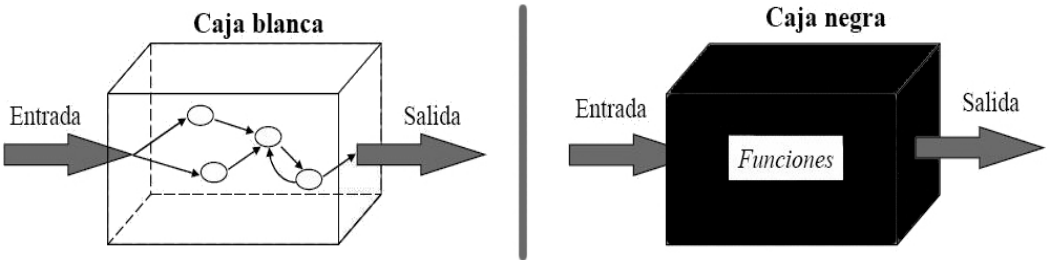


Figura 22. Filosofía de las pruebas de caja negra y caja blanca

Método de caja blanca o estructural

A este método de prueba se le conoce también como prueba de caja transparente o de cristal y se basan en un minucioso exámen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa. Estas pruebas se realizan sobre las funciones internas de un módulo en concreto. Su objetivo es diseñar casos de prueba para que se ejecuten, al menos una vez todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa.

Existen ciertas técnicas para el diseño de este tipo de pruebas que permiten decidir qué sentencias o caminos se deben examinar con los casos de prueba. Entre las más usadas se encuentran las siguientes:

- ✓ Prueba de condición.
- ✓ Prueba de flujo de datos.
- ✓ Prueba de bucles.
- ✓ Prueba del camino básico.

Método de caja negra o funcional

El método de caja negra o funcional consiste en realizar pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. Estas pruebas examinan algunas

algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del *software*, por lo que para su aplicación no es necesario conocer la lógica del programa sino únicamente la funcionalidad que debe realizar. Su objetivo es revelar el incorrecto o incompleto funcionamiento del sistema, así como los errores tanto de interfaz como de inicialización y terminación.
Las pruebas de caja negra permiten encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores de estructuras de datos o en accesos a las bases de datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

3.5.3 Pruebas unitarias

Según *Pressman* las pruebas de unidad se centran en la verificación de los elementos más pequeños del *software* que se puedan probar. Antes de realizar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del componente. Si los datos no entran correctamente, todas las demás pruebas no tienen sentido. Para la realización de esta prueba se aplica el método de caja blanca con el objetivo de verificar el resultado real de la prueba para cada uno de los caminos, se empleó un mecanismo que posee *CodeIgniter* (parte estructural del marco de trabajo GUUD) para la automatización de pruebas unitarias.

- **Pruebas unitarias en CodeIgniter**

CodeIgniter posee una librería bastante sencilla especializada en la ejecución de pruebas estructurales. Cuenta con una sola función de evaluación y dos funciones de resultados, permite determinar con certeza si un código específico produce el tipo de dato y resultado esperado. Para correr una prueba utilizando dicha librería es necesario suministrar el código a probar y un resultado esperado de la siguiente forma:

$this->unit->run (código, resultado, 'nombreprueba');

Donde “código”, es el segmento de código que se desea probar, “resultado”, es lo que se espera que devuelva la evaluación del código y “nombreprueba” es un nombre opcional que se le puede dar a la prueba. Los tipos de datos posibles son: *“is_string”,* “*is_bool*”,“*is_true*”, “*is_false*”, “*is_int*”, “*is_numeric*”, “*is_float*”, “*is_double*”, “*is_array*”, “*is_null*”.
El **Anexo 8** muestra de forma detallada los casos de pruebas realizados al sistema, con los que se obtuvieron resultados satisfactorios.

- **Técnica utilizada**

La prueba del camino básico es una técnica de prueba de caja blanca propuesta inicialmente por *Tom McCabe*[27]. El método permite obtener una medida de la complejidad del diseño procedimental de un programa (o de la lógica del programa), además representa un límite inferior para el número de casos de pruebas que se deben realizar para asegurar que se ejecuta por lo menos una vez cada camino del programa. A continuación se hace mención de los pasos a realizar para aplicar la prueba del camino básico y en el **Anexo 9** se muestra una descripción detallada de cada uno de los pasos.

Pasos a realizar para aplicar la técnica del camino básico:

- ✓ Paso 1: Representar el programa en un grafo de flujo.
- ✓ Paso 2: Calcular la complejidad ciclomática.
- ✓ Paso 3: Determinar el conjunto básico de caminos independientes.
- ✓ Paso 4: Derivar los casos de prueba que fuerzan la ejecución de cada camino.

3.5.4 Prueba de integración

Según *Pressman* la prueba de integración es una técnica sistemática para construir la estructura de un 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 está de acuerdo con lo que dicta el diseño. Como la solución informática "Seguridad de Negocio" se integra al módulo Seguridad del Núcleo y a los módulos del Subsistema de Gestión Académica de Pregrado del Sistema de Gestión Universitaria, es necesario probar la integración entre las agrupaciones funcionales y además, su integración con dichos módulos. Para la realización de esta prueba se decide utilizar el método de caja negra, aplicando como técnica la integración incremental.

- **Técnica utilizada**

Existen dos tipos de integración: no incremental e incremental. En el primer caso se combinan todos los módulos y se prueba el programa en su conjunto, como es lógico pensar el resultado puede ser caótico con un gran número de fallos y la consiguiente dificultad para identificar el módulo que los provocó. Por su parte, la integración incremental es la antítesis del enfoque no incremental. El programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir. Por esta razón se escogió el enfoque incremental para la realización de las pruebas de integración de la solución informática.

[27] *Tom McCabe: Licenciado en matemáticas de la Universidad de Connecticut. Miembro de la Asociación Americana de Matemáticas. Enunció la técnica del camino mínimo en 1976.*

Al finalizar las pruebas de integración no fueron detectados errores asociados a la interacción de la solución informática con el módulo Seguridad y los módulos del Subsistema de Gestión Académica de Pregrado del Sistema de Gestión Universitaria. La siguiente tabla muestra un ejemplo de los diseños de casos de pruebas de integración y el **Anexo 10** se muestra todos los diseños de casos de prueba realizados.

Tabla 5. Casos de prueba de integración con el módulo Seguridad

Caso de Prueba: Int1_S
Módulo al que se integra: Seguridad
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas a este módulo. Además, debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Seguridad.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Seguridad.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

3.5.5 Prueba del Sistema

Las pruebas del sistema se hacen cuando el *software* está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes han sido integrados. Según *Pressman* este tipo de prueba está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema para verificar que se han integrado adecuadamente todos los elementos del mismo y que realizan las funciones adecuadas.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien. Para la realización de las pruebas se generaron los artefactos “Diseño de casos de pruebas basado en requisitos”. Por cada requisito funcional del sistema se generó un

documento en donde se recogen todos los datos necesarios para probar la interfaz. En el **Anexo 11** se muestran los diseños de casos de pruebas realizados.

- **Pruebas funcionales**

Las pruebas funcionales son pruebas basadas en el análisis de la especificación funcional de un componente o de un sistema, tienen como objetivo probar que los sistemas desarrollados cumplen con las funciones específicas para los que han sido creados. La función de un sistema es “lo que hace” dicho sistema, y normalmente es descrita en la especificación de requisitos, una especificación funcional o en casos de uso. Para la realización de esta prueba se decide utilizar el método de prueba de caja negra, aplicando la técnica partición de equivalencia.

- **Técnica utilizada**

La partición de equivalencia es una técnica de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de casos de prueba a desarrollar. Esta técnica intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia. De tal forma que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente a una prueba realizada con cualquier otro valor de dicha clase. Esto quiere decir que si el caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error. El diseño de casos de prueba según esta técnica consta básicamente de dos pasos:

- ✓ Paso 1: Identificar las clases de equivalencia.
- ✓ Paso 2: Identificar los casos de prueba.

En el **Anexo 12**, muestra una descripción detallada de cada uno de los pasos para aplicar esta técnica.

- **Resultados de las pruebas funcionales**

Con la realización de las pruebas funcionales se detectaron un conjunto de no conformidades las cuales fueron erradicadas inmediatamente. La Figura 23 muestra gráficamente los resultados obtenidos por iteraciones durante la aplicación de las pruebas funcionales.

Finalizadas las pruebas se alcanzaron los siguientes resultados por iteraciones:

- En la primera iteración se obtuvieron veinticinco no conformidades. Entre las que se encuentran: la no indicación de la acción de los íconos internos, la no visualización de algunos mensajes de

advertencia y la no ejecución de la acción del ícono “Listar”. Estas no conformidades fueron corregidas.

- En la segunda iteración se detectaron 8 no conformidades. Dentro de las que se encuentran: la no validación de algunos campos, errores ortográficos y el ícono de “Listar” mostraba el mensaje de asociar. Estas no conformidades también fueron erradicadas.
- En la tercera iteración no fueron detectadas no conformidades, por tanto se cumplieron con todos los requisitos funcionales expuestos.

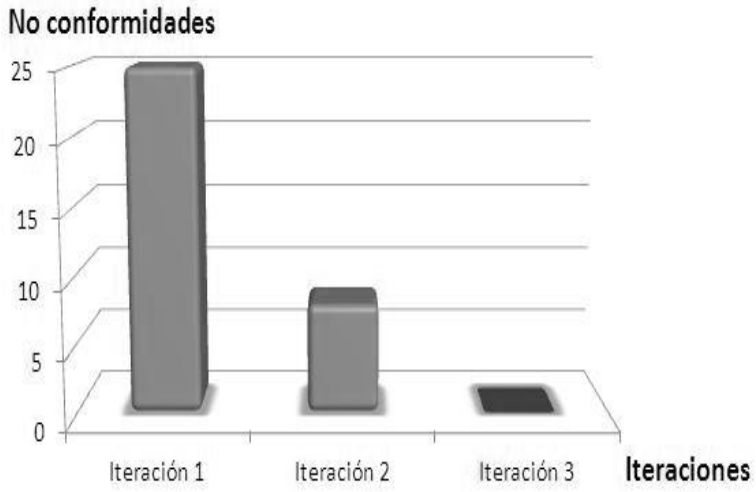


Figura 23. Resultado de las pruebas funcionales

3.6 Conclusiones parciales

En este capítulo fueron abordados temas referentes a los estándares de codificación, técnicas de programación y tratamiento de errores, que sirvieron como guía para el desarrollo de la solución informática. El adecuado diseño de los casos de prueba brindó la posibilidad de probar las funcionalidades implementadas. En el capítulo quedaron propuestas un conjunto de pruebas definidas para lograr el correcto funcionamiento de la aplicación y la calidad de la misma de acuerdo a las especificaciones del cliente. Esto permitió arribar a la conclusión de que la detección de inconsistencias es un paso fundamental para obtener una correcta implementación. Durante todo el desarrollo del capítulo se obtuvieron resultados muy favorables y con la culminación del mismo, se puede afirmar que la solución informática implementada cumple con todas las restricciones y requisitos necesarios para ser utilizado en el Subsistema de Gestión Académica de Pregrado y no solo en este subsistema, sino que al ser extensible da la posibilidad ser aplicado a todo el Sistema de Gestión Universitaria.

Conclusiones generales

Una vez finalizada la implementación de la solución informática, se pudo comprobar el cumplimiento de los objetivos trazados, obteniéndose así resultados satisfactorios. Teniendo en cuenta lo planteado se puede arribar a las siguientes conclusiones:

- ✓ Se adquirió el conocimiento necesario acerca de los procesos relacionados con la seguridad de negocio dentro de los sistemas informáticos.
- ✓ Se realizó el análisis de soluciones existentes, logrando así dominar un conjunto de conocimientos sobre sus principales características, constituyendo la base de la propuesta de solución.
- ✓ El estudio de las herramientas, lenguajes de programación y proceso de desarrollo a utilizar permitió la familiarización con los elementos del entorno de desarrollo y la obtención de conocimientos necesarios sobre sus características generales.
- ✓ El proceso de desarrollo de *software* utilizado guió satisfactoriamente la elaboración de la solución, garantizando la obtención de los artefactos definidos.
- ✓ La validación de los requisitos y las pruebas realizadas, comprobaron el correcto funcionamiento de la solución.

Al aplicar las pautas establecidas por el proceso de desarrollo utilizado, se logró la documentación de la investigación y la obtención los artefactos necesarios, quedando registrada en el Expediente del Proyecto de Pregrado.

Recomendaciones

Luego de haber cumplido con los objetivos propuestos mediante la realización del trabajo de diploma, obteniendo como resultado del mismo una solución informática que permita gestionar la seguridad en el Sistema de Gestión Académica de Pregrado se recomienda:

- Extender el uso de la solución propuesta a los demás subsistemas del SGU.
- Crear funcionalidades que permitan acotar los datos de las categorías de filtros para un mejor aprovechamiento de la solución.
- Liberar la solución por parte de la dirección de Calidad.

Bibliografías referenciadas

Álvarez, Aguayo, Daniel Alejandro. (2013). "¿Qué es CSS?". [En línea: 2013]. [Citado el: 9 de enero del 2013]. [Disponible en: http://nanysman.wordpress.com/category/4to/].

Aspajo, Quirós, Valeria. (2010). "Técnicas para definir requerimientos". [En línea: 2010]. [Citado el: 10 de febrero del 2013]. [Disponible en: http://www.slideshare.net/vaspajoq/tcnicas-para-definir-requerimientos-5281759].

Benjamín, Galacho, Federico (2009). Manual de Sistemas de Información.

Carbajal, Pimentel, Gino (2012). "Base De Datos: Arquitectura Cliente/Servidor". [En línea:2012]. [Citado el: 12 de febrero del 2013]. [Disponible en: http://www.inei.gob.pe/web/metodologias/attach/lib616/cap0302.HTM].

Castro, Jaqueline. (2012). "Sistemas de Información". [En línea: 2012]. [Citado el: 20 de enero del 2012] [Disponible en: http://jacquelinecastro.wordpress.com/category/sistemas-de-informacion/].

EcuRed. (2013). "Lenguajes de Programación". [En línea: 2013]. [Citado el: 16 de mayo del 2013]. [Disponible en:http://www.ecured.cu/index.php/Lenguaje_de_programaci%C3%B3n].

Figueras, García, Thais (2012). “Solución informática para la Compartimentación de la información en los sistemas que utilizan el Sistema de Gestión Integral de Seguridad ACAXIA”. [En línea: 2012]. [Citado el: 27 de febrero del 2013].

GALEANO, Gill. (2009). "HTML: manual imprescindible", ISBN: 9788441525030. [En línea: 2009] [Citado el: 16 de abril del 2013]. [Disponible en: http://books.google.com.cu/books?idWuJPgAACAAJ&dq=HTML&hl=es&sa=X&ei=8LqzT7u8LojBtgf4kIWeAw&ved=0CFkQ6AEwBzgK].

García, Vidal, Yanio. (2012). Documento de Arquitectura de Gestión Universitaria. La Habana , 2012.

Hurtado, Sola, Elianys (2008). "Sistema de Gestión de Sesiones". [En línea: 2008]. [Citado el: 18 de febrero del 2013]. [Disponible en: http://repositorio_institucional.uci.cu/jspui/simple-search?query=%EF%83%98%09Sistema+de+Gesti%C3%B3n+de+Sesiones&submit=Buscar+].

Informático, (2012). "Net Beans". [En línea: 2012]. [Citado el: 18 de enero del 2013]. [Disponible en: http://turinconinformaticoduocuc.blogspot.com/p/programas.html].

ISE (2011). Software Engineering Institute. What is CMMI?. [En línea: 2011]. [Citada el: 4 marzo del 2013]. [Disponible en: http://www.sei.cmu.edu/cmmi/index.cfm].

ITESCAM. (2012). "Herramientas CASE". [En línea: 2012]. [Citado el 10 de febrero del 2013]. [Dispobible en:

ones+inform%C3%A1ticas+destinadas+a+aumentar+la+productividad+en+el+desarrollo+de+software+reduciendo+el+costo+de+las+mismas+en+t%C3%A9rminos+de+tiempo+y+dinero&source=web&cd=2&ved=0CDcQFjAB&url=http%3A%2F%2Fwww.itescam.edu.mx%2Fprincipal%2Fsylabus%2Ffpdb%2Frecursos%2Fr49237.DOCX&ei=87SZUfR8sP3gA--kgLgP&usg=AFQjCNHnJltMfFwgxGzWCt_eipFjrsmygw&bvm=bv.46751780,d.dmg].

Javascript, Sourse. (2012). "JavaScript". [En línea: 2012]. [Citado el: 19 de enero del 2013]. [Disponible en: http://www.javascriptsource.com/].

Latham, Donald C. (1985). "DEPARTMENT OF DEFENSE STANDARD". Library No. S2257ll, 1126 p, CSC-STD-00l-83. [En línea: 1985] [Citado el: 2 de frebrero del 2012]. [Disponible en:http://csrc.nist.gov/publications/history/dod85.pdf].

Lengua Española. (2010). "Diccionario Manual de la Lengua Española". Vox. © 2010 Larousse Editorial. [En línea: 2010]. [Citado el: 20 de mayo del 2013]. [Disponible en: http://es.thefreedictionary.com/negocio].

Llano, Castro, Eileén (2012). "PROPUESTA PARA LA INTEGRACIÓN DE PRÁCTICAS DE LAS METODOLOGÍAS XP Y SCRUM CON EL PROCESO DE ADMINISTRACIÓN DE REQUISITOS DEL NIVEL 2 DE CMMI". 2012, 10p.

Marqués, María Mercedes. (2001). "Apuntes de Ficheros y Bases de Datos". [Citado el: 6 de enero del 2013]. [Disponible en: http://repositori.uji.es/xmlui/bitstream/handle/10234/7314/Apuntes%20de%20Ficheros%20y%20Bases%20de%20Datos.htm?sequence=1].

Melián, Montalvo, Marlene. (2012). "XML el nuevo lenguaje universal". [En línea: 2012]. [Citado el: 14 de enero del 2013.]. [Disponible en:http://www.bibliociencias.cu/gsdl/collect/eventos/archives/HASH0104/f016d031.dir/doc.pdf].

Naranjo, García, Adrián. (2010). Sistema de Autenticación y Control de Acceso para aplicaciones del Departamento de Soluciones para la Aduana. [En línea: 2010]. [Citado el: 10 de febrero del 2013]. [Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_03315_10].

Office. (2013). "Conceptos básicos del diseño de una base de datos". [En línea: 2013]. [Citado el: 10 de marzo del 2013]. [Disponible en: http://office.microsoft.com/es-es/access-help/conceptos-basicos-del-diseno-de-una-base-de-datos-HA001224247.aspx].

Olson, Jhon. (2010). "¿Que es el PHP?". [En línea: 2010]. [Citado el: 18 de enero del 2013.]. [Disponible en: http://www.taringa.net/comunidades/softwaresysistemas/791421/Que-es-el-PHP.html].

Ortega, Ruiz, Idelvis (2006). " Sistema de Autenticación de Aplicaciones de la Intranet". [En línea: 2006] [Citado el: 10 de febrero del 2013]. [Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_0154_06].

Pressman, Roger. (2002). Ingeniería de Software: Un enfoque práctico. Quinta Edición. Editorial McGraw-Hill, 2002. 640 p. ISBN: 8448132149

PROJECT, PENCIL. (2010). "Sketching and Prototyping with Firefox". [En línea: 2010]. [Citada el: 2 de febrero de 2013]. [Disponible en: http://pencil.evolus.vn/en-US/Home.aspx]

Rendón, David. (2012). "¿Qué es jQuery?". [En línea: 2012]. [Citado el: 9 de enero del 2013]. [Disponible en: http://daverndn.com/2012/10/09/scroll-horizontal-jquery-webmatrix/]

SIG, Implementación. (2013). "Que es un sistema de gestión". [En línea: 2013]. [Citado el: 30 de mayo del 2013]. [Disponible en: http://www.implementacionsig.com/index.php/23-noticiac/28-que-es-un-sistema-de-gestion].

slideshare. (2009). "Programación y Algoritmos". [En línea: 2009]. [Citado el: 5 Marzo de 2013]. [Disponible en: http://sunshine.prod.uci.cu/search/programaci%C3%B3n%20orientada%20a%20objetos].

slideshare. (2010). "SEGURIDAD EN LOS SISTEMAS DE INFORMACION.". [En línea: 2010]. [Citado el: 4 de enero del 2013]. [Disponible en: http://www.slideshare.net/nyzapersa/curso-seguridad-en-sistemas-de-informacion].

Sommerville, Ian. (2005). Ingeniería del software. 7ma Edición. Pearson Educación, 2005. ISBN: 8478290745.

Sparx Systems, Pty Ltd. (2009). "El Modelo Físico" [En línea: 2009]. [Citado el: 10 de febrero del 2013]. [Disponible en: http://www.sparxsystems.com.ar/resources/tutorial/physical_models.html].

Ubuntu, Guía. (2008). "PgAdmin" [En línea: 2008]. [Citado el: 16 de febrero del 2013]. [Disponible en: http://www.guia-ubuntu.com/index.php?title=PgAdmin_III].

Ubuntu, Guía. (2011). "PostgreSQL". [En línea: 2011]. [Citado el: 1 de enero del 2012]. [Disponible en: http://www.guia-ubuntu.com/index.php?title=PostgreSQL].

Viera, Víctor. (2010). "Frameworks y arquitecturas de software". [En línea: 2010]. [Citado el: 16 de marzo del 2013]. [Disponible en: http://www.emagister.com/curso-programacion-avanzada/frameworks-arquitecturas-software].

Za, Angélica. (2012). "Lenguaje unificado modelado". [En línea: 2012]. [Citado el: 2 de febrero del 2012]. [Disponible en: http://www.buenastareas.com/ensayos/Lenguaje-Unificado-Modelado/3527759.html].

Bibliografías consultadas

Bsi. (2013). "¿Qué son los sistemas de gestión?". [En línea: 2013]. [Citado el: 30 de mayo del 2013]. [Disponible en: http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/].

EcuRed. (2013). "Lenguajes de Programación". [En línea: 2013]. [Citado el:16 de mayo del 2013] [Disponible en: http://www.ecured.cu/index.php/Lenguaje_de_programaci%C3%B3n].

EditBoard.com. (2010). "Tecnicas de programación". [En línea: 2010]. [Citado el: 19 de Marzo de 2013] [Disponible en: http://catedraprogramacion.foroactivos.net/t117-que-es-tecnica-de-programacion]15/10/2010].

HERNÁNDEZ, SAMPIERI. (2006). "Metodología de la investigación". 4ta Edición. México D.F.: McGraw-Hill, 2006. 882p. ISBN: 9701036322.

Hurtado, Sola, Elianys (2008). "Sistema de Gestión de Sesiones". ¨[En línea: 2008]. [Citado el: 18 de Febrero del 2013]. [Disponible en: http://repositorio_institucional.uci.cu/jspui/simple-search?query=%EF%83%98%09Sistema+de+Gesti%C3%B3n+de+Sesiones&submit=Buscar+].

JACOBSON, Ivar. (2001). ""El Proceso Unificado de Desarrollo de Software". [En línea: 2001]. [Citado el: 15 de febrero del 2013]. [Disponible en: http://eva.uci.cu/mod/resource/view.php?id=8500&subdir=/El_Proceso_Unificado_de_Desarrollo.].

LANCKER, VAN. (2011). ""CSS 1 y CSS 2.1: Hojas de estilo para enriquecer el código HTML". .Barcelona: Ediciones ENI. 2007. [En línea: 2011]. [Disponible en: http://books.google.com.cu/books?id=xBrgWWa31pcC&printsec=frontcover&dq=css&hl=es&sa=X&ei=n7WzT_SCDIiXtweouY3NCA&ved=0CGYQ6AEwCQ#v=onepage&q=css&f=false. ISBN: 9782746035836].

Larman, Craing. (2005). Uml y Patrones:Introducción al análisis y diseño orientado a objetos.2ed.PrenticeHall,2005.

Marchese, Javier. (2006). La implementación del proceso de revisión de requerimientos.

Marqués, María Mercedes. (2001). "Apuntes de Ficheros y Bases de Datos". [Citado el: 6 de Enero de 2013]. [Disponible en: http://repositori.uji.es/xmlui/bitstream/handle/10234/7314/Apuntes%20de%20Ficheros%20y%20Bases%20de%20Datos.htm?sequence=1].

MATOS, GARCÍA, ROSA MARIA (1999). DISEÑO de BASES DE DATOS. 40 p. [En línea: 1999]. [Citado el:18 de febrero del 2013].

Paradigm (2011). ""Visual Paradigm for UML". [En línea: 2011]. [Disponible en: http://www.visual-paradigm.com/product/vpuml/].

Pérez, Valdés, Damián. (2007). "¿Qué es Javascript?". [En línea: 2007]. [Citado el: 15 de enero del 2013]. [Disponible en: http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript]

slideshare. (2009). "Programación y Algoritmos". [En línea: 2009]. [Citado el: 5 marzo del 2013] [Disponible en: http://sunshine.prod.uci.cu/search/programaci%C3%B3n%20orientada%20a%20objetos.].

Valdés, Cabrera, Yandy (2005). Subsistema Gestión de Seguridad. [En línea: 2005]. [Citado el 15 de Marzo de 2013]. [Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_0135_05].

WikiLibros. (2011). "Técnicas básicas de programación". [En línea: 2011]. [Citado el: 18 de marzo del 2013]. [Disponible en: http://es.wikibooks.org/wiki/Fundamentos_de_programaci%C3%B3n/T%C3%A9cnicas_b%C3%A1sicas_de_programaci%C3%B3n].

WordPress. (2009). "Sistemas de Gestión". [En línea:2009]. [Citada el: 25 de abril del 2013]. [Disponible en: http://gcarenas.wordpress.com/2009/02/24/unidad-1-conceptos-basicos/].

Glosario de términos

Área de procesos: Aquellas actividades que facilitan el camino de la mejora. En cada una de estas áreas se define qué hay que hacer pero no cómo hay que hacerlo.

Capacidad de un proceso: Atributo de los procesos. El nivel de capacidad de un proceso indica si solo se ejecuta, o si también se planifica se encuentra organizativa y formalmente definido, se mide y se mejora de forma sistemática.

HTTPS: El Protocolo Seguro de Transferencia de Hipertexto es un protocolo de red basado en *HTTP* por lo que está orientado a transacciones, sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores y sigue el esquema petición-respuesta entre un cliente y un servidor. La principal diferencia entre ellos es que este está destinado a la transferencia segura de datos de hipertexto, en otras palabras, es la versión segura de *HTTP*.

IP: EL Protocolo de Internet (IP, por sus siglas en inglés) es un protocolo de comunicación de datos digitales clasificado funcionalmente en la Capa de Red según el modelo internacional OSI.

Madurez de un proceso: Atributo de las organizaciones que desarrollan o mantienen los sistemas de *software*. En la medida que estas llevan a cabo su trabajo siguiendo procesos, y en la que estos se encuentran homogéneamente implantados, definidos con mayor o menor rigor; conocidos y ejecutados por todos los equipos de la empresa; y medidos y mejorados de forma constante, las organizaciones serán más o menos maduras.

PC: Computadora personal (PC, por sus siglas en inglés).

Slony-I: Es un sistema de replicación asíncrono para PostgreSQL de una base de datos maestra hacia una o múltiples bases de datos hijas. Realiza las actualizaciones a través de disparadores o triggers por lo que actualmente solo puede realizar replicación de tablas y secuencias.

TCP/IP: Es un conjunto de protocolos. La sigla TCP/IP significa "Protocolo de control de transmisión/Protocolo de Internet". Proviene de los nombres de dos protocolos importantes del conjunto de protocolos, es decir, del protocolo TCP y del protocolo IP.

UML: Es uno de los lenguajes de modelado más conocidos y utilizados en la actualidad. Fue estandarizado por el Grupo de Gestión de Objetos en noviembre de 1997 y pensado para ser usado en sistemas desarrollados en diferentes lenguajes y plataformas.

URI: *(Uniform Resource Identifier)* es una cadena de caracteres corta que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia). Su principal diferencia con el *Uniform Resource Identifier* o *URL* (Localizador Uniforme de Recurso) es que permite especificar que parte del recurso se solicita.

Anexos

Anexo 1. Características de un SGBD

Independencia: La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

Consistencia: En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la base de datos representa aspectos de la realidad que tiene determinadas condiciones, por ejemplo que los menores de edad no pueden tener licencia de conducir. El sistema no debería aceptar datos de un conductor menor de edad. En los SGBD existen herramientas que facilitan la programación de este tipo de condiciones.

Seguridad: La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero desorientado. Normalmente, los SGBD disponen de un complejo sistema de permisos.

Integridad: Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de *hardware*, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada. Los SGBD proveen mecanismos para garantizar la recuperación de la base de datos hasta un estado consistente conocido en forma automática.

Control de la concurrencia: En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, o para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

Manejo de transacciones: Una transacción es un programa que se ejecuta como una sola operación. Esto quiere decir que el estado luego de una ejecución en la que se produce una falla es el mismo que se obtendría si el programa no se hubiera ejecutado. Los SGBD proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.

Respaldo: Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

Tiempo de respuesta: Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

Anexo 2. Fases del proceso de mejora

El proceso de mejora basado en el nivel 2 de *CMMI* cuenta con varias fases para el desarrollo del producto. Estas fases se describen a continuación:

- **Inicio**

Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la evaluación de la factibilidad del proyecto, la planeación del proyecto a un alto nivel y el registro de este. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, y decidir si se ejecuta o no el proyecto. Los objetivos de esta fase son asegurar la factibilidad del proyecto y establecer un plan para la ejecución del proyecto.

- **Desarrollo**

En esta fase se ejecutan las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se refinan los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. El objetivo de esta fase es obtener un sistema que satisfaga las necesidades de los clientes y usuarios finales.

- **Transición**

Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del *software*. Los objetivos de esta fase son desplegar el producto en un ambiente operacional y transferir el producto al cliente.

- **Cierre**

En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto. El objetivo de esta fase es analizar los resultados y las experiencias.

Anexo 3. Ventajas y características de un marco de trabajo

El uso de un marco de trabajo tiene muchas ventajas entre las que se encuentran:

- El programador no necesita plantearse una estructura global de la aplicación, sino que el *framework* le proporciona un esqueleto que hay que "rellenar".
- Facilita la colaboración. Cualquiera que haya tenido que "pelearse" con el código fuente de otro programador (o incluso con el propio, pasado algún tiempo) sabrá lo difícil que es entenderlo y modificarlo; por tanto, todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.

- Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al *framework* concreto para facilitar el desarrollo.

A continuación se presentan las características de un marco de trabajo:

- Versatilidad: Es capaz de trabajar la mayoría de los entornos o servidores, incluso en sistemas de alojamiento compartido, donde sólo se tiene acceso por *FTP* para enviar los archivos al servidor y donde no se cuenta con acceso a su configuración.
- Compatibilidad: Es compatible con la versión *PHP 4*, lo que hace que se pueda utilizar en cualquier servidor, incluso en algunos antiguos. Funciona correctamente también en *PHP 5*.
- Actualizado: Desde la versión 2 ya solo es compatible con la versión 5 de *PHP*. Donde se usa aún la versión 4 de *PHP* se debe hacer uso de la versión antigua del marco de trabajo.
- Facilidad de instalación: Solo es necesaria una cuenta *FTP* para subir *CodeIgniter* al servidor y su configuración se realiza con apenas la edición de un archivo, donde se debe escribir el acceso a la base de datos.
- Flexibilidad: Es bastante menos rígido que otros *frameworks*. Define una manera de trabajar específica, pero en muchos de los casos sus reglas de codificación obviadas. Algunos módulos como el uso de plantillas son totalmente opcionales.
- Ligereza: El núcleo es bastante ligero, lo que permite que el servidor no se sobrecargue interpretando o ejecutando grandes porciones de código. La mayoría de los módulos o clases que ofrece se pueden cargar de manera opcional sólo cuando se van a utilizar.
- Documentación tutorializada: La documentación de *CodeIgniter* es fácil de seguir y de asimilar, porque está escrita en modo de tutorial. Esto no facilita mucho la referencia rápida, cuando ya sabemos acerca del *framework* y queremos consultar sobre una función o un método en concreto, pero para iniciar el trabajo con este *framework* ayuda mucho.

Anexo 4. Modelo de entrevista

Tabla 6. Modelo de entrevista

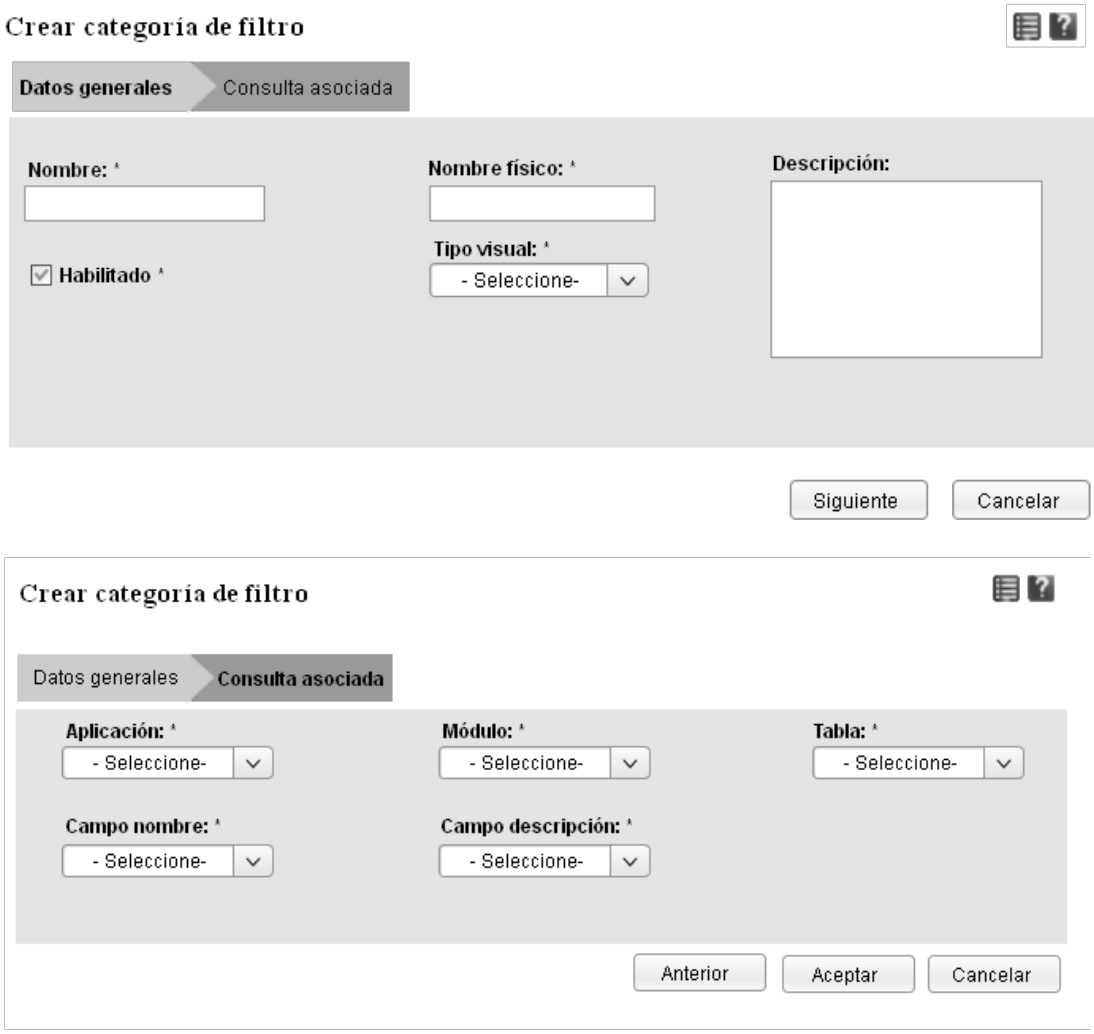
Modelo de entrevista para la obtención de requisitos
Nombre(s) del entrevistado:
Apellidos del entrevistado:
Cargo que ocupa:
Centro al que pertenece:
Preguntas

1. ¿En que se basa la seguridad en el SGU? 2. ¿Por qué se hace necesario gestionar la seguridad en el SGU? 3. ¿Cómo se gestiona actualmente la seguridad en el SGU? 4. ¿Qué beneficios traerá para el sistema la solución informática? 5. ¿A qué parte del SGU se integrará la solución informática? 6. ¿Qué funcionalidades se deben brindar? 7. ¿Qué personas tendrán acceso al módulo de Seguridad del SGU?
Resultados obtenidos:

Anexo 5. Especificación de requisitos funcionales

Tabla 7. Especificación de requisitos. Crear categoría de filtro

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_1	Crear categoría de filtro	El requisito permite crear una categoría de filtro. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción Categorías de Filtros. El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. En el área de iconos flotantes se selecciona la opción: Crear. Se introducen los datos: Nombre, Nombre físico, Descripción, Estado (Habilitado/Deshabilitado), Tipo visual, Aplicación, Módulo, Tabla, Campo nombre, Campo descripción.	Alta	Alta
Prototipo				



Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• Varchar	• Es un campo obligatorio. • No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Nombre físico	• Varchar	• Es un campo obligatorio. • No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Descripción	• Varchar	• Admite cualquier tipo de carácter. • Solo admite 30 caracteres por palabra.

		• Admite entre 0 y 100 caracteres válidos.
• Estado	• Boolean	• Selección. • Es un campo obligatorio.
• Tipo visual	• Varchar	• Selección. • Es un campo obligatorio.
• Aplicación	• Varchar	• Selección. • Es un campo obligatorio.
• Módulo	• Varchar	• Selección. • Es un campo obligatorio.
• Tabla	• Varchar	• Selección. • Es un campo obligatorio.
• Campo nombre	• Varchar	• Selección. • Es un campo obligatorio.
• Campo descripción	• Varchar	• Selección. • Es un campo obligatorio.

Tabla 8. Especificación de requisitos. Listar categoría de filtro

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_2	Listar categoría de filtro	El requisito permite mostrar un listado con todas las categorías de filtros existentes hasta el momento. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción Categorías de Filtros. Se muestra un listado con las categorías de filtros existentes y en el área de iconos internos las opciones: Ver detalles y Modificar, según el tipo de categoría. En el área de iconos flotantes se muestran	Media	Media

		las opciones: Crear y Ayuda.		

Prototipo

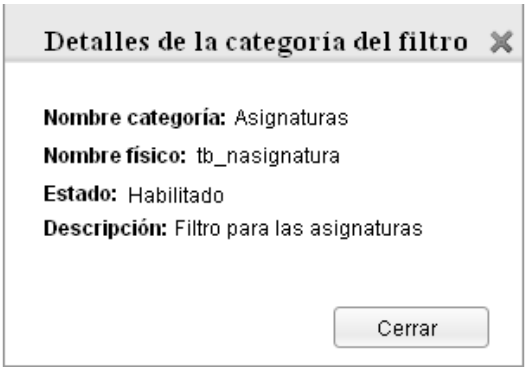
Campos	Tipos de Datos	Reglas o Restricciones
• Nombre categoría	• Varchar	• No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Cantidad por página	• Varchar	• Selección.

Tabla 9. Especificación de requisitos. Ver detalles de categoría de filtro

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_3	Ver detalles de categoría de filtro	El requisito permite ver los detalles de la categoría de filtro. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción Categorías de	Media	Media

		Filtros. En el listado de las categorías de filtro que muestra el sistema se selecciona la acción interna: Ver Detalles de Categoría. Se muestran los datos en forma de ventana emergente: Nombre categoría, Nombre físico, Estado (Habilitado/Deshabilitado), Descripción.		

Prototipo



Campos	Tipos de Datos	Reglas o Restricciones
• Nombre categoría	• Varchar	• Solo lectura.
• Nombre físico	• Varchar	• Solo lectura.
• Estado	• Varchar	• Solo lectura.
• Descripción	• Varchar	• Solo lectura.

Tabla 10. Especificación de requisitos. Modificar categoría de filtro

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_4	Modificar categoría de filtro	El requisito permite modificar una categoría de filtro. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en	Alta	Alta

		la agrupación funcional Seguridad de negocio selecciona la opción Categorías de Filtros. En el listado de las categorías de filtros que muestra el sistema se selecciona la acción interna Modificar Categoría. Se muestran los datos almacenados de dicha categoría: Nombre, Nombre físico, Descripción, Estado (Habilitado/Deshabilitado), Tipo visual. Se modifican los datos.		

Prototipo

Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• Varchar	• Es un campo obligatorio. • No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Nombre físico	• Varchar	• Es un campo obligatorio. • No admite números ni apóstrofes ni caracteres especiales.

		• Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Descripción	• Varchar	• Admite entre 0 y 100 caracteres válidos. • Admite cualquier tipo de carácter. • Solo admite 30 caracteres por palabra.
• Estado	• Boolean	• Selección. • Es un campo obligatorio.
• Tipo visual	• Varchar	• Selección. • Es un campo obligatorio.

Tabla 11. Especificación de requisitos. Listar reglas de negocio

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_5	Listar reglas de negocio	El requisito muestra un listado con todos los usuarios registrados en el sistema, brindando la posibilidad de asociarles determinados permisos. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Reglas de Negocio. Se introduce un criterio de búsqueda y el sistema muestra un listado con los usuarios registrados. En el listado mostrado, el área a de íconos internos se muestran las opciones: Crear permisos, Crear grupo de permisos, Asociar permiso a usuario, Asociar grupo de permisos a usuarios. En el área de íconos flotantes se muestra la opción: Ayuda.	Media	Media

Prototipo

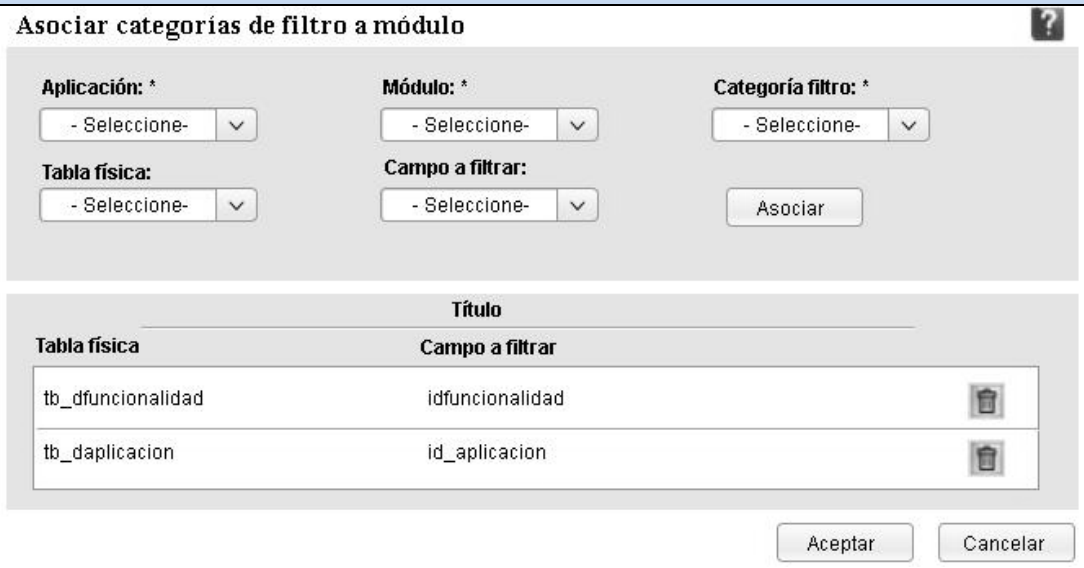

Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• Varchar	• No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Filtro de búsqueda	• Varchar	• Selección.
• Filtro de roles	• Varchar	• No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Cantidad por página	• Varchar	• Selección.

Tabla 12. Especificación de requisitos. Asociar filtro a módulo

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_6	Asociar filtro a módulo	El requisito permite asociar filtros a	Alta	Alta

		módulos. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Asociar filtros El sistema muestra los campos a llenar. Se introducen los datos: Aplicación, Módulo, Categoría filtro, Tabla física, Campo a filtrar.		

Prototipo

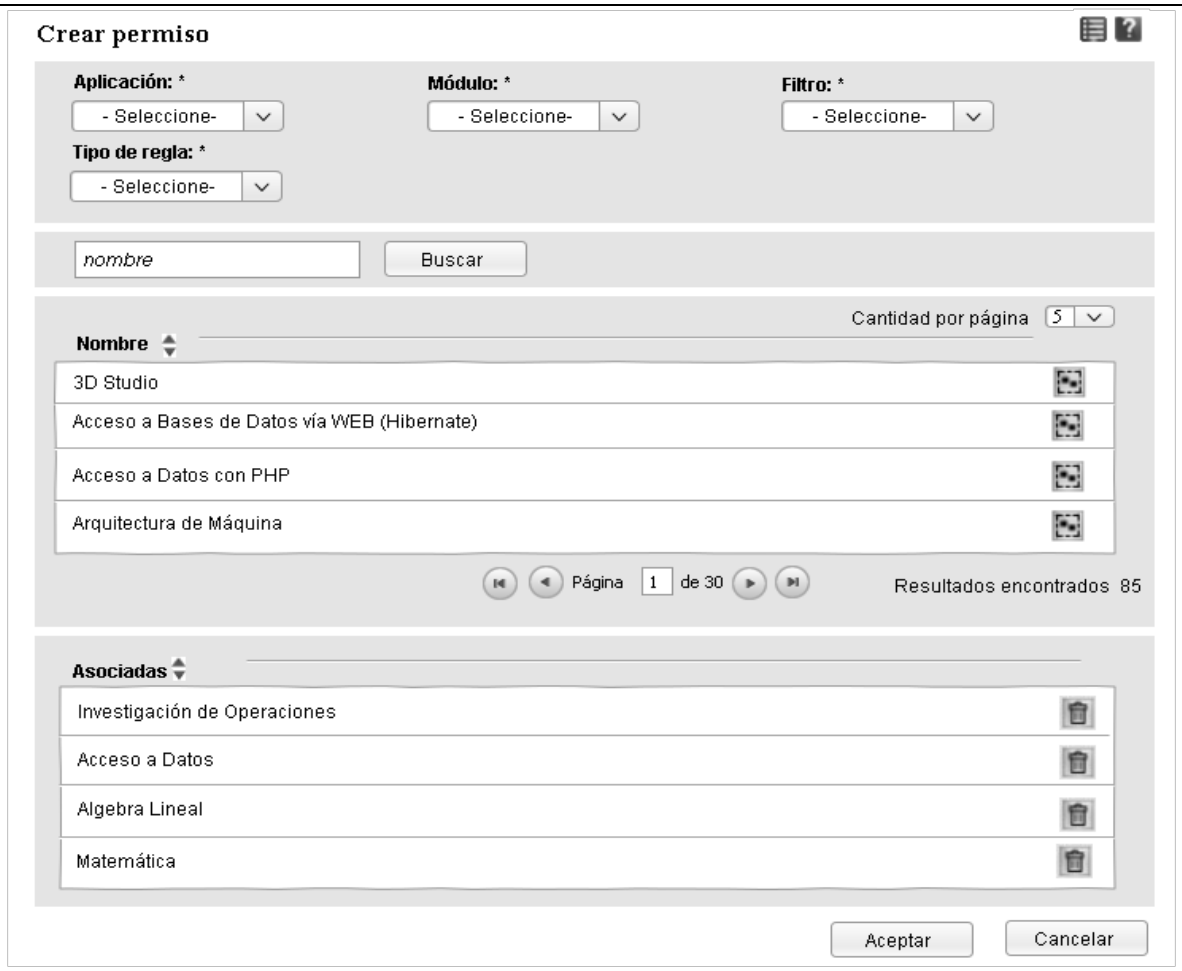


Campos	Tipos de Datos	Reglas o Restricciones
• Aplicación	• Varchar	• Es un campo obligatorio. • Selección. • Es un campo obligatorio.
• Módulo	• Varchar	• Es un campo obligatorio. • Selección. • Es un campo obligatorio.
• Categoría filtro	• Varchar	• Es un campo obligatorio. • Selección. • Es un campo obligatorio.

• Tabla física	• Varchar	• Selección. • Es un campo obligatorio.
• Campo a filtrar	• Varchar	• Selección. • Es un campo obligatorio.

Tabla 13. Especificación de requisitos. Crear permiso

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_7	Crear permiso	El requisito permite crear un permiso. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Reglas de Negocio. En el listado que muestra el sistema con todos los usuarios registrados, se selecciona en el área de íconos internos la opción: Crear permiso. Se introducen los datos: Aplicación, Módulo, Filtro, Tipo de regla, Nombre del elemento.	Alta	Alta
Prototipo				



Campos	Tipos de Datos	Reglas o Restricciones
• Aplicación	• Varchar	• Es un campo obligatorio. • Selección.
• Módulo	• Varchar	• Es un campo obligatorio. • Selección.
• Filtro	• Varchar	• Es un campo obligatorio. • Selección.
• Tipo de regla	• Varchar	• Es un campo obligatorio. • Selección.
• Nombre del elemento	• Varchar	• No admite números ni apóstrofes ni caracteres especiales.

		• Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.

Tabla 14. Especificación de requisitos. Listar permiso

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_8	Listar permiso	El requisito permite mostrar un listado con todos los permisos que tiene asociado un usuario. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Reglas de Negocio. Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se muestran las opciones: Crear permiso, Crear grupo de permisos, Asociar permiso a usuario y Asociar grupo de permisos a usuarios. Se selecciona la opción: Crear permiso y en el área de íconos flotantes se ejecuta la acción Listar. Se muestra un listado con los permisos asociados y en el área de íconos internos las opciones: Ver Detalles, Modificar y Eliminar. En el área de íconos flotantes se muestran las opciones: Crear permiso, Listar y Ayuda.	Media	Media
Prototipo				

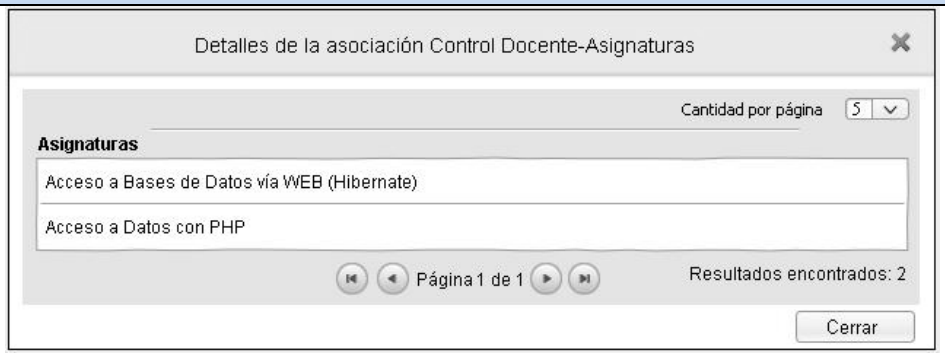
Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• Varchar	• No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Cantidad por página	• Varchar	• Selección.

Tabla 15. Especificación de requisitos. Ver detalles del permiso

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_9	Ver detalles del permiso	El requisito permite ver los detalles los permisos que tiene asociado una persona. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Reglas de Negocio. Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se muestran las opciones: Crear permiso, Crear grupo de	Media	Media

		permisos, Asociar permiso a usuario y Asociar grupo de permisos a usuarios. Una vez listado el permiso, se muestran los permisos asociados y en el área de íconos internos se selecciona la opción: Ver Detalles. En el área de íconos flotantes se muestran las opciones: Crear permiso, Listar y Ayuda. Se muestran los datos en forma de ventana emergente.		

Prototipo

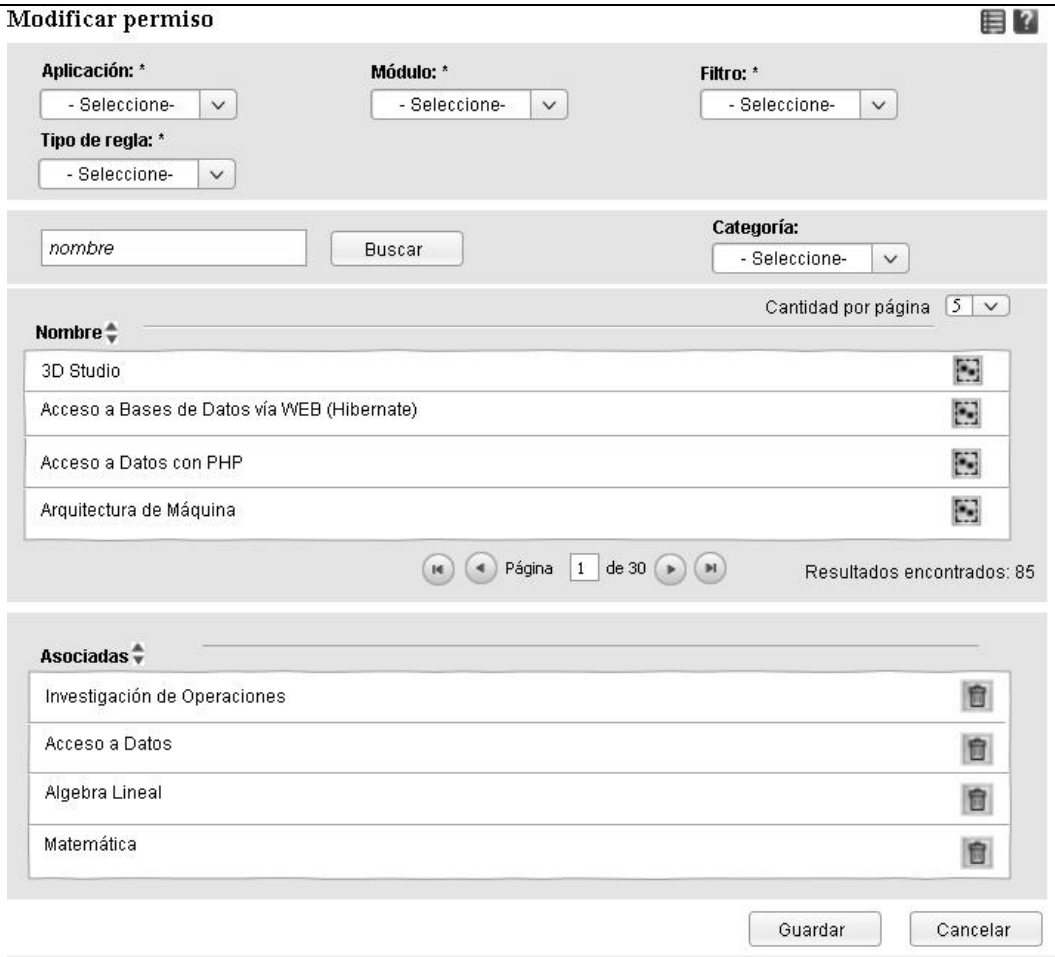


Campos	Tipos de Datos	Reglas o Restricciones
• Cantidad por página	• Varchar	• Selección

Tabla 16. Especificación de requisitos. Modificar permiso

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_10	Modificar permiso	El requisito permite modificar un permiso. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Reglas de Negocio.	Alta	Alta

		Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se muestran las opciones: Crear permiso, Crear grupo de permisos, Asociar permiso a usuario y Asociar grupo de permisos a usuario. Una vez listado el permiso, se muestran los permisos asociados y en el área de íconos internos se selecciona la opción: Modificar. El usuario modifica el permiso añadiendo o eliminando alguno de los elementos asociados. En el área de íconos flotantes se muestran las opciones: Crear permiso, Listar y Ayuda.		
Prototipo				



Campos	Tipos de Datos	Reglas o Restricciones
• Aplicación	• Varchar	• Es un campo obligatorio. • Selección.
• Módulo	• Varchar	• Es un campo obligatorio. • Selección.
• Filtro	• Varchar	• Es un campo obligatorio. • Selección.
• Tipo de regla	• Varchar	• Es un campo obligatorio. • Selección.
• Nombre del elemento	• Varchar	• No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres.

		• Solo admite 30 caracteres por palabra.
• Filtro de categoría	• Varchar	• Selección.
• Cantidad por página	• Varchar	• Selección.

Tabla 17. Especificación de requisitos. Eliminar permiso

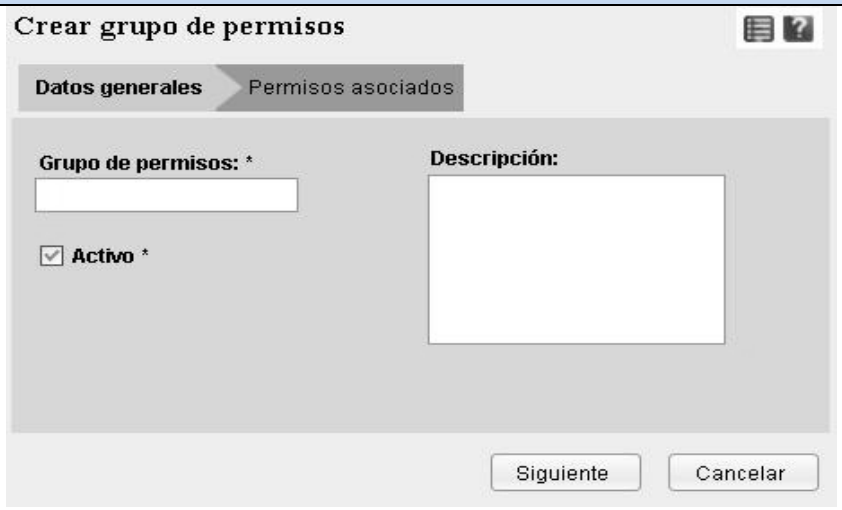
No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_11	Eliminar permiso	El requisito permite eliminar un permiso. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Reglas de Negocio. Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se muestran las opciones: Crear permiso, Crear grupo de permisos, Asociar permiso a usuario y Asociar grupo de permisos a usuario. Una vez creado y listado el permiso, se muestran los permisos asociados y en el área de íconos internos se selecciona la opción: Eliminar. El sistema muestra el mensaje de advertencia: ¿Desea realmente eliminar esta asociación? En el área de íconos flotantes se muestran las opciones: Crear permiso, Listar y Ayuda.	Media	Media

Tabla 18. Especificación de requisitos. Crear grupo de permisos

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_12	Crear grupo de permisos	El requisito permite crear un grupo de permisos. Versión 1. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Reglas de Negocio. En el listado que muestra el sistema con todos los usuarios registrados, se selecciona en el área de íconos internos la opción: Crear grupo de permisos. Se introducen los datos: Nombre del grupo de permisos, Descripción, Estado (Habilitado/ Deshabilitado), Aplicación, Módulo, Filtro, Tipo de regla, Nombre del elemento, Filtro de categoría. En el área de íconos flotantes se muestran las opciones: Listar y Ayuda. Versión 2. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Grupo de permisos. El sistema muestra un listado con todos los grupos de permisos existentes hasta el momento. En el área de íconos flotantes se selecciona la opción: Crear grupo de permisos.	Alta	Alta

		Se introducen los datos: Nombre del grupo de permisos, Descripción, Estado (Habilitado/ Deshabilitado), Aplicación, Módulo, Filtro, Tipo de regla, Nombre del elemento, Filtro de categoría. En el área de íconos flotantes se muestran las opciones: Listar y Ayuda.		

Prototipo



Campos	Tipos de Datos	Reglas o Restricciones
• Nombre del grupo de permisos	• Varchar	• Es un campo obligatorio. • No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres.

		• Solo admite 30 caracteres por palabra.
• Descripción	• Varchar	• Admite entre 0 y 100 caracteres válidos. • Admite cualquier tipo de carácter. • Solo admite 30 caracteres por palabra.
• Estado	• Boolean	• Es un campo obligatorio. • Selección.
• Aplicación	• Varchar	• Selección.
• Módulo	• Varchar	• Selección.
• Filtro	• Varchar	• Selección.
• Tipo de regla	• Varchar	• Selección.
• Nombre del elemento	• Varchar	• Es un campo obligatorio. • No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Filtro de categoría	• Varchar	• Selección.

Tabla 19. Especificación de requisitos. Listar grupo de permisos

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_13	Listar grupo de permisos	El requisito permite mostrar un listado con todos los grupos de permisos que una persona tiene asociado. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Grupos de permisos. El sistema muestra un listado con los	Media	Media

		grupos de permisos existentes. En el área de íconos internos se muestran las opciones: Ver detalles, Modificar, Agregar un usuario a un grupo de permisos y Desagregar usuario de un grupo de permisos. En el área de íconos flotantes se muestran las opciones: Crear, Actualizar y Ayuda.		

Prototipo

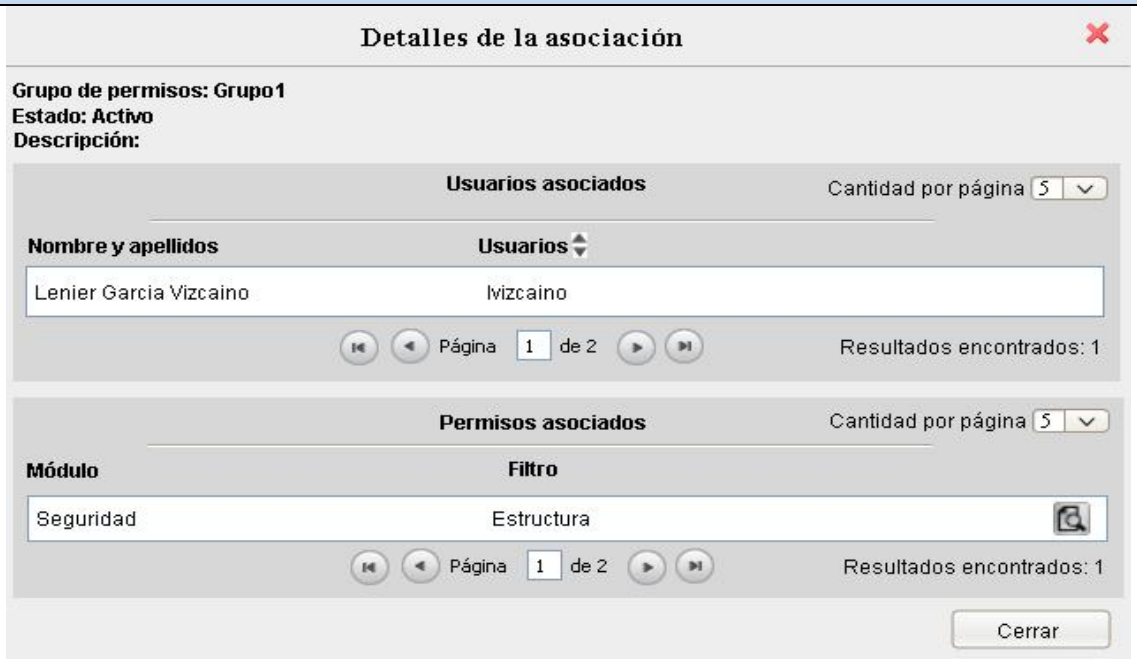
Campos	Tipos de Datos	Reglas o Restricciones
• Cantidad por página	• Varchar	• Selección.

Tabla 20. Especificación de requisitos. Ver detalles de grupo de permisos

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_14	Ver detalles de grupo de permisos	El requisito permite mostrar un listado con todos los grupos de permisos que una persona tiene asociado. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en	Media	Media

		la agrupación funcional Seguridad de negocio selecciona la opción: Grupos de permisos. En el listado que muestra el sistema con los grupos de permisos existentes se selecciona en el área de íconos internos la opción: Ver detalles. Se muestran los datos en forma de ventana emergente: Nombre del grupo de permisos, Estado, y Descripción.		

Prototipo



Campos	Tipos de Datos	Reglas o Restricciones
• Nombre del grupo de permisos	• Varchar	• Solo lectura.
• Estado	• Boolean	• Solo lectura.
• Descripción	• Varchar	• Solo lectura.

Tabla 21. Especificación de requisitos. Modificar grupo de permisos

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_15	Modificar grupo de permisos	El requisito permite modificar un grupo de permisos. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Grupos de permisos. Se muestra un listado con los grupos de permisos creados y el área de íconos internos se selecciona la opción: Modificar. Se modifican los datos deseados. En el área de íconos flotantes se muestran las opciones: Ver detalles y Ayuda.	Alta	Alta
Prototipo				
				

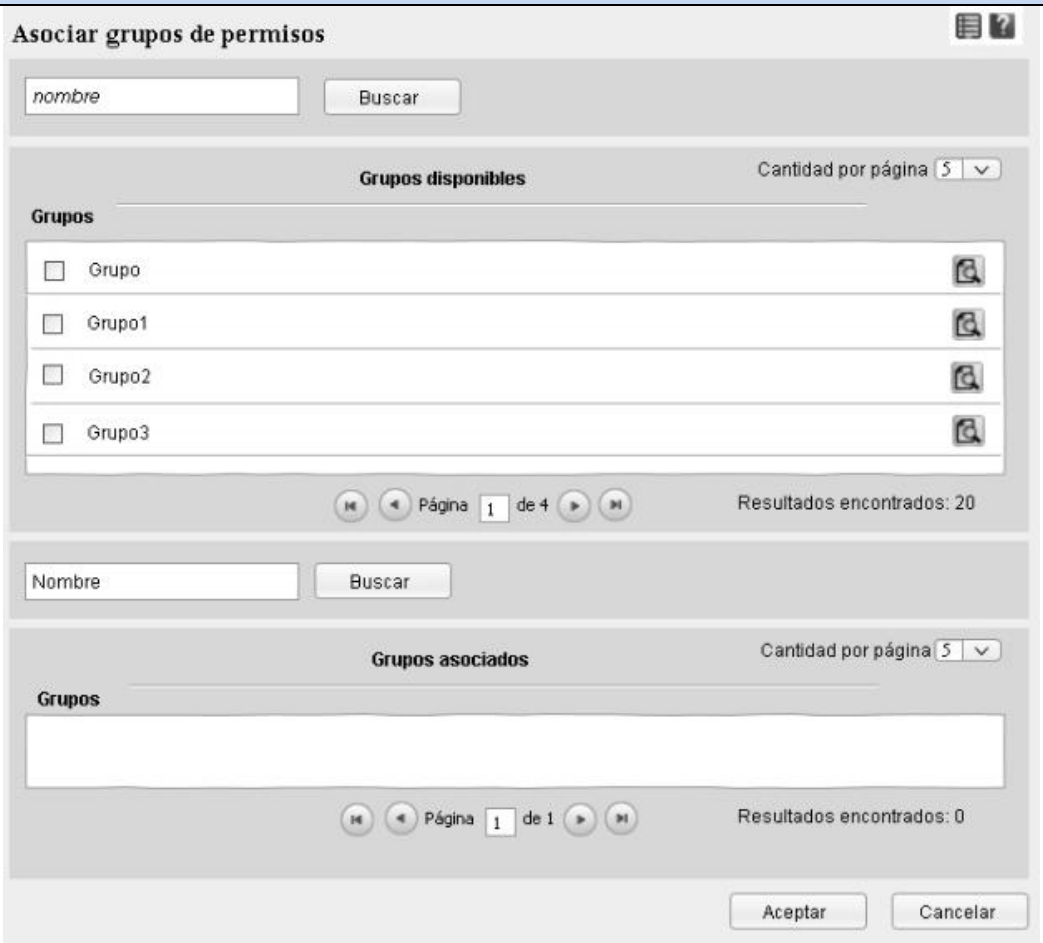
Campos	Tipos de Datos	Reglas o Restricciones
• Nombre del grupo de permisos	• Varchar	• No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Descripción	• Varchar	• Admite entre 0 y 100 caracteres válidos. • Admite cualquier tipo de carácter. • Solo admite 30 caracteres por palabra.
• Estado	• Boolean	• Es un campo obligatorio.
• Aplicación	• Varchar	• Selección.
• Módulo	• Varchar	• Selección.

• Filtro	• Varchar	• Selección.
• Tipo de regla	• Varchar	• Selección.
• Nombre del elemento	• Varchar	• Es un campo obligatorio. • No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Filtro categoría	• Varchar	• Selección.

Tabla 22. Especificación de requisitos. Asociar grupo de permisos a usuario

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_16	Asociar grupo de permisos a usuario	El requisito permite asociar un grupo de permisos a un usuario. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Reglas de Negocio. En el listado que muestra el sistema con todos los usuarios registrados, se selecciona en el área de íconos internos la opción: Asociar grupo de permisos. El sistema muestra los grupos de permisos disponibles para ese usuario y los grupos de permisos que ya tiene asociados. Además, muestra también un buscador para cada grupo. En el área de íconos internos se muestran las opciones: Ver detalles. En el área de íconos flotantes se muestran las opciones Listar y Ayuda.	Alta	Alta

Prototipo



Campos	Tipos de Datos	Reglas o Restricciones
• Grupos disponibles	• Varchar	• No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.
• Grupos asociados	• Varchar	• No admite números ni apóstrofes ni caracteres especiales. • Admite entre 2 y 100 caracteres. • Solo admite 30 caracteres por palabra.

Tabla 23. Especificación de requisitos. Agregar usuarios a grupo de permisos

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_17	Agregar usuarios a un grupo de permisos.	El requisito permite agregar uno o varios usuarios a un grupo de permisos. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Grupo de permisos. El sistema muestra un listado con todos los grupos de permisos existentes hasta el momento. En el área de íconos internos se selecciona la opción: Agregar usuarios a grupo de permiso. Se seleccionan los usuarios que se desean agregar. En el área de íconos flotantes las opciones: Ver detalles y Ayuda.	Alta	Alta
Prototipo				

Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• Varchar	• Es un campo obligatorio. • No admite números ni apóstrofes. • Admite entre 2 y 250 caracteres. • Solo admite 30 caracteres por palabra.

Tabla 24. Especificación de requisitos. Desagregar usuarios de un grupo de permisos

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_18	Desagregar un usuario de un grupo de permisos	El requisito permite desagregar uno o varios usuarios de un grupo de permisos. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Grupo de permisos. El sistema muestra un listado con todos los grupos de permisos existentes hasta el momento.	Alta	Alta

		En el área de íconos internos se selecciona la opción: Desagregar usuarios a grupo de permiso. Se seleccionan los usuarios que se desean quitar del grupo de permisos. En el área de íconos flotantes las opciones: Ver detalles y Ayuda.		

Prototipo

Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• Varchar	• Es un campo obligatorio. • No admite números ni apóstrofes. • Admite entre 2 y 250 caracteres. • Solo admite 30 caracteres por palabra.
• Cantidad por página	• Varchar	• Selección.

Tabla 25. Especificación de requisitos. Asociar permiso a usuario

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF_19	Asociar permiso a usuario	El requisito permite asociar un permiso a un usuario. El administrador selecciona el módulo Seguridad del Sistema de Gestión Universitaria y luego en el menú lateral, en la agrupación funcional Seguridad de negocio selecciona la opción: Reglas de Negocio. En el listado que muestra el sistema con todos los usuarios registrados, se selecciona en el área de íconos internos la opción: Asociar permisos a usuarios. Se muestra un listado con todos los permisos asociados hasta el momento y en el área de íconos internos se muestran las opciones: Ver detalles, Modificar y Eliminar. En el área de íconos flotantes se brinda la posibilidad de asociarle un nuevo permiso al usuario seleccionando la opción: Crear permiso. Se introducen los datos. (La Vista para crear un permiso fue descrita anteriormente, en la especificación del RF_7. Crear permiso).	Alta	Alta
Prototipo				

Campos	Tipos de Datos	Reglas o Restricciones
• Nombre categoría	• Varchar	• Es un campo obligatorio • No admite números ni apóstrofes. • Admite entre 2 y 250 caracteres. • Solo admite 30 caracteres por palabra.
• Nombre físico	• Varchar	• No admite números ni apóstrofes. • Admite entre 2 y 250 caracteres. • Solo admite 30 caracteres por palabra.
• Descripción	• Varchar	• Admite cualquier tipo de carácter. • Solo admite 30 caracteres por palabra. • Admite entre 0 y 100 caracteres válidos.
• Estado	• Varchar	• Es un campo obligatorio. • Selección.
• Tipo visual	• Varchar	• Es un campo obligatorio. • Selección.
• Aplicación	• Varchar	• Es un campo obligatorio • Selección.
• Módulo	• Varchar	• Es un campo obligatorio. • Selección.

• Tabla	• Varchar	• Es un campo obligatorio. • Selección.
• Campo nombre	• Varchar	• Es un campo obligatorio. • Selección.
• Campo descripción	• Varchar	• Es un campo obligatorio. • Selección.

Anexo 6. Especificación de requisitos no funcionales

Tabla 26. Especificación de requisitos no funcionales

Requisitos no Funcionales	
Usabilidad	
RNFP_1	Facilidad de uso por parte de los usuarios: El sistema debe presentar una interfaz amigable que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Debe, además, ser una interfaz de manejo cómodo que posibilite a los usuarios sin experiencia una rápida adaptación.
RNFP_2	Especificación de la terminología utilizada: El sistema debe adaptarse al lenguaje y términos utilizados por los usuarios en la rama abordada con vista a una mayor comprensión por parte del cliente de la herramienta de trabajo.
RNFP_3	Potencialidades de capacitación orientadas a interfaces intuitivas: lo que enaltece la posibilidad de que el usuario aprenda mediante el uso y explotación de la herramienta.
RNFP_4	Menús: El sistema debe presentar una serie de menús tanto laterales como en barra de íconos flotantes que permitan el acceso rápido a la información por parte de los usuarios, aprovechando así las potencialidades de estas estructuras.
RNFP_5	Emplear perfiles de usuario: Diferenciar las interfaces y opciones para los usuarios que accedan al sistema según los diferentes roles que estos tengan dentro del mismo.
Seguridad	
RNFP_6	La seguridad de la base de datos está a nivel de roles, con el fin de mantener la integridad de los datos en función del acceso de cada uno de ellos, trayendo consigo además la protección de la información.
RNFP_7	Políticas de seguridad por usuarios y roles: El sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.

RNFP_8	Registro sistemáticos de incidencias: El sistema debe ser capaz de registrar el accionar del usuario, así como permitir auditorías y exámenes de las trazas tanto en tiempo real como en históricos.
Eficiencia	
RNFP_9	El sistema debe soportar un tiempo de respuesta menor o igual a 5 segundos.
RNFP_10	El sistema debe soportar una conexión simultánea de más de 3000 usuarios.
Soporte	
RNFP_11	El sistema debe brindar como apoyo una Ayuda contextual en la cual se refleja detalladamente la explicación de cada una de las pantallas con sus respectivas funcionalidades.
RNFP_12	Grupo de soporte y asesoría: El sistema contará con un grupo de soporte y asesoría al cliente del producto destinado a brindar asesoría y soporte técnico al mismo.
Hardware	
RNFP_13	Para el desarrollo se requiere de una PC Intel Pentium 4 o superior, CPU 3GHZ o superior, 512 MB RAM o superior, 160 GB HDD o superior.
RNFP_14	Para la explotación del cliente se requiere de una PC Pentium 3 o superior, CPU 133 MHZ o superior, 256 RAM mínimo 512 RAM recomendada o superior.
RNFP_15	Para la explotación del servidor se requiere de un CPU Dual Core 2.0 GHZ o superior, memoria RAM de 4 GB (recomendado 6 GB), 250 GB HDD.
Documentación de usuarios en línea y ayuda del sistema	
RNFP_16	Manual de usuario: El sistema deberá presentar un manual de usuario, permitiendo con ello un correcto uso de sus funcionalidades y brindarle al usuario una mayor experiencia del trabajo con el mismo.
RNFP_17	Documentación actualizada del grupo de desarrollo: Se precisa que la documentación del sistema esté actualizada en todos los aspectos, fases de trabajo y ciclos de desarrollo del mismo, permitiendo con ello un respaldo tanto ingenieril como legal del desarrollo de dicho sistema.
Interfaz	
RNFP_18	La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo TCP/IP.
RNFP_19	La comunicación entre el cliente y el servidor de aplicaciones se lleva a través del protocolo de conexión segura HTTPS.
RNFP_20	La comunicación entre el servidor y el directorio activo se hará mediante el protocolo LDAP.
Restricciones de diseño	

RNFP_21	IDE de desarrollo: NetBeans 7.1.
RNFP_22	Sistema Gestor de BD: PostgreSQL 8.4.
RNFP_23	Lenguaje de programación: PHP 5.3.
RNFP_24	Navegador Web: Mozilla Firefox 6 o superior.
RNFP_25	Marco de trabajo base de desarrollo: GUUD con la integración de CodeIgniter 1.7.2 y JQuery 1.3.2.

Anexo 7. Modelo lógico de la Base de datos

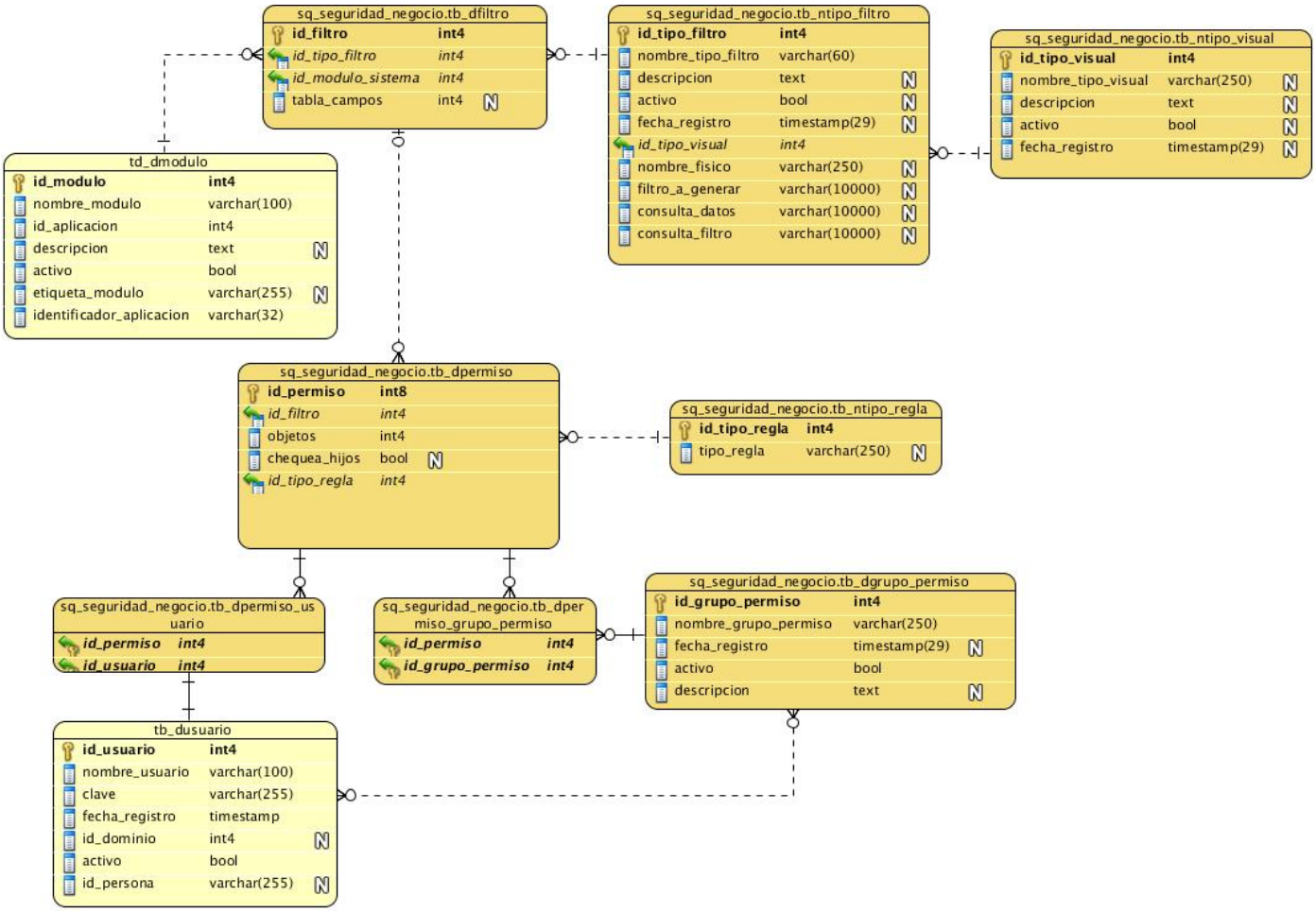


Figura 28. Modelo físico de la base de datos

Anexo 8. Pruebas unitarias realizadas a la solución informática

Casos de prueba de caja blanca

Tabla 27. CENIA_SGU_S_DCP_GP

Prueba estructural de caja blanca	**Código caso de prueba: CENIA_SGU_S_DCP_CP**
Probador: Lenier Garcia Vizcaino	

Código al que se aplica:

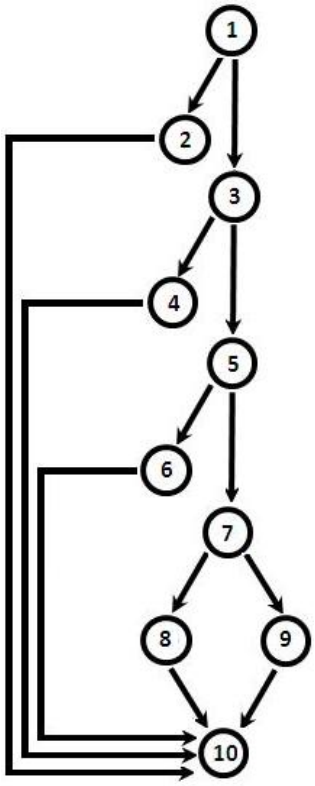
```
public function registrarPermiso($post_vars) {
    $result = $this->reglas_negocio_lib->registrarPermiso($post_vars);
    if ($result == "crear") {
        return "El elemento ha sido creado satisfactoriamente.";
    }elseif ($result == "modificar") {
        return "El elemento ha sido modificado satisfactoriamente.";
    }elseif ($result == "array vacio") {
       return "Debe seleccionar un elemento.";
    }elseif ($result == "campo vacio") {
       return "Uno o varios elementos se ha introducido de forma incorrecta.";
    }else {
       return "Ocurrió un error durante la operación. Intente más tarde.";
    }
}
```

Complejidad ciclomática:

V (G) = (A - N) + 2 = (13 -10) + 2 = 5

Caminos independientes

1) 1 – 2 – 10
2) 1 – 3 – 4 – 10
3) 1 – 3 – 5 – 6 – 10
4) 1 – 3 – 5 – 7 – 8 – 10
5) 1 – 3 – 5 – 7 – 9 – 10



Caso de prueba para el camino básico 1	
Descripción: los datos de entrada serán los atributos del permiso a crear o modificar.	
Condición de ejecución: los datos de entrada son válidos y no se encuentran en la base de datos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo vacío.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="El elemento ha sido creado satisfactoriamente." $nombrePrueba="Prueba:'Registrar Permiso'"; echo $this->ejecutarPrueba($this->reglas_negocio->registrarPermiso($post_vars),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 2	
Descripción: los datos de entrada serán los atributos del permiso a crear o modificar.	
Condición de ejecución: los datos de entrada son válidos y se encuentran en la base de datos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos del permiso insertados por el usuario.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="El elemento ha sido modificado satisfactoriamente." $nombrePrueba="Prueba:'Registrar Permiso'"; echo $this->ejecutarPrueba($this->reglas_negocio->registrarPermiso($post_vars),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 3	
Descripción: los datos de entrada serán los atributos del permiso a crear o modificar.	
Condición de ejecución: los datos de entrada relacionados con los objetos de negocio están vacíos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos del permiso insertados por el usuario.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="Debe seleccionar un elemento." $nombrePrueba="Prueba:'Registrar Permiso'"; echo $this->ejecutarPrueba($this->reglas_negocio-	

>registrarPermiso($post_vars),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 4	
Descripción: los datos de entrada serán los atributos del permiso a crear o modificar.	
Condición de ejecución: los datos de entrada se introducen de manera incorrecta.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos del permiso insertados por el usuario.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="Uno o varios elementos se han introducido de forma incorrecta." $nombrePrueba="Prueba:'Registrar Permiso'"; echo $this->ejecutarPrueba($this->reglas_negocio->registrarPermiso($post_vars),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 5	
Descripción: los datos de entrada serán los atributos del permiso a crear o modificar.	
Condición de ejecución: ocurre un error a la hora de guardar los datos en la base de datos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos del permiso insertados por el usuario.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="Ocurrió un error durante la operación. Intente más tarde." $nombrePrueba="Prueba:'Registrar Permiso'"; echo $this->ejecutarPrueba($this->reglas_negocio->registrarPermiso($post_vars),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	**Satisfactorio**
Resultado final de la prueba: Prueba satisfactoria.	

Tabla 28. CENIA_SGU_S_DCP_CGP

Prueba estructural de caja blanca	**Código caso de prueba: CENIA_SGU_S_DCP_CGP**
Probador: Lenier Garcia Vizcaino.	

Código al que se aplica:

```
public function registrarGrupoPermiso($grupo_permiso) {
    if ($this->input->is_post_back('grupo_permiso')) {
        $verificar = $this->grupo_permiso_lib->verificarGrupoPermiso($grupo_permiso);
        if (empty($verificar)) {
            if ($this->grupo_permiso_lib->registrarGrupoPermiso($grupo_permiso) != false) {
                return "El elemento ha sido creado satisfactoriamente.";
            } else {
                return "Ocurrió un error durante la operación. Intente más tarde.";
            }
        } else {
            return "El elemento ya existe.";
        }
    } else {
        return "Uno o varios elementos se ha introducido de forma incorrecta.";
    }
}
```

Complejidad ciclomática:

V (G) = (A - N) + 2 = (10 - 8) + 2 = 4

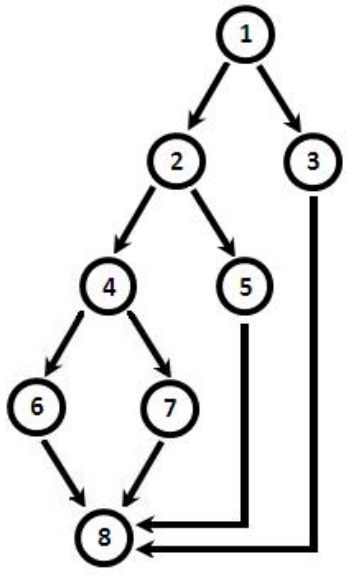
Caminos independientes

1) 1 – 3 – 8

2) 1 – 2 – 5 – 8

3) 1 – 2 – 4 – 6 – 8

4) 1 – 2 – 4 – 7 – 8



Caso de prueba para el camino básico 1	
Descripción: los datos de entrada serán los atributos del grupo de permisos a crear.	
Condición de ejecución: los datos de entrada no son válidos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo vacío.
Tipo de dato esperado:	is_string

Función de evaluación: $resultadoEsperado="Uno o varios elementos se ha introducido de forma incorrecta." $nombrePrueba="Prueba:'Crear Grupo Permisos '"; echo $this->ejecutarPrueba($this->grupo_permiso->registrarGrupoPermiso($post_vars),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	Satisfactoria
Caso de prueba para el camino básico 2	
Descripción: los datos de entrada serán los atributos del grupo de permisos a crear.	
Condición de ejecución: los datos de entrada están previamente insertados en la base de datos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos del grupo de permisos insertados por el usuario.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="El elemento ya existe." $nombrePrueba="Prueba:'Crear Grupo Permisos '"; echo $this->ejecutarPrueba($this->grupo_permiso->registrarGrupoPermiso($post_vars),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 3	
Descripción: los datos de entrada serán los atributos del grupo de permisos a crear.	
Condición de ejecución: los datos de entrada son válidos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos del grupo de permisos insertados por el usuario.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="El elemento ha sido creado satisfactoriamente." $nombrePrueba="Prueba:'Crear Grupo Permisos '"; echo $this->ejecutarPrueba($this->grupo_permiso->registrarGrupoPermiso($post_vars),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 4	
Descripción: los datos de entrada serán los atributos del grupo de permisos a crear.	
Condición de ejecución: ocurre un error a la hora de insertar los datos en la base de datos.	
Procedimiento prueba automatizada	

Datos de entrada:	arreglo con los datos del grupo de permisos insertados por el usuario.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="Ocurrió un error durante la operación. Intente más tarde." $nombrePrueba="Prueba:'Crear Grupo Permisos '"; echo $this->ejecutarPrueba($this->grupo_permiso->registrarGrupoPermiso($post_vars),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba: Satisfactorio	
Resultado final de la prueba: Prueba satisfactoria.	

Tabla 39. CENIA_SGU_S_DCP_LLP

Prueba estructural de caja blanca	**Código caso de prueba: CENIA_SGU_S_DCP_LLP.**
Probador: Lenier Garcia Vizcaino	
Código al que se aplica:	

```
public function llenarPermisos($modulo) {
    $_ci = &get_instance();
    $permisos=array(array());
    if ($this->modulo == "" || $this->modulo != $modulo) {
        $this->modulo = "$modulo";
        $token = $_ci->session->userdata('token');
        $user_data = $_ci->ioc->seguridad->obtenerElementosDadoToken($token);
        $usuario = $user_data['id_usuario'];
        $modulo_permisos = $_ci->ioc->seguridad->obtenerModuloInyectar($modulo);
        $permisos = $_ci->ioc->seguridad->obtenerPermisosInyector($modulo_permisos, $usuario);
        foreach ($permisos as $perm) {
            $tablas = $this->extractTables($perm);
            $filtros = $this->prepareFilter($perm);
            for ($index = 0; $index < count($tablas); $index++) {
                $permisos[][$tablas[$index]] = $filtros[$index];
            }
        }
    }
    return $permisos;
}
```

Complejidad ciclomática:

V (G) = (A - N) + 2 = (9 - 7) + 2 = 4

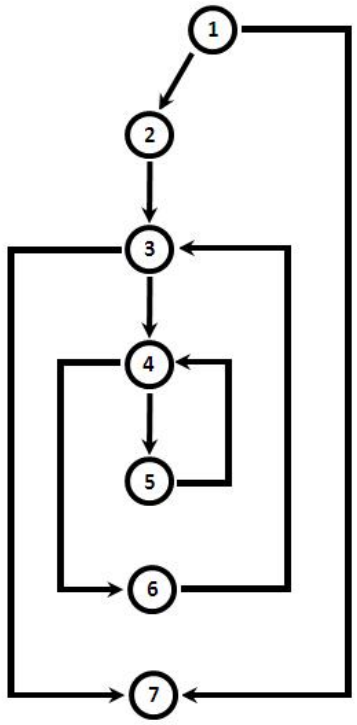
Caminos independientes

1) 1 – 7

2) 1 – 2 – 3 – 7

3) 1 – 2 – 3 – 4 – 6 – 3 – 7

4) 1 – 2 – 3 – 4 – 5 – 4 – 6 – 3 – 7



Caso de prueba para el camino básico 1	
Descripción: el dato de entrada será el módulo del sistema al que el usuario accedió.	
Condición de ejecución: el dato de entrada coincide con el almacenado en la clase.	
Procedimiento prueba automatizada	
Datos de entrada:	módulo del sistema.
Tipo de dato esperado:	is_array
Función de evaluación: $resultadoEsperado="Array" $nombrePrueba="Prueba:' Obtener Permisos'"; echo $this->ejecutarPrueba($this->sqlfiltrer->llenarPermisos($modulo),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria

Caso de prueba para el camino básico 2	
Descripción: el dato de entrada será el módulo del sistema al que el usuario accedió.	
Condición de ejecución: el dato de entrada coincide con el almacenado en la clase y no existen permisos relacionados con el módulo.	
Procedimiento prueba automatizada	
Datos de entrada:	módulo del sistema.
Tipo de dato esperado:	is_array
Función de evaluación:	

$resultadoEsperado="Array" $nombrePrueba="Prueba:' Obtener Permisos'"; echo $this->ejecutarPrueba($this->sqlfiltrer->llenarPermisos($modulo),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 3	
Descripción: el dato de entrada será el módulo del sistema al que el usuario accedió.	
Condición de ejecución: el dato de entrada coincide con el almacenado en la clase, existen permisos relacionados con el módulo y no hay tablas asociadas a dichos permisos.	
Procedimiento prueba automatizada	
Datos de entrada:	módulo del sistema.
Tipo de dato esperado:	is_array
Función de evaluación: $resultadoEsperado="Array" $nombrePrueba="Prueba:' Obtener Permisos'"; echo $this->ejecutarPrueba($this->sqlfiltrer->llenarPermisos($modulo),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 4	
Descripción: el dato de entrada será el módulo del sistema al que el usuario accedió.	
Condición de ejecución: el dato de entrada coincide con el almacenado en la clase, existen permisos relacionados con el módulo y no hay tablas asociadas a dichos permisos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo vacío.
Tipo de dato esperado:	is_array
Función de evaluación: $resultadoEsperado="Array" $nombrePrueba="Prueba:' Obtener Permisos'"; echo $this->ejecutarPrueba($this->sqlfiltrer->llenarPermisos($modulo),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba: Satisfactoria.	
Resultado final de la prueba: Prueba Satisfactoria.	

Tabla 30. CENIA_SGU_S_DCP_ODU

Prueba estructural de caja blanca	**Código caso de prueba: CENIA_PRE_R_DP_ODU.**
Probador: Lenier Garcia Vizcaino	

Código al que se aplica:

```php
public function obtenerDatosDeUsuario($id_usuario) {
    $array_ids = array("id_usuario" => $id_usuario);
    $permisos = $this->_ci->tb_dpermiso_mdl->obtenerDadoAtributos($array_ids);
    if (empty($permisos))
        return "empty";
    else {
        $result = array();
        foreach ($permisos as $perm) {
            $objeto = new stdClass();
            $id_filtro = array('id_filtro' => $perm->id_filtro);
            $filtro = $this->_ci->tb_dfiltro_mdl->obtenerDadoId($id_filtro);
            $objeto->id_tipo_filtro = $filtro->id_tipo_filtro;
            $objeto->id_modulo_sistema = $filtro->id_modulo_sistema;
            $objeto->id_tipo_regla = $perm->id_tipo_regla;
            $tipo_filtro = $this->_ci->tb_ntipo_filtro_mdl->obtenerDadoId(array('id_tipo_filtro' => $filtro->id_tipo_filtro));
            $objeto->nombre_tipo_filtro = $tipo_filtro->nombre_tipo_filtro;
            $modulo = $this->_ci->tb_dmodulo_mdl->obtenerDadoId(array('id_modulo' => $filtro->id_modulo_sistema));
            $objeto->etiqueta_modulo = $modulo->etiqueta_modulo;
            $cadena_objetos = $perm->objetos;
            $cadena_objetos = str_replace('{', '', $cadena_objetos);
            $cadena_objetos = str_replace('}', '', $cadena_objetos);
            $cadena_objetos = str_replace('\'', '', $cadena_objetos);
            $objectos_asoc = explode(',', $cadena_objetos);
            $objeto->objetos_asociados = $objectos_asoc;
            array_push($result, $objeto);
        }
        return $result;
    }
}
```

Complejidad ciclomática:

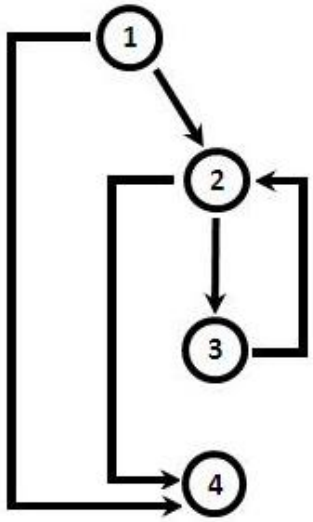
V (G) = (A - N) + 2 = (5 -4) + 2 = 3

Caminos independientes

1) 1 – 4

2) 1 – 2 – 4

3) 1 – 2 – 3 – 2 – 4



Caso de prueba para el camino básico 1

Descripción: el dato de entrada será una variable con el identificador del usuario.	
Condición de ejecución: el dato de entrada no tiene asociados permisos en el sistema.	
Procedimiento prueba automatizada	
Datos de entrada:	identificador del usuario.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="empty" $nombrePrueba="Prueba:'Obtener Permisos Usuario'"; echo $this->ejecutarPrueba($this->reglas_negocio_lib->obtenerDatosUsuario($id_usuario),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 2	
Descripción: el dato de entrada será una variable con el identificador del usuario.	
Condición de ejecución: el dato de entrada tiene asociados permisos en el sistema y no contienen información.	
Procedimiento prueba automatizada	
Datos de entrada:	identificador del usuario.
Tipo de dato esperado:	is_array
Función de evaluación: $resultadoEsperado="Array" $nombrePrueba="Prueba:'Obtener Permisos Usuario'"; echo $this->ejecutarPrueba($this->reglas_negocio_lib->obtenerDatosUsuario($id_usuario),$resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 3	
Descripción: el dato de entrada será una variable con el identificador del usuario.	
Condición de ejecución: el dato de entrada tiene asociados permisos en el sistema.	
Procedimiento prueba automatizada	
Datos de entrada:	identificador del usuario.
Tipo de dato esperado:	is_array
Función de evaluación: $resultadoEsperado="Array" $nombrePrueba="Prueba:'Obtener Permisos Usuario'"; echo $this->ejecutarPrueba($this->reglas_negocio_lib->obtenerDatosUsuario($id_usuario),$resultadoEsperado,$nombrePrueba);	

Evaluación del caso de prueba: Satisfactoria
Resultado final de la prueba: Prueba Satisfactoria

Anexo 9. Pasos a realizar para aplicar la prueba del camino básico

- **Paso 1. Representar el programa en un grafo de flujo**

El grafo de flujo se utiliza para representar flujo de control lógico de un programa. Para ello se utilizan los tres elementos siguientes:

- ✓ Nodos: representan cero, una o varias sentencias en secuencia. Cada nodo comprende como máximo una sentencia de decisión (bifurcación).
- ✓ Aristas: líneas que unen dos nodos.
- ✓ Regiones: áreas delimitadas por aristas y nodos. Cuando se contabilizan las regiones de un programa debe incluirse el área externa como una región más.
- ✓ Nodos predicados: cuando en una condición aparecen uno o más operadores lógicos (AND, OR, XOR) se crea un nodo distinto por cada una de las condiciones simples. Cada nodo generado de esta forma se denomina nodo predicado.

Cada construcción lógica de un programa tiene una representación. La siguiente figura muestra dichas representaciones.

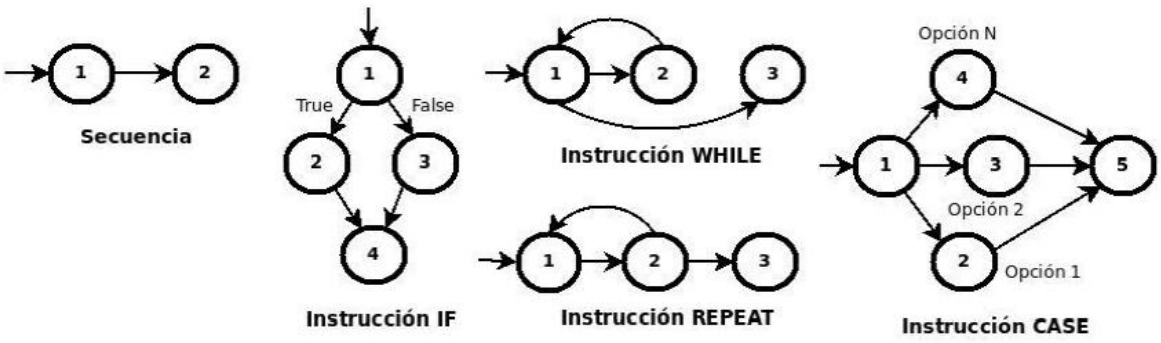


Figura 24. Representación en grafo del flujo de las estructuras lógicas de un programa.

- **Paso 2. Calcular la complejidad ciclomática**

La complejidad ciclomática es una métrica del *software* que proporciona una medida cuantitativa de la complejidad lógica de un programa. En el contexto del método de prueba del camino básico, el valor de la complejidad ciclomática define el número de caminos independientes de dicho programa, y por lo tanto, el

número de casos de prueba a realizar. Primero se mostrará cómo calcular ciclomática, a partir de un grafo de flujo, para obtener el número de caminos a identificar. Posteriormente se demostrará cómo se identifican esos caminos.

Existen varias formas de calcular la complejidad ciclomática de un programa a partir de un grafo de flujo:

- ✓ El número de regiones cerradas del grafo (r) coincide con la complejidad ciclomática, V (G) = *r*.
- ✓ La complejidad ciclomática, V (G), de un grafo de flujo G se define como: *V (G) = aristas – nodos + 2.*
- ✓ La complejidad ciclomática, V (G), de un grafo de flujo G se define como: *V (G) = nodos predicado +1.*

La complejidad ciclomática determina el número de casos de prueba que deben ejecutarse para garantizar que todas las sentencias de un programa se han ejecutado al menos una vez, y que cada condición se habrá ejecutado en sus vertientes verdadera y falsa.

- ➢ **Paso 3. Determinar el conjunto básico de caminos**

Un camino independiente es cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una condición, respecto a los caminos existentes. En términos del diagrama de flujo, un camino independiente está constituido por lo menos por una arista que no haya sido recorrida antes de la definición del camino. En la identificación de los distintos caminos de un programa para probar se debe tener en cuenta que cada nuevo camino debe tener el mínimo número de sentencias nuevas o condiciones nuevas respecto a los que ya existen. De esta manera, se intenta que el proceso de depuración sea más sencillo. El conjunto de caminos independientes de un grafo no es único. No obstante, a continuación se muestran algunas heurísticas para identificar dichos caminos:

- ✓ Elegir un camino principal que represente una función válida que no sea un tratamiento de error. Debe intentar elegirse el camino que atraviese el máximo número de decisiones en el grafo.
- ✓ Identificar el segundo camino mediante la localización de la primera decisión en el camino de la línea básica alternando su resultado mientras se mantiene el máximo número de decisiones originales del camino inicial.
- ✓ Identificar un tercer camino, colocando la primera decisión en su valor original a la vez que se altera la segunda decisión del camino básico, mientras se intenta mantener el resto de decisiones originales.
- ✓ Continuar el proceso hasta haber conseguido tratar todas las decisiones, intentando mantener como en su origen el resto de ellas.

Anexo 10. Diseño de casos de prueba de integración

Tabla 31. Casos de prueba de integración con el módulo Seguridad

Caso de Prueba: Int1_S
Módulo al que se integra: Seguridad
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas a este módulo. Además, Debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Seguridad.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Seguridad.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Tabla 32. Casos de prueba de integración con el módulo Estructura y Composición

Caso de Prueba: Int2_EC
Módulo al que se integra: Estructura y Composición
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas al módulo Estructura y Composición. Además, Debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Estructura y Composición.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Estructura y Composición.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Tabla 33. Casos de prueba de integración con el módulo Configuración

Caso de Prueba: Int3_Cf
Módulo al que se integra: Configuración
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas al módulo Configuración. Además, debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Configuración.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Configuración.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Tabla 34. Casos de prueba de integración con el módulo Eventos

Caso de Prueba: Int5_E
Módulo al que se integra: Eventos
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas al módulo Eventos. Además, debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Eventos.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Eventos.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Tabla 35. Casos de prueba de integración con el módulo Documentos Acreditativos

Caso de Prueba: Int6_DA
Módulo al que se integra: Documentos Acreditativos
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas al módulo Documentos Acreditativos. Además, debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Documentos Acreditativos.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Documentos Acreditativos.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Tabla 36. Casos de prueba de integración con el módulo Inmuebles

Caso de Prueba: Int7_I
Módulo al que se integra: Inmuebles
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas al módulo Inmuebles. Además, debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Inmuebles.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Inmuebles.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Tabla 37. Casos de prueba de integración con el módulo Carrera

Caso de Prueba: Int8_C
Módulo al que se integra: Carrera
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas al módulo Carrera. Además, debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Carrera.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Carrera.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Tabla 38. Casos de prueba de integración con el módulo Personal y Secretaría

Caso de Prueba: Int9_PS
Módulo al que se integra: Personal y Secretaría
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas al módulo Personal y Secretaría. Además, debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Personal y Secretaría.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Personal y Secretaría.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Tabla 39. Casos de prueba de integración con el módulo Control Docente

Caso de Prueba: Int10_CD
Módulo al que se integra: Control Docente
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas al módulo Control Docente. Además, debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Personal y Secretaría.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Control Docente.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Tabla 40. Casos de prueba de integración con el módulo Estudiante

Caso de Prueba: Int11_E
Módulo al que se integra: Estudiante
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas al módulo Estudiante. Además, debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Estudiante.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Estudiante.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Tabla 41. Casos de prueba de integración con el módulo Reportes

Caso de Prueba: Int12_R
Módulo al que se integra: Reportes
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas al módulo Reportes. Además, debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Reportes.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Reportes.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Tabla 42. Casos de prueba de integración con el módulo Tesis y Título

Caso de Prueba: Int13_TT
Módulo al que se integra: Tesis y Título
Condición de ejecución: En la agrupación funcional “Seguridad de negocio” del módulo Seguridad deben existir categorías de filtro asociadas al módulo Tesis y Título. Además, debe existir conexión con la base de datos central.
Descripción de las pruebas: Comprobar que la agrupación funcional “Seguridad de negocio” es capaz de gestionar los permisos con la información manejada en el módulo Tesis y Título.
Entrada/Pasos de ejecución: La agrupación funcional “Seguridad de negocio” introduce en la base de datos las categorías de filtro asociadas al módulo Tesis y Título.
Resultado esperado: Al usuario acceder al sistema se muestra únicamente la información gestionada a través de los permisos que le son otorgados.
Evaluación: Prueba satisfactoria.

Anexo 11. Diseño de casos de pruebas basados en requisitos

Debido la extensión de los diseños de casos de pruebas, en este anexo se muestran solo algunos ejemplos. Para ver la totalidad de casos de pruebas realizados se puede consultar el expediente del Proyecto Pregrado.

Para una mejor comprensión de los mismos es preciso hacer las siguientes aclaraciones respecto a los campos presentes en la tabla.

Descripción de los Campos:

Condiciones de ejecución: Contiene como su nombre lo indica todas las condiciones que deben existir para que el caso de prueba sea ejecutado sin dificultad.

SC Nombre del requisito a probar: corresponde al nombre que se le da al requisito en el documento de "*Especificación de requisitos"*.

Escenario: en este campo se nombran los diferentes escenarios a probar donde cada uno estará compuesto por las siglas *EC* y una numeración consecutiva *1.1… 1.n + el nombre del escenario.*

Descripción: corresponde como dice su nombre a la descripción del escenario a probar.

Variable 1…n: este campo va a estar compuesto por dos partes: la primera podrá tomar los valores *V, I y NA* (válido, inválido y no aplica respectivamente), la segunda estará conformada por un ejemplo según el escenario que se pretende realizar.

Respuesta del sistema: se escribe el resultado que se espera al realizar la prueba.

Flujo central: corresponde a los pasos a desarrollar para probar la funcionalidad que se indicó.

Tabla 43. DCP_Crear Categoria Filtro

Condiciones de ejecución: El usuario debe de estar autenticado en el sistema con el rol de administrador.													
SC Crear categoría de filtro													
Escenario	**Descripción**	**Variable 1. Nombre**	**Variable 2. Nombre físico**	**Variable 3. Descripción**	**Variable 4 Estado**	**Variable 5. Tipo visual**	**Variable 6 Aplicación**	**Variable 7. Módulo**	**Variable 8 Tabla**	**Variable 9. Campo nombre**	**Variable 10. Campo descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Insertar datos correctamente	Este escenario permite crear una categoría de filtro insertando los datos correctamente.	V	V	N/A	V	V	V	V	V	V	V	El sistema muestra el mensaje: El elemento ha sido creado satisfactoriamente.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos flotantes se selecciona la opción de "Crear". El usuario introduce los datos correctamente y presiona el botón Aceptar.
		Estructuras	estructura	Pequeña descripción de la categoría de filtro	Habilitado	Filtrable	Sistema	Estructura y Composición	tb_natributo_estructura	nombre_atributo_estructura	valor_atributo_estructura		
EC 1.2 Insertar	Mediante este	I	I	N/A	I	I	I	I	I	I	I	El sistema muestra	El usuario una vez autenticado en el Sistema de

datos repetidos.	escenario se pretende insertar una nueva categoría de filtro utilizando datos existentes en el sistema.	Estructuras	estructura	Pequeña descripción de la categoría de filtro.	Habilitado	Filtrable	Sistema	Estructura y Composición	tb_natributo_estructura	nombre_atributo_estructura	valor_atributo_estructura	el mensaje: El elemento ya existe.	Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos flotantes se selecciona la opción de "Crear". El usuario introduce los datos incorrectamente y presiona el botón Aceptar.
EC 1.3 Insertar datos incompletos.	Mediante este escenario se dejan campos requeridos sin llenar para crear una categoría de filtro.	I	I	N/A	V	V	V	V	V	V	V	El sistema muestra en rojo el mensaje: “Campo requerido” sobre el campo que debe ser llenado de forma obligatoria.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos flotantes se selecciona la opción de "Crear". El usuario introduce los datos incorrectamente y presiona el botón Aceptar.
		(Vacío)	(Vacío)	Pequeña descripción de la categoría de filtro.	Habilitado	Filtrable	Sistema	Estructura y Composición	tb_natributo_estructura	nombre_atributo_estructura	valor_atributo_estructura		
EC 1.4 Insertar datos incorrectos.	Mediante este escenario se introducen datos incorrectos para crear una categoría de filtro.	I	I	N/A	V	V	N/A	N/A	N/A	N/A	N/A	El sistema muestra el mensaje de error: “Entre al menos 2 caracteres” sobre el campo requerido.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos flotantes se selecciona la opción de "Crear". El usuario introduce los datos incorrectamente y presiona el botón Aceptar.
		El usuario introduce menos de 2 caracteres.	El usuario introduce menos de 2 caracteres.	_	Habilitado	Filtrable	_	_	_	_	_		
		I	I	N/A	V	V	N/A	N/A	N/A	N/A	N/A	El sistema muestra en rojo el mensaje: “No se admite apóstrofes ni caracteres especiales” sobre el campo requerido.	
		"fgc%' a'"	"fgc%' a'"	Pequeña descripción de la categoría de filtro.	Habilitado	Filtrable	_	_	_	_	_		
		I	I	I	V	V	N/A	N/A	N/A	N/A	N/A		

		El usuario introduce más de 100 caracteres.	El usuario introduce más de 100 caracteres.	El usuario introduce más de 100 caracteres.	Habilitado	Filtrable	_	_	_	_	_	El sistema no permite seguir escribiendo.	
		V	V	I	V	V	N/A	N/A	N/A	N/A	N/A		
		Dominio	dominio	El usuario introduce más de 30 caracteres por palabras.	Habilitado	Filtrable	_	_	_	_	_	El sistema muestra el mensaje: "Ha excedido el número de letras permitidas para una palabra".	
EC 1.5 Cancelar operación	Mediante este escenario se cancela la operación de crear una categoría de filtro.	I	NA	NA	NA	NA	NA	NA	NA	NA	NA		El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos flotantes se selecciona la opción de "Crear". El usuario introduce los datos incorrectamente y presiona el botón Cancelar.
		fgc%' a'"	_	_	_	_	_	_	_	_	-	El sistema muestra el mensaje de confirmación ¿Está seguro de realizar la acción?	

Tabla 44 DCP_Listar Categoria Filtro

Condiciones de ejecución: El usuario debe de estar autenticado en el sistema con el rol de administrador.						
SC Listar categoría de filtro						
Escenario	**Descripción**	**Variable 1. Nombre Categoría**	**Variable 2. Cantidad por página**	**Encabezado del grid**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Listar datos correctamente.	Mediante este escenario se mostrará al usuario un listado con todas las categorías de filtro existentes en el sistema. En el listado se muestra en el área de íconos internos las acciones a desarrollar sobre cada elemento: Ver detalles y Modificar.	N/A	N/A	N/A		El usuario una vez autenticado en el sistema selecciona el "Sistema de Gestión Académica de Pregrado" y luego el módulo "Tesis y Títulos". El sistema muestra las opciones de menú y el usuario selecciona en la agrupación funcional "Configuración" la
					El sistema muestra el listado de	

	Además, se muestra en el área de flotantes las acciones: Crear y Ayuda.				los elementos actualizados.	funcionalidad "Agente de negocio".
EC 1.2 Seleccionar correctamente cantidad de elementos por página.	Mediante este escenario el usuario escoge la cantidad de elementos a mostrar por página (los valores a escoger son 5, 10, 15 y 20).	N/A	V	N/A	El sistema muestra la cantidad de elementos según la opción escogida (Se puede escoger 5, 10, 15,20).	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes.
			El usuario escoge la opción de 5.			
		N/A	V	N/A		
			El usuario escoge la opción de 10.			
		N/A	V	N/A		
			El usuario escoge la opción de 15.			
		N/A	V	N/A		
			El usuario escoge la opción de 20.			
EC 1.3 Ordenar elementos correctamente.	Mediante este escenario se da clic en el encabezado del listado ordenándolo así alfabéticamente (Icono que se muestra como una flecha).	N/A	N/A	V		El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes.
				Se da clic encima del elemento y se activa la flecha de arriba.	El sistema ordena los elementos de forma ascendente. Ordena primero los habilitados y luego los deshabilitados.	
		N/A	N/A	V		
				Se da clic encima del elemento y se activa la flecha de abajo.	El sistema ordena los elementos de forma descendente. Ordena primero los habilitados y luego los deshabilitados.	
EC 1.4 No existen elementos	Mediante este escenario, en caso de que no exista ninguna	N/A	N/A	N/A		El usuario una vez autenticado en el Sistema de Gestión

creados.	categoría de filtro creada se muestra un listado vacío.				El sistema muestra el listado vacío.	Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes.

Tabla 45. DCP_Ver Detalles Categoria Filtro

Condiciones de ejecución: El usuario debe de estar autenticado en el sistema con el rol de administrador.			
SC Ver detalles de categoría			
Escenario	**Descripción**	**Respuesta del sistema**	
EC 1.1 Ver detalles de categoría de filtro	Mediante este escenario se muestra toda la información de la categoría de filtro seleccionada.	El sistema muestra una ventana emergente con los siguientes datos: Nombre categoría, Nombre físico, Estado (Habilitado/ Deshabilitado), Descripción.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos internos selecciona la opción: "Ver detalles".
EC 1.2 Cerrar la ventana de Ver detalles por el botón Cerrar	Mediante este escenario se puede cerrar la ventana de Ver detalles.	El sistema cierra la ventana y muestra el listado con los elementos existentes.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos internos selecciona la opción: "Ver detalles".

Tabla 46. DCP_Modificar Categoria Filtro

Condiciones de ejecución: El usuario debe de estar autenticado en el sistema con el rol de administrador.								
SC Modificar categoría de filtro.								
Escenario	**Descripción**	**Variable 1. Nombre**	**Variable 2. Nombre físico**	**Variable 3. Descripción**	**Variable 4. Estado**	**Variable 5. Tipo visual**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Modificar datos correctamente	Este escenario permite modificar correctamente una categoría de filtro.	V	V	N/A	V	V	El sistema muestra el mensaje: El elemento ha sido modificado satisfactoriamente.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos internos selecciona la opción: "Modificar". El usuario introduce elementos y presiona el botón Guardar.
		Estructuras	estructura	Pequeña descripción de la categoría de filtro	Habilitado	No filtrable		

EC 1.2 Modificar insertando elementos repetidos.	Mediante este escenario se pretende modificar una nueva categoría de filtro utilizando datos existentes en el sistema.	I	I	N/A	V	V	El sistema muestra el mensaje: El elemento ya existe.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos internos selecciona la opción: "Modificar". El usuario introduce elementos y presiona el botón Guardar.
		Estructuras	estructura	Pequeña descripción de la categoría de filtro	Habilitado	No filtrable		
EC 1.3 Modificar insertando datos incompletos.	Mediante este escenario no se introducen todos los datos para modificar un agente de negocio correctamente.	I	I	N/A	V	V	El sistema muestra en rojo el mensaje: “Campo requerido” sobre el campo que debe ser llenado de forma obligatoria.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos internos selecciona la opción: "Modificar". El usuario introduce elementos y presiona el botón Guardar.
		(Vacío)	(Vacío)	Pequeña descripción de la categoría de filtro	Habilitado	No filtrable		
EC 1.4 Modificar insertando datos incorrectos.	Mediante este escenario se introducen datos incorrectos para modificar una categoría de filtro.	I	I	N/A	V	V	El sistema muestra el mensaje de error: “Entre al menos 2 caracteres.” sobre el campo requerido.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos internos selecciona la opción: "Modificar". El usuario introduce elementos y presiona el botón Guardar.
		El usuario introduce menos de 2 caracteres.	El usuario introduce menos de 2 caracteres.	_	Deshabilitado	Filtrable		
		I	I	V	V	V	El sistema muestra en rojo el mensaje: “No se admite apóstrofes ni caracteres especiales” sobre el campo requerido.	
		"fgc%' a'"	"fgc%' a'"	"fgc%' a'"	Deshabilitado	Filtrable		
		I	I	I	V	V	El sistema no permite seguir escribiendo.	
		El usuario introduce más de 100 caracteres.	El usuario introduce más de 100 caracteres.	El usuario introduce más de 100 caracteres.	Deshabilitado	Filtrable		
		V	V	I	V	V	El sistema muestra el mensaje: "Ha excedido el número de letras permitidas	
		Dominio	dominio	El usuario introduce más de 30 caracteres por palabras.	Deshabilitado	Filtrable		

							para una palabra"	
EC 1.5 Cancelar operación	Mediante este escenario se cancela la operación de modificar una categoría de filtro.	V Dominio	V dominio	N/A –	V Deshabilitado	V Filtrable	El sistema muestra el mensaje de confirmación ¿Está seguro de realizar la acción?	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Categorías de filtros." El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos internos selecciona la opción: "Modificar". El usuario introduce elementos y presiona el botón Cancelar.

Tabla 47. DCP_Crear Permiso

Condiciones de ejecución: El usuario debe de estar autenticado en el sistema con el rol de administrador.							
SC Crear permiso							
Escenario	**Descripción**	**Variable 1 Aplicación**	**Variable 2. Módulo**	**Variable 3. Categoría de filtro**	**Variable 4. Tipo de regla**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Insertar datos correctamente	Este escenario permite crear un permiso insertando los datos correctamente.	V Sistema	V Seguridad	V Asignaturas	V Incluyente	El sistema muestra el mensaje: El elemento ha sido creado satisfactoriamente.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Reglas de negocio". En el listado que muestra el sistema con todos los usuarios registrados, se selecciona en el área de íconos internos la opción: Crear permiso. El usuario introduce los datos correctamente y presiona el botón Aceptar.
EC 1.2 Insertar datos incompletos.	Mediante este escenario se dejan campos requeridos sin llenar para crear un permiso.	I (Vacío)	I (Vacío)	I (Vacío)	I (Vacío)	El sistema muestra en rojo el mensaje: “Campo requerido” sobre el campo que debe ser llenado de forma obligatoria.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Reglas de negocio". En el listado que muestra el sistema con todos los usuarios registrados, se selecciona en el área de íconos internos la opción: Crear permiso. El usuario introduce los datos incompletos y presiona el botón Aceptar.
EC 1.3 Cancelar	Mediante este escenario se	V	V	V	V	El sistema muestra el mensaje de	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y

operación	cancela la operación de crear un permiso por el botón Canelar.	Sistema	Seguridad	Asignaturas	Excluyente	confirmación: Se perderán los datos que no han sido guardados ¿Desea realmente realizar la acción?	luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Reglas de negocio". En el listado que muestra el sistema con todos los usuarios registrados, se selecciona en el área de íconos internos la opción: Crear permiso. El usuario introduce los datos y presiona el botón Cancelar.

Tabla 48. DCP_Listar Permiso

Escenario	**Descripción**	**Variable 1. Nombre**	**Variable 2. Cantidad por página**	**Variable 3 Encabezado del grid 1**	**Variable 4 Encabezado del grid 2**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Listar datos correctamente.	Mediante este escenario se mostrará al usuario un listado con todos los permisos que tiene asociado. En el listado se muestra en el área de íconos internos las acciones a desarrollar sobre cada elemento: Ver detalles y Modificar.	N/A	N/A	N/A	N/A	El sistema muestra el listado de los elementos actualizados.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad Reglas de Negocio. Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se selecciona la opción: Crear permiso y en el área de íconos flotantes se ejecuta la acción Listar. Se muestra un listado con los permisos asociados.
		–	–	–	–		
		V	N/A	N/A	N/A		
		Control Docente	–	–	–		
EC 1.2 Seleccionar correctamente cantidad de elementos por página.	Mediante este escenario el usuario escoge la cantidad de elementos a mostrar por página (los valores a escoger son 5, 10, 15 y 20).	N/A	V	N/A	N/A	El sistema muestra la cantidad de elementos según la opción escogida (Se puede escoger 5, 10, 15,20).	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad Reglas de Negocio. Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se selecciona la opción: Crear permiso y en el área de íconos flotantes se ejecuta la acción Listar. Se muestra un listado con los permisos asociados.
		–	El usuario escoge la opción de 5.	_	–		
		N/A	V		N/A		
		–	El usuario escoge la opción de 10.	_	–		
		N/A	V	N/A	N/A		
		–	El usuario escoge la opción de 15.	_	–		
		N/A	V	N/A	N/A		
		–	El usuario escoge la opción de 20.	_	–		
EC 1.3 Ordenar los datos	Mediante este escenario se da clic en el encabezado del	N/A	N/A	N/A	V	El sistema ordena los	El usuario una vez autenticado en el Sistema de Gestión

correctamente.	listado ordenándolo así alfabéticamente (Icono que se muestra como una flecha).	_	_	_	Se da clic encima del elemento y se activa la flecha de arriba.	elementos de forma ascendente.	Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad Reglas de Negocio. Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se selecciona la opción: Crear permiso y en el área de íconos flotantes se ejecuta la acción Listar. Se muestra un listado con los permisos asociados.
		N/A	N/A	V	N/A		
		_	_	Se da clic encima del elemento y se activa la flecha de arriba.	_		
		N/A	N/A	N/A	V	El sistema ordena los elementos de forma descendente.	
		_	_	_	Se da clic encima del elemento y se activa la flecha de abajo.		
		N/A	N/A	V	N/A		
		_	_	Se da clic encima del elemento y se activa la flecha de abajo.	_		
EC 1.4 No existen elementos creados.	Mediante este escenario, en caso de que no existan permisos asociados se muestra un listado vacío.	N/A	N/A	N/A	N/A		El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad Reglas de Negocio. Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se selecciona la opción: Crear permiso y en el área de íconos flotantes se ejecuta la acción Listar. Se muestra un listado con los permisos asociados.
		_	_	_	_	El sistema muestra el listado vacío.	

Tabla 49. DCP_Ver Detalles de Permiso

Condiciones de ejecución: El usuario debe de estar autenticado en el sistema con el rol de administrador.			
SC <Ver detalles del permiso>			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Ver Detalles del permiso.	Mediante este escenario se muestra toda la información del permiso seleccionado.	El sistema muestra una ventana emergente con la información correspondiente del permiso seleccionado.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Reglas de negocio". Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos

			internos se selecciona la opción: Crear permiso. Seguidamente se selecciona en el área de íconos flotantes la opción Listar. Luego de ejecutar la acción el sistema muestra los permisos asociados y en el área de íconos internos se selecciona la opción: Ver Detalles. El sistema muestra un listado con todas las categorías de filtros existentes hasta el momento. A continuación en el área de iconos internos selecciona la opción: "Ver detalles". El sistema cierra la ventana y muestra el listado con los elementos existentes.
EC 1.2 Cerrar la ventana de Cerrar.	Mediante este escenario se puede cerrar la ventana de Ver detalles.	El sistema cierra la ventana y muestra el listado con los elementos existentes.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Reglas de negocio". Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se selecciona la opción: Crear permiso. Seguidamente se selecciona en el área de íconos flotantes la opción Listar. Luego de ejecutar la acción el sistema muestra los permisos asociados y en el área de íconos internos se selecciona la opción: Ver Detalles.

Tabla 50. DCP_Modificar Permiso

Escenario	Descripción	Variable 1. Elementos del Grid 1	Variable 2. Elementos del Grid 2	Respuesta del sistema	Flujo central
EC 1.1 Modificar datos correctamente	Este escenario permite modificar un permiso eliminando un elemento por el ícono Eliminar.	N/A	V	El sistema muestra el mensaje de información: El elemento ha sido modificado satisfactoriamente.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Reglas de negocio". Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se selecciona la opción: Crear permiso. Luego de selecciona en el área de íconos flotantes la opción Listar. El sistema muestra un listado con los permisos asociados y en el área de íconos internos se selecciona la opción: Modificar. El usuario modifica el permiso añadiendo o eliminando alguno de los elementos asociados y presiona el botón Guardar.
		_	1101 (elemento a eliminar)		
	Este escenario permite modificar un permiso añadiendo un elemento por el ícono Adicionar elemento.	V	N/A		
		1104 (elemento a adicionar)	_		
EC 1.2 Modificar datos incorrectamente	El usuario no elimina ningún elemento y presiona el botón Guardar.	N/A	N/A	El sistema muestra un mensaje en rojo con la siguiente información: Debe agregar o eliminar al menos un elemento.	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Reglas de negocio". Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se selecciona la opción: Crear permiso. Luego de selecciona en el área de íconos flotantes la opción Listar. El sistema muestra un listado con los permisos asociados y en el área de íconos internos se selecciona la opción: Modificar. El usuario presiona el botón Guardar.
		_	_		

	El usuario no adiciona ningún elemento y presiona el botón Guardar.	N/A –	N/A –		
EC 1.3 Cancelar operación	Mediante este escenario se cancela la operación de modificar un permiso.	V -	V -	El sistema muestra el mensaje de confirmación: Se perderán los datos que no han sido guardados ¿Desea realmente realizar la acción?	El usuario una vez autenticado en el Sistema de Gestión Universitaria, selecciona el módulo "Seguridad" y luego en el menú lateral, en la agrupación funcional "Seguridad de negocio" selecciona la funcionalidad "Reglas de negocio". Se muestra un listado con todas las personas registradas en el sistema. En el área de íconos internos se selecciona la opción: Crear permiso. Luego de selecciona en el área de íconos flotantes la opción Listar. El sistema muestra un listado con los permisos asociados y en el área de íconos internos se selecciona la opción: Modificar. El usuario modifica el permiso añadiendo o eliminando alguno de los elementos asociados y presiona el botón Cancelar.

Anexo 12. Pasos para diseñar los casos de pruebas según la técnica: Partición de equivalencia

- **Paso 1: Identificar las clases de equivalencia.**

Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Las clases de equivalencia se identifican examinando cada condición de entrada (normalmente una frase en la especificación) y dividiéndola en dos o más grupos. Se definen dos tipos de clases de equivalencia, las clases de equivalencia válidas, que representan entradas válidas al programa, y las clases de equivalencia no válidas, que representan valores de entrada erróneos.

Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

- ✓ Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
- ✓ Si una condición de entrada requiere un *valor* específico, se define una clase de equivalencia válida y dos no válidas.
- ✓ Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida.

Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida.

- **Paso 2: Identificar los casos de prueba.**

El objetivo de este paso es minimizar el número de casos pruebas, así cada caso de prueba debe considerar tantas condiciones de entrada como sea posible. No obstante, es necesario realizar con cierto cuidado los casos de prueba, de manera que no se enmascaren faltas. Para crear los casos de prueba a partir de las clases de equivalencia se han de seguir los siguientes pasos:

- ✓ Asignar a cada clase de equivalencia un número único.
- ✓ Hasta que todas las clases de equivalencia hayan sido cubiertas por los casos de prueba, se tratará de escribir un caso que cubra tantas clases válidas no incorporadas como sea posible.
- ✓ Hasta que todas las clases de equivalencia no válidas hayan sido cubiertas por casos de prueba, escribir un caso para cubrir una única clase no válida no cubierta.

La razón de cubrir con casos individuales las clases no válidas es que ciertos controles de entrada pueden enmascarar o invalidar otros controles similares. Por ejemplo, si tenemos dos clases válidas: *"introducir cantidad entre 1 y 99"* y *"seguir con letra entre A y Z"*, el caso *"105 1"* (dos errores) puede dar como resultado *105 fuera de rango de cantidad*, y no examina el resto de las entradas no comprobando así la respuesta del sistema ante una posible entrada no válida.