Universidad de las Ciencias Informáticas

Facultad 1

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

“Solución para informatizar el proceso de baja en la Universidad de las Ciencias Informáticas”

Autor: Edwin Ruiz Llanes

Tutores: Lic. Elizabet González Alemán

Ing. Dairo Roberto Gil Martín

Ing. Arlennys S. Velázquez Hidalgo

La Habana, Cuba
Junio de 2014

MINISTERIO DE EDUCACIÓN SUPERIOR

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Declaramos ser autores del presente trabajo de diploma y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Autorizamos a dicho centro para que haga el uso que estime pertinente con este trabajo.

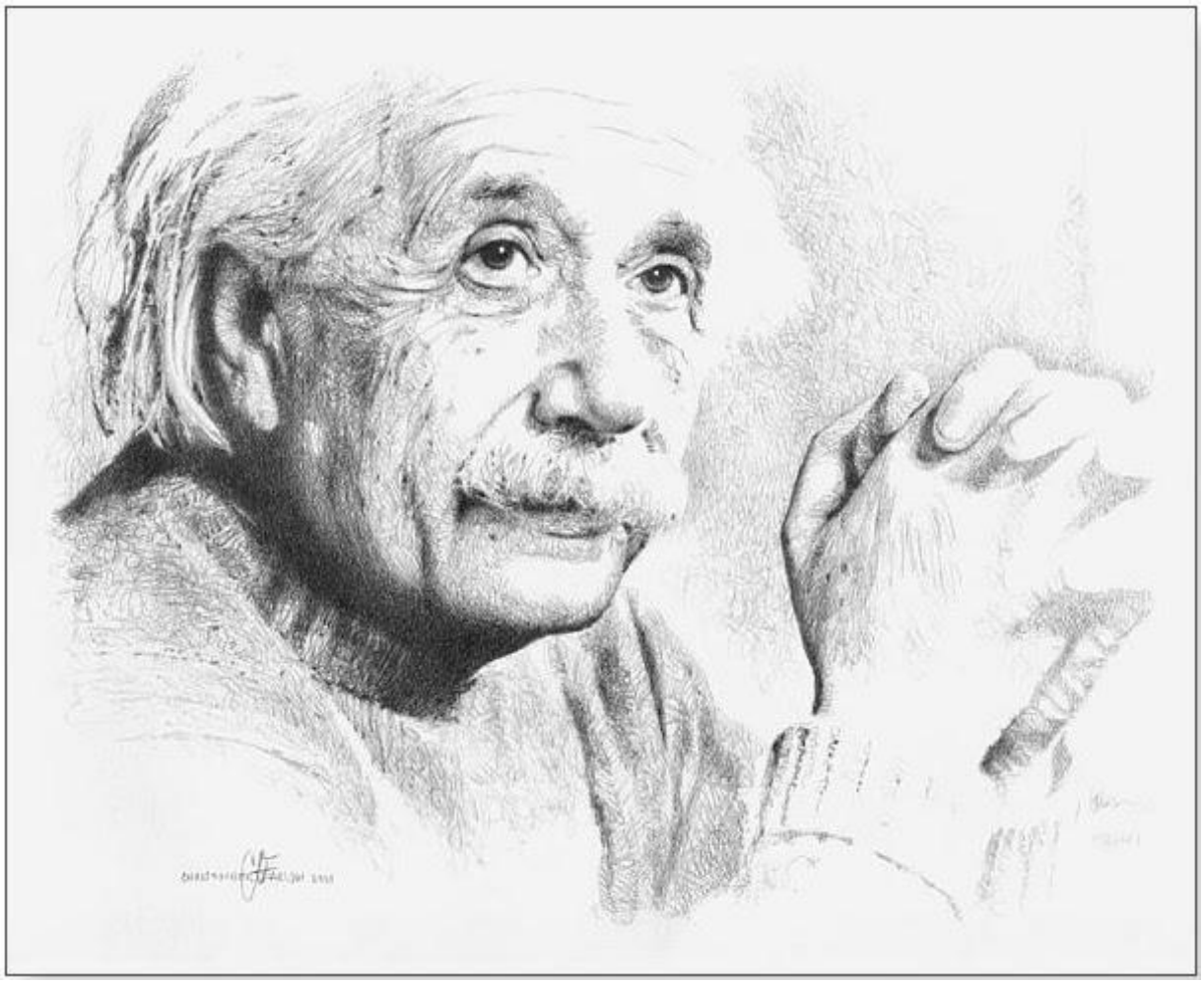
Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Edwin Ruiz Llanes

_____

Firma del Autor

Lic. Elizabet González Alemán	Ing. Dairo R. Gil Martín	Ing. Arlennys S. Velázquez Hidalgo
_____	_____	_____
Firma del Tutor	Firma del Tutor	Firma del Tutor



“El genio se hace con 1% de talento, y un 99% de trabajo”

Agradecimiento

Quisiera comenzar agradeciendo a la persona más importante de mi vida: mi mamá, gracias mami por todo. Agradecer a mi familia por todo el apoyo brindado en especial a mi papá, a mi hermano y a mi mujer. A todos aquellos que de una forma u otra me ayudaron a llegar hasta aquí: mis amigos, mis tutores, mis profesores. A todos muchas gracias.

Dedico

A mi mamá por ser la persona que siempre ha estado conmigo dándome fuerzas para seguir adelante y apoyándome en todo momento. A mi papá por como dice él, darme este cuerpo que tengo y ayudarme. A mi hermano por estar siempre ahí. A mis abuelos que siempre me ayudaron y aconsejaron. A mi mujer por apoyarme, ayudarme y guiarme.

RESUMEN

Incorporar nuevos procesos al Sistema de Gestión Universitaria para que este sistema constituya la puerta de entrada a todos los procesos de la universidad, constituye un reto diario para el equipo que desarrolla esta solución.

Uno de los procesos no cubierto por este sistema es el proceso de baja de la institución, conocido comúnmente como “Andar UCI” y cuyo objetivo radica en garantizar la devolución de los recursos, así como el cierre de los servicios que fueron puestos a disposición del personal cuando entró a la universidad.

Este proceso a pesar de estar definido y estandarizado presenta inconvenientes en su ejecución que provocan molestias a los involucrados, que la información que se genera esté dispersa y sea poco fiable; afectando la calidad y veracidad de la misma. Por lo que el objetivo de la presente investigación es desarrollar una funcionalidad dentro del módulo Personal, que informatice el proceso de baja, centralice la información asociada a dicho proceso, logre erradicar las deficiencias actuales en la gestión de las bajas y optimice la gestión de estos trámites.

Para la elaboración de dicha propuesta, él proceso fue guiado por la metodología DAC y su construcción se basó en herramientas y tecnologías libres como: *NetBeans, PostgreSQL y Visual Paradigm*.

Palabras clave: proceso de baja, gestión de información, gestión de baja.

INTRODUCCIÓN

El desarrollo acelerado de la informática así como las posibilidades que ofrecen las tecnologías de la información y las comunicaciones (TIC) hacen que las instituciones, en especial las universidades, la consideren un elemento clave para su evolución y desarrollo.

Como parte de esta evolución y como un desafío importante surge un nuevo paradigma de universidad, la Universidad Digital, donde las TIC juegan un papel trascendental en el soporte de sus procesos puesto que garantiza la actualización del conocimiento más rápido, mejora su capacidad innovadora y eleva la calidad de sus servicios (Laviña Orueta , y otros, 2010).

La Universidad de las Ciencias Informáticas (UCI), en su esfuerzo por llegar a ser una Universidad Digital, dispone valiosos recursos para el desarrollo de una solución informática que cubra todo el proceso de gestión universitaria. Muestra palpable de este esfuerzo lo constituye el Sistema de Gestión Universitaria (SGU), desarrollado por la Dirección de Informatización de la UCI y puesto a disposición de sus usuarios desde hace varios años y perfeccionándose paulatinamente en aras de ser el punto de entrada a todos los procesos y servicios de la universidad.

Uno de los procesos no cubierto por este sistema lo constituye el proceso de baja de la institución, conocido comúnmente como “Andar UCI” y cuyo objetivo radica en garantizar la devolución de los recursos así como el cierre de los servicios que fueron puestos a disposición del personal cuando entró a la universidad.

Actualmente en la universidad se tramitan alrededor de 70 u 80 bajas mensuales y la forma de realizar este proceso presenta ineficiencias en todas sus aristas, desde la persona como actor principal del proceso que tiene que hacer recorridos por todas las áreas, hasta la institución al no contar con la información suficiente para la toma de decisiones.

A continuación se recogen los principales inconvenientes que afectan la calidad del proceso de baja de la institución visto desde todas las aristas del proceso:

- La obligación por parte de la persona que abandona la universidad de presentarse a todas las áreas, aún cuando no le fue asignado ningún recurso, lo cual provoca colas y lentitud en el proceso sobre todo en períodos de fin de curso. No se hace distinción entre estudiantes, trabajadores internos, externos, terceros, etc.

- Desconocimiento por parte del solicitante de la dependencia entre las áreas, que imponen un orden en su recorrido. Esto unido a la mala planificación del horario de atención de cada una de las áreas, hacen que por lo general la persona debe presentarse en más de una ocasión en el área causando molestias y tardanza en el trámite.

- La demora en la realización de todo el proceso provoca afectación en el índice de asistencia a clases, producto que los estudiantes una vez iniciado el proceso no asisten a las clases afectando de esta forma los indicadores de la universidad.

- La información de las diferentes áreas se gestiona en formato duro y de forma diferente lo que puede provocar su deterioro y/o pérdida.

- Los servicios asociados a la persona como son: el servicio de alimentación, servicios telemáticos, servicio de acceso a la institución, entre otros se mantienen disponibles en el tiempo a pesar de que el proceso esté concluido, lo cual trae consigo que la información la mayoría de las veces no es fiable y de pie a actos de delito y corrupción dentro de la universidad.

- Dispersión de la información puesto que cada área maneja la información de forma independiente lo que impide contar con una vista real del estado del trámite.

- No disponibilidad de los datos en tiempo real lo cual complejiza su análisis y afecta la toma de decisiones oportunas por parte de los directivos de la institución.

Todos estos elementos evidencian la ineficiencia del proceso al no contar con una solución que le brinde soporte y que sea capaz de registrar la información requerida de las solicitudes de baja y sus diferentes estados ocasionando que el conjunto de actividades involucradas en el proceso se realice de forma engorrosa y poco confiable.

A partir de la situación antes descrita se define como **problema de investigación:** El proceso de baja de la institución se realiza de forma ineficiente a pesar de estar definido y estandarizado.

Para dar solución al problema planteado se propone como **objeto de estudio** el proceso de baja en las instituciones enmarcado el **campo de acción** en el sistema de gestión del proceso de baja institucional en la Universidad de las Ciencias Informáticas.

Se define como **objetivo general:** Desarrollar una solución informática capaz de controlar el proceso de baja en la Universidad de las Ciencias Informáticas que garantice la disponibilidad y fiabilidad de la información que se maneja.

Durante la investigación se defiende la **idea** que con el desarrollo de una solución informática que automatice y centralice la información referente al proceso de baja de la institución, se favorecerá la gestión de este proceso en la Universidad de las Ciencias Informáticas.

Para darle cumplimiento al objetivo planteado se definen las siguientes **tareas de investigación**:

- Caracterización del proceso de baja en la UCI.
- Análisis de los principales sistemas informáticos relacionados con el proceso de baja en la institución.
- Análisis de las tecnologías a emplear en el proceso de desarrollo del software.
- Caracterización de las herramientas que se utilizarán en el entorno de desarrollo.
- Análisis y diseño del sistema para la gestión del proceso de baja en la UCI.
- Modelado de la base de datos.
- Definición de las pruebas a realizar a la solución.
- Implementación de los requerimientos identificados en el análisis.
- Validación de las funcionalidades del sistema.

Durante la investigación se emplearon los siguientes métodos científicos con el propósito de dar cumplimiento a estas tareas.

Métodos Teóricos

Como método teórico se utiliza el método **Histórico-Lógico** para realizar un estudio del estado del arte referente al proceso de baja de la institución así como de los sistemas similares desarrollados que puedan aportar al desarrollo de esta investigación. Se utiliza el método **Analítico-sintético** para

analizar la información referente al proceso de baja, con el objetivo de facilitar su comprensión y arribar a conclusiones válidas de la investigación y el método **Sistémico** con el objetivo de revisar cada una de las relaciones que se establecen dentro del proceso de baja de la institución y formular una propuesta que unifique los diversos elementos.

Métodos Empíricos

Como método empírico se utiliza la **Entrevista** para obtener opiniones y criterios referentes al proceso de baja de la institución y conocer las diferentes alternativas empleadas para dar solución al problema, así como las normas que rigen dicho proceso. También se realiza la **Revisión de documentos** al revisar la bibliografía existente y estudiar los documentos vinculados al proceso que definen las actividades a realizar además del flujo de información en cada paso.

La presente investigación cuenta con tres capítulos de los cuales se muestra una breve descripción.

Capítulo 1. Fundamentación teórica

Se presentan los elementos teóricos que sirven de base a la investigación del problema planteado, analizando los principales conceptos relacionados con el objeto de estudio. Se realiza un análisis del proceso de baja en otras instituciones educaciones así como de soluciones existentes que puedan aportar a la propuesta de solución. Finalmente se presentan las herramientas y metodologías a utilizar para el desarrollo de la propuesta de solución.

Capítulo 2. Análisis y diseño del sistema

Se especifican los principales elementos que caracterizan la propuesta de solución, entre ellos los procesos de negocio, los requisitos y principales funcionalidades de la solución así como su incorporación, como un módulo, dentro del SGU.

Capítulo 3. Implementación y pruebas

Se describen las técnicas y estándares de codificación empleados para la construcción de la propuesta además de los resultados obtenidos durante la ejecución de las pruebas para verificar que el sistema cumple con los requisitos establecidos.

CAPÍTULO 1. FUNDAMENTACIÓN TÉORICA

1.1 Introducción

En el presente capítulo se describen los elementos teóricos que forman parte de la investigación y que están relacionados con el proceso de baja en las instituciones. Se analizan los principales conceptos relacionados con el objeto de estudio y se describen las características de los lenguajes, tecnologías y herramientas a utilizar durante el proceso de desarrollo. Finalmente se realiza un estudio de los principales sistemas que cubren, dentro de sus funcionalidades, el proceso de baja en las instituciones.

1.2 Conceptos asociados al dominio del problema

Para lograr un entendimiento del tema resulta imprescindible apropiarse de los conceptos fundamentales relacionados con el objeto de estudio, los cuales son abordados en el presente epígrafe.

Proceso: Conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados (ISO, 2000).
Baja: Cese de una persona en una institución, profesión o carrera.
Proceso de baja: Conjunto de actividades que establecen el flujo para la tramitación de la baja en la institución. Este proceso es conocido en la UCI como “Andar UCI”.
Solicitud de baja: Solicitud que realiza la persona que se dispone a abandonar la institución al jefe administrativo.
Estado de solicitud: Son las diferentes fases por las que debe transitar una solicitud de baja. Estas pueden ser: iniciada, aceptada, en trámite y finalizada.
Áreas de baja: Relación de áreas, direcciones u oficinas por las que debe transitar la persona que solicita la baja con el objetivo de devolver los recursos que le fueron asignados durante su estancia en la universidad.
Tipo de baja: Clasificación según el tipo de persona; estudiante, trabajador interno, etc. y que define un comportamiento diferente dentro de la solución.
Causa de la baja: Motivo por el cual se solicita la baja del centro.

1.3 Estudio del estado del arte

A partir del problema planteado se realizó un estudio de varios sistemas que cubren el proceso de baja en las instituciones con el objetivo de validar si se ajustan a las necesidades de nuestra institución o evaluar los elementos positivos que cada uno pudiera aportar a la propuesta de solución.

Como parte del estudio se realizó un análisis de cómo se maneja este proceso en diversas universidades con el propósito de contrastar las actividades definidas y lograr que la propuesta cubra y se ajuste en la medida de lo posible a cualquier escenario educacional.

1.3.1 Análisis de la definición del proceso de baja de la institución.

Universidades Cubanas

Para el estudio y análisis del proceso de baja de la institución se tomaron de muestra: la Universidad de La Habana (UH), el Instituto Superior Politécnico “José Antonio Echeverría” (CUJAE), la Universidad Central “Martha Abreu” de Las Villas (UCLV) y la Universidad de las Ciencias Informáticas (UCI), arrojando como resultado que el proceso establecido es básicamente igual en todas las universidades y solo difieren en los niveles de aprobación de la solicitud. A continuación se describe el proceso establecido en la UH y en la UCI puesto que el resto queda cubierto en estas dos variantes presentadas.

En la UH se lleva un control de todos los procedimientos y actividades que se desarrollan en la misma. El proceso de dar baja a estudiantes y trabajadores es uno de los más importantes para la institución. Este proceso comienza cuando el interesado solicita su baja al jefe inmediato superior. Recibida esta solicitud, la administración confecciona el “Modelo de Propuesta de Movimiento de Personal” debidamente aprobado por la máxima autoridad del área. La administración del área hace entrega en la Dirección de Recursos Humanos (DRH) de la solicitud de baja, la cual es revisada y certificada por un especialista, dándose la aprobación para que se inicie el proceso. Luego, el director de la DRH firma el “Modelo de Propuesta de Movimiento de Personal”, como conformidad del mismo, y lo envía al rector para su aprobación. Al contar con la aprobación del Rector, la DRH le informa al área. La administración del área universitaria le entrega al solicitante de la baja, el “Modelo de No Adeudo”, el cual debe ser llenado con las firmas y cuños correspondientes, avalando que el interesado no presenta deudas con la institución. Luego, el documento es entregado por las administraciones de cada área al especialista del Grupo de Relaciones Laborales (GRL) que atiende el área en la DRH. Finalmente se procede a dar baja al solicitante por parte de la DRH (Humanos, 2010).

En la UCI, el proceso de baja de la institución se inicia cuando la persona que desea abandonar la universidad hace la solicitud a su jefe inmediato superior y este a su vez lo informa al director del área. El director del área da su aprobación y su secretaria entrega al solicitante el "Modelo de Tramitación de Baja" para iniciar el recorrido por diferentes áreas que ya están preestablecidas; dígase Residencia, Biblioteca, Servicios Telemáticos, etc. con las que la persona puede tener alguna deuda. En cada una de las áreas se realiza una revisión para comprobar si la persona tiene asignado algún recurso, en cuyo caso, después de realizar la devolución el jefe de área coloca su firma confirmando que la persona ya no tiene deudas con el área. En caso de pérdida la persona debe pagar dicho recurso. Para esto el área de Contabilidad y Finanzas elabora un comprobante con la relación de medios que debe pagar cuyo dinero es depositado en la caja. Una vez concluido el proceso de recolección de las firmas, la persona presenta el documento a la Dirección de Capital Humano para dar inicio al proceso de término de la relación entre la institución y la persona que abandona la universidad (Institucional, 2013).

Universidades Extranjeras

Análogamente se realizó un análisis del proceso en universidades extranjeras el cual dió como resultado que prácticamente ninguna de las universidades incorpora este proceso como parte de sus actividades, puesto que a sus miembros no se les asigna ningún recurso sino que los mismos son contratados a terceros o facturados (pagados) previo a su utilización.

No obstante se identificó que la Universidad de Cuauhtémoc, en México, Asociación Civil dedicada exclusivamente a la formación y superación de estudiantes de nivel superior (Universidad Cuauhtemoc, 2014) tiene establecido dentro de sus actividades un proceso similar. En dicha universidad se lleva a cabo un seguimiento de todas las personas que son dadas de baja de la institución. El proceso se inicia cuando el estudiante solicita su baja y comienza el llenado del formulario de Solicitud de baja con sus datos personales y las causas por las que solicita su baja. Después de llenado el formulario debe ser impreso y firmado por los responsables en cada una de las áreas involucradas. Luego es escaneado y enviado vía correo electrónico a la persona encargada de tramitar la baja. El tiempo total de realización del proceso depende del tiempo de respuesta de los departamentos involucrados: Rectoría, Cobranza, Biblioteca y Educación a Distancia. Todo el proceso se realiza a través de internet y en ocasiones los estudiantes deben pagar una cuota para poder ser dados de baja de la universidad (Universidad Cuauhtemoc, 2014).

Aspectos positivos de los procesos estudiados.

Entre las características identificadas, en los procesos estudiados, que aportan al desarrollo de la solución del problema se encuentran:

- Sistema en línea para la realización de todo el proceso.
- Facilidad para la configuración de las áreas de baja, teniendo en cuenta que pueden variar a lo largo del tiempo.
- Notificación a todos los involucrados de cada una de las actividades completadas.
- Visualizar el estado del trámite para supervisar y controlar el proceso.

Todos los procesos estudiados aportaron valor a la investigación pues contribuyeron a formalizar una propuesta genérica al problema planteado.

1.3.2 Sistemas homólogos.

Para complementar la propuesta se realizó un estudio de varios sistemas que pudiesen aportar elementos a la solución. A pesar de realizar una búsqueda exhaustiva de sistemas desarrollados internacionalmente no se identificó ninguno que añadiera valor al desarrollo de esta investigación, no obstante se valoraron dos sistemas institucionalizados en universidades cubanas cuyos elementos se exponen a continuación.

Sistema de Gestión de la Nueva Universidad (SIGENU)

SIGENU es un sistema que surge a solicitud de la dirección del Ministerio de Educación Superior de Cuba (MES), que automatiza los procesos fundamentales de la gestión académica de una Institución de Educación Superior en todas sus modalidades de estudio. Está compuesto por varios módulos que gestionan toda la información de un estudiante desde que se matricula hasta que se gradúa o causa baja.

Este sistema registra la información de los estudiantes durante toda su carrera. Asociado al proceso de baja implementa funcionalidades que permiten dar seguimiento y control al proceso de los estudiantes y realiza distinción entre bajas definitivas, bajas temporales, licencias, entre otros (CUJAE, 2011).

El sistema utiliza el modelo cliente-servidor cuyo cliente es una aplicación de escritorio haciendo del cliente una aplicación más robusta. Está desarrollado en Java y utiliza PostgreSQL como servidor de base de datos.

ASSETS

ASSETS es un sistema de gestión integral que permite el control de los procesos de compras, ventas, producción, taller, inventario, finanzas, contabilidad, presupuesto, activos fijos, útiles y herramientas y recursos humanos. Proporciona opciones de seguridad que permiten limitar el acceso a los diferentes procesos del sistema de acuerdo con el perfil de cada usuario. Facilita el uso de la parametrización para adaptarse a las exigencias de cada entidad (ASSETS NS, 2014) garantizando que los reportes tengan la forma y el contenido que el usuario le defina.

El módulo de Recursos Humanos implementa funcionalidades que cubren el proceso de baja pero centradas, fundamentalmente, en aquellas actividades relacionadas con el término de la relación entre el trabajador y la institución.

Es una aplicación de escritorio, desarrollada bajo la filosofía cliente-servidor que utiliza como lenjuaje de programación Visual Basic, SQL Server como sistema gestor de base de datos y Crystal Reports para la generación de reportes; elementos que obligan que su utilización sea en estaciones con sistema operativo Windows. Se distribuye por licencias que requieren una cuota anual para extender su uso.

Beneficios de los sistemas estudiados.

Los sistemas estudiados a pesar de que no implementan el proceso de baja a cabalidad aportan elementos a la solución del problema planteado, entre los beneficios que brinda se encuentran:

- La forma organizada de registrar, controlar y gestionar los procesos dentro de los sistemas.
- Total accesibilidad a la información manejada referente al proceso.
- Trazabilidad de todas las acciones realizadas en el sistema.
- Seguridad en todos los niveles.

1.4 Metodología a utilizar en el desarrollo de la solución.

La solución al problema planteado se concibe como un módulo dentro del SGU por lo que para el desarrollo de la solución se asume la metodología establecida por la DIN para garantizar la coherencia de todos los artefactos que se desarrollan como parte de la solución.

La metodología a utilizar se nombra Metodología DAC (Desarrollo Ágil con Calidad) la cual se detalla a continuación.

1.4.1 Desarrollo Ágil con Calidad (DAC)

La metodología DAC (Desarrollo Ágil con Calidad) es un proceso de desarrollo de software que combina las metas y prácticas de las áreas de procesos del nivel 2 de CMMI (Capability Maturity Model Integration) con las buenas prácticas de la dirección y desarrollo ágil de proyectos de software. Es un proceso colaborativo, recursivo, iterativo, incremental y guiado por procesos y requisitos. Su modelo del proceso es una adaptación del modelo en cascada a los modelos Programación Extrema y Desarrollo Concurrente. Está enfocado a proyectos pequeños o proyectos grandes divididos en sub-proyectos que desarrollan software de gestión basado en componentes (Méndez, 2013).

Este proceso tiene 8 actividades dentro del marco de trabajo llamadas Fases o Procesos del Ciclo de Vida: Inicio, Análisis y Diseño Arquitectónico, Requisitos, Construcción, Cierre de iteración (opcional), Liberación, Transición y Cierre, ocurriendo las iteraciones concurrentes entre las fases de Requisitos, Construcción y Cierre de iteración. El proceso tiene además dos Áreas de Procesos de Protección: Gestión de proyectos y Soporte así como dos Fases o Procesos Horizontales cuyas tareas están presentes en varias de las fases del proceso común en forma de subprocesos: Arquitectura y Planificación.

La Figura 1 muestra el proceso establecido por esta metodología.

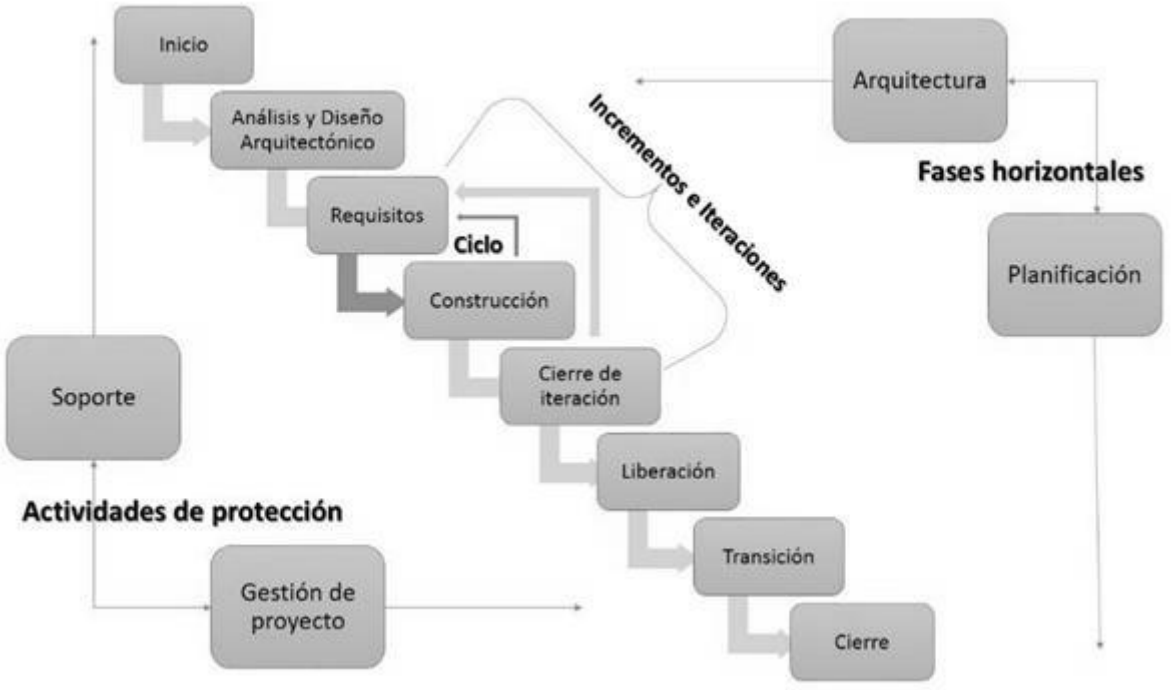


Figura 1 Modelo de Diseño DAC

El proceso DAC es un proceso recursivo (Méndez, 2013), que divide el problema en componentes de software, el desarrollo de cada componente se realizará por separado en las fases de Requisitos, Construcción y Cierre de Iteración.

Durante el desarrollo de esta investigación se utilizarán las cinco primeras fases del proceso DAC las cuales han sido especificadas por la DIN. Estas fases son:

Inicio: En la fase inicial del proyecto se llevan a cabo las actividades relacionadas con la evaluación de la factibilidad, la planeación a un alto nivel y el registro de este (Méndez, 2013). También 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. Se obtienen los requisitos de alto nivel traducidos en objetivos y se establecen los primeros acuerdos con proveedores y clientes.

Análisis y Diseño Arquitectónico: Durante esta fase es modelado el sistema para que soporte todos los requisitos de alto nivel definidos (Méndez, 2013). En ella quedan especificados los mapas

conceptuales y de procesos de la organización, el negocio o el problema, la estructuración por paquetes y componentes de la arquitectura, la priorización de los requisitos de alto nivel de impacto en la arquitectura, el entorno de desarrollo, la arquitectura de datos general, seguridad, integración, presentación y despliegue a un alto nivel.

Requisitos: Destinada primeramente a comprender los procesos de negocio de la organización para lograr obtener los requisitos del cliente a partir del estudio arquitectónico realizado en la fase anterior. Se comprende cómo funciona el negocio que se desea informatizar/automatizar. El principal objetivo es obtener la especificación de los requisitos del componente que se va a obtener en la iteración y la validación de los mismos (Méndez, 2013). Se especifican los requisitos funcionales y no funcionales a partir de las vistas de la arquitectura y los requerimientos del cliente y se diseñan los casos de prueba. Durante esta fase es modelado el sistema para que soporte todos los requisitos de la iteración actualizando en caso de ser necesario la arquitectura del software.

Construcción: En esta fase se implementa el sistema en términos de componentes (Méndez, 2013); al reutilizar componentes de software ya implementados se lleva a cabo el desarrollo necesario para ajustar los requisitos actuales y posteriormente realizar la integración de los componentes. Se crea la base de datos a partir del diseño realizado, la estructura del componte en agrupaciones funcionales y se analizan los patrones de diseño definidos en la arquitectura buscando la forma óptima de implementar los requisitos especificados.

Cierre de Iteración: Durante esta fase se desarrollan las pruebas internas a los componentes desarrollados verificando el resultado de la implementación e identificando errores tanto en la documentación como en el software.

1.5 Lenguajes y herramientas a emplear en el desarrollo de la solución

Análogamente la investigación se desarrollará utilizando las tecnologías y herramientas definidas por la DIN de manera que se asegure satisfactoriamente la inclusión de la solución dentro del SGU y pueda compartir y reutilizar todos los elementos que el SGU pone a disposición de sus módulos.

Como lenguajes para el desarrollo se utilizarán HTML v4.0, JavaScript v1.8 y PHP v5.3.6. Se empleará como marco de trabajo GUUD 1.0 y PostgreSQL como sistema gestor de base de datos.

1.5.1 Lenguajes

Nombre del Lenguaje: Lenguaje Unificado de Modelado (UML)

Descripción: Lenguaje Unificado de Modelado (UML, por sus siglas en inglés Unified Modeling Language). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo) (Jacobson, y otros, 2000), 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 reutilizables.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas, entre ellos se encuentran el “Diagrama de Clases” a través del cual se describe la estructura de un sistema (Jacobson, y otros, 2000); sus clases, atributos y las relaciones entre las clases. El “Diagrama de Componentes” que describe cómo se divide un sistema y muestra las dependencias entre estos componentes así como el “Diagrama de Secuencia” que muestra cómo los objetos se comunican entre sí en términos de una secuencia de mensajes (Jacobson, y otros, 2000).

Este lenguaje fue utilizado en la investigación para la realización de todos los diagramas, por ejemplo el modelado del proceso de solicitud de baja, facilitando el levantamiento de los requisitos y la comprensión del negocio.

Nombre del lenguaje: HTML 4
Descripción: HTML (HyperText Markup Language, Lenguaje de Enmarcado de Hipertexto), es un lenguaje diseñado para estructurar textos y es el formato más extendido para la construcción de páginas web. Es utilizado como base para crear las páginas web generadas como parte de la solución. HTML se limita a describir la estructura y el contenido de un documento, nunca el formato de la página y su apariencia, ya que éstos son muy dependientes del navegador utilizado (Pérez, 2007). Es un lenguaje portable (W3C Corporations, 2014), es decir, se pueden visualizar las páginas con cualquier sistema operativo.

Nombre del lenguaje: Hojas de Estilo en Cascada (CSS 3)
Descripción: CSS(*Cascading Style Sheets*, CSS por sus siglas en inglés), es un lenguaje para controlar la presentación de los documentos definidos con HTML (Pérez, 2007). Permite separar el contenido de su presentación y es imprescindible para la creación de páginas web. El lenguaje CSS se utiliza para definir el aspecto de todos los contenidos y para proveer estilos visuales a los elementos del documento que muestren una pauta común dentro de la solución.

Nombre del lenguaje: JavaScript 1.8

Descripción: JavaScript es un lenguaje de programación interpretado que se utiliza principalmente para crear páginas web dinámicas, con una sintaxis semejante a la del lenguaje Java y lenguaje C# (Pérez, 2009). Es un lenguaje script orientado a documento puesto que permite agregarle características a las páginas web.

Nombre del lenguaje: PHP 5.3.6
Descripción: PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor (Rosselott, 2003). Permite escribir páginas web dinámicas de una manera rápida y fácil. La característica más potente y destacable de este lenguaje es que soporta una gran cantidad de bases de datos y puede ser utilizado en cualquier sistema operativo.

1.5.2 Herramientas

Nombre de la herramienta: Suite Visual Paradigm 5.0
Descripción: Herramienta para el desarrollo de aplicaciones, utilizando UML, que permite desarrollar software de manera eficiente, rápida y de forma colaborativa (Jacobson, y otros, 2000). Soporta todas las necesidades de diseño y modelado a lo largo del ciclo de vida de desarrollo de software, es una herramienta que ayuda a construir aplicaciones de calidad, de manera más rápida, óptima y más barata. Permite la generación de bases de datos, conversión de diagramas entidad-relación a tablas de base de datos. Se caracteriza por ofrecer:

- Navegación intuitiva entre la escritura del código y su visualización.
- Estar disponible para múltiples plataformas (Windows, Linux).
- Permitir la realización de ingeniería directa e inversa.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Licencia: gratuita y comercial.

Nombre de la herramienta: Evolus Pencil 1.3.5
Descripción: Evolus Pencil es una herramienta de prototipado de código abierto que está disponible para todas las plataformas, construida con el propósito de ofrecer una herramienta de prototipado libre y de código abierto de interfaz gráfica de usuario que la gente pueda fácilmente instalar y utilizar para

crear maquetas (Evolus, 2012). Tiene entre sus características principales que los proyectos pueden ser exportados en los formatos HTML, PNG[1], documento Openoffice.org, documento de Word y PDF[2], además permite la instalación de plantillas definidas por el usuario, las operaciones de dibujo estándar: alinear, escalar y rotar y la adición de los objetos externos.

Nombre de la herramienta: Entorno de Desarrollo Integrado (IDE) NetBeans 7.4
Descripción: NetBeans es un IDE escrito en Java, de código abierto (OpenSource), gratuito para desarrolladores de software. Ofrece todas las herramientas necesarias para crear aplicaciones profesionales, empresariales, web y móviles con el lenguaje Java, JavaFX, C/C ++ y lenguajes dinámicos como PHP, JavaScript, Groovy y Ruby. Es fácil de instalar y se puede ejecutar tanto en Windows, como en Linux, Mac OS X y Solaris. La plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio de gran tamaño (Oracle Corporation, 2013). Ofrece servicios comunes permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación.

Nombre de la herramienta: PostgreSQL 8.4
Descripción: Sistema de gestión de base de datos relacional orientada a objetos y libre (Guerrero, 2013), publicado bajo la licencia BSD[3]. Incluye características de la orientación a objetos como son: la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Entre sus principales características se encuentran: permite alta concurrencia, claves foráneas (foreign keys), disparadores (triggers) y funciones.

Nombre de la herramienta: PgAdmin III 1.12.1
Descripción: Aplicación gráfica para gestionar el sistema gestor de base de datos PostgreSQL. Diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas (PostgreSQL, 2014). La interfaz gráfica soporta

[1] Portable Network Graphics, Gráficos de Red Portátiles: formato gráfico basado en un algoritmo de compresión sin pérdidas.

[2] Portable Document Format, Formato de Documento Portátil: formato de almacenamiento de documentos digitales independiente.

[3]Berkeley Software Distribution: Distribuciones de código fuente que se hicieron en la Universidad de Berkeley en California, en su origen eran extensiones del sistema operativo UNIX.

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 programados entre otros. Es una herramienta multiplataforma puesto que puede ejecutarse en disimiles sistemas operativos (Linux y Windows) (W3C Corporations, 2014). La conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas unix) y puede encriptarse mediante SSL para mayor seguridad. Además, es una de las herramientas más completa y popular con licencia Open Source (PostgreSQL, 2014).

Nombre de la herramienta: Apache 2.2.22
Descripción: Servidor de código abierto, para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP. Entre las características que más sobresalen están que es altamente configurable, con base de datos de autenticación y su amplia aceptación en la red. Es usado principalmente para enviar páginas web dinámicas en la web. Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable (Apache Software Foundation, 2014).

1.6 Marco de trabajo

Nombre:GUUD 1.0
Descripción: Marco de trabajo desarrollado por el grupo de arquitectura de la DIN para la creación de aplicaciones web escritas en PHP. Constituye un híbrido entre el framework de PHP CodeIgniter y la librería de Javascript jQuery. Implementa el patrón Modelo-Vista-Controlador (MVC) así como la programación orientada a aspectos. Realiza manejo de excepciones y mensajes. Modulariza el marco de trabajo CodeIgniter e implementa como estrategia de comunicación entre módulos la Inversión de Control (IoC) (Collazo, 2013). Hace uso de plantillas para el renderizado de las vistas. Contiene una serie de componentes visuales que permiten la fácil interacción del usuario con la aplicación, tales como el calendario y el grid, entre otros.

1.7 Conclusiones parciales

Los procesos y sistemas estudiados como parte de este capítulo aportaron elementos claves para la generalización de la solución al problema planteado. Permitió conocer la carencia de un sistema informático que cubra en su totalidad el proceso de baja en las instituciones. El estudio de las herramientas y tecnologías establecidas por la DIN, permitió profundizar los conocimientos necesarios y su validez para la implementación de la solución lo cual conlleva a utilizar PHP 5 como lenguaje de

programación, GUUD como marco de trabajo, el SGBD PostgreSQL 8.4 y el servidor web Apache 2.2.22.

CAPÍTULO 2.ANÁLISIS Y DISEÑO DELSISTEMA

2.1 Introducción

En el presente capítulo se hace una propuesta general del sistema a desarrollar, después de haber realizado un análisis crítico del proceso de negocio que tiene lugar en la universidad. Se presenta la vista de procesos a través del modelo de negocio, las reglas y requisitos funcionales y no funcionales que debe cumplir la propuesta así como las técnicas empleadas para la obtención de estos requisitos. Finalmente se describen otras vistas de la arquitectura y se presentan los principales patrones a utilizar en la construcción de la propuesta.

2.2 Modelado del negocio

Con el objetivo de proporcionar una vista simplificada del proceso de baja en la universidad y comprender los problema actuales que presentan, así como identificar las posibles mejoras, se realizó el modelado de procesos del negocio el cual, a través de la descripción de las actividades involucradas ofrece una vista de cómo se realiza el trabajo, posibilita representar y visualizar el funcionamiento del sistema, facilita su entendimiento y contribuye a su automatización y mejora (Pérez, 2005).

2.2.1 Flujo actual del proceso

Actualmente el proceso de baja en la universidad tiene el siguiente flujo, el cual puede verse además en la Figura 2.

Solicitud de baja de la universidad: La persona que se dispone a abandonar la universidad, ya sea estudiante, trabajador interno o externo, se dirige a su jefe administrativo con el objetivo de informar su decisión y pedir que se le inicie el proceso de baja.

Autorización e inicio del proceso de baja: El jefe administrativo realiza un análisis de la solicitud y de estar de acuerdo se le hace entrega a la persona de un documento en formato duro denominado “Modelo de Tramitación de Bajas”, que enmarca las áreas por las que debe transitar la persona y en las que se le autoriza la baja según corresponde.

Realización del proceso de baja por las áreas: Una vez que la persona tiene en sus manos el documento indicado para comenzar con el proceso de baja de la universidad, comienza el recorrido por cada una de las áreas (Contabilidad y Finanzas, ATM, Residencia, UJC, Biblioteca, Cultura Física,

Servicios Telemáticos, Identificación, Área de Atención Militar), con el objetivo de obtener la firma que avale que la persona devolvió todos los recursos que le fueron entregados por la universidad.

Aprobación final: Avalado por las áreas pertinentes el solicitante entrega el documento a su jefe administrativo quien firma y acuña el documento para que el mismo sea entregado en Capital Humano. En el caso de los estudiantes la secretaria docente de su facultad es la encargada de cambiar el estado de docente a baja.

Archivo: Una vez el Modelo de Tramitación de Bajas es completado la persona que desea abandonar la universidad lo presenta en el área correspondiente el cual es archivado y con su entrega da inicio al proceso de término de la relación entre la persona y la universidad. En el caso de los trabajadores en la dirección de Capital Humano y en el caso de los estudiantes en la secretaría general de la facultad.

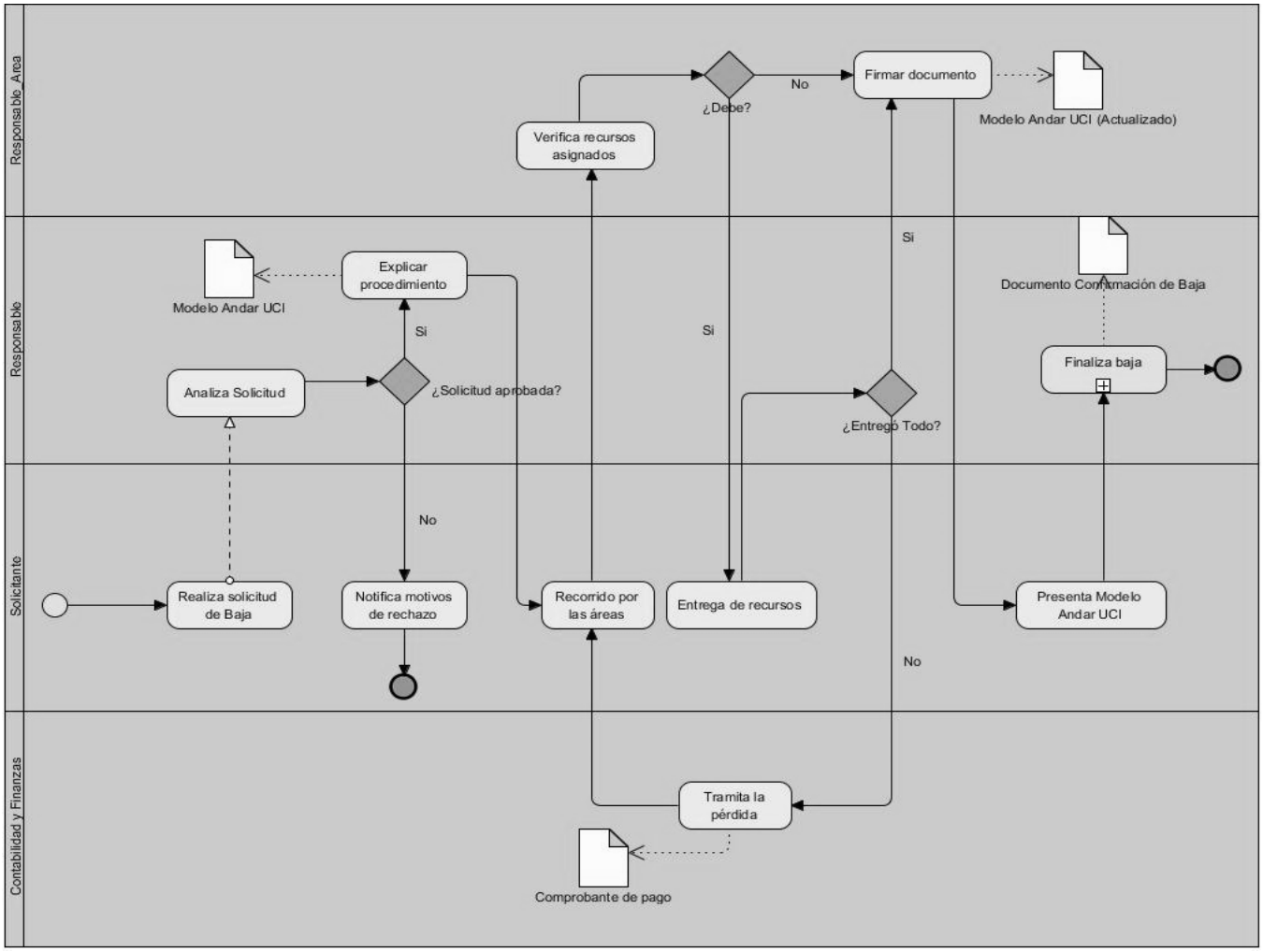


Figura 2 Modelo de procesos

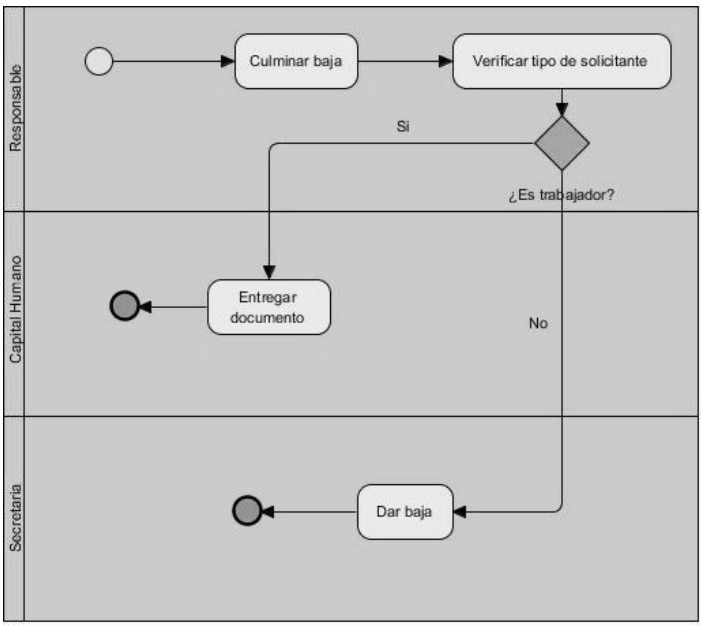


Figura 3 Modelo de subproceso

2.3 Reglas del negocio

A continuación se relacionan las reglas establecidas por el negocio las cuales describen las operaciones, normas, políticas que deben ser cumplidas (Pérez, 2005) así como las restricciones que son vitales dentro del proceso de baja de la institución.

1. El jefe administrativo de la persona es el único que puede crear una nueva solicitud de baja.
2. Una solicitud de baja solo puede ser modificada por la persona que la creó.
3. Solo puede generar el documento final de la baja la persona que la creo o el administrador del sistema.
4. Cada estado de solicitud se identifica por varios parámetros dentro de los que se encuentra el color:
   -***negro:*** utilizado para representar el estado de *iniciada*.
   -***azul:*** utilizado para representar el estado de *en trámite*.
   -***verde:*** utilizado para representar el estado de *finalizada*.
   -***rojo:*** utilizado para representar el estado de *baja*.
5. La cantidad de áreas de visita obligatoria dependerá del tipo de baja de la solicitud.
6. Cuando el solicitante tenga adeudos, las personas responsables de las áreas de baja podrán emitir las causas del adeudo y posteriormente dar baja. Estas causas serán reflejadas en el

documento que se emite al concluir el proceso.

7. Los estados de la solicitud son unidireccionales.
8. Los reportes de solicitudes solo serán accedidos por personas autorizadas.
9. Las bajas generales solo serán tramitadas por personas autorizadas.
10. Los documentos finales solo serán consultados por el personal autorizado.
11. En el proceso de notificación solo estarán involucrados el solicitante, el directivo al cual se subordina la persona que solicita la baja y las personas autorizadas por áreas de baja.
12. Solo se le mostrarán los iconos flotantes de exportar documento al usuario que esté registrado como administrador del sistema.

2.4 Técnicas de obtención de requisitos

Para la realización de este trabajo se emplearon diferentes técnicas para la obtención de los requisitos las cuales permitieron comprender el dominio del sistema, buscar y recolectar información para definir sus límites y restricciones así como identificar a las personas involucradas en el sistema (Sommerville, 2005). Las técnicas aplicadas para el desarrollo de este trabajo fueron: la entrevista y los cuestionarios, en los que con la participación de los clientes y usuarios se obtuvo una colección completa de los requerimientos del sistema.

Entrevistas: se realizaron reuniones entre analistas y cliente que permitieron entender el dominio del problema y sus necesidades. Estas se basaron en un conjunto de preguntas y respuestas, buscando obtener una comprensión general de lo que hacen las personas que interactúan con el sistema, las dificultades que presentan actualmente y las expectativas con respecto al nuevo sistema. Estas fueron realizadas tanto personal como en grupo.

Cuestionarios: se realizaron preguntas a clientes y usuarios involucrados, con el negocio recopilando opiniones y criterios relevantes para la implementación (**Ver Anexo 1**). Combinando esta técnica con la de la entrevista, se pudo obtener información para complementar una propuesta que cumpliera con las expectativas del cliente.

2.5 Definición de los requisitos funcionales y no funcionales

El proceso de descubrir, analizar, documentar y verificar las necesidades del cliente o usuario para un sistema es llamado Ingeniería de Requerimientos (RE). Este tiene como meta entregar una especificación de requisitos de software correcta y completa (Sommerville, 2005).

Teniendo en cuenta esta definición se identificaron los requerimientos a cumplir por la solución los cuales se clasificaron en requisitos funcionales y no funcionales para una mejor comprensión (Sommerville, 2005). Los siguientes epígrafes abarcan los requerimientos identificados para el desarrollo de la solución.

2.5.1 Definición de requisitos funcionales

Para la definición de los requisitos funcionales se tuvo en cuenta la definición brindada por Pressman (Pressman, 2005) donde expresa que los requisitos funcionales de un sistema son los servicios que este debe proporcionar, de la manera que debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares.

En el presente trabajo se identificaron 30 requisitos funcionales, atendiendo a las necesidades del cliente y al enfoque general de la organización. Estos fueron clasificados teniendo en cuenta su complejidad y prioridad. La siguiente tabla recoge estos requisitos.

Requisitos	Complejidad	Prioridad
1. Mostrar solicitudes de baja.	Baja	Media
2. Ver detalles de solicitud de baja.	Media	Media
3. Crear solicitud de baja.	Media	Alta
4. Modificar solicitud de baja.	Media	Media
5. Mostrar bajas pendientes por área.	Media	Media
6. Dar baja por área.	Media	Media
7. Mostrar solicitudes de bajas a finalizar.	Media	Media
8. Generar documento de baja.	Media	Media
9. Mostrar áreas de baja.	Baja	Media
10. Crear área de baja.	Baja	Media
11. Modificar área de baja.	Baja	Media
12. Ver detalles de área de baja.	Baja	Media
13. Configurar áreas de visita obligatoria.	Baja	Media
14. Mostrar estados de solicitud.	Baja	Media
15. Crear estado de solicitud.	Baja	Media
16. Modificar estado de solicitud.	Baja	Media
17. Ver detalles de estado de solicitud.	Baja	Media
18. Configurar secuencia de estado de solicitud.	Baja	Media
19. Mostrar tipos de baja.	Baja	Media
20. Crear tipo de baja.	Baja	Media

21. Modificar tipo de baja.	Baja	Media
22. Ver detalles de tipo de baja.	Baja	Media
23. Asociar áreas de baja a tipo de baja.	Media	Media
24. Mostrar causas de baja.	Baja	Media
25. Modificar causa de baja.	Baja	Media
26. Ver detalles de causa de baja.	Baja	Media
27. Crear causa de baja.	Baja	Media
28. Configurar personas autorizadas a dar baja en el área de baja.	Media	Media
29. Exponer razones.	Media	Media
30. Notificar personas.	Media	Baja

Tabla 1 Requisitos funcionales

2.5.2 Especificación de los requisitos funcionales

Con el objetivo de describir detalladamente cada requisito funcional, controlar el alcance de los objetivos definidos en el proyecto y verificar si se están cumpliendo los objetivos inicialmente trazados (Pressman, 2005) se realizó como parte de este trabajo una especificación de todos los requisitos funcionales identificados.

Para ello fueron utilizados los artefactos: “Especificación de requisitos del software”, donde se listan todos los requerimientos de la solución y una breve descripción de los mismos y “Descripción de requisitos del software”, donde se describe detalladamente cada acción que se realiza así como el comportamiento de la solución ante una funcionalidad dada. Ambos artefactos están definidos dentro de la Metodología DAC (Méndez, 2013).

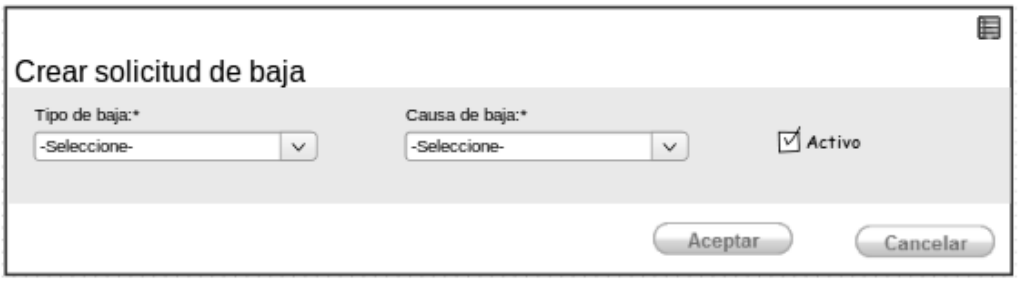
A manera de ejemplo se presenta el artefacto “Descripción de requisitos” correspondiente a uno de los requisitos funcionales más importantes del sistema (**Ver Anexo 2**).

Nº	Nombre	Descripción	Complejidad	Prioridad

RFPers3	Crear solicitud de baja.	• El requisito permite crear una solicitud de baja. El usuario con los permisos asignados selecciona del módulo Personal del SGU, la opción “Solicitudes de baja” del menú de funcionalidades “Proceso de baja”. • Se muestra un listado con las solicitudes de baja registradas en el sistema. • En el área de íconos flotantes selecciona la acción “Crear”. • Se muestra una interfaz que permite seleccionar el tipo de baja, la causa y el estado de la solicitud de baja (Activo/Inactivo). • Se permite buscar la persona a la que se le creará la solicitud por los criterios: *nombre, apellidos, CI, usuario, solapín.* • Se muestra un listado con las personas que coincidan con el criterio de búsqueda, con los datos: *Foto, CI, Nombre y apellidos.* • Se selecciona del listado la persona especificada, se	Media	Alta

		presiona el botón “Aceptar” para crear la solicitud de baja. • Se permite crear una nueva solicitud de baja o regresar al listado de solicitudes.		

Prototipo



Campos	Tipos de Datos	Reglas o Restricciones

Tipo de baja	Varchar	• Único • Selección • Obligatorio
Causa de baja	Varchar	• Único • Obligatorio
Estado	Boolean	• Selección
Buscar	Varchar	• No admite porcientos y apóstrofos • Admite 30 caracteres por palabra
Foto	Bytea	• Solo lectura
CI	Varchar	• Solo lectura
Nombre y apellidos	Varchar	• Solo lectura
	Observaciones	• El usuario con los permisos asignados debe estar autenticado en el sistema. • Si no ocurren errores debe mostrarse el mensaje de información: *"El elemento ha sido creado satisfactoriamente".* • Si ocurre un error durante la operación se muestra el mensaje: *"La persona que intenta crear una solicitud de baja no es responsable de la persona seleccionada"*. • En caso que el valor de un campo único exista se muestra un mensaje de error: *"El elemento ya existe"*. • En caso que se deje un campo de los obligatorios vacío se muestra un mensaje en rojo *"Campo requerido"* encima del campo que debe ser llenado obligatoriamente. • Los campos de texto solo admiten 30 caracteres por palabra. • La cantidad de elementos a mostrar en la lista son 5, 10,

		15 y 20 según la preferencia del usuario. • En caso de cancelar la acción se muestra un mensaje de advertencia *"¿Está seguro de realizar la acción?".* • *Ver la regla de negocio #1_Crear solicitud.*

Tabla 2 Descripción del requisito "Crear solicitud"

2.5.3 Requisitos no funcionales

En esta sección se recogen los principales atributos de calidad que el sistema debe cumplir, agrupados como requisitos no funcionales los cuales no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento (Sommerville, 2005) de la solución.

Los requisitos no funcionales que debe cumplir la propuesta de solución son:

Requisitos no funcionales	
Usabilidad	
RNF 1	Solo se mostrarán a los usuarios las acciones o informaciones a las que por su responsabilidad o rol dentro del negocio necesitan acceder.

RNF 2	Las vistas del sistema deben indicar en cada momento la acción que se está realizando así como los íconos deben estar representados por una imagen acorde a la acción que se realiza mediante el mismo.
Seguridad	
RNF 3	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, facilitando además la protección de la información.
Disponibilidad	
RNF 4	El sistema estará disponible las 24 horas del día y los siete días de la semana.
Eficiencia	
RNF 5	El sistema soportará la conexión simultánea de un promedio entre 100 y 200 usuarios.
Soporte	
RNF 6	El sistema se regirá por un estándar de código debidamente documentado en el expediente de proyecto.
Restricciones de diseño	
RNF 7	El sistema debe cumplir con la arquitectura de información definida para el SGU.
RNF 8	El sistema debe ser desarrollado con las herramientas definidas por la DIN.
Interfaz	
RNF 9	El sistema debe cumplir con la arquitectura de información definida para el SGU.
RNF 10	La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo TCP/IP.
Requisitos de licencia	
RNF 11	El sistema se caracterizará por un licenciamiento cerrado.
Requisitos legales, de derecho de autor y otros	
RNF 12	El sistema debe ser sometido a un análisis legal por parte de los abogados y personal autorizado con vistas a declarar su autenticidad y evitar restricciones legales para su uso y comercialización.

Requisitos de hardware	
RNF 13	Para el despliegue se requiere, por parte del servidor: • PC Intel Pentium 4 o superior. • CPU 3GHz o superior. • 512 MB RAM o superior. • 160 GB HDD o superior.
RNF 14	Para el uso de la solución por parte del cliente se requiere: • PC Pentium 3 o superior. • CPU 133 MHz o superior. • 256 MB RAM mínimo o superior (512 MB RAM recomendada).
Estándares aplicables	
RNF 15	Para exportar datos del sistema se usa el Formato de Documento Portátil (.pdf), norma ISO 32000.
RNF 16	Para los datos del carné de identidad, registro civil, provincia, municipio, lugar de nacimiento, entre otros, se utilizarán las normas nacionales establecidas.

Tabla 3 Requisitos no funcionales

2.5.4 Validación de los Requerimientos funcionales

Para la validación de los requisitos se emplearon diversas técnicas con el objetivo de demostrar que la propuesta cumple con los requerimientos del cliente. Los aportes de cada una de estas técnicas se recogen a continuación.

Técnicas de validación

Del conjunto de técnicas definidas por Somerville (Sommerville, 2005), las cuales aseguran la calidad del proceso de validación se emplearon como parte de este trabajo las siguientes:

Construcción de prototipos: esta práctica permitió la representación de aquellos aspectos del software que serán visibles al usuario, con el objetivo de construir una maqueta de la futura solución a partir de los requisitos recogidos en la especificación.

Generación de casos de pruebas: esta técnica permitió comprobar la validez de los requisitos funcionales mediante los diseños de casos de prueba ejecutados al sistema.

2.6 Propuesta de solución

Luego de realizado el estudio y análisis de la problemática existente en la UCI y de detectar la ausencia de un sistema capaz de registrar, controlar y centralizar la información relacionada con el proceso de baja, se propone desarrollar como parte del SGU, la funcionalidad **"Proceso de baja"** dentro del módulo **"Personal"** que aporte una solución real y satisfactoria al problema planteado.

La funcionalidad gestionará las principales actividades vinculadas al proceso como son: creación de una solicitud de baja, facilitar la baja en las áreas involucradas, configurar las áreas y personas autorizadas a dar baja así como la generación de documentos asociados. Análogamente la solución brindará opciones para configurar previamente aquellos campos del sistema que lo requieran como son: causas de baja, tipo de baja y estados de la solicitud.

La propuesta integrará un mecanismo de notificación mediante el correo electrónico permitiendo mantener informado tanto al solicitante como a su directivo y a las personas involucradas en cada una de las áreas de baja.

La definición de las áreas de visita obligatoria, están definidas dentro de la propuesta como un elemento de configuración y estarán en correspondencia con el tipo de baja. Estas áreas podrán actualizarse por el administrador del sistema siempre que se requiera. El aval que emite cada área confirmando que la persona que desea abandonar la universidad no tiene adeudos con el área se gestionará a través del sistema y la compilación de todos los avales queda plasmada en el documento final de la baja emitido por el sistema en formato pdf.

Los elementos descritos anteriormente constituyen la base de la propuesta. Adicionalmente se incorporarán funcionalidades como "Solicitudes de baja", "Bajas pendientes" y "Bajas Finalizadas" las cuales servirán de control, supervisión y apoyo al proceso. Mediante la funcionalidad "Solicitudes de baja", se proveerá de información relevante a la universidad sobre las solicitudes que han sido procesadas. Esta información será clave para la toma de decisiones y la misma estará disponible a través de consultas parametrizadas que visualizarán la información a través de reportes en formato pdf. Esta información tendrá las garantías de seguridad necesaria para que solo tengan acceso las personas definidas en el sistema.

Todas las acciones incorporadas a la solución contarán con opciones para su creación, actualización y eliminación de ser requeridos y deberá incluir elementos que permitan la trazabilidad de cada una las acciones realizadas dentro del sistema.

Una vez que el estado de la solicitud de la baja es puesto en Finalizada el sistema reportará la incidencia al resto de los sistemas involucrados y a partir de ese momento los servicios puestos a disposición de la persona que desea abandonar la universidad dejan de tener vigencia.

La solución tendrá opciones de seguridad que garanticen el nivel de acceso al sistema y sus funcionalidades teniendo en cuenta los roles establecidos dentro del sistema.

Se espera que con el desarrollo de la solución se reduzca el tiempo para la realización del proceso de baja. Se elimine la gestión de la información en formato duro. Se minimicen las molestias causadas tanto al solicitante como a las personas que brindan el servicio y se convierta en una herramienta útil e imprescindible para el seguimiento y control de todo el ciclo de vida de este proceso.

2.7 Descripción de la arquitectura

Como ya se ha mencionado la propuesta de solución se enmarca dentro del SGU, por lo que deberá ajustarse a la arquitectura definida para este sistema. Este sistema establece la arquitectura cliente-servidor y emplea para su desarrollo el patrón Modelo Vista Controlador (MVC). El marco de trabajo GUUD, que como se vio en el Capítulo 1, es una de las herramientas a utilizar para el desarrollo de esta solución e incorpora este patrón como base para el desarrollo.

La arquitectura cliente-servidor consiste básicamente en un cliente que realiza peticiones a otro programa, el servidor, que le da respuesta. En esta arquitectura coinciden una serie de aplicaciones basadas en dos categorías que ejecutan funciones desiguales, una solicita servicios y la otra los ofrece aunque pueden realizar actividades en forma conjunta o independiente.

Características del cliente:

- Es quien comienza las solicitudes o peticiones, por lo que tienen un papel activo en la comunicación.
- Aguarda y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.

Características del servidor:

- Aguarda a que lleguen las solicitudes de los clientes, por lo que realiza un papel pasivo.
- Después de la recepción de una solicitud, la procesa y le envía la respuesta al cliente.
- Permiten conexiones desde un gran número de clientes aunque puede limitarse el número de peticiones.

2.8 Patrón de arquitectura

Como parte de la arquitectura se define el patrón arquitectónico Modelo-Vista-Controlador (MVC) (Reynoso, 2004) el cual separa los datos, la interfaz de usuario y la lógica de control en tres componentes distintos. En este patrón la “Vista” constituye la interfaz que ve el usuario y el código que provee datos dinámicos a la página. El “Modelo” lo constituye el Sistema de Gestión de Base de Datos y la lógica de negocio y el “Controlador” es el responsable de recibir los eventos de entrada desde la vista.

Las responsabilidades de los elementos de este patrón son (Reynoso, 2004):

Modelo

- Representa a toda la información con la que opera la aplicación.
- Gestiona el comportamiento y los datos del dominio.
- Responde a las peticiones de información sobre el estado, que vienen de la Vista.
- Responde a instrucciones de cambio de estado, provenientes del Controlador.

Vista

- Gestiona la presentación de la información de la aplicación. Todo lo relativo a la interfaz de usuario, los datos de que dispone para seguir interactuando con la aplicación.
- Desde la interfaz gráfica a los estímulos que recibe del usuario, visual, auditiva o sensitivamente.

Controlador

- Respuesta a eventos invocados desde la Vista.
- Llama a la lógica de negocio para procesar y producir una respuesta.
- Interpreta las entradas del usuario, informando al modelo y/o a la vista de los cambios que supongan esas entradas.

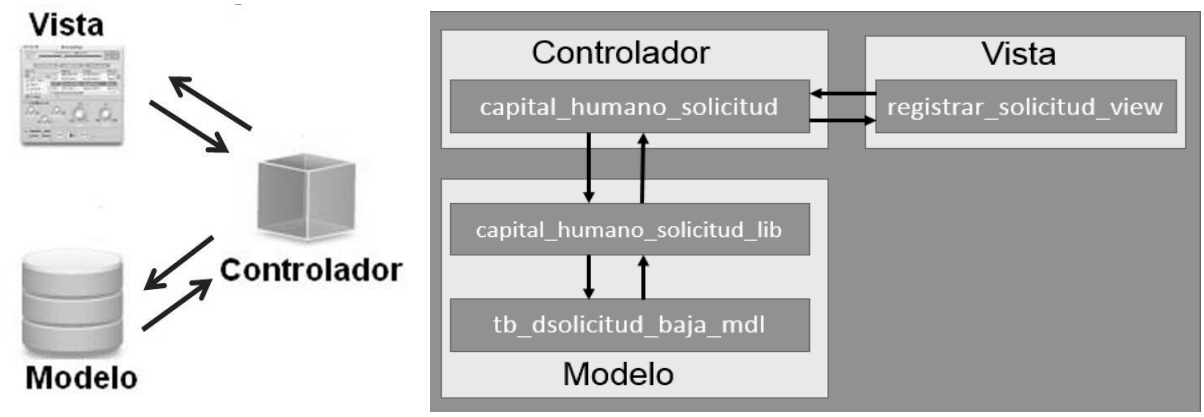


Figura 4 Modelo-Vista-Controlador y su aplicación**.**

El primer paso en el ciclo de vida del patrón **MVC** (Reynoso, 2004) comienza cuando el usuario hace una solicitud al "Controlador" con información sobre lo que el usuario desea realizar. El "Controlador" decide a quién debe delegar la tarea y es ahí donde el "Modelo" inicia su trabajo. En esta etapa, el "Modelo" se encarga de realizar operaciones sobre la información que maneja para cumplir lo que le solicita el controlador. Una vez que termina su labor, le regresa al "Controlador" la información resultante de sus operaciones el cual redirige a la "Vista". La "Vista" se encarga de transformar los datos en información visualmente entendible para el usuario. Finalmente, la representación gráfica es transmitida de regreso al "Controlador" y este se encarga de transmitírsela al usuario. El ciclo entero puede comenzar de nuevo si el usuario así lo requiere.

La aplicación del patrón MVC evidenciado en la solución se puede comprender de mejor manera en el ejemplo representado en la Figura 4. Para crear una nueva solicitud de baja se introducen los datos desde la vista *registrar_solicitud_view*, los datos son procesados a partir de la clase controladora *capital_humano_solicitud*, la cual realiza una petición a la librería *capital_humano_solicitud_lib* que es responsable de procesar los datos e implementar la lógica de negocio, esta a su vez se comunica con la clase de acceso a datos *tb_dsolicitud_baja_mdl*, realiza la petición y devuelve el resultado, a través del flujo de comunicación, en la vista correspondiente.

2.9 Patrones de diseño

Para el desarrollo de la propuesta se tuvieron en cuenta aquellos patrones de diseño que contribuyeran a reutilizar el diseño gráfico por las ventajas en cuanto a reducción de esfuerzo, mejora de la seguridad informática, eficiencia y consistencia que proporcionan. A continuación se realiza un resumen de cada uno de ellos.

2.9.1 Patrones de la "pandilla de los cuatro" (Gang-of-Four)

Los patrones de diseño GoF **(**Gang of Four, Pandilla de los Cuatro**)** llevan este nombre debido a sus creadores: Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides (Gamma , y otros, 2003) y describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Existen tres tipos de patrones: creacionales, estructurales y de comportamiento, de los cuales se emplearon para el diseño de la propuesta de solución los de creación y los de comportamiento.

Creacionales:

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia (Gamma , y otros, 2003). Este patrón se refleja en las clases controladoras que son instancias únicas y pueden ser utilizadas por las demás clases. Ejemplo: la clase controladora *capital_humano_baja_pendiente*.

Comportamiento:

Mediator (Mediador): Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto (Gamma , y otros, 2003). Se evidencia en las librerías, las cuales son mediadoras entre las clases controladoras y las modelos. Un ejemplo de ello es la librería *capital_humano_solicitud_lib* la cual es la encargada de manejar la información entre la controladora *capital_humano_solicitud* y la modelo *tb_dsolicitud_baja_mdl*.

2.9.2 GRASP: Patrones de Principios Generales para Asignar Responsabilidades

GRASP es un acrónimo de *General Responsibility Assignment Software Patterns* (patrones generales de software para asignar responsabilidades) (Gamma , y otros, 2003). Estos patrones describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades.

Experto: Asigna una responsabilidad al experto en información, o sea, la clase que cuenta con la información necesaria para cumplir la responsabilidad (Kruchten , 2000). En la solución, el uso de este patrón se evidencia en las librerías pues cuentan con la información necesaria para cumplir la responsabilidad sobre los elementos del negocio. Un ejemplo de ello es en la librería *capital_humano_solicitud_lib* en la llamada al método *existeElemento()* el cual es el encargado de verificar la autenticidad de una solicitud de baja.

Creador: el patrón creador permite identificar quién debe ser el responsable de la instanciación de nuevos objetos o clases (Kruchten , 2000). Se evidencia en las clases controladoras las cuales son las

encargadas de instanciar las librerías. Dentro de la clase controladora *capital_humano_solicitud* se instancia la librería *capital_humano_tipo_baja_lib* para obtener datos relacionados a los tipos de baja.

Controlador: Asigna la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase (Kruchten , 2000). Se evidencia en las clases controladoras que se encargan de obtener datos y enviarlos a las vistas y librerías. Una vez que la clase controladora *capital_humano_solicitud* obtiene los datos mediante la secuencia *all_post*(), enviados por la vista *registrar_solicitud_view*, envía dichos datos a la librería *capital_humano_solicitud_lib* donde se procesan y se retornan los resultados a la controladora y esta a su vez a la vista.

Bajo acoplamiento: Asigna una responsabilidad de manera que el acoplamiento permanezca bajo (Kruchten , 2000). El nivel de acoplamiento no se puede considerar de manera aislada a otros principios como el Experto o Alta Cohesión. Es un factor importante dentro del diseño de la solución.

Alta Cohesión: Asigna una responsabilidad de modo que la cohesión siga siendo alta (Kruchten , 2000). La propia implementación de CodeIgniter contiene los patrones bajo acoplamiento y alta cohesión nivelados pues permite el uso de los componentes de forma individual, evidenciando el bajo acoplamiento así como la dependencia entre ellos o alta cohesión.

2.10 Modelo físico de la Base de Datos

El modelo de datos definido a partir de esta propuesta evidencia de qué manera los datos van a ser guardados, organizados y manipulados en un sistema gestor de base de datos. Define su estructura, los datos, las dependencias y relaciones entre ellos, así como las restricciones que deben cumplir. Los modelos de datos contienen también un conjunto de operaciones básicas para la realización de consultas y actualizaciones de estos datos. La Figura 5 muestra dicho modelo.

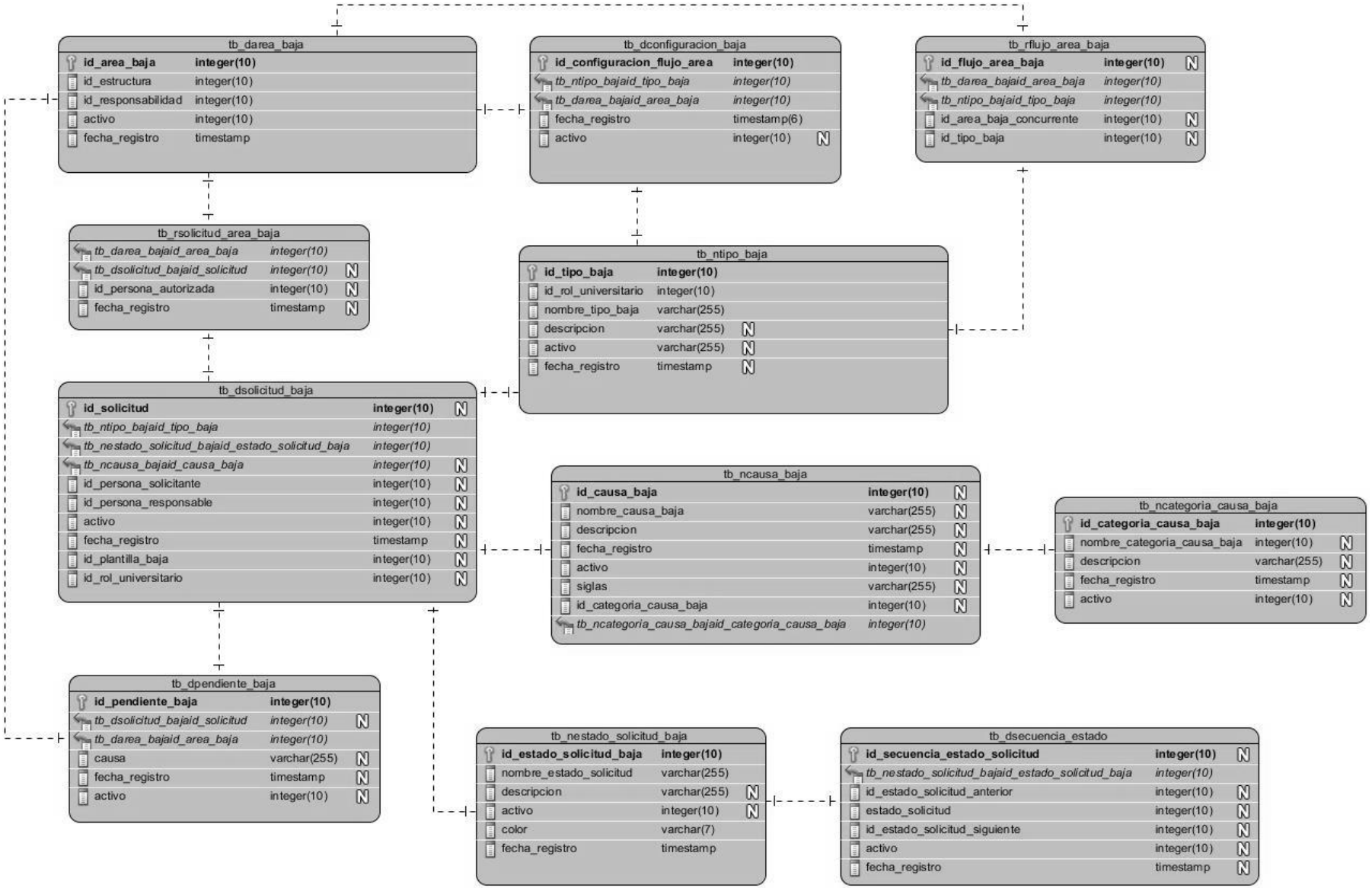


Figura 5 Modelo físico de datos

Este modelo cuenta con 11 tablas persistentes que cubren el proceso de baja aunque en la propuesta se establecen relaciones con otras tablas del SGU las cuales sirven de soporte para complementar la información y evitar la introducción de errores al sistema. Dentro de estas tablas se encuentran: persona, cargos, áreas, entre otras.

2.11 Diagrama de despliegue

Otro elemento clave dentro de la arquitectura, es la vista de despliegue puesto que describe cómo se distribuyen las funcionalidades entre los nodos de cómputo. Los nodos modelan los elementos de hardware sobre los que se ejecuta el sistema.

La Figura 6 muestra los activos físicos presentes en la propuesta de solución.

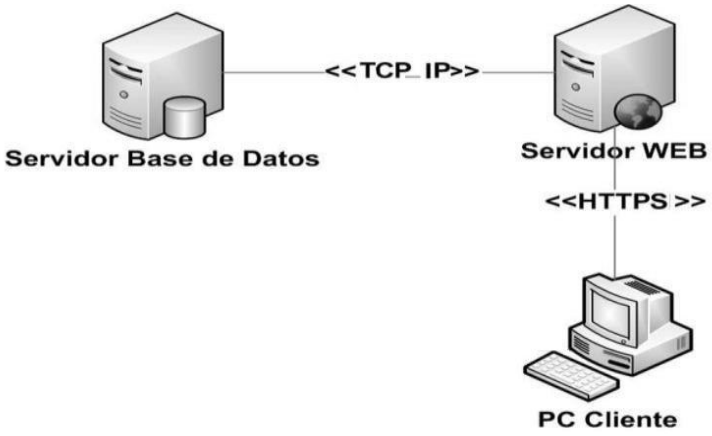


Figura 6 Diagrama de despliegue

Nodos y enlaces de comunicación	Descripción
PC Cliente	Ordenador desde donde los usuarios acceden a la aplicación. Puede tener instalado cualquier sistema operativo y navegador web, aunque que se recomienda Mozilla Firefox 8 o superior.
Servidor Web	Servidor Web Apache 2.2.22
Servidor de BD	Servidor donde se encuentra ubicada la base de datos y de donde los servicios se nutren de información. El servidor debe contar con el sistema operativo Linux Ubuntu 12.04 o superior y el SGDB PostgreSQL 8.4.
HTTPS	Protocolo utilizado para la conexión entre las PC cliente y el servidor web.

TCP/IP	Protocolo utilizado para la conexión entre el servidor web y el servidor donde se encuentra ubicada la base de datos.

Tabla 4 Características de los componentes del despliegue

La orquestación de todos estos elementos garantiza el funcionamiento y soporte de la propuesta de solución.

2.12 Conclusiones parciales

La descripción del proceso de negocio así como la identificación, especificación y descripción de los requisitos permitió obtener una visión clara y concreta de las necesidades del cliente los cuales se evidenciaron en la propuesta de solución y sentaron las bases para la construcción del sistema a través de la arquitectura propuesta.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción

El presente capítulo recoge los elementos fundamentales referentes a la construcción de la propuesta de solución. Se exponen los estándares de codificación a emplear así como el mapa de navegación para orientar al usuario con respecto a la estructura de la aplicación. Finalmente se realiza la validación de la propuesta a través de la realización de pruebas que permiten verificar el funcionamiento, seguridad y completitud del sistema.

3.2 Implementación de la propuesta

Para la implementación de la propuesta y como un requisito no funcional establecido se mantuvieron las pautas establecidas en el manual de directrices del SGU en aras de garantizar la coherencia entre los elementos visuales de la propuesta.

Figura 7 Vista de gestión de procesos

Los elementos fundamentales de esta vista son:

1. Área de subprocesos horizontales.
2. Área del nombre de la aplicación.
3. Área de líneas de procesos.
4. Área de nombre de usuario.
5. Área de funcionalidades.
6. Funcionalidades asociadas al proceso de baja.
7. Área de pie de página.

El mapa de navegación definido para la propuesta se incorpora como parte del mapa de navegación global del SGU. La siguiente figura muestra la ruta para acceder a las opciones relacionadas con el proceso de baja.

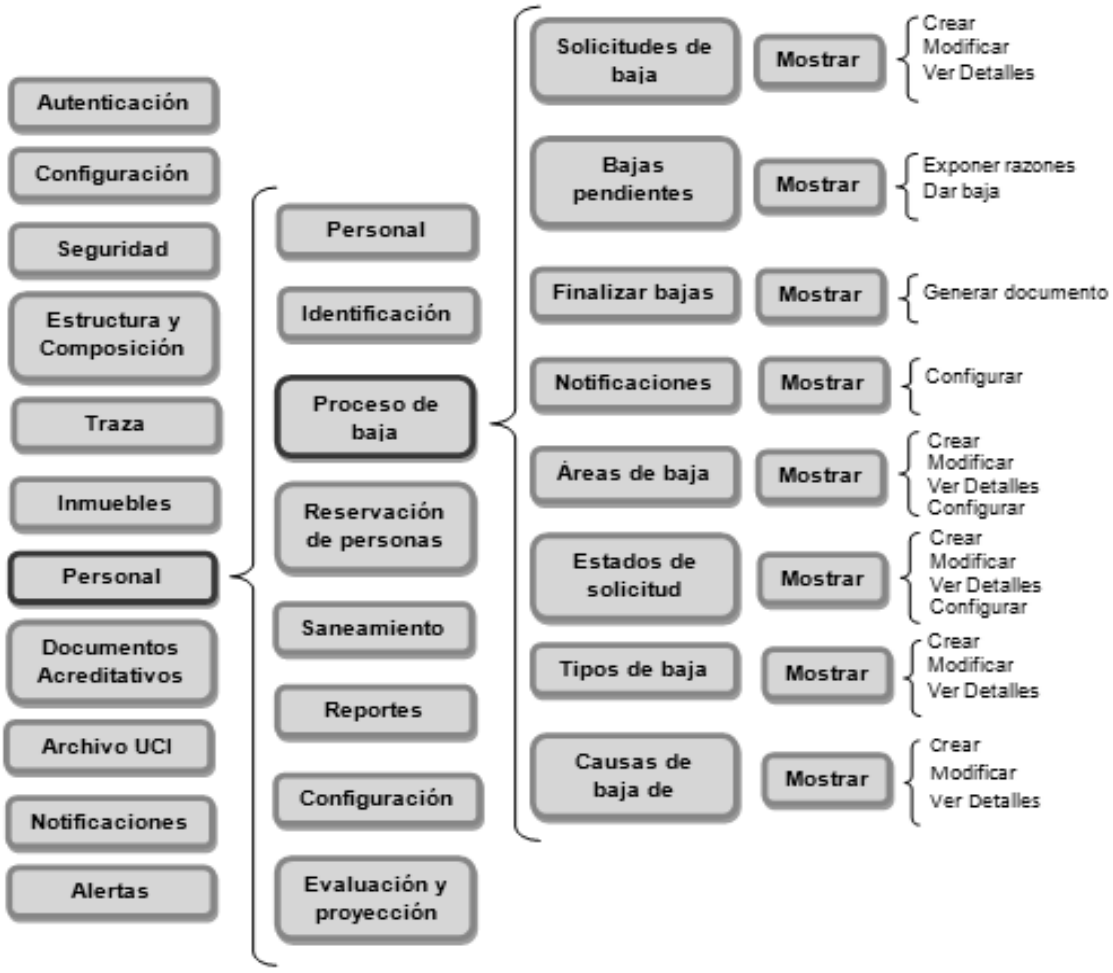


Figura 8 Mapa de navegación

3.3 Integración de la funcionalidad al Sistema de Gestión Universitaria

Todas las funcionalidades relacionadas con el proceso de baja están incluidas dentro del módulo Personal del SGU. Como parte de la implementación del sistema se utilizaron funciones y servicios brindados por otros módulos que facilitaron y apoyaron el proceso de integración con el resto de las funcionalidades. Los módulos con los cuales se vinculó la propuesta fueron:

Personal: la comunicación con esta funcionalidad permite la gestión de la información de todo el personal de la institución.

Configuración: la comunicación con esta funcionalidad permite la configuración de las personas registradas en el sistema definiendo los distintos elementos que la identifican.

Seguridad: la integración con este módulo permite definir la seguridad en cuanto a acceso a funcionalidades y seguridad del negocio.

Estructura y composición: la integración con este módulo permite la gestión de la estructura jerárquica de la institución así como la definición de la composición y la plantilla de las áreas.

Alojamiento: la integración con este módulo permite la gestión de la información de todo el personal de la universidad en la residencia.

3.4 Estándar de codificación

En la implementación de la propuesta se emplearon estándares de codificación para garantizar el estilo de programación establecido dentro del SGU, que son de vital importancia para entender y garantizar la organización, formato y estructura del código fuente.

3.4.1 Indentación, llaves de apertura y cierre y tamaño de líneas

La indentación utilizada es sin tabulaciones, con un equivalente a 4 espacios, para mantener integridad en las revisiones. El uso de las llaves “{}” constituye una nueva línea de código. La longitud de las líneas de código es aproximadamente de 75-80 caracteres, para mantener la legibilidad del código.

```
public function listar() {
    echo $this->template->render('capital_humano_causa_baja/listar_view');
}
```

Figura 9 Denotación y llaves

3.4.2 Convención de nomenclatura

Variables: se rigen por la nomenclatura camelCase (Vidal, 2014). Siempre comienzan por minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.

```
$data = $this->input->all_post();
$id = $data['id'];
```

Figura 10 Variables

Clases: siempre comienzan con mayúscula, en caso de nombres compuestos las palabras se separan con el caracter subrayado “_” y el resto en minúscula.

```
class capital_humano_solicitud extends MY_Controller {

    public function __construct() {
        parent::__construct();
        $this->load->library('capital_humano_solicitud_lib');
        $this->load->library('capital_humano_tipo_baja_lib');
        $this->load->library('capital_humano_estado_solicitud_lib');
        $this->load->library('personas_lib');
        $this->load->library('capital_humano_causa_baja_lib');
        $this->load->library('reservacion_persona_lib');
    }
```

Figura 11 Clases

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.

```
public function obtenerCausaBaja() {
    $resultado = $this->capital_humano_causa_baja_lib->obtenerCausaBajaFiltros();
    echo json_encode($resultado);
}
```

Figura 12 Funciones

Ficheros: Los ficheros siempre se escriben en minúscula 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.
Librerías: con el mismo nombre de la clase que representa que contiene en el nombre el sufijo _lib.
Controladoras: con el mismo nombre de la clase que representa.

3.4.3 Estructuras de control

Se incluye un espacio entre las estructuras de control (if, for, foreach, while, switch) y los paréntesis. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales. Esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos.

```
public function modificarSolicitud() {
  $datos = $this->input->all_post(true);
  if ($this->capital_humano_solicitud_lib->puedeModificarElemento($datos)) {
      if ($this->capital_humano_solicitud_lib->modificar($datos) != false) {
          $idTipoAccion = (int) $this->xml->configuraciones->traza->tipo_accion->modificar_solicitud_baja;
          $elemento = $this->ioc->seguridad->obtenerElementosDadoToken($this->session->userdata('token'));
          $idPersonaRealiza = $elemento['id_persona'];
          $this->ioc->traza->registrarTrazaObjetoPersona($idTipoAccion, $datos['idPersona'], $idPersonaRealiza);

          $this->message('SYS002');
      } else {
          throw new Exception_Error('SYS006');
      }
  } else {
      throw new Exception_Error('SYS005');
  }
}
```

Figura 13 Estructuras de control

3.4.4 Documentación

Todos los archivos deben tener la documentación asociada al mismo. Para esto debe de cumplir con el siguiente bloque al principio de cada clase.

```
* Description of Solicitud
*
* @author Edwin Ruiz Llanes <eruiz@estudiantes.uci.cu>
```

Figura 14 Descripción de clases

3.4.5 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.

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

Figura 15 Buenas prácticas

3.5 Descripción de las clases

Para la implementación del sistema se pusieron en práctica los elementos planteados en el epígrafe anterior, de esta manera la descripción de las clases involucradas en la propuesta cumplen con los estándares de codificación y buenas prácticas. El **Anexo 3** evidencia su cumplimiento a partir de la descripción de una de las clases fundamentales implementadas para la construcción del sistema.

3.6 Pruebas realizadas al sistema

Con el objetivo de identificar y descubrir posibles errores dentro del sistema, se realizaron un conjunto de pruebas que permitieron validar el buen funcionamiento de la solución.

Para la realización de estas pruebas se tuvieron en cuenta los diferentes tipos de objetivos, escenarios y niveles de trabajo propuestos por Pressman (Pressman, 2005).

Pruebas de unidad: La prueba de unidad se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos y que ellos funcionan como se espera.

Pruebas de integración: Las pruebas de integración se concentran en la concepción y la arquitectura del software. Son realizadas para garantizar que los componentes del modelo de implementación funcionan correctamente cuando se combinan para ejecutar un guión de uso. Es el proceso de combinar y probar múltiples componentes juntos.

Pruebas de sistema: Se realizan al sistema como un todo. Está dirigida a verificar el programa final, después que todos los componentes han sido integrados.

Pruebas de aceptación: La prueba de aceptación del usuario es la última acción de prueba antes del despliegue del sistema. El objetivo de la misma es comprobar si el sistema está preparado y lo pueden utilizar los usuarios para realizar las funciones y tareas para las que se diseñó así como verificar que el mismo funciona de tal manera que satisface las expectativas del cliente. Como resultado de esta prueba se obtuvo la plantilla de aceptación del cliente la cual fue aceptada y firmada satisfactoriamente por los mismos.

3.6.1 Métodos de prueba

Los métodos empleados para la realización de las pruebas fueron los definidos por Pressman (Pressman, 2005) los cuales se agrupan en dos grupos.

Pruebas de caja blanca: También llamadas “prueba de caja de cristal” usan la estructura de control del diseño procedimental para obtener los casos de prueba (Pressman, 2005). Mediante estas pruebas se obtienen casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; se ejerciten todas las decisiones lógicas en sus vertientes verdadera

y falsa; se ejecuten todos los bucles en sus límites operacionales y se ejerciten las estructuras internas de datos para asegurar su validez.

Pruebas de caja negra: También denominadas “pruebas de comportamiento”, se centran en los requisitos funcionales del sistema. Están basadas en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el sistema y permiten obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales.

Con este grupo de pruebas se pretende encontrar:

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

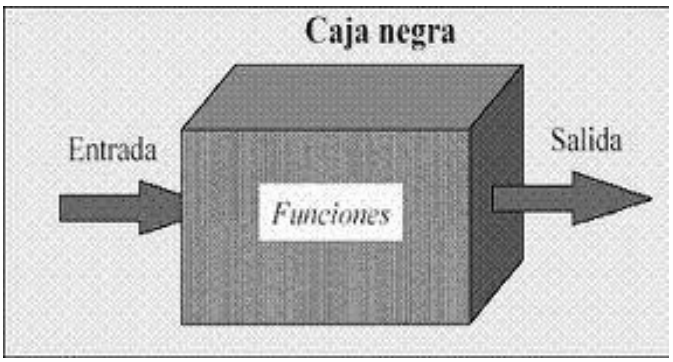


Figura 16 Representación de prueba de caja negra

Dentro de este método, y como parte de esta investigación se utilizó la técnica de la Partición de Equivalencia al ser una de las técnicas más efectivas. Permite examinar los valores válidos e inválidos de las entradas existentes en el sistema, descubre de forma inmediata clases de errores reduciendo así el número de clases de prueba que hay que desarrollar.

3.6.2 Diseño de casos de pruebas

Un caso de prueba es una especificación, de un conjunto de entradas de prueba, condiciones de ejecución y resultados esperados, identificados con el propósito de hacer una evaluación de aspectos particulares de un elemento objeto de prueba (Pressman, 2005).

Como parte de esta investigación se diseñaron casos de prueba para cada uno de los requisitos definidos. La siguiente tabla muestra un caso de prueba realizado para el requisito funcional: “Crear solicitud de baja”, extraído del **Anexo 4**.

Escenario	Descripción	Variable1. Tipo de baja	Variable 2. Causa de baja	Variable 3. Activo	Respuesta del sistema	Flujo central
EC1. Insertar datos correctamente	Este escenario permite crear correctamente una solicitud de baja utilizando datos correctos.	V Trabajadores internos	V A voluntad del trabajador	V Activo	El sistema muestra el mensaje de información: “El elemento ha sido creado satisfactoriamente”	1. Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. 2. Una vez autenticado el usuario en el sistema, selecciona “Personal” en la barra de íconos de los procesos horizontales. 3. Se selecciona en la agrupación funcional "Proceso de baja" de la derecha, la funcionalidad “Solicitudes de baja”. 4. El sistema muestra el listado de solicitudes de baja existentes hasta la fecha. 5. En el área de íconos flotantes selecciona la opción “Crear”. 6. Se llenan los campos: Tipo de baja, Causa de baja, Estado. 7. Se selecciona la persona a la cual se le asignará la solicitud de baja.

						8. Presiona el botón “Aceptar”.

Tabla 5 Diseño de caso de prueba “Crear solicitud”

3.6.3 Resultado de las pruebas

Pruebas de unidad

Para la realización de estas pruebas, que siempre están orientas a pruebas de caja blanca, se utilizó la técnica del camino básico la cual se detalla a continuación.

Prueba de camino básico: con esta técnica se obtiene una medida de complejidad lógica del diseño procedimental (Pressman, 2005) la cual se usó como guía para definir un conjunto básico de rutas de ejecución. La finalidad de esta prueba es derivar un conjunto de casos de prueba a partir de caminos independientes por los cuales debe transitar el flujo de control.

Para obtener el conjunto de caminos independientes se construyó el grafo de flujo asociado y se calculó la complejidad ciclomática del mismo. Dicho algoritmo se presenta a continuación.

```
public function registrarSolicitud() {
    $datos = $this->input->all_post(true);
    if ($this->capital_humano_solicitud_lib->existeElemento($datos) == false) {
        if ($this->capital_humano_solicitud_lib->registrar($datos) != false) {
            $idTipoAccion = (int) $this->xml->configuraciones->traza->tipo_accion->registrar_solicitud_baja;
            $elemento = $this->ioc->seguridad->obtenerElementosDadoToken($this->session->userdata('token'));
            $idPersonaRealiza = $elemento['id_persona'];
            $this->ioc->traza->registrarTrazaObjetoPersona($idTipoAccion, $datos['idPersona'], $idPersonaRealiza);

            $this->message('SYS001');
        } else {
            throw new Exception_Error('SYS006');
        }
    } else {
        throw new Exception_Error('SYS005');
    }
}
```

n.

Figura 17 Código de registrar solicitud

En el grafo de flujo asociado se representaron distintos componentes como fueron:

Complejidad Ciclomática: Es una medida que proporciona una idea de la complejidad lógica de un programa.

La complejidad ciclomática V (G), de un grafo de flujo G, se define como:

V (G) = A – N +2

V (G) = P + 1

A: Número de aristas del grafo.

N: Número de nodos.

P: Número de nodos predicados.

Nodos: son los círculos representados en el grafo, los cuales representan una o más sentencias procedimentales. Un solo nodo puede corresponder a una secuencia de cuadros de proceso y a un rumbo de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: representan el flujo de control y son análogas a las flechas del diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.

Regiones: son las áreas delimitadas por aristas y nodos, donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

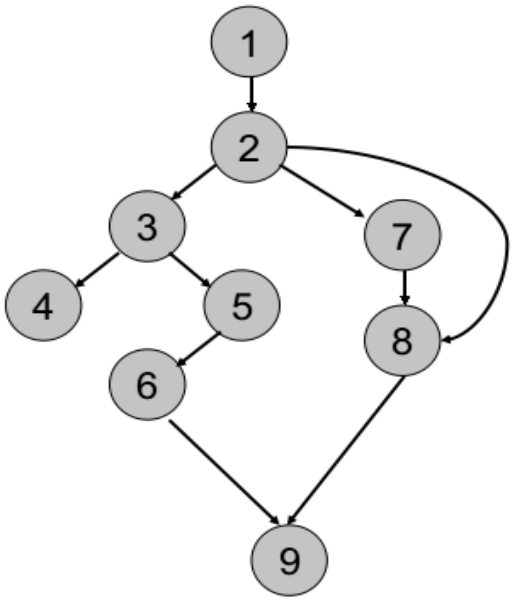


Figura 18 Grafo de flujo

V (G) = A - N + 2

V (G) = 10 - 9 + 2

V (G) = 3

La complejidad ciclomática coincide con el número de regiones del grafo de flujo lo que significa que existen a lo sumo cinco posibles caminos por donde el flujo puede circular. A partir del valor de la complejidad ciclomática se obtiene el número de caminos independientes, los cuales representan el límite mínimo del número total de casos de pruebas para el procedimiento tratado:
Los caminos básicos por los que definen el flujo son:

Camino básico 1: 1-2-5-6-9

Camino básico 2: 1-2-7-8-9

Camino básico 3: 1-2-8-9

Después de haber extraído los caminos básicos, se ejecutaron los casos de pruebas realizando al menos un caso de prueba por cada camino básico.

Casos de prueba

Camino 1: 1-2-5-6-9
Descripción: todos los campos estarán vacíos.
Entrada: un arreglo vacío.
Resultados esperados: el sistema muestra un mensaje de error.
Camino 2: 1-2-7-8-9
Descripción: todos los campos estarán vacíos menos el campo de tipo de baja.
Entrada: un arreglo que contiene datos relacionados al tipo de baja.
Resultados esperados: el sistema muestra un mensaje de error.
Camino 3: 1-2-8-9
Descripción: todos los campos estarán llenos.
Entrada: un arreglo que contiene datos de tipos de baja, categoría causa de baja y causa de baja.
Resultados esperados: el sistema registra la solicitud.

Pruebas de Integración

Para la realización de estas pruebas se escogió el enfoque incremental, de los dos tipos existentes (Pressman, 2005), puesto que permite probar el sistema en pequeños incrementos en los cuales aislar y corregir los errores resulta más fácil.

Estas pruebas proponen un diseño de casos de pruebas específicos para este nivel de prueba. En ellos se especifican los datos del sistema al cual se debe integrar. Cuando el sistema funciona correctamente con estos datos, se asume que las pruebas de integración fueron satisfactorias. A continuación se muestra un ejemplo de los diseños de casos de pruebas realizados.

Caso de Prueba integración del módulo Personal
Sistema/módulo al que se integra: Alojamiento
Condiciones de ejecución: Se realiza la petición de los datos.
Descripción de la prueba: Comprobar que el módulo proporcione la información referente a los datos de las personas en la Residencia.
Pasos de ejecución: El controlador realiza una petición a la librería para obtener información de las personas, gestionados por el módulo Alojamiento.
Resultados esperados: El módulo Alojamiento brinda toda la información solicitada.
Evaluación: Prueba satisfactoria.

Tabla 6 Diseño de casos de prueba Integración al módulo Alojamiento.

Después de realizadas las pruebas y analizando los resultados obtenidos, se puedo constatar la calidad del producto con respecto a los requisitos de integración. Estos elementos fueron tenidos en cuenta para la entrega final al cliente.

Pruebas de sistema

Dentro de esta categoría de pruebas se establece la realización de pruebas de recuperación, pruebas de seguridad, pruebas de resistencia y pruebas de rendimiento. Como parte de esta investigación se realizaron las pruebas de rendimiento detallada a continuación.

La prueba de rendimiento, está diseñada para probar el rendimiento del sistema en tiempo de ejecución dentro del contexto de un sistema integrado (Pressman, 2005). Fueron utilizadas para demostrar que el sistema satisface sus requerimientos, además de monitorizar los tiempos de respuesta de una petición, de la base de datos, el servidor de aplicaciones y otros componentes.

Las pruebas de rendimiento se efectuaron con la herramienta JMeter con el objetivo de realizar mediciones exactas, revisar el comportamiento funcional y medir el rendimiento de la solución. La realización de estas pruebas fueron divididas en test de 50, 100 y 200 hilos cada uno, los cuales simulan la cantidad de usuarios que acceden a las funcionalidades concurrentemente. Se definió un grupo de enlaces a los cuales se simuló el acceso y se tomaron de ellos los datos más significativos para su interpretación. A continuación se muestran los resultados obtenidos con el uso de esta herramienta.

Funcionalidad	# Muestras	Media	Min	Max	% Error	Rendimiento	Kb/sec
Crear Causa Baja.	50	13598	2052	24371	0.00	3.1/sec	5.76
	100	18358	242	37882	0.00	4.6/sec	2.42
	200	20361	1774	38717	2.25	7.8/sec	17.19
Crear Estado Solicitud.	50	4980	2	22978	0.00	8.8/sec	11.78
	100	29291	950	38438	0.00	3.2/sec	3.69
	200	28447	11034	54570	3.05	6.3/sec	8.83
Mostrar Solicitudes de baja.	50	22557	1397	38609	0.00	2.1/sec	4.66
	100	33254	768	73116	0.00	2.7/sec	4.14
	200	27341	2832	58590	1.08	6.5/sec	8.57
Asociar áreas de baja a tipo de baja.	50	24847	2396	34343	0.00	1.9/sec	1.18
	100	39675	4612	56633	1.06	2.3/sec	1.41
	200	28017	1609	61986	2.88	6.0/sec	4.82

Tabla 7 Resultado de las pruebas con jMeter

Los resultados presentados en la tabla anterior expresan de manera general que las peticiones son procesadas en tiempos relativamente rápido, en el caso de la ocurrencia de errores se evidencia que cuando existe una elevada cantidad de peticiones concurrentes el sistema comienza a mostrar algunos errores los cuales no son significativos. Se puede asumir que el resultado obtenido es satisfactorio, lo que demuestra que el sistema es capaz de soportar la conexión simultánea de gran cantidad de usuarios, elemento este establecido dentro de los requisitos del sistema.

Prueba piloto

Otra de las pruebas realizadas al sistema fue la prueba piloto donde se puso a disposición de los usuarios finales, durante un período de dos meses, el sistema para su explotación en un entorno real. En esta prueba estuvieron involucradas diez áreas las cuales validaron el cumplimiento de los requisitos establecidos con la aceptación del sistema. Durante este período se tramitaron satisfactoriamente 170 solicitudes de baja cuyo proceso fue cubierto de manera íntegra hasta alcanzar el estado de “Baja”.

La realización de esta prueba arrojó los siguientes resultados en cada iteración.

- 1ra Iteración: Detectadas 15 no conformidades. Los errores detectados estaban relacionados básicamente con los íconos internos al no realizar la acción correspondiente así como en los mensajes que debía mostrar el sistema al completar una acción.
- 2da Iteración: Detectadas 8 no conformidades relacionadas con el funcionamiento del buscador y los filtros asociados a este además de errores en algunas interfaces que no se mostraban correctamente.
- 3ra Iteración: Dos no conformidades encontradas referidas a errores ortográficos.
- 4ta Iteración: No se detectaron errores.

Todos los errores y no conformidades fueron resueltos en un breve período de tiempo, garantizando la estabilidad y funcionamiento del sistema durante todo el período de prueba.

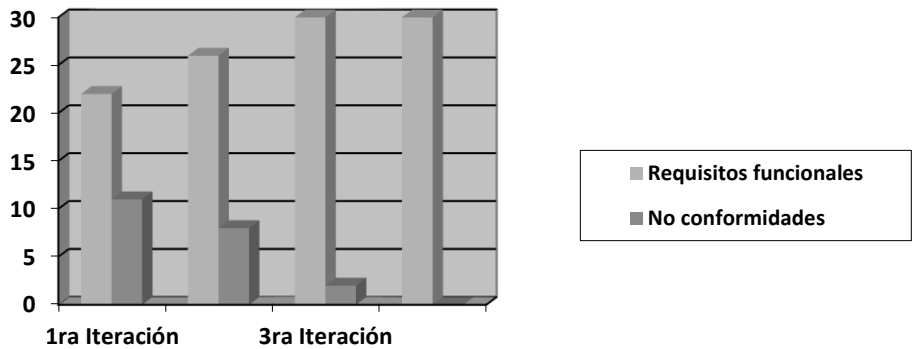


Figura 19 No conformidades

3.7 Conclusiones parciales

Con la implementación de la propuesta de solución se obtuvo un módulo, dentro del SGU, capaz de gestionar, monitorear y controlar el proceso de baja de la institución cuyos requisitos fueron validados a través de diversas técnicas que permitieron validar las funcionalidades desarrolladas. La realización de la prueba piloto tuvo especial importancia puesto que permitió obtener una retroalimentación de primera mano con el cliente, el cual a través del uso de la aplicación pudo constatar la efectividad de la solución.

CONCLUSIONES GENERALES

El desarrollo de esta investigación permitió obtener una solución informática que cubre el proceso de baja en las instituciones universitarias cubanas, favorece la gestión centralizada, el monitoreo, control y supervisión del proceso así como la toma de decisiones por parte de los directivos de nuestra universidad. Con este sistema además se cumple el objetivo trazado como parte de esta investigación puesto que se garantiza la disponibilidad de la información al poner a disposición de los usuarios una aplicación en línea que toma de fuentes válidas y seguras la información para hacer del proceso de baja un proceso eficaz y óptimo, además de lograr gran fiabilidad en los datos ya que una vez que la solicitud de baja es creada solo puede ser gestionada por los responsables asignados en cada una de las áreas y por el directivo que la creó. De manera general se concluye lo siguiente:

- El estudio de los diferentes sistemas homólogos, permitió obtener los conocimientos necesarios para una mayor comprensión del objeto de estudio y para sentar las bases de la investigación.

- El estudio del proceso de baja llevado a cabo en universidades nacionales y extranjeras, permitió establecer las características de la propuesta de solución y todo lo necesario para realizar el análisis y diseño de la funcionalidad a desarrollar.

- Se cumplió el objetivo de la investigación, pues se realizó la implementación de la funcionalidad Proceso de baja para estudiantes y trabajadores de la UCI, como parte del SGU, cumpliendo con todos los requisitos establecidos.

- Los resultados arrojados por las pruebas realizadas, mostraron el correcto funcionamiento de la solución.

RECOMENDACIONES

Se recomienda, con el objetivo de agregar valor a la solución y como parte del proceso de mejora continua del software incorporar un algoritmo de optimización de rutas para que en función del lugar donde reside la persona o el área de trabajo el sistema le recomiende una ruta o camino mínimo a seguir para la visita de aquellas áreas en las que el solicitante tenga adeudos.

REFERENCIAS BIBLIOGRÁFICAS

1. **Apache Software Foundation. 2014.** Apache HTTP Server Project. [En línea] 2014. http://httpd.apache.org/.

2. **ASSETS NS. 2014.** ASSETS NS. Sitio oficial. [En línea] 2014. https://assets.co.cu.

3. **Buschmann, Frank, y otros. 2000.** *Pattern-Oriented Software Architecture.* Alemania : Siemens AG, 2000. 0 471 95889 7.

4. **Collazo, Yasmany Tellez. 2013.** *Arquitectura de Software, Vista de Tecnología e Infraestructura.* La Habana : s.n., 2013.

5. **CUJAE. 2011.***Manual de usuario del SIGENU, Módulo Web de Secretaría.* La Habana : s.n., 2011.

6. **Evolus. 2012.** Pencil Project. [En línea] 2012. http://pencil.evolus.vn/.

7. **Gamma , Erich , y otros. 2003.** *Design Patterns, Elements of Reusable Object-Oriented Software Gamma.* 2003.

8. **Guerrero, Rafael Martinez. 2013.** PostgreSQL. [En línea] 2013. http://www.postgresql.org.es/.

9. **Humanos, Dirección de Recursos. 2010.** *Manual de Normas y Procedimientos.* La Habana : s.n., 2010.

10. **Institucional, Dirección de Desarrollo. 2013.** *Manual de procesos y procedimientos.* La Habana : s.n., 2013.

11. **ISO. 2000.** *Sistema de gestión de la calidad.* Ginebra : Secretaría Central de ISO, 2000.

12. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación, S.A, 2000. 84-7829-036-2.

13. **Kruchten , Philippe. 2000.** *UML y Patrones.* Canada : Vancouver, BC , 2000.

14. **Laviña Orueta , Jaime y Mengual Pavón, Laura . 2010.** *Libro Blanco de la Universidad Digital 2010.* Barcelona : Editorial Ariel, S.A., 2010. 978-84-08-08417-4.

15. **Méndez, Alelí Sánchez. 2013.** *Proceso de desarrollo de software DAC.* La habana : s.n., 2013.

16. **Oracle Corporation. 2013.** NetBeans. [En línea] 2013. https://netbeans.org/features/index.html.

17. **Pérez, Javier Eguíluz. 2009.** ¿Qué es JavaScript? [En línea] 2009. http://www.librosweb.es/javascript/index.html.

18. **Pérez, Javier Eguíluz. 2007.** *Introducción a CSS.* España : s.n., 2007.

19. **Pérez, Javier Eguiluz. 2007.** *Introducción a JavaScript.* 2007.

20. **Pérez, Javier Eguíluz. 2007.** *Introducción a XHTML.* 2007.

21. **Pérez, Juan Diego. 2005.** *Notaciones y lenguajes de procesos. Una visión global.* 2005. 25179398X.

22. **PostgreSQL. 2014.** PgAdmin, Sitio oficial. [En línea] 2014. http://www.pgadmin.org/.

23. **Pressman, Rogger S. 2005.** *Ingeniería del Software, Un enfoque práctico.* 2005. 9789701054734.

24. **Reynoso, Carlos Billy. 2004.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

25. **Rosselott, Marcelo Maraboli. 2003.** *Manual de Programación en PHP.* Valparaiso : s.n., 2003.

26. **Sommerville, Ian. 2005.** *Ingeniería del Software.* Madrid : Pearson Educación, S.A., 2005. 84-7829-074-5.

27. **Universidad Cuauhtemoc. 2014.** Sitio oficial de la Universidad. [En línea] 2014. http://www.ucuauhtemoc.edu.mx.

28. **Vidal, Yanio García. 2014.** *Estandar de codificación.* 2014.

29. **W3C Corporations. 2014.** W3C. [En línea] 2014. http://www.w3.org/.

30. **White, Stephen A. 2005.** *Introduction to BPMN.* 2005.

31. **Young, Ralph R. 2004.** *The Requirements Engineering Handbook.* 2004. 1-58053-266-7.

BIBLIOGRAFÍA CONSULTADA

1. **Álvarez, Miguel Ángel.** DesarrolloWeb.com. [En línea] 2009. http://www.desarrolloweb.com.

2. **Arias Chaves, Michael.** *La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software.* Universidad de Costa Rica : s.n., 2005.

3. **Foundation, The jQuery.** JQuery is a new kind of JavaScript Library. [En línea] 2012. http://jquery.com/.

4. **Martínez, Ander.** Apache JMeter. *Apache JMeter. Manual de usuario v1.2.* [En línea] Mayo de 2009. http://jakarta.apache.org/jmeter/.

5. **González Castellanos, Roberto A., Yll Lavín, Curiel Lorenzo, Lilian D.** *Metodología de la Investigación Científica para las Ciencias Técnicas.* Matanzas : Universidad de Matanzas, 2003.

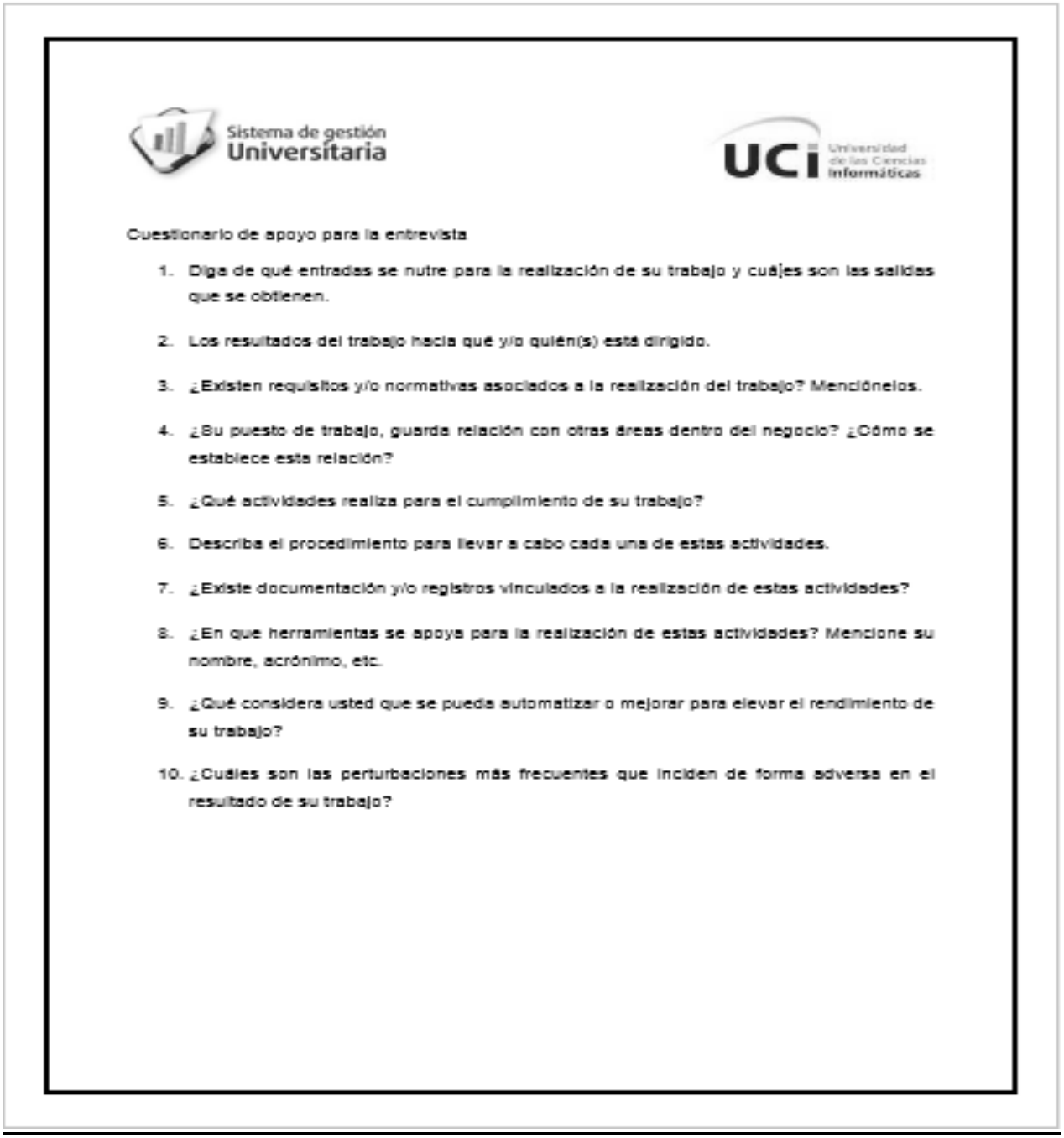
6. **Pérez, José R. Capó. 2013.** Mayabeque : s.n., 2013. *Sistema de gestión integrada de capital humano para las universidades cubanas.*

7. **Universidad de Celaya.** Honduras : s.n. *Procedimiento para efectuar la baja del personal no académico.*

8. **García, William Claro. 2013.** La Habana : s.n., 2013. 978-959-7213-02-4. *Propuesta de Sistema para el Control de Residentes en Universidades Médicas.*

ANEXOS

Anexo 1: Cuestionario para la Entrevista



Sistema de gestión Universitaria

UCI Universidad de las Ciencias Informáticas

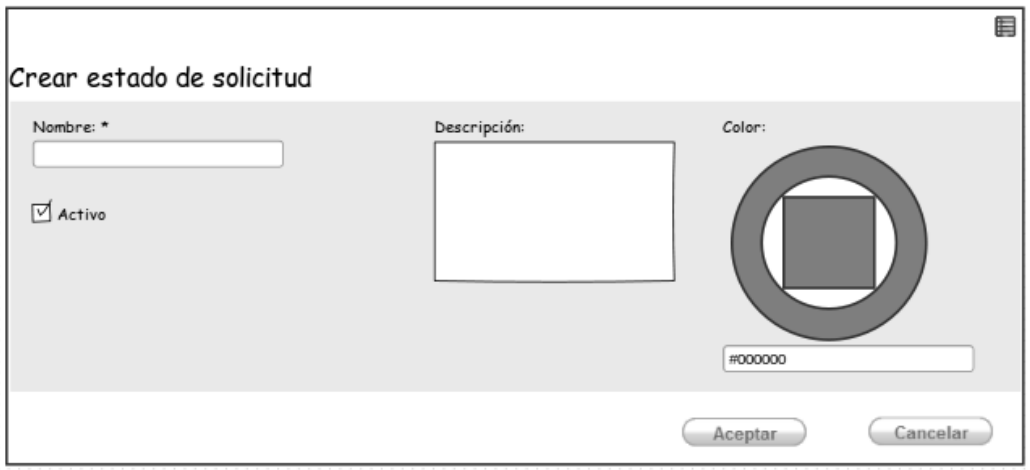
Cuestionario de apoyo para la entrevista

1. Diga de qué entradas se nutre para la realización de su trabajo y cuáles son las salidas que se obtienen.
2. Los resultados del trabajo hacia qué y/o quién(s) está dirigido.
3. ¿Existen requisitos y/o normativas asociados a la realización del trabajo? Menciónelos.
4. ¿Su puesto de trabajo, guarda relación con otras áreas dentro del negocio? ¿Cómo se establece esta relación?
5. ¿Qué actividades realiza para el cumplimiento de su trabajo?
6. Describa el procedimiento para llevar a cabo cada una de estas actividades.
7. ¿Existe documentación y/o registros vinculados a la realización de estas actividades?
8. ¿En que herramientas se apoya para la realización de estas actividades? Mencione su nombre, acrónimo, etc.
9. ¿Qué considera usted que se pueda automatizar o mejorar para elevar el rendimiento de su trabajo?
10. ¿Cuáles son las perturbaciones más frecuentes que inciden de forma adversa en el resultado de su trabajo?

Figura 20 Cuestionario para la entrevista

Anexo 2: Especificación de requerimientos

Nº	***Nombre***	***Descripción***	***Complejidad***	***Prioridad para cliente***
FPers87	Crear estado de solicitud	• El requisito permite crear un nuevo estado de solicitud. El usuario con los permisos asignados selecciona del módulo Personal, del "Sistema de Gestión Universitaria", la opción Estados de solicitud del menú de funcionalidades Proceso de baja. • Se muestra un listado con los estados de solicitud existentes en el sistema. • En el área de íconos flotantes selecciona la acción Crear. • Se muestra una interfaz que permite introducir los datos: nombre, descripción, estado (Activo/Inactivo) y seleccionar el color y se presiona el botón Aceptar para crear un estado de solicitud. • Se permite crear un nuevo estado o volver al listado de estados de solicitud.	*Baja*	*Media*
	Prototipo			

<table>
<tr><td></td><td colspan="3"> 
</td></tr>
<tr><td></td><td>Campos</td><td>Tipos de Datos</td><td>Reglas o Restricciones</td></tr>
<tr><td rowspan="5"></td><td>Nombre</td><td>Varchar</td><td>• Obligatorio.
• Único.
• Solo admite letras, números, y espacio o guión bajo entre palabras.
• Admite entre 2 y 50 caracteres.
• Solo admite 30 caracteres por palabra.</td></tr>
<tr><td>Descripción</td><td>Varchar</td><td>• Admite hasta 200 caracteres.
• Admite 30 caracteres por palabra.</td></tr>
<tr><td>Estado</td><td>Boolean</td><td>• Selección.</td></tr>
<tr><td>Color</td><td>Varchar</td><td>• Selección.
• Obligatorio.</td></tr>
<tr><td>Observaciones</td><td colspan="2">1. El usuario con los permisos asignados debe estar autenticado en el sistema.
2. Si no ocurren errores debe mostrarse el mensaje de información: “El elemento ha sido creado satisfactoriamente”.
3. Si ocurre un error durante la operación se muestra el mensaje: “Ha ocurrido un error cuando intentaba realizar la acción”.
4. En caso que el valor de un campo único exista se muestra un mensaje de error: “El elemento ya existe”.
5. En caso que se deje un campo de los obligatorios vacío se muestra un mensaje en rojo “Campo requerido” encima del campo que debe ser llenado obligatoriamente.
6. En caso de que en el campo Nombre introduzca menos cantidad de</td></tr>
</table>

		caracteres de los que están definidos se muestra un mensaje en rojo encima del campo “Entre al menos 2 caracteres”. 7. Los campos de texto solo admiten 30 caracteres por palabra. 8. Si excede el número de caracteres por palabra en los campos que lo requieran el sistema muestra un mensaje en rojo: "Ha excedido el número de letras permitidas para una palabra" encima del campo que debe ser llenado correctamente. 9. Para todos los campos de texto cuando llegue a la cantidad máxima de palabras definidas el sistema no permitirá continuar escribiendo. 10. Si se introducen caracteres extraños en el campo Nombre se muestra un mensaje en rojo “Entre solo letras, números, y espacio o guión bajo entre palabras.” Encima del campo que debe ser llenado correctamente. 11. En caso de cancelar la acción se muestra un mensaje de advertencia “¿Está seguro de realizar la acción?”. 12. Ver en el documento RN, la regla de negocio RNPB_4 Color de estados.

Tabla 8Descripción del requisito Crear estado de solicitud

Nº	***Nombre***	***Descripción***	***Complejidad***	***Prioridad para cliente***
RFPers6	Dar baja por área	• El requisito permite dar baja a una persona en un área determinada. El usuario con los permisos asignados selecciona del módulo Personal, del “Sistema de Gestión Universitaria”, la opción Bajas pendientes del menú de funcionalidades Proceso de baja. • Se muestra un listado con las solicitudes de bajas pendientes en el sistema. • En el área de íconos internos se selecciona lo opción Dar baja a la solicitud.	*Media*	*Media*
	Prototipo			

	Campos	***Tipos de Datos***	***Reglas o Restricciones***
	Buscar	Varchar	• No admite porcientos y apóstrofes. • Solo admite 30 caracteres por palabra.
	Filtrar búsqueda	Varchar	• Selección.
	Foto	Bytea	• Solo lectura.
	Solicitante	Varchar	• Solo lectura.
	Responsable	Varchar	• Solo lectura.
	Estado	Boolean	• Solo lectura.
	Número Expediente	Varchar	• Solo lectura.
	Fecha	Varchar	• Solo lectura.
	Observaciones	1. El usuario con los permisos asignados debe estar autenticado en el sistema. 2. Las bajas pendientes se ordenan en la lista por el nombre. 3. La cantidad de elementos a mostrar en la lista son 5, 10, 15 y 20 según la preferencia del usuario. 4. Los campos de texto solo admiten 30 caracteres por palabra. 5. Cuando no existen elementos creados se muestra el mensaje “No existen elementos a mostrar”. 6. Si el usuario introduce un número de página mayor que la cantidad de páginas el sistema muestra la última página. 7. Ver en el documento RN, la regla de negocio RNPB_6 Causar baja. 8. Ver en el documento RN, la regla de negocio RNPB_9 Dar baja.	

Tabla 9 Descripción del requisito Dar baja por área

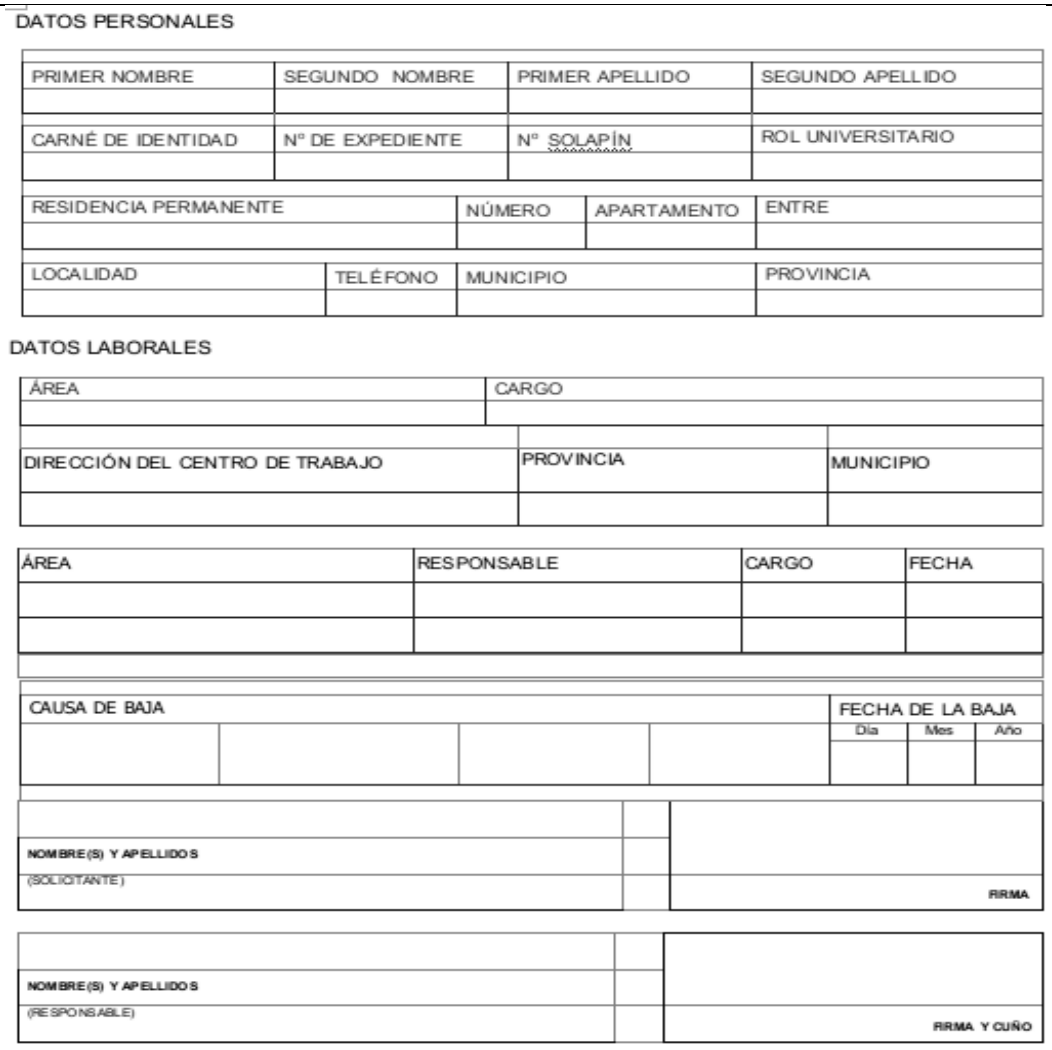
Nº	***Nombre***	***Descripción***	***Complejidad***	***Prioridad para cliente***

RFPers85	Configurar áreas de visita obligatoria	• El requisito permite configurar las áreas de visita obligatoria para un tipo de baja. El usuario con los permisos asignados selecciona del módulo Personal, del "Sistema de Gestión Universitaria", la opción Áreas de baja del menú de funcionalidades Proceso de baja. • Se muestra un listado de las áreas de baja existentes en el sistema. • Se permite buscar un área según el nombre. En el área de íconos internos selecciona la acción Configurar área de visita obligatoria. • Se muestra una interfaz que permite seleccionar el Tipo de baja y asociar las áreas de visita obligatoria. • Se permite configurar el área o volver al listado de áreas de baja.	*Baja*	*Media*
	Prototipo			

Campos	***Tipos de Datos***	***Reglas o Restricciones***
Área de baja	Varchar	• Único. • Solo lectura. • Obligatorio.
Tipos de baja	Varchar	• Obligatorio. • Selección.
Áreas de visita obligatoria	Varchar	• Selección.
Observaciones	1. El usuario con los permisos asignados debe estar autenticado en el sistema. 2. Si no ocurren errores debe mostrarse el mensaje de información: “La configuración se ha realizado satisfactoriamente”. 3. Si ocurre un error durante la operación se muestra el mensaje: “Ha ocurrido un error cuando intentaba realizar la acción”. 4. En caso que se deje un campo de los obligatorios vacío se muestra un mensaje en rojo “Campo requerido” encima del campo que debe ser llenado obligatoriamente. 5. En caso de cancelar la acción se muestra un mensaje de advertencia “¿Está seguro de realizar la acción?”. 6. Ver en el documento RN, la regla de negocio RNPB_5 Visita a áreas.	

Tabla 10 Descripción del requisito Configurar áreas de visita obligatoria

Nº	***Nombre***	***Descripción***	***Complejidad***	***Prioridad para cliente***
RFPers80	Generar documento de baja	• El requisito permite generar un documento en formato pdf con los datos de la solicitud y el permiso de baja en todas las áreas requeridas. El usuario con los permisos asignados selecciona del módulo Personal, del "Sistema de Gestión Universitaria", la opción Finalizar baja del menú de funcionalidades Proceso de baja. • Se muestra un listado con las solicitudes de baja a finalizar en el sistema. • En el área de íconos internos selecciona la opción Generar documento de baja.	*Media*	*Media*
	Prototipo			



DATOS PERSONALES

PRIMER NOMBRE	SEGUNDO NOMBRE	PRIMER APELLIDO	SEGUNDO APELLIDO
CARNÉ DE IDENTIDAD	Nº DE EXPEDIENTE	Nº SOLAPÍN	ROL UNIVERSITARIO

RESIDENCIA PERMANENTE	NÚMERO	APARTAMENTO	ENTRE

LOCALIDAD	TELÉFONO	MUNICIPIO	PROVINCIA

DATOS LABORALES

ÁREA	CARGO

DIRECCIÓN DEL CENTRO DE TRABAJO	PROVINCIA	MUNICIPIO

ÁREA	RESPONSABLE	CARGO	FECHA

CAUSA DE BAJA				FECHA DE LA BAJA		
				Día	Mes	Año

NOMBRE(S) Y APELLIDOS	
(SOLICITANTE)	FIRMA

NOMBRE(S) Y APELLIDOS	
(RESPONSABLE)	FIRMA Y CUÑO

	Campos	***Tipos de Datos***	***Reglas o Restricciones***
	Buscar	Varchar	• No admite porcientos y apóstrofes. • Admite hasta 30 caracteres por palabra.
	Filtrar búsqueda	Varchar	• Selección.
	Foto	Bytea	• Solo lectura.
	Solicitante	Varchar	• Solo lectura.
	Responsable	Varchar	• Solo lectura.
	Estado	Boolean	• Solo lectura.
	Fecha	Varchar	• Solo lectura.

	Observaciones	1. El usuario con los permisos asignados debe estar autenticado en el sistema. 2. Si ocurre un error durante la operación se muestra el mensaje: “Solo puede generar el documento final la persona que creó la solicitud”. 3. Los campos de texto solo admiten 30 caracteres por palabra. 4. La cantidad de elementos a mostrar en la lista son 5, 10, 15 y 20 según la preferencia del usuario. 5. Ver en el documento RN, la regla de negocio RNPB_3 Generar documento. 6. Ver en el documento RN, la regla de negocio RNPB_10 Documento final.

Tabla 11 Descripción del requisito Generar documento de baja

Anexo 3: Descripción de las clases

Nombre de la clase: capital_humano_solicitud
Tipo de clase: controladora
Para cada responsabilidad
Nombre: _construct ()
Descripción: constructor de la clase. Instancia el núcleo de CodeIgniter y carga las librerías necesarias.
Nombre: index ()
Descripción: renderiza la clase vista listar_solicitud_view.
Nombre: obtenerRolUniversitario ()
Descripción: obtiene los datos enviados por la clase librería personas_lib.
Nombre: obtenerEstadoSolicitud ()
Descripción: obtiene los datos de los estados de solicitud enviados por la clase librería capital_humano_estado_solicitud_lib.
Nombre: obtenerTipoBaja ()
Descripción: obtiene los datos de los tipos de baja enviados por la clase librería capital_humano_tipo_baja_lib.
Nombre: obtenerAreaBaja ()
Descripción: obtiene los datos de las áreas de baja enviados por la clase librería capital_humano_tipo_baja_lib.
Nombre: obtenerCausaBaja ()
Descripción: obtiene los datos de las causas de baja enviados por la clase librería capital_humano_causa_baja_lib.

Nombre: obtenerEstructura ()
Descripción: obtiene los datos de las estructuras enviados por la clase librería personas_lib.
Nombre: obtenerCausasDeBajaDadoIdCategoriaCausaBaja()
Descripción: obtiene las causas de baja dado un id de una categoría enviados por la clase librería capital_humano_causa_baja_lib.
Nombre: obtenerSolicitud ()
Descripción: obtiene los datos de las solicitudes de baja.
Nombre: detallesSolicitud()
Descripción: obtiene los datos de las solicitudes de baja y los renderiza a la clase vista detalles_solicitud_view.
Nombre: registrarSolicitud()
Descripción: obtiene los datos de las solicitudes de baja y registra una nueva solicitud
Nombre: modificarSolicitud()
Descripción: modifica una solicitud de baja.

Tabla 12 Descripción de la clase controladora “capital_humano_solicitud”.

Nombre de la clase: capital_humano_solicitud_lib
Tipo de clase: librería
Para cada responsabilidad
Nombre: _construct()
Descripción: constructor de la clase. Instancia el núcleo de CodeIgniter y carga las clases del modelo necesarias.
Nombre: obtenerCantidadSolicitudes ()
Descripción: recibe un arreglo y lo envía a la clase modelo.
Nombre: obtenerSolicitudDadoId ()
Descripción:recibe el id de unasolicitud y obtiene los datos de la clase modelo.
Nombre: obtenerAreasBajasAVisitarDadoIdSolicitud ()
Descripción:recibe el id de una solicitud y obtiene las áreas que deben ser visitadas de la modelo.
Nombre: existeElemento ()
Descripción: recibe los datos de una solicitud y comprueba si existe mediante la modelo.
Nombre: registrar ()
Descripción: recibe los datos de una solicitud y los registraen la modelo.
Nombre: modificar ()
Descripción: recibe los datos de una solicitud y los registra en la modelo.
Nombre: realizarNotificacionAlCrearSolicitud ()

Descripción: recibe el id de una solicitud y notifica a cada uno de los responsables en cada área.

Tabla 13 Descripción de la clase librería “capital_humano_solicitud_lib”.

Nombre de la clase: tb_dsolicitud_baja_mdl
Tipo de clase: modelo
Para cada responsabilidad
Nombre: _construct()
Descripción: constructor de la clase.
Nombre: obtenerSolicitudes ()
Descripción: obtiene las solicitudes de baja registradas de forma ascendente.
Nombre: obtenerSolicitudDadoId ()
Descripción: recibe el id de una solicitud y obtiene los datos.
Nombre: obtenerAreasBajasAVisitarDadoIdSolicitud ()
Descripción: recibe el id de una solicitud y obtiene los datos.
Nombre: obtenerAreasBajaAsociadasASolicitud ()
Descripción: recibe el id de una solicitud y obtiene las áreas de baja asociadas.
Nombre: obtenerNombreEstructuraDadoIdAreaBaja ()
Descripción: recibe el id de una área de baja y obtiene sus estructuras.
Nombre: obtenerDirectivosDeAreasBaja ()
Descripción: recibe el id de un tipo de baja y obtiene los directivos asociados.
Nombre: tieneAreasDependientesDadoIdSolicitud ()
Descripción: recibe los id de un tipo de baja, solicitud y una estructura y obtiene las áreas dependientes.

Tabla 14 Descripción de la clase modelo “tb_dsolicitud_baja_mdl”

Nombre de la clase: listar_solicitud_view
Tipo de clase: vista
Para cada responsabilidad
Nombre: solicitud.init()
Descripción: carga los datos de las solicitudes.

Tabla 15 Descripción de la clase modelo “listar_solicitud_view”.

Anexo 4: Diseño de casos de prueba

Escenario	Descripción	Variable 1.	Variable 2.	Variable 3.	Respuesta	Flujo central

		Tipo de baja	**Causa de baja**	**Activo**	**del sistema**	
EC1. Insertar datos correctamente	Este escenario permite crear correctamente una solicitud de baja utilizando datos correctos.	*V* Trabajadores internos	*V* A Voluntad del Trabajador	*V* Activo	El sistema muestra el mensaje de información: "El elemento ha sido creado satisfactoriamente"	1. Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. 2. Una vez autenticado el usuario en el sistema, selecciona "Personal" en la barra de íconos de los procesos horizontales. 3. Se selecciona en la agrupación funcional "Proceso de baja" de la derecha, la funcionalidad "Solicitudes de baja". 4. El sistema muestra el listado de áreas de baja existentes hasta la fecha. 5. En el área de íconos flotantes selecciona la opción "Crear". Se

						llenan los campos: Tipo de baja, Causa de baja, Estado. 7. Se selecciona la persona a la cual se le asignará la solicitud de baja. 8. Presiona el botón “Aceptar”.
EC2. Insertar elementos repetidos	Este escenario no permite crear una solicitud de baja que ya exista en el sistema.	*I* Trabajadores internos	*I* A Voluntad del Trabajador	*V* Activo	El sistema muestra el mensaje de error: “El elemento ya existe”.	1. Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. 2. Una vez autenticado el usuario en el sistema, selecciona “Personal” en la barra de íconos de los procesos horizontales. 3. Se selecciona en la agrupación funcional "Proceso de baja" de la derecha, la funcionalidad “Solicitudes de baja”. 4. El sistema

						muestra el listado de áreas de baja existentes hasta la fecha. 5. En el área de íconos flotantes selecciona la opción “Crear”. Se llenan los campos: Tipo de baja, Causa de baja, Estado. 7. Se selecciona la persona a la cual se le asignará la solicitud de baja. 8. Presiona el botón “Aceptar”.
EC3. Insertar dejando campos vacíos	Este escenario no permite crear una solicitud de baja dejando campos vacíos.	*V* *Trabajadores internos*	*NA* *-*	*NA* *Activo*	El sistema muestra en rojo el mensaje “Campo requerido” sobre el componente que debe ser llenado de forma obligatoria.	1. Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. 2. Una vez autenticado el usuario en el sistema, selecciona “Personal” en la barra de íconos de los procesos horizontales.

						3. Se selecciona en la agrupación funcional "Proceso de baja" de la derecha, la funcionalidad "Áreas de baja". 4. El sistema muestra el listado de áreas de baja existentes hasta la fecha. 5. En el área de iconos flotantes selecciona la opción "Crear". 6. Se llenan los campos: Categoría de estructura, Estructura, Estado, Responsabilidad. 7. Presiona el botón "Aceptar".
EC4.Canc	Este escenario				El sistema	1. Al autenticarse el

elar operación	permite cancelar la operación de crear una solicitud de baja.				muestra el mensaje de advertencia: “¿Está seguro de realizar la acción?”.	usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. 2. Una vez autenticado el usuario en el sistema, selecciona “Personal” en la barra de íconos de los procesos horizontales. 3. Se selecciona en la agrupación funcional "Proceso de baja" de la derecha, la funcionalidad “Solicitudes de baja”. 4. El sistema muestra el listado de solicitudes de baja existentes hasta la fecha. 5. En el área de íconos flotantes selecciona la opción “Crear”. 8. Presiona el botón “Cancelar”.

Tabla 16Diseño de caso de prueba Crear solicitud de baja

Escenario	Descripción	Respuesta del sistema	Flujo central
EC1. Generar documento correctamente.	Mediante este escenario se generará el documento final de la baja.	El sistema muestra genera el documento final de la baja.	1. Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. 2. Una vez autenticado el usuario en el sistema, selecciona “Personal” en la barra de íconos de los procesos horizontales. 3. Se selecciona en la agrupación funcional "Proceso de baja" de la derecha, la funcionalidad “Finalizar bajas”. 4. En el área de iconos internos selecciona la opción “Generar documento” 5. Presiona el botón “Aceptar” 6. El sistema muestra el listado de solicitudes de baja finalizadas.
EC2. Cancelar operación	Este escenario permite cancelar la operación de crear una solicitud de baja.	El sistema muestra el mensaje de advertencia: “¿Está seguro de realizar la acción?”.	1. Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. 2. Una vez autenticado el usuario en el sistema, selecciona “Personal” en la barra de íconos de los procesos horizontales. 3. Se selecciona en la agrupación funcional "Proceso de baja" de la derecha, la funcionalidad “Finalizar bajas”. 4. El sistema muestra el listado de solicitudes de baja finalizadas. 5. En el área de íconos internos selecciona la opción “Generar documento”. 8. Presiona el botón “Cancelar”.

Tabla 17Diseño de caso de prueba Generar documento de baja

Escenario	Descripción	Variable1.	Variable2.	Respuesta del	Flujo central

		Tipo de baja	Áreas de baja	sistema	
EC1. Seleccionar datos correctos	Este escenario permite seleccionar las áreas de baja asociadas al tipo de baja seleccionado.	*V* Estudiantes	*V* "Departamento de cultura física"	El sistema muestra el mensaje de información: *"Los datos han sido asociados satisfactoriamente"*	1. Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. 2. Una vez autenticado el usuario en el sistema, selecciona "Personal" en la barra de íconos de los procesos horizontales. 3. Se selecciona en la agrupación funcional "Proceso de baja" de la derecha, la funcionalidad "Tipos de baja". 4. El sistema muestra el listado de tipos de baja existentes hasta la fecha. 5. En el área de iconos internos selecciona la opción "Asociar áreas a tipo de

					bajar”. 6. Se seleccionan las áreas de baja. 7. Presiona el botón “Aceptar”.
EC2. Dejar campos vacíos	Este escenario no permite dejar campos obligatorios vacíos.	*V* “Trabajadores externos”	*I* “vacío”	El sistema muestra un mensaje de error: *“Campo requerido”,* en rojo encima del campo obligatorio.	1. Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. 2. Una vez autenticado el usuario en el sistema, selecciona “Personal” en la barra de íconos de los procesos horizontales. 3. Se selecciona en la agrupación funcional "Proceso de baja" de la derecha, la funcionalidad “Tipos de baja”. 4. El sistema muestra el listado de tipos de baja existentes hasta la fecha. 5. En el área de

					iconos internos selecciona la opción “Asociar áreas a tipo de baja”. 6. Se seleccionan las áreas de baja. 7. Presiona el botón “Aceptar”.
EC3. Cancelar operación	Este escenario permite cancelar la operación de asociar áreas de baja al tipo de baja seleccionado.	*NA*	*NA*	El sistema muestra un mensaje de advertencia: “*¿Está seguro de realizar la acción?”*	1. Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. 2. Una vez autenticado el usuario en el sistema, selecciona “Personal” en la barra de íconos de los procesos horizontales. 3. Se selecciona en la agrupación funcional "Proceso de baja" de la derecha, la funcionalidad “Tipos de baja”. 4. El sistema muestra el listado de tipos de baja

					existentes hasta la fecha. 5. En el área de íconos internos selecciona la opción “Asociar áreas a tipo de baja”. 6. Presiona el botón “Cancelar”.

Tabla 18 Diseño de caso de prueba Asociar área de baja a tipo de baja

Anexo 5: Pruebas de Caja blanca

```
public function darBaja() {
    $datos = $this->input->all_post(true);
    $resul = $this->capital_humano_baja_pendiente_lib->darBajaDeArea($datos);
    $idPersonaSolicitante = $this->capital_humano_baja_pendiente_lib->buscarPersonaSolicitanteDadoIdSolicitudBaja($datos);

    $idTipoAccion = (int) $this->xml->configuraciones->traza->tipo_accion->baja_para_un_area;
    $elemento = $this->ioc->seguridad->obtenerElementosDadoToken($this->session->userdata('token'));
    $idPersonaRealiza = $elemento['id_persona'];
    $this->ioc->traza->registrarTrazaObjetoPersona($idTipoAccion, $idPersonaSolicitante, $idPersonaRealiza);

    $obj = new stdClass();
    $obj->dioBaja = true;

    if (!is_array($resul)) {
        echo json_encode($obj);
    } else {
        $obj = new stdClass();
        $obj->dioBaja = false;
        $obj->message = $resul;
        echo json_encode($obj);
    }
}
```

Figura 21 Código Dar baja por área

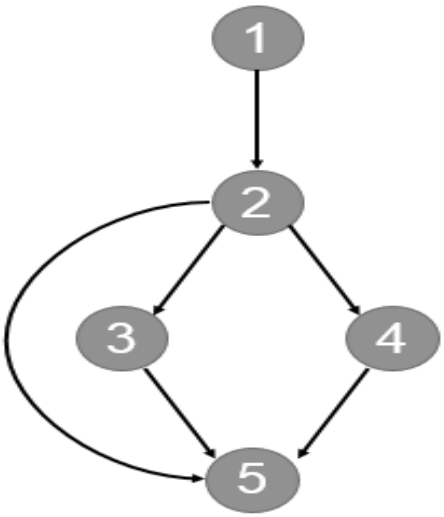


Figura 22 Grafo de flujo

```
public function obtenerSolicitudesPorFinalizar($inicio = null, $limite = null, $elementoOrdenar = null, $dirOrdenar = null, $filtros)
    if (isset($filtros->text_buscar))
        $filtros->text_buscar = $this->_ci->capital_humano_causa_baja_lib->valida_cadena($filtros->text_buscar);

    $filtros->rol_universitario_alta = (int) $this->_ci->xml->configuraciones->personal->estadoPersonaRolUniversitario->alta;

    $resul = $this->_ci->tb_dsolicitud_baja_mdl->obtenerSolicitudesPorFinalizar($inicio, $limite, $elementoOrdenar, $dirOrdenar, $filt
    $elemento = $this->_ci->ioc->seguridad->obtenerElementosDadoToken($this->_ci->session->userdata('token'));
    foreach ($resul as $value) {
        if ($value->id_persona_responsable == $elemento['id_persona']) {
            $value->puede_generar_doc = true;
        } else {
            $value->puede_generar_doc = false;
        }
        $value->fecha_registro = date_format(new DateTime($value->fecha_registro), "d/m/Y");
    }
    return $resul;
}
```

Figura 23 Código Obtener solicitudes por finalizar

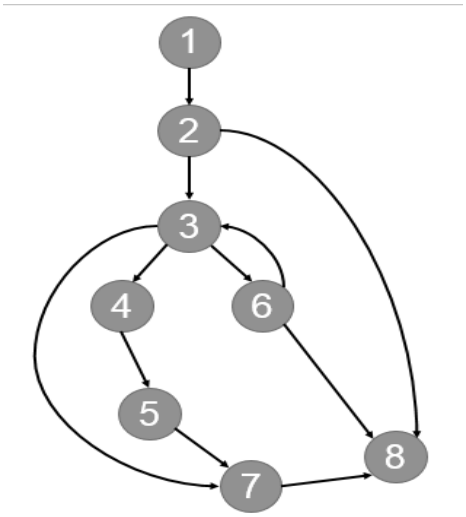


Figura 24 Grafo de flujo

```php
public function obtenerCapitalHumanoBajaPendiente() {
    $this->load->helper('grid');
    $post_vars = $this->input->all_post(true);
    $datos_buscador = new stdClass();

    if (isset($post_vars['cadena'])) {
        $datos_buscador = json_decode($post_vars['cadena']);
        $arrayPersonas = array('array_parametros' => $datos_buscador, 'guardados' => 'true');
        $this->session->set_userdata('personas_buscadas_personal', $arrayPersonas);
    } else {
        $arrayPersonas = $this->session->userdata('personas_buscadas_personal');

        if (!empty($arrayPersonas) && isset($arrayPersonas['array_parametros']) && !empty($arrayPersonas['array_parametros'])) {
            if (isset($arrayPersonas['array_parametros']) && !empty($arrayPersonas['array_parametros']))
                $datos_buscador = $arrayPersonas['array_parametros'];

            if ($arrayPersonas['guardados'] == 'true') {
                $datos_buscador->guardados = 'true';
                $arrayPersonas = array('array_parametros' => $datos_buscador, 'guardados' => 'false');
                $this->session->set_userdata('personas_buscadas_personal', $arrayPersonas);
            }
            else
                $datos_buscador->guardados = 'false';
        }
    }
```

Figura 25 Código Obtener bajas pendientes

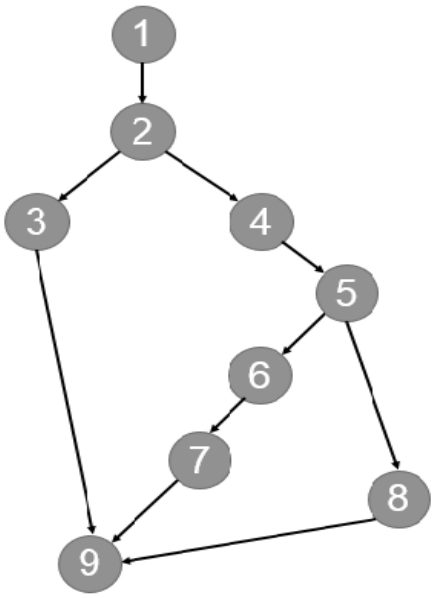


Figura 26 Grafo de flujo

Anexo 6: Pruebas de Integración

Caso de Prueba integración del módulo Personal
Sistema/módulo al que se integra: Seguridad.
Condiciones de ejecución: Se hayan introducido los datos en la base de datos central y exista conexión con la misma.
Descripción de la prueba: Comprobar que el módulo permite el acceso a la funcionalidad a partir de la información gestionada por el módulo Seguridad.
Pasos de ejecución: El módulo de Seguridad introduce en la base de datos central los datos y el módulo Personal consulta estos datos y define la seguridad para acceder al mismo.
Resultados esperados: Los usuarios tienen acceso a la funcionalidad de acuerdo a su rol y responsabilidades.
Evaluación: Prueba satisfactoria.

Tabla 19 Prueba de integración al módulo Seguridad

Caso de Prueba integración del módulo Personal

Sistema/módulo al que se integra: Estructura y composición.
Condiciones de ejecución: Se realiza la petición de los datos.
Descripción de la prueba: Comprobar que el módulo Estructura y Composición proporcione los datos referentes a la estructura organizativa definida por la universidad.
Pasos de ejecución: El controlador de la agrupación funcional Proceso de baja, realiza una petición a la librería para obtener los datos de las personas, gestionados por el módulo Estructura y Composición.
Resultados esperados: El módulo Estructura y Composición brinda la información solicitada.
Evaluación: Prueba satisfactoria.

Tabla 20 Prueba de integración al módulo Estructura y composición

Caso de Prueba integración del módulo Personal
Sistema/módulo al que se integra: Traza
Condiciones de ejecución: el usuario realiza una acción sobre la agrupación funcional.
Descripción de la prueba: Comprobar que son registradas todas las incidencias de un usuario sobre la agrupación funcional Proceso de baja.
Pasos de ejecución: Luego de realizada una acción, el sistema registra los datos de la incidencia: nombre de usuario, dirección IP, acción ejecutada, fecha de la ejecución, etc. Después se procede a buscar mediante el registro de trazas, la incidencia deseada.
Resultados esperados: Se muestran los datos de la traza registrada.
Evaluación: Prueba satisfactoria.

Tabla 21 Prueba de integración al módulo Traza

Anexo 7: Pruebas de sistema con la herramienta JMeter

Media	Mín	Máx	Std. Dev.	% Error	Rendimiento	Kb/sec
32334	11034	54570	12863.74	0.00%	3.6/sec	0.58
31422	22840	38889	4553.17	0.05%	2.8/sec	4.39
21586	12087	25403	3110.42	1.00%	4.1/sec	10.41
28447	11034	54570	9432.49	3.05%	6.3/sec	8.83

Tabla 22 Resultado con 200 usuarios para la funcionalidad Crear estado de solicitud

Media	Mín	Máx	Std. Dev.	% Error	Rendimiento	Kb/sec
21115	1774	38717	10349.80	0.25%	5.0/sec	6.94
19606	7861	34453	5543.17	2.00%	4.1/sec	12.17
20361	1774	38717	8336.15	2.25%	7.8/sec	17.19

Tabla 23 Resultado con 200 usuarios para la funcionalidad Crear causa de baja

Media	Mín	Máx	Std. Dev.	% Error	Rendimien...	Kb/sec
32869	3286	61986	14504.29	0.00%	3.2/sec	0.44
24368	6620	45150	8737.63	0.04%	2.5/sec	1.88
17273	3750	46893	9735.44	0.84%	2.6/sec	2.06
37559	1609	60456	17338.50	2.00%	2.1/sec	3.33
28017	1609	61986	15210.72	2.88%	6.0/sec	4.82

Tabla 24 Resultado con 200 usuarios para la funcionalidad Asociar áreas de visita obligatoria.

Media	Mín	Máx	Std. Dev.	% Error	Rendimiento	Kb/sec
37711	10151	58590	14260.75	0.00%	3.3/sec	0.34
34371	15490	57139	12587.94	0.08%	2.3/sec	1.10
19328	6556	46133	9649.48	0.60%	2.9/sec	10.09
17953	2832	47928	9840.32	0.40%	3.4/sec	4.00
27341	2832	58590	14672.00	1.08%	6.5/sec	8.57

Tabla 25 Resultado con 200 usuarios para la funcionalidad Mostrar solicitud de baja.