Universidad Central "Marta Abreu" de Las Villas

Facultad de Matemática, Física y Computación

Sistema Estudiante – GNU. Gestión de la Nueva Universidad.
Módulo de Estipendio

Autores

Dariem Pérez Herrera, José Jorge Lorenzo Vila

Tutores

M.Sc. Libia García Águila, Lic. Alcides Morales Guedes

Consultante

Lic. Yuniesky Alemán Caballero

Centro de Estudios de Informática
Santa Clara, Junio de 2006

... a mi madre, por su amor incondicional y por estar siempre a mi lado.

... a mi padre, que me abrió las puertas al mundo de la Computación

... a mi amigo Luis, que con su apoyo forjó el camino que me trajo hasta aquí.

Dariem

... a Dios, por acordarse y olvidarse de mi, por haberte encontrado, por haberme dado a eve, a misni y a milo, por hacerme tan feliz.

... a ti papi, por enseñarme a morir, porque cada día te extraño más.

... a ti mami, por enseñarme a vivir, porque cada día siento más tu amor y tu dolor.

... a ti mile, por cuidarme y quererme siempre, por cuidar tanto a papi y a abuelo, porque tu también has permitido que yo haya llegado.

... a mi familia, porque soy lo que hemos sido, porque la vida juntos ha sido distinta.

José Jorge

Resumen

La Universidad Central “Marta Abreu” de Las Villas y varias otras universidades han elaborado sistemas automatizados para controlar el estipendio estudiantil pero no existe uno unificado a nivel de Ministerio de Educación Superior. De hecho varios de estos sistemas son independientes y no siempre articulan con un sistema de control docente que tampoco ha existido de forma estable y generalizada para todos los centros. Los sistemas de control de estipendio se han implementado así a partir de bases de datos con plataforma y estructuras muy diferentes.

A partir del proyecto “Sistema Estudiante – GNU” ya ha sido diseñado y comienza a probarse en particular un nuevo Sistema de Control de Estudiantes, elaborado sobre plataforma libre y que responde a las características de la nueva universidad, incluyendo la continuidad de estudios. Este sistema de control de estudiantes deberá ser implantado próximamente en la totalidad de los Centros de Educación Superior del país.

El presente trabajo pretende desarrollar el Módulo de Control de Estipendio Estudiantil que articule con dicho Sistema de Control de Estudiantes de la Nueva Universidad.

Desde el punto de vista computacional este trabajo pretende desarrollar este módulo utilizando una arquitectura robusta, escalable y libre. Las herramientas utilizadas son de novedoso uso en nuestro ámbito.

Summary

Universidad Central "Marta Abreu" de Las Villas and many other universities have developed automated systems to control students stipend but there is not an unified one in Superior Education Ministry. In fact many of these systems are independent each other and they are not always related to a students control system, that it doesn't existed in stable release. Thus stipend control systems have been implemented using very different structures and platforms.

Beginning from the project "*Sistema Estudiante - GNU"*, it has been designed and proved a new Students Control System developed over free software platform. This system responds to the vision of the new university. The students control system should be deployed very soon at all the Superior Education Centers all over the country.

The present work intends to develop the Students Stipend Control Module related to the Students Control System. From the computational point of view it pretends to develop this module using robust, scalable and free architecture. The toolkit used is new in our ambit.

Índice

Introducción

Las tecnologías de la información van progresando cada vez más y la sociedad automatiza los procesos de negocio según sus necesidades para resolver problemas cotidianos. En la actualidad existe la necesidad de incrementar la eficiencia y calidad en los procesos que se realizan en los Centros de Educación Superior (CES) y el propio MES. Con este objetivo surgió en el año 2004 el proyecto SISGENU para elaborar un Sistema de Gestión de la Nueva Universidad: Sistema – GNU, porque en esos momentos el MES carecía de un sistema informatizado de la organización de la Educación Superior, cuestión que dificultaba la preparación de datos e informes de centros y ministerio, tanto para el Gobierno, como para uso interno. (Grupo Nacional de Diseño Sistema-GNU 2005)

Entre las primeras tareas que emergieron de este proyecto, apareció la necesidad de elaborar un Sistema de Control de Estudiantes que permitiera controlar la actividad docente de todos los CES y las SUM (Sedes Universitarias Municipales). Este sistema estaría compuesto, según el proyecto original de los siguientes módulos:

Orden	Nombre
1	**Matrícula (incluye módulo para SUM)**
2	**Estadísticas Versión 1.0**
3	**Secretaría Docente**
4	**Plan de Estudio**
5	**Profesor**
6	**Recuperación de Información Web**
7	**Estadísticas Versión 2.0**
8	**Alumno Ayudante**
9	**Estipendio**
10	**Estudiantes Extranjeros**

Orden	Nombre
11	**Planificación Docente**
12	**Archivo Histórico**
13	**Becas**
14	**Extracurricular**
15	**Guardia**

Sobre la base de la discusión en talleres nacionales de la prioridad de los sistemas a desarrollar, durante el primer año del proyecto se logró diseñar, programar, y comenzar a probar, seis módulos fundamentales de tal sistema: Matrícula (Martínez and Vallejo 2005), Control Docente (Gutiérrez and Pérez 2005), Plan de Estudio (Apezteguía 2005), Estadísticas del CES (Espinosa and Amador 2005), Estadísticas del MES (Pino 2005), Coordinador de Transacciones Distribuidas de Datos para Java (Saúco 2005).

La Universidad Central "Marta Abreu" de Las Villas participó desde su inicio en la concepción general del Sistema de Control Docente para la nueva universidad, en la definición de los codificadores fundamentales que requieren los CES, el MES y el CEPES, en el análisis de requerimientos del sistema y en el diseño de la base de datos. Ahora está llamada a contribuir sistemáticamente con el incremento de módulos para el mismo. Concretamente en el presente trabajo se pretende desarrollar una nueva aplicación sobre la base de los mismos principios del SISGENU para el Control del Estipendio Estudiantil requerido para los CES que imparten el tipo de curso regular diurno[1].

El desarrollo de este sistema fue encargado a la UCLV teniendo en cuenta entre otros aspectos, que tiene la experiencia del desarrollo y la explotación de un Sistema de Control de Estipendio automatizado. En particular, en la UCLV, el Sistema de Control de Estudiantes que está vigente actualmente está montado sobre SQL2000 (García 1995). El

[1] Los estudiantes de las SUM reciben normalmente salario como trabajadores, no estipendio estudiantil.

Sistema de Estipendio Estudiantil fue elaborado con ambiente *web* utilizando PHP (Alemán 2005). Dicho sistema ha sido muy bien evaluado por la última auditoría económica que recibió la UCLV; pero no puede ser generalizado a nivel de MES porque habría que generalizar también la estructura y plataforma de la base de datos sobre la cual se sustenta y el MES ha trazado otra política al respecto.

Otras universidades del país tienen también su propio sistema de estipendio pero no necesariamente articulados al sistema de control docente y mucho menos a los sistemas de control económico. Ellos además pueden estar elaborados sobre bases de datos y/o productos de software bien diferentes.

Una de las principales definiciones del proyecto SISGENU fue la de utilizar plataformas de software libre y multiplataformas. Esta es una necesidad que emerge del tránsito necesario de nuestro país hacia el uso del software libre, de compromisos internacionales que tiene Cuba en este sentido, además de brindar posibilidades de extensión de tales productos de software a otros países. Como en el caso de los anteriores sistemas de control docente, ninguno de los sistemas de control de estipendio actualmente vigentes en los CES del país, están elaborados sobre tales plataformas.

Entonces se plantea el siguiente Problema de Investigación:

No existe en nuestro país un Sistema de Control de Estipendio Estudiantil para los Estudiantes de la Educación Superior de Curso Regular Diurno(CRD), que responda a los intereses de todos los posibles usuarios, que tenga en cuenta las nuevas regulaciones que surgen de las concepciones de la nueva universidad, que articule con el Sistema de Control de Estudiantes que se pretende generalizar con el proyecto SISGENU, a veces con limitaciones en conectividad, y que en particular esté elaborado sobre las plataformas definidas por dicho proyecto.

Preguntas de investigación:

1. ¿Cuáles son realmente las necesidades de todos los posibles usuarios de un tal Sistema de Control de Estipendio Estudiantil, teniendo en cuenta las actuales concepciones de la nueva universidad y su posible evolución?
2. Será posible implementar un tal Sistema de forma relativamente sencilla que sea absolutamente compatible con el SISGENU y que responda a requerimientos tales como:
    a. Las particularidades de la nueva universidad relativas a los estudiantes de cursos regulares diurnos.
    b. El uso de plataformas *Open Source* compatibles con el SISGENU.
    c. Las posibilidades de responder tanto a un ambiente *web* como a un ambiente *desktop*.
    d. Las posibilidades de responder a los requerimientos de todos los posibles usuarios.

Objetivo general:

Desarrollar un Sistema de Control del Estipendio Estudiantil para la Educación Superior Cubana, usando Plataforma Libre, de manera que sea compatible con el Sistema–GNU, y que permita automatizar el control de los estipendios estudiantiles en cualquier Centro de Educación Superior del país por todas las dependencias o personas que estén relacionadas con el mismo.

Objetivos específicos:

- Analizar cuáles son las particularidades que la concepción de la nueva universidad impone o pueda imponer en un futuro inmediato sobre la gestión del estipendio estudiantil en cursos regulares diurnos.
- Desarrollar el Sistema de Control de Estipendios de manera que responda a una plataforma libre compatible con el SISGENU.

- Desarrollar tal sistema de manera que pueda responder a diferentes interfaces de usuario, permitiendo que pueda utilizarse en cualquier lugar del país en dependencia de las condiciones de conectividad.
- Responder a los requerimientos de todos los posibles usuarios del sistema, en particular, las Secretarías Docentes de las Facultades y la Dirección de Economía del CES.

Tareas de investigación:

Para responder a los objetivos de la investigación es necesario realizar las siguientes tareas:

1. Estudiar cuáles son las regulaciones actuales del MES sobre el estipendio estudiantil, en particular en CRD y las regulaciones alternativas o modificantes por las nuevas concepciones actuales de la universidad. Concebir un sistema automatizado previendo flexibilidad para un futuro más o menos inmediato.
2. Diseñar e implementar el nuevo sistema (módulo) para el control automatizado de estipendios, siguiendo como ejemplo de concepción y compatibilidad, los seis módulos que han sido previamente desarrollados por la CUJAE para el Sistema de Control de Estudiantes del SISGENU y además el Sistema de Control de Alumnos Ayudantes que se desarrolla paralelamente a este en la UCLV.

Concretamente, para la implementación de este Módulo de Control del Estipendio Estudiantil, se utilizará PostgreSQL como gestor de base de datos, JBoss™ como servidor de aplicaciones, J2EE™ como entorno distribuido aplicación servidor, de forma tal que sea totalmente compatible con el Sistema Estudiante – GNU, y que permita el control de los estipendios estudiantiles en cualquier Centro de Educación Superior del país por todas las dependencias o personas que estén relacionadas con el mismo, incluyendo el propio estudiante.

La presente tesis se estructura en tres capítulos:

En el Capítulo I. Elementos teóricos básicos para el desarrollo del trabajo, en el se expondrá el sustento teórico de nuestra investigación, que son las regulaciones para el pago de estipendio y la plataforma de desarrollo J2EE™.

En el Capítulo II. Consideraciones metodológicas en torno al diseño e implementación del sistema. Donde se definen los actores y casos de uso, se describen las entidades persistentes del sistema y la solución propuesta usando las herramientas seleccionadas.

En el Capítulo III. Manual de usuario. Es una guía detallada del uso del sistema con el propósito de que al módulo de la secretaria en las facultades y al módulo principal en la dirección de economía se les explote sus funcionalidades de forma eficiente.

Capítulo I. Elementos teóricos básicos para el desarrollo del trabajo

El análisis de los elementos teóricos que sustentan está investigación, hace necesario dividir el mismo en dos vertientes fundamentales: la primera de ellas, destinada al estudio del aparato legal que ampara las formas de pago a los estudiantes universitarios del curso regular diurno ya sean cubanos o extranjeros actuales en nuestro país, entiéndase por esto las Resoluciones y Normas Ministeriales que han sido emitidas con tal efecto; y la otra, encaminada al análisis de los principales elementos y tendencias tecnológicas existentes para el desarrollo del software que es objeto de nuestro estudio.

1.1 El Estipendio en Cuba

La forma de pago que determinados estudiantes universitarios reciben en Cuba bajo la categoría de Estipendio Estudiantil, está amparada por la Resolución 173/94 del Ministro de Educación Superior, emitida el 28 de noviembre de 1994. Existe además la norma No 34.2005 DCF sobre la contabilización de las operaciones de pago del Estipendio Estudiantil, en ella se puntualizan todas las resoluciones que modifican o ajustan las cuantías del estipendio estudiantil o establecen nuevas formas de pago que reciben los estudiantes que participan en actividades de la Batalla de Ideas.

En esencia la Resolución 173/94 dispone:

ARTÍCULO 9: Tienen derecho a recibir estipendio estudiantil:

a) Los becarios extranjeros.
b) Los estudiantes beneficiados de la Orden 18/79 del Ministro de Las Fuerzas Armadas Revolucionarias.

c) Los estudiantes externos matriculados en el curso escolar 1993-94, hasta la terminación de sus estudios, cuyo núcleo familiar perciba un ingreso per cápita mensual de cincuenta pesos ($ 50.00) o menos.
d) Los demás becarios de los centros de educación superior, que hayan estado matriculados en el curso escolar 1993-94 hasta la terminación de sus estudios de pregrado, o hasta que causen baja por cualquier motivo.

ARTÍCULO 10: Se podrá otorgar adicionalmente una subvención a los estudiantes sin amparo familiar.

ARTÍCULO 11: Los trabajadores matriculados en los cursos diurnos que reciban salario o estipendio del Ministerio de Trabajo y Seguridad Social, no tendrán derecho a recibir el estipendio estudiantil. Tampoco tendrán ese derecho los estudiantes que reciban estipendio de otro organismo estatal o institución. No obstante, podrán recibir el estipendio estudiantil, previa renuncia a los que recibían.

ARTÍCULO 14: Los estudiantes que causen baja, incluyendo aquellos con licencia de matrícula autorizada, dejarán de recibir estipendio a partir de la fecha de la misma.

ARTÍCULO 15: Los estudiantes de los cursos diurnos a los que se conceda licencias culturales y deportivas, de acuerdo a lo regulado, continuarán recibiendo durante el tiempo que duren las normas, el estipendio que recibían.

ARTÍCULO 16: Los estudiantes del curso regular diurno, a los que se haya aplicado una sanción disciplinaria que implique la separación de los estudios por cualquier término y se reincorporen cuando cumplan la medida, tienen derecho a recibir nuevamente el estipendio una vez autorizado su reingreso a la educación superior y mediante la aprobación de una nueva solicitud a la Comisión que corresponde.

ARTÍCULO 17: En el caso de los traslados autorizados, se procederá acorde a lo siguiente:

a) Cuando el traslado conlleve cambio de año, el estudiante recibirá el estipendio que corresponda al año que curse.
b) Cuando el traslado sea autorizado de cursos para trabajadores al curso diurno y el estudiante mantenga su vínculo laboral, devengando salario a otra subvención, no tendrá derecho a recibir estipendio estudiantil.
c) Cuando el traslado se autorice del curso diurno a cursos para trabajadores y el estudiante no se vincule a un centro laboral recibiendo salario, tendrá derecho a recibir estipendio estudiantil mientras se mantenga esa situación.

ARTÍCULO 20: El estipendio y la subvención se entregará a los estudiantes autorizados a recibirlo con una periodicidad mensual y ajustándose a la fecha que se indique el efecto.

ARTÍCULO 21: Los estudiantes están obligados a cobrar el importe de su estipendio o subvención mensual o ambos, en las fechas establecidas.

Al estudiante que le sea reintegrado el estipendio por no presentarse a cobrar el mismo en dos meses consecutivos, perderá automáticamente el derecho a recibir estipendio. Igualmente perderá este beneficio el estudiante que, durante el curso académico no haya efectuado el cobro en tres ocasiones.

ARTÍCULO 24: La asignación de un estipendio mensual a estudiantes que hayan estado matriculados en el curso 1993-94, estará en relación con el año que cursen incluyendo a becarios extranjeros en la forma siguiente:

a) Becarios:
   Primer año_____ $ 20.00 mensuales.
   Segundo año_____ $ 25.00 mensuales.

Tercero y Cuarto año_____ $ 30.00 mensuales.
Quinto y Sexto año_____ $ 40.00 mensuales.
Extranjeros_____ $ 100.00 mensuales.

Las cuantías reflejadas en el artículo anterior son las actuales pues las originales fueron sucesivamente modificadas por la resoluciones 125/99 y 10/2000, quedando finalmente como aquí se muestra.

ARTÍCULO 26: Disponer el pago de una subvención a los estudiantes incorporados al Movimiento de Alumnos Ayudantes o que realicen actividades similares que no formen parte de sus planes de estudio, de acuerdo con la etapa en que transiten, en la forma siguiente:

ETAPA DE ALUMNO AYUDANTE.
Durante el primer año de servicio_____$ 10.00 mensuales.
Durante el segundo año de servicio_____ $ 15.00 mensuales.

ETAPA DE INSTRUCTOR NO GRADUADO.
Durante el tercer año de servicio_____ $ 25.00 mensuales.
Durante el cuarto y quinto año de servicio_____ $ 35.00 mensuales.

ARTÍCULO 29: A todos los alumnos de la educación superior beneficiados por la Orden 18/79 de las Fuerzas Armadas Revolucionarias, que no perciban salario, se les otorga un estipendio de acuerdo al año que cursen, en la forma siguiente:

a) Primer Año_____ $ 50.00 mensuales.
b) Segundo Año_____ $ 65.00 mensuales.
c) Tercer Año_____ $ 80.00 mensuales.
d) Cuarto Año_____ $ 90.00 mensuales.

e) Quinto Año _____ $ 110.00 mensuales.
f) Sexto Año_____ $ 110.00 mensuales.

Las cuantías reflejadas en el artículo anterior son las actuales pues las originales fueron modificadas por la resolución 102/2000.

ARTÍCULO 32: Los estudiantes de la educación superior de los cursos regulares diurnos que lo requieran, puede otorgársele por un tiempo determinado (trimestre, semestre, curso), préstamos reintegrables una vez concluidos sus estudios e iniciada la vida laboral a partir de pagos mensuales conveniados.

ARTÍCULO 33: Los préstamos se realizarán de acuerdo a las necesidades del estudiante, que fundamentará ante la Comisión de Subvenciones a Estudiantes de la Facultad y hasta la cifra máxima de $50.00 mensuales.

ARTÍCULO 37: Los estudiantes que abandonen sus estudios, salvo las excepciones que se establezcan, deberán reintegrar el adeudo acumulado por el préstamo recibido, para lo cual se podrán establecer convenios de pagos aplazados.

ARTÍCULO 38: Con los estudiantes que se gradúen con préstamo otorgado, se procederá igual que en el artículo anterior, con excepción de los que se gradúen con una nota general de cinco puntos, a los que el monto del adeudo le será condonado, y a los que obtengan más de 4,5 puntos como promedio durante la carrera, que serán beneficiados con un cincuenta por ciento (50%) de rebaja.

En esencia la norma No 34.2005 DCF dispone:

- El estipendio estudiantil se rige por la Resolución 173/94.
- La resolución 125/99 modifica a $100.00 la cuantía del estipendio que reciben los extranjeros.
- La resolución 10/2000 del Ministro de Educación Superior modifica el artículo 24 ajustando las cuantías por año académico del Curso Regular Diurno.
- La resolución 102/2000 del Ministro de Educación Superior modifica el artículo 29 ajustando las cuantías por año académico de la orden 18.
- Por la orden 306/2001 del Ministro de las FAR se les impide recibir otra remuneración a los cadetes FAR.
- Por la orden 2/2005 del Ministro del Interior se les impide recibir otra remuneración a los cadetes MININT.

Básicamente estás son las disposiciones que conforman el aparato legal que rige el pago del estipendio estudiantil a los estudiantes del CRD y las reglas del negocio del sistema a desarrollar.

1.2 Estado del Arte

Para el desarrollo satisfactorio de cualquier aplicación como la que abordamos en esta investigación, se hace necesario partir de un estudio detallado de las principales corrientes actuales en cuanto al desarrollo de un software.

Como consecuencia de la manera en que el procesamiento computarizado de datos ha sido introducido en el mercado mundial, ha prevalecido entre los usuarios el llamado *software* propietario. Este consiste en sistemas expresados en código inaccesibles para el usuario.

Según la Compilación de Ensayos sobre el Software Libre (Stallman, Olivera et al. 2004), las principales desventajas que acarrea la utilización de *software* propietario, las podemos resumir de la siguiente forma:

- Posibilidad de que existan funcionalidades no deseadas en dicho *software*. Dependiendo de la programación realizada, algunas funcionalidades podrán ser activadas o desactivadas por el usuario, pero pueden existir también funcionalidades que no se puedan desactivar o que, incluso, no se encuentren documentadas. Llevándolo al extremo se podría hablar de "puertas traseras" abiertas por el fabricante del *software* que, después de todo, es un agente comercial y, por tanto, tiene sus propios intereses que pueden ser contrarios a los de la compañía que instala un *software* de seguridad específico.

- Desconocimiento del código por parte del usuario. Esto puede llevar a que el fabricante pueda llegar a tener una falsa sensación de seguridad por oscuridad, es decir, las vulnerabilidades de su producto no tienen por qué ser conocidas porque nadie tiene acceso a las "tripas" del mismo. De igual forma, esto puede llevar a que el fabricante no tenga interés en desarrollar el código de una forma adecuada porque, al fin y al cabo, el usuario no va a ver dicho código ni evaluar la calidad de su implementación.

- Necesidad de confiar totalmente en el fabricante. Esto es así por cuanto éste ha implementado los algoritmos de seguridad y el usuario no puede garantizar por sí mismo que su implementación ha sido correcta y que, por ejemplo, las propiedades matemáticas necesarias para que estos algoritmos funcionen correctamente se cumplan en todas las condiciones.

- Dependencia de una tercera entidad, ya que es el fabricante del producto es el único que puede ofrecer nuevas versiones de éste en caso de fallo o incluir nuevas funcionalidades que puedan ser necesarias. Esto es una desventaja debido a que el usuario no puede transferir esta dependencia a otra entidad, en caso de que el fabricante original haya traicionado su confianza (demasiados errores en la

implementación, demasiado tiempo en la generación de parches para arreglar problemas graves, etc.)

Estos inconvenientes han llevado a que un gran número de desarrolladores han migrado a plataformas de desarrollo libres.

1.2.1 Software Libre

El *software* libre nació de la mano del propio *software* en la década de los años 60. Entonces las gigantescas máquinas a las que llamaban computadoras hacían uso de programas cuyo código fuente estaba a la vista de todos y se podía distribuir libremente. Esto provocó que ya en esos tiempos, "prehistóricos" desde el punto de vista de la informática, existiera una pequeña comunidad de científicos y programadores que intercambiara código, informes de errores e ideas. (Stallman, Olivera et al. 2004)

El *software* libre, tal y como lo conocemos hoy, dio sus primeros pasos a mediados de los años 80 con un manifiesto en favor de la libertad de expresión y un proyecto conocido hoy mundialmente, el proyecto **GNU(**acrónimo recursivo de **G**NU's **N**ot **U**nix**)** (Stallman 1985) . Con él, vio la luz probablemente una nueva forma de ver y entender el *software*, que se ha visto acelerada con la masiva implantación de *Internet* en las postrimerías del siglo XX y principios del actual.

Según el mundialmente conocido Manifiesto GNU, la palabra "libre" se refiere a libertad, no a precio. Puedes o no pagar un precio por obtener *software* GNU. De cualquier manera, una vez que obtienes el *software*, tienes tres libertades específicas para usarlo:

- la libertad de copiar el programa y darlo a tus amigos o compañeros de trabajo.
- la libertad de cambiar el programa como desees, por tener acceso completo al código fuente.
- la libertad de distribuir una versión mejorada ayudando así a construir la comunidad.(Stallman 1985)

La concepción del Software Libre, permite que varios proyectos sean desarrollados y sostenidos por comunidades de programadores del mundo entero, mejorando sustancialmente las garantías de estabilidad y confiabilidad del producto.

El hecho de que el *software* libre, pueda ser utilizado libremente, entiéndase sin solicitar permiso alguno a nadie, hace que su utilización sea ideal. Para nuestro país es vital porque es la forma de evitar las absurdas restricciones que nos impone el bloqueo, para adquirir legalmente el *software* propietario, incluso si estuviésemos en condiciones de pagarlo.

1.2.2 Aplicaciones Empresariales y J2EE™

Una aplicación empresarial es un *software* que automatiza procesos o mecanismos específicos de una empresa u organización. En la mayoría de los casos se encuentra alojada en un servidor y es utilizada por múltiples usuarios de la misma organización, proporcionándole servicios de forma simultánea.

Según la documentación oficial del estándar J2EE™ (Armstrong, Ball et al. 2005), una aplicación empresarial correctamente diseñada e implementada, debe cumplir con los siguientes requisitos:

- Robustez: como este tipo de *software* son productos esenciales en la automatización de un proceso, se espera que sea confiable y esté exento de errores.

- Excelente desempeño y escalabilidad: deben cumplir con las expectativas de funcionamiento esperadas por sus usuarios, teniendo en cuenta el *hardware* apropiado. Es vital este punto, pues en la mayoría de los casos, no es posible predecir el número de usuarios simultáneos a los cuales hay que servir con la aplicación.

- Explotación de la Programación Orientada a Objetos: dicha programación ofrece beneficios indudables al desarrollo de sistemas complejos.

- Niveles de complejidad adecuados: un buen análisis asegura no tener una vista simplista del proceso a automatizar a la vez que elimina una complejidad excesiva que indica igualmente un uso incorrecto de la arquitectura.

- Fácil mantenimiento y extensibilidad: es la fase más costosa en el ciclo de vida del *software* y por lo tanto debe tenerse en cuenta en la etapa de diseño. Se debe garantizar la correcta modularidad, de forma tal que la modificación de algún componente no afecte sobremanera a los demás.

- Re-usabilidad: una aplicación empresarial debe estar incluida en los planes a largo plazo, de automatización de los procesos de la organización, por lo tanto debe permitir que la automatización de otros procesos afines con el ya automatizado, aproveche las ventajas del trabajo ya realizado.

En los últimos años, el desarrollo de la actividad empresarial está siendo cada vez más sustentado por la automatización de las mismas. Frente a esta demanda surgen dos plataformas para el desarrollo de aplicaciones capaces de cumplirla: J2EE™ de Sun Microsystems ® y .NET de Microsoft Corporation ®.

En los aspectos principales, las dos plataformas son muy parecidas, la principal diferencia que nos ocupa es quizás el énfasis de las grandes compañías sobre el estándar J2EE™ y el hecho de que existen numerosas implementaciones libres de gran calidad de esta plataforma estándar, razón por la cual, nos detendremos en el análisis de la misma, pues constituye objeto de nuestro estudio.

J2EE™ es esencialmente un entorno distribuido aplicación-servidor, especifica tanto la infraestructura para gestionar las aplicaciones, como las interfaces de programación (API por sus siglas en inglés) para construirlas. La arquitectura puede ser dividida en cinco elementos fundamentales:

1. El lenguaje de programación Java.
2. El modelo de programación del cliente.
3. La infraestructura de la capa de *middleware*.
4. Las interfaces de programación de negocios para los programadores
5. Las interfaces de programación no visibles para los programadores.

El modelo de funcionamiento de las aplicaciones empresariales bajo J2EE™ es el siguiente:

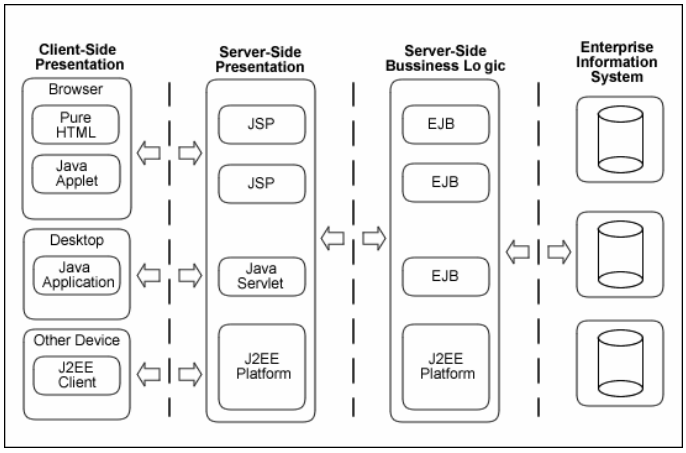


Figura 1.1 Modelo de Funcionamiento de J2EE

J2EE™ concibe a una aplicación empresarial como un conjunto de capas que se interrelacionan entre sí, en dichas capas se encuentran los componentes de la aplicación agrupados por roles o funcionalidades en el sistema. Las principales capas que conforman a una aplicación empresarial en J2EE™ y que se muestran en la Figura. 1.1 son:
La Capa de Presentación del lado del Cliente, la Capa de Presentación del lado del Servidor, la Capa de Negocio y la Capa del Sistema de Información Empresarial.

1.2.2.1 Componentes J2EE™

Los componentes J2EE™ son unidades de *software* auto contenidos y funcionales que son ensamblados dentro de la aplicación empresarial junto con sus clases y ficheros y son capaces de comunicarse con otros componentes. El estándar J2EE™ define los siguientes componentes:

- Las aplicaciones clientes y las *applets* son componentes que se ejecutan en el cliente.
- Los *Java Servlet* y los componentes pertenecientes a la tecnología *JavaServer Pages*™ (JSP en la Figura 1.1) son componentes *web* que se ejecutan en el servidor.
- *Enterprise JavaBeans* ™ (EJB en la Figura 1.1) son componentes del negocio que se ejecutan en el servidor.

Dichos componentes son programados en Java y compilados de la misma manera que los demás programas en este lenguaje. La diferencia entre los componentes J2EE™ y una clase estándar de Java, radica en el hecho de que los componentes son ensamblados dentro de una aplicación J2EE™, son verificados contra la especificación J2EE™ y son desplegados en el servidor de aplicaciones. A continuación explicamos el rol esencial de cada tipo de componentes.

1.2.2.2 Aplicaciones Clientes

Las aplicaciones clientes se ejecutan en la máquina cliente y proporcionan el camino a los usuarios para manipular las tareas. Típicamente están compuestas por una interfaz gráfica de usuario (GUI por sus siglas en inglés) creadas con las interfaces de programación que brindan las bibliotecas de clases *Swing* o *Abstract Window Toolkit* (AWT), aunque es válida también una interfaz por línea de comando.

La aplicación cliente directamente accede a los *Enterprise Java Bean*™ (EJB™) que se encuentran desplegados en la capa de negocio del servidor, o puede suceder que la aplicación cliente inicie una conexión *http* para comunicarse con un *servlet* que se encuentra ejecutándose en la capa de presentación del lado del servidor. Particularmente en esta investigación se utilizó la primera variante, mediante la Invocación de Procedimientos Remotos (RMI).

1.2.2.3 Enterprise Java Bean™

Un *Enterprise Java Bean*™ (EJB™) representa un objeto con las características de acceso seguro, por roles, transaccional, con balance de carga y, cuando se desea, persistente. Los EJB™ radican en el Contenedor EJB™ del servidor de aplicaciones. Escritos en Java, los EJB™ son componentes que encapsulan la lógica del negocio de una aplicación.

Se pueden clasificar según su propósito general en:

- *SessionBean*: en sus métodos se implementa la lógica del negocio.
- *EntityBean*: representa un objeto del negocio que existe almacenado de forma persistente.

1.2.2.4 SessionBean

Los *SessionBean* son las fachadas del modelo; establecen la capa de enlace entre la de presentación y la de negocio. Se clasifican según su capacidad para mantener sus estado: *Stateful SessionBean* (mantienen su estado durante la sesión) y *Stateless SessionBean* (no poseen estado alguno). En ellos se codifican casi en su totalidad los métodos del negocio utilizados en la aplicación.

Una de las características fundamentales que hace preferir la utilización de los EJB™, es precisamente el manejo automático y transparente de las transacciones por parte de los contenedores, característica perteneciente a los EJB™ y denominada *Container-Managed*

Transactions (CMT). El código referente a las transacciones no tiene que ser escrito por el desarrollador porque es definido declarativamente en los descriptores de despliegue del componente.

1.2.2.5 EntityBean

Los *EntityBean* son componentes que representan una vista de entidades persistidas en un almacenamiento permanente. Son objetos distribuidos, compartidos, transaccionales y persistentes. Estas características se ajustan al hecho de que los *EntityBean* son los componentes de acceso a datos y no deben estar involucrados en la lógica del negocio.

Pueden ser clasificados en la forma de manejar la persistencia en:

- *Container Management Persistente* (CMP): el contenedor EJB™ del servidor de aplicaciones maneja la persistencia.
- *Bean Management Persistente* (BMP): el componente maneja su persistencia y por tanto el desarrollador es responsable de implementar los métodos para tal efecto.

Pueden ser clasificados en la forma que realizan la abstracción de los datos en:

- Grano Fino: los datos que muestra cada componente representa una fila de una tabla persistente.
- Grano Grueso: los datos que muestra cada componente representa un registro lógico que toma datos de varias tablas.

1.2.3 Servidor de Aplicaciones y Jboss™

Un servidor de aplicaciones no es más que un servidor que se encarga de alojar aplicaciones. Es el responsable de brindar una serie de servicios a otros componentes alojados en él para que sean utilizados por las aplicaciones que se encuentran en el mismo.

El servidor que se utilizó para alojar el sistema a desarrollar fue JBoss™, debido a que es libre, gratuito, cumple de forma excelente con la especificación J2EE™ y está respaldado por una gran comunidad de programadores. Está basado en Java, por lo tanto es multiplataforma.

Entre los servicios ofrecidos por este Servidor de Aplicaciones están:

- CMP: este servicio brindado por el servidor se responsabiliza de establecer toda la comunicación entre el *bean* de entidad y el ambiente persistente (una Base de Datos o un simple fichero XML). El desarrollador debe decidir si utiliza este mecanismo o se responsabiliza él de la persistencia de los datos.

- CMR: El contenedor maneja las relaciones que se establecen entre los *beans* de entidad que pertenecen a él. Puede ser relaciones de tipo uno a uno, uno a muchos y muchos a muchos; además pueden clasificarse en unidireccionales o bidireccionales. Este servicio permite modelar las relaciones sin ser definidas en el sistema gestor de base de datos y elevar dichas relaciones a un nivel superior de abstracción de los datos, al ser definidas entre objetos o componentes del sistema y no por tablas.

- Servicio de Transacciones: la arquitectura de JBoss™ permite utilizar cualquier implementación de las interfaces de programación de transacciones para java (JTA™ por sus siglas en inglés). Además incluye por defecto la implementación más rápida del manejador de transacciones compatible con JTA™.

Como ya se explicó anteriormente, los *beans* de entidad manejan la persistencia de forma automática. Aunque este mecanismo aísla a la aplicación de la fuente de datos, se hace necesaria la selección de un Sistema Gestor de Base de Datos Relacional donde residirán los datos relativos a la aplicación.

1.2.4 Sistemas Gestores de Base de Datos y PostreSQL

Hoy en día, son muchas las aplicaciones que requieren acceder a datos, bien sea un sencillo programa doméstico, bien una suite para la gestión empresarial.

Estos datos se deben almacenar en algún soporte permanente, y las aplicaciones deben disponer de un medio para acceder a ellos. Normalmente, la forma en que un programa accede a un fichero es a través del Sistema Operativo.

Un Sistema Gestor de Bases de Datos (SGBD) es entonces el software capaz de proporcionar una interfaz entre aplicaciones y Sistema Operativo; para conseguir, entre otras cosas, que el acceso a los datos se realice de una forma más eficiente, más fácil de implementar, y sobre todo, más segura.

Según la documentación oficial (The PostgreSQL Global Development Group 2003), PostgreSQL es un sistema gestor de bases de datos relacional, de código abierto, descendiente de Postgres que fue desarrollado en la Universidad de Berkeley.

Entre las características que tiene se encuentran:

- Consultas complejas.
- Llaves extranjeras.
- Disparadores.
- Vistas.
- Integridad transaccional.
- Acceso concurrente multiversión.

Es un sistema gestor libre y gratis, cumpliendo las nuevas estrategias de nuestro país y del proyecto SISGENU en particular. Por el hecho de ser de uno de los mejores gestores de bases de datos relacionales, fue elegido para el desarrollo del sistema.

1.2.5 Estrategias de desarrollo

Estrategia de Desarrollo, en el mundo informático, no es más que una serie de modos y actividades que se realizan para crear un software. Define, por tanto, un modo de hacer las cosas.

Una vía que promete acelerar el desarrollo de aplicaciones, simplificar la integración entre distintas tecnologías y reducir el costo de la migración de las aplicaciones a nuevas plataformas, lo es, sin dudas, *Model Driven Architecture* (MDA).

1.2.6 Model Driven Architecture

El *Object Management Group* (OMG) es un consorcio de empresas de informática creado en el año 1990, con el objetivo de potenciar el desarrollo de aplicaciones orientadas a objeto y distribuidas. Para ello, desde un principio se prestó especial atención al problema de la interoperatividad e integración de sistemas, lo que ha llevado al OMG a definir numerosas especificaciones y estándares como CORBA, UML, MOF, XMI y CWM. En el año 2001, el OMG estableció el *framework* MDA como arquitectura para el desarrollo de aplicaciones. (Soley 2004)

En la actualidad, la construcción de *software* se enfrenta a continuos cambios en las tecnologías de implementación, lo que implica esfuerzos importantes en el diseño de la aplicación para integrar las diferentes tecnologías de implementación. Por otra parte, las

aplicaciones distribuidas *Business to Business* (B2B) y *Client to Business* (C2B) son cada vez más comunes, por lo que resulta difícil satisfacer los requisitos de escalabilidad, seguridad y eficiencia. La idea clave que subyace a MDA es que si el desarrollo está guiado por los modelos del *software*, se obtendrán beneficios importantes en aspectos como son la productividad, la portabilidad, la interoperabilidad y el mantenimiento.

Para conseguir estos beneficios, MDA plantea el siguiente proceso de desarrollo: de los requisitos se obtiene un modelo independiente de la plataforma (PIM por sus siglas en inglés), luego este modelo es transformado con la ayuda de herramientas en uno o más modelos específicos de la plataforma (PSM por sus siglas en inglés), y finalmente cada PSM es transformado en código. Por tanto, MDA incorpora la idea de transformaciones entre modelos (PIM a PSM y PSM a código), por lo que se necesitarán herramientas para automatizar esta tarea. Estas herramientas de transformación son, de hecho, parte de los elementos básicos de MDA. (The AndroMDA Core Team 2006)

Una de estas herramientas es AndroMDA, una plataforma MDA *open source*, la cual utiliza modelos, usualmente modelos UML almacenados en XMI (ficheros XML de intercambio) producidos por herramientas *case*, los combina con *plug-ins* (librerías de traducción o cartuchos) y produce los componentes adecuados según el modelo. (The AndroMDA Core Team 2006)

En la actualidad AndroMDA es utilizada principalmente por los desarrolladores que trabajan con tecnologías J2EE™. Entrando en detalles, AndroMDA puede crear un proyecto J2EE™ desde cero, en el cual el código es generado a partir del modelo UML. El código generado es integrado automáticamente en el proceso de compilación. Es muy eficiente en la generación de código, permitiendo que el desarrollador se mantenga enfocado en la capa de negocio.

1.2.7 Arquitectura de plug-ins

La arquitectura de *plug-ins* permite extender las funcionalidades de la aplicación sin necesidad de tener acceso al código fuente de la misma, tampoco es necesario recompilarla para redistribuirla a los usuarios, basta con distribuir el nuevo *plug-in*.

Este tipo de arquitectura resulta ser muy atractiva para los desarrolladores porque permite centrarse en proveer una funcionalidad modular al usuario final. Este enfoque resulta además muy beneficioso a la hora de manejar el problema de que las reglas de negocio pueden cambiar frecuentemente o bien pudieran surgir reglas nuevas. De esta forma se puede personalizar fácilmente una aplicación mezclando y acoplando *plug-ins* según se necesite o creando uno nuevo para alguna opción inexistente.(Birsan 2005)

En la arquitectura de *plug-ins* tradicional estos no son compilados dentro de la aplicación servidora, son enlazados a través de interfaces bien definidas y la aplicación puede organizar y activar a conveniencia las funciones que implementan. (Birsan 2005)

En la arquitectura de *plug-ins* pura, que es la que se ha usado en este trabajo, todo es un *plug-in* (figura 1.2). La aplicación servidora queda reducida entonces a un motor de ejecución de *plug-ins* sin ninguna funcionalidad inerte, donde cada *plug-in* se ejecuta bajo las reglas definidas por el motor y por él mismo. Para garantizar la infraestructura básica de *plug-ins*, este motor de ejecución busca, carga y ejecuta el código correcto, administrando el modelo de extensión y las dependencias (Birsan 2005). La aplicación base por sí sola, no muestra más que un marco vacío para incorporar elementos de interfaz de usuario, como pueden ser barras de tarea, menús, diálogos, conjunto de acciones, etc. Resulta ser entonces, pequeña y simple, pero lo suficientemente robusta para soportar tanta extensión de las funcionalidades como sea requerida, tomando el menor esfuerzo posible.

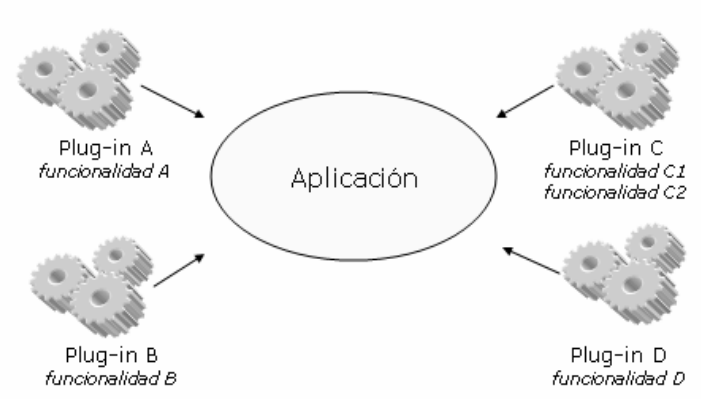


Figura 1.2 Arquitectura de plug-in.

La estructura física de la aplicación se muestra en la Figura 1.3.

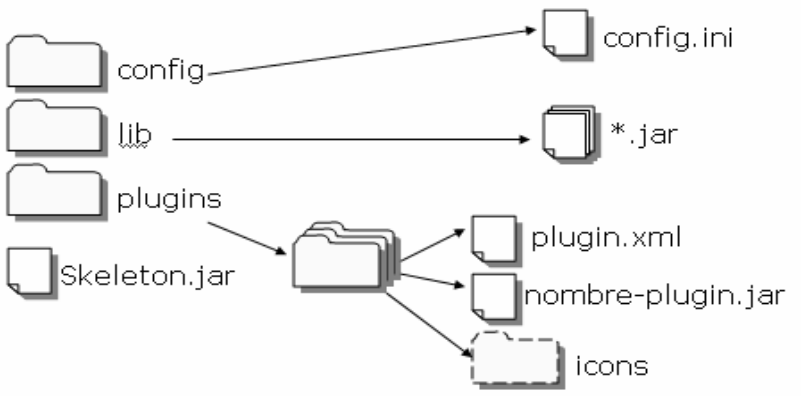


Figura 1.3 Estructura física de la aplicación cliente.

Ficheros	Descripción
config\config.ini	Configuración de la aplicación (Servidor de Aplicaciones y parámetros necesarios para establecer conexión con el Servidor de Base de Datos mediante JDBC)
lib*.jar	Librerías de clases utilizadas por los distintos *plug-ins*
plugins\	*Plug-ins* activos en la aplicación
Skeleton.jar	Cargador de la aplicación

1.2.8 Patrones de diseño

Los Patrones de Diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces. Seguidamente se explicaran brevemente en que consisten los principales patrones que se utilizaron el desarrollo del sistema.

1.2.8.1 Fachada de Sesión

Típicamente, para ejecutar un caso de uso, son accedidos y posiblemente modificados múltiples objetos de entidad del lado del servidor. Un problema, consiste en que tantas llamadas y transacciones provocan una posible sobrecarga en la red, lo que puede comprometer el rendimiento de la aplicación. Por otro lado este tipo de diseño resulta ser poco mantenible, ya que tanto los datos como el flujo de trabajo y la lógica del negocio se acceden directamente por los clientes, creando una estrecha dependencia entre ellos, por esta misma razón la reusabilidad se reduce considerablemente al no poderse reutilizar una misma lógica por clientes diferentes. Una práctica común en proyectos de gran escala, es separar las tareas de los desarrolladores de la lógica de presentación (*front-end programmers*), de las de los programadores de la lógica del negocio (*middleware programmers*). Si la lógica del negocio es codificada en la capa cliente, esta clara separación por roles no es posible. Se hace necesario entonces un nivel de abstracción del lado del servidor que actúe como intermediario para acceder a los objetos de entidades. (Marinescu 2002)

El patrón fachada de sesión aplica los beneficios del patrón de fachada tradicional a los EJB™ ocultando por completo el modelo de objetos en el servidor de la capa cliente, teniendo una capa de objetos que brindan servicios (*sessionbean*) como único punto de acceso para el cliente (Figura 1.4). La fachada permite encapsular el flujo y la lógica del negocio de los casos de uso.(Marinescu 2002)

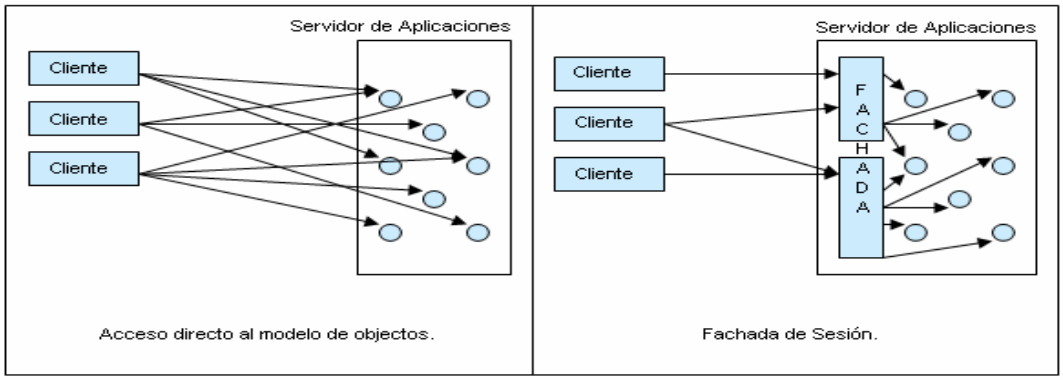


Figura 1.4 Beneficios de la fachada de sesión para la arquitectura.

La fachada de sesión es el patrón fundamental relacionado con los EJB™ hoy en día, no solo provee un beneficio en el rendimiento y permite elaborar un diseño eficiente y reutilizable, sugiere además un estándar de arquitectura para desarrollar aplicaciones J2EE™ donde el cliente y el servidor quedan separados por una capa de *sessionbean*, cuyos métodos ejecutan la lógica de todos los casos de uso de la aplicación. (Marinescu 2002)

1.2.8.2 Value Objects

Usualmente la lógica de negocio modifica grupo de entidades y con ello el tráfico de información por la red se dificulta, para optimizar esta transferencia se implementó el patrón de diseño *value object* el cual consiste en una clase plana de java que encapsule todos los datos a transmitir a través de la red o entre las capas del sistema. (Marinescu 2002)

1.2.8.3 Model-View-Controller

Es un patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

- **Modelo**: Esta es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos.
- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.
- **Controlador**: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

En MVC, la capa de presentación está partida en controlador y vista. La principal separación es entre presentación y dominio.

Aunque se pueden encontrar diferentes implementaciones del patrón MVC, el flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma.
2. El controlador recibe la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

1.2.8.4 Singleton

Otro de los patrones que se utilizó en la implementación del sistema es el *singleton*, principalmente en las clases controladoras, para mejorar el rendimiento y lograr mayor consistencia en la comunicación entre las capas de la aplicación.

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

El patrón *singleton* provee un única instancia global gracias a que:

- La propia clase es responsable de crear la única instancia.
- Permite el acceso global a dicha instancia mediante un método de clase.
- Declara el constructor de clase como privado para que no sea instanciable directamente.

Las clases controladoras se implementaron utilizando este patrón de diseño.

1.2.9 JasperReports e IReports

Los reportes económicos se generan utilizando JasperReports y se diseñan utilizando iReports.

JasperReports es una poderosa librería *Open Source* para la elaboración de reportes, escrita completamente en Java y que puede ser utilizada en diferentes aplicaciones Java incluyendo J2EE™ o Aplicaciones Web, para generar contenido dinámico. (Danciu 2005)

JasperReports trabaja de forma similar a un compilador y un intérprete, usa un fichero XML definido por el usuario con el diseño del reporte, lo compila y crea un fichero *jasper* y conjuntamente con una interfaz especial de JasperReports llamada JRDataSource obtiene los resultados del reporte para luego ser mostrados usando la librería de JasperReports que permite mostrar los reportes en diferentes ambientes y luego llamarlos en la aplicación. (Toffoli 2005)

La herramienta iReport es un constructor/diseñador de informes, visual, poderoso, intuitivo y fácil de usar para JasperReports, y está escrito totalmente en Java. Este instrumento permite que los usuarios corrijan visualmente informes complejos con cartas, imágenes, sub-informes.

IReport está además integrado con JFreeChart, una de las bibliotecas gráficas *Open Source* más difundida para Java. Los datos para imprimir pueden ser recuperados por varias vías, incluso múltiples uniones JDBC, *TableModels*, *JavaBeans*, XML, etc.

Como se explica en Introducción a iReport (Basurto 2005), las características más relevantes de iReport son:

- 100% escrito en Java y además *Open Source* y gratuito.
- Maneja el 98% de las etiquetas de JasperReports.
- Permite diseñar con sus propias herramientas: rectángulos, líneas, elipses, campos de los *textfields*, cartas, y sub-reportes.
- Soporta internacionalización nativamente.
- Recopilador y exportador integrados.
- Soporta JDBC.
- Incluye asistentes para crear automáticamente informes.
- Tiene asistentes para generar los sub-reportes.

1.3 Resumen del capítulo

En este capítulo se han expuesto las normas que rigen el proceso de pago de estipendio estudiantil en la Educación Superior.

Resulta importante señalar que la elección de la tecnología y en general la metodología utilizada garantiza la novedad y el carácter científico de la presente investigación, ya que es el resultado del estudio de las tendencias actuales en cuanto al desarrollo de las aplicaciones empresariales, herramientas que presentan una serie de ventajas, justificando su uso en cada una de las soluciones propuestas que trataremos en el capítulo siguiente.

Capítulo II. Consideraciones metodológicas en torno al diseño e implementación del sistema.

La revisión bibliográfica además de iniciar la investigación, se llevó a cabo durante todo el proceso de desarrollo. Esta tarea estuvo encaminada, en mayor medida, a la elaboración del marco teórico y permitió realizar análisis críticos de la investigación.

Otro momento importante lo constituyó el estudio de los procesos que definen el estipendio estudiantil. Para ello se contó con la valiosa ayuda del Lic. Yuniesky Alemán Caballero, desarrollador del actual Sistema de Estipendio y adiestrado en la Dirección de Economía de la Universidad Central de las Villas.

Con los procesos que definen el estipendio, como objeto de estudio y con la comprensión teórica del proceso que lo sustenta, se ya podía comenzar a desarrollar el sistema.

El desarrollo del mismo fue dividido en tres etapas fundamentales:

1. Selección del entorno y herramientas de desarrollo, manteniendo la compatibilidad con el Sistema GNU.
2. Modelado del sistema.
3. Implementación del sistema sobre las herramientas antes seleccionadas.

2.1 Selección del entorno y herramientas de desarrollo

La selección de la plataforma sobre la cual se desarrollaría la aplicación, partió de la premisa de que ésta debía desarrollarse sobre *software* no propietario, pues en nuestro país hay una política a nivel de gobierno de migración hacia el *software* libre. Quedó descartado así .Net®, una de las principales tecnologías actuales, pues las herramientas libres que implementan dicho estándar, están solo en etapa de desarrollo.

El debate técnico que se llevó a cabo por el equipo de trabajo del proyecto con el MES y colaboradores favoreció a J2EE™ sobre PHP. Los principales motivos que apoyan esta decisión son:

1. La escalabilidad que se puede lograr con la utilización de EJB™ es muy superior que la que se puede lograr utilizando *scripts* de PHP.
2. Con la utilización de Java Empresarial es posible utilizar una arquitectura multicapa e implementar correctamente el patrón Modelo-Vista-Controlador (MVC por sus siglas en inglés) que separa el diseño de la aplicación.
3. Con Java es posible agrupar lógicamente bajo un mismo espacio de nombre, un conjunto de clases, conformando así una biblioteca de clases.
4. El uso de EJB™ permite desplegar en un servidor de aplicaciones a las entidades persistentes y a las que manipulan la lógica del negocio, de manera que puedan ser utilizados tanto por un cliente WEB como Stand-Alone.
5. El mecanismo de documentación de Java para el *software* desarrollado es superior, mediante *tags* embebidos en el código fuente, se genera la documentación utilizando el estándar *web*.

Como servidor de aplicaciones, se decidió utilizar a Jboss™ por ser la implementación libre más completa del estándar J2EE™.

Definida la plataforma, queda aun la selección de las herramientas libres para el desarrollo de la aplicación, de manera que se simplificara el proceso de implementación y hacerlos flexible a cambios en los requerimientos. Se decidió utilizar entonces el paradigma de desarrollo MDA. La herramienta libre que genera una aplicación J2EE™ siguiendo el patrón MVC indicada por su potencial y utilización fue AndroMDA. Seleccionadas todas las herramientas, solo quedaba la modelación e implementación de la aplicación.

2.2 Modelación de la Aplicación

A continuación se realiza el análisis de los requerimientos y el diseño de la aplicación como partes de la modelación de la misma.

2.2.1 Análisis de requerimientos

En esta primera etapa contamos con la importantísima colaboración del desarrollador del actual sistema de estipendio funcional en la UCLV. Definimos los principales conceptos de nuestro dominio de aplicación. Nos pusimos en contacto con las diferentes resoluciones y disposiciones ministeriales, las que constituyen el aparato legal que rige el proceso de estipendio. Los resultados de este análisis ya han sido referidos en la primera parte del Capítulo I.

2.2.2 Diagramas de Casos de uso del Sistema

Los diagramas de casos de usos delinean gráficamente el comportamiento del sistema. Representan como el sistema es usado desde la perspectiva del actor. Pueden bosquejar todos o algunos casos de uso del sistema. Representan con un alto nivel de detalle como va a ser utilizado el sistema por los actores, además contiene interacciones y relaciones entre casos de uso o actores. (Rational Software Corporation 2003)

El proceso de la gestión del estipendio estudiantil, tiene dos etapas fundamentales, la primera ocurre en la dirección de economía del CES y la segunda en las distintas facultades que forman dicho CES.

A continuación (Figura 2.1 y Figura 2.2) se muestran los diagramas de casos de uso del sistema para cada uno de los actores que intervienen en las dos etapas anteriormente mencionadas.

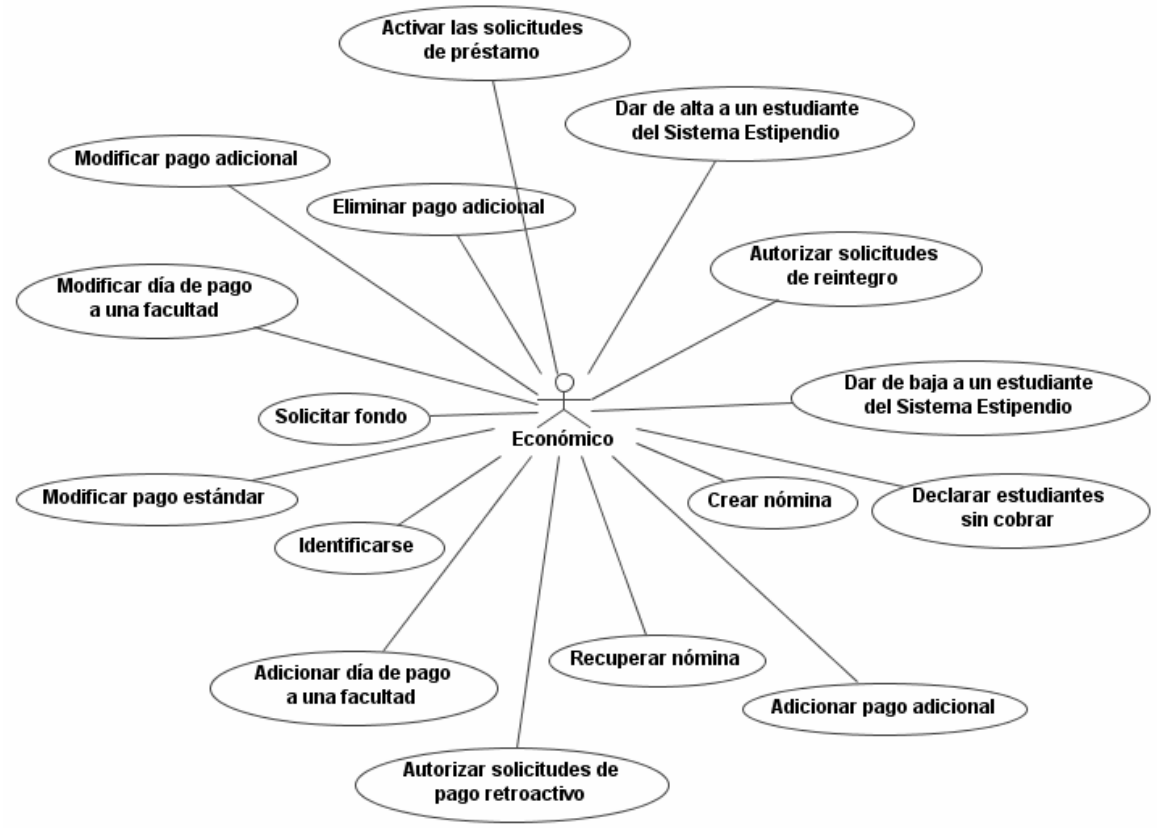


Figura. 2.1 Casos de uso del actor Económico.

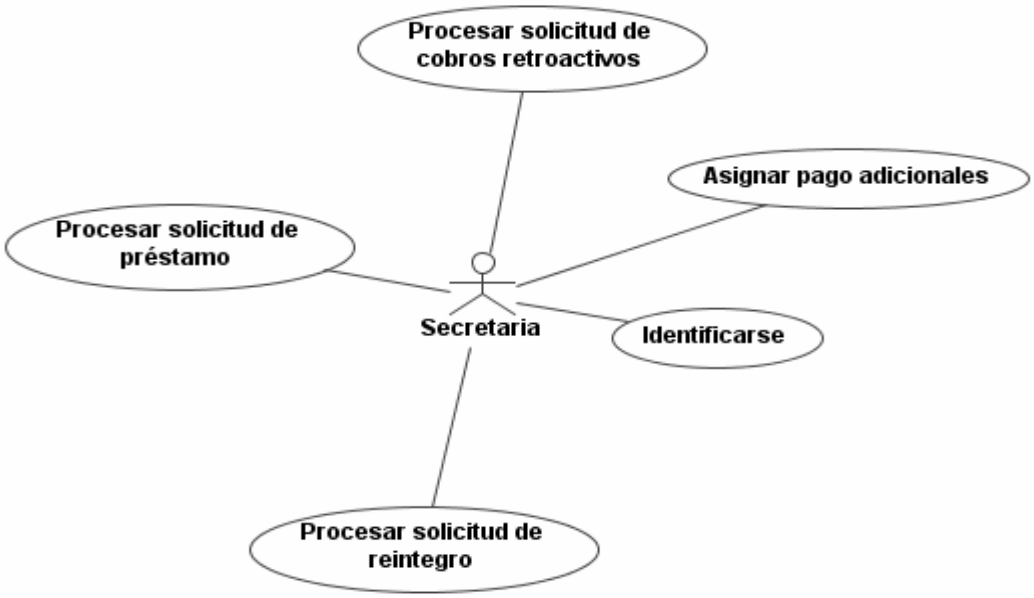


Figura. 2.2 Casos de uso del actor secretaria.

2.2.3 Descripción de los casos de uso

Actor Económico:

En el proceso de estipendio interviene activamente el económico encargado del mismo. Esta persona generalmente es única en el CES, en caso de no ser única, igualmente es un solo actor. Ella es la encargada de actualizar los estudiantes del sistema y los cambios manejables en las resoluciones que amparan el proceso del estipendio estudiantil. Las responsabilidades de este usuario, se describen a continuación.

Caso de uso	Modificar pago estándar
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema.
Propósito	Modificar los valores monetarios de los conceptos de pago estándar.
Resumen Esta operación se realiza bien en la implantación del sistema, o una vez que está funcional. Si existen cambios en las resoluciones que amparan el proceso de estipendio, mediante esta operación, se garantiza que el sistema se mantenga actualizado.	
Poscondiciones	Quedan modificados los montos pertenecientes a los conceptos de pago estándar.

Caso de uso	Adicionar pago adicional
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema.
Propósito	Añadir al sistema un concepto de pago adicional.
Resumen Mediante esta operación, el sistema es capaz de manejar los conceptos adicionales de pago a que se someten determinados estudiantes por tareas específicas que realizan.	
Poscondiciones	El sistema cuenta con un nuevo concepto de pago, listo para ser asignado a los estudiantes.

Caso de uso	Modificar pago adicional
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y el concepto de pago adicional debe existir.
Propósito	Modificar un concepto de pago adicional.
Resumen Mediante esta operación, el sistema es capaz de modificar el monto asociado a un determinado concepto de pago adicional.	
Poscondiciones	El sistema cuenta con dicho concepto de pago ya asignado a determinados estudiantes actualizado.

Caso de uso	Eliminar pago adicional
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y el concepto de pago adicional debe existir.
Propósito	Eliminar un concepto de pago adicional.
Resumen Mediante esta operación, el sistema es capaz de desechar un concepto de pago adicional que haya quedado obsoleto.	
Poscondiciones	El sistema elimina el concepto de pago adicional y elimina la relación del mismo con los estudiantes pertinentes

Caso de uso	Adicionar día de pago a una facultad
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y la facultad no puede tener día de pago asignado.
Propósito	Insertar el primer día de pago para la facultad.
Resumen Mediante esta operación, los estudiantes de la facultad antes mencionada, recibirán estipendio en el día de pago ahora establecido.	
Poscondiciones	El sistema es capaz de procesar dicha facultad para pagarles a sus estudiantes en el día de pago establecido anteriormente.

Caso de uso	Modificar día de pago a una facultad
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y debe existir el día de pago asignado a la facultad.
Propósito	Modificar las fechas relativas al día de pago y el estado de las operaciones del mismo, preparándolo para un nuevo pago.
Resumen La información que pertenece a los días de pago anteriores al actual, se almacenan en forma de nóminas, por lo tanto el día de pago activo, es el actual en proceso y para iniciar uno nuevo, las nóminas deben estar construidas y solamente se necesita modificar las fechas y reiniciar el estado de las operaciones.	
Poscondiciones	El sistema queda preparado para realizar un nuevo pago a la facultad involucrada.

Caso de uso	Crear nómina
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y la facultad debe tener día de pago asignado.
Propósito	Construir la nómina, del día de pago activo, para la facultad especificada.
Resumen Mediante esta operación, el sistema procesa a cada uno de los estudiantes que forman parte de la nómina especificada, buscando todos los conceptos de pago a que dicho estudiante está sometido y calculando el monto monetario que debe recibir el estudiante.	
Poscondiciones	La información se procesa y se almacena, quedando lista para conformar una nómina de estipendio estudiantil.

Caso de uso	Recuperar nómina
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y la nómina debe haber sido creada previamente.
Propósito	Obtener la nómina del curso, mes, facultad, especialidad y año de estudio especificado.
Resumen Mediante esta operación, se visualiza una nómina específica.	
Poscondiciones	La nómina visualizada puede ser consultada o impresa.

Caso de uso	Autorizar solicitudes de reintegro
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y la solicitud debe haber sido ingresada por la secretaria.
Propósito	Autorizar las solicitudes de los estudiantes para el reintegro del estipendio perdido en el anterior día de pago.
Resumen Mediante esta operación, se autoriza la solicitud de reintegro hecha por un estudiante.	
Poscondiciones	El estudiante queda sujeto a un reintegro del estipendio en el próximo día de pago

Caso de uso	Autorizar solicitudes de pago retroactivo
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y la solicitud debe haber sido ingresada por la secretaria.
Propósito	Autorizar las solicitudes de pago retroactivo a estudiantes con los cuales ha existido alguna anomalía en el pago de su estipendio.
Resumen Mediante esta operación, se autoriza la solicitud de pago retroactivo hecha por un estudiante.	
Poscondiciones	El estudiante queda sujeto a un cobro retroactivo en el próximo día de pago.

Caso de uso	Activar las solicitudes de préstamo
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y la solicitud debe haber sido ingresada por la secretaria.
Propósito	Activar una solicitud de préstamo al estudiante.
Resumen Cuando la solicitud fue aprobada por el rector de CES y el estudiante ya efectúa el contrato económico, la solicitud se activa finalmente.	
Poscondiciones	El estudiante queda sujeto a un préstamo reintegrable.

Caso de uso	**Declarar estudiantes sin cobrar**
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y debe haber culminado recientemente un día de pago.
Propósito	Notificar los estudiantes que han dejado de cobrar en el día de pago anterior.
Resumen Cuando un día de pago determinado culmina, es responsabilidad del Económico notificar los estudiantes que dejaron de cobrar.	
Poscondiciones	Los estudiantes declarados, quedan disponibles para realizar una solicitud de reintegro.

Caso de uso	Dar de alta a un estudiante en el Sistema Estipendio
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y mediante el módulo de matrícula, un estudiante ha sido incorporado al CES.
Propósito	Ingresar al estudiante en el Sistema de Estipendio.
Resumen Las altas de estudiantes, debido al módulo de matrícula, no son actualizadas en cascada en el sistema de estipendio. Es por eso que el económico tiene la responsabilidad de actualizar las tablas del sistema de estipendio.	
Poscondiciones	El estudiante añadido, recibirá estipendio en el próximo día de pago perteneciente a su facultad.

Caso de uso	Dar de baja a un estudiante en el Sistema Estipendio
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y mediante el módulo de matrícula, un estudiante ha sido dado de baja del CES.
Propósito	Dar de baja al estudiante en el Sistema de Estipendio.
Resumen Las bajas de estudiantes, debido al módulo de matrícula, no son actualizadas en cascada en el sistema de estipendio. Es por eso que el económico tiene la responsabilidad de actualizar las tablas del sistema de estipendio.	
Poscondiciones	El estudiante eliminado, no recibirá estipendio en el próximo día de pago perteneciente a su facultad.

Caso de uso	Identificarse
Actor	Económico
Precondiciones	
Propósito	Ingresar al sistema con los privilegios pertenecientes al rol ECONÓMICO.
Resumen	
Poscondiciones	Si la identificación es satisfactoria, el usuario ingresa al sistema.

Caso de uso	Solicitar fondo
Actor	Económico
Precondiciones	El Económico debe estar autenticado en el sistema y la nómina debe estar creada.
Propósito	Generar el reporte solicitud de fondos.
Resumen Mediante esta operación, se crea el reporte Solicitud de Fondos para una facultad y un día de pago específico.	
Poscondiciones	El reporte puede ser impreso o almacenado.

Actor Secretaria:

Cada facultad debe tener una persona encargada de gestionar todas las actividades del estipendio estudiantil relacionadas con los estudiantes que pertenecen a la misma. Normalmente es la propia Secretaria de la facultad.

Caso de uso	Procesar solicitud de cobros retroactivos
Actor	Secretaria
Precondiciones	El estudiante debe realizar la solicitud previamente.
Propósito	Incorporar al sistema una solicitud de este tipo, para que el Económico las apruebe.
Resumen Si ocurre alguna irregularidad en el pago a un estudiante, la secretaria ingresa la solicitud al sistema siempre y cuando las operaciones del día de pago actual estén abiertas aun.	
Poscondiciones	La solicitud queda en espera de aprobación por el Económico del sistema.

Caso de uso	Procesar solicitud de reintegro
Actor	Secretaria
Precondiciones	El estudiante debe realizar la solicitud previamente y debe estar declarado como un estudiante que dejó de cobrar.
Propósito	Incorporar al sistema una solicitud de este tipo, para que el Económico las apruebe.
Resumen Si el estudiante deja de cobrar en el día de pago anterior y es declarado como tal, la secretaria ingresa la solicitud al sistema siempre y cuando las operaciones del día de pago actual estén abiertas aun.	
Poscondiciones	La solicitud queda en espera de aprobación por el Económico del sistema.

Caso de uso	Procesar solicitud de préstamo
Actor	Secretaria
Precondiciones	El rector del CES tiene que aprobar la solicitud presentada por la comisión de subvención de la facultad.
Propósito	Ingresar una solicitud aprobada por el rector.
Resumen La comisión de subvención económica de la facultad, analiza la solicitud del estudiante y posteriormente se la presenta al rector del CES.	
Poscondiciones	La solicitud queda en espera de aprobación por el Económico del sistema, lo cual ocurrirá una vez que el estudiante efectúe el contrato económico con las dependencias pertinentes.

Caso de uso	Identificarse
Actor	Secretaria
Precondiciones	
Propósito	Ingresar al sistema con los privilegios pertenecientes al rol SECRETARIA_FACULTAD.
Resumen	
Poscondiciones	Si la identificación es satisfactoria, el usuario ingresa al sistema.

2.2.4 Espacio de nombres

El espacio de nombres, se integra con el que se había adoptado en los módulos anteriores, para garantizar la unicidad de los términos. El espacio de nombres utilizado para el modelo de clases persistentes, la aplicación cliente y la aplicación empresarial de estipendio, es **cu.mes.stipend** (Figura 2.3).

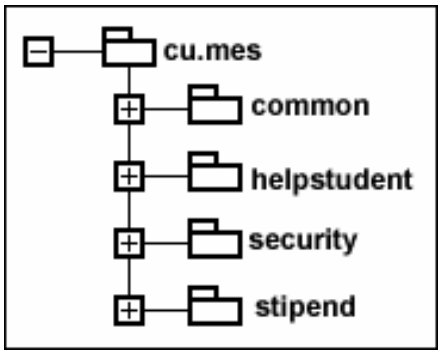
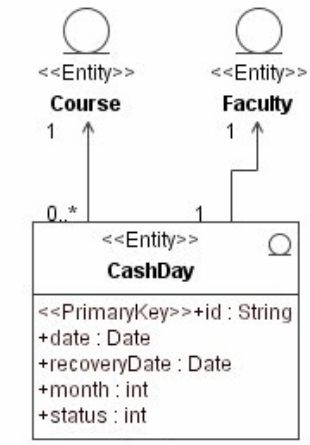


Figura 2.3 Espacio de nombres utilizados en el Sistema Estudiante.

2. 3 Implementación del Sistema

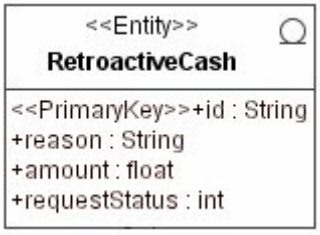
Atendiendo a las especificaciones y las normas y resoluciones del Ministerio de Educación Superior, se modelaron las entidades que conforman el núcleo persistente del sistema. Dichas entidades conforman un sistema capaz de ser auditable, pues mantiene un historial de las principales operaciones que se efectúan sobre el mismo.

La entidad **CashDay**, controla la información perteneciente a los días de pago. Pueden existir tantos objetos de esta entidad, como facultades tenga el CES; inicialmente el primer día de pago crea el objeto perteneciente a esa facultad y los sucesivos solo modifican las fechas y reinician el estado de las operaciones.



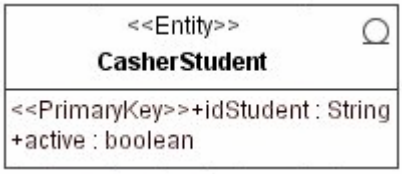
Propiedad	Descripción
Id	Identificador utilizado para manejar internamente los días de pago de cada facultad.
Date	Fecha programada para el día de pago.
RecoveryDate	Fecha programada para el día de recuperación.
Month	Necesario para poder efectuar un pago fuera del mes que le corresponde.
Status	Estado de sus operaciones.

La entidad **RetroactiveCash** es la encargada de archivar las solicitudes de cobros retroactivos. Si la solicitud fue aprobada el estudiante recibirá el dinero correspondiente.



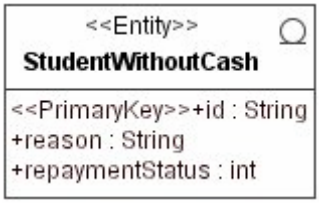
Propiedad	Descripción
Id	Identificador utilizado para manejar internamente las solicitudes de cobro retroactivo.
Reason	Motivo de la solicitud.
Amount	Monto a cobrar por dicha solicitud.
RequestStatus	Estado de la aprobación de dicha solicitud.

CasherStudent es la entidad encargada de almacenar los estudiantes matriculados y que tienen derecho a recibir estipendio, el identificador coincide con el de **Student**, para facilitar el trabajo ya que es conveniente manejar la relación **CasherStudent - Student** de forma manual y no utilizando una relación CMR. La consistencia entre las dos entidades se logra mediante el caso de uso Actualizar Base de Datos del Económico.



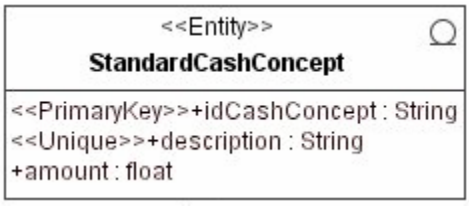
Propiedad	Descripción
idStudent	Identificador utilizado para manejar internamente a los estudiantes que están dentro del sistema.
Active	Propiedad que indica si el estudiante se encuentra activo o inactivo (por dejar de cobrar principalmente).

La entidad **StudentsWithOutCash** es encargada de almacenar los estudiantes que fueron declarados por el Económico por dejar de cobrar.



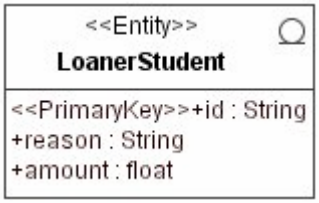
Propiedad	Descripción
Id	Identificador para manejar internamente a los estudiantes que dejaron de cobrar.
Reason	Justificación dada por el estudiante para tener derecho a cobrar nuevamente.
RepaymentStatus	Estado de la aprobación de dicha solicitud.

La entidad **StandardCashConcept** está encargada de almacenar los conceptos de pago estándar, entiéndase los referidos a estudiantes de curso regular, de orden18 y por ayudantía.



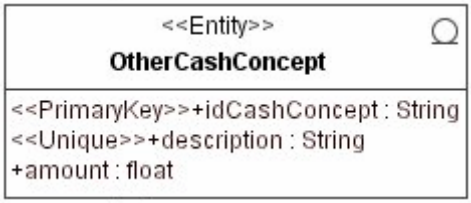
Propiedad	Descripción
idCashConcept	Identificador para manejar internamente los conceptos de pago.
Description	Descripción del concepto de pago.
Amont	Cantidad de dinero a cobrar por dicho concepto.

LoanerStudent es la entidad encargada de almacenar los préstamos activos.



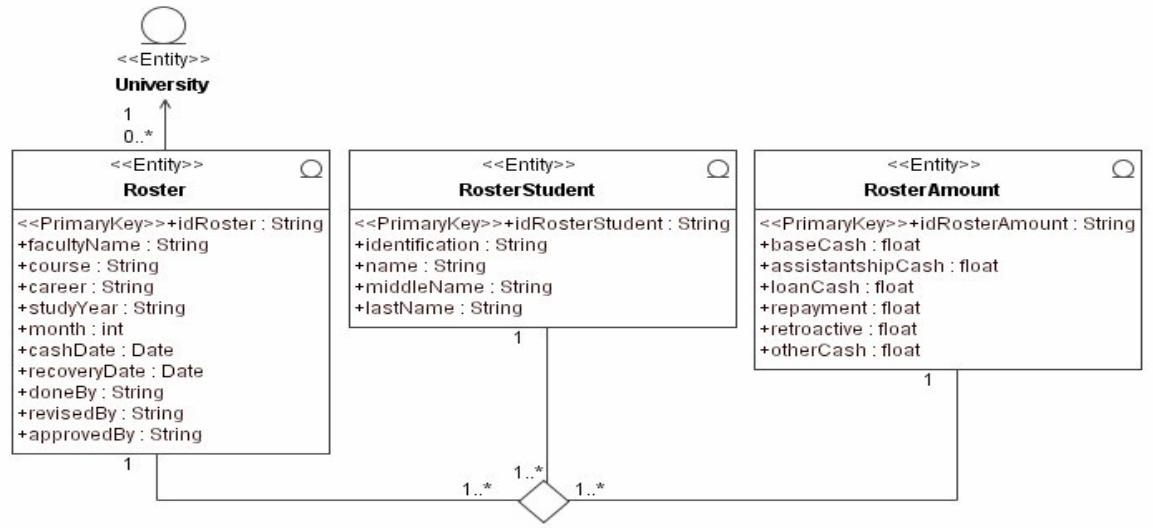
Propiedad	Descripción
Id	Identificador para manejar los estudiantes que tienen préstamos asignados.
Reason	Motivo del préstamo
Amount	Cantidad del préstamo

La entidad **OtherCashConcept** está encargada de almacenar los conceptos de pago adicionales.



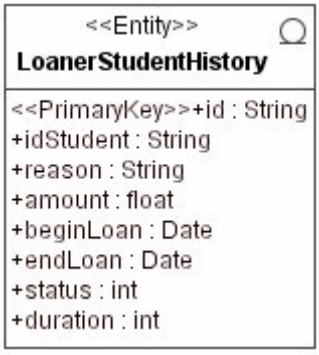
Propiedad	Descripción
idCashConcept	Identificador para manejar los conceptos de pago adicionales.
Description	Descripción del concepto de pago.
Amout	Cantidad de dinero a cobrar por dicho concepto.

Las siguientes entidades son las encargadas de almacenar la información necesaria para conformar las nóminas de pago.



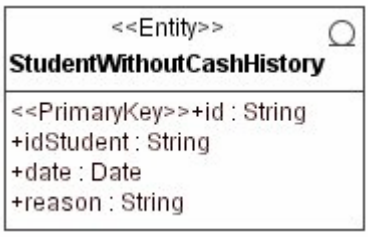
Propiedad	Descripción
idRoster	identificador de los atributos de la nómina
facultyName	facultad
course	curso
career	especialidad
studyYear	año de estudio
month	mes
cashDate	fecha en que ocurrió el pago
recoveryDate	fecha de recuperación del pago
doneBy	persona que emite la nómina
revisedBy	persona que revisó la nómina
approvedBy	persona que aprueba la nómina
idRosterAmount	identificador de los montos de las nóminas
baseCash	cantidad por salario básico
assistantshipCash	cantidad por ayudantía
loanCash	cantidad por préstamo
repayment	cantidad por reintegro
retroactive	cantidad por retroactivo
otherCash	cantidad por otros conceptos adicionales
idRosterStudent	identificador de los estudiantes
identification	número de identificación
name	nombre
middlename	primer apellido
lastname	segundo apellido

LoanerStudentHistory es la encargada de mantener el historial de préstamos a estudiantes, para que al finalizar la carrera, se establezca el reintegro de ese dinero.



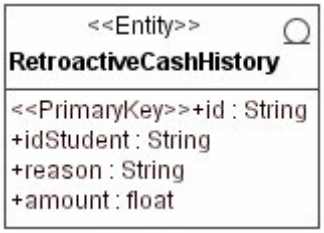
Propiedad	Descripción
Id	Identificador para manejar el historial de los préstamos.
idStudent	Identificador del estudiante que obtuvo el préstamo.
Reason	Motivo del préstamo.
Amount	Cantidad de dinero cobrado por mes de préstamo.
beginLoan	Fecha de inicio del préstamo.
endLoan	Fecha de culminación del préstamo.
Status	Indicador del estado del préstamo.
Duration	Cantidad de meses hábiles, del préstamo.

La entidad encargada de mantener el historial de estudiantes que han dejado de cobrar es **StudentWithoutCashHistory.**



Propiedad	Descripción
Id	Identificador para manejar las ocurrencias de estudiantes que han dejado de cobrar.
idStudent	Identificador del estudiante que dejó de cobrar.
Date	Fecha de la ocurrencia.
Reason	Motivo de la solicitud si existió para el reintegro del dinero.

La entidad encargada de mantener el historial de los cobros retroactivos es **RetroactiveCashHistory**.



Propiedad	Descripción
Id	Identificador para manejar las solicitudes de cobros retroactivos.
idStudent	Estudiante que realizó la solicitud.
Reason	Motivo para la solicitud.
Amount	Cantidad a cobrar por dicha solicitud.

La figura 2.4 ilustra las relaciones entre las principales entidades que forman el núcleo del sistema.

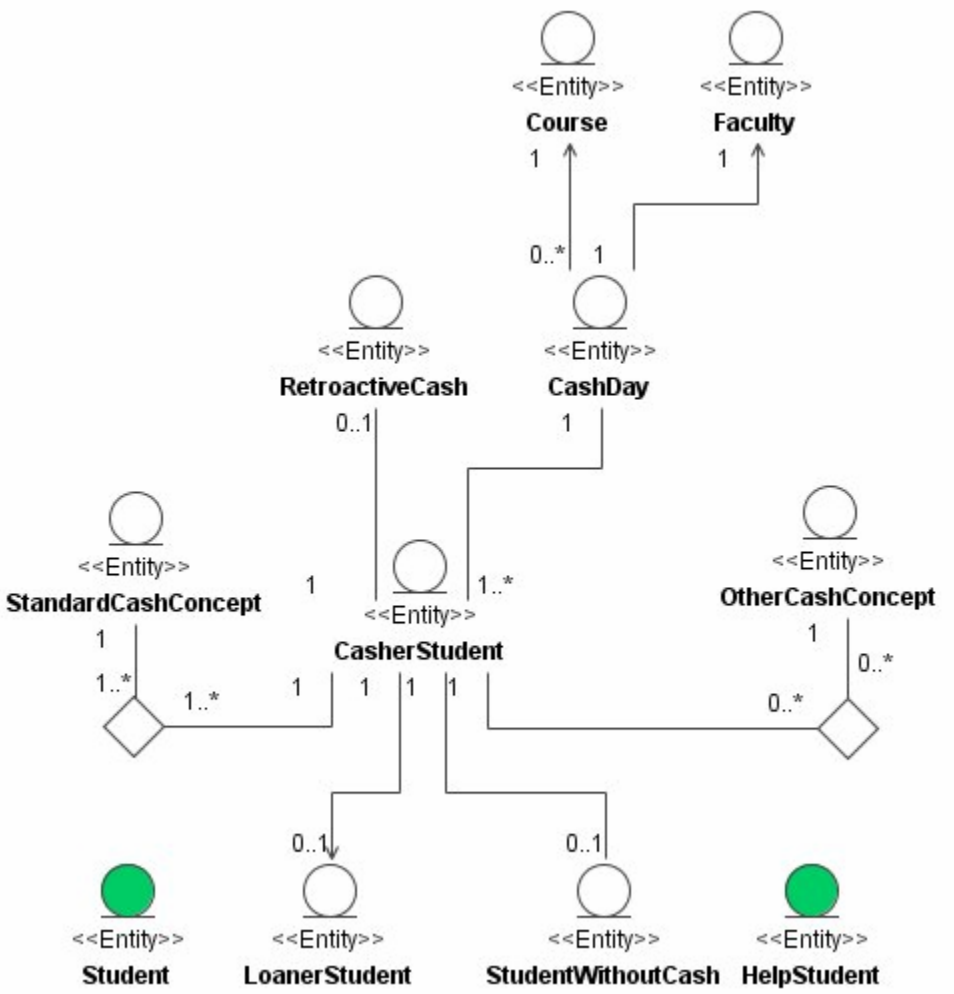
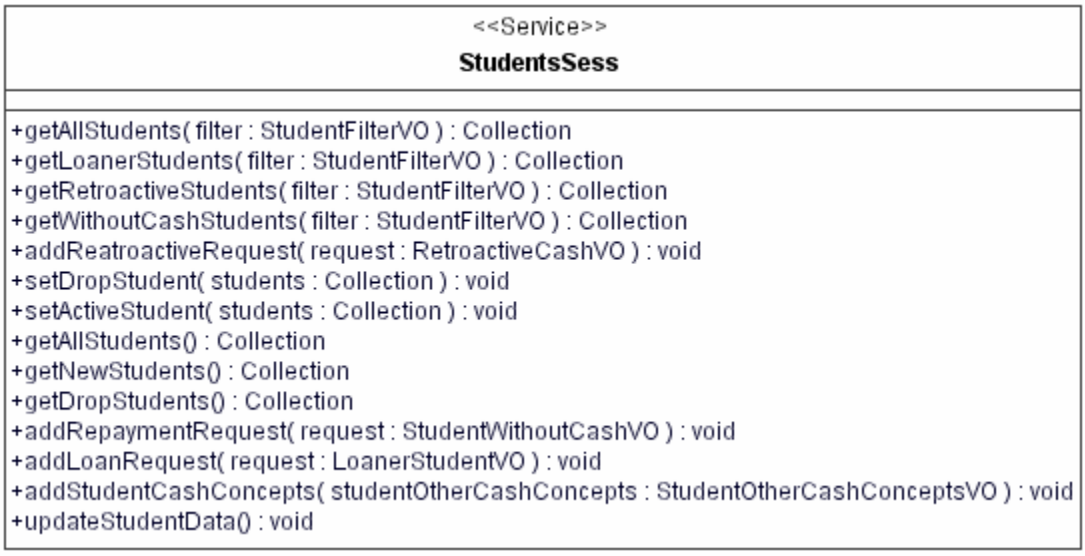


Figura 2.4 Núcleo Persistente del Sistema Estipendio.

Debido a las particularidades de nuestro sistema las clases persistentes se encuentran empaquetadas dentro de la aplicación empresarial **minotaur**. Las clases controladoras se encuentran en el cliente y tiene como función principal aislar a la interfaz de la aplicación. Las clases encargadas de la lógica del negocio se empaquetan en la aplicación empresarial **stipendSystem,** utilizándose el patrón Fachada de Sesión.

A continuación se muestran los *SessionBean* que intervienen en el sistema, cada uno de ellos agrupa funcionalidades lógicamente relacionadas.

Los métodos del negocio, relacionados de alguna manera con la gestión de los estudiantes se agruparon en el *SessionBean* **StudentsSess.**



Retornar los Estudiantes con préstamos.

Collection: **getLoanerStudents**(StudentFilterVO)

Parámetros:

Filtro para la búsqueda, en ella se especifica facultad, especialidad, modo de selección (facultad completa o un año específico), número de identificación personal, y subcadena del nombre o apellidos.

Devuelve:

Un *ArrayList* formado por *value objects* de tipo LoanerStudentVO y tantos como estudiantes con préstamo asignado, existan.

Retornar los Estudiantes con solicitudes de cobro retroactivo.

Collection: **getRetroactiveStudents**(StudentFilterVO)

Parámetros:

Filtro para la búsqueda, en ella se especifica facultad, especialidad, modo de selección (facultad completa o un año específico), número de identificación personal, y subcadena del nombre o apellidos.

Devuelve:

Un *ArrayList* formado por *value objects* de tipo RetroactiveCashVO y tantos como estudiantes con solicitudes de cobro retroactivo existan.

Dar baja a un grupo de estudiantes en el sistema.

void: setDropStudent(Collection)

Parámetros:

Estudiantes que deben salir del sistema.

Dar alta a un grupo de estudiantes en el sistema.

void: setActiveStudent(Collection)

Parámetros:

Estudiantes que deben entrar al sistema.

Obtener los estudiantes que han sido matriculados y no se encuentran insertados aun en el sistema.

Collection: getNewStudents()

Devuelve:

Los estudiantes que deben entrar al sistema.

Obtener los estudiantes que han causado baja, pero que aun se encuentran dentro del sistema.

Collection: getDropStudents()

Devuelve:

Los estudiantes que deben salir del sistema.

Adicionar una nueva solicitud de cobro retroactivo

void: addReatroactiveRequest(RetroactiveCashVO)

Parámetros:

Value Object que contiene la información de la solicitud.

Adicionar una nueva solicitud de préstamo.

void: addLoanRequest(LoanerStudentVO)

Parámetros:

Value Object que contiene la información de la solicitud.

En el proceso de estipendio, surgen, como resultado del mismo, las nóminas y reportes económicos; el *SessionBean* **RostersSess** posee esta responsabilidad.

Obtener una nómina determinada.

Collection: getRoster(RosterInfoVO)

Parámetros:

Value Object que contiene la información para buscar la nómina.

Devuelve:

Elementos que forman la nómina solicitada.

Crear la nómina correspondiente a una facultad y una carrera para el día de pago activo.

void: setRoster(RosterInfoVO)

Parámetros:

Value Object que contiene la información para crear la nómina.

Obtener los datos relativos a un día de pago.

void: getcashDay(CashDayVO)

Parámetros:

Value Object que contiene la información para buscar el día de pago.

Establecer nuevo día de pago.

void: setcashDay(CashDayVO)

Parámetros:

Value Object que contiene la información para crear el día de pago.

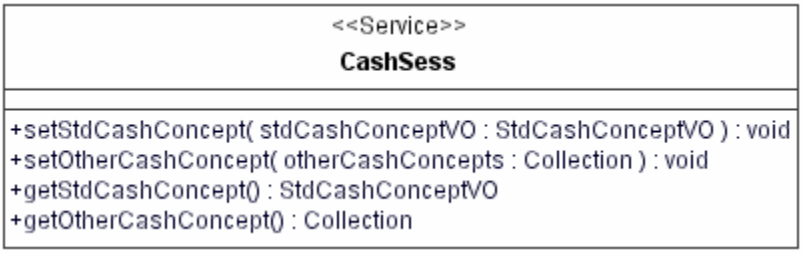
Obtener la solicitud de fondo para una facultad y un día de pago.

void: getFunds(RosterInfoVO)

Parámetros:

Value Object que contiene la información necesaria para realizar la solicitud de fondos.

Las funcionalidades del sistema relacionadas con los días de pago, se agruparon en el *SessionBean* **CashSess**.



Establecer o actualizar los conceptos de pago estándar.

void: setStdCashConcept(StdCashConceptVO)

Parámetros:

Datos necesarios para efectuar la actualización de los conceptos de pago estándar.

Obtener los conceptos de pago estándar.

StdCashConceptVO: getStdCashConcept()

Devuelve:

Información perteneciente a los conceptos de pago estándar.

Adicionar o modificar los conceptos de pago adicionales.

void : setOtherCashConcept(Collection)

Parámetros:

Los *value objects* con la información necesaria para insertar los conceptos de pago adicionales.

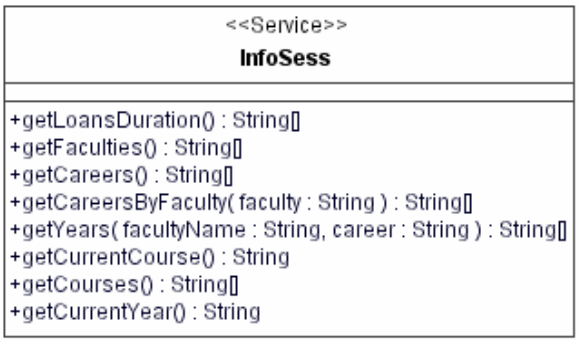
Obtener los conceptos de pago adicionales.

Collection: getOtherCashConcept()

Devuelve:

Todos los conceptos de pago adicionales que estén vigentes en el sistema.

El *SessionBean* **InfoSess** agrupa funcionalidades que no son propias del negocio. Es utilizado por la interfaz de usuario para mostrar cierta información.



Obtener las posibles duraciones de los préstamos.

String[]: getLoansDuration()

Devuelve:

Todos los posibles periodos de préstamos.

Obtener las facultades.

String[]: getFaculties()

Devuelve:

Todas las facultades.

Obtener las especialidades.

String[]: getCareersByFaculty(String)

Devuelve:

Todas las especialidades de una facultad.

Obtener los años.

Integer[]: getYears()

Devuelve:

Todos los años de estudio.

2.4 Conclusiones del Capítulo

En el presente capítulo se definieron los principales actores y casos de uso del sistema brindándose una breve descripción de los mismos. Se describieron todos los EJB™ utilizados, los atributos en las persistentes y los métodos de negocio en los de sesión. Se muestra además la solución propuesta.

La utilización de la metodología y plataforma seleccionada contribuyó a un gran ahorro de código y de tiempo para desarrollarlo, necesario para contrarrestar el largo período utilizado en el estudio y adaptación del nuevo perfil de trabajo.

En fin se ha construido un producto novedoso y robusto que satisface muchas de las operaciones necesarias para el control del estipendio estudiantil en la Educación Superior y de fácil manejo y acceso para los actores del sistema, cumpliendo así con los objetivos propuestos para dar solución al problema de investigación.

Capítulo III. Manual de usuario

El módulo de Estipendio Estudiantil, es un componente fundamental del Sistema Estudiante – GNU. Comprende el conjunto de funcionalidades relacionadas con el pago monetario que reciben los estudiantes de la enseñanza superior amparados por las resoluciones ministeriales. La asignación de los conceptos de pago a los estudiantes, la construcción de las nóminas, la gestión de las solicitudes, etc. son los principales elementos que se automatizan en el sistema.

3.1 Instalación del módulo

El Sistema Estudiante – GNU constará con un instalador para facilitar su implantación. El módulo de Estipendio Estudiantil consta de dos grupos principales de funcionalidades, las de las secretarias de facultades y las del económico encargado del proceso a nivel de CES.

El primer submódulo se une al cliente que agrupa a los módulos que se han desarrollado en la CUJAE y que utilizarán las secretarias en el proceso de la gestión de los estudiantes, el segundo, se distribuirá junto al mismo cargador de *plug-ins*, pero con ese sub-módulo únicamente y será utilizado por el económico encargado.

3.2 Control de acceso al Sistema

El mecanismo de acceso al sistema se garantiza mediante la utilización de usuarios y roles asociado a los mismos. Este proceso comprende a un módulo que se encuentra en etapa de desarrollo y no entra entre los objetivos de la investigación. Mediante esta práctica, el módulo de Estipendio, se subordina a las políticas de seguridad del sistema general.

3.3 Requisitos para la explotación

Para el correcto funcionamiento del módulo, es importante que el proceso de matrícula se efectúe previamente, debido a que el proceso de pago, se aplica a los estudiantes que se encuentran matriculados en el CES. Además existen algunos codificadores necesarios para la construcción de las nóminas, tales como nombre de la persona que la conforma, la revisa y la aprueba.

3.4 Opciones principales del Sistema

Módulo del Económico

3.4.1 Actualizar Sistema

Esta opción es utilizada cuando el Económico desea actualizar el sistema, sincronizándolo con el Sistema Estudiante-GNU.

Seleccionar en el menú **Cobros** la opción **Actualizar Sistema**. (Figura 3.1)

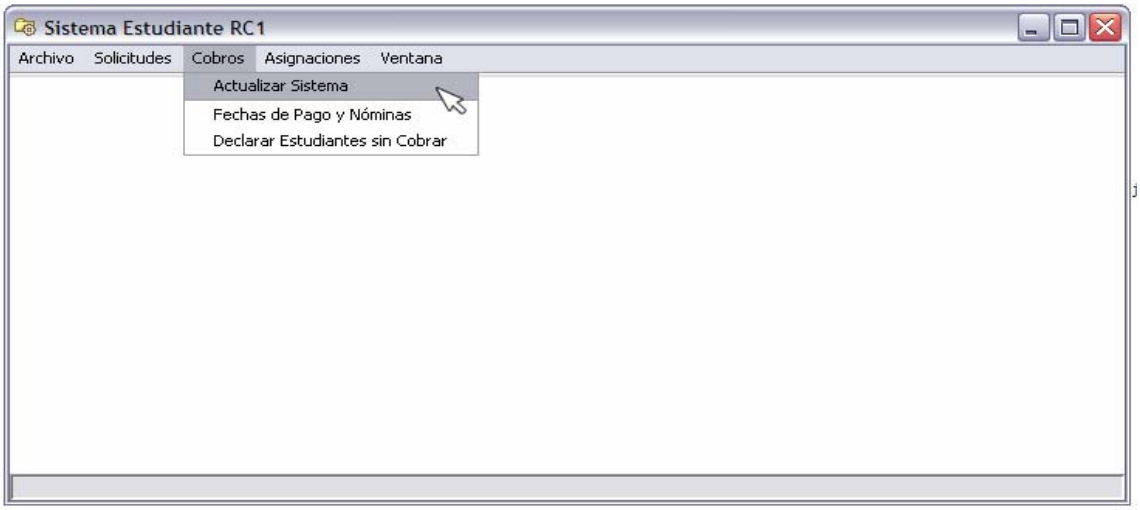


Figura 3.1 : Interfaz de la aplicación - opción del menú Cobros – Actualizar Sistema.

Se visualiza la ventana de actualización. (Figura 3.2)

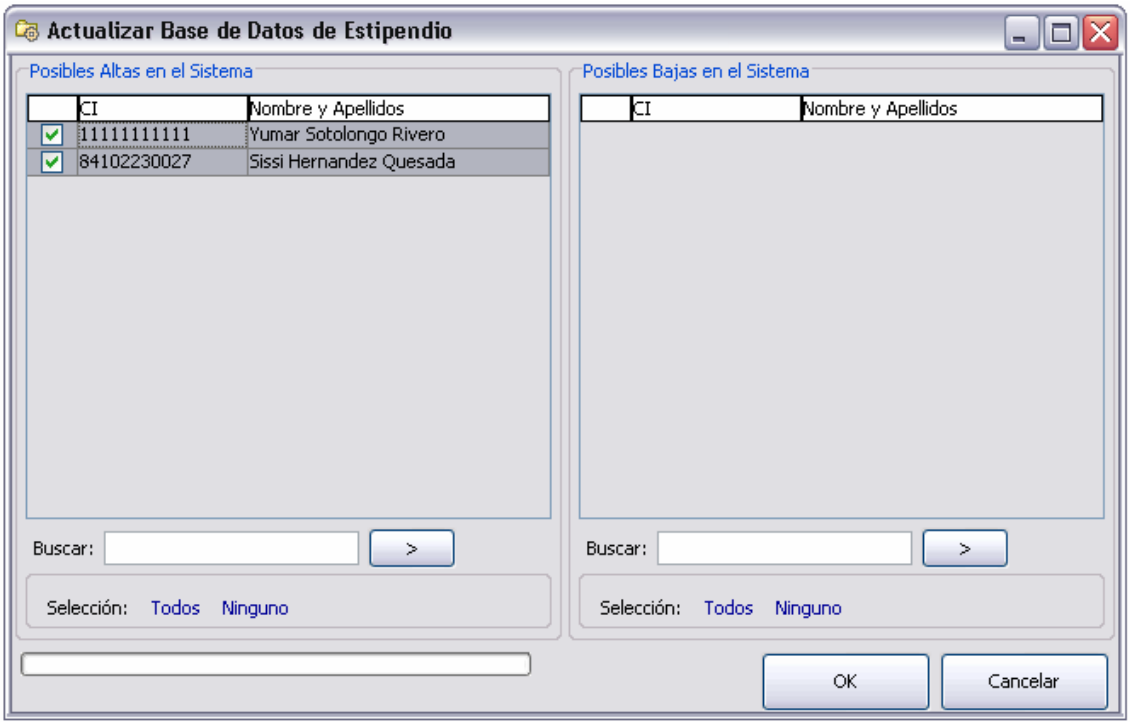


Figura 3.2 : Interfaz de Actualizar Base de Datos Estipendio.

En el panel de la izquierda aparecen lo estudiantes que deben ser dados de alta y en el de la derecha, los que deben ser bajas.

3.4.2 Establecer los montos por conceptos de pago

Esta opción es utilizada para establecer los montos a cobrar por todos los conceptos de pago activos en el sistema.

Seleccionar en el menú **Asignaciones** la opción **Establecer Cantidades a Pagar**. (Figura 3.3)

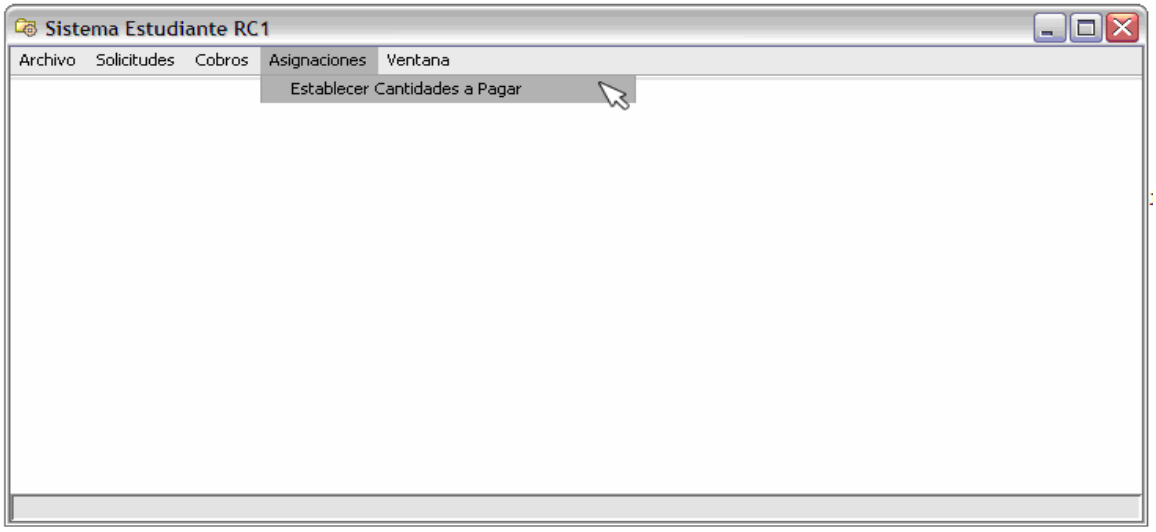


Figura 3.3 : Interfaz de la aplicación - opción del menú Asignaciones – Establecer Cantidades a Pagar.

Se visualiza la siguiente ventana que permite establecer los montos por todos los conceptos de pago. (Figura 3. 4)

Figura 3.4 : Interfaz de Cantidades a Pagar a los Estudiantes.

Aparecen los conceptos de pago estándar con sus montos y en el panel inferior derecho los conceptos adicionales.

3.4.3 Fecha de Pago y Nóminas

Esta opción es utilizada cuando el Económico desea establecer la fecha para un día de pago, construir las nóminas de un día de pago, visualizar la misma y visualizar la solicitud de fondo para un día de pago.

Seleccionar en el menú **Cobros** la opción **Fechas de Pago y Nóminas**. (Figura 3.5)

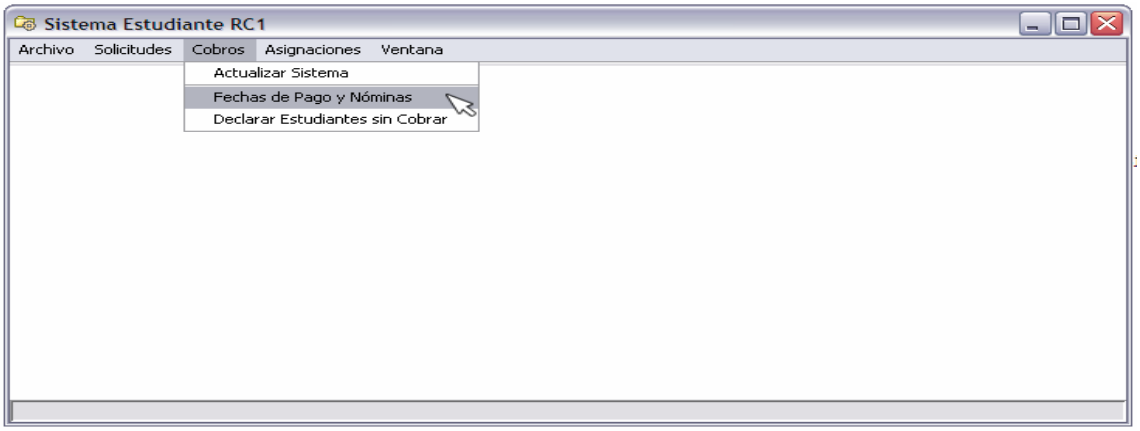


Figura 3.5 : Interfaz de la aplicación - opción del menú Cobros – Fechas de Pago y Nóminas.

Se visualiza la ventana de Cobros y Nóminas. (Figura 3.6)

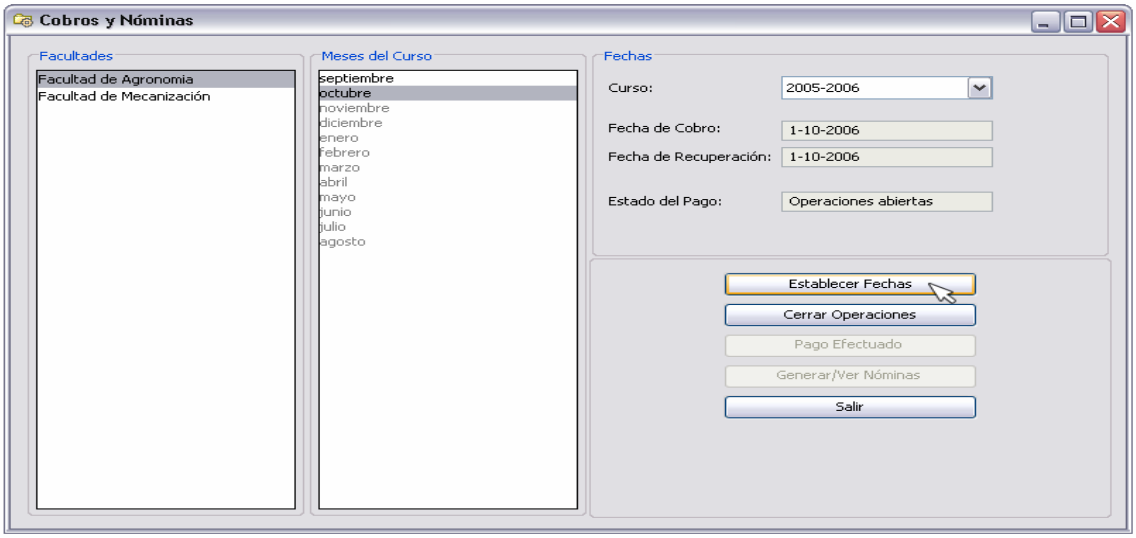


Figura 3.6 : Interfaz de Cobros y Nóminas.

En el panel de la izquierda aparecen las facultades que posee el CES, en el de la derecha los meses del curso actual, en extremo superior derecho de la ventana, la información de los días de pago ya establecidos y en el inferior derecho las operaciones válidas a efectuar con dicho día de pago.

3.4.4 Solicitudes de reintegro

Esta opción es utilizada cuando el Económico desea consultar las solicitudes de reintegro que han sido pedidas por los estudiantes e ingresadas por la secretaria.

Seleccionar en el menú **Solicitudes** la opción **Reintegros**. (Figura 3.7)

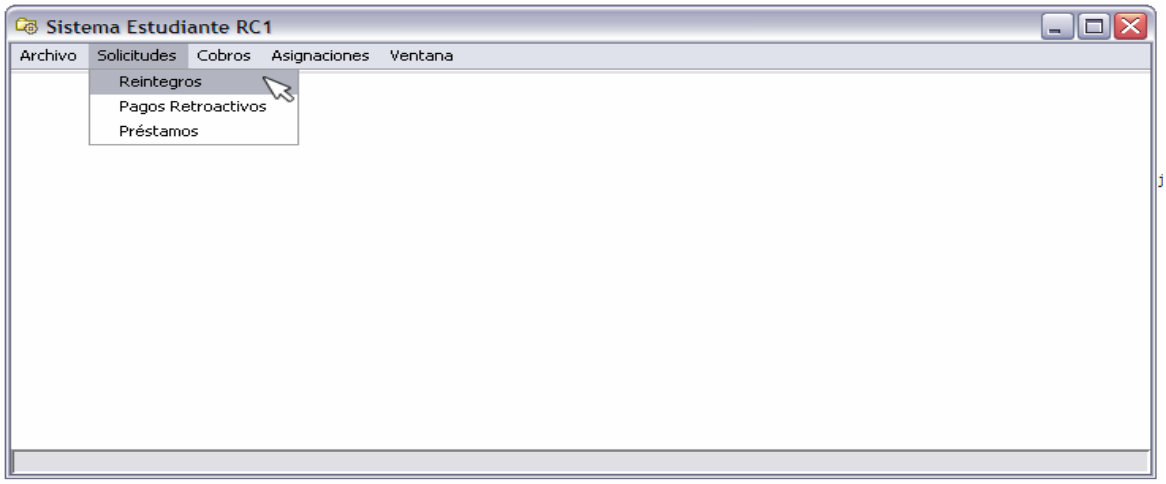


Figura 3.7 : Interfaz de la aplicación - opción del menú Solicitudes – Reintegros.

Se visualiza la siguiente ventana que permite consultar, aprobar o rechazar las solicitudes de reintegro ingresadas al sistema. (Figura 3.8)

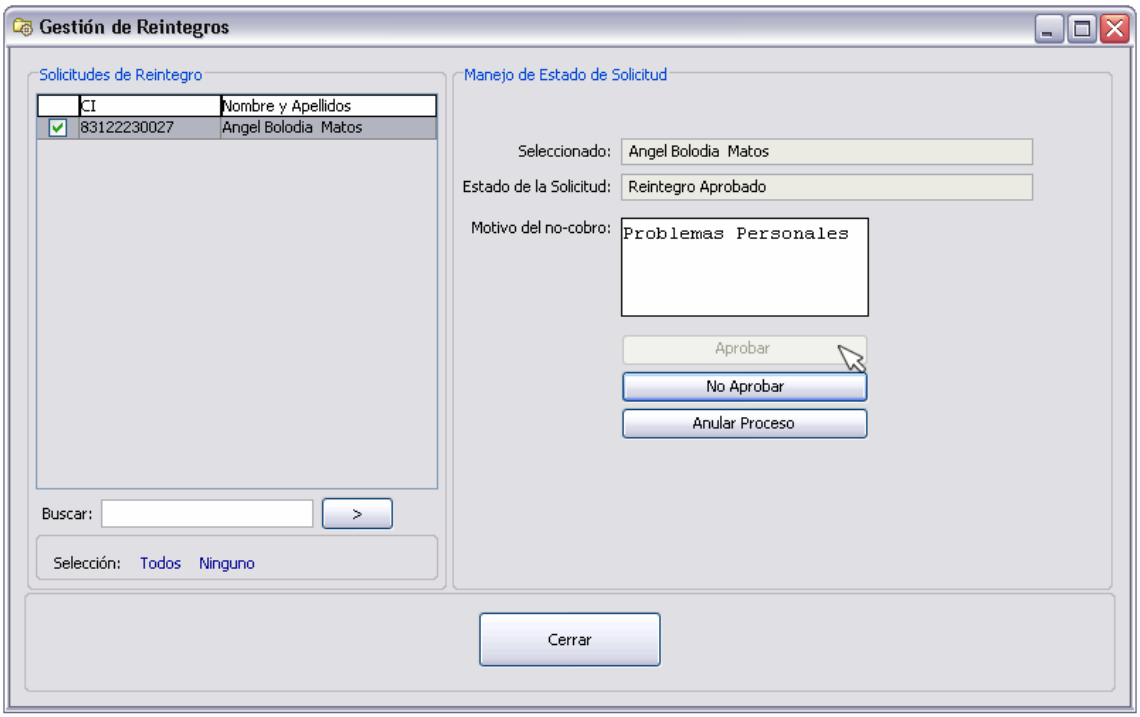


Figura 3.8 : Interfaz de Gestión de Reintegros.

En el panel de la izquierda aparecen los estudiantes que han hecho solicitudes, y en el de la derecha la información referente a la solicitud. El Económico puede aprobar, no aprobar o rectificar su decisión.

3.4.5 Solicitudes de pagos retroactivos

Esta opción es utilizada cuando el Económico desea consultar las solicitudes de pagos retroactivos que han sido pedidas por los estudiantes e ingresadas por la secretaria.

Seleccionar en el menú **Solicitudes** la opción **Pagos Retroactivos**. (Figura 3.9)

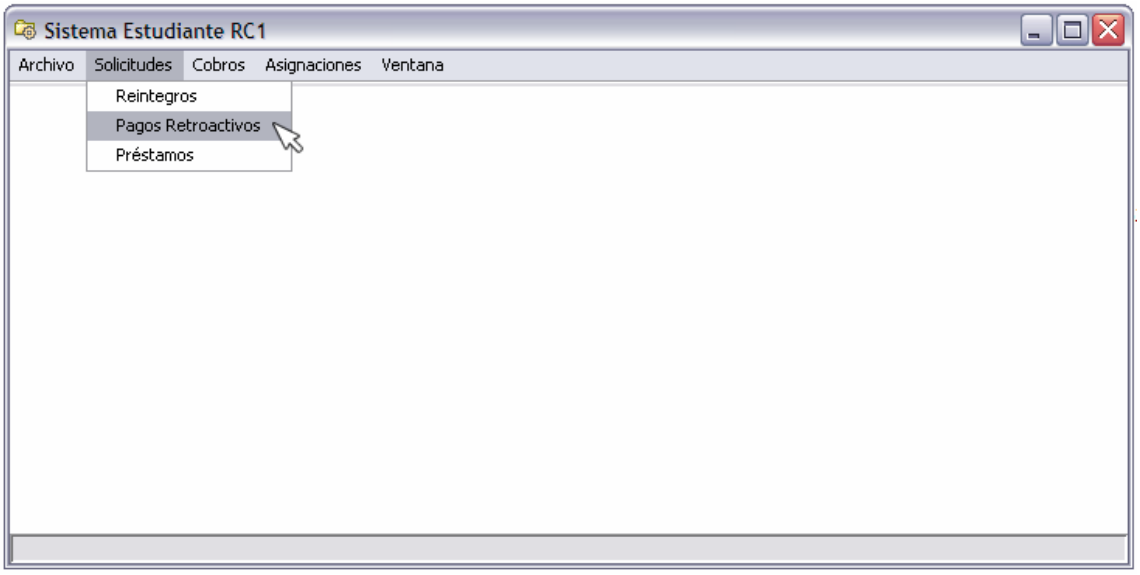


Figura 3.9 : Interfaz de la aplicación - opción del menú Solicitudes – Pagos Retroactivos.

Se visualiza la siguiente ventana que permite consultar, aprobar o rechazar las solicitudes de pago retroactivo ingresadas al sistema. (Figura 3.10)

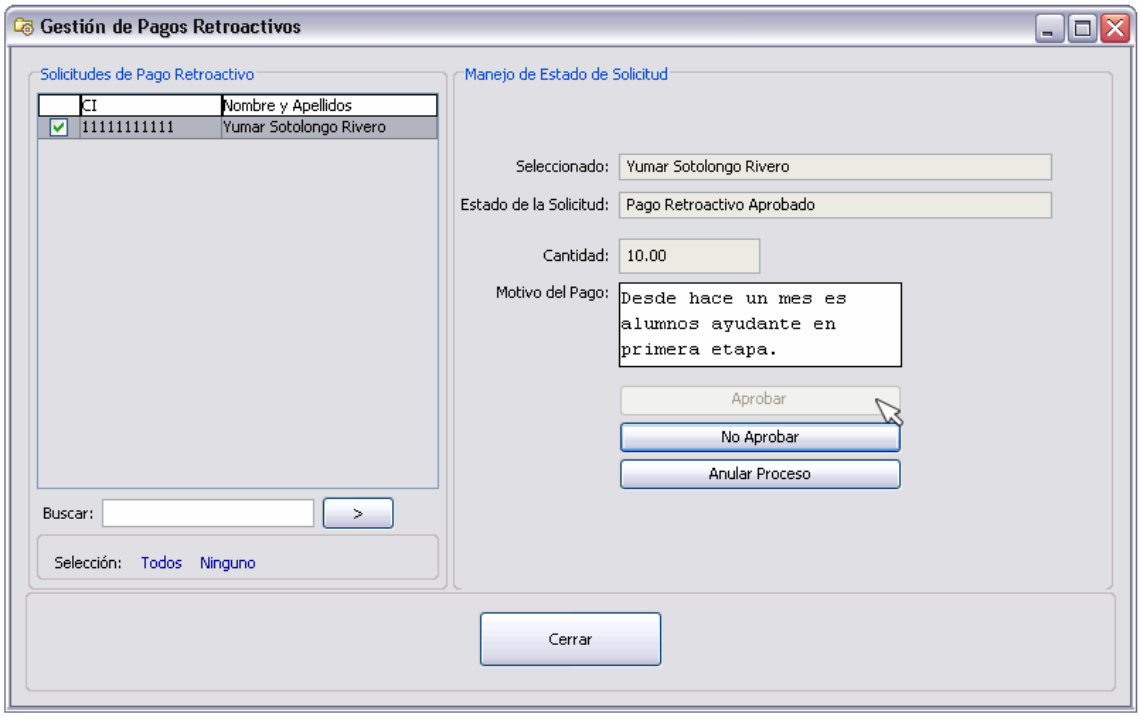


Figura 3.10 : Interfaz de Gestión de Pagos Retroactivos

En el panel de la izquierda aparecen los estudiantes que han hecho solicitudes, y en el de la derecha la información referente a la solicitud. El Económico puede aprobar, no aprobar o rectificar su decisión.

3.4.6 Solicitudes de Préstamos

Esta opción es utilizada cuando el Económico desea consultar las solicitudes de préstamos que han sido pedidas por los estudiantes, aprobadas por el rector e ingresadas por la secretaria.

Seleccionar en el menú **Solicitudes** la opción **Préstamos**. (Figura 3.11)

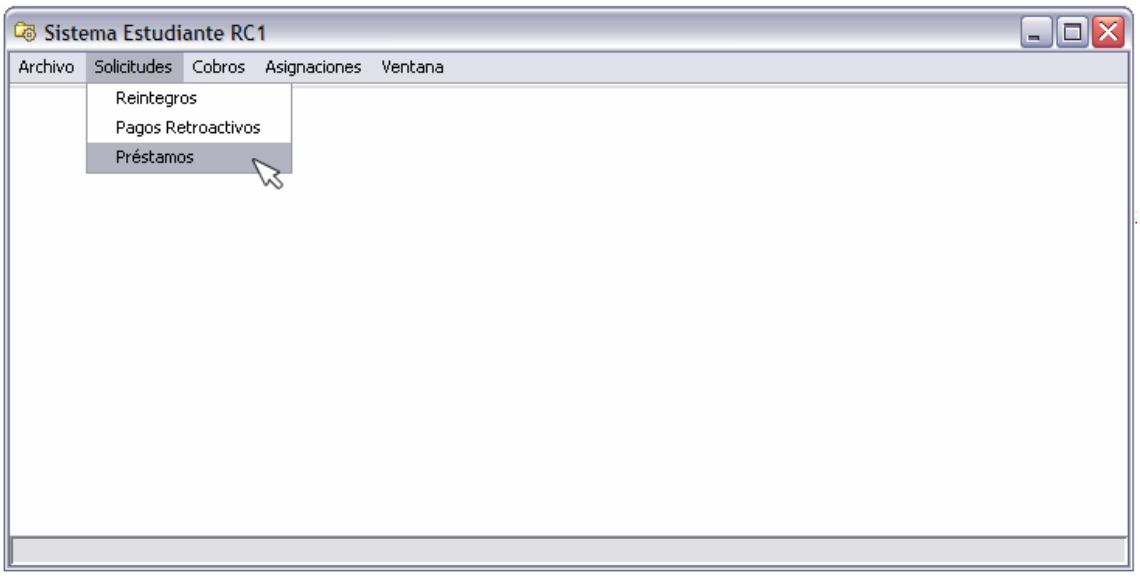


Figura 3.11 : Interfaz de la aplicación - opción del menú Solicitudes – Préstamos.

Se visualiza la siguiente ventana que permite consultar, aprobar o rechazar las solicitudes de préstamos ingresadas al sistema. (Figura 3.12)

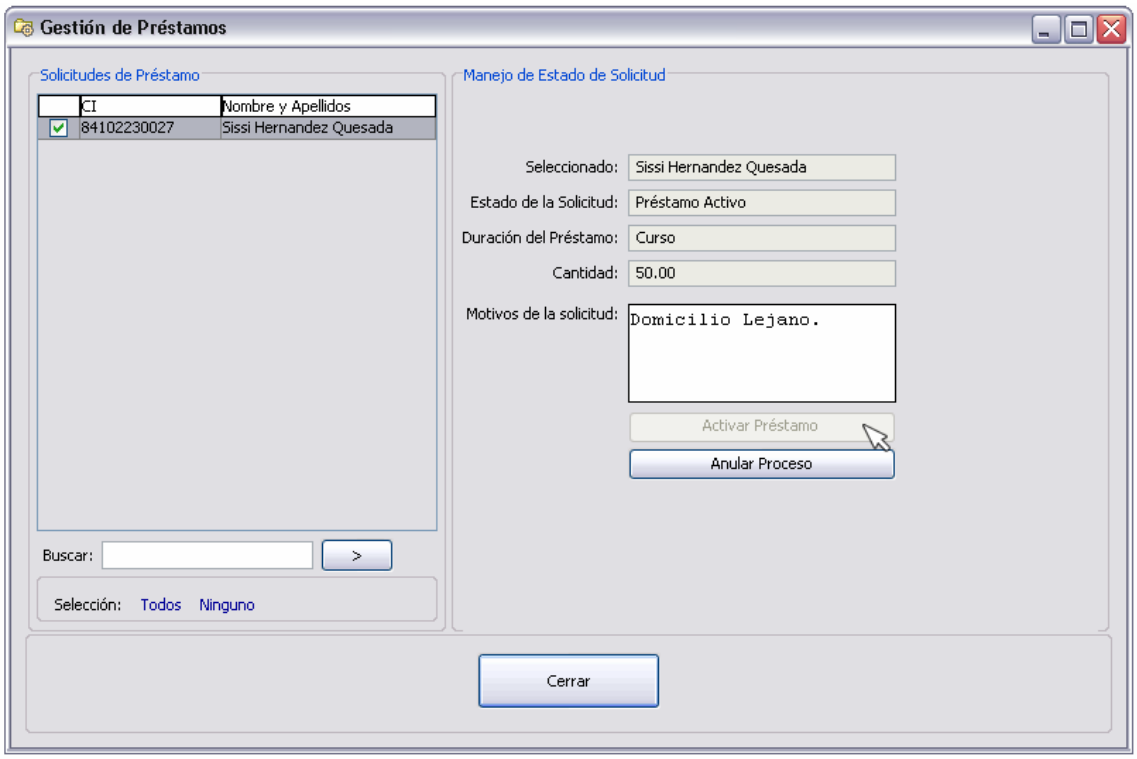


Figura 3.12 : Interfaz de Gestión de Préstamos

En el panel de la izquierda aparecen los estudiantes que han hecho solicitudes, y en el de la derecha la información referente a la solicitud. El Económico puede aprobar, no aprobar o rectificar su decisión.

3.4.7 Gestión de las Nóminas

Esta opción es utilizada cuando el Económico desea construir o visualizar las nóminas y las solicitudes de fondo.

En la ventana de Cobros y Nóminas se oprime el botón **Generar/Ver Nóminas.**

Se visualiza la siguiente ventana para la gestión de las nóminas y solicitudes de fondo. (Figura 3.13)

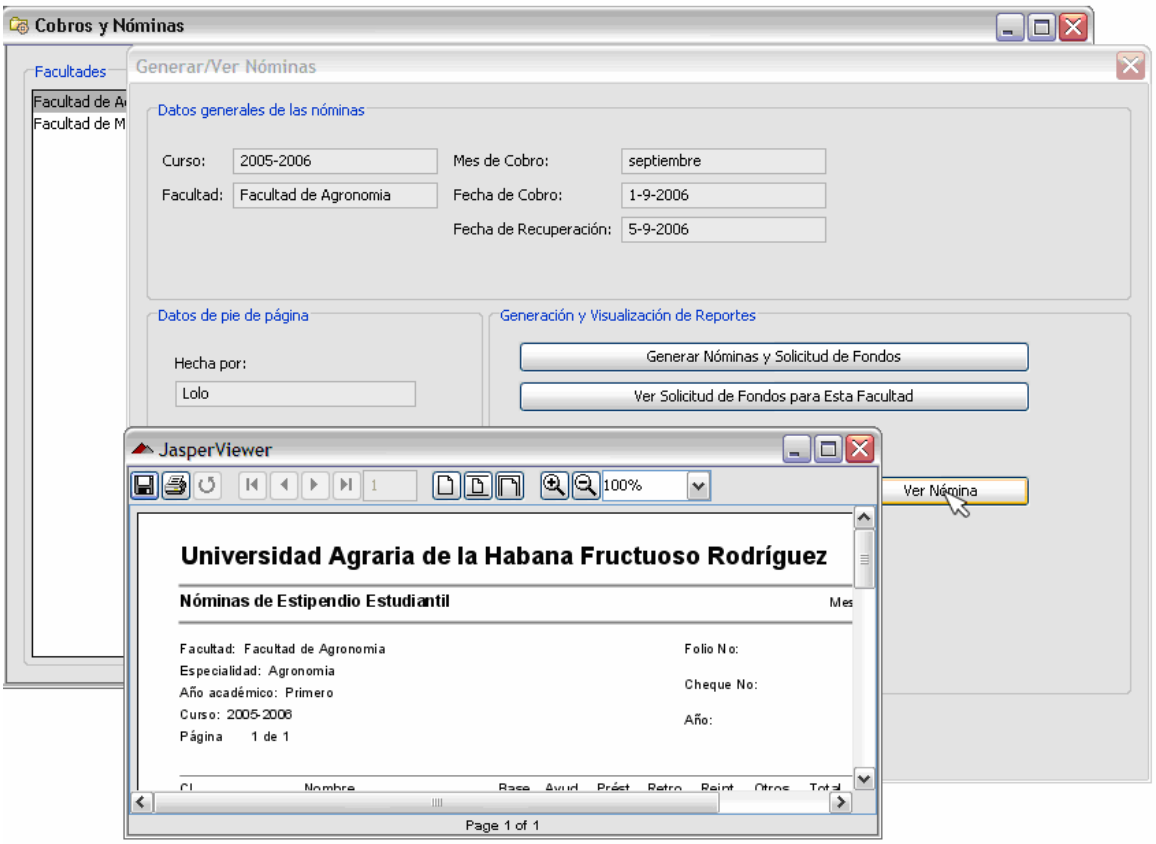


Figura 3.13 : Interfaz de Gestión de Nóminas.

3.4.8 Declaración de estudiantes que no cobraron

Esta opción es utilizada para notificar los estudiantes que dejaron de cobrar cuando se efectúa el último día de pago del estudiante.

Seleccionar en el menú **Cobros** la opción **Declarar Estudiantes sin Cobrar**. (Figura 3.14)

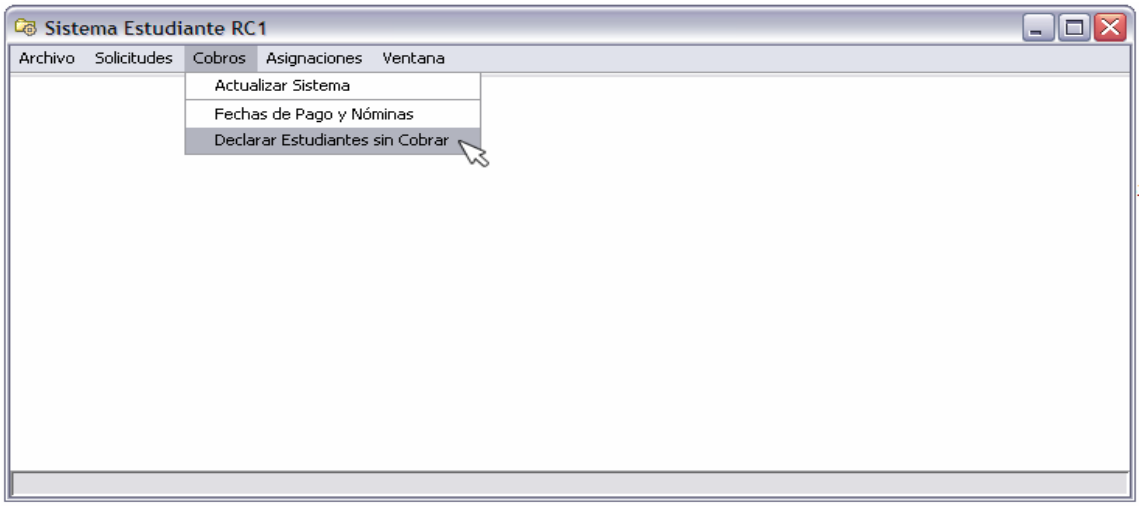


Figura 3.14 : Interfaz de la aplicación - opción del menú Cobros – Declarar Estudiantes sin Cobrar.

Se visualiza la siguiente ventana que permite notificar los estudiantes que dejaron de cobrar. (Figura 3.15)

Figura 3.15 : Interfaz de Declaración de Estudiantes que no Cobraron.

En el extremo superior, aparecen los filtros de búsqueda. En el panel de la izquierda aparecen los estudiantes buscados, y en el de la derecha los que ya están notificados.

Módulo de la Secretaria

3.4.9 Asignación de conceptos de pago adicionales

Esta opción es utilizada cuando se desean relacionar a estudiantes con determinados conceptos de pago adicionales.

Seleccionar en el menú **Estipendio** la opción **Asignar Conceptos de Pago a Estudiantes**. (Figura 3.16)

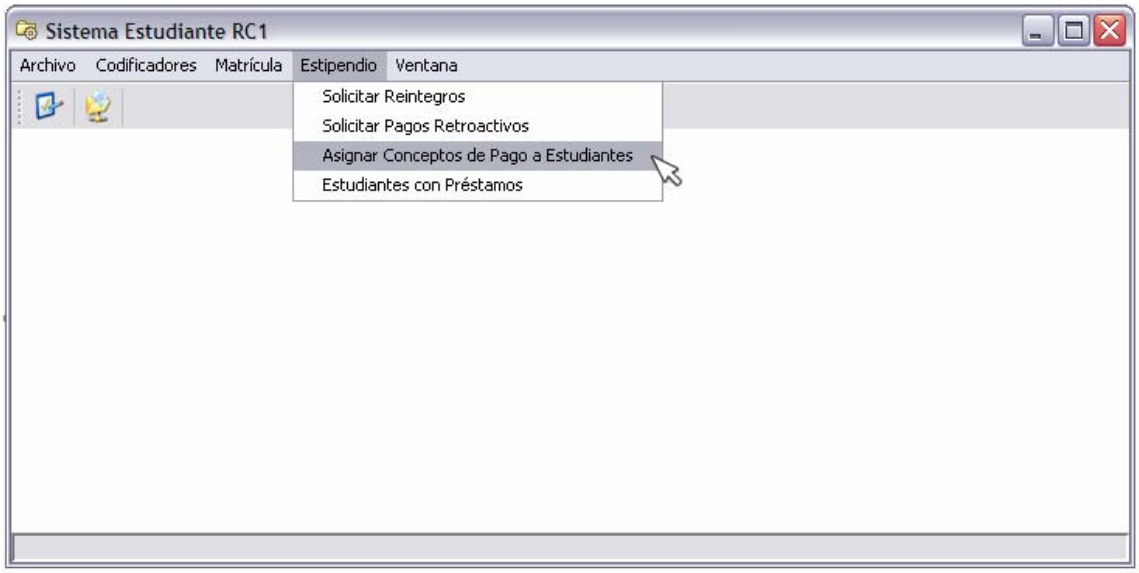


Figura 3.16 : Interfaz de la aplicación - opción del menú Asignaciones – Asignar Conceptos de Pago a Estudiantes.

Se visualiza la siguiente ventana que permite asignar a los estudiantes los conceptos de pago. (Figura 3.17)

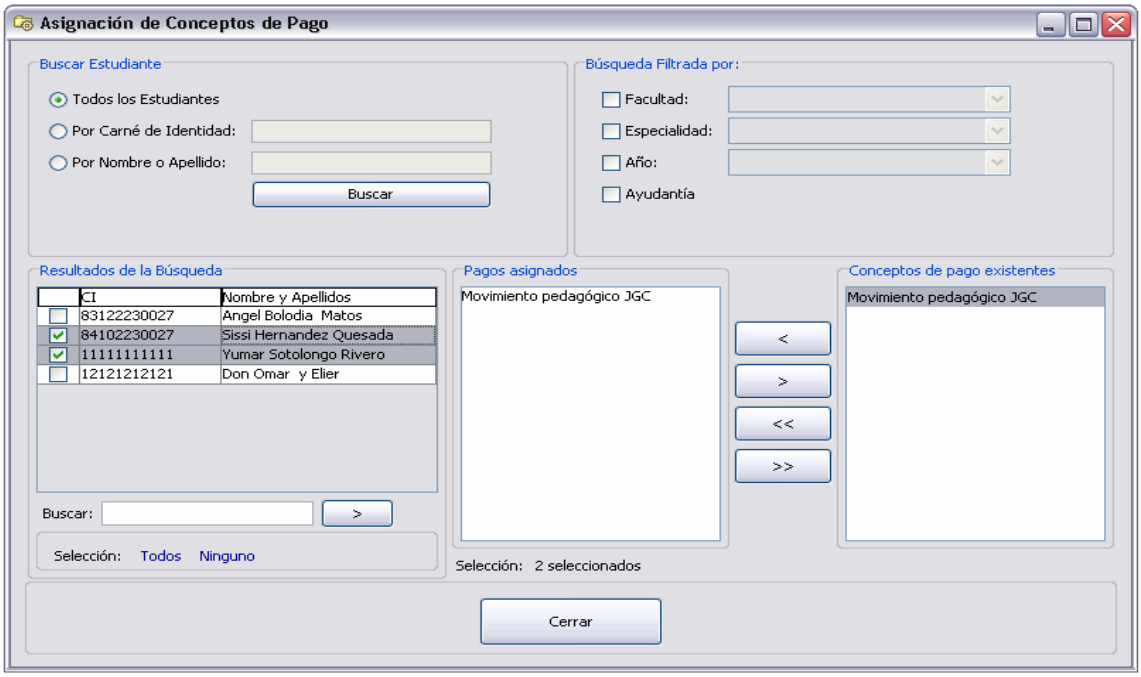


Figura 3.17 : Interfaz de Asignación de Conceptos de Pago.

En el extremo superior aparecen los filtros de búsqueda. En el panel de la izquierda, los estudiantes buscados, en el del centro los pagos asignados a ellos y en el de la derecha, todos los conceptos activos en el sistema.

3.4.10 Solicitar reintegros

Esta opción es utilizada cuando la secretaria desea adicionar las solicitudes de reintegro que han sido pedidas por los estudiantes.

Seleccionar en el menú **Estipendio** la opción **Solicitar Reintegros**. (Figura 3.18)

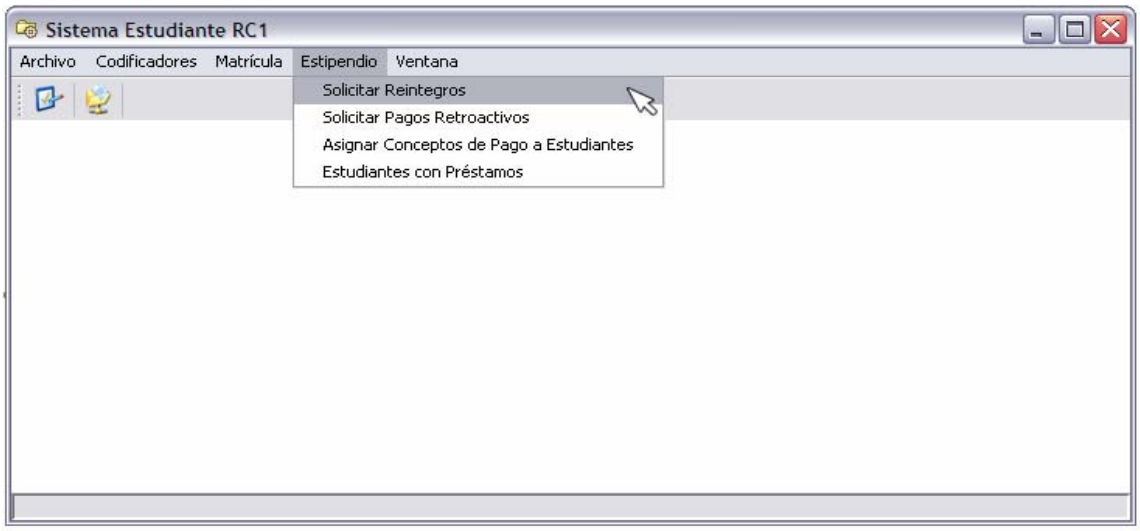


Figura 3.18: Interfaz de la aplicación - opción del menú Estipendio – Solicitar Reintegros.

Se visualiza la siguiente ventana que permite añadir las solicitudes de reintegros efectuadas por los estudiantes. (Figura 3.19)

Figura 3.19 : Interfaz de Reintegros.

En el panel de la izquierda aparecen los estudiantes que han sido declarados como sin cobrar, y en el de la derecha la información referente a la solicitud.

3.4.11 Solicitud de pagos retroactivos

Esta opción es utilizada cuando la secretaria desea adicionar las solicitudes de pago retroactivos que han sido pedidas por los estudiantes.

Seleccionar en el menú **Estipendio** la opción **Solicitar Pagos Retroactivos** . (Figura 3.20)

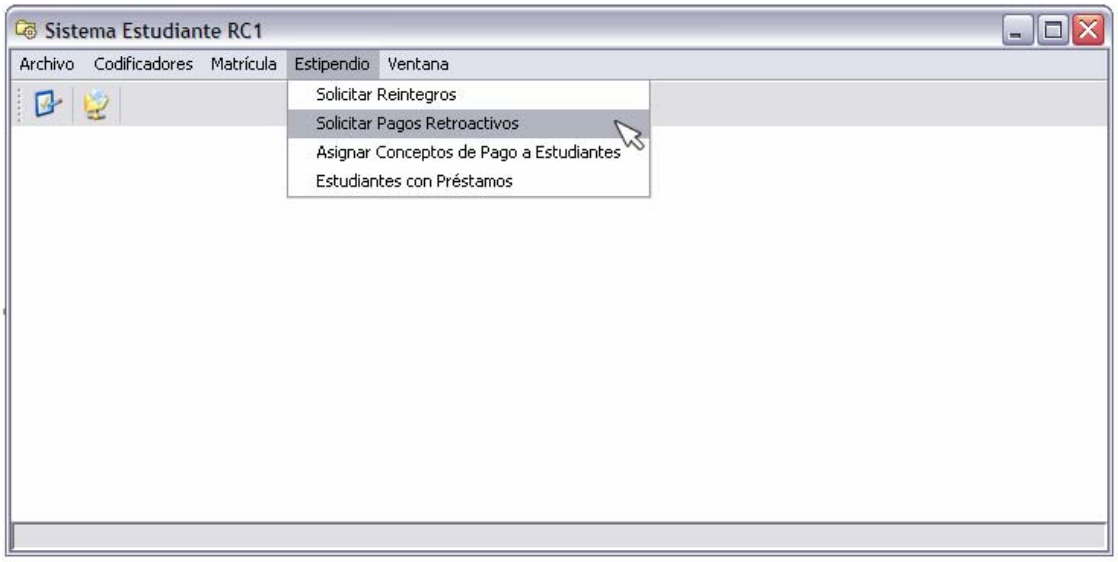


Figura 3.20: Interfaz de la aplicación - opción del menú Estipendio – Solicitar Pagos Retroactivos.

Se visualiza la siguiente ventana que permite añadir las solicitudes de pagos retroactivos efectuadas por los estudiantes. (Figura 3.21)

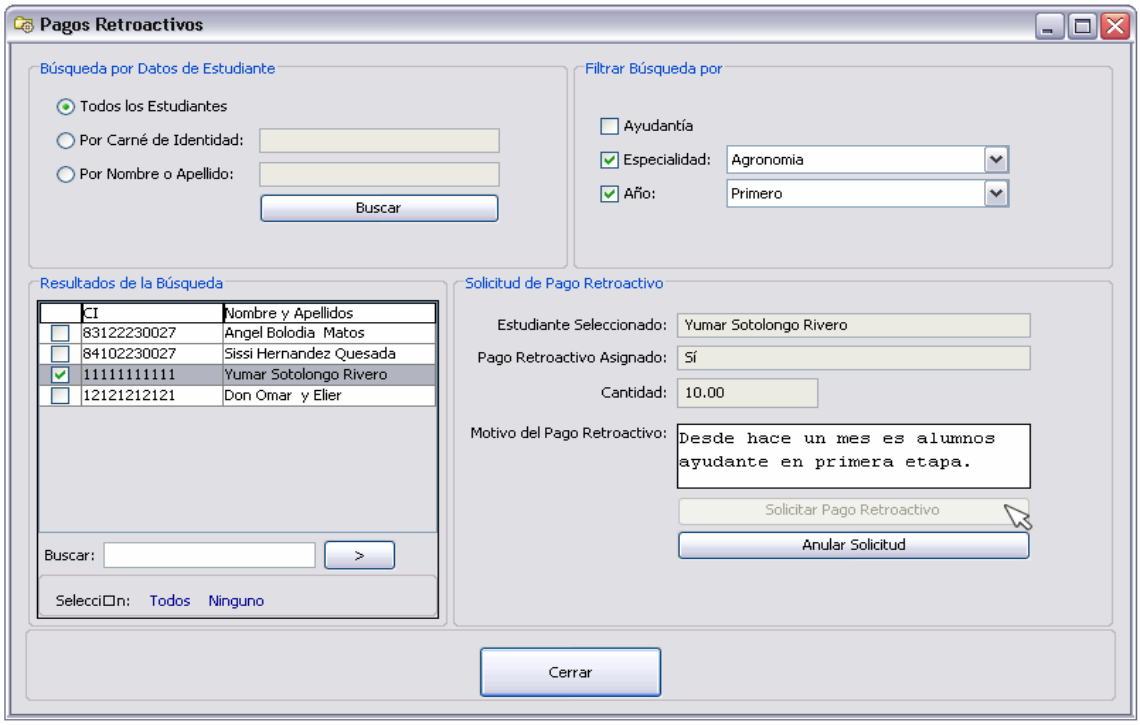


Figura 3.21 : Interfaz de Pagos Retroactivos.

En el extremo superior aparecen los filtros de búsqueda. En el panel de la izquierda, los estudiantes buscados y en el de la derecha, la información referente a la solicitud.

3.4.12 Solicitud de préstamos

Esta opción es utilizada cuando la secretaria desea adicionar las solicitudes de préstamos que han sido pedidas por los estudiantes y aprobadas por el rector del CES.

Seleccionar en el menú **Estipendio** la opción **Estudiantes con Préstamos**. (Figura 3.22)

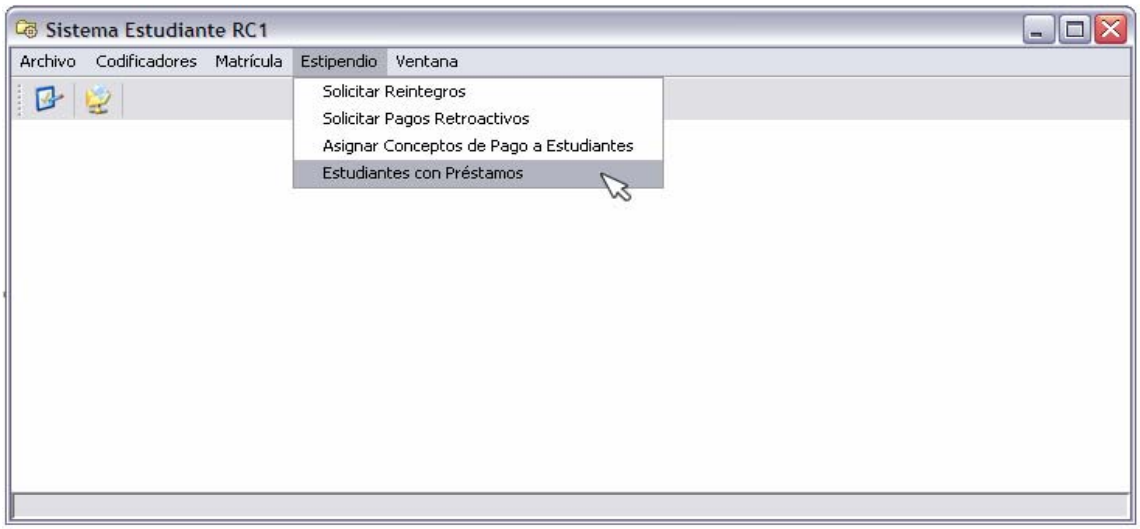


Figura 3.22: Interfaz de la aplicación - opción del menú Estipendio – Estudiantes con Préstamos.

Se visualiza la siguiente ventana que permite añadir las solicitudes de préstamos efectuadas por los estudiantes. (Figura 3.23)

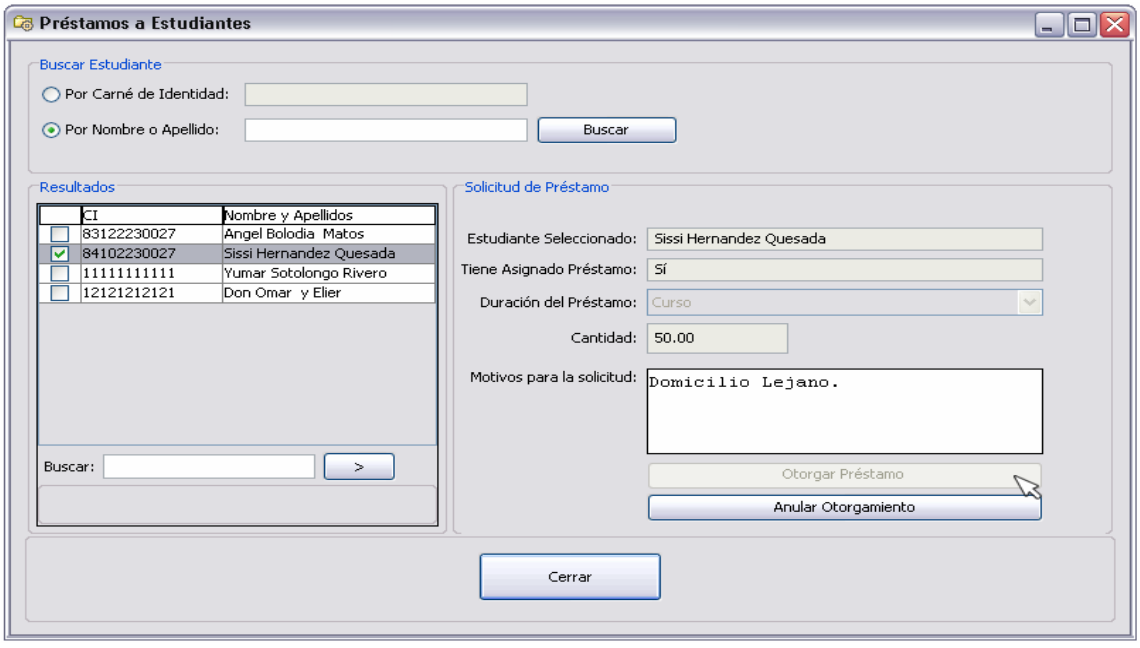


Figura 3.23 : Interfaz de Préstamos a Estudiantes.

En el extremo superior aparecen los filtros de búsqueda. En el panel de la izquierda, los estudiantes buscados y en el de la derecha, la información referente a la solicitud.

3.5 Análisis de los resultados

Sin lugar a dudas este sistema contribuye de manera significativa al aumento de la calidad del trabajo en la gestión del estipendio estudiantil en el MES, permitiendo una mayor organización y confiabilidad de la información relacionada con el proceso.

Conclusiones

Se diseñó un sistema capaz de adaptarse a las particularidades que la concepción de la Nueva Universidad impone al proceso de estipendio estudiantil y se implementó utilizando una plataforma libre, robusta y escalable. El sistema está implementado de tal forma que se almacenen registros de las principales operaciones, de manera que sea fácilmente auditable.

La comunicación con la aplicación empresarial se efectúa de forma remota o local, dependiendo de las condiciones de conectividad existentes en el CES.

Las tareas correspondientes a las secretarias docentes, así como al económico encargado han quedado automatizadas, permitiendo reducir el desgaste físico, lo que les hará ganar en eficiencia y productividad.

La implantación de este módulo en los centros universitarios adscritos al MES, junto a los manuales de normas y procedimientos hará posible una gestión del estipendio homogénea, facilitando la regulación y control del mismo.

Recomendaciones

Se propone que en una siguiente versión del módulo, el estudiante pueda ser actor del sistema y mediante una interfase *web* hacer la reclamación de su estipendio dejado de cobrar.

Continuar el presente trabajo automatizando la contabilización de las operaciones de pago del Estipendio Estudiantil, de forma que se genere el comprobante de operaciones y pueda ser exportado al Assets.NS.

Bibliografía

[1] Eckel, Bruce. Thinking in Java. Third Edition. Prentice Hall, New Jersey, 2003.

[2] Ministerio de Educación Superior. Resolución Ministerial No 173/94.

[3] Ministerio de Educación Superior. Norma 34/2005 DCF.

[4] http://java.sun.com/javaee

Referencias Bibliográficas

Alemán, Y. (2005). Aplicaciones y servicios para la Web de la Dirección de Economía. Santa Clara, Universidad Central de las Villas.

Apezteguía, M. G. (2005). Sistema GNU Módulo de Plan de Estudio. Ciudad de La Habana, Instituto Superior Politécnico "José Antonio Echeverría".

Armstrong, E., J. Ball, et al. (2005). The J2EE™ 1.4 Tutorial.

Basurto, C. K. H. (2005). Introducción a iReport. Madrid, España.

Birsan, D. (2005). "On Plug-ins and Extensible Architectures." ACM Queue Architectures Tomorrow's Computing Vol. 3, No. 2.

Danciu, T. (2005). Jasper Reports.

Espinosa, L. M. P. and L. C. Amador (2005). Sistema GNU Módulo de Estadística CES. Ciudad de La Habana, Instituto Superior Politécnico "José Antonio Echeverría".

García, L. (1995). Sistema automatizado de Control de Estudiantes en Red. Santa Clara, Universidad Central de las Villas.

Grupo Nacional de Diseño Sistema-GNU. (2005, Noviembre 2005). "Sistema para la Informatización de la Gestión de la Nueva Universidad." Taller Software MIC-MES.

Gutiérrez, A. and A. Pérez (2005). Sistema GNU Módulo de Control Docente. Ciudad de La Habana, Instituto Superior Politécnico "José Antonio Echeverría".

Marinescu, F. (2002). EJB Design Patterns, John Wiley & Sons.

Martínez, A. and Y. Vallejo (2005). Sistema GNU Módulo de Matrícula. Ciudad de La Habana, Instituto Superior Politécnico "José Antonio Echeverría".

Pino, M. C. P. (2005). Sistema GNU Módulo de Estadística MES. Ciudad de La Habana, Instituto Superior Politécnico "José Antonio Echeverría".

Rational Software Corporation (2003). Online Help.

Saúco, I. A. F. (2005). Coordinador de Transacciones Distribuidas de Datos para Java. Ciudad de La Habana, Instituto Superior Politécnico "José Antonio Echeverría".

Soley, R. (2004). Model Driven Architecture, OMG.

Stallman, R. (1985) Manifiesto GNU. Volume, DOI:

Stallman, R., V. M. Olivera, et al. (2004). Sobre Software Libre. Compilación de ensayos sobre software libre. V. M. Olivera, J. M. G. Barahona, P. d. l. H. Quirós and G. R. Martínez, Universidad Rey Juan Carlos.

The AndroMDA Core Team (2006). The AndroMDA Reference Documentation.

The PostgreSQL Global Development Group (2003). PostgreSQL Official Documentation.

Toffoli, T. (2005). A design tool iReport for JasperReports.