Universidad Central “Marta Abreu” de Las Villas

Facultad de Matemática, Física y Computación

Proyecto SIGENU. Módulo de estipendio versión 2.0

Autores

Maia Viera Cañive

Roberto Eng Acosta

Tutores

M.Sc. Libia García Águila

Ing. Lianny Ofarrill Fernández

Departamento

Departamento de promoción y desarrollo de productos informáticos

Centro de Estudios de Informática Santa Clara, Julio de 2008

“Año 50 de la Revolución”

Roberto Eng Acosta:

A mi familia por guiarme por el camino correcto.

Maia Viera Cañive:

A mi familia y en especial a mi abuelo Gustavo.

Roberto Eng Acosta:

Quiero agradecerles a mis amigos que son excepcionales. A mis tutoras por estar siempre al tanto y disponibles.

Maia Viera Cañive:

Quiero agradecerles a mis tutoras por su atención y disposición. A mis amigas Anabel Alpízar y Arianne Santiago por ser las de siempre, a mis amigos cibernéticos, y entre todos a Enriquito, "mi gran amigo". A Roberto porque a él le debo casi todo lo bueno que me llevo de estos cinco años.

Declaración de autoría

Hacemos constar que el presente Trabajo de Diploma ha sido realizado en la facultad de Matemática, Física y Computación de la Universidad Central “Marta Abreu” de Las Villas (UCLV) como parte de la culminación de los estudios de Licenciatura en Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución para los fines que estime conveniente, tanto de forma total como parcial y que además no podrá ser presentado en eventos ni publicado sin la previa autorización de la UCLV.

_____ _____

Firma del Autor Firma del Autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y que el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Tutor Firma del Jefe
Seminario de Programación

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 SIGENU (Sistema de Gestión de La Nueva Universidad) ya ha sido diseñado y comienza a explotarse 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 está implantado en la totalidad de los Centros de Educación Superior del país.

El presente trabajo es una tesis sobre el Módulo de Control de Estipendio Estudiantil que articula con dicho Sistema de Control de Estudiantes de la Nueva Universidad. Se describe la tecnología de desarrollo del módulo y su diseño. Desde el punto de vista computacional este paquete para manejar el estipendio se desarrolló utilizando una arquitectura robusta, escalable y libre. Las herramientas utilizadas son de novedoso uso en nuestro ámbito.

Summary

Automated systems to control students stipend has been developed in the Central University *"Marta Abreu de Las Villas"* and many other universities, but there is not a unified one in Superior Education Ministry. In fact many of these systems are independent and they are not always articulated to a students control system, which one has not existed in stable way and generalized for all the institutions. Thus stipend control systems have been implemented using very different structures and platforms.

The new Students Control System is the first result of the project SIGENU (System of Administration of The New University). It was developed on a free platform and it responds to the characteristics of the New University. This system it's being used in all Centers of Superior Education in the country.

The present work is a thesis about the Module of Student Stipend Control that articulates with the previous Students Control System of the New University. It is described the technology of development of the module and its design. From a technical point of view, this package to manage the stipend was developed using a robust, scalable and free architecture. The used tools are of novel use in our environment.

Índice

Introducción....10
Capítulo I. Esbozo de los elementos teóricos básicos para el desarrollo del sistema. 16
1.1 El estipendio en Cuba....16
1.2 Marco teórico 16
1.2.1 Arquitectura de plug-ins....17
1.2.2 Patrones de Diseño....18
1.2.2.1 Model-View-Controller 18
1.2.2.2 Fachada 19
1.2.2.3 Singleton 21
1.2.2.4 Value Objects....22
1.2.3 Model Driven Architecture framework....22
1.2.4 Gestor de Base de Datos. 23
1.2.5 Servidor de Aplicaciones y Jboss™....24
1.2.6 JasperReports e IReports....24
1.2.7 Aplicaciones Empresariales, J2EE, Spring framework....25
1.3 Resumen del capítulo 32
Capítulo II. Consideraciones metodológicas en torno al diseño, implementación y despliegue del sistema. 33
2.1 Modelación del sistema.....33
2.1.1 Diagramas y descripción de Casos de Uso del Sistema....33
2.1.1.1 Casos de uso del economista....34
2.1.1.2 Casos de uso de la secretaria....40
2.1.2 Descripción de los procesos fundamentales del sistema....43
2.1.2.1 Solicitar y analizar solicitudes 44
2.1.2.2 Período de pago....45
2.1.3 Diseño de las interfaces e interacción del usuario con el sistema....49
2.1.3.1 Cliente de secretaría....49
2.1.3.2 Cliente del economista 55
2.2 Implementación del sistema....57

2.2.1 Clases persistentes....57
2.2.2 Clases de servicios....71
2.2.3 Seguridad del sistema de estipendio....73
2.2.4 Capas en la implementación del módulo....74
2.2.5 Componentes del módulo....75
2.3 Resumen del capítulo....77
Capítulo III. Manual de usuario....78
3.1 Requisitos para la explotación....78
3.2 Opciones principales del módulo de la secretaria....78
3.2.1 Asignar conceptos de pago adicionales....78
3.2.2 Información del día de cobro y operaciones....79
3.2.3 Procesar solicitudes de reintegro....80
3.2.4 Procesar solicitudes de pagos retroactivos....82
3.2.5 Procesar solicitudes de préstamos....83
3.2.6 Procesar solicitudes de activación....84
3.3 Opciones principales del módulo del economista....85
3.3.1 Abrir las operaciones y establecer el día de pago....86
3.3.2 Cerrar las operaciones y crear las nóminas....87
3.3.3 Declarar los estudiantes que no cobraron....88
3.3.4 Actualizar el sistema....90
3.3.5 Modificar los conceptos de pagos.....90
3.3.6 Establecer los responsables de firmar las nóminas....91
3.3.7 Procesar solicitudes de reintegro....92
3.3.8 Procesar solicitudes de pagos retroactivos....94
3.3.9 Procesar solicitudes de Préstamos....95
3.3.10 Procesar solicitudes de Activación....96
3.4 Análisis de los resultados....98
3.5 Resumen del capítulo....98
Conclusiones generales....99
Recomendaciones....99
Anexo....101

Términos 101
Bibliotecas.....101
Enumerativos 101
Objetos de presentación 105

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 preocupación y urgencia de incrementar la eficiencia y calidad en los procesos que se realizan en los Centros de Educación Superior (CES) y el propio Ministerio de Educación Superior (MES). Con este objetivo surgió en el año 2004 el proyecto SIGENU para elaborar un Sistema de Gestión de la Nueva Universidad, 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. [1]

Entre las primeras tareas que emergieron de este proyecto, fue la creación de 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 módulos siguientes:

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**
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 [2], Control Docente [3], Plan de Estudio [4], Estadísticas del CES [5], Estadísticas del MES [6], Coordinador de Transacciones Distribuidas de Datos para Java [7].

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 y el MES, 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 desarrolló una aplicación lista para desplegar sobre la base de los mismos principios del SIGENU 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. 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.

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

Una de las principales definiciones del proyecto SIGENU 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 sea único en toda la nación y 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 y que articule con el proyecto SIGENU.

Breves Antecedentes

El primer sistema que se implementó para manejar el estipendio estudiantil en la UCLV fue un sistema desarrollado en Access bastante sencillo que no cumplía con todas las resoluciones legales relacionadas con el pago del estipendio. Luego, en el 2005 se termina un sistema que resolvía el problema de las regulaciones y publicaba, utilizando una interfaz Web, informaciones de la dirección de economía. Su autor fue Yunieski Alemán Caballero y su tutora MSc. Carmen Elena Ruiz Espino [8]. Todavía este sistema se utiliza para el control del pago a los estudiantes en la Universidad de las Villas, con algunos cambios para que esté actualizado con el nuevo Módulo de Estudiantes.

En el 2004 surge el proyecto SIGENU al consolidarse la idea de hacer un sistema común para todas las universidades que contemplara una serie de módulos que fueran la automatización de procesos rutinarios y obligados en las universidades. Se le asigna a la UCLV la creación del módulo de estipendio, ahora con los requerimientos que establece el SIGENU. En el 2006 se creó la primera versión de este módulo. Sus autores fueron Dariem Pérez Herrera y José Jorge Lorenzo Vila y sus tutores M.Sc. Libia García Águila y Lic. Alcides Morales Guedes[9].

En el año que se cierra esta primera propuesta del módulo de estipendio, la dirección del proyecto SIGENU no había reportado una versión sólida del módulo de seguridad,

por tanto no se pudo controlar este aspecto en esta versión, también surgen nuevas formas de pago y responsabilidades relacionadas al estipendio; provocando la presente investigación para llegar a estas nuevas metas.

Preguntas de Investigación

- ¿Cuáles son los nuevos requerimientos del estipendio, dados por los cambios acaecidos después del VII Congreso de la FEU?
- ¿Cómo modelar la persistencia para satisfacer los requerimientos y lograr aislar lo mejor posible nuestro diseño del resto de las clases persistentes?
- ¿Cuál es la mejor tecnología compatible con J2EE™ y SIGENU, además *Open Source*, que sea apropiada para modelar el negocio del sistema de estipendio?

Objetivo General

Desarrollar una segunda versión de un módulo de Control del Estipendio Estudiantil, integrado correctamente con la aplicación principal y los módulos existentes del proyecto SIGENU, incorporando nuevas facilidades para el control del pago y el manejo de la seguridad, no existentes en la versión anterior y utilizando software libre y específicamente *Spring* e *Hibernate* para el manejo del negocio y la persistencia respectivamente.

Objetivos Específicos

1. Crear e integrar los clientes del Sistema de Control de Estipendio al módulo de secretaría respondiendo a la plataforma *Open Source* compatible con el SIGENU e integrados con el módulo de seguridad.

2. Crear e integrar los clientes del Sistema de Control de Estipendio al módulo de economía respondiendo a la plataforma *Open Source* compatible con SIGENU.

3. Esclarecer y resolver los problemas relacionados con la integridad de las relaciones, la compartimentación de los datos y el control de acceso buscando aislar lo mejor posible el módulo de estipendio del resto de los módulos evitando problemas de versiones.

El cumplimiento de los objetivos anteriores, exige por supuesto, reformular ante todo los requisitos del Sistema de Control de Estipendio Estudiantil a la luz de los cambios ocurridos en las formas de pago por las actividades que realizan los estudiantes en la Nueva Universidad, incorporando las nuevas necesidades de información de las secretarias docentes, el estudiante y la dirección de economía en la gestión del estipendio estudiantil.

Justificación de la investigación

El proceso del estipendio es una tarea esencial en el proceso de la universidad; este módulo ahorraría tiempo, delimitaría responsabilidades y agilizaría el trabajo y al ser compatible con el SIGENU armonizaría con el resto de los módulos, aportando una utilidad importante a un sistema nacional. El hecho de que el *software* con que se desarrolló, pueda ser utilizado libremente, entiéndase sin solicitar licencia alguna a nadie, hace que su utilización sea ideal para el Gobierno.

Hipótesis

- Utilizar herramientas que cumplen con las restricciones establecidas por la dirección del proyecto de ser *Open Source* y compatibles con J2EE™ permitirá obtener un *plug-in* integrado exitosamente en el SIGENU que modela correctamente los requerimientos legales del pago del estipendio estudiantil en los CES de Cuba.
- Evitando la pesada estructura de los EJBs y usando estructuras simples no dependientes de contenedor alguno, se podrá implementar la lógica del negocio en una forma eficiente.

Estructura de la tesis

En el Capítulo I se expone el sustento teórico básico de nuestra investigación, es decir se describe brevemente las ventajas de la tecnología y las herramientas de desarrollo utilizadas.

En el Capítulo II se expone el diseño y la implementación del sistema. Se describen los procesos fundamentales del estipendio, el funcionamiento interno de la aplicación, así como la estructura de clases del código.

El Capítulo III es una guía detallada del uso del sistema con el propósito de que al módulo de secretaría en las facultades y al módulo principal en la dirección de economía se les explote sus funcionalidades de forma eficiente.

Capítulo I. Esbozo de los elementos teóricos básicos para el desarrollo del sistema.

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 en nuestro país, ya sean cubanos o extranjeros, entiéndase por esto las Resoluciones y Normas Ministeriales que han sido emitidas con tal efecto; el otro punto está encaminado al análisis de los principales elementos y tendencias tecnológicas existentes para el desarrollo del software, reduciéndonos a un entorno libre.

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 24/07 del Ministro de Educación Superior, emitida el 29 de Enero de 2007[10], 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.

Se realizaron entrevistas a las secretarias y economistas de la Universidad Marta Abreu de Las Villas, sobre la ejecución del proceso del estipendio en el centro, para apoyar el estudio de las leyes.

1.2 Marco teórico

Debido a regulaciones de la dirección del SIGENU, nuestro espectro de tecnologías se vio reducido al campo de la tecnología libre. Dentro de este subconjunto el equipo de desarrollo decidió hacer una variación; cambiar una arquitectura basada únicamente en los *Enterprise Java Bean* de la plataforma Java para mezclarlos con *Spring* e *Hibernate*, ganando en flexibilidad y ligereza.

El resto de las herramientas fueron establecidas por la dirección nacional.

1.2.1 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 para manejar las reglas de negocio que 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.[11]

En la arquitectura de *plug-ins* pura, que es la se usa en este trabajo, todo es un *plug-in* (figura 1.1). La aplicación servidora queda reducida entonces a un motor de ejecución de *plug-ins* sin ninguna funcionalidad, 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 [11]. 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.

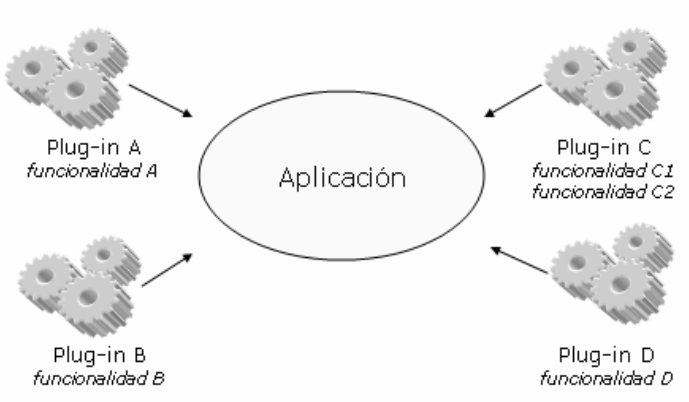


Figura 1.1 Arquitectura pura de *Plug-ins*

La estructura física de la aplicación se muestra en la Figura 1.2.

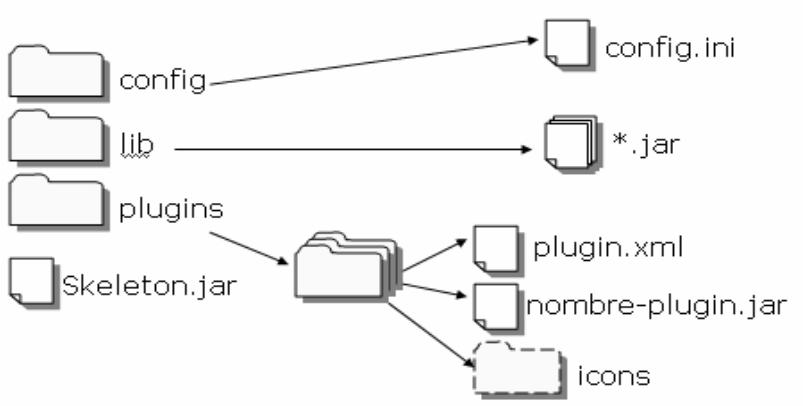


Figura 1.2 Estructura física de la aplicación cliente.

Ficheros	Descripción
config\config.ini	Configuración de la aplicación (Parámetros necesarios para establecer conexión con los servidores)
lib*.jar	Librerías de clases utilizadas por los distintos *plug-ins*
plugins\	*Plug-ins* activos en la aplicación
Skeleton.jar	Cargador de la aplicación

1.2.2 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 explicarán brevemente en que consisten los principales patrones que se utilizaron en el desarrollo del sistema.

1.2.2.1 Model-View-Controller

Este patrón surge en año 1979 y en la actualidad tiene una vigencia y una utilidad muy generalizada. 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 (MVC):

- 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 dividida en controlador y vista. La principal separación es entre presentación y modelo. 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 reflejan los cambios en el modelo. En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos de este a la vista.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

1.2.2.2 Fachada

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 dificulta el mantenimiento, 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 reutilización se reduce considerablemente al no poderse volver a usar una misma lógica por clientes diferentes.

Para resolver el problema anterior los especialistas definieron el patrón fachada. Este patrón provee una interfaz unificada para un conjunto de interfaces en un subsistema. La fachada define una interfaz un nivel más arriba que hace al subsistema más fácil de usar. [12]

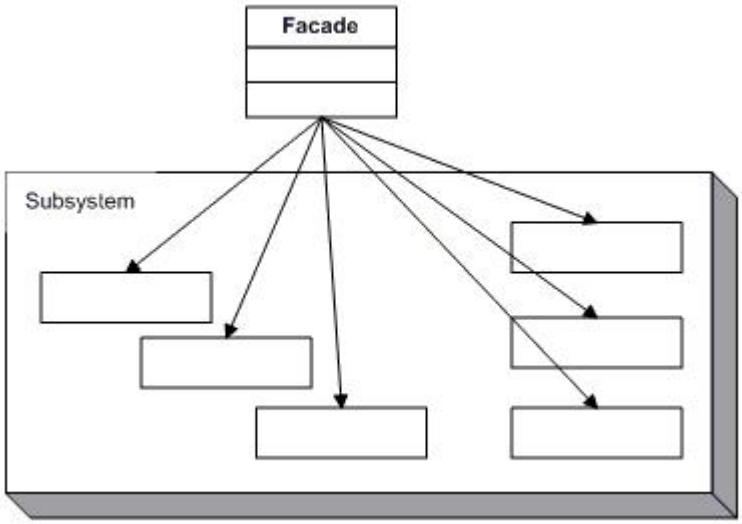


Figura 1.3 Esquema del patrón fachada.

La fachada no hace el trabajo real del sistema, sino que ésta delega responsabilidades en otros objetos.[13]

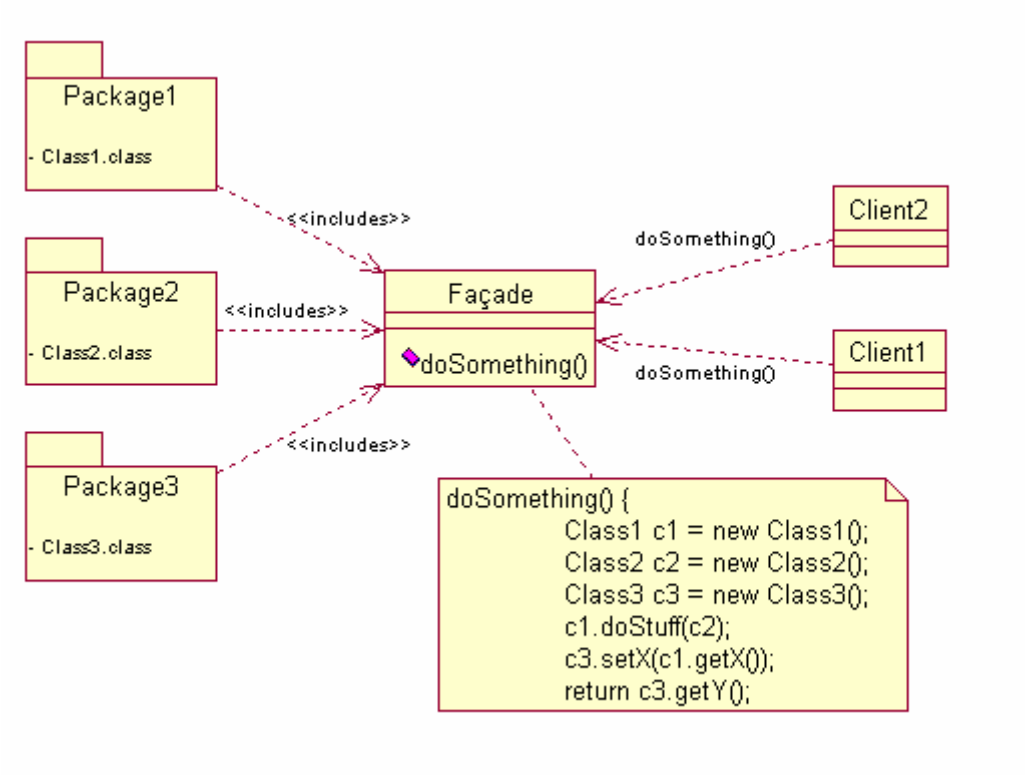


Figura 1.4 Ejemplo de uso del patrón Fachada

En el uso de este patrón se siguieron las siguientes recomendaciones.[13]:

- El número de los métodos de cada interfaz debe ser relativamente pequeño, no más de dos docenas.
- Las interfaces son la cara al cliente del subsistema. Ellas deben encapsular el conocimiento que subyace, evitando su exposición

1.2.2.3 Singleton

Otro de los patrones que se usa 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.

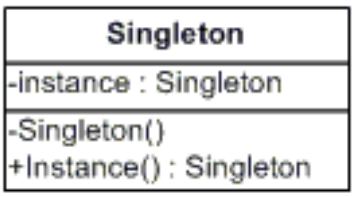


Figura 1.5 Ejemplo de uso del patrón singleton.

1.2.2.4 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 implementa 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.

1.2.3 Model Driven Architecture framework

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. En el año 2001, el OMG estableció el *framework* MDA (*Model Driven Architecture*) como arquitectura para el desarrollo de aplicaciones.

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.

Una de estas herramientas es AndroMDA, una plataforma MDA *open source*, la cual utiliza modelos, usualmente 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.

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.

1.2.4 Gestor de Base de Datos.

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 [14], 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

1.2.5 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.

Brinda un contenedor para EJB, así como servicios de nombre.

1.2.6 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. 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.

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 [15], las características más relevantes de este constructor de informes 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.
- Recopilador y exportador integrados.
- Soporta JDBC.
- Incluye asistentes para crear automáticamente informes.
- Tiene asistentes para generar los sub-reportes.

1.2.7 Aplicaciones Empresariales, J2EE, Spring framework

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 Rod Jonson [16], una aplicación empresarial correctamente diseñada e implementada, debe cumplir con los siguientes requisitos:

- Robustez: como este tipo de *software* es un producto esencial 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 una correcta división en módulos, 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 ®.

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 clásico, bajo J2EE™ es el siguiente:

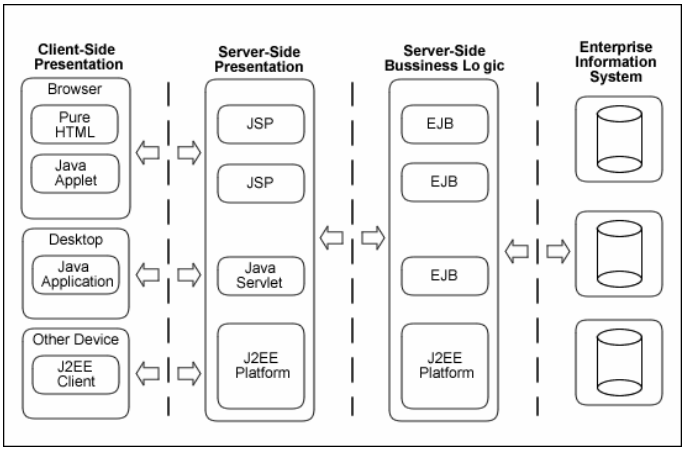


Figura 1.6 Modelo de Funcionamiento de J2EE

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. 1.6 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.

J2EE™ brinda la tecnología *Enterprise JavaBeans* ™ (EJB). Son componentes del negocio que se ejecutan en el servidor. Según Rod Jonson [17], la experiencia ha demostrado que los EJB incurren en grandes costos y proporcionan menos beneficios de los que se esperó inicialmente.

- Los EJB no necesariamente disminuyen la complejidad, en la mayoría de los casos todo lo contrario.
- Los Entiy Bean para manejar la persistencia han fallado grandemente
- Las aplicaciones usando EJB tienden a ser menos portables entre los servidores de aplicaciones que otras aplicaciones usando otras tecnologías de J2EE™, como los servlets. Son muy dependientes de su contenedor.
- Los sistemas usando EJB usualmente poseen poca o ninguna escalabilidad.
- EJB puede convertir algo simple en algo realmente complicado.

A pesar de lo anterior la tecnología EJB está muy difundida entre los desarrolladores de todo el mundo.

¿Para qué es realmente útil el uso de esta tecnología?

- Para aplicaciones que genuinamente necesiten objetos distribuidos.
- Aplicaciones que estén fuertemente basadas en el manejo de mensajes.
- Aplicaciones de finanzas que conllevan un alto costo computacional.

Las anteriores aplicaciones se considera son el 10%. Pero J2EE™ no es solo los EJB. La plataforma J2EE implementa una serie de servicios para el desarrollo de aplicaciones empresariales, como son: el servicio de nombres y directorios (JNDI), ofrece servicios de manejo de transacciones (JTA y JTS), servicio de conexión a sistemas legados (JCA), administración de recursos y manejos de hilos. El gran poder de esta plataforma cae sobre estos servicios, cuya estandarización le ha hecho mucho bien a la industria.

Para el acceso a la capa de datos J2EE™ soporta la tecnología JDBC y los *Entity Bean*.

Podemos encontrar diferentes arquitecturas J2EE™:

- Está la clásica que usa EJBs remotos y *Entity Beans*.
- La arquitectura EJB local, que usa EJBs locales
- La arquitectura Ad hoc sin EJB
- Y la que podríamos llamar Arquitectura de Contenedor Ligero, que es la que se utiliza en este trabajo.

En el 2003 ha habido un florecimiento de lo que se denominan armazones ligeras (*lightweight frameworks*), utilizando la plataforma J2EE™. Proveen manejos de objetos del negocio y servicios de empresa sin la pesada infraestructura de los EJB.

La arquitectura de Contenedor Ligero tiene como similitud con la arquitectura basada en EJB que está centrada alrededor de una capa que maneja los objetos del negocio; pero por lo demás todo cambia.

- Los objetos de negocio en vez de correr en un contenedor EJB corren en un contenedor ligero, que no está atado a ninguna tecnología específica, puede correr

en un contenedor Web, en una aplicación aislada o incluso dentro del propio contenedor de EJB si es necesario.

- El contenedor ligero provee una manera de localizar y manejar objetos del negocio. No es necesario el uso de búsquedas JNDI; mantiene un registro de los objetos de la aplicación.

Un contenedor es una bolsa dentro de la cuál el código de la aplicación se ejecuta. Los objetos de la aplicación, la mayoría de las veces objetos de negocio, corren dentro del contenedor y se dicen que son manejados por el contenedor.

¿Qué debe hacer en esencia un contenedor ligero?:

- Manejo del ciclo de vida de los objetos del negocio: una distinción con el contenedor de EJB, es que el ligero puede manejar POJOs (*Plain Old Java Object*) en vez de EJB.
- Búsquedas: debe proporcionar una vía de obtener referencias para el manejo de objetos
- Configuración: debe proveer una forma consistente para configurar los objetos que corren dentro de él. Configuraciones simples deben poderse realizar sin necesidad de recompilar el código.
- Resolución de dependencias: Debe ser capaz de resolver relaciones entre los objetos que maneja.

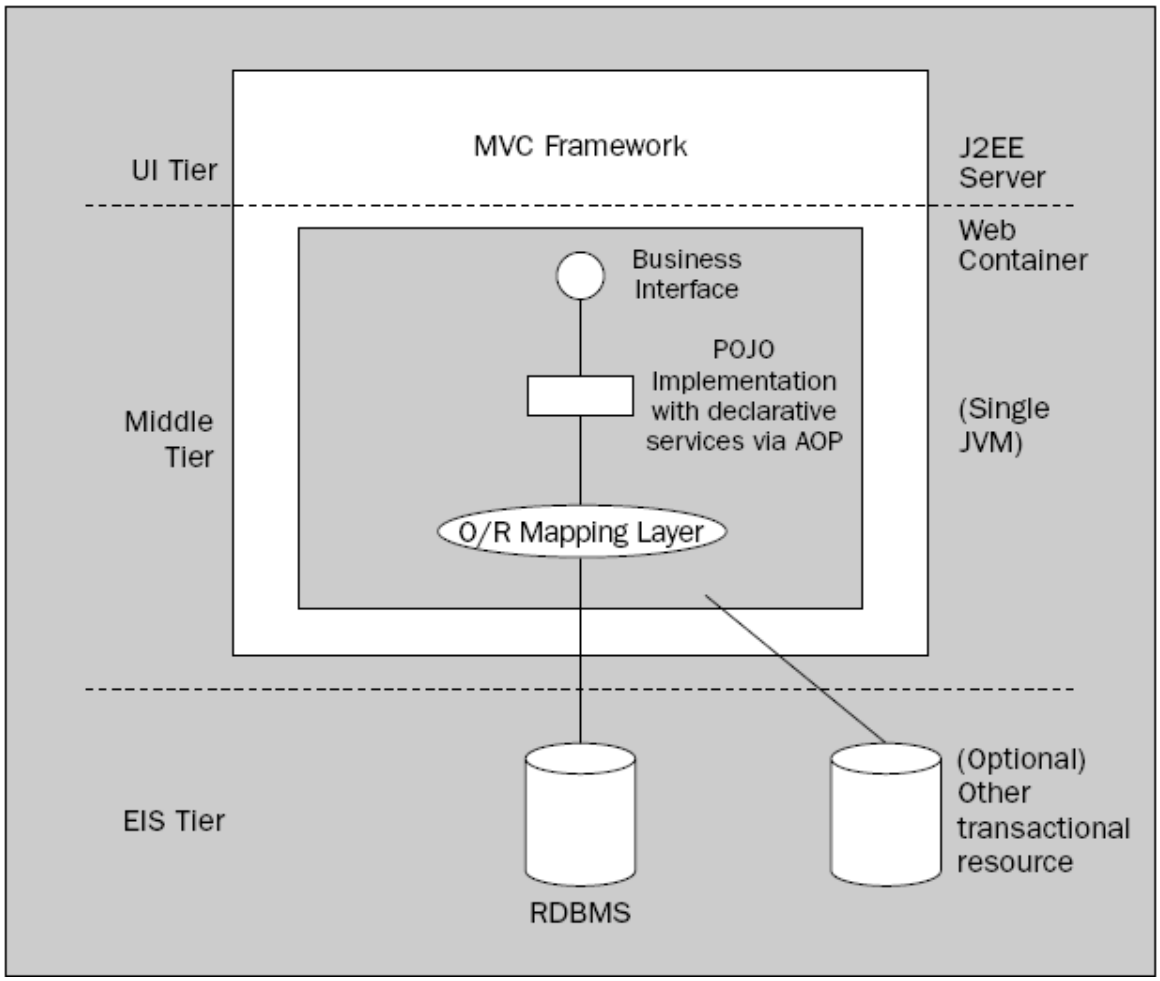


Figura 1.7 Arquitectura del Contenedor Ligero.

Algunos contenedores ligeros, disponibles y libres actualmente son: Spring y PicoContainer. Spring es un popular producto *Open Source*, dedicado a proveer simples y efectivas soluciones para problemas comunes en la plataforma J2EE™. No fue concebido como una alternativa para EJB, pero provee fuertes utilidades como el manejo de transacciones declarativas, para objetos planos de Java.

En el corazón de Spring está la Inversión de Control (*Inversión of Control*, Ioc), que al aplicarla a los proyectos su código se convierte más simple, más fáciles de entender y probar. La Inversión de Control también es conocida por Inyección de Dependencias (*Dependency Injection*). Cualquier aplicación no trivial está compuesta por dos o más clases que colaboran entre ellas para representar alguna lógica de negocio. Lo normal es que sea cada instancia la responsable por obtener sus referencias a otras instancias con las cuales colabora. Aplicando IoC, a los objetos le son dadas sus

dependencias en el momento de su creación por una entidad externa que coordina cada objeto en el sistema, es decir las dependencias se le inyectan al objeto. Por lo tanto IoC significa una inversión de la responsabilidad de cómo un objeto obtiene sus dependencias. [18]

Spring está compuesto por siete módulos bien definidos:

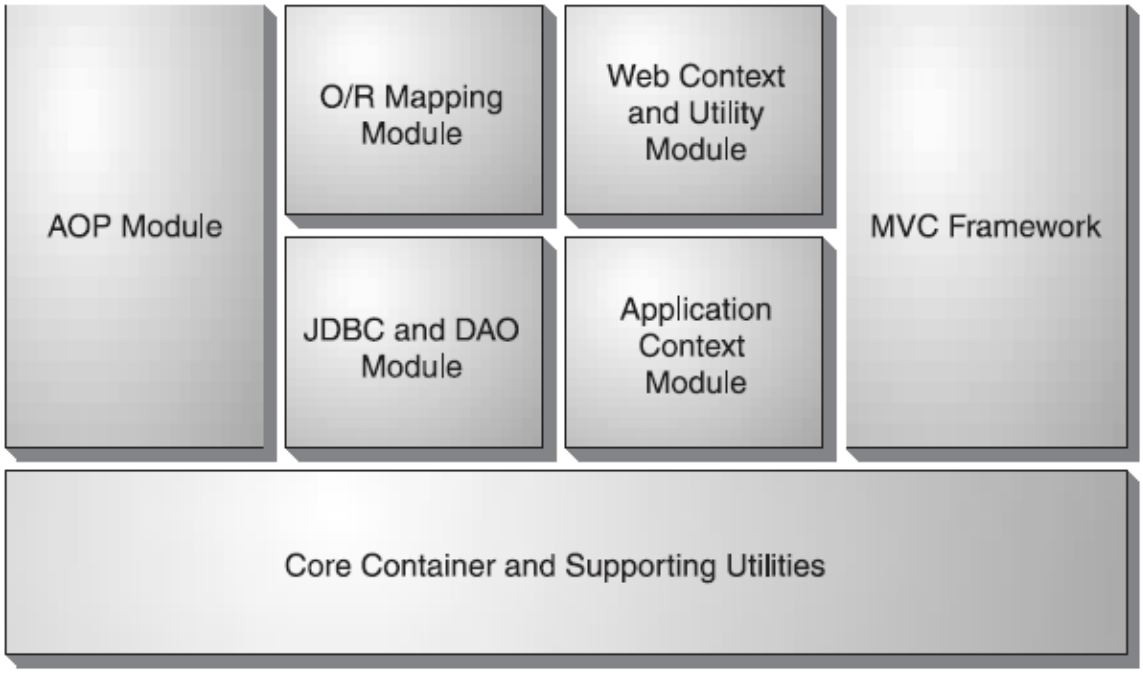


Figura 1.8 Módulos del *Framework* Spring.

- El núcleo del contenedor (*Core*): provee las funcionalidades fundamentales de *Spring*. Este módulo contiene los *BeanFactory*, que son una implementación del patrón factoría que aplica IoC para separar del código la configuración de la aplicación y las especificaciones de dependencia.
- El módulo Contexto de Aplicación (*Aplication Context*): que extiende el concepto de *BeanFactory*, adicionándole soporte para mensajes internacionalizados (I18N), eventos del ciclo de vida de los objetos y validación. Además provee servicios como el correo, acceso JNDI, integración con EJB, acceso remoto y planificación.
- Módulo AOP (*Aspect Oriented Programing: AOP*): provee un excelente soporte para el manejo de la programación orientada a aspectos.
- El módulo DAO (*Data Access Object*) y abstracción JDBC: abstrae al desarrollador de lo reiterativo y enmarañado que puede ser el código de la base de

datos usando JDBC puro, dejándolo limpio y simple. Contiene una capa de excepciones con significado que maneja los mensajes de error dados por los servidores de las bases de datos. Además provee servicios para el control de transacciones para objetos en la aplicación Spring utilizando el módulo AOP.

- Módulo de mapeo Objetos/Relación (O/R *Mapping*): para los que deseen el uso del mapeo O/R (*Object/Relational mapping*: ORM) en vez de JDBC, está hecho este módulo. Spring no implementa su propia solución sino que provee enlaces a estructuras ya implementadas como Hibernate, JDO e iBATIS SQL Maps.
- Módulo Web (*Web Context and Utility*): provee un contexto que es apropiado para aplicaciones basadas en Web. Además contiene soporte para numerosas tareas orientadas a Web y Jacarta Struts.
- El módulo MVC: es una implementación de Modelo/Vista/Controlador (*Model-View-Controller:* MVC) completamente dotada de utilidades para construir aplicaciones Web. Spring puede integrase fácilmente con otras implementaciones de MVC como Struts.

1.3 Resumen del capítulo

En este capítulo se han descrito los patrones de diseño utilizados y se ha explicado las principales ventajas de las tecnologías de desarrollo de software y herramientas utilizadas.

El módulo de estipendio aporta como novedad al resto del sistema el uso de *Spring* e *Hibernate*, dos *frameworks* ligeros que se adaptan muy bien a los requisitos del SIGENU.

Capítulo II. Consideraciones metodológicas en torno al diseño, implementación y despliegue del sistema.

Haciendo un análisis de la resolución 24/07 y de la información obtenida de las entrevistas con los economistas que atienden el estipendio en la UCLV, se diseñó el sistema de estipendio, con la comprensión teórica del proceso que lo sustenta.

El desarrollo del mismo fue dividido en tres etapas fundamentales:

1. Selección del entorno y herramientas de desarrollo, manteniendo la compatibilidad con el SIGENU. Descritas en el capítulo uno.
2. Modelado de los procesos del sistema.
3. Modelo de la implementación y componentes del sistema.

2.1 Modelación del sistema.

El modelo del sistema incluye la definición de los casos de uso del sistema, la descripción de los procesos fundamentales que implementa el módulo y el diseño de las interfaces de usuario.

2.1.1 Diagramas y descripción de Casos de Uso del Sistema

Los diagramas de casos de uso delinean gráficamente el comportamiento del sistema. Representan como el sistema es usado desde la perspectiva del actor. Muestran 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. [19]

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.

2.1.1.1 Casos de uso del economista

A continuación (figura 2.1) se muestra el diagrama de casos de uso del sistema para el Economista.

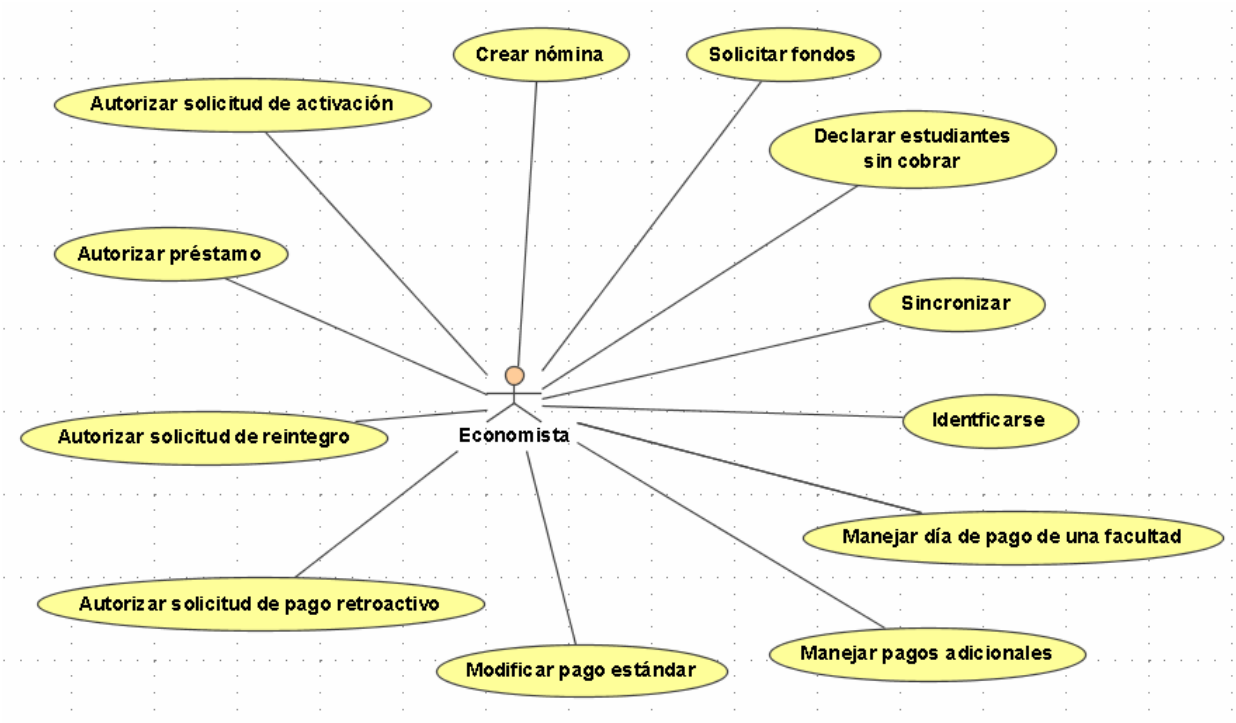


Figura. 2.1 Casos de uso del actor Economista.

En el proceso de estipendio interviene activamente el economista encargado del mismo. Esta persona generalmente es única en el CES, en caso de no ser única, igualmente es un solo actor. Este actor es el encargado 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	Economista
Precondiciones	El Economista 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	Economista
Precondiciones	El Economista 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	Economista
Precondiciones	El Economista 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 actualizado.

Caso de uso	Eliminar pago adicional
Actor	Economista

Precondiciones	El Economista 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	Economista
Precondiciones	El Economista debe estar autenticado en el sistema. 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	Economista
Precondiciones	El Economista 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 Se modifican las fechas de pago y se reinician el estado de las operaciones.	
Poscondiciones	El sistema queda preparado para realizar un nuevo pago a la facultad involucrada.

Caso de uso	Generar nómina
Actor	Economista
Precondiciones	El Economista debe estar autenticado en el sistema, la facultad debe tener un día de pago asignado y debe haberse abierto el período de pago que equivale a cerrar las operaciones.
Propósito	Hacer el cálculo del dinero que hay que pagarle a cada estudiante de la facultad.
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. Queda registrado en el historial de forma detallada los pagos que forman el monto total del estipendio de cada estudiante.	
Poscondiciones	Es posible la impresión de los reportes de las nóminas.

Caso de uso	Autorizar solicitudes de reintegro
Actor	Economista
Precondiciones	El Economista 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 último mes pagado.
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 activación
Actor	Economista
Precondiciones	El Economista debe estar autenticado en el sistema y la solicitud debe haber sido ingresada por la secretaria.

Propósito	Autorizar las solicitudes de activación a estudiantes que por algún motivo fueron desactivados del sistema de estipendio y después de un análisis de las razones se decidió reactivarlos.
Resumen Mediante esta operación, se autoriza la solicitud de activación hecha por un estudiante.	
Poscondiciones	El estudiante queda nuevamente activo y listo para cobrar el próximo mes de pago.

Caso de uso	Autorizar solicitudes de pago retroactivo
Actor	Economista
Precondiciones	El Economista 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	Economista
Precondiciones	El Economista debe estar autenticado en el sistema y la solicitud debe haber sido ingresada por la secretaria. El estudiante ya ha efectuado el contrato económico.
Propósito	Activar una solicitud de préstamo al estudiante.
Resumen Mediante esta operación, se autoriza la solicitud de préstamo hecha por un estudiante.	
Poscondiciones	El estudiante queda sujeto a un préstamo reintegrable.

Caso de uso	Declarar estudiantes sin cobrar
Actor	Economista
Precondiciones	El Economista debe estar autenticado en el sistema.
Propósito	Notificar los estudiantes que han dejado de cobrar en el día de pago anterior.
Resumen Para cerrar un período de pago es necesario notificar los estudiantes que dejaron de cobrar.	
Poscondiciones	Los estudiantes declarados, quedan disponibles para realizar una solicitud de reintegro.

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

Caso de uso	Solicitar fondo
Actor	Economista
Precondiciones	El Economista 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.

Caso de uso	Sincronizar
Actor	Economista
Precondiciones	El Economista debe estar autenticado en el sistema.
Propósito	Sincronizar las tablas de estipendio con las tablas de *Student* y *HelpStudent* del sistema.
Resumen Mediante esta operación, se actualizan el estado, los pagos por ayudantía y conceptos estándares de los estudiantes, que pueden haber sido modificados por otros módulos como el de matrícula y alumnos ayudantes.	
Poscondiciones	Los estudiantes que pueden cobrar quedan actualizados en el sistema así como sus conceptos de pagos.

2.1.1.2 Casos de uso de la secretaria

A continuación (figura 2.2) se muestra el diagrama de casos de uso del sistema para la secretaria.

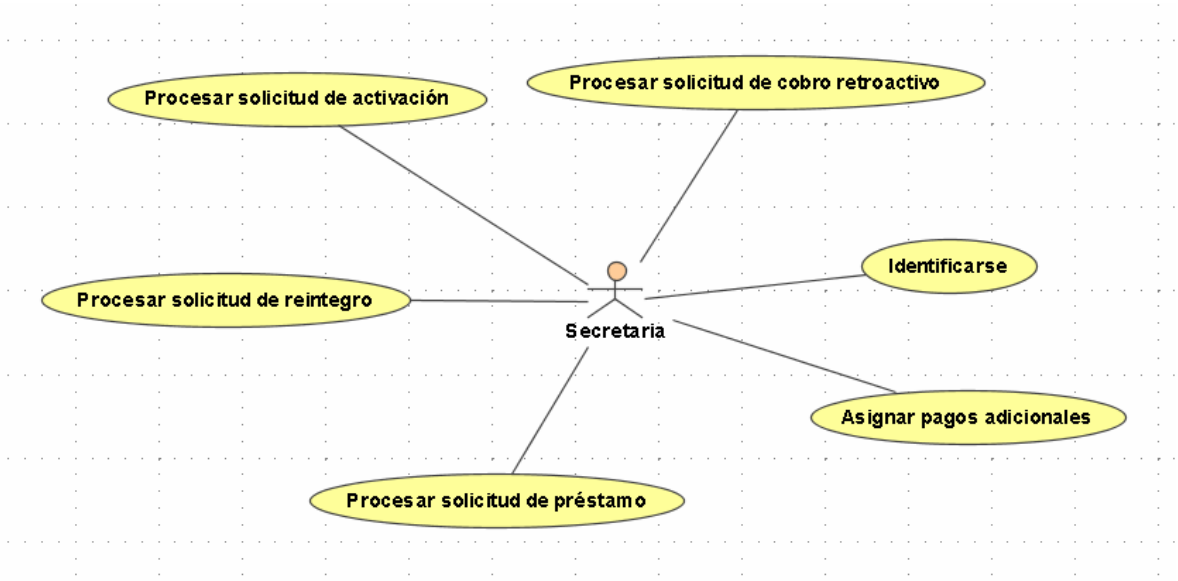


Figura. 2.2 Casos de uso del actor secretaria.

Cada facultad debe tener una persona encargada de gestionar todas las actividades del estipendio estudiantil relacionadas con sus estudiantes. Normalmente es la propia secretaria de la facultad.

Caso de uso	Procesar solicitud de cobros retroactivos
Actor	Secretaria
Precondiciones	La Secretaria debe autenticarse en el sistema y las operaciones del día de pago actual deben estar abiertas.
Propósito	Incorporar al sistema una solicitud de cobros retroactivos, para que el economista la apruebe.
Resumen Si ocurre alguna irregularidad en el pago a un estudiante, la Secretaria ingresa la solicitud al sistema.	
Poscondiciones	La solicitud queda en espera de aprobación por el economista del sistema.

Caso de uso	Procesar solicitud de activación
Actor	Secretaria
Precondiciones	La Secretaria debe autenticarse en el sistema y las operaciones del día de pago actual deben estar abiertas.
Propósito	Incorporar al sistema una solicitud del tipo activación, para que el economista la apruebe.
Resumen Si por cualquiera de los motivos posibles el estudiante resulta desactivado en el sistema de estipendio, puede solicitar su reactivación por este medio.	
Poscondiciones	La solicitud queda en espera de aprobación por el economista del sistema.

Caso de uso	Procesar solicitud de reintegro
Actor	Secretaria
Precondiciones	El estudiante debe estar declarado como un estudiante que dejó de cobrar. La Secretaria debe autenticarse en el sistema y las operaciones del día de pago actual deben estar abiertas.
Propósito	Incorporar al sistema una solicitud de reintegro, para que el economista la 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.	
Poscondiciones	La solicitud queda en espera de aprobación por el economista 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. Las operaciones del día de pago deben estar abiertas.
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. Si es aprobada la solicitud del estudiante la Secretaria la ingresa al sistema.	
Poscondiciones	La solicitud queda en espera de aprobación por el economista 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.

Caso de uso	Asignar pagos adicionales
Actor	Secretaria
Precondiciones	Los conceptos de pago deben haberse introducido en el sistema.
Propósito	Asignarle al estudiante otro tipo de concepto de pago que no se asignan automáticamente por el sistema.
Resumen Seleccionar de los otros conceptos de pagos en el sistema aquellos que le corresponden al estudiante.	
Poscondiciones	El estudiante en los próximos cobros obtendrá un dinero adicional por el concepto de pago establecido.

2.1.2 Descripción de los procesos fundamentales del sistema

El proceso del estipendio se puede separar en dos procesos fundamentales:

- Proceso pedir y revisar solicitudes.
- Proceso del pago a los estudiantes.

Para poder comenzar el proceso relacionado con las solicitudes de una facultad, el economista debe abrir las operaciones de esa facultad. Mientras las operaciones están abiertas la secretaria tiene la posibilidad de ingresar al sistema las solicitudes y el economista encargado de su análisis revisarlas. Luego se cierran las operaciones y se abre un período de pago, para el cual tienen que haber quedado establecidos un día de pago y el mes o los meses a pagar en ese tiempo. En este período se calculan y generan las nóminas de la facultad y para los meses seleccionados con antelación, después del cálculo pueden imprimirse las solicitudes de fondo y las nóminas de pago; todo queda listo entonces para pagarles a los estudiantes. Por último deben registrarse en el sistema aquellos estudiantes que no recibieron el estipendio, estando las condiciones creadas para cerrar el pago y abrir nuevamente las operaciones comenzando el ciclo.

2.1.2.1 Solicitar y analizar solicitudes

En esta sección se muestran un grupo de diagramas que pueden ayudar a comprender el subproceso de las solicitudes dentro del proceso de estipendio.

El siguiente diagrama muestra la forma general de las peticiones de las solicitudes. Se buscan los estudiantes que desean las solicitudes, se selecciona el estudiante y se puede leer, modificar, crear o eliminar las solicitudes, después de hacer los cambios deseados deben salvarse para que queden registrados en el sistema.

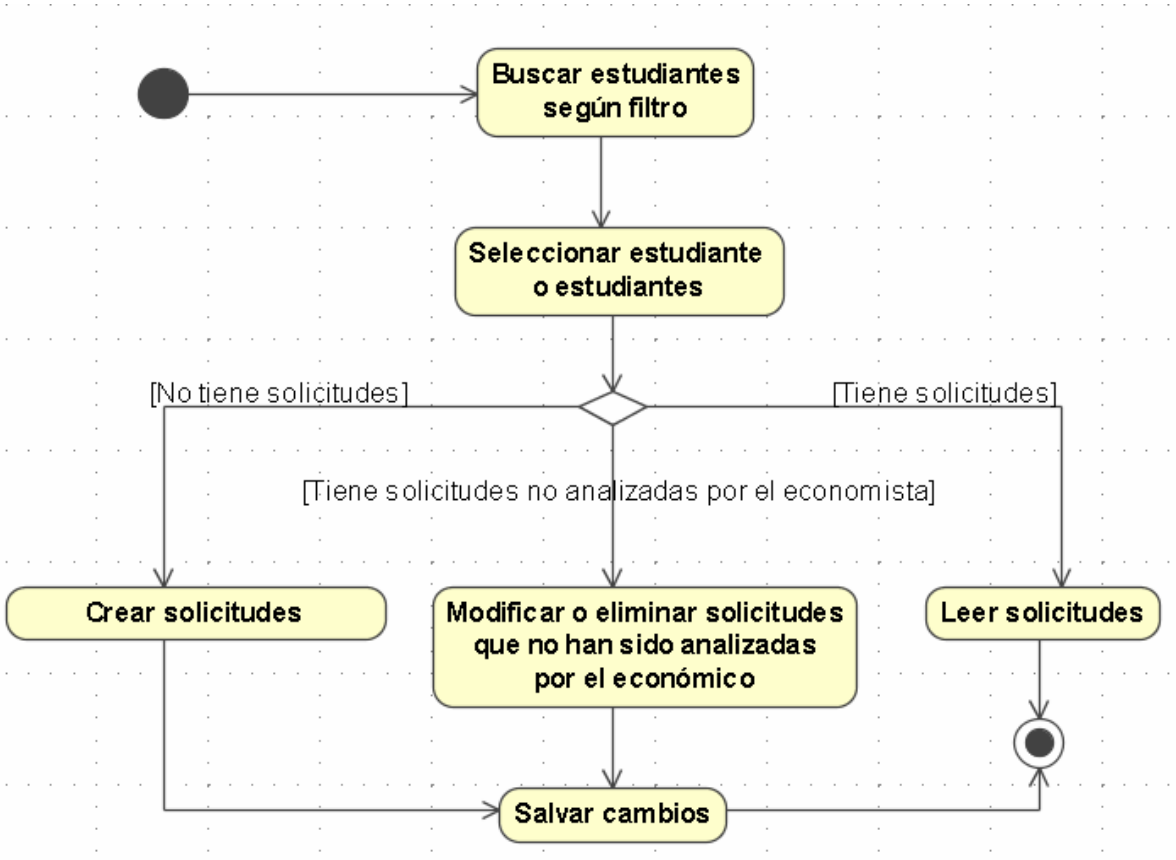


Figura 2.3. Estructura general de la petición de las solicitudes

Cuando la solicitud ya fue salvada queda disponible para el economista encargado de revisar que esté bien formada y no incumpla ninguna regla ni contradiga otras informaciones. El siguiente diagrama es la forma general de este proceso. Se buscan las solicitudes y se van revisando una por

una y aprobándola o rechazándola en dependencia del análisis, de igual modo deben salvarse los cambios para que quede registrada la decisión en el sistema.

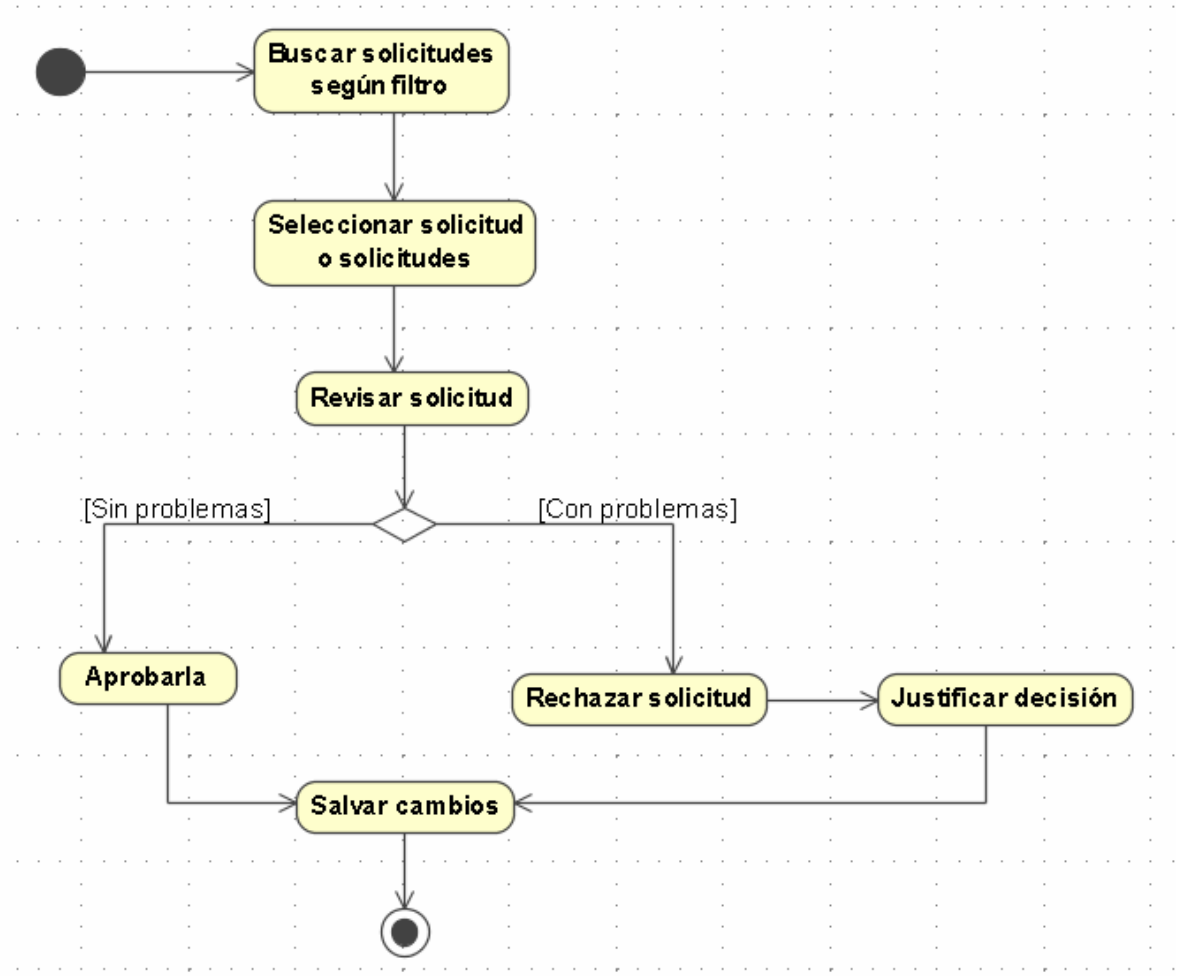


Figura 2.4. Diagrama del análisis de las solicitudes en economía.

2.1.2.2 Período de pago

En este tiempo es cuando se produce la solicitud del dinero al banco, y se paga a los estudiantes. Este proceso se describe en la figura 2.5. Los períodos de pago se abren por facultades.

Este proceso se debe iniciar y cerrar, subprocesos que se describen más adelante. El fundamento de este período es hacer el cálculo de las nóminas de la facultad. Por cada carrera y año dentro de la carrera se toman los estudiantes activos en el sistema de estipendio, que serían los integrantes de

esa nómina. Con las nóminas en el sistema se pueden generar los reportes de solicitud de fondo y las planillas resumen de las mismas.

Llegados los días de pago establecidos con anterioridad, se le realiza el pago a los estudiantes, después del día de recuperación el economista tiene que introducir los estudiantes que no cobraron de cada planilla.

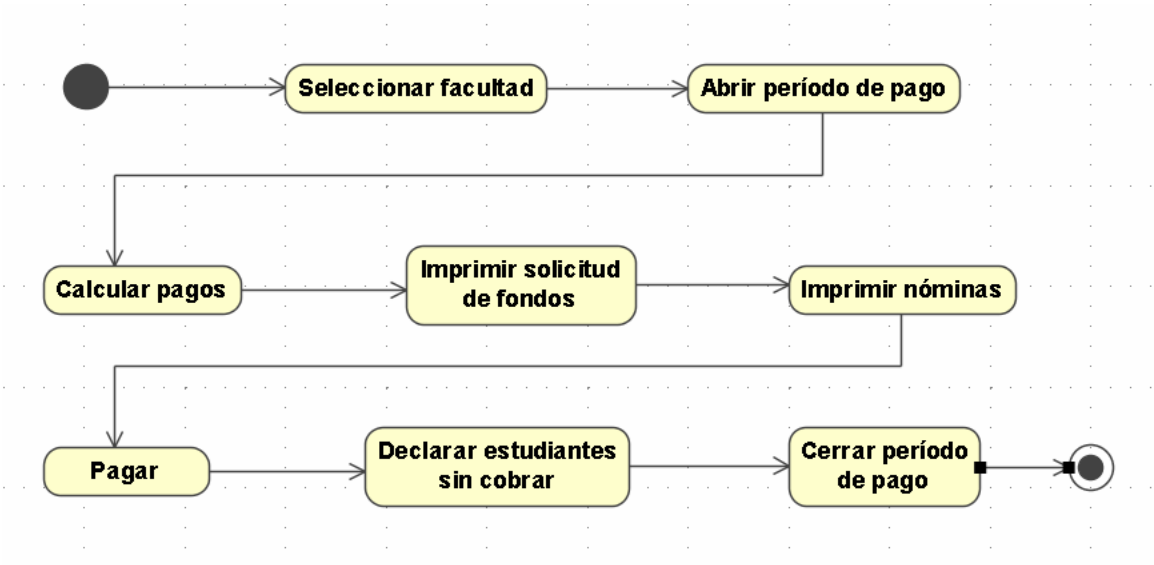


Figura 2.5. Diagrama del proceso realizado durante el período de pago

En la figura 2.6 se describe el subproceso de inicio del período de pago. El proceso se inicia cerrando las operaciones de la facultad a la que se le está iniciando el período, a cada estudiante de esa facultad que tenga asignado una solicitud de activación aprobada se le reactiva su estado en el sistema de estipendio, así mismo sucede con los que tengan solicitudes de nuevo préstamo aprobadas, se le crea un préstamo y se pasa como préstamo activo y si tienen una solicitud de cancelación de préstamo se le cancela el mismo. Las solicitudes aprobadas que ya fueron ejecutadas se eliminan del sistema.

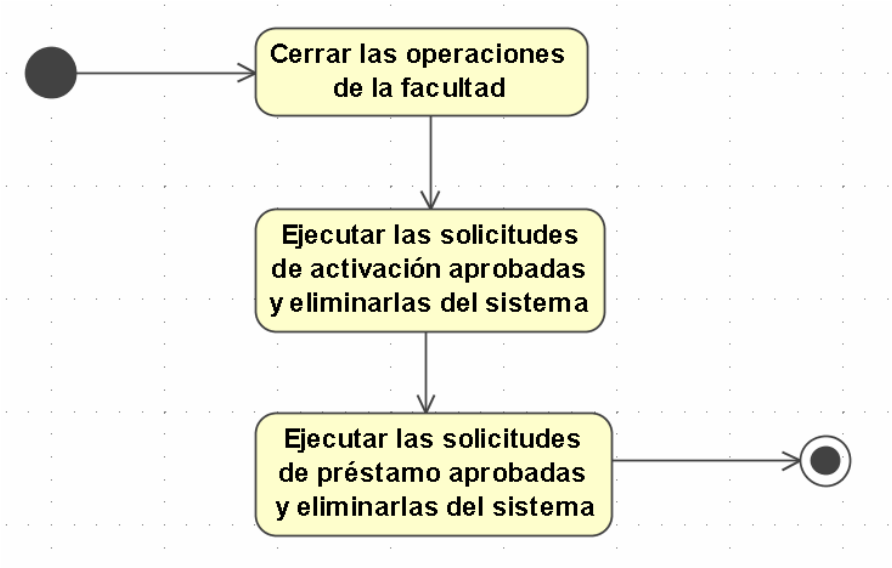


Figura 2.6. Abrir período de pago. Diagrama de actividad.

En la figura 2.7 se describe el subproceso del cálculo de las nóminas de forma general y en la figura 2.8 se especifican los pagos posibles de un estudiante. Si el estudiante quedó sin cobrar un mes e hizo una solicitud de reintegro se le reintegra el dinero de la nómina anterior. Si tiene una solicitud de pago retroactivo aprobada se le adiciona un pago retroactivo, si tiene préstamo activo se le hace un pago reembolsable, si es alumno ayudante se le paga una subvención en correspondencia con el tipo de ayudantía que realiza así como el nivel; el estudiante en dependencia de su nacionalidad, tipo de matrícula y año que cursa cobra un salario base, el que se adiciona también al monto total de la nómina y por último si tiene asignados otros conceptos de pago se registra el dinero por cada concepto que tenga asignado y se adicionan como otro pago.

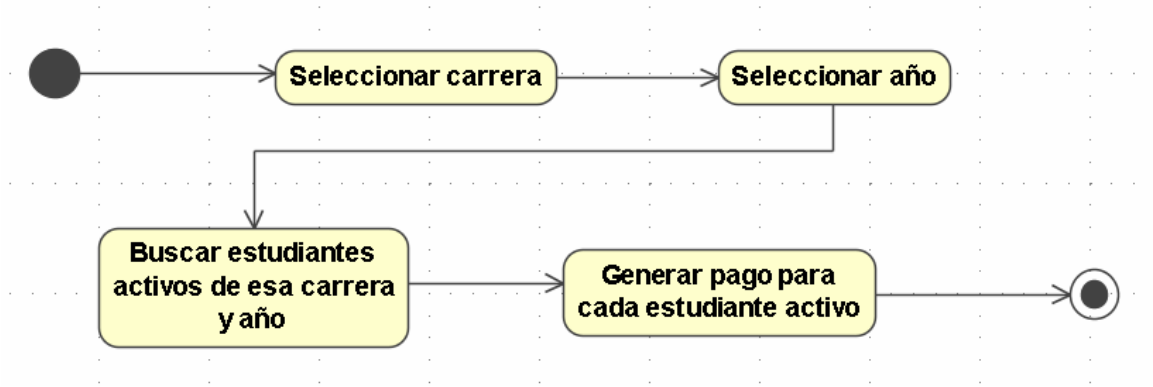


Figura 2.7. Calcular pago de las nóminas. Diagrama de actividad.

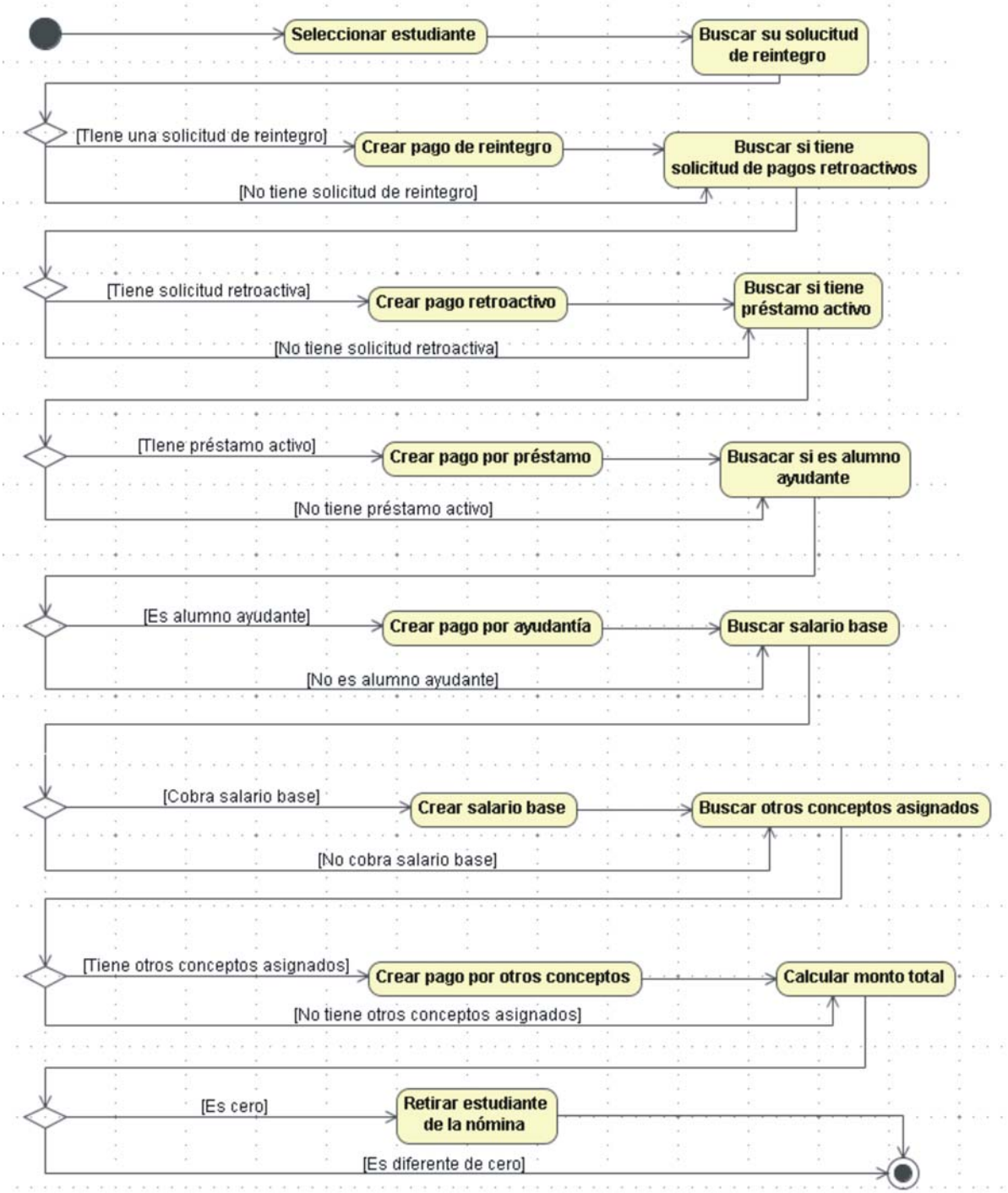


Figura 2.8. Pago de un estudiante. Diagrama de actividad.

2.1.3 Diseño de las interfaces e interacción del usuario con el sistema

Como se ha definido antes son dos los actores del módulo de estipendio, la secretaria y el economista. Para cada uno se concibió un cliente de escritorio. El cliente de secretaría se caracteriza por las facilidades para la gestión de las solicitudes y el del economista por la gestión del pago y la revisión de las peticiones de las secretarías.

2.1.3.1 Cliente de secretaría

El cliente de secretaría está formado por cuatro ventanas para las peticiones de las distintas solicitudes (solicitud de pago retroactivo, solicitud de reintegro, solicitud de préstamo y solicitud de activación), una ventana para la asignación de otros conceptos de pago y un diálogo que brinda información sobre las operaciones y el día de cobro de la facultad.

La figura 2.9 muestra la ventana para solicitar los pagos retroactivos, esta unida a las tablas 2.1 y 2.2 que describen el flujo de los eventos desencadenados brinda una panorámica bastante completa sobre el funcionamiento de la interfaz.

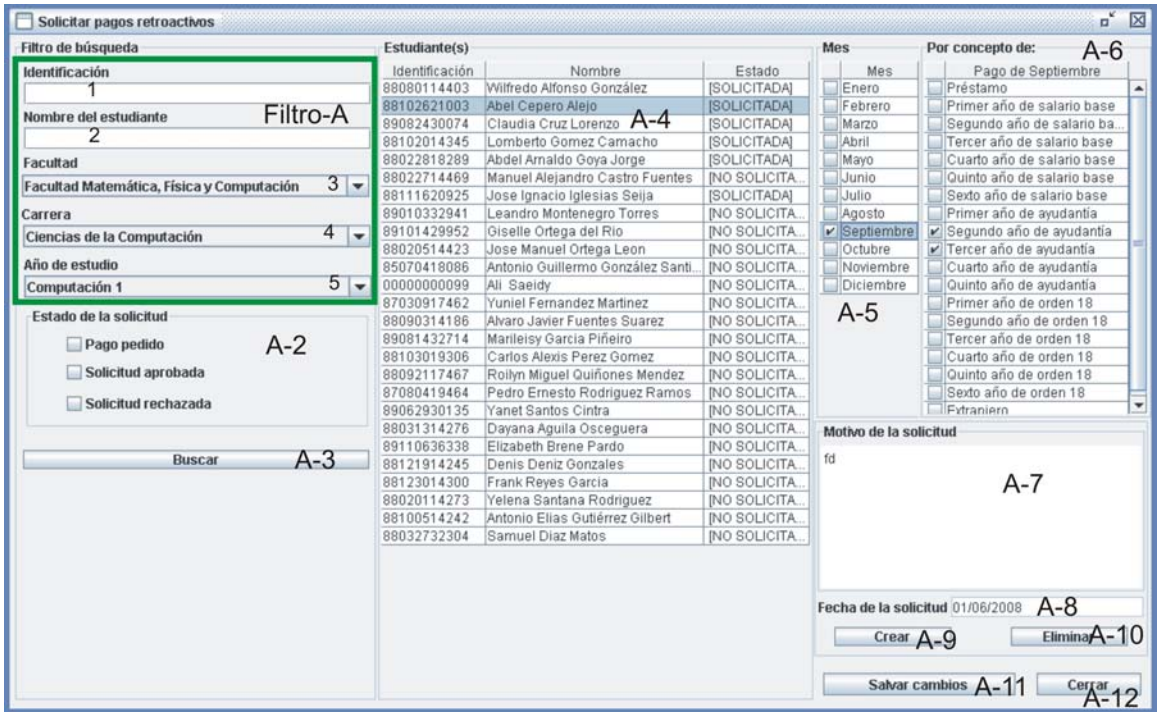


Figura 2.9. Ventana para la solicitud de pagos retroactivos

El Filtro-A agrupa varios parámetros. En el componente 1 se especifica el carné de identidad del estudiante o una subcadena del mismo, así mismo sucede con el componente 2 pero con el nombre del estudiante, incluyendo los dos apellidos. En el componente 3 se especifica la facultad, en el 4 la carrera de esa facultad y en el 5 el año de estudio. Se pueden especificar los parámetros de búsqueda que se deseen. Este filtro es común a todas las ventanas de solicitudes.

¿Qué hace el usuario?	¿Qué hace el sistema?
1. Establecer los parámetros de búsqueda de los estudiantes en el Filtro-A. Si se desea solo buscar aquellos estudiantes que tengan solicitudes pedidas aprobadas o rechazadas se pueden marcar los cuadros de chequeo que se desee en A-2.	
2. Cuando queden definidos los parámetros deseados se hace la búsqueda de los estudiantes con sus solicitudes de pagos retroactivos presionando el botón A-3.	3. Busca los estudiantes que cumplan con los parámetros establecidos en el Filtro-A, si se especificó algún tipo de solicitud en A-2, se escogen de este grupo solo aquellos estudiantes que tengan solicitudes de ese tipo. Para los estudiantes que queden después de aplicar el filtro se buscan sus solicitudes para ser devueltas también, si no tienen ninguna se asigna solicitud vacía.
4. Selecciona un estudiante en el componente A-4.	
5. En A-5 se selecciona el mes que se le debe al estudiante y en A-6 se seleccionan los conceptos de pago que se le deben en ese mes, en A-7 se	

justifica la solicitud, en A-8 se muestra la fecha de la solicitud.	
6. Después de especificados los datos de la solicitud se puede crear una petición de pagos retroactivos presionando el botón A-9 o si el estudiante tenía una solicitud que no había sido revisada aún por el economista se puede eliminar presionando el botón A-10. Si lo que se desea es modificar se puede presionar el botón A-9.	
7. Para salvar los cambios en el sistema se presiona el botón A-11.	8. Se revisan las solicitudes que no entren en conflicto con ningún dato en el sistema luego se procede a ejecutar la acción que corresponde, si es eliminar una solicitud, crearla o modificarla.

Tabla 2.1. Flujo normal de los eventos de la ventana de solicitud de pagos retroactivos.

¿Qué hace el usuario?	¿Qué hace el sistema?
El usuario presiona el botón A-9 y no ha introducido todos los datos de la solicitud.	El sistema lanza una excepción especificando el dato que le falta de la solicitud. Volver al paso 5.
El usuario presiona el botón A-11 y las operaciones están cerradas.	Lanza una excepción diciendo que no se puede proceder porque las operaciones están cerradas.
El usuario presiona el botón A-12.	Se cierra la aplicación y se pierden los cambios que no hayan sido salvados.
El usuario presiona A-3 dos veces consecutivas sin presionar entre las dos veces A-11.	El sistema realiza la nueva búsqueda y se pierden los cambios que no hayan sido guardados.

Tabla 2.2. Flujo alternativo de los eventos de la ventana de solicitud de pagos retroactivos.

La figura 2.10 muestra la interfaz para las solicitudes de préstamo y las tablas 2.3 y 2.4 el flujo de los eventos. En esta interfaz se pude manejar la petición de la creación de un nuevo préstamo como también la cancelación de un préstamo activo.

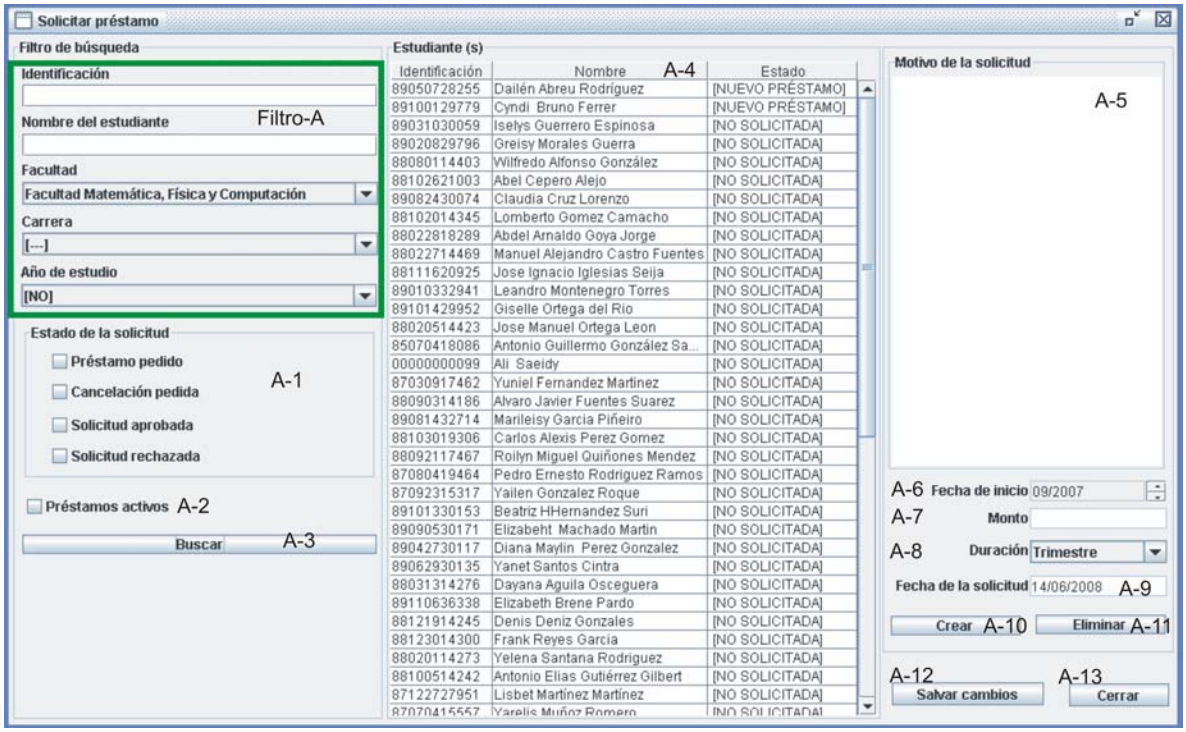


Figura 2.10. Ventana para las solicitudes de los préstamos.

¿Qué hace el usuario?	¿Qué hace el sistema?
1. Establecer los parámetros de búsqueda de los estudiantes en el Filtro-A. Si se desea solo buscar aquellos estudiantes que tengan solicitudes pedidas de nuevo préstamo o cancelaciones pedidas o solicitudes aprobadas o rechazadas se pueden marcar los cuadros de chequeo que se desee en A-1. En A-2 se puede especificar que se	

quieren filtrar los estudiantes que tengan préstamos activos.	
2. Cuando queden definidos los parámetros deseados se hace la búsqueda de los estudiantes con sus solicitudes de pagos retroactivos presionando el botón A-3.	3. Busca los estudiantes que cumplan con los parámetros establecidos en el Filtro-A, si se especificó algún tipo de solicitud en A-1, se escogen de este grupo solo aquellos estudiantes que tengan solicitudes de ese tipo, y si se marcó A-2 se incluyen en el grupo los que tengan préstamos activos. Para los estudiantes que queden después de aplicar el filtro se buscan sus solicitudes para ser devueltas también, si no tienen ninguna se asigna solicitud vacía.
4. Seleccionar un estudiante en el componente A-4.	
5. En A-5 se justifica la solicitud, en A-6 se especifica la fecha de inicio del préstamo, en A-7 se determina el monto del préstamo, en A-8 la duración del préstamo, en A-9 la fecha de la solicitud.	

6. Después de especificados los datos de la solicitud se puede crear un nuevo préstamo o cancelar uno activo presionando el botón A-10 o si el estudiante tenía una solicitud que no había sido revisada aún por el economista se puede eliminar presionando el botón A-11. Si lo que se desea es modificar una ya existente se puede presionar le botón A-10.	
7. Para salvar los cambios en el sistema se presiona el botón A-12.	8. Se revisan las solicitudes que no entren en conflicto con ningún dato en el sistema, luego se procede a ejecutar la acción que corresponde, si es eliminar una solicitud, crearla o modificarla.

Tabla 2.3. Flujo normal de los eventos de la ventana de solicitudes de préstamos.

¿Qué hace el usuario?	¿Qué hace el sistema?
El usuario presiona el botón A-10 y no ha introducido todos los datos de la solicitud.	El sistema lanza una excepción especificando el dato que le falta de la solicitud. Volver al paso 5.
El usuario presiona el botón A-12 y las operaciones están cerradas.	Lanza una excepción diciendo que no se puede proceder porque las operaciones están cerradas.
El usuario presiona el botón A-13.	Se cierra la aplicación y se pierden los cambios que no hayan sido salvaos.
El usuario presiona A-3 dos veces consecutivas sin presionar entre las dos veces A-12.	El sistema realiza la nueva búsqueda y se pierden los cambios que no hayan sido guardados.

El usuario presiona el botón A-12 y ha introducido un préstamo que paga más de 100 pesos mensuales.	El sistema lanza una excepción informando que no puede proceder porque no se permiten préstamos superiores a 100 pesos. Volver al paso 4.

Tabla 2.4. Flujo alternativo de la ventana de solicitud de préstamo.

2.1.3.2 Cliente del economista

El cliente del economista tiene cuatro interfaces para el análisis de las solicitudes, una para el manejo de los dos tipos de conceptos, una para sincronizar el sistema, una para abrir el período de pago, otra para cerrarlo y una para declarar a los estudiantes sin cobrar.

En la próxima ventana se funden dos acciones que están muy relacionadas, cerrar un pago y establecer el día del próximo.

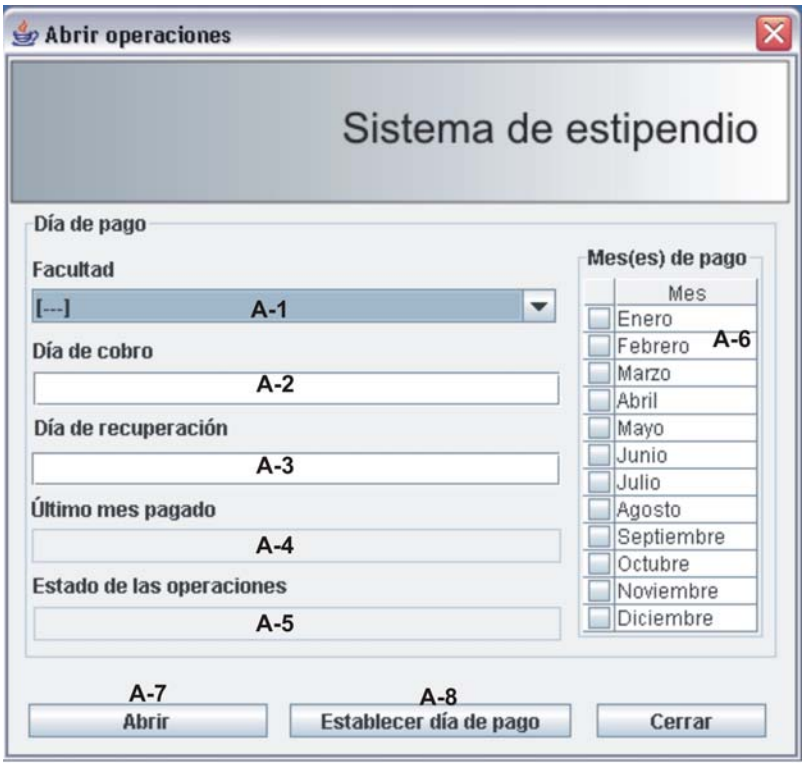


Figura 2.11. Ventana para cerrar el período de pago.

¿Qué hace el usuario?	¿Qué hace el sistema?
1. Seleccionar la facultad en el control A-1.	
2. Presionar el botón A-7 para abrir las operaciones.	3. El sistema procede a cerrar el período de pago abierto. Cambia el estado de las operaciones a abierto. Limpia las solicitudes de los estudiantes de esa facultad. Revisa si existe algún estudiante que ha dejado de cobrar tres meses consecutivos, si es cierto los pone inactivo.
4. Se pone en los controles A-2 y A-3 las fechas de cobro del próximo mes. Seleccionar en el control A-6 los meses que se van a pagar en el próximo período de pago.	
5. Presionar el botón A-8.	6. Se actualizan los campos de la tabla *CashDay* con los nuevos valores.

Tabla 2.5. Flujo normal de los eventos de la ventana de cerrar pagos.

¿Qué hace el usuario?	¿Qué hace el sistema?
El usuario presiona el botón A-8 y ha establecido que el día de recuperación es anterior al día de cobro.	No se puede proceder lanzando una excepción diciendo que hay problemas con las fechas.
El usuario presiona el botón A-8 y los meses a pagar no son consecutivos entre sí y con el último mes pagado.	No se puede proceder y se lanza una excepción pidiendo que se corrija este problema.
El usuario presiona el botón A-7 y existen estudiantes por declarar su estado de cobro.	El sistema no puede abrir las operaciones y lanza una excepción recomendando que defina el estado de cobro de los estudiantes del último día de pago.

Tabla 2.6. Flujo alternativo de los eventos de la ventana de cerrar pagos

2.2 Implementación del sistema

Todas las clases del módulo están incluidas en el espacio de nombres *cu.mes.StipendSystem*, dentro del cual se encuentran tres paquetes:

VO***:*** contiene las clases de presentación.
DAO***:*** contienen las clases persistentes.
*Services****:*** contiene las clases de los servicios.

Figura 2.12. Espacios de nombre que contienen las clases del módulo de estipendio.

El diseño de las clases del sistema se realizó con la idea de aislar lo mejor posible nuestro módulo del resto de los módulos del sistema general, agilizar las búsquedas en las tablas de la base de datos y orientar la organización de los datos a conceptos de pago para flexibilizar el sistema ante posibles cambios en el modo de pago.

2.2.1 Clases persistentes

En el diseño de las clases persistentes hay dos clases fundamentales, las clases que son historia y las clases de uso temporal.

Las clases de historia son los datos que interesa guardar por tiempo indeterminado, la clase más importante de todas ellas es la clase de préstamos, por la cual al finalizar los estudios un estudiante con préstamos se podrá determinar de cuanto es su adeudo. Las temporales son clases con información que no trascenderá más de un curso o simplemente son codificadores como el período de tiempo de un préstamo o los conceptos de pagos normados.

En la figura 2.13 se muestra el diagrama de las clases persistentes.

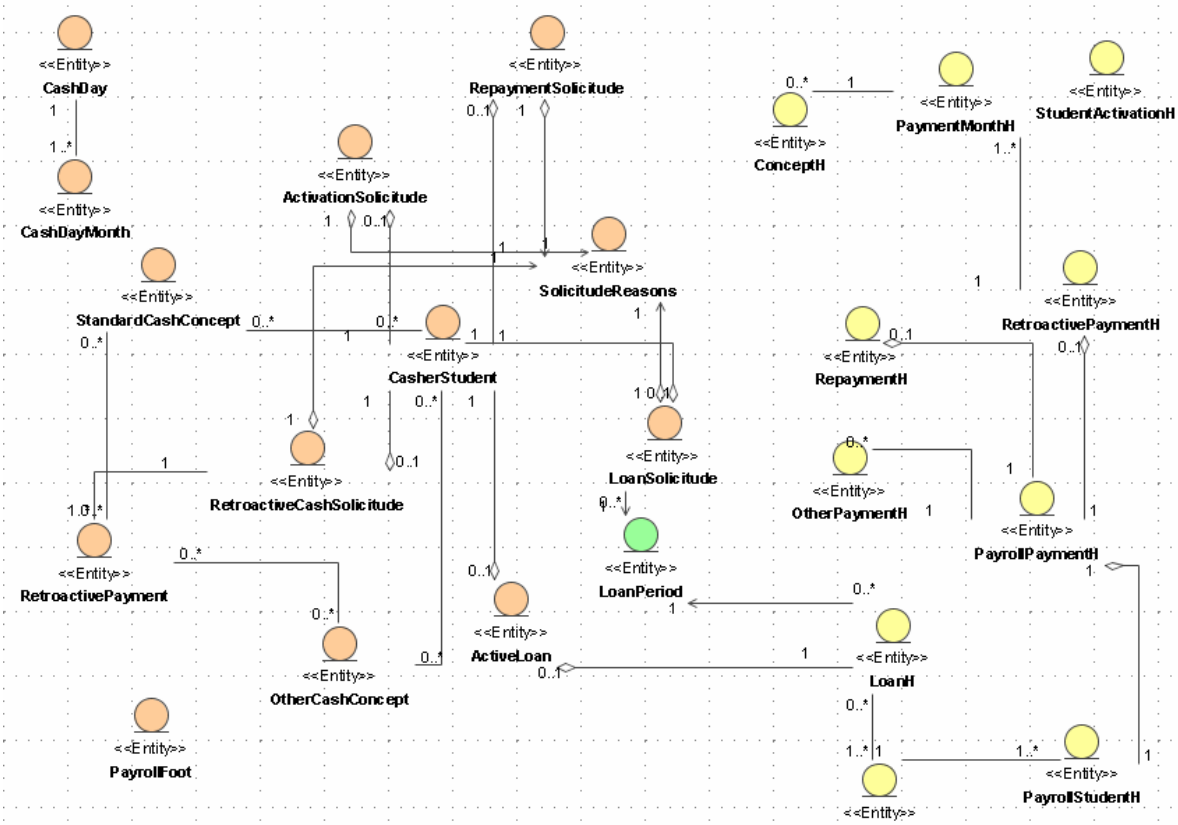


Figura 2.13. Diagrama de clases persistentes del sistema de estipendio. Clases temporales a la izquierda, clases de historia a la derecha.

Toda la información almacenada por el sistema de estipendio tiene como eje central los estudiantes que pueden cobrar estipendio; entiéndase los estudiantes que puedan recibir cualquier pago manejado en la resolución 24/07. La clase de estos estudiantes es *CasherStudents*, la tabla 2.7 describe sus atributos.

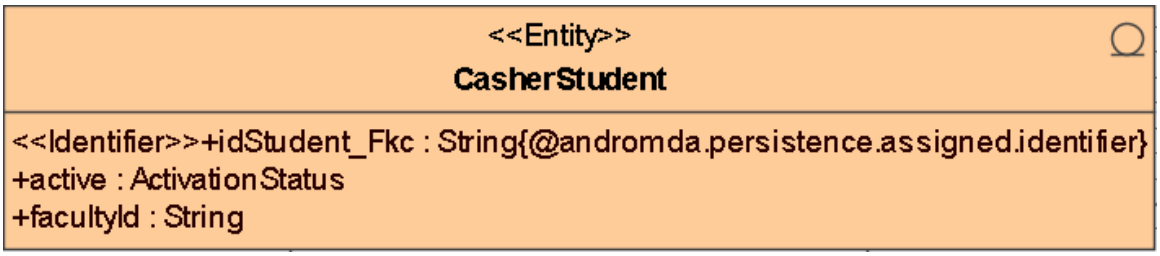


Figura 2.14. Especificación de la clase que describe los estudiantes del sistema de estipendio.

Propiedad	Descripción
idStudent	Llave de la clase. Es el identificador de los estudiantes en la tabla *Student*, este atributo identifica unívocamente a cada estudiante en todo el sistema en general.
active	Determina si el estudiante esta activo en el sistema de estipendio.
facultyId	Identificador de la facultad del estudiante. Campo adicionado para agilizar las búsquedas de los estudiantes.

Tabla 2.7. Descripción de los atributos de la clase de los estudiantes del sistema de estipendio.

Los conceptos de pago son los requisitos que cumple un estudiante, aportando cada concepto una determinada cantidad de dinero, conformando la suma de estos montos su estipendio estudiantil. Los conceptos de pago pueden ser conceptos estándares u otros conceptos. Los estándares, están basados en el tipo de curso del estudiante, el año académico y su pertenencia al movimiento de alumnos ayudantes de la universidad, los tipos de estos conceptos son los salarios base de cada estudiante, los salarios por orden 18 y las ayudantías universitarias. Los otros conceptos son los nuevos pagos que puedan surgir en cualquier momento, con un carácter más temporal.

Cada estudiante en el sistema de estipendio tiene asignado los conceptos de pago que le corresponden y cada concepto la cantidad de dinero que debe cobrar cada estudiante por cumplir con el mismo.

Figura 2.15. Clase para los conceptos de pago estándares.

Propiedad	Descripción
idName	Llave de la clase. Identificador del concepto
concept	Nombre formal del concepto.
amount	Cantidad de dinero en pesos que se debe pagar por el concepto
type	Tipo del concepto.

Tabla 2.8. Descripción de los atributos de la clase de conceptos de pago estándares.

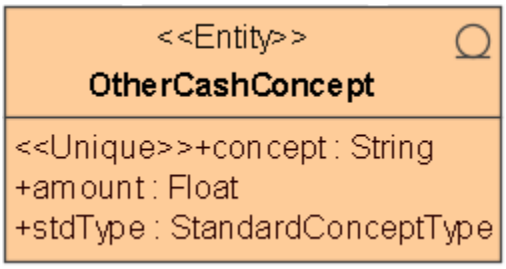


Figura 2.16. Clase para los otros conceptos de pago.

Propiedad	Descripción
concept	Llave de la clase. Nombre formal del concepto.
amount	Cantidad de dinero en pesos que se debe pagar por el concepto.
stdType	Se utiliza para representar que el nuevo concepto ingresado al sistema es de tipo ayudantía. Permitiendo que se realice el pago de solo una ayudantía.

Tabla 2.9. Descripción de los atributos de la clase de otros conceptos de pago.

Las solicitudes son el medio de comunicación del estudiante con economía, se utilizan para hacer peticiones de préstamos, reintegros, activación al sistema de estipendio y pagos retroactivos.

La solicitud de préstamo se utiliza para pedir un nuevo préstamo o cancelar un préstamo ya activo.

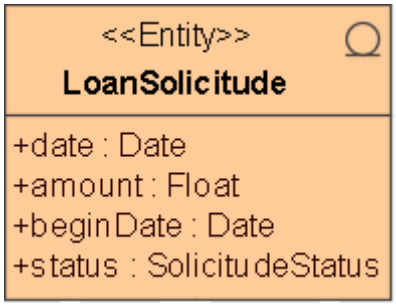


Figura 2.17. Clase que define la solicitud de préstamo.

Propiedad	Descripción
date	Fecha en que se creó la solicitud.
amount	Cantidad de dinero a pagar mensualmente por ese préstamo.
beginDate	Fecha en que se comienzan a hacer los pagos del préstamo.
status	Estado de la solicitud.

Tabla 2.10. Descripción de los atributos de la clase de solicitud de préstamo.

Solo es posible reintegrar el último mes pagado y hacer la solicitud aquellos estudiantes que están reportados por dejar de cobrar dicho mes.

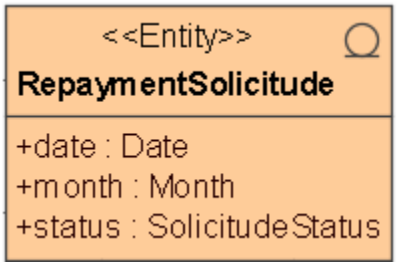


Figura 2.18. Clase que define la solicitud de reintegro.

Propiedad	Descripción
date	Fecha en que se creó la solicitud.
month	Mes a reintegrar.
status	Estado de la solicitud.

Tabla 2.11. Descripción de los atributos de la clase de solicitud de reintegro.

Las solicitudes de activación se realizan porque el estudiante ya ha resuelto los problemas que lo han llevado a la inactividad o los a justificado, por lo que es su derecho estar nuevamente activo en el sistema.

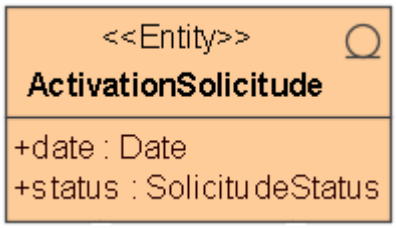


Figura 2.19. Clase que define la solicitud de activación.

Propiedad	Descripción
date	Fecha en que se creó la solicitud.
status	Estado de la solicitud.

Tabla 2.12. Descripción de los atributos de la clase de solicitud de activación.

Las solicitudes de pagos retroactivos son para corregir errores humanos al crear las condiciones para que el estudiante cobre su estipendio correctamente en el tiempo establecido. Se define por cada mes, con problemas en el pago, los conceptos que dejaron de cobrar los estudiantes en ese momento. También se puede declarar como pago retroactivo el préstamo. La clase *RetroactivePayment* tiene una relación de muchos con las tablas de conceptos, permitiendo así el pago retroactivo de los conceptos.

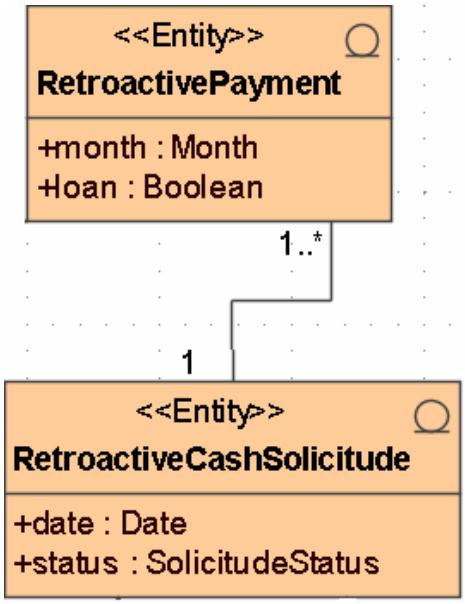


Figura 2.20. Clases que definen la solicitud de pago retroactivo.

Propiedad	Descripción
date	Fecha en que se creó la solicitud.
status	Estado de la solicitud.
month	Mes con problemas en el pago.
loan	Es cierto si se debe pagar el préstamo como pago retroactivo.

Tabla 2.13. Descripción de los atributos de la clase de pagos retroactivos.

Todas las solicitudes anteriores deben ser justificadas, por parte del estudiante; y si el economista cuando las analiza decide justificar la decisión que se tomó con esa solicitud puede hacerlo también; por tanto se tiene una clase para representar las justificaciones dadas por parte de su fomentador y analista.

Figura 2.21. Clase que representa las justificaciones de una solicitud.

Propiedad	Descripción
studentReason	Justificación del estudiante.
economicReason	Justificación de la decisión del economista.

Tabla 2.14. Descripción de los atributos de la clase de las justificaciones de una solicitud.

Los días de pago son las fechas declaradas por el economista en las que se puede cobrar el estipendio. Por cada facultad se crea una entidad de *CashDay*, actualizándose sus campos cada vez que sea requerido.

En un mismo día de pago se pueden pagar más de un mes, ya sea porque llegó el fin del curso o porque ocurrió algún retraso. Así además de definir los días de cobro se debe definir los meses a pagar. Los meses de pago deben ser consecutivos entre si y consecutivos al último mes pagado.

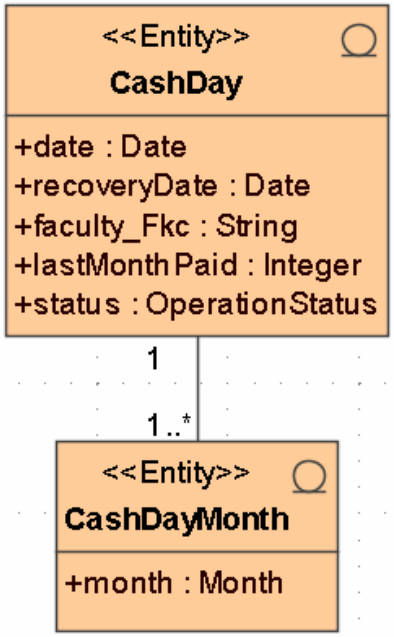


Figura 2.22. Clases que definen el estado del cobro de una facultad.

Propiedad	Descripción
date	Primera fecha para que los estudiantes de la facultad realicen el cobro.

recoveryDate	Segunda fecha para que los estudiantes de la facultad realicen el cobro.
faculty_Fkc	Identificador de la facultad. Este identificador es único para cada facultad en todo el sistema principal.
lastMonthPaid	Último mes pagado. Se introduce con motivos de agilizar los procesos de búsqueda.
status	Define el estado de las operaciones para la facultad.
month	Mes que se va pagar en ese día de pago

Tabla 2.15. Descripción de los atributos de la clase de estado de cobro de una facultad.

Como veremos más adelante la seguridad del cliente del economista es controlada por el propio módulo de estipendio, para este fin se creó una tabla para almacenar los usuarios del cliente de economía.

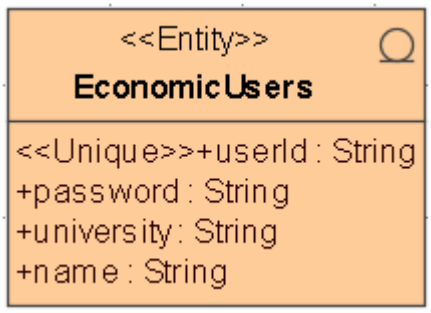


Figura 2.23. Clase que define a un usuario del cliente de economía.

Propiedad	Descripción
userId	Identificador del usuario.
password	Contraseña del usuario.
university	Universidad del usuario.
name	Nombre del usuario.

Tabla 2.16. Descripción de los atributos de la clase que representa los usuarios del cliente de economía.

Los préstamos una vez aprobados se almacenan en una tabla historial y tienen una referencia en una tabla que maneja los préstamos activos, esta entrada permanece hasta que se cancele el préstamo o se termine. La tabla de los préstamos se describe a continuación.

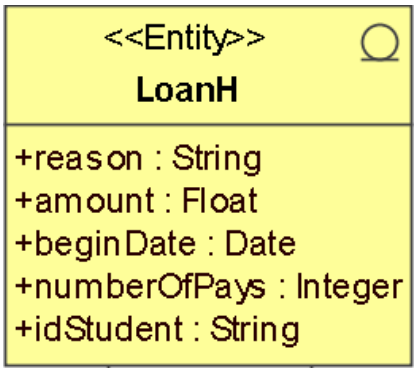


Figura 2.24. Clase que representa el préstamo de un estudiante.

Propiedad	Descripción
reason	Justificación del préstamo.
amount	Cantidad de dinero a pagar por el préstamo
beginDate	Fecha de inicio del préstamo.
numberOfPays	Cantidad de pagos cobrados por el estudiante. Este campo se introduce para agilizar las búsquedas.
idStudent	Identificador del estudiante. Este campo es el identificador del estudiante en la tabla *Student*. Identifica a un estudiante unívocamente en el sistema principal.

Tabla 2.17. Descripción de los atributos de la clase que representa al préstamo.

Las nóminas y sus pagos se almacenan en tablas historiales de igual modo que el préstamo. Cada nómina tiene un conjunto de estudiantes y por cada estudiante se define un pago que está compuesto por el dinero del préstamo, del reintegro, del pago retroactivo, y otros pagos. Las clases que se encargan de definir este conjunto que representa una nómina se definen a continuación.

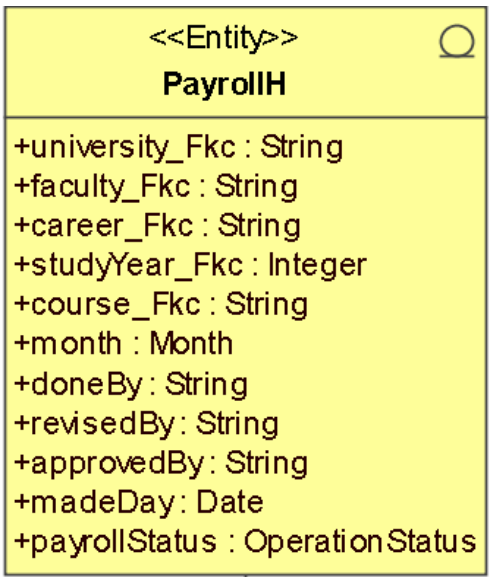


Figura 2.25. Clase que representa una nómina.

Propiedad	Descripción
university_Fkc	Identificador de la universidad. Este atributo identifica a una universidad unívocamente en todo el sistema principal.
faculty_Fkc	Identificador de la facultad. Este atributo identifica a una facultad unívocamente en todo el sistema principal.
career_Fkc	Identificador de la carrera. Este atributo identifica una carrera en todo el sistema principal.
studyYear	Año de estudio de la carrera.
course_Fkc	Identificador del curso. Este atributo identifica unívocamente un curso en todo el sistema principal.
month	Mes que paga la nómina.
doneBy	Persona que elaboró la nómina.
revisedBy	Persona que revisó la nómina.
approvedBy	Persona que aprobó la nómina.
madeDay	Fecha de confección de la nómina.

payrollStatus	Determina si la nómina está abierta por tener estudiantes a los que no se les ha definido su estado de cobro, o cerrada en otro caso.

Tabla 2.18. Descripción de los atributos de la clase que representa a la nómina.

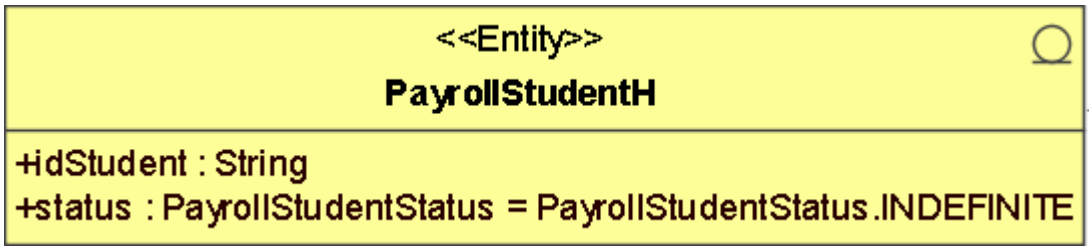


Figura 2.26. Clase que representa los estudiantes de la nómina.

Propiedad	Descripción
idStudent	Identificador del estudiante. Este campo es el identificador del estudiante en la tabla *Student*. Identifica a un estudiante unívocamente en el sistema principal.
status	Define el estado de cobro del estudiante.

Tabla 2.19. Descripción de los atributos de la clase que representa al los estudiantes de la nómina.

A continuación se describen las clases que definen los pagos del estudiante.

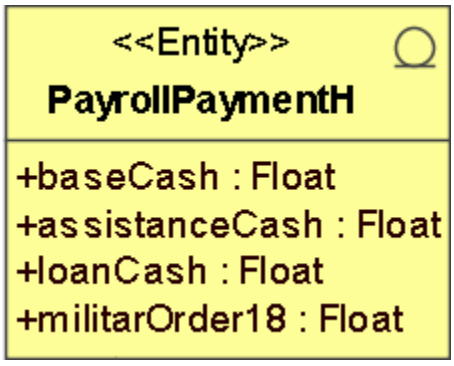


Figura 2.27. Clase que representa los pagos estándares y préstamo del estudiante.

Propiedad	Descripción
baseCash	Salario base de los estudiantes que no son orden 18.
assistanceCash	Subvención por ayudantía.
loanCash	Préstamo.
militarOrder18	Salario base de los estudiantes que son orden 18.

Tabla 2.20. Descripción de los atributos de la clase que representa los pagos estándares y préstamo de un estudiante.

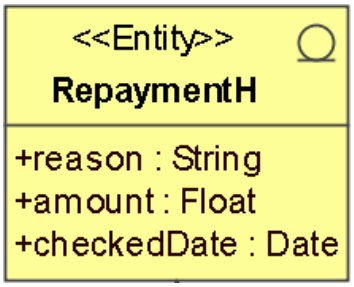


Figura 2.28. Clase que describe el pago por reintegro.

Propiedad	Descripción
reason	Justificación del reintegro.
amount	Cantidad de dinero que dejó de cobrar el estudiante.
checkedDate	Fecha de la solicitud.

Tabla 2.21. Descripción de los atributos de la clase que representa el pago de reintegro.

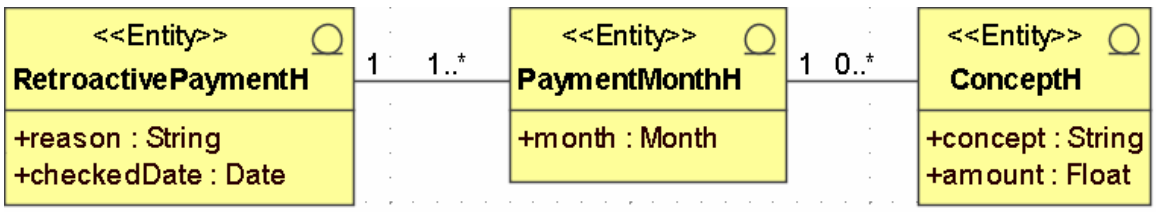


Figura 2.29. Clases que representan el pago retroactivo.

Propiedad	Descripción
reason	Justificación del pago retroactivo.
checkedDate	Fecha de la solicitud.
month	Mes del pago retroactivo.
concept	Nombre del concepto.
amount	Cantidad de dinero que se paga por el concepto.

Tabla 2.22. Descripción de los atributos del pago retroactivo.

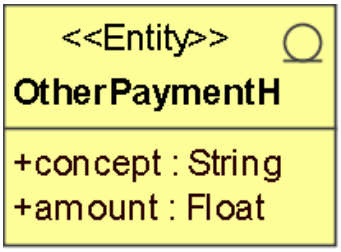


Figura 2.30. Clase que representa el pago por otros conceptos.

Propiedad	Descripción
concept	Nombre del concepto.
amount	Cantidad de dinero que se paga por el concepto.

Tabla 2.23. Descripción de los atributos de la clase que representa al pago de otros conceptos.

Cada vez que un estudiante cambia su estado de cobro, de activo a inactivo se almacena este cambio en una tabla historial. Esta tabla es solo informativa, no tiene peso en el proceso del negocio.

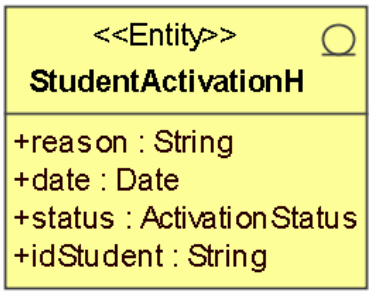


Figura 2.31. Clase que representa la historia de los cambios de activación de los estudiantes.

Propiedad	Descripción
reason	Justificación del cambio de activación.
date	Momento en que se hizo el cambio de estado.
status	Estado al que cambió.
idStudent	Identificador del estudiante. Este campo es el identificador del estudiante en la tabla *Student.* Identifica a un estudiante unívocamente en el sistema principal.

Tabla 2.24. Descripción de los atributos de la tabla de historia de los cambios de activación de los estudiantes.

2.2.2 Clases de servicios

Son seis las clases de servicios del sistema de estipendio que se encargan de modelar la lógica del negocio.

- *SolicitudeSess**:*** maneja las solicitudes. Contiene entre otros métodos para crear y buscar, ejemplo:
  public void createRepaymentSolicitude(cu.mes.StipendSystem.VO.RepaymentSolicitudeVO info): crea, actualiza y elimina la solicitud de reintegro en dependencia de la información almacenada en info.

  public java.util.Collection getRepaymentSolicitudes(cu.mes.StipendSystem.VO.Filter filter): retorna las solicitudes de reintegro que cumplan con el filtro especificado.

- *StudentSess**:*** brinda métodos para el manejo de los estudiantes. A modo de ejemplo se muestran la facilidad de buscar los estudiantes que no han cobrado y asignar conceptos de pago.

  public java.util.Collection getWhithoutCashStudents(cu.mes.StipendSystem.VO.Filter filter): busca los estudiantes que no cobraron según el filtro pasado como parámetro.

public void setAssignedOtherConcepts(java.util.Collection info): como parámetro se pasan los conceptos adicionales de un estudiantes y el método asigna los nuevos y elimina los otros.

- *PaymentConceptSess:* maneja los conceptos de pago. Modifica, crea y elimina los conceptos.

  public void setOtherConcepts(java.util.Collection concepts): modifica, crea y elimina los conceptos de pago adicionales, según la colección pasada como parámetro.

  public void setStandardConcepts(java.util.Collection concepts): modifica, crea y elimina los conceptos de pago estándares, según la colección pasada como parámetro.

  public void resetStandardConcepts(): este método se utilizó por el equipo de desarrollo para crear por primera vez los conceptos de pago estándares definidos en la resolución 24/07.

- *LoanSess:* maneja los préstamos. A modo de ejemplo mencionamos cerrar un préstamo activo y revisar si un préstamo ya terminó.

  public void closeLoan(cu.mes.StipendSystem.VO.LoanHVO activeLoan): cierra el préstamo activo pasado como parámetro. Ya un estudiante no cobrará más por ese préstamo.

  public java.lang.Boolean isLoanFinished(cu.mes.StipendSystem.VO.LoanHVO activeLoan, cu.mes.StipendSystem.DAOs.Month month): dado un préstamo activo y un mes determina si el préstamo ya terminó el tiempo por el que fue pedido.

- *PayrollSess:* brinda métodos para la confección de la nómina y el manejo del período de pago.

  public void calculatePay(cu.mes.StipendSystem.VO.Filter filter): hace el cálculo de las nóminas según el filtro pasado como parámetro.

public cu.mes.StipendSystem.VO.FoundReportVO foundSolicitudeReport(cu.mes.StipendSystem.VO.Filter filter): devuelve un objeto con los datos necesarios para hacer un reporte de los fondos de la facultad y el mes pasados en el filtro.

- *Micsellaneus:* brinda varios servicios sobre todo de recuperación de información. Como ejemplo mostramos la obtención de las facultades y del día de pago.

  public java.util.Collection getFacultiesByUserAndPassword(cu.mes.StipendSystem.VO.Filter filter): obtiene las facultades que atiende el usuario que entró al sistema. Este método solo lo utiliza el cliente de secretaría.

  public cu.mes.StipendSystem.VO.CashDayVO getCashDay(cu.mes.StipendSystem.VO.Filter faculty): devuelve el día de pago de la facultad especificada en el filtro.

2.2.3 Seguridad del sistema de estipendio

El cliente de secretaría es un *plug-in* del sistema general, así mismo lo es el cliente de seguridad del SIGENU, que le sirve al resto de los módulos para controlar este tema. Este módulo de seguridad se encarga de autenticar a los usuarios y brindar la información obtenida al resto de los componentes.

El cliente del economista es un cliente solitario, que no puede usar el módulo de seguridad del SIGENU, por lo tanto se tuvo que desarrollar herramientas para controlar este asunto. Se creó la tabla de usuarios descrita anteriormente y se utilizó el algoritmo de encriptación MD5 para codificar la contraseña. No existen roles, pues solo es un sistema para economistas.

2.2.4 Capas en la implementación del módulo

El sistema del estipendio es un sistema cliente-servidor. Los clientes como ya habíamos visto son de escritorio y en el servidor se despliegan los servicios y las clases persistentes.

El proyecto esta basado en la arquitectura de capas permitiendo la escalabilidad y mantenimiento del módulo.

Figura 2.32. Capas del módulo de estipendio.

Capa clases swing**:** define el diseño de las ventanas, la manera en que se van a mostrar los datos y las facilidades de interacción que le brinda la interfaz al usuario.

Capa clases controladores**:** esta capa solo accede a los servicios que necesitará la capa inferior, de modo que el diseño de las interfaces de usuario no acceda directamente a los métodos del negocio. Las clases de esta capa se implementaron siguiendo el patrón *singleton*. Esta capa cuenta con una clase muy importante, que es la encargada de comunicarse con el servidor y entregarle a los controladores los objetos que necesiten, concentrando en un punto el acceso remoto.

Capa interfaces remotas de los servicios**:** contiene las interfaces necesarias para publicar los servicios. Utiliza de la tecnología *Enterprise Java Bean***,** los *stateless session bean.*

Capa clases de implementación de los servicios**:** esta capa es la que contiene las clases que implementan la lógica del negocio. La anterior capa tiene clases que implementan los servicios que

publican como parte de la tecnología utilizada, pero en realidad lo que hacen estas clases es llamar a los métodos contenidos en esta capa.

Capa clases persistentes**:** esta capa contiene las clases que manejan las tablas de estipendio. Esta capa utiliza las librerías que brinda *Hibernate*, facilitando considerablemente el trabajo con la base de datos.

2.2.5 Componentes del módulo

El módulo de estipendio está compuesto por tres componentes fundamentales

*secretary-client-1.0.jar**:** plug-in* que es el cliente de la secretaria, que se le adiciona el proyecto *Skeleton***.**

*economic-client-1.0.jar**:*** aplicación cliente del economista.

*stipend-sys-app-2.0.ear**:*** paquete de las reglas del negocio que se despliega en el servidor. Se creó una librería que permite la comunicación de este componente con el empaquetado de las clases que manejan las tablas del resto del sistema, es decir con la distribución *aminotaur-app-1.0.0.ear*, por razones de tecnología, escalabilidad y mantenimiento. Esta librería accede a los *Entity bean* que implementa la aplicación *aminotaur-app-1.0.0.ear* para acceder a las tablas y *stipend-sys-app-2.0.ear* utiliza los *DAOs* que brinda la herramienta *Spring* para mapearse a las tablas de estipendio**.**

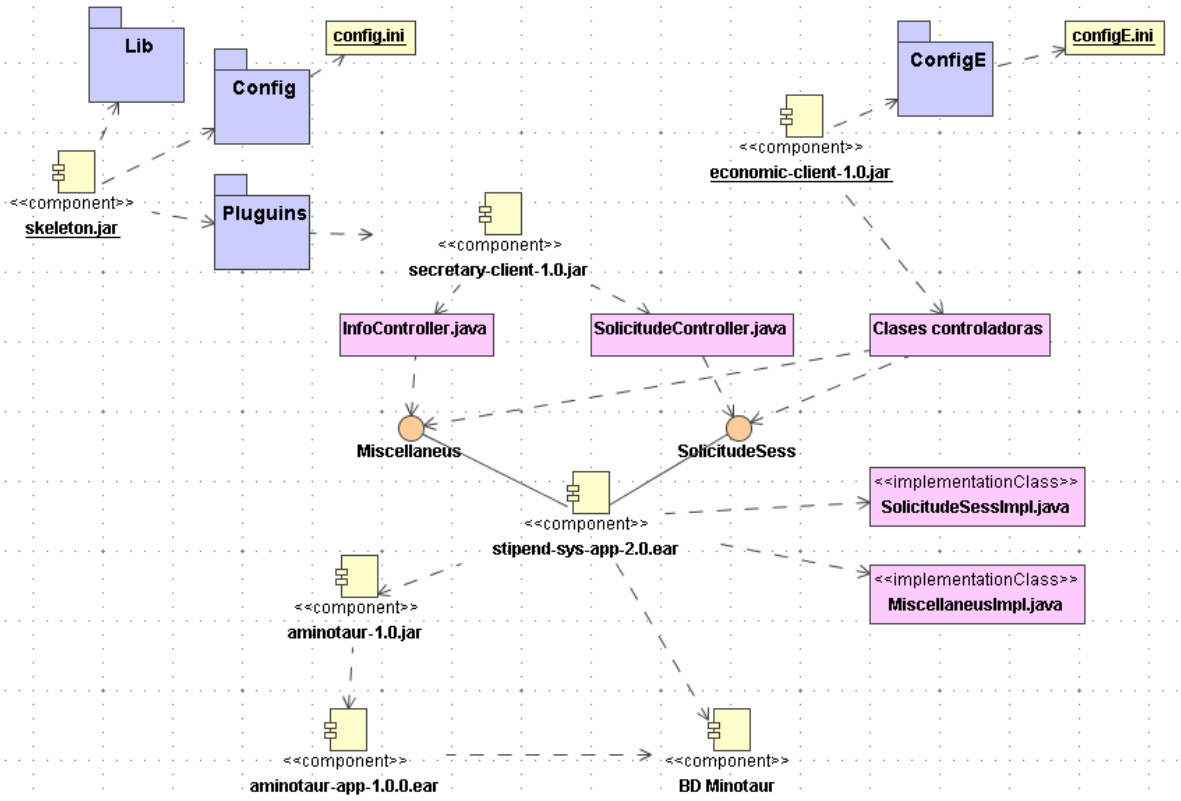


Figura 2.33. Interacción de los componentes del sistema de estipendio. Diagrama de componentes.

El sistema general es una aplicación cliente servidor. El siguiente diagrama ilustra el despliegue físico de los componentes. El servidor de aplicaciones y el de base de datos por lo general están juntos es una sola estación, pero no tiene porque ser así obligatoriamente.

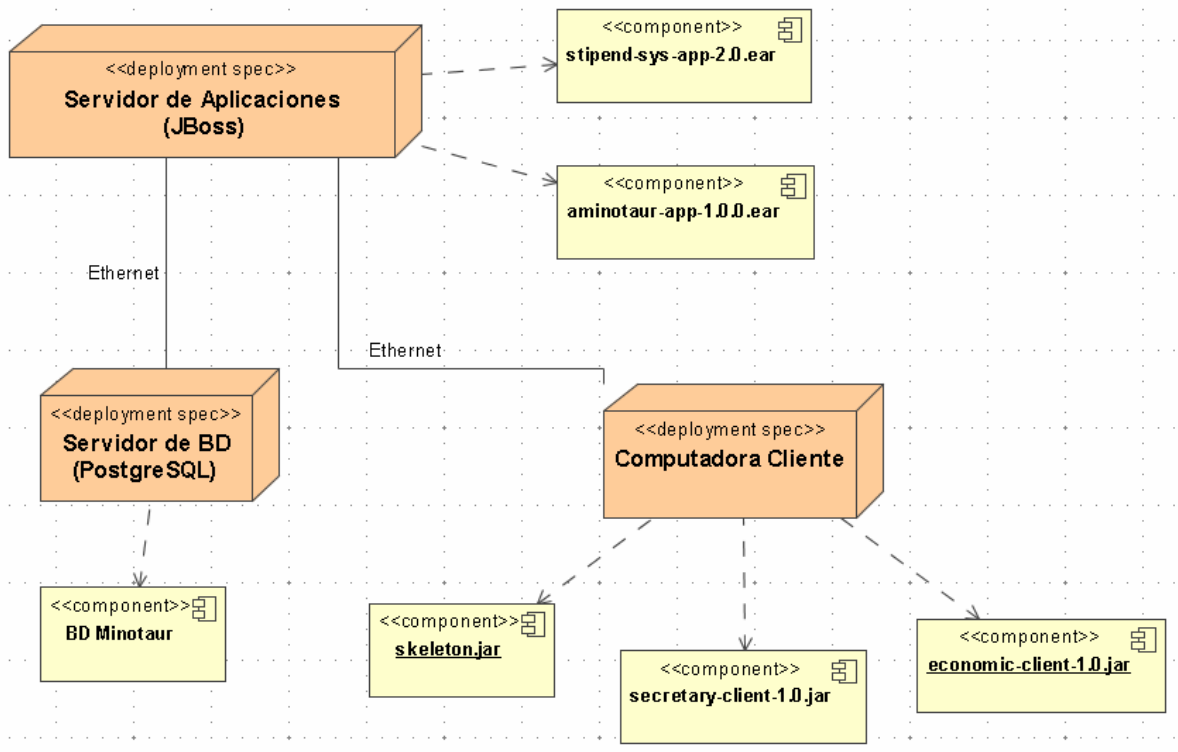


Figura 2.34. Estructura para la distribución de los componentes. Diagrama de despliegue.

2.3 Resumen 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 los procesos del negocio. Se describieron los atributos de las clases persistentes.

Se mostró la estructura del proyecto desde tres perspectivas diferentes.

- Usuario.
- Programador.
- Responsable de la distribución del módulo.

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

Capítulo III. Manual de usuario

El módulo de Estipendio Estudiantil, es un componente fundamental del Sistema de Estudiante del SIGENU. 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, entre otras, son los principales elementos que se automatizan en el sistema.

3.1 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.

3.2 Opciones principales del módulo de la secretaria

3.2.1 Asignar conceptos de pago adicionales

Esta opción es utilizada cuando la secretaria desea relacionar a los estudiantes con determinados conceptos de pagos adicionales.

Seleccionar en el menú **Estipendio** la opción **Asignar conceptos de pago**. (Figura 3.1)

Figura 3.1: Interfaz de la aplicación - opción del menú Estipendio – Asignar conceptos de pago.

Se visualiza la siguiente ventana que permite asignarles a los estudiantes los conceptos de pagos adicionales (Figura 3.2).

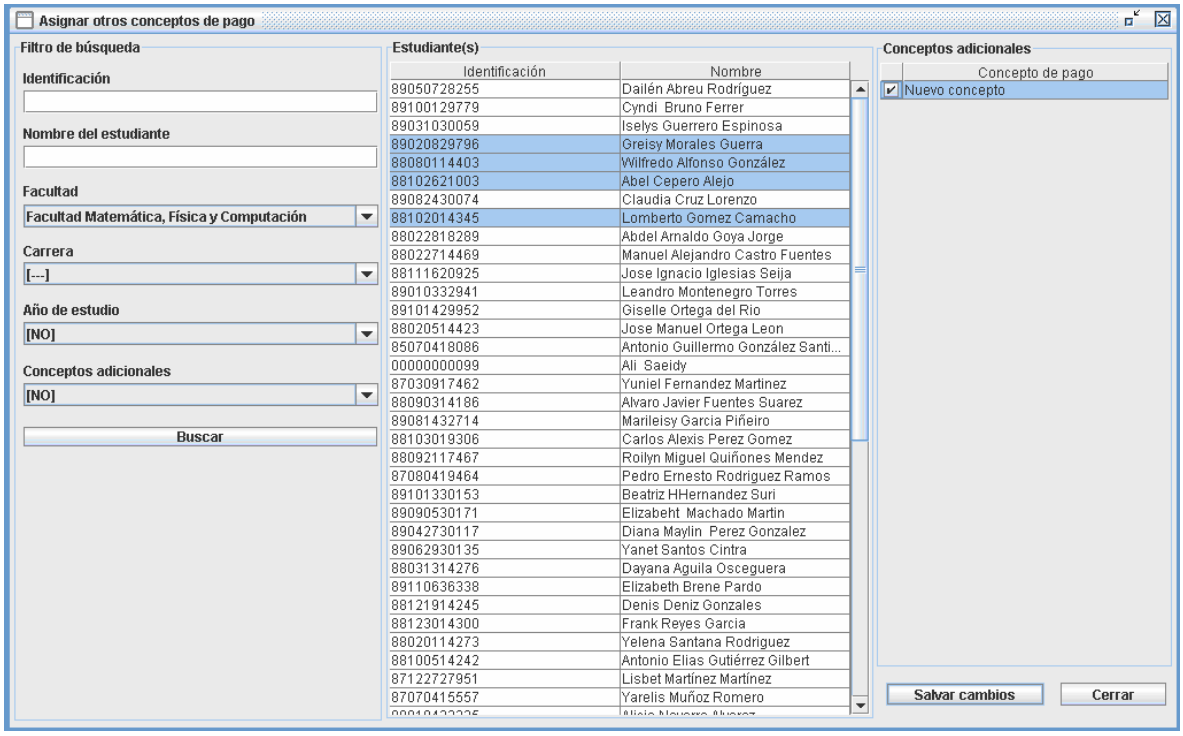


Figura 3.2: Interfaz de Asignación de conceptos de pagos adicionales

En el panel de la izquierda aparecen los filtros de búsqueda. En el panel de la derecha aparecen todos los conceptos adicionales activos .Después de seleccionarse el criterio por el cual se va a realizar la búsqueda aparecen en el panel del centro los estudiantes buscados. Se selecciona un estudiante o varios estudiantes para asignarles los conceptos adicionales. Se salvarán los cambios presionando el botón 

.

3.2.2 Información del día de cobro y operaciones

Esta opción es utilizada cuando la secretaria desea ver la información del día de cobro y el estado de las operaciones.

Seleccionar en el menú **Estipendio** la opción **Información del día de cobro y operaciones** (Figura 3.3).

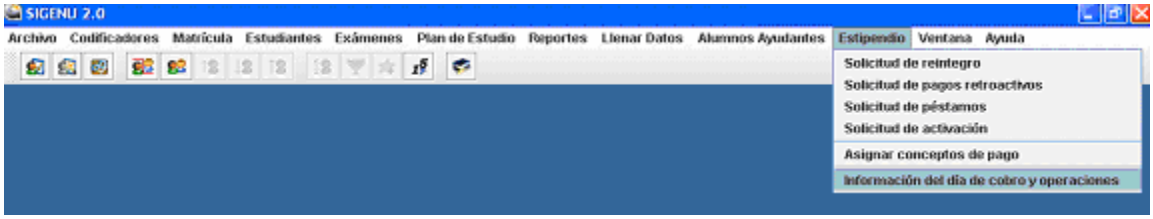


Figura 3.3: Interfaz de la aplicación - opción del menú Estipendio –Información del día de cobro y operaciones.

Se visualiza la siguiente ventana que muestra la información del día de pago y operaciones. (Figura 3.4)

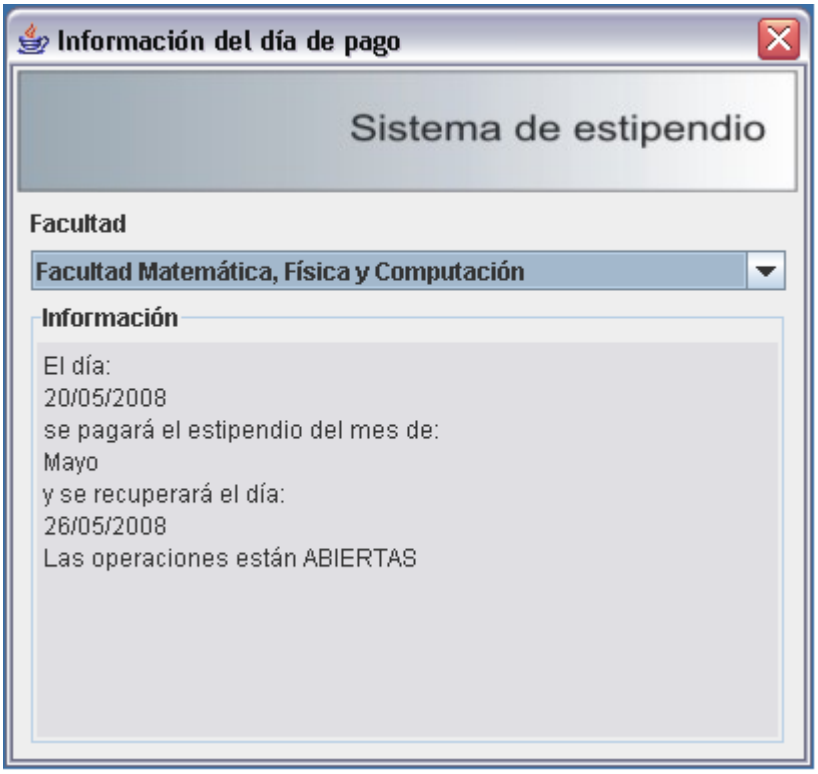


Figura 3.4: Interfaz de Información del día de cobro y operaciones.

3.2.3 Procesar solicitudes de reintegro

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

Seleccionar en el menú **Estipendio** la opción **Solicitud de reintegro** (Figura 3.5)

Figura 3.5: Interfaz de la aplicación - opción del menú Estipendio – Solicitud de reintegro.

Se visualiza la siguiente ventana que permite procesar las solicitudes de reintegros efectuadas por los estudiantes. (Figura 3.6)

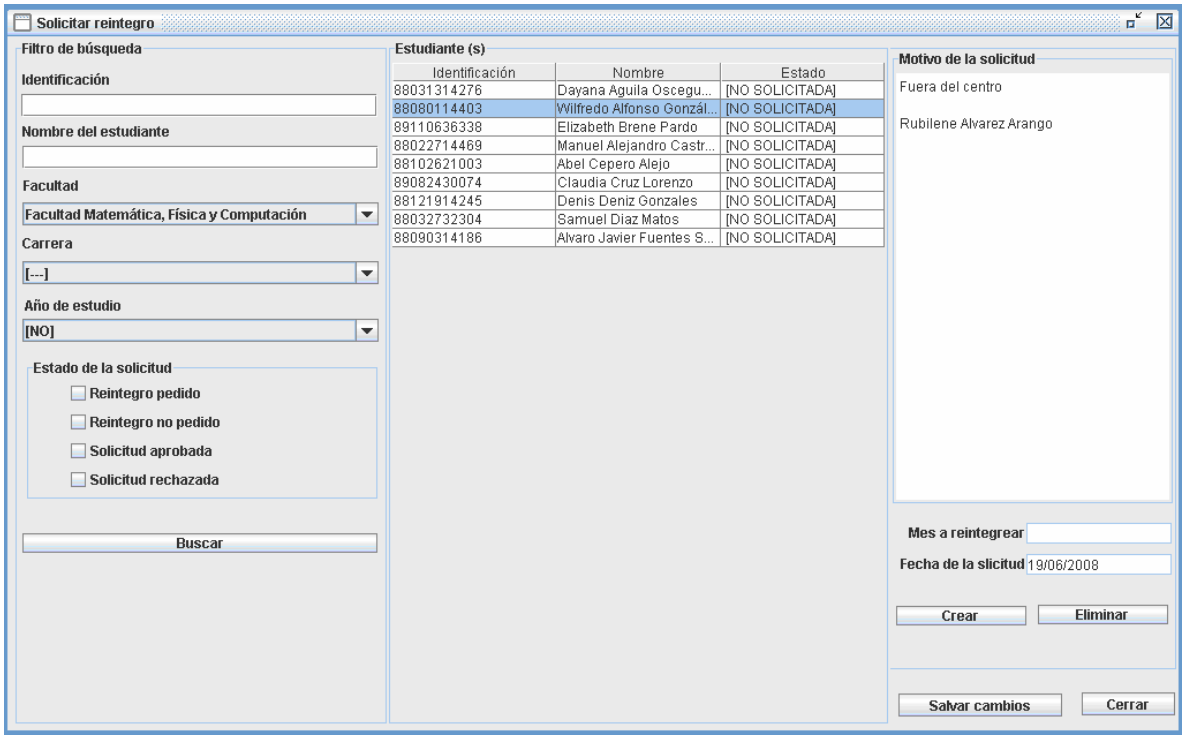


Figura 3.6: Interfaz de Pagos de Reintegros.

En el panel de la izquierda aparecen los filtros de búsqueda. Después de seleccionarse el criterio por el cual se va a realizar la búsqueda aparece en el panel del centro los estudiantes buscados. Se selecciona el estudiante o los estudiantes a los cuales se les quieren procesar las solicitudes de reintegros y en el panel derecho se ingresan el motivo por el cual no cobro, el mes a reintegrar y la fecha. Estas solicitudes se pueden crear o eliminar presionando los botones Crear y Eliminar. Se salvarán los cambios presionando el botón Salvar cambios.

3.2.4 Procesar solicitudes de pagos retroactivos

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

Seleccionar en el menú **Estipendio** la opción **Solicitud de pagos retroactivos** (Figura 3.7).

Figura 3.7: Interfaz de la aplicación - opción del menú Estipendio – Solicitud de pagos retroactivos.

Se visualiza la siguiente ventana que permite procesar las solicitudes de pagos retroactivos efectuadas por los estudiantes. (Figura 3.8).

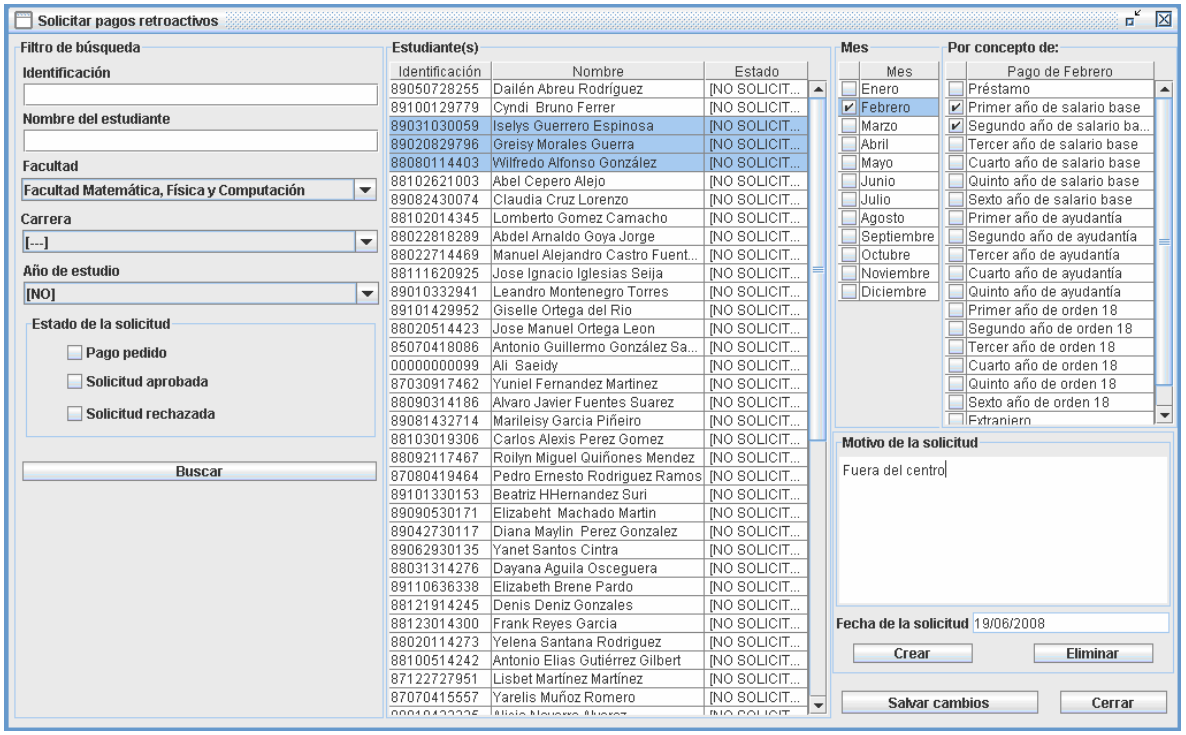


Figura 3.8: Interfaz de Pagos Retroactivos.

En el panel de la izquierda aparecen los filtros de búsqueda. En el panel de la derecha aparecen los meses y los conceptos de pagos activos. Después de seleccionarse el criterio por el cual se va a realizar la búsqueda aparecen en el panel del centro los estudiantes buscados. Se selecciona el estudiante o los estudiantes a los cuales se les quieren procesar la solicitudes de pagos retroactivos, se selecciona primero el mes del pago retroactivo y después los conceptos de ese mes y se ingresa el motivo por el cual no cobro. Estas solicitudes se pueden crear o eliminar presionando los botones Crear y Eliminar. Se salvarán los cambios presionando el botón Salvar cambios .

3.2.5 Procesar solicitudes de préstamos

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

Seleccionar en el menú **Estipendio** la opción **Solicitud de préstamos** (Figura 3.9).

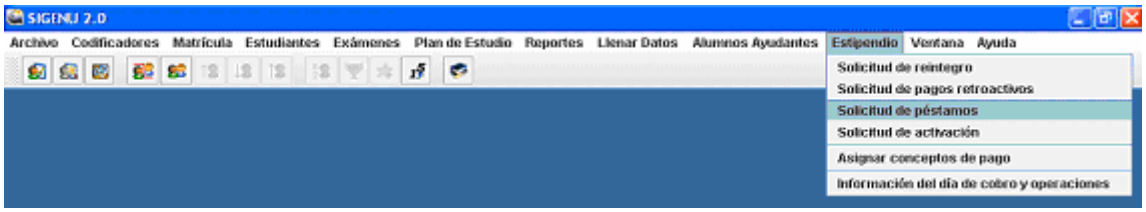


Figura 3.9: Interfaz de la aplicación - opción del menú Estipendio – Solicitud de préstamos

Se visualiza la siguiente ventana que permite procesar las solicitudes de préstamos efectuadas por los estudiantes. (Figura 3.10)

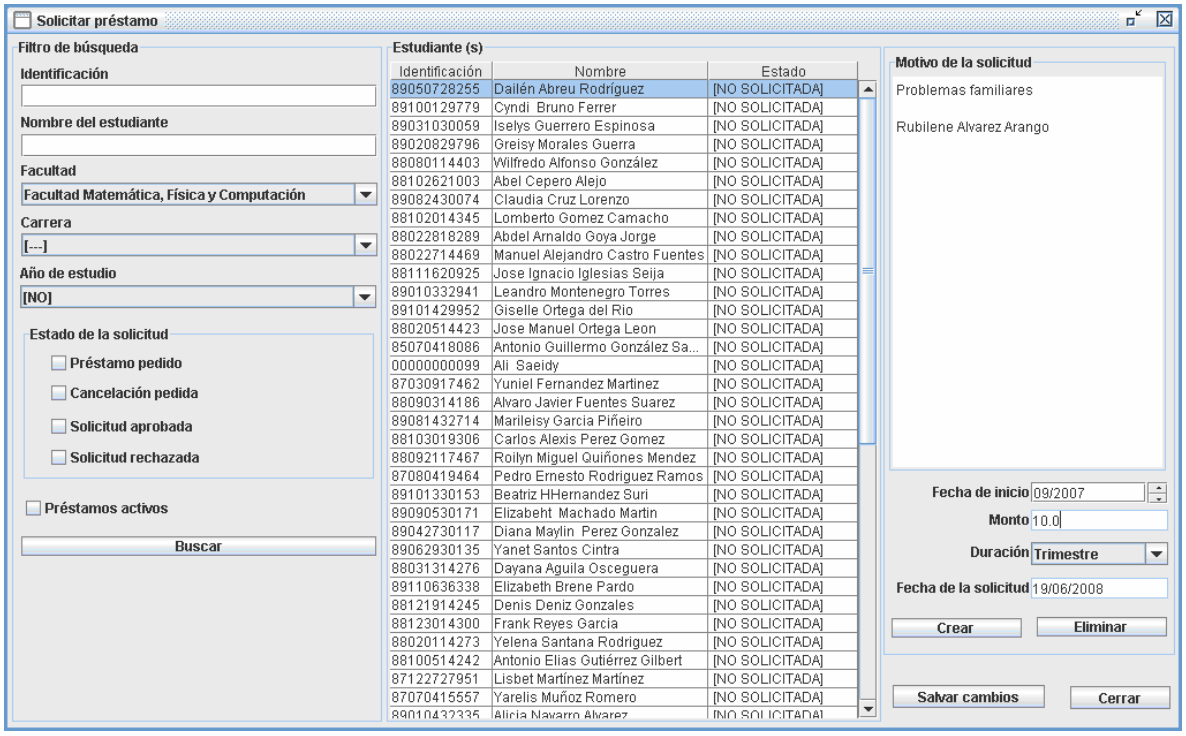


Figura 3.10: Interfaz de Solicitud de préstamos.

En el panel de la izquierda aparecen los filtros de búsqueda. Después de seleccionarse el criterio por el cual se va a realizar la búsqueda aparecen en el panel del centro los estudiantes buscados. Se selecciona el estudiante o los estudiantes a los cuales se les quieren procesar las solicitudes de préstamos y en el panel de la derecha se ingresa el motivo del préstamo, y los demás datos como la fecha de inicio, el monto, la duración y la fecha de las solicitudes. Estas solicitudes se pueden crear o eliminar presionando los botones Crear y Eliminar. Se salvarán los cambios presionando el botón Salvar cambios.

3.2.6 Procesar solicitudes de activación

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

Seleccionar en el menú **Estipendio** la opción **Solicitud de activación**. (Figura 3.11)

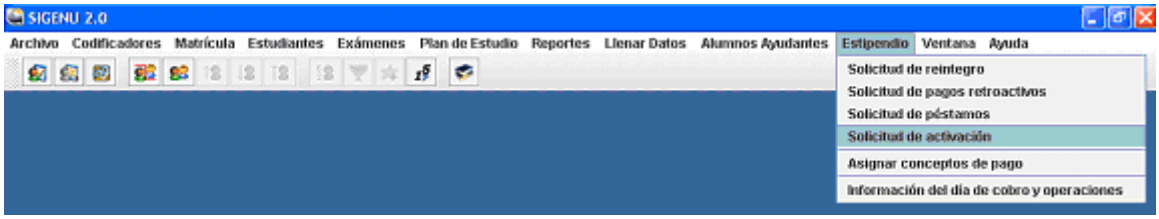


Figura 3.11: Interfaz de la aplicación - opción del menú Estipendio – Solicitud de activación

Se visualiza la siguiente ventana que permite procesar las solicitudes de activación efectuadas por los estudiantes (Figura 3.12).

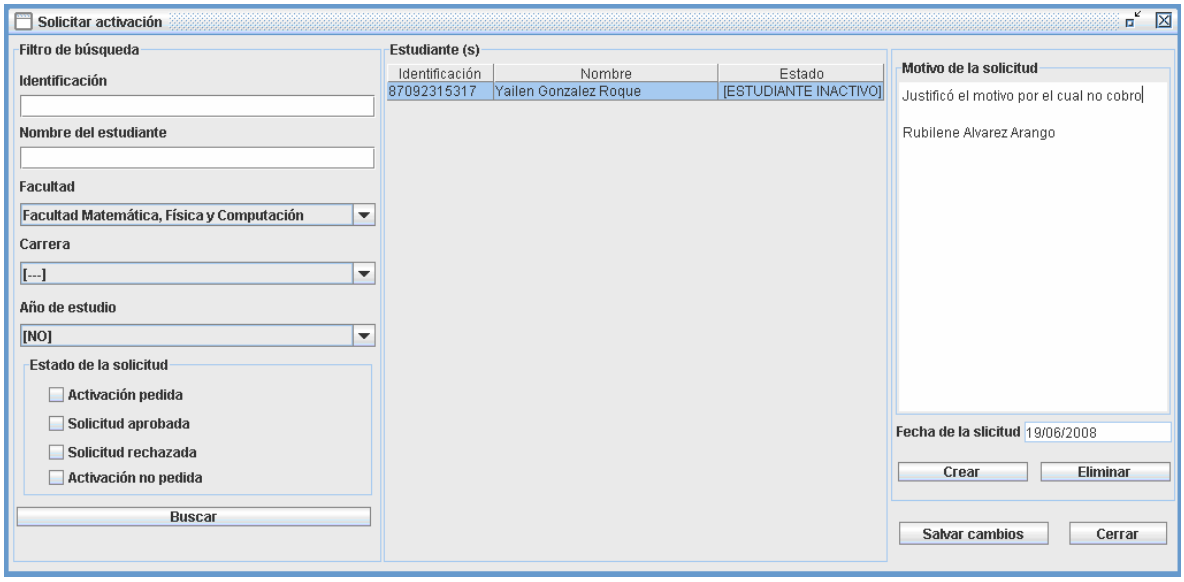


Figura 3.12: Interfaz de Solicitud de activación.

En el panel de la izquierda aparecen los filtros de búsqueda. Después de seleccionarse el criterio por el cual se va realizar la búsqueda aparecen en el panel del centro los estudiantes buscados. Se selecciona el estudiante o los estudiantes a los cuales se les quieren procesar las solicitudes de activación y en el panel derecho se ingresa el motivo de las solicitudes, y los demás datos como la fecha. Estas solicitudes se pueden crear o eliminar presionando los botones Crear y Eliminar. Se salvarán los cambios presionando el botón Salvar cambios .

3.3 Opciones principales del módulo del economista

3.3.1 Abrir las operaciones y establecer el día de pago

Esta opción es utilizada cuando el Economista desea abrir las operaciones del pago estudiantil y establecer el día de pago y de recuperación

Seleccionar en el menú **Período de pago** la opción **Reiniciar sistema\Cerrar** (Figura 3.13).

Figura 3.13: Interfaz de la aplicación - opción del menú Período de pago – Reiniciar sistema\Cerrar

Se visualiza la siguiente ventana que permite abrir las operaciones y establecer el día de pago y el de recuperación. (Figura 3. 14).

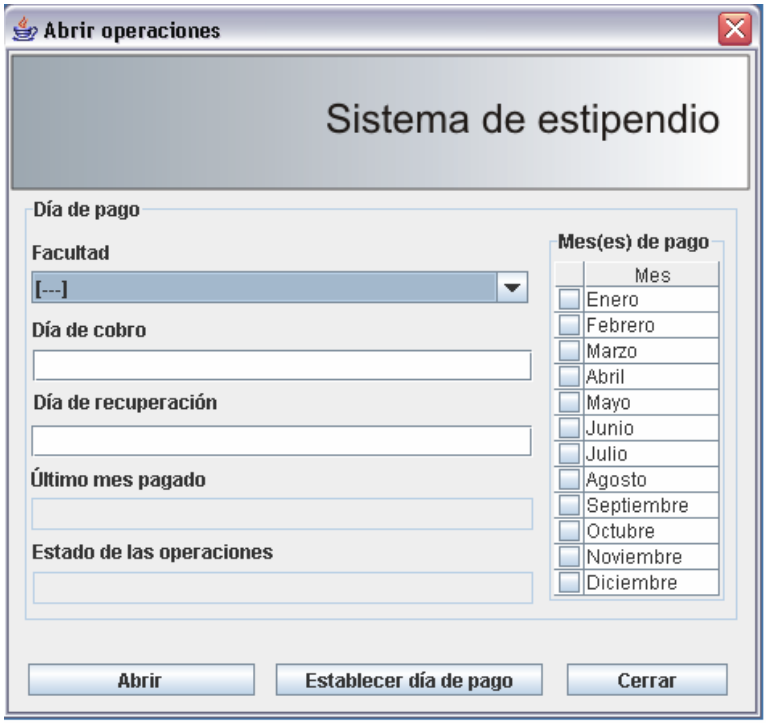


Figura 3.14: Interfaz de Abrir las operaciones.

En el panel de la izquierda en la parte superior aparece el filtro por facultad. Después de seleccionarse la facultad, debe establecerse el día de pago y de recuperación. Las operaciones se abrirán al presionarse el botón 

, si las operaciones están cerradas este botón estará

deshabilitado. Después de abrir las operaciones se deben establecerse los meses de pagos que aparecen en el panel derecho, estos deben comenzar siempre por Septiembre y si se va a pagar más de mes, deben de ser consecutivos. Después de seleccionar los meses de pagos correctamente se presiona el botón 

para guardar los meses seleccionados.

3.3.2 Cerrar las operaciones y crear las nóminas

Esta opción es utilizada cuando el Economista desea cerrar las operaciones del día de pago, es decir no se permitirá la entrada de más ningún dato al sistema, así como construir las nóminas de un día de pago, visualizar las mismas y visualizar la solicitud de fondo para un día de pago.

Seleccionar en el menú **Período de pago** la opción **Iniciar**. (Figura 3.15)

Figura 3.15: Interfaz de la aplicación - opción del menú Período de pago – Iniciar

Se visualiza la siguiente ventana que permite cerrar las operaciones y crear las nóminas. (Figura 3. 16).

Figura 3.16: Interfaz de Cerrar las operaciones y Calcular las nóminas

En el panel de la izquierda en la parte superior aparece el filtro por facultad, después de seleccionarse la facultad aparece la información acerca del día de cobro de la misma, si las operaciones están abiertas para la facultad seleccionada el botón Cerrar las operaciones estará habilitado de lo contrario estará deshabilitado. Para poder crear las nóminas primero se debe calcular el pago presionando el botón Calcular pago . En el panel de la derecha aparece el filtro para crear la nómina de fondo y las nóminas de pagos. Después de seleccionarse el filtro se presiona el botón Solicitud de fondos para crear la nómina de fondo y Mostrar nómina para crear las nóminas de pagos.

3.3.3 Declarar los estudiantes que no cobraron

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

Seleccionar en el menú **Período de pago** la opción **Declarar estudiantes sin cobrar** (Figura 3.17).

Figura 3.17: Interfaz de la aplicación - opción del menú Período de pago – Declarar estudiantes sin cobrar.

Se visualiza la siguiente ventana que permite notificar los estudiantes que dejaron de cobrar. (Figura 3.18).

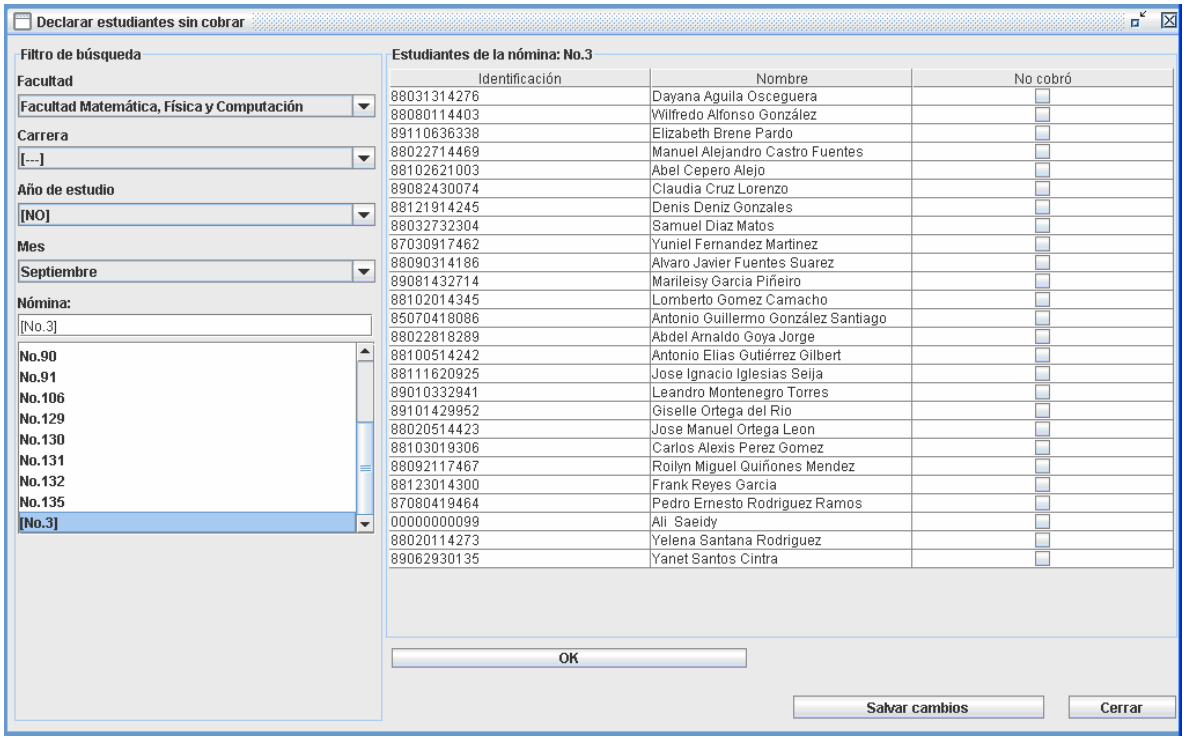


Figura 3.18: Interfaz de Declaración de estudiantes que no cobraron.

En el panel de la izquierda aparecen los filtros de búsqueda. Después de seleccionarse el criterio por el cual se va a realizar la búsqueda aparecen en el mismo panel izquierdo en la parte inferior las nóminas pagadas correspondientes a los datos del filtro, después de seleccionarse cualquiera de

ellas, aparecen en el panel de la derecha los estudiantes de la nómina seleccionada y si se quiere declar uno de ellos como que no ha cobrado solo tiene que marcarlo , después de terminado el análisis de la nómina seleccionada se debe presionar el botón 

para indicar que esa nómina fue analizada y se pondrá entre corchetes la nómina analizada. Se salvarán los cambios presionando el botón 

.

3.3.4 Actualizar el sistema

Esta opción es utilizada por el Economista para actualizar el sistema de estipendio.
Seleccionar en el menú **Período de pago** la opción **Sincronizar sistema de estipendio (**Figura 3.19).

Figura 3.19: Interfaz de la aplicación - opción del menú Período de pago – Sincronizar sistema de estipendio.

Se visualiza la siguiente ventana que permite actualizar el sistema (Figura 3.20).

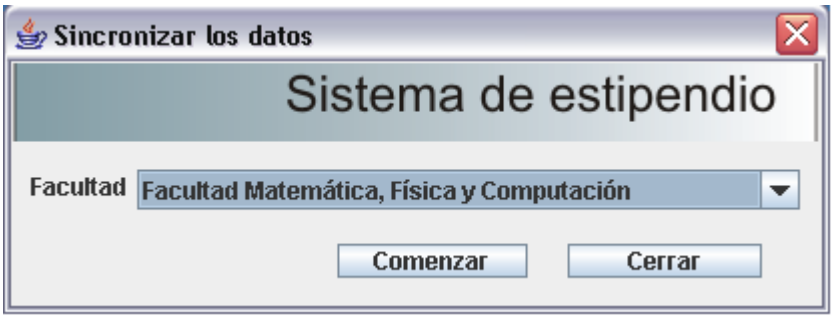


Figura 3.20: Interfaz de actualizar el sistema.

Después de seleccionarse la facultad que se quiere actualizar se presiona el botón 

3.3.5 Modificar los conceptos de pagos.

Esta opción es utilizada por el Economista para establecer los montos a cobrar por todos los conceptos de pago activos en el sistema o para adicionar un nuevo concepto de pago.

Seleccionar en el menú **Codificadores** la opción **Conceptos de pago** (Figura 3.21).

Figura 3.21: Interfaz de la aplicación - opción del menú Codificadores -Conceptos de pago.

Se visualiza la siguiente ventana que permite establecer los montos para los conceptos de pagos estándares o adicionar un nuevo concepto. (Figura 3. 22)

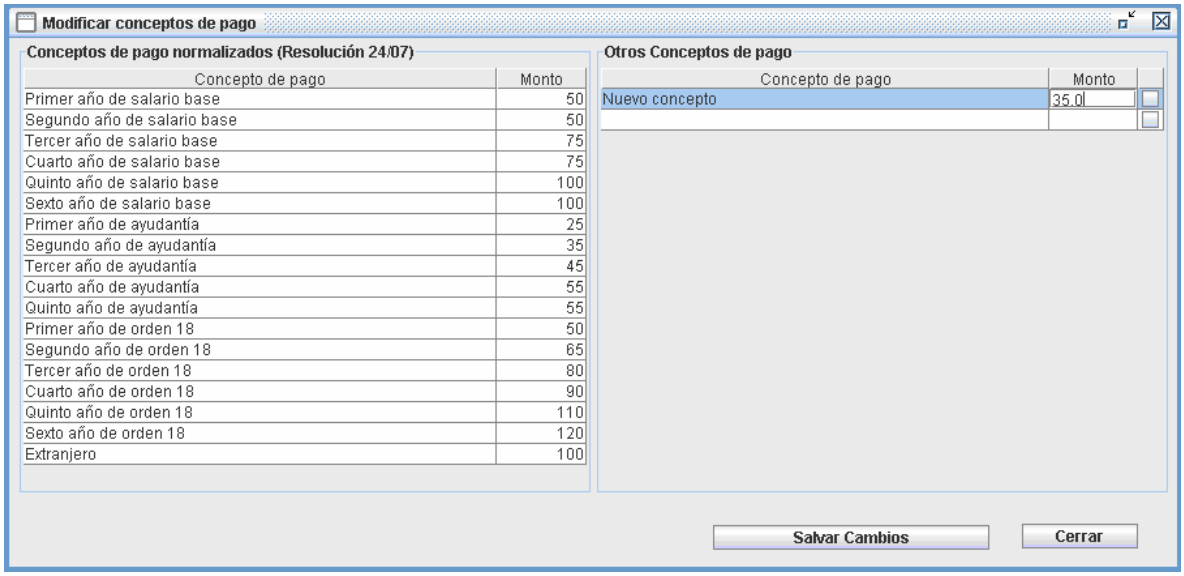


Figura 3.22: Interfaz de Cantidades a pagar a los estudiantes.

En el panel izquierdo aparecen los conceptos de pagos estándares y en el panel de la derecha aparecen los otros conceptos de pagos adicionales. Los conceptos estándares solo se pueden modificar cambiándole el monto. Los conceptos de pagos adicionales se pueden modificar o crear. Se salvarán los cambios presionando el botón Salvar cambios. .

3.3.6 Establecer los responsables de firmar las nóminas

Esta opción es utilizada por el Economista para establecer quienes intervienen en las validaciones de las nóminas

Seleccionar en el menú **Codificadores** la opción **Pies de página de las nóminas** (Figura 3.23).

Figura 3.23: Interfaz de la aplicación –Codificadores – Pies de página de las nóminas**.**

Se visualiza la siguiente ventana que permite establecer los responsables de las nóminas (Figura 3.24).

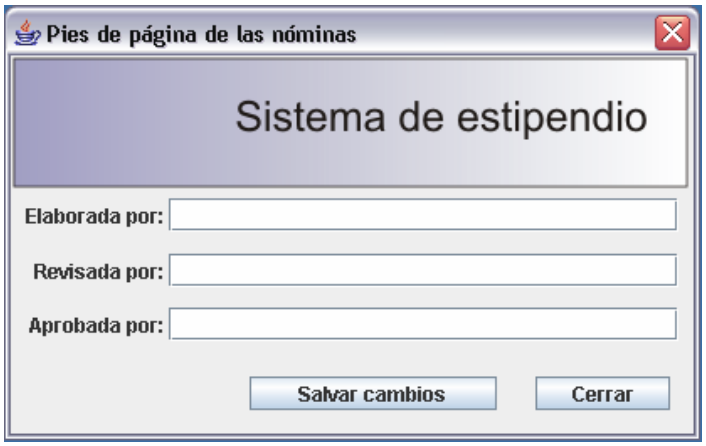


Figura 3.24: Interfaz de responsables de las nóminas.

En esta ventana se tiene que ingresar todos los responsables de las nóminas. Se salvarán los cambios al presionarse el botón 

.

3.3.7 Procesar solicitudes de reintegro

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

Seleccionar en el menú **Analizar solicitudes** la opción **Reintegro** (Figura 3.25).

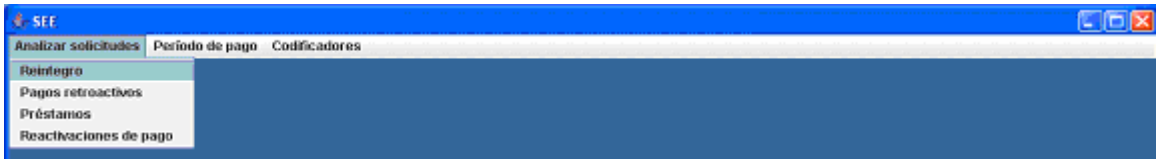


Figura 3.25: Interfaz de la aplicación –Analizar solicitudes – Reintegro

Se visualiza la siguiente ventana que permite consultar, aprobar o rechazar las solicitudes de reintegros ingresadas al sistema. (Figura 3.26)

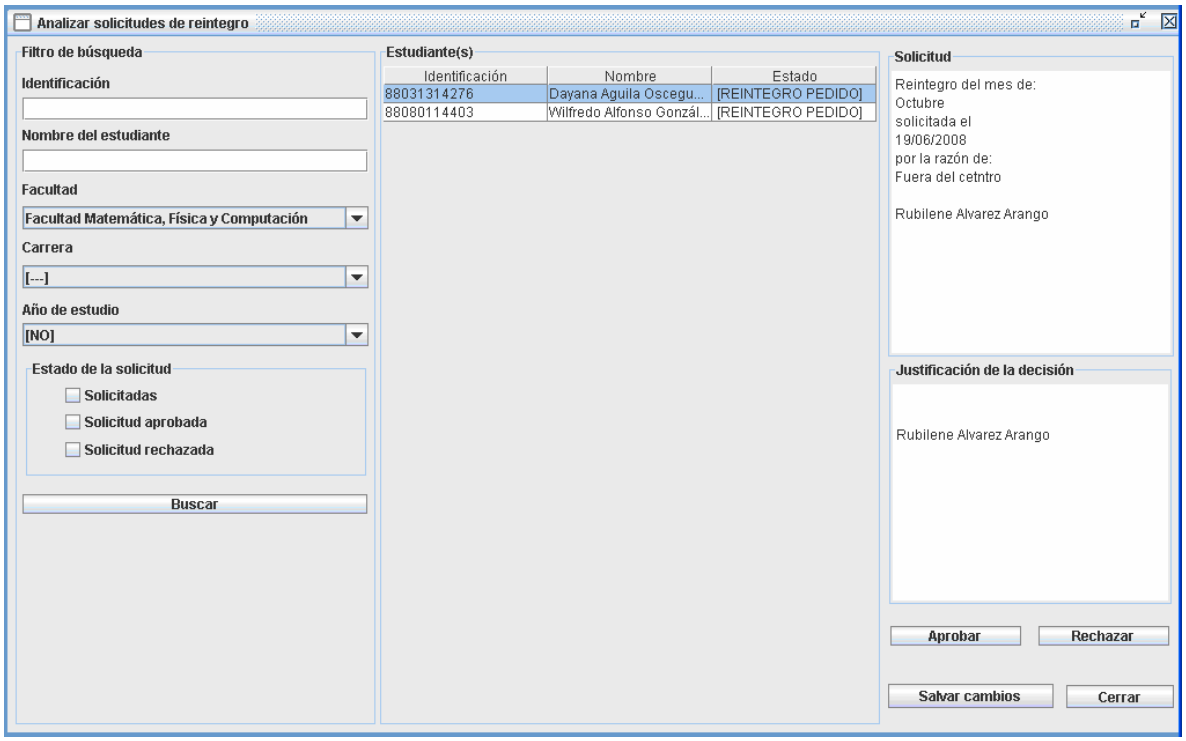


Figura 3.26: Interfaz de Analizar las solicitudes de reintegros.

En el panel de la izquierda aparecen los filtros de búsqueda. Después de seleccionarse el criterio por el cual se va a realizar la búsqueda aparece en el panel del centro los estudiantes buscados, cuando se selecciona uno de estos estudiantes, en el panel de la derecha en la parte superior aparece los datos de la solicitud del estudiante seleccionado .Se ingresa la justificación de la decisión tomada Las solicitudes se pueden rechazar y aprobar presionando los botones Aprobar y Rechazar. Se salvarán los cambios presionando el botón Salvar cambios. .

3.3.8 Procesar solicitudes de pagos retroactivos

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

Seleccionar en el menú **Analizar solicitudes** la opción **Pagos retroactivos**. (Figura 3.27)

Figura 3.27: Interfaz de la aplicación - opción del menú Estipendio –Chequeo de las solicitudes de pagos retroactivos

Se visualiza la siguiente ventana que permite consultar, aprobar o rechazar las solicitudes de pagos retroactivos ingresadas al sistema. (Figura 3.28)

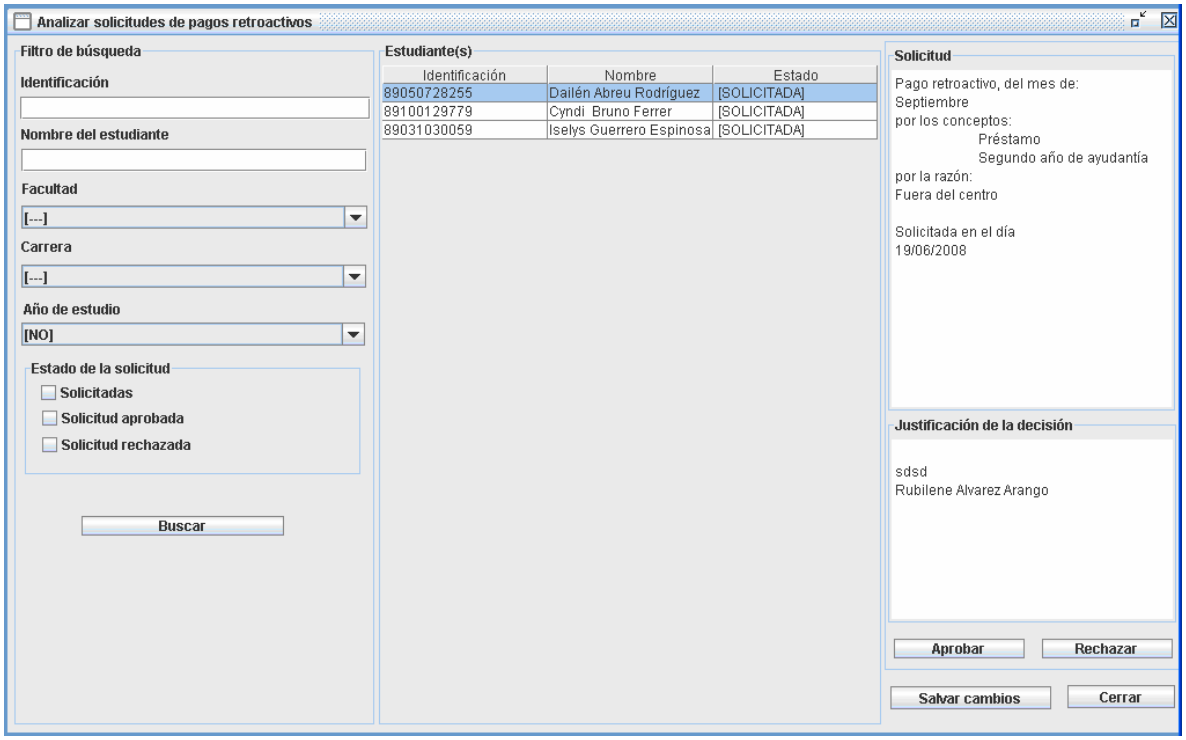


Figura 3.28: Interfaz de Analizar las solicitudes de pagos retroactivos

En el panel de la izquierda aparecen los filtros de búsqueda. Después de seleccionarse el criterio por el cual se va a realizar la búsqueda aparecen en el panel del centro los estudiantes buscados, cuando se selecciona uno de estos estudiantes, en el panel de la derecha en la parte superior aparece los datos de la solicitud del estudiante seleccionado. Se ingresa la justificación de la decisión tomada .Las solicitudes se pueden rechazar y aprobar presionando los botones Aprobar y Rechazar. Se salvarán los cambios presionando el botón Salvar cambios.

3.3.9 Procesar solicitudes de Préstamos

Esta opción es utilizada cuando el Economista 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 **Analizar solicitudes** la opción **Préstamos** (Figura 3.29).

Figura 3.29: Interfaz de la aplicación - opción del menú Analizar solicitudes –Préstamos

Se visualiza la siguiente ventana que permite consultar, aprobar o rechazar las solicitudes de préstamos ingresadas al sistema. (Figura 3.30)

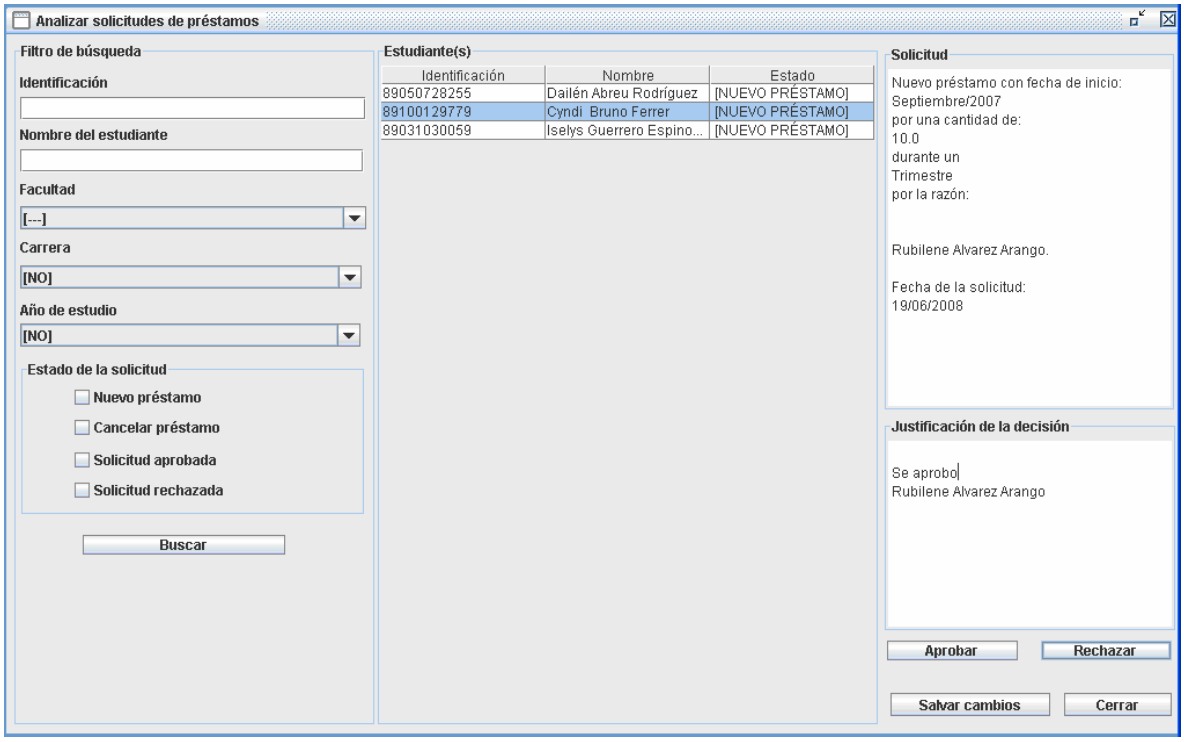


Figura 3.30: Interfaz de Analizar las solicitudes de préstamos

En el panel de la izquierda aparecen los filtros de búsqueda. Después de seleccionarse el criterio por el cual se va a realizar la búsqueda aparecen en el panel del centro los estudiantes buscados, cuando se selecciona uno de estos estudiantes, en el panel de la derecha en la parte superior aparece los datos de la solicitud del estudiante seleccionado. Se ingresa la justificación de la decisión tomada. Las solicitudes de puede rechazar y aprobar presionando los botones Aprobar y Rechazar. Se salvarán los cambios presionando el botón Salvar cambios. .

3.3.10 Procesar solicitudes de Activación

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

Seleccionar en el menú **Analizar solicitudes** la opción **Reactivación de pago**. (Figura 3.31).

Figura 3.31: Interfaz de la aplicación - opción del menú Analizar solicitudes –Reactivación de pago

Se visualiza la siguiente ventana que permite consultar, aprobar o rechazar las solicitudes de activación ingresadas al sistema. (Figura 3.32)

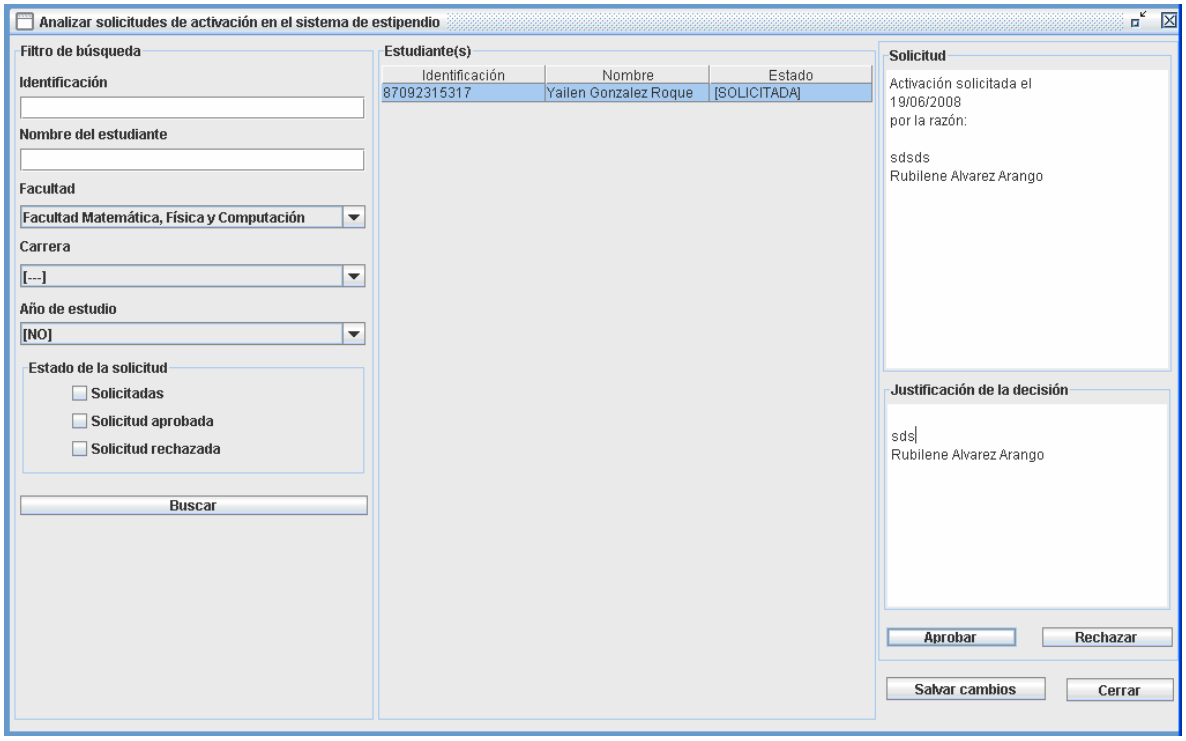


Figura 3.32: Interfaz de Analizar las solicitudes de activación

En el panel de la izquierda aparecen los filtros de búsqueda. Después de seleccionarse el criterio por el cual se va a realizar la búsqueda aparece en el panel del centro los estudiantes buscados, cuando se selecciona uno de estos estudiantes, en el panel de la derecha en la parte superior aparece los datos de la solicitud del estudiante seleccionado. Se ingresa la justificación de la decisión tomada. Las solicitudes de puede rechazar y aprobar presionando los botones Aprobar y Rechazar. Se salvarán los cambios presionando el botón Salvar cambios.

3.4 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.

3.5 Resumen del capítulo

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 permita con facilidad su auditoría.

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 economista 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.

Conclusiones generales.

En este trabajo quedaron demostradas las hipótesis inicialmente planteadas. En particular:

- Se pudo obtener un *plug-in* integrado exitosamente en el SIGENU que modela correctamente los requerimientos legales del pago del estipendio estudiantil en los CES de Cuba, a partir del uso de herramientas que cumplen con las definiciones establecidas por la dirección del proyecto SIGENU.
- Se destaca el uso de herramientas novedosas como el Spring e Hibernate para el manejo del negocio y la persistencia respectivamente.
- Evitando la pesada estructura de los EJBs se pudo implementar la lógica del negocio en una forma eficiente.

El producto final es un módulo que cumple con las restricciones básicas para el manejo del estipendio y sólidamente implementadas. Este proyecto integró el conocimiento aprendido en diferentes materias de la carrera, así como aportó nuevas habilidades y saber de otras áreas de la computación.

Recomendaciones

Como recomendaciones para próximas versiones se podría adicionar a los clientes facilidades para borrar los historiales y corregir ciertos errores que pudieran cometer los economistas, así como controlar sus cuentas de usuario en el sistema. Para esto habría que introducir un nuevo actor que ocuparía el rol de administrador.

Bibliografía

1. Grupo Nacional de Diseño Sistema-GNU. *Sistema para la Informatización de la Gestión de la Nueva Universidad*. Taller Software MIC-MES 2005 Noviembre 2005 [cited; 23].
2. Martínez, A. and Y. Vallejo, *Sistema GNU Módulo de Matrícula*. 2005, Instituto Superior Politécnico "José Antonio Echeverría": Ciudad de La Habana.
3. Gutiérrez, A. and A. Pérez, *Sistema GNU Módulo de Control Docente*. 2005, Instituto Superior Politécnico "José Antonio Echeverría": Ciudad de La Habana.
4. Apezteguía, M.G., *Sistema GNU Módulo de Plan de Estudio*. 2005, Instituto Superior Politécnico "José Antonio Echeverría": Ciudad de La Habana.
5. Espinosa, L.M.P. and L.C. Amador, *Sistema GNU Módulo de Estadística CES*. 2005, Instituto Superior Politécnico "José Antonio Echeverría": Ciudad de La Habana.
6. Pino, M.C.P., *Sistema GNU Módulo de Estadística MES*. 2005, Instituto Superior Politécnico "José Antonio Echeverría": Ciudad de La Habana.
7. Saúco, I.A.F., *Coordinador de Transacciones Distribuidas de Datos para Java*. 2005, Instituto Superior Politécnico "José Antonio Echeverría": Ciudad de La Habana.
8. Alemán, Y. 2005.
9. Dariem Pérez, J.J.L., *Sistema Estudiante – GNU. Gestión de la Nueva Universidad. Módulo de Estipendio* 2006, UCLV.
10. superior, M.d.e., *Resolución 24/07*, M.d.E. Superior, Editor. 2007.
11. Birsan, D. (2005) *On Plug-ins and Extensible Architectures*. **Volume**, 4
12. *Design Patterns*. **Volume**,
13. Brown, K. (2001) *Rules and Patterns for Session Facades*. **Volume**,
14. The PostgreSQL Global Development Group, *PostgreSQL Official Documentation*. 2003.
15. *New IReport Manual*. 2005.
16. Johnson, R., *Expert One-on-One J2EE Design and Development*. 2003, Indianápolis, Indiana: Wiley Publishing.
17. Johnson, R. and J. Hoeller, *Expert One-on-One J2EE Development without EJB*. 2004, Indianápolis, Indiana: Wiley Publishing. 578.
18. Walls, C. and R. Breidenbach, *Spring in Action*. 2005: Manning Publications Co. Copyeditor: Liz Welch

209 Bruce Park Avenue Typesetter: Denis Dalinnik

Greenwich, CT 06830 Cover designer: Leslie Haimes. 472.

19. Rational Software Corporation, *Online Help*. 2003.

Anexo

Términos

Activo en el sistema de estipendio: un estudiante puede ser un estudiante que puede cobrar estipendio, pero no estar activo en el sistema por varias razones, como pueden ser haber dejado de cobrar tres meses consecutivos.

Nómina abierta o cerrada: cuando se crea la nómina sus estudiantes aparecen con un estado indefinido, esto significa que como no se ha realizado el pago no se sabe cuales estudiantes cobraron y cuales no. El economista debe revisar la nómina y actualizar este estado con el estado de cobro. Si aparece una nómina con algún estudiante con estado indefinido se dice que la nómina está abierta.

Abrir operaciones: abrir las operaciones es indicar que las secretarias tienen el sistema nuevamente habilitado para realizar las solicitudes de estipendio.

Bibliotecas

Para que el *plug-in* de secretaría funcione correctamente deben adicionarse a la carpeta *lib* del *Skeleton* las siguientes bibliotecas:

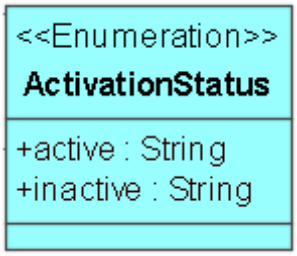
spring-1.2.5.jar
stipend-sys-common-2.0.jar
stipend-sys-core-2.0.jar

Cuando se recompile el negocio debe tenerse en cuenta actualizar las bibliotecas *stipend-sys-common-2.0.jar, stipend-sys-core-2.0.jar.*

Enumerativos

Para la implementación del código se utilizaron varios enumerativos para la representación de valores nominales y así hacer el código mucho más legible y fácil de mantener.

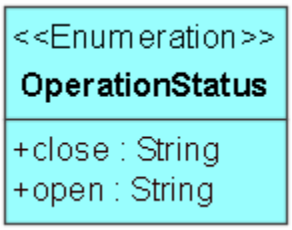
ActivationStatus se utiliza para definir el estado de un estudiante en el sistema de estipendio.



Propiedad	Descripción
active	Activo en el sistema
inactive	Inactivo en el sistema

Descripción de los valores de *ActivationStatus*.

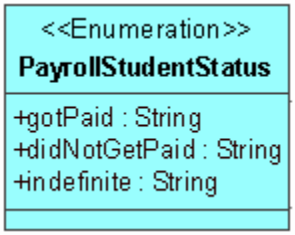
OperationStatus se utiliza para definir el estado de las operaciones de una facultad y para definir el estado de la nómina.



Propiedad	Descripción
close	Las operaciones están cerradas. La nómina no tiene ningún estudiante en estado inactivo.
open	Las operaciones están abiertas. La nómina tiene estudiantes con estado inactivo.

Descripción de los valores de *OperationStatus*

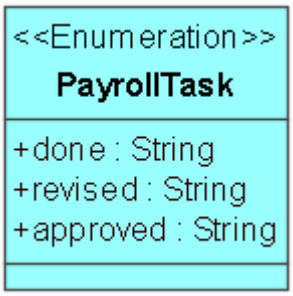
PayrollStudentStatus se utiliza para definir el estado de cobro de los estudiantes de una nómina.



Propiedad	Descripción
gotPaid	El estudiante cobró el estipendio.
didNotGetPaid	El estudiante no cobró el estipendio.
indefinite	No se le ha declarado el estado de cobro al estudiante.

Descripción de los valores de *PayrollStudentStatus*

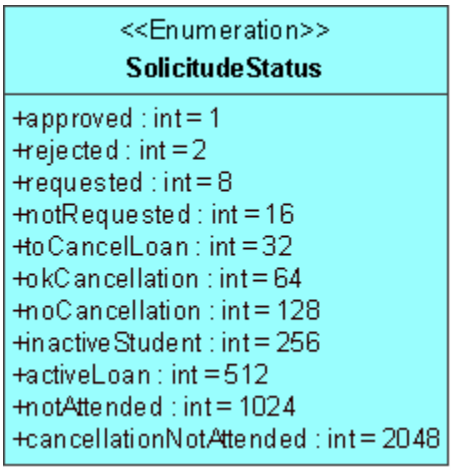
PayrollTask son las acciones que le realizan a la nómina los economistas. Quienes las realizan van al pie de la página de la nómina.



Propiedad	Descripción
done	Elaborar.
revised	Revisar.
approved	Aprobar.

Descripción de los valores de *PayrollTask.*

SolicitudeStatus estados por los que pueden pasar las solicitudes.

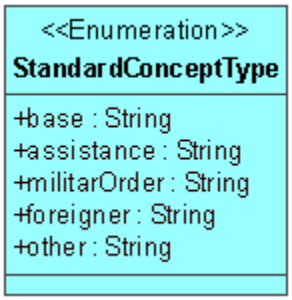


Propiedad	Descripción
approved	La solicitud ya fue aprobada por el economista.
rejected	La solicitud ya fue rechazada por el economista.
requested	La solicitud está pedida pero no ha sido analizada por la dirección económica.
notRequested	Se utiliza para representar en el filtro de búsqueda que se quiere buscar aquellos estudiantes que no han hecho una solicitud.
toCancelLoan	Es una solicitud de cancelación de préstamo que no ha sido analizada por la dirección económica.
okCancellation	Es una solicitud de cancelación de préstamo aprobada por la dirección de economía.
noCancellation	Es una solicitud de cancelación que ha sido rechazada por la dirección de economía.
inactiveStudents	Se utiliza en los filtros para buscar las solicitudes de los estudiantes que están inactivos.
activeLoan	Se utiliza en los filtros para buscar las

	solicitudes de los estudiantes con préstamos activos.
notAttended	Para representar las solicitudes que no fueron revisadas por la dirección de economía.
cancellationNotAttended	Representa las solicitudes de cancelación de préstamo que no fueron revisadas por la dirección de economía.

Descripción de los valores de *SolicitudeStatus*

StandardConceptType representan los tipos de conceptos estándares.



Propiedad	Descripción
base	Salario base de los estudiantes que no son orden 18.
assistance	Ayudantías.
militarOrder	Salario base de los estudiantes que son orden 18.
foreigner	Salario de los extranjeros subvencionados.
other	Para representar en los conceptos de pago adicionales que no es de ayudantía.

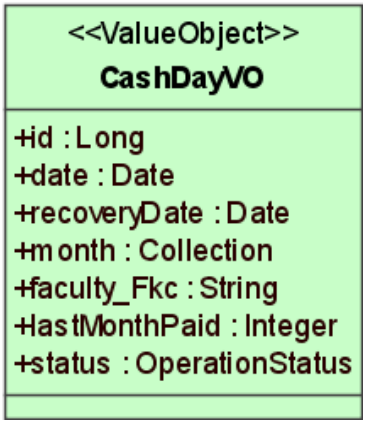
Descripción de los valores de *StandardConceptType*

Objetos de presentación

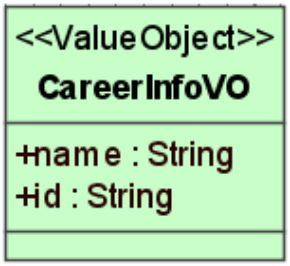
Para viajar a través de las capas de la aplicación se hizo un diseño de clases. Por cada clase persistente del estipendio se creó un objeto de presentación con los mismos atributos y para las

clases que no son nuestras solo se le adicionó la información necesaria, también se diseño un filtro que sirviera para restringir las diferentes búsquedas.

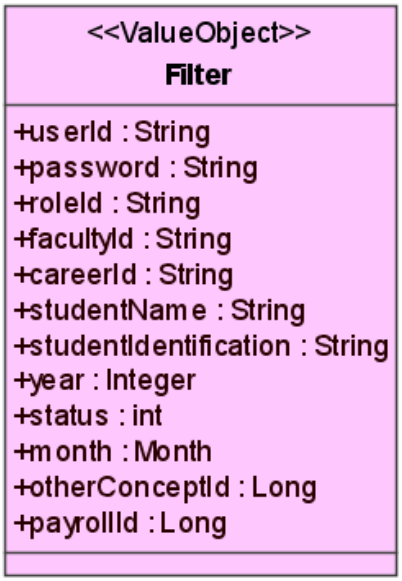
CashDayVO: objeto de presentación asociado a la clase *CashDay*.



CareerInfoVO: objeto de presentación de una clase persistente que no es de estipendio. Agrupa la información relevante de una carrera.



Filter: filtro de las diferentes búsquedas de la aplicación.



Propiedad	Descripción
userId	Identificador del usuario que entra al sistema.
password	Contraseña del usuario en el sistema
roleId	Rol del usuario en el sistema.
facultyId	Identificador de la facultad.
careerId	Identificador de la carrera.
studentName	Subcadena del nombre del estudiante.
StudentIdentification	Subcadena del carné de identidad del estudiante.
year	Año de estudio.
status	Es un entero que recoge el resultado de una operación lógica con los estados de las solicitudes.
month	Mes del curso.
otherConceptId	Identificador de un concepto adicional.
payrollId	Identificador de una nómina.

Descripción de los valores de *Filter*.