Universidad Central “Martha Abreu” de Las Villas

Facultad de Matemática – Física – Computación

Título

Módulo de Control de Alumnos Ayudantes del SIGENU. Versión 2.0

Autores

Dasiel Rodríguez Hernández

Enrique Escobar Ramírez

Tutores

M.Sc Libia García Águila

Lic. Alcides Morales Guedes

Carrera

Ciencias de la Computación

Departamento

Desarrollo y Producción de Software

Resumen

En la actualidad existe la necesidad de incrementar la eficiencia y calidad de los diferentes procesos docentes en los Centros de Educación Superior o Sedes Universitarias Municipales, por lo que se planteó la tarea de implementar un sistema que automatizara todos estos procesos, de ahí surge el Sistema de Gestión de la Nueva Universidad, separado en diferentes módulos, entre los que se encuentra el de Control de Alumnos Ayudantes, del cual trata el presente trabajo.

En el trabajo se manejan diferentes conceptos como: Software Libre, *Java 2 Enterprise Edition*, *Model Driven Architecture*, *Model View Controller*, los cuales constituyen el sustento teórico de nuestra investigación.

Según la metodología *Model Driven Architecture* la aplicación se hizo en tres etapas, primeramente se realizó el diseño que luego se utilizaría para la segunda etapa: la generación de código, y por último la implementación del sistema.

El diseño del sistema se basa en el patrón Model View Controller, un patrón de diseño que separa la aplicación en componentes: modelo, vista y controlador.

El sistema está implementado sobre plataforma *Java 2 Enterprise Edition*, brindando así mayor seguridad, robustez y confiabilidad al sistema, y permitiendo su implantación en cualquier plataforma.

Abstract

Today, there is a growing need for better and more effective ways to approach the teaching-learning process in the area of Higher Education and its Adjacent Academic Headquarters. Hence, the importance of automating such processes and the genesis of a Management System for the New University that deals with separate modules among which a System for the Control of Assistant Students is inserted; and therefore the purpose of the present thesis work.

It comprises a body of theoretical background information that includes concepts like Free Software, Java 2 Enterprise Edition, Model Driven Architecture and Model View Controller.

Based on the Model Driven Architecture methodology, the application is developed in three stages. First, the application design, second, the code generation process, and finally the custom code implementation.

The system design lies in the basic pattern Model View Controller; a design pattern that splits application into model, view and controller.

The system will be implemented on Java 2 Enterprise Edition platform, thus offering greater security, robustness and reliability to the system, and allowing its implantation in any platform.

Índice

Introducción

En principio la Universidad Central "Martha Abreu" de Las Villas y otras universidades elaboraron sistemas automatizados para controlar la gestión estudiantil, pues no existía uno unificado a nivel de Ministerio de Educación Superior, por lo que el control de la actividad docente se realizaba a partir de bases de datos con plataformas y estructuras muy diferentes en cada centro.

En particular, en la UCLV, el Sistema de Control de Estudiantes estaba implementado sobre SQL2000 (García, 1995). El Sistema no tenía un módulo para la gestión de alumnos ayudantes.

Debido a las actuales necesidades de elevar la eficiencia y calidad de todos los procesos docentes de la educación superior, la dirección de Informatización del Ministerio de Educación Superior (MES), organizó el Taller "Automatización de la Gestión Universitaria" donde participaron numerosos especialistas de diferentes universidades, con el objetivo concreto de aunar esfuerzos para lograr el desarrollo de un sistema automatizado capaz de controlar a nivel nacional la gestión académica de los estudiantes que ingresan a los distintos cursos de nivel superior. Así surge el proyecto Sistema GNU (SIGENU: Sistema de Gestión de la Nueva Universidad).

Concretamente, ya está en uso 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 y la Educación a Distancia. Se han comenzado a desarrollar diversas aplicaciones consecuentes con esto, como por ejemplo el Sistema de Matrícula y Control Docente (Cliente SIGENU), Sistema de Estadísticas para el CES y para el MES, todos formando módulos del Sistema GNU (SIGENU).

La UCLV participó también en el diseño de la base de datos del nuevo sistema SIGENU y ha sido vinculada en el desarrollo de aplicaciones para el mismo, por lo que

se planteó la tarea de implementar un sistema que automatizara todos estos procesos, separado en diferentes módulos, entre los que se encuentra el de Control de Alumnos Ayudantes, del cual trata el presente trabajo. Con anterioridad ya este ha sido un punto de investigación en el que se han obtenido resultados sólidos, lográndose la versión 1.0 del Sistema de Control de Alumnos Ayudantes. A modo de continuidad y como respuesta a los nuevos problemas y resoluciones planteadas en el VII Congreso de la FEU, el Ministerio de Educación Superior se ha propuesto una serie de tareas entre las que se encuentra el desarrollo de una nueva versión para el Sistema de Control de Alumnos Ayudantes.

Una de las principales definiciones del proyecto SIGENU fue la de utilizar plataformas de *software* libre y multiplataformas J2EE™ / DBMS PostgreSQL /servidor de aplicaciones JBOSS™ /Generador AndroMDA (que pueden funcionar sobre *Windows* y *LINUX-UNIX*). *CASE* propietario MagicDraw). 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.

Nuestro problema de investigación consiste entonces en la elaboración de un Sistema de Control de toda la actividad relacionada con los Alumnos Ayudantes, capaz de responder a las nuevas necesidades surgidas después de la implementación de la versión 1.0 y formalizadas en el Capitulo V de la Resolución Ministerial No. 210/07, que sea compatible con las definiciones y módulos ya establecidos por el SIGENU y que en particular esté elaborado sobre plataforma de *software* libre. Se tiene pues, la ventaja de la existencia de tales definiciones y módulos anteriores además de un sistema de referencia.

Preguntas de investigación:

1. ¿Cuáles son las necesidades de todos los posibles usuarios de un Sistema de Control de Alumnos Ayudantes de un Centro de Educación Superior?
2. ¿Cuál será la vía más apropiada para implementar el Sistema de forma relativamente sencilla y que garantice a su vez la absoluta compatibilidad con el SIGENU.
3. ¿Es el ambiente *LAN* la técnica apropiada para desarrollar un sistema que responda rápido y de forma precisa a las necesidades del cliente?
4. ¿La menor o mayor eficiencia de la seguridad del sistema varía según la implementación de un ambiente *Desktop* para la secretaria?

Consecuentemente el presente trabajo se planteó como objetivo general el siguiente:

Objetivo general:

Desarrollar un Sistema de Control de Alumnos Ayudantes para la Educación Superior Cubana, usando Plataforma *Open Source*, en particular PostgreSQL y Java, de manera que sea compatible con el Sistema de Control de Estudiantes de la Nueva Universidad, que responda a los requerimientos formulados en el Capítulo V de la Resolución Ministerial No. 210/07 y que permita el control de los alumnos ayudantes en cualquier Centro de Educación Superior del país por todas las dependencias o personas que estén relacionadas con el mismo, incluyendo al propio estudiante.

Como objetivos específicos tenemos:

Objetivos específicos:

1. Desarrollar el Sistema de Control de Alumnos Ayudantes de manera que responda a una plataforma *Open Source* compatible con el SIGENU.
2. Desarrollar tal sistema de manera que responda a un ambiente *Web* para que pueda utilizarse desde cualquier lugar en que se encuentre el usuario interesado.

3. Desarrollar una aplicación *Desktop* para adicionarla al módulo de matrícula utilizado por las secretarias, garantizando así la seguridad y consistencia de la Base de Datos.
4. Responder a los requerimientos de todos los posibles usuarios del sistema, en particular, los jefes de departamentos, las secretarias y el propio estudiante.

El cumplimiento de estos objetivos específicos supone, como es lógico, analizar cuáles son las particularidades que trata la nueva Resolución Ministerial sobre la gestión de alumnos ayudantes y las condiciones actuales de la universalización.

El Módulo de Control de Alumnos Ayudantes traerá como beneficio esencial la eliminación del trabajo manual de las secretarias de los CES y de las SUM, además de brindar un papel protagónico a los profesores, quienes realmente son los que tienen el control del trabajo de los alumnos ayudantes. Se tendrá en cuenta también la elaboración de una respuesta rápida ante cualquier solicitud por parte del MES y una mayor cantidad de funcionalidades que actualmente no existen. De esta manera se debe incrementar la eficiencia y calidad en el proceso de gestión de alumnos ayudantes.

El sistema estará implementado sobre la plataforma J2EE™ ya que se requiere que sea seguro, confiable y que posea una arquitectura robusta, además que sea implantado sobre cualquier plataforma de *software*, cumpliendo así con las tendencias actuales del *software* libre.

Hipótesis de Trabajo.

1- Se diseñará un modelo del negocio capaz de validar y resolver las actividades relacionadas con todos los actores que intervienen en el trabajo de los Alumnos Ayudantes.

2- La implementación de una aplicación web se encargará de manipular toda la información referente a los Alumnos Ayudantes además de ser accesible desde cualquier sitio por los usuarios del sistema.
3- La implementación de una aplicación *Desktop* para ser insertada en el módulo de la secretaria, producirá mejores resultados para manejar la entrada/salida de la base de datos y a su vez jugará un papel fundamental en la seguridad del sistema, permitiendo únicamente a la secretaria acceder a los datos reales que maneja la aplicación.

Forma en que se comprobarán las hipótesis del trabajo.

Una vez terminada la fase de implementación del módulo de Alumnos Ayudantes se pasará a comprobar su funcionamiento y rendimiento. Se implantará a modo de prueba en la Universidad Central "Martha Abreu" de Las Villas y en algunas Sedes Universitarias Municipales de Villa Clara. Para comprobar el nivel de aceptación y eficacia se realizarán encuestas a un gran número de usuarios potenciales del sistema, principalmente a secretarias y jefes de departamento. Con los resultados obtenidos se hará una valoración teniendo en cuenta las recomendaciones brindadas y se efectuarán las modificaciones necesarias a modo de conclusión.

CAPITULO I.

COMPONENTES Y ELEMENTOS TEORICOS RELACIONADOS CON EL ENTORNO DEL SISTEMA.

Para el desarrollo del sistema fue necesario realizar un proceso de investigación acerca del control de los alumnos ayudantes en nuestro país, así como la tecnología que iba a ser utilizada.

1.1 Los alumnos ayudantes

Con la revolución se hicieron asequibles las Universidades, las cuales estaban fuera del alcance de las personas carentes de recursos financieros para costear su matrícula y los libros de texto. Con el transcurso del período revolucionario ingresaron a los centros de altos estudios de la isla muchas personas que, con la matrícula gratuita, pudieron culminar especialidades de diversos cortes, incluyendo las científicas, de donde han surgido tantos destacados profesionales.

Esto trajo consigo que existiera un déficit de profesores, sin embargo, el modelo pedagógico cubano, donde la educación es uno de los pilares básicos de la sociedad, permitió crear el Movimiento de Alumnos Ayudantes.

Como bien queda expresado en la Resolución Ministerial 210/07 el Movimiento de Alumnos Ayudantes (MAA) lo constituye el conjunto de estudiantes previamente seleccionados en las carreras, preferentemente a partir del segundo año de la especialidad, que se distingan por mostrar ritmos de asimilación más rápidos, aptitudes favorables para el aprendizaje de alguna o algunas asignaturas del plan de estudio y que puedan asimilar tareas complementarias, con el propósito de formarlos como docentes y contribuir así a satisfacer las necesidades de las diferentes disciplinas y asignaturas (Ministerio de Educación Superior, 2007).

Los alumnos ayudantes constituyen una de las principales fuentes de apoyo en todo el proceso de la enseñanza superior en nuestro país, no solo contribuyendo en la docencia sino también en otras funciones no menos importantes como la investigación, la atención de laboratorios, realización de experimentos, etc.

Como se expresa en la resolución antes mencionada, cada estudiante seleccionado deberá ser asignado a un departamento docente, que será el responsable de su plan de formación. Para ello designará a los profesores que atenderán la formación de cada estudiante, tanto de las sedes centrales como de las sedes universitarias. Los profesores designados poseerán los conocimientos y la experiencia necesarios para desarrollar esta labor con calidad y dispondrán del tiempo requerido para la adecuada atención a los estudiantes seleccionados. Se procurará que el profesor atienda al estudiante durante toda la etapa en que sea alumno ayudante.

La evaluación del trabajo de los alumnos ayudantes es responsabilidad del jefe del departamento docente correspondiente. Esta evaluación se realizará a partir del cumplimiento del plan de trabajo como alumno ayudante, los resultados docentes alcanzados y la disciplina observada. Se tendrán en cuenta, como aspectos esenciales, los criterios del profesor designado para su atención y la opinión de las organizaciones juveniles. Esta evaluación se realizará al finalizar cada curso académico.

El jefe del departamento docente analizará el resultado de la evaluación con el estudiante y el profesor designado para la formación, y se incluirá en el expediente del estudiante. Se estimularán los resultados de esta labor. En la evaluación del trabajo de los alumnos ayudantes que desarrollan su actividad docente o realizan sus estudios superiores en las sedes universitarias, el director de la sede participará en la evaluación.

Las secretarías de cada facultad apenas reciben información de los departamentos sobre los Alumnos Ayudantes (AA) con el propósito de actualizar su estipendio. En la práctica, puede ocurrir fácilmente que un estudiante realice funciones de AA y no sea

considerado como tal o viceversa. Por lo que dejamos en manos del profesor que los atiende por el departamento docente las formas de control necesarias para esta actividad.

Sin embargo, en cualquier caso la actividad de ayudantía es remunerada al estudiante de acuerdo a la Resolución Ministerial 210/07. Por ello también nuestro sistema debe estar en comunicación con el Sistema de Estipendio que se ha elaborado paralelamente al presente.

Otro punto importante que se trata en el Artículo 222 de dicha resolución plantea que: Los Rectores quedan facultados para:

a) Ajustar la planificación y organización del proceso docente educativo para el alumno ayudante, en caso necesario, de modo que disponga de mayor tiempo para realizar las tareas previstas en su plan de trabajo.
b) Modificar el plan de estudio de los alumnos ayudantes a partir de la propuesta de los decanos, para adicionar, suprimir o sustituir asignaturas o disciplinas del plan de estudio que cursa, en aras de lograr la formación necesaria para la labor que realiza.
c) Aprobar la modificación del tiempo de estudios, de modo que el alumno ayudante pueda graduarse en un plazo menor que el establecido, para aquellos casos en que el plan de formación establece un tiempo mínimo determinado para graduarse. Se tendrá en cuenta para dicha aprobación el aprovechamiento demostrado, las tareas que cumple como alumno ayudante, las opiniones del profesor designado para su atención y la de las organizaciones estudiantiles. (Ministerio de Educación Superior, 2007).

1.2 Marco Teórico

En el desarrollo de toda aplicación, es necesario hacer un análisis previo sobre la tecnología a usar. Nuestro sistema no estuvo exento a esto. Se hizo un estudio en

cuanto a las nuevas tendencias de producción de s*oftware*, además del análisis de las ventajas de usar s*oftware* libre en vez de *software* propietario.

1.2.1 Aplicaciones empresariales

Una aplicación empresarial no es más que una aplicación que brinda una serie de servicios a un gran número de usuarios de una misma organización o empresa.

El desarrollo de aplicaciones empresariales, y el desarrollo de aplicaciones *Web*, han tenido un auge muy significativo durante los últimos años. Frente a esta nueva demanda surgen varias plataformas para el desarrollo de este tipo de aplicaciones. Se destacan: J2EE™ (*Java 2 Enterprise Edition*) de *Sun Microsystems* ® y .NET de *Microsoft Corporation* ®.

Teniendo en cuenta las demandas de *software* libre, la antigüedad en el mercado y la certificación de las grandes compañías sobre el estándar J2EE™, ampliamos un poco más sobre la que se usa en el presente trabajo.

1.2.2 J2EE™

La plataforma *Java 2 Enterprise Edition* define un estándar para el diseño, desarrollo, ensamblaje y despliegue de aplicaciones empresariales. Al ser un estándar, no un producto, es necesario trabajar con alguna de las plataformas de desarrollo basadas en J2EE™ como *Sun ONE* o Jboss™. (Armstrong et al., 2005)

En J2EE™, las aplicaciones usan un modelo multicapas distribuido, permitiendo dividir el *software* en componentes según los roles y la funcionalidad necesaria, reduciendo así la complejidad del desarrollo distribuido. Los componentes de una

aplicación empresarial se dividen en cuatro capas: la capa cliente, la capa *Web*, la capa del negocio y la capa del Sistema de Información Empresarial.

1.2.3 Componentes en J2EE™

Un componente J2EE™ es una unidad de *software* auto-contenida y funcional que se ensambla en la aplicación, logrando la comunicación necesaria con otros componentes del sistema.

La especificación J2EE™ define los siguientes componentes:

1. Aplicaciones clientes y *applets* que se ejecutan en el cliente.
2. Los *Java Servlets* y *Java Server Pages*™ (JSP) son componentes *Web* que se ejecutan en el servidor.
3. Los *Enterprise JavaBeans*™ (EJB™) son componentes del negocio que se ejecutan en el servidor. (Armstrong et al., 2005)

Trataremos más adelante los conceptos de EJB™ y JSP por ser los utilizados.

1.2.3.1 Enterprise JavaBeans™

Un *Enterprise JavaBean* es un componente que encapsula la lógica del negocio de la aplicación. Por varias razones los *enterprise beans* simplifican el desarrollo de aplicaciones distribuidas. Primero, porque el contenedor de EJB™ provee distintos servicios como la seguridad y el manejo de transacciones, permitiendo al programador concentrarse en los servicios del negocio. Segundo, porque al encapsular la lógica del negocio de la aplicación los desarrolladores de la parte cliente se enfocan en la interfaz cliente. Tercero, porque son componentes portables que pueden utilizarse para el desarrollo de nuevas aplicaciones. (Armstrong et al., 2005)

Los *enterprise beans* se despliegan en el servidor de aplicaciones, donde el contenedor de EJB™ se encarga de manejarlos.

Existen varios tipos de componentes EJB™:

- *Session Beans*
- *Entity Beans*
- *Message-Driven Beans*

En los epígrafes siguientes explicamos éstos.

1.2.3.1.1 Session Beans

Los *Session Beans* son los componentes que se encargan de tareas o funciones para un cliente específico. Los *session beans* implementan la lógica del negocio. Dentro del servidor de aplicaciones, *un session bean* representa un cliente, por su misma naturaleza no es persistente, es decir, cuando el cliente cierra la sesión, el *bean* se destruye.

Hay dos tipos de *session beans*: con estado y sin estado, los primeros mantienen los valores de sus variables (estado) durante todo el tiempo de la comunicación cliente-*bean* mientras que en los segundos las variables del *bean* tienen valores pero solo mientras dura la invocación. (Armstrong et al., 2005)

1.2.3.1.2 Entity Beans

Un *Entity Bean* representa un objeto dentro del mecanismo de persistencia. Brinda una representación orientada a objetos de la base de datos. A diferencia de los *session beans*, los *entity beans* una vez que son creados, persisten hasta que se destruyen por invocaciones del cliente. Cuando se instancia un *entity bean,* él representa una fila específica en el caso de que sea de "grano fino", y en el caso que sea de "grano duro" se refiere a datos que se encuentren en varias tablas. (Armstrong et al., 2005)

Hay dos tipos de persistencia en los *entity beans*:

- Persistencia manejada por el *bean* (BMP).
- Persistencia manejada por el contenedor (CMP).

Si se usa BMP, el desarrollador del *bean* es el que se encarga de hacer las llamadas a la base de datos, en cambio, en CMP, el contenedor es el que las hace automáticamente, haciendo independiente la aplicación del mecanismo de persistencia usado. Por la ventaja anteriormente expuesta, y por las facilidades a la hora de desarrollar el *bean* escogimos usar CMP.

1.2.3.2 Java Server Pages™ (JSP)

JSP es la tecnología usada para generar páginas *Web* de forma dinámica en el servidor, desarrollado por *Sun Microsystems*®, está basado en *scripts* que utilizan una variante del lenguaje Java. (Armstrong et al., 2005)

La tecnología JSP, o de *Java Server Pages*™, es una tecnología Java que permite a los programadores mezclar HTML estático con HTML dinámico. Esta tecnología permite al código Java y a algunas acciones predefinidas ser mezcladas con el contenido estático de la *Web*. En las JSP, se escribe el texto que va a ser devuelto en la salida (normalmente código HTML) incluyendo código Java dentro de él para poder modificar o generar contenido dinámicamente.

El código Java es incluido dentro de las marcas de etiqueta <% y %>.
La principal ventaja de JSP es que la parte dinámica está escrita en Java. Por eso, en primer lugar, es mucho más poderosa y fácil de usar, y en segundo lugar, es portable a otros sistemas operativos y servidores *Web*.

JSP no se puede considerar un *script* al 100% ya que antes de ejecutarse, el servidor *Web* compila el *script* y genera un *servlet*; por lo tanto se puede decir que aunque este proceso sea transparente para el programador, no deja de ser una aplicación compilada.

Debido a todas las características explicadas, la tecnología JSP tiene en la actualidad mucho peso en el desarrollo *Web* profesional (sobre todo en *intranets*).

Microsoft, el más cercano competidor de Sun, ha visto en esta estrategia una amenaza, lo que le ha llevado a que su plataforma .NET incluya su lenguaje de *scripts* ASP.NET, que permite ser integrado con clases .NET (ya estén hechas en C++, Visual Basic o C#) del mismo modo que JSP se integra con clases Java. Sin embargo, las herramientas libres que implementan este estándar están solo en desarrollo.

1.2.4 Contenedores en J2EE™

En aplicaciones multicapas los clientes sencillos usualmente pueden complicarse por tener que manejar transacciones, multi-hilos, almacenamiento de recursos, manejo de estado y otros servicios de bajo nivel.

La arquitectura independiente de la plataforma y basada en componentes de J2EE™ hace que el desarrollo sea más simple, dado que la lógica del negocio se organiza en componentes reusables. Además, el servidor de J2EE™ provee servicios en la forma de contenedores para cada tipo de componentes.

Un contenedor es una interfaz entre un componente y la funcionalidad de bajo nivel de la plataforma que soporta ese componente. Antes de que un *enterprise bean* o un componente *Web* pueda ser ejecutado es necesario que se ensamble en un módulo J2EE™ y se despliegue en el servidor.

El proceso de ensamblaje implica especificar las configuraciones del servidor para cada componente de la aplicación J2EE™, como pueden ser la seguridad, el manejo de transacciones, los nombres *jndi* (*Java name and directory interface*) que serán usados por el servidor para la localización de los distintos componentes, etc.

El servidor de J2EE™ provee varios tipos de contenedores:

- Contenedor de EJB™
- Contenedor WEB
- Contenedor de aplicaciones cliente
- Contenedor de *Applets*

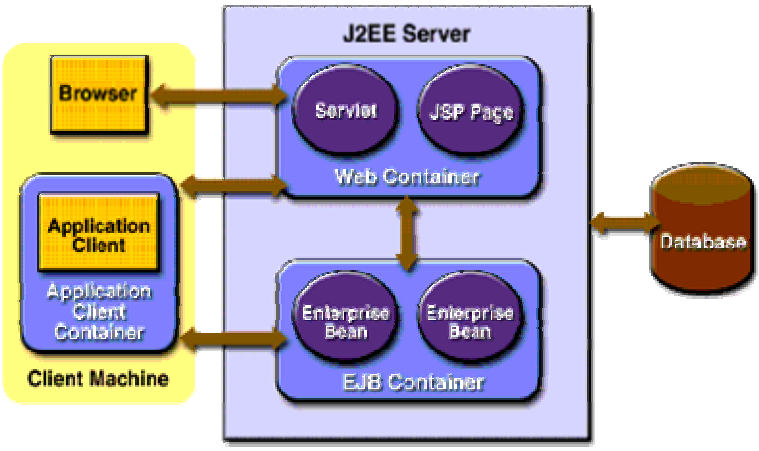


Figura 1.1 Servidor y Contenedor J2EE™.

Por la importancia que tiene el servidor de aplicaciones, se hace imprescindible hablar del Jboss™ (epígrafe 1.2.8), que es el que será usado en nuestro caso.

1.2.5 Aplicaciones clientes

Las aplicaciones clientes se ejecutan en la máquina cliente y proporcionan el camino a los usuarios para manipular las tareas. Típicamente están compuestas por una interfaz

gráfica de usuario (GUI por sus siglas en inglés) creadas con las interfaces de programación que brindan las bibliotecas de clases *Swing* o *Abstract Window Toolkit* (AWT), aunque es válida también una interfaz por línea de comando.
La aplicación cliente directamente accede a los *Enterprise Java Bean*™ (EJB™) que se encuentran desplegados en la capa de negocio del servidor, o puede suceder que la aplicación cliente inicie una conexión *http* para comunicarse con un *servlet* que se encuentra ejecutándose en la capa de presentación del lado del servidor. Particularmente en esta investigación se utilizó la primera variante, mediante la Invocación de Procedimientos Remotos (RMI).

1.2.6 Arquitectura de plug-ins necesaria para nuestro ambiente *Desktop*

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 es además muy beneficioso a la hora de manejar el problema de que las reglas de negocio pueden cambiar frecuentemente o bien pudieran surgir reglas nuevas. De esta forma se puede personalizar fácilmente una aplicación, mezclando y acoplando *plug-ins* según se necesite o creando uno nuevo para alguna opción inexistente.(Birsan 2005)

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

En la arquitectura de *plug-ins* pura, que es la que se ha usado en este trabajo, todo es un *plug-in* (figura 1.2). La aplicación servidora queda reducida entonces a un motor de

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

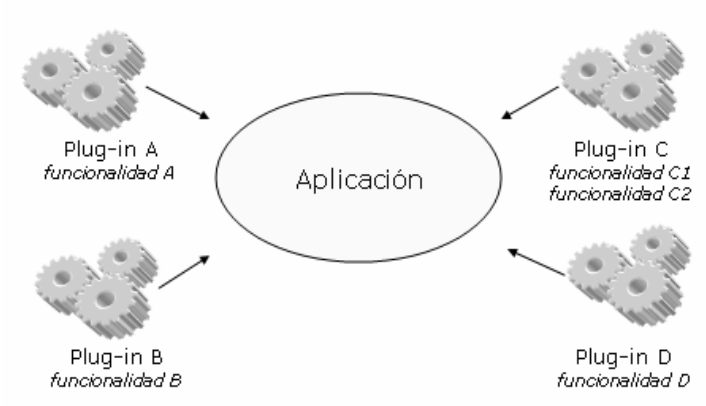


Figura 1.2 Arquitectura de *plug-in.*

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

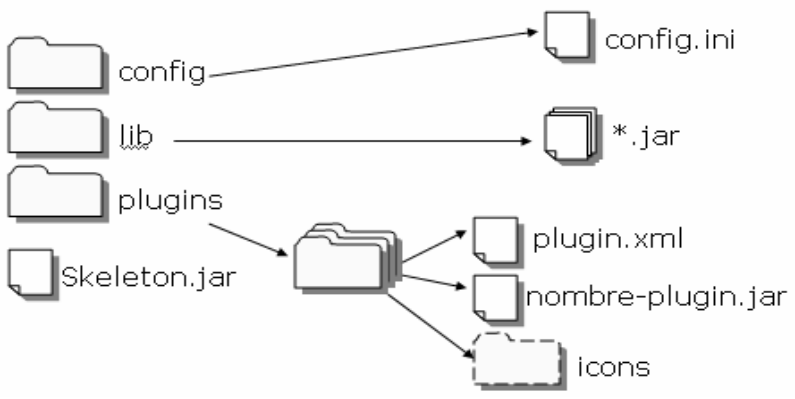


Figura 1.3 Estructura física de la aplicación cliente.

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

1.2.7 Hibernate

Con el soporte de la última versión disponible conocida como EJB3™, se simplifica notablemente el proceso de creación, cluster y persistencia de los componentes, sin sacrificar rendimiento. Una de sus principales ventajas es explotada para resolver la interrelación del modelo orientado a objetos de Java con el modelo relacional de persistencia del gestor de datos, mediante una herramienta de mapeo objeto-relacional nombrada Hibernate, que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones (Wikipedia, 2007). Como gestor de datos, se ha empleado PostgreSQL, servidor de base de datos objeto relacional libre, liberado bajo la licencia BSD que brinda altas prestaciones y soporta la funcionalidad SQL de reconocidos gestores propietarios como Oracle.

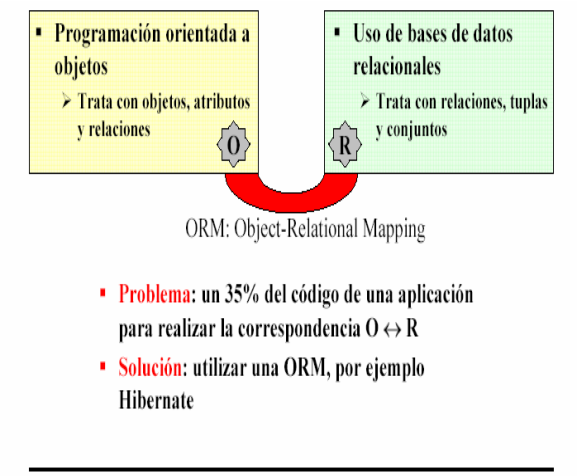


Figura 1.4 Importancia del uso de Hibernate.

1.2.8 Servidor de Aplicaciones Jboss™

El Servidor de Aplicaciones Jboss™ se ha convertido en el líder reconocido en el mercado de los servidores de aplicaciones para Java, además ha alcanzado una alta popularidad entre los desarrolladores de aplicaciones Java, en el mundo *Open Source* y de *software* libre. (Jboss™ Inc., 2005)

Jboss™ es un servidor de aplicaciones J2EE™ de código abierto implementado en Java puro, lo cual permite que sea utilizado en cualquier sistema operativo que lo soporte. Los principales desarrolladores de Jboss™ trabajan para una empresa de servicios: Jboss™ Inc, adquirida por Red Hat en abril del 2006, y fundada por *Marc Fleury*, el creador de la primera versión de Jboss™.

En el libro "*The Jboss™ 4 Application Server Guide*" se exponen las principales características de Jboss™:

- Producto de licencia de código abierto sin costo adicional.
- Cumple los estándares.
- Confiable a nivel de empresa.
- Incrustable, orientado a arquitectura de servicios.

- Flexibilidad consistente.
- Servicios del *middleware* para cualquier objeto de Java.
- Ayuda profesional 24x7 de la fuente.
- Soporte completo para JMX. (Jboss™ Inc., 2005)

Jboss™ implementa todo el paquete de servicios de J2EE™.

Entre los servicios brindados por este servidor están:

- CMP (*Container-Managed Persistence*): Este servicio es el encargado de establecer toda la comunicación entre el *entity bean* y el ambiente donde se está realizando la persistencia. El desarrollador debe decidir si utiliza este servicio o desarrolla por si mismo los métodos que definen la persistencia.
- CMR (*Container-Managed Relationships*): Son relaciones que se establecen entre los e*ntity beans* que pertenecen a un mismo contenedor. Estas relaciones pueden ser de uno a uno, uno a muchos o muchos a muchos; y pueden clasificarse en unidireccionales o bidireccionales.

Otros servicios brindados son:

- Servicio de seguridad por roles
- Servicio de Transacciones

1.2.9 Sistema gestor de base de datos

Actualmente son muchas las aplicaciones que requieren acceder a datos y se necesita de un medio para lograrlo. Cuando el volumen de la información es muy grande este medio de acceso debe ser altamente eficaz. Es aquí donde aparecen los Sistemas Gestores de Bases de Datos, los cuales proporcionan una interfaz entre las aplicaciones y el sistema operativo, permitiendo que el acceso a los datos se realice de una forma más eficiente, más fácil de interpretar y sobre todo, más segura.

Los Sistemas Gestores de Bases de Datos cumplen con determinados objetivos:

- Abstracción de la información: Los usuarios de los SGBD no tienen detalles acerca del almacenamiento físico de los datos. No determina que una base de datos ocupe uno o cientos de archivos, este hecho se hace transparente al usuario. Así se definen varios niveles de abstracción.
- Independencia: La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- Redundancia mínima: Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria o al menos conveniente la aparición de redundancias.
- Consistencia: En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
- Seguridad: La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero sin la responsabilidad necesaria. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de acceso.
- Integridad: Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de *hardware*, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

- Respaldo y recuperación: Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
- Control de la concurrencia: En la mayoría de los entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.
- Tiempo de respuesta: Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

Entre los Sistemas Gestores de Bases de Datos (SGBD) libres se encuentran MySQL y PostgreSQL. Este último es el que utilizamos en nuestra aplicación y explicamos a continuación.

1.2.9.1 PostgreSQL

PostgreSQL es un sistema gestor de bases de datos relacional, de código abierto, descendiente de Postgres que fue desarrollado en la Universidad de Berkeley.

En "*PostgreSQL 8.0.0beta5 Documentation*" se presentan algunas de las características siguientes:

- Consultas complejas
- Llaves extranjeras
- Disparadores
- Vistas
- Integridad transaccional

- Acceso concurrente multiversión (The PostgreSQL Global Development Group, 2004)

Además de ser de código abierto, es libre, cumpliendo las nuevas estrategias de nuestro país acerca del *software* libre. Por el hecho de ser de los mejores gestores de bases de datos además de las ventajas antes expuestas, fue elegido para el desarrollo del sistema GNU y en particular para nuestra aplicación.

1.2.10 Estrategias de desarrollo

Una estrategia de desarrollo, no es más que un conjunto de procedimientos y actividades que se planifican para crear un software o sistema. Define por tanto, el camino a seguir para llevar a cabo una investigación y la forma de enfrentar el problema en la práctica.

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

1.2.10.1 MDA (Model Driven Architecture)

El *Object Management Group* (OMG) es un consorcio de empresas de informática creado en el año 1990, con el objetivo de potenciar el desarrollo de aplicaciones orientadas a objetos y distribuidas. Para ello, desde un principio se prestó especial atención al problema de la interoperatividad e integración de sistemas, lo que ha llevado al OMG a definir numerosas especificaciones y estándares como CORBA, UML, MOF, XMI y CWM. En el año 2001 el OMG crea el *framework* MDA para el desarrollo de aplicaciones. (Bohlen et al., 2005)

Con el paradigma MDA se simplifica el desarrollo de una aplicación. Inicialmente en el proceso de la ingeniería de requerimientos el desarrollador lo que tiene son ideas o lo que se llama modelo mental (MM), luego se digitalizan esas ideas en un modelo independiente de la plataforma (PIM), siendo este modelo un mapeo del modelo mental en algún formato como UML.

Esta metodología tiene varias ventajas:

- El desarrollador se centra en la lógica del negocio y no en la plataforma subyacente.
- El PIM puede ser reusado luego, sin estar atado a algún tipo específico de plataforma.
- El PIM es un buen medio para la comunicación entre desarrolladores.

AndroMDA es una herramienta para generar código, va refinando por pasos el PIM en un modelo más específico de la plataforma (PSM), y al final organiza el modelo en código para ser desarrollado manualmente por el programador.

1.2.10.2 La herramienta AndroMDA

AndroMDA es un *framework* de generación de código que sigue el paradigma MDA. Toma generalmente modelos de una herramienta *case* y genera aplicaciones desplegables y otros componentes. Además es de código abierto, permitiendo al usuario hacer modificaciones. (Bohlen et al., 2005)

AndroMDA usa *plugins* (cartuchos y librerías para generación) y puede generar código para diferentes tipos de lenguajes como Java, .Net, HTML, PHP. Sólo es necesario tener el *plugin* adecuado.

El gran propósito de AndroMDA es generar la parte repetitiva, así como la estructura de directorios, como en nuestro caso que es una aplicación J2EE™ con varios paquetes;

también los descriptores de despliegue; y al final es capaz de ensamblar y desplegar la aplicación en un servidor de aplicaciones como el Jboss™. Además, a partir de los diagramas de actividad, es capaz de crear todo un sitio *Web*, siguiendo el patrón de diseño MVC (*Model View Controller*) que se describe a continuación.

1.2.10.3 Patrones de diseño

Un patrón J2EE™ no es más que un estándar que representa una solución experta que proponen un conjunto de desarrolladores de la plataforma, cuyo objetivo principal es proponer estrategias para diseñar un sistema más escalable, mantenible, interoperable y reusable. Un patrón puede tener las siguientes categorías:

- ✓ Diseño
- ✓ Arquitectura
- ✓ Creación
- ✓ Estructural
- ✓ Comportamiento

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

1.2.10.3.1 MVC (Model View Controller)

MVC es un patrón de diseño que separa la aplicación en componentes:

- Modelo
- Vista
- Controlador (Armstrong et al., 2005)

El modelo representa los datos junto con las operaciones que permiten el acceso a ellos, este notifica a la vista cuando existe algún cambio y le permite a ésta consultar por el

estado de los datos. También le permite al controlador acceder a la funcionalidad de la aplicación encapsulada por él.

Las vistas muestran el contenido del modelo, es decir, toman datos del modelo y especifican en qué forma serán mostrados. También pueden pasar al controlador las entradas del usuario.

El controlador define la lógica de la aplicación. Responde a los pedidos de los usuarios y selecciona las vistas que serán presentadas, además convierte los pedidos en acciones que serán ejecutadas por el modelo.

1.2.10.3.2 Patrón singleton

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

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

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

- La propia clase es responsable de crear la única instancia.
- Permite el acceso global a dicha instancia mediante un método de clase.

Declara el constructor de clase como privado para que no sea instanciable directamente.

1.2.10.3.3 Fachada de Sesión

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

El patrón fachada de sesión aplica los beneficios del patrón de fachada tradicional a los EJB™, ocultando por completo el modelo de objetos en el servidor de la capa cliente, teniendo una capa de objetos que brindan servicios (*sessionbean*) como único punto de acceso para el cliente (Figura 1.5). La fachada permite encapsular el flujo y la lógica del negocio de los casos de uso.(Marinescu 2002)

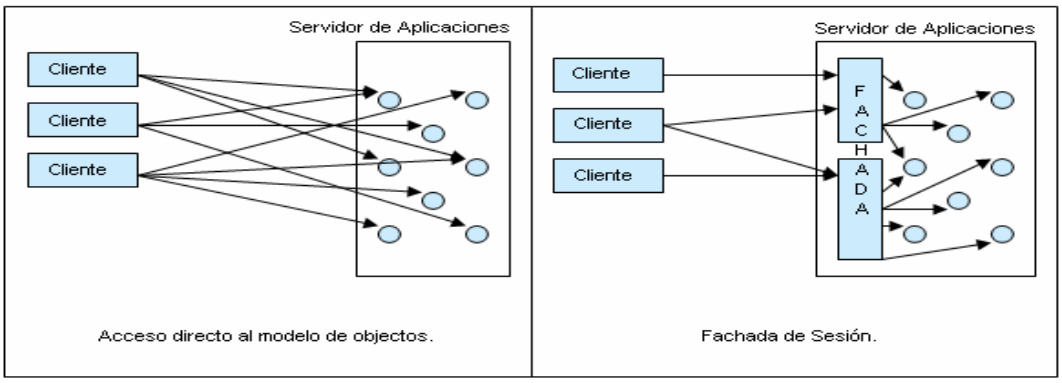


Figura 1.5 Beneficios de la fachada de sesión para la arquitectura.

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

1.2.10.3.4 Value Objects

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

1.2.11 IReport y JasperReports

La herramienta iReport es un constructor/diseñador de informes, visual, poderoso, intuitivo y fácil de usar para JasperReports, y está escrito en Java. Este instrumento permite que los usuarios corrijan visualmente informes complejos con cartas, imágenes, subinformes, etc. iReport está además integrado con JFreeChart, una de las bibliotecas gráficas *Open Source* más difundidas 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" las características más relevantes de iReport son:

- 100% escrito en Java y además *Open Source* y gratuito.
- Maneja el 98% de las etiquetas de JasperReports

- Permite diseñar con sus propias herramientas: rectángulos, líneas, elipses, campos de los *textfields*, cartas, y subreportes.
- Soporta internacionalización nativamente.
- *Browser* de la estructura del documento.
- Recopilador y exportador integrados.
- Soporta JDBC.
- Soporta *JavaBeans*™ como orígenes de datos (estos deben implementar la interfaz JRDataSource).
- Incluye *Wizard's* (asistentes) para crear automáticamente informes.
- Tiene asistentes para generar los subreportes.
- Tiene asistentes para las plantillas.
- Facilidad de instalación. (Herrera, 2005)

JasperReports es una poderosa librería *Open Source* para la elaboración de reportes, escrita completamente en Java y que puede ser utilizada en diferentes aplicaciones Java incluyendo J2EE™ o Aplicaciones *Web*, para generar contenido dinámico. (Danciu, 2005) Este 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 *jasper file* (*.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, por ejemplo una aplicación *Web* generándolos en formato pdf y luego llamarlos en la aplicación. (Toffoli, 2005)

1.3 Conclusiones del Capítulo

En este capítulo se han descrito algunos elementos y conceptos que se tuvieron en cuenta, así como las herramientas fundamentales utilizadas para el desarrollo del sistema, justificando su uso con las ventajas que ofrece cada una de ellas para la solución propuesta. Además se ha hecho una breve síntesis del funcionamiento y la gestión académica del movimiento de Alumnos Ayudantes de la Educación Superior en nuestro país, con el propósito de facilitar la comprensión del presente trabajo.

CAPITULO II.

ASPECTOS METODOLÓGICOS REFERENTES AL DISEÑO E IMPLEMENTACIÓN DEL SISTEMA.

Después de consultar las reglamentaciones generales dadas a conocer a inicios del curso 2007-2008 relacionadas con los Alumnos Ayudantes, se realizó la investigación de los procesos que se realizan en torno a este tema en la Universidad Central "Martha Abreu" de Las Villas. Para la búsqueda de la información necesaria a nuestro sistema, utilizamos la técnica de la entrevista, recogiendo así los datos proporcionados por las secretarias y jefes de departamento de nuestro centro. Con todo esto en nuestras manos, nos dispusimos a obtener una comprensión teórica del problema que fuera capaz de brindarnos el conocimiento necesario para comenzar el desarrollo de la aplicación. Este proceso de automatización de los procesos se concibió en tres etapas fundamentales:

Primera Etapa: Dedicada a la asimilación del entorno y las herramientas de desarrollo.

Segunda Etapa: Concebida para el modelado del sistema.

Tercera Etapa: Enfocada a la implementación de la aplicación sobre estas herramientas.

Hablamos de asimilación – y no de selección – del entorno y de las herramientas, porque ellas han sido en general previamente definidas por el proyecto SISGENU, como se detalla en el epígrafe siguiente.

2.1 Asimilación del entorno y de las herramientas de desarrollo.

Como ya se comentó en el primer capítulo, la dirección de Informatización del Ministerio de Educación Superior (MES), organizó el Taller "Automatización de la Gestión Universitaria" en el cual se acordó el uso de una plataforma libre para el desarrollo de los módulos pertenecientes al Sistema de Control de Estudiantes, y en

particular el módulo nuestro, teniendo en cuenta las ventajas que este tipo de plataformas traería.

Debido a la antigüedad en el mercado, a que está soportado por más de 30 proveedores certificados en el mundo, y a que es más adaptable a los cambios del *hardware,* resultó J2EE™ la plataforma de desarrollo escogida nacionalmente. Como servidor de aplicaciones se decidió utilizar Jboss™, precisamente por ser de todas las variantes libres, la que con más estándares de J2EE™ cumple. Estas definiciones, aplicables todas a nuestro módulo.

Una vez definida la plataforma de trabajo, SIGENU trazó pautas sobre el enfoque para el desarrollo del *software*, de modo que simplificara el proceso de construcción del sistema y fuera menos vulnerable a cambios, por estas razones se escogió el paradigma de desarrollo MDA. Siguiendo la filosofía de *software* libre y que se adecuara al paradigma MDA, se decidió entonces que AndroMDA sería la herramienta más indicada para la generación de aplicaciones J2EE™, decisión también consecuente con las del SIGENU.

Definidas ya las herramientas a utilizar, nos encaminamos a la asimilación de estas técnicas con espíritu creativo, con el propósito de explotarlas al máximo. Una vez vencida esta asimilación estábamos en condiciones de pasar a la segunda etapa de desarrollo de nuestro sistema: la modelación.

2.2 Modelación del sistema.

Debido a que esta nueva versión del Módulo de Control de Alumnos Ayudantes, como ya se ha venido explicando anteriormente, consta de dos ambientes (*Web y Desktop*), se hizo necesaria la modelación de estos de forma independiente, para facilitar sus posteriores modificaciones. Así entonces, si en el futuro las nuevas resoluciones introducen otros conceptos no contemplados en nuestro sistema, se facilita en gran

medida el trabajo para controlar estos nuevos acápites. A partir de la indagación de todos los procesos relacionados con los alumnos ayudantes, se obtuvieron los datos necesarios para el desarrollo de la aplicación: los actores que intervenían en él, así como los casos de uso relacionados con cada uno de ellos.

Antes de pasar al análisis de los casos de uso de cada actor, creemos conveniente mostrar de forma general, el modo en que funcionará nuestra aplicación. A continuación, los diagramas de paquetes y estados respectivamente, dan una idea visual de cómo el sistema opera e interactúa con los usuarios del mismo.

Diagrama de paquetes de nuestro sistema:

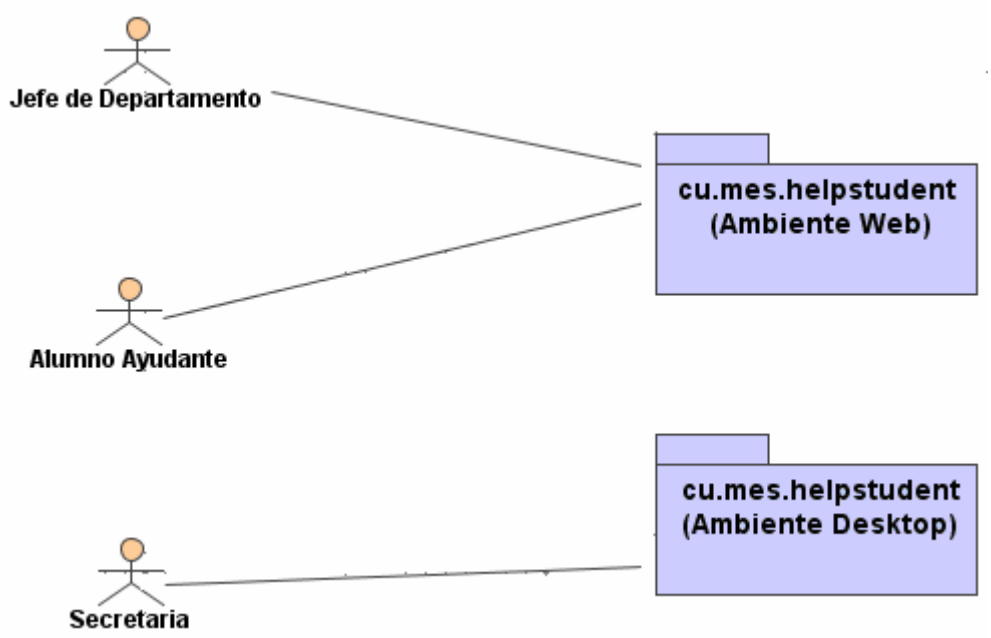


Figura 2.1 Diagrama que muestra de forma general con quién se relaciona cada actor.

Diagrama de estado para el ambiente *Web* del sistema.

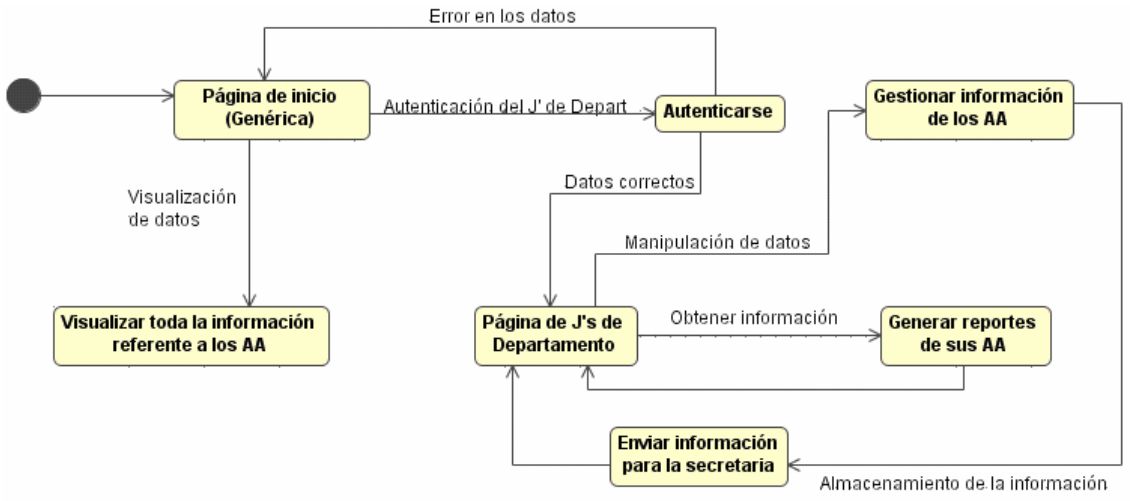


Figura 2.2 Diagrama que muestra el funcionamiento del ambiente *Web*.

Diagrama de estado para el ambiente *Desktop* del sistema.

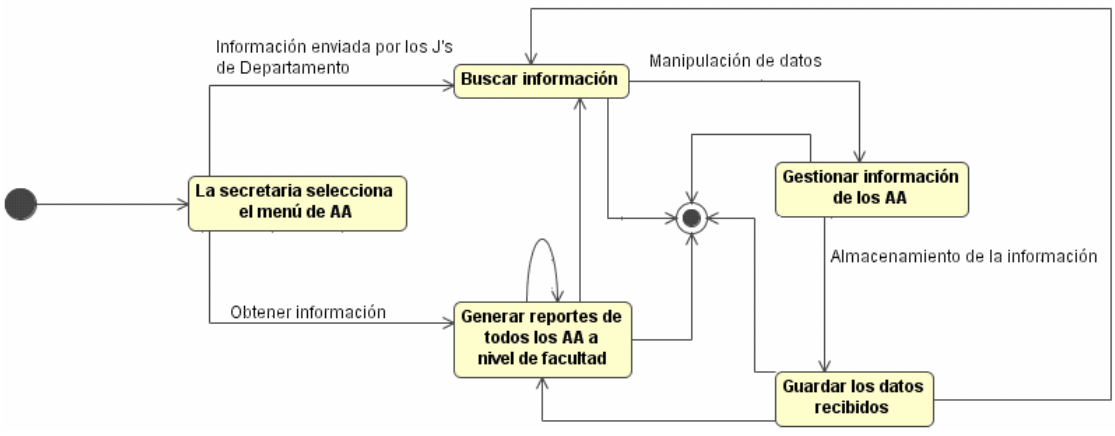


Figura 2.3 Diagrama que muestra el funcionamiento del ambiente *Desktop*.

Debido a que las páginas del ambiente *Web* de nuestro sistema son generadas a partir de los diagramas de actividad asociados a los casos de uso, fue necesario definir un caso de uso "**Inicio**" que es el punto de entrada de nuestra aplicación, de manera que sea la página de bienvenida del sistema. Esta página tiene un carácter genérico en el módulo, pues puede ser accedida por los usuarios desde cualquier parte y sin necesidad de autenticación, o sea, no se necesitan privilegios especiales para acceder a la información que brinda.

A continuación pasamos a la definición de los casos de uso para cada uno de los actores.

- Actores que intervienen en el ambiente Web.
  - ✓ Alumno Ayudante.
  - ✓ Jefe de Departamento.

- Actor que interviene en el ambiente Desktop.
  - ✓ Secretaria.

Nota: Aclarar aquí que todas las búsquedas se hacen en nuestras entidades excepto la que se realiza en el caso de uso "Gestionar Adición", que es sobre entidades de otro proyecto (Minotaur). Además, las operaciones de Adición, Eliminación, Modificación y Ratificación no se efectúan sobre la entidad real de los Alumnos Ayudantes, sino que pasa a una entidad temporal desde donde la Secretaria va a obtener la nueva información de los estudiantes y la operación que tiene que realizar con ellos. De esta forma se garantiza que solo la secretaria sea quien escriba en la entidad real.

2.3 Descripción de los casos de uso.

Teniendo en cuenta la cantidad de casos de uso que manejamos en el sistema, tanto para el ambiente *Web* como para el *Desktop,* solamente mostraremos dos tablas de eventos, pues de forma análoga se pueden interpretar las demás acciones que ejecutan los usuarios. Así entonces, detallamos las tablas de eventos para *Visualizar Horario Docente* e *Introducir Horario Docente,* esta última, accesible únicamente por el jefe de departamento.

Diagrama de casos de uso para los Alumnos Ayudantes.

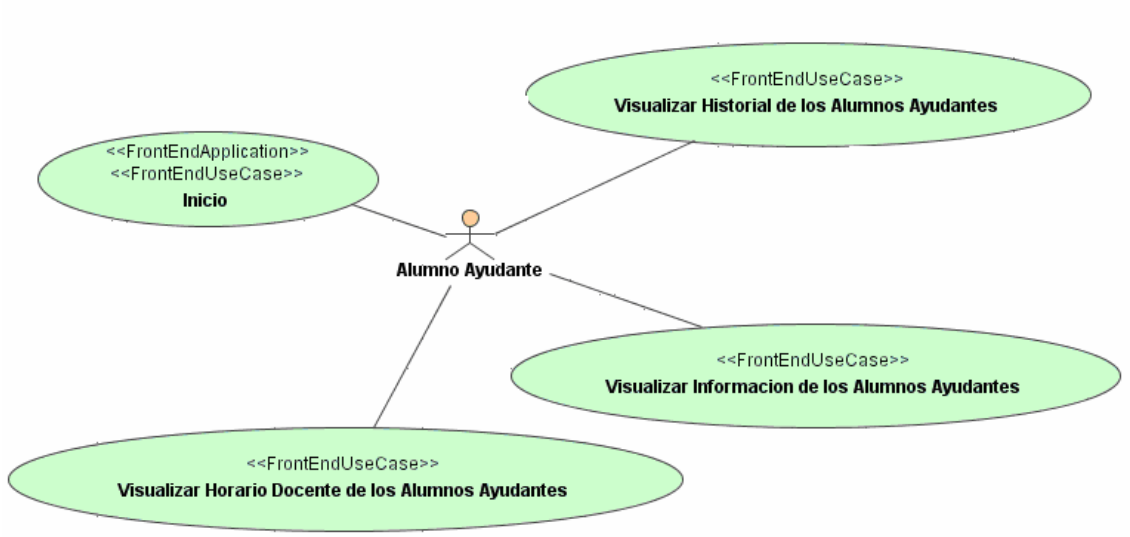


Figura 2.4 Casos de uso para los Alumnos Ayudantes.

Nombre del caso de uso	**Visualizar Información de los Alumnos Ayudantes**
Actores	Alumnos Ayudantes y Jefes de Departamentos. (En general puede ser cualquier usuario que tenga cuenta en el dominio donde se implante el sistema).
Precondiciones	El usuario selecciona la opción de Ver Información de los Alumnos Ayudantes.
Propósito	Mostrar toda la información del estudiante solicitado, referente a su labor como Alumno Ayudante.

Resumen:

El usuario entra al sistema con el propósito de obtener información de un Alumno Ayudante (no necesariamente de él), introduce los datos que se le solicitan y se hace la búsqueda, aparecen entonces todos los que cumplen con los requisitos especificados. Se selecciona ahora el estudiante que se busca y se muestra la información del mismo.

Poscondiciones	Es mostrada la información del Alumno Ayudante solicitado.
Requisitos Especiales	El estudiante solicitado tiene que ser Alumno Ayudante y estar activo.

Nombre del caso de uso	**Visualizar Horario Docente de los Alumnos Ayudantes**
Actores	Alumnos Ayudantes y Jefes de Departamentos. (En general puede ser cualquier usuario que tenga cuenta en el dominio donde se implante el sistema).
Precondiciones	El usuario selecciona la opción de Ver Horario Docente de los Alumnos Ayudantes.
Propósito	Mostrar el horario docente del estudiante solicitado, referente a sus clases de ayudantía.

Resumen:

El usuario entra al sistema con el propósito de obtener la información relacionada con el horario docente de las clases a impartir por un Alumno Ayudante (no necesariamente de él), introduce los datos que se le solicitan y se hace la búsqueda, aparecen entonces todos los que cumplen con los requisitos especificados. Se selecciona ahora el estudiante que se desea y se muestran todos los turnos de clases que debe impartir, incluyendo el día, la sesión y el turno que le corresponde.

Poscondiciones	Es mostrado el horario docente del Alumno Ayudante solicitado.
Requisitos Especiales	El estudiante solicitado tiene que ser Alumno Ayudante y estar activo.

Nombre del caso de uso	**Visualizar Historial de los Alumnos Ayudantes**
Actores	Alumnos Ayudantes y Jefes de Departamentos. (En general puede ser cualquier usuario que tenga cuenta en el dominio donde se implante el sistema).
Precondiciones	El estudiante selecciona la opción de Ver Historial de los Alumnos Ayudantes.
Propósito	Mostrar el historial como Alumno Ayudante del estudiante solicitado.

Resumen:

El usuario entra al sistema con el propósito de ver el historial de un Alumno Ayudante (no necesariamente de él), introduce los datos que se le solicitan y se hace la búsqueda, aparecen entonces todos los que cumplen con los requisitos especificados. Se selecciona ahora el estudiante que se desea y se muestra toda su historia como Alumno Ayudante.

Poscondiciones	Se muestra la historia como Alumno Ayudante del estudiante solicitado.
Requisitos Especiales	El estudiante solicitado tiene que haber sido Alumno Ayudante en algún momento de la carrera y no necesariamente tiene que estar activo.

Secuencia de páginas que se muestran para visualizar el horario de un estudiante.

Figura 2.5 Visualizar Horario Docente

Tabla de eventos que describe la interacción usuario – sistema para el caso de uso mostrado en la figura anterior.

¿Qué hace el usuario?	*¿Cómo responde el sistema?*
Seleccionar *Visualizar Horario Docente*	Se muestra un buscador para que el usuario inserte los datos del estudiante a buscar.
Introduce datos para realizar la búsqueda y da clic en *Buscar* o en *Resetear*	Clic en *Buscar:* se efectúa una búsqueda que devuelve todos los estudiantes que cumplen con los requisitos especificados. Clic en *Resetear:* se limpian todos los campos para que el usuario vuelva a introducir los datos.
Selecciona el estudiante que desea. Para esto puede dar clic en el botón *Ver Horario Docente* o en el *link* sobre el carné de identidad	Se visualiza además del nombre, la asignatura que imparte, la facultad donde debe dar la clase y el área de trabajo del estudiante seleccionado. Debajo aparece su horario docente como alumno ayudante, donde se especifica la fecha, la sesión y el turno que tiene que impartir.
Oprime el botón *Buscar estudiante*	Se brinda la posibilidad de volver a realizar otra búsqueda en caso que el interés del usuario sea este.

Diagramas de casos de uso para los Jefes de Departamentos.

En el caso del Jefe de Departamento, de forma general, tenemos tres grupos de casos de uso, *Visualizar información*, *Realizar Operación* y *Generar Reporte*. A continuación se detalla cada caso de uso de forma independiente.

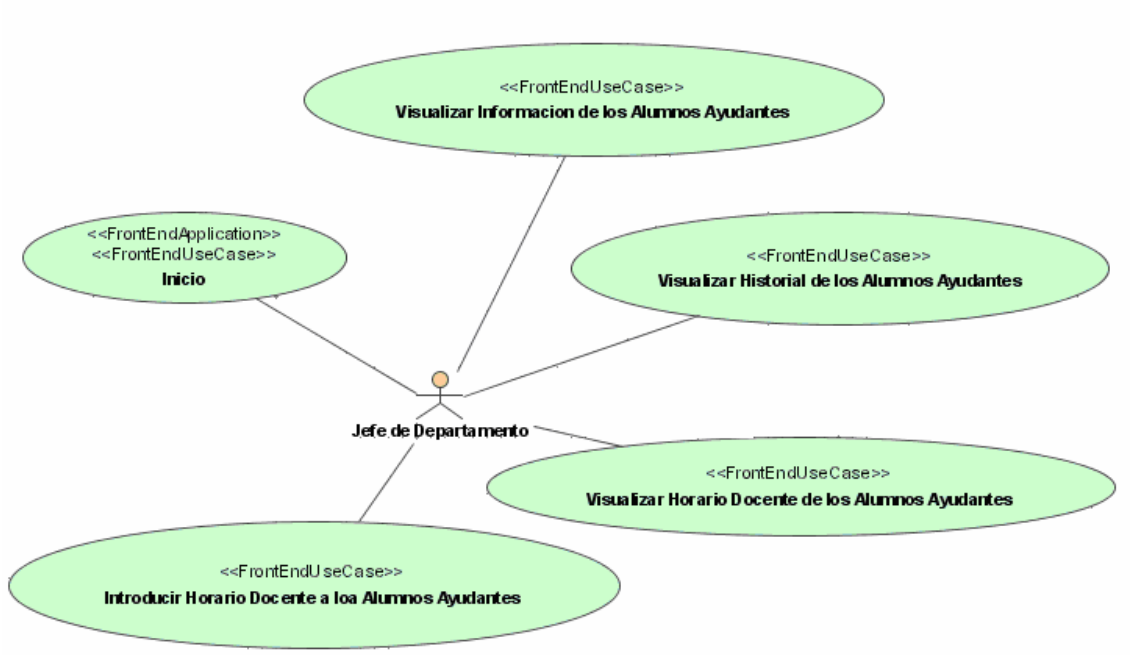


Figura 2.6 Casos de uso para los Jefes de Departamentos (Relacionados con la información de los Alumnos Ayudantes).

Como se puede apreciar, en este primer diagrama de casos de uso para los Jefes de Departamentos, solo existe una variación con respecto al de los Alumnos Ayudantes, la diferencia es que este actor además de poder ver todo lo referente a ellos (AA), puede introducir los horarios docentes a los estudiantes que realizan trabajo de ayudantía. Esto responde al enfoque genérico con que se implementó el ambiente Web de nuestra aplicación y que anteriormente hemos comentado.

Es bueno aclarar también que después que un estudiante entra al sistema nunca más sale, pues la eliminación se maneja como activo o no activo, o sea, la filosofía que se sigue a la hora de eliminar un estudiante como Alumno Ayudante equivale a ponerlo como no activo.

Nombre del caso de uso	**Introducir Horario Docente a los Alumnos Ayudantes**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción de Introducir Horario Docente a los Alumnos Ayudantes.
Propósito	Introducir Horario Docente al Alumno Ayudante que se desee.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de introducir el horario docente de un Alumno Ayudante, introduce los datos que se le solicitan y se hace la búsqueda, se muestran entonces todos los que cumplen con los requisitos especificados. Se selecciona ahora el estudiante al que se le desea introducir el horario y aparecen las opciones necesarias para realizar la operación.

Poscondiciones	Queda recogido en el sistema el horario docente del Alumno Ayudante al que se le introdujo.
Requisitos Especiales	El estudiante solicitado tiene que ser Alumno Ayudante y estar activo.

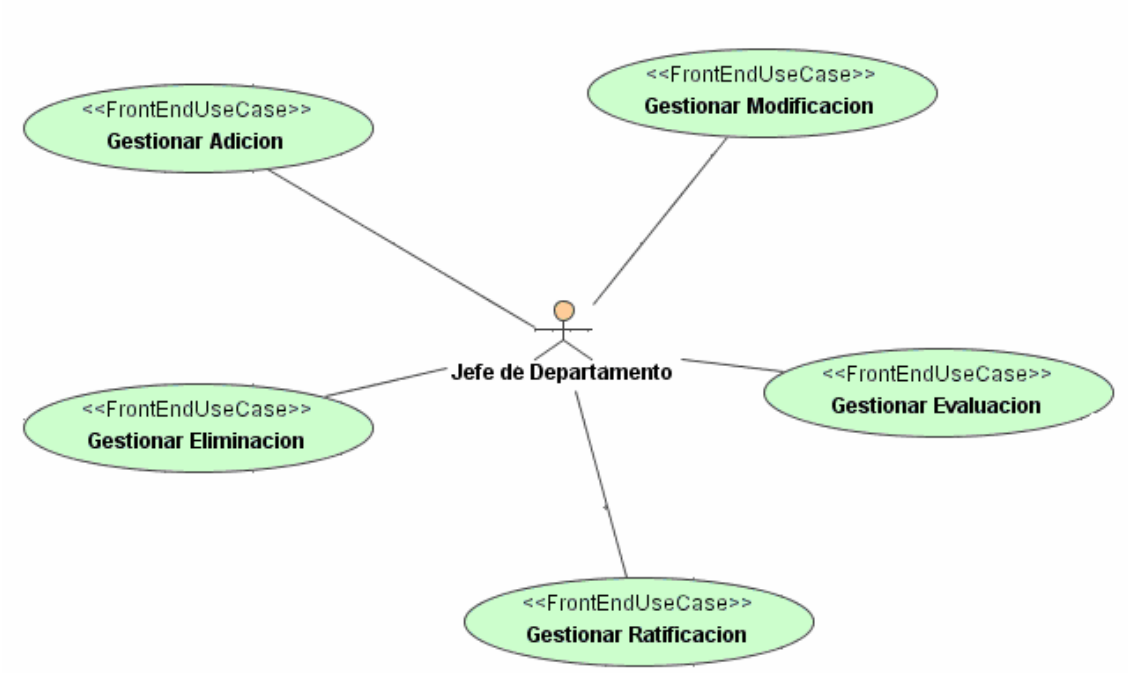


Figura 2.7 Casos de uso para los Jefes de Departamentos (Relacionados con las operaciones que puede efectuar sobre los Alumnos Ayudantes).

Nombre del caso de uso	**Gestionar Adición**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción Gestionar Adición.
Propósito	Introducir un nuevo Alumno Ayudante al sistema.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de adicionar un nuevo Alumno Ayudante, introduce los datos que se le solicitan y se hace la búsqueda, aparecen entonces todos los que cumplen con los requisitos especificados. Se selecciona ahora el estudiante que será Alumno Ayudante, y se especifican todos los requisitos que se piden.

Poscondiciones	Es adicionado en el sistema el nuevo Alumno Ayudante que se seleccionó.
Requisitos Especiales	El estudiante solicitado tiene que estar activo.

Nombre del caso de uso	**Gestionar Eliminación**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción de Gestionar Eliminación.
Propósito	Eliminar un Alumno Ayudante del sistema.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de eliminar un Alumno Ayudante, introduce los datos que se le solicitan y se hace la búsqueda, aparecen entonces todos los que cumplen con los requisitos especificados. Se selecciona ahora el estudiante que será eliminado como Alumno Ayudante, antes de ser eliminado se guarda su historia para ser mostrado en otro momento.

Poscondiciones	Es "eliminado" (no activo) del sistema el Alumno Ayudante que se seleccionó.
Requisitos Especiales	El estudiante solicitado tiene que ser Alumno Ayudante y estar activo.

Nombre del caso de uso	**Gestionar Modificación**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción de Gestionar Modificación.
Propósito	Modificar los datos de un Alumno Ayudante que exista en el sistema.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de modificar los datos de un Alumno Ayudante, introduce los datos que se le solicitan y se hace la búsqueda, aparecen entonces todos los que cumplen con los requisitos especificados. Se selecciona ahora el estudiante al que se le modificará la información como Alumno Ayudante, se editan todos sus datos con el fin de que el Jefe de Departamento pueda cambiar los campos que desee.

Poscondiciones	Son modificados los datos del Alumno Ayudante que se seleccionó.
Requisitos Especiales	El estudiante solicitado tiene que ser Alumno Ayudante y estar activo.

Nombre del caso de uso	**Gestionar Ratificación**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción de Gestionar Ratificación.
Propósito	Ratificar un Alumno Ayudante que exista en el sistema.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de ratificar a un Alumno Ayudante, introduce los datos que se le solicitan y se hace la búsqueda, aparecen entonces todos los que cumplen con los requisitos especificados. Se selecciona ahora el estudiante que será ratificado, o sea, permite ratificar a los alumnos ayudantes al inicio de cada semestre, en este momento se guarda la historia del alumno ayudante con los datos del semestre anterior.

Poscondiciones	Son introducidos los datos del Alumno Ayudante que se seleccionó.
Requisitos Especiales	El estudiante solicitado tiene que ser Alumno Ayudante y estar activo.

Nombre del caso de uso	**Gestionar Evaluación**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción de Gestionar Evaluación.
Propósito	Evaluar un Alumno Ayudante que exista en el sistema.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de evaluar a un Alumno Ayudante, introduce los datos que se le solicitan y se hace la búsqueda, aparecen entonces todos los que cumplen con los requisitos especificados. Se selecciona ahora el estudiante que será evaluado, o sea, permite evaluar al alumno ayudante al final de cada semestre y especificar si es destacado o no, en este momento se guarda la historia del alumno ayudante con los nuevos datos introducidos.

Poscondiciones	Queda registrado en el sistema la evaluación del Alumno Ayudante que se seleccionó y si es destacado o no.
Requisitos Especiales	El estudiante solicitado tiene que ser Alumno Ayudante y estar activo.

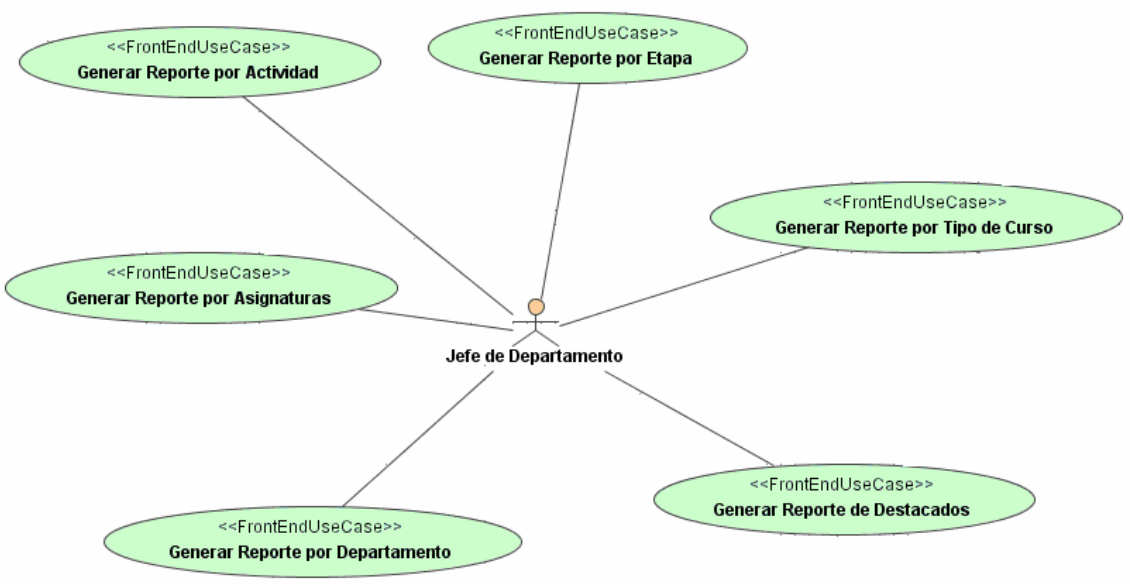


Figura 2.8 Casos de uso para los Jefes de Departamentos (Relacionados con la Generación de reportes de los Alumnos Ayudantes).

Nombre del caso de uso	**Generar Reporte por Etapa**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción de Generar Reporte por Etapa.
Propósito	Obtener un reporte de todos los Alumnos Ayudantes que pertenecen a una misma etapa de ayudantía.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de obtener un reporte completo de todos los estudiantes que pertenecen a una etapa en específico. Para ello selecciona la etapa de la que desea recibir la información y genera el reporte.

Poscondiciones	Es generado en formato PDF un reporte de los estudiantes que pertenecen a una misma etapa de ayudantía.

Nombre del caso de uso	**Generar Reporte por Tipo de Curso**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción de Generar Reporte por Tipo de Curso.
Propósito	Obtener un reporte de todos los Alumnos Ayudantes que pertenecen a un mismo Tipo de Curso.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de obtener un reporte completo de todos los estudiantes que pertenecen a un mismo Tipo de Curso. Para ello selecciona el Tipo de Curso del que desea recibir la información y genera el reporte.

Poscondiciones	Es generado en formato PDF un reporte de los estudiantes que pertenecen al mismo Tipo de Curso.

Nombre del caso de uso	**Generar Reporte de Destacados**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción de Generar Reporte de Destacados.
Propósito	Obtener un reporte de todos los Alumnos Ayudantes que son destacados en su labor de ayudantía y que trabajan en su departamento.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de obtener un reporte completo de todos los estudiantes que son destacados y pertenecen a su Facultad. Para ello solo tiene que elegir la opción de genera reporte.

Poscondiciones	Es generado en formato PDF un reporte de los estudiantes que son destacados y pertenecen a la facultad en que trabaja.

Nombre del caso de uso	**Generar Reporte por Departamento**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción de Generar Reporte por Departamento.
Propósito	Obtener un reporte de todos los Alumnos Ayudantes que pertenecen a su departamento.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de obtener un reporte completo de todos los estudiantes que trabajan en su departamento. Para ello solo tiene que elegir la opción de genera reporte.

Poscondiciones	Es generado en formato PDF un reporte de los estudiantes que trabajan en el departamento del usuario (Jefe de Departamento) que está haciendo la petición.

Nombre del caso de uso	**Generar Reporte por Asignatura**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción de Generar Reporte por Asignatura.
Propósito	Obtener un reporte de todos los Alumnos Ayudantes que imparten la misma asignatura dentro de ese departamento.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de obtener un reporte completo de todos los estudiantes que imparten una asignatura en específico. Para ello solo tiene que seleccionar la asignatura de la que quieren obtener la información y elegir la opción de genera reporte.

Poscondiciones	Es generado en formato PDF un reporte de los estudiantes que trabajan impartiendo la asignatura que se seleccionó.

Nombre del caso de uso	**Generar Reporte por Actividad**
Actores	Jefes de Departamentos.
Precondiciones	El Jefe de Departamento se autentica correctamente y selecciona la opción de Generar Reporte por Actividad.
Propósito	Obtener un reporte de todos los Alumnos Ayudantes que realizan la misma actividad en su labor de ayudantía y que son de su departamento.

Resumen:

El Jefe de Departamento entra al sistema con el propósito de obtener un reporte completo de todos los estudiantes realizan una actividad en específico. Para ello solo tiene que seleccionar la actividad de la que quieren obtener la información y elegir la opción de genera reporte.

Poscondiciones	Es generado en formato PDF un reporte de los estudiantes que realizan la actividad que se seleccionó.

Secuencia de páginas que se muestran para insertar el horario docente de un estudiante.

Figura 2.9 Introducir Horario Docente

Tabla de eventos que describe la interacción usuario – sistema para el caso de uso mostrado en la figura anterior.

¿Qué hace el usuario?	*¿Cómo responde el sistema?*
Seleccionar *Introducir Horario Docente*	Se muestra un buscador para que el usuario inserte los datos del estudiante a buscar.
Introduce datos para realizar la búsqueda y da clic en *Buscar* o en *Resetear*	Clic en *Buscar:* se efectúa una búsqueda que devuelve todos los estudiantes que cumplen con los requisitos especificados. Clic en *Resetear:* se limpian todos los campos para que el usuario vuelva a introducir los datos.
Selecciona el estudiante que desea. Para esto puede dar clic en el botón *Gestionar Horario* o en el *link* sobre el carné de identidad	Se visualiza además del nombre, el tutor que lo atiende y el departamento para el que brinda servicios de ayudantía. Debajo aparecen entonces los campos necesarios para introducirle a ese estudiante su horario, o sea, un calendario para seleccionar el día y dos *combos* para especificar la sesión y el turno en que se impartirá la clase.
Oprime el botón *Insertar* o *Resetear*	Clic en *Buscar:* se inserta el turno y se brinda la posibilidad de volver a introducir otro en caso que sea interés del usuario.
	Clic en *Resetear:* se limpian todos los campos para que el usuario vuelva a introducir los datos.

Diagramas de casos de uso para las Secretarias.

Como se muestra a continuación en las figuras 2.10 y 2.11, los casos de uso para la secretaria no difieren mucho de los que están relacionados con los Jefes de Departamento, sin embargo, es de gran importancia hacer la aclaración para que se entienda cuál es la diferencia. Primero, las operaciones que efectúa la secretaria sí son sobre las entidades reales de la base de datos, y es además, la encargada de determinar si al realizar una de ellas, se le incrementa la etapa de ayudantía al estudiante o si mantiene la que tenía. Segundo, los reportes que hace la secretaria son a nivel de

facultad, mientras que los que hace el Jefe de Departamento son únicamente en su departamento. A pesar de esto, los cambios en la lógica del funcionamiento del negocio no es significativa, por lo que solo se entrará en los detalles de Generar Reporte por Facultad, que sí es totalmente de la secretaria.

Como se muestra a continuación, al igual que el Jefe de Departamento, los casos de uso de la secretaria se dividen en dos grupos, *Realizar Operaciones* y *Generar Reportes.*

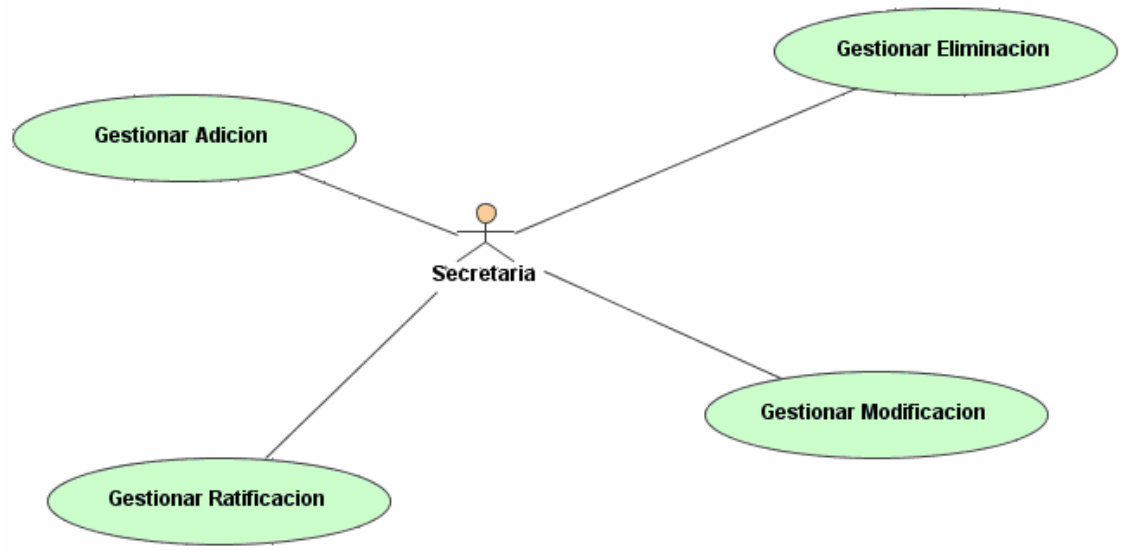


Figura 2.10 Casos de uso para las Secretarias (Relacionados con las operaciones que pueden hacer sobre los Alumnos Ayudantes).

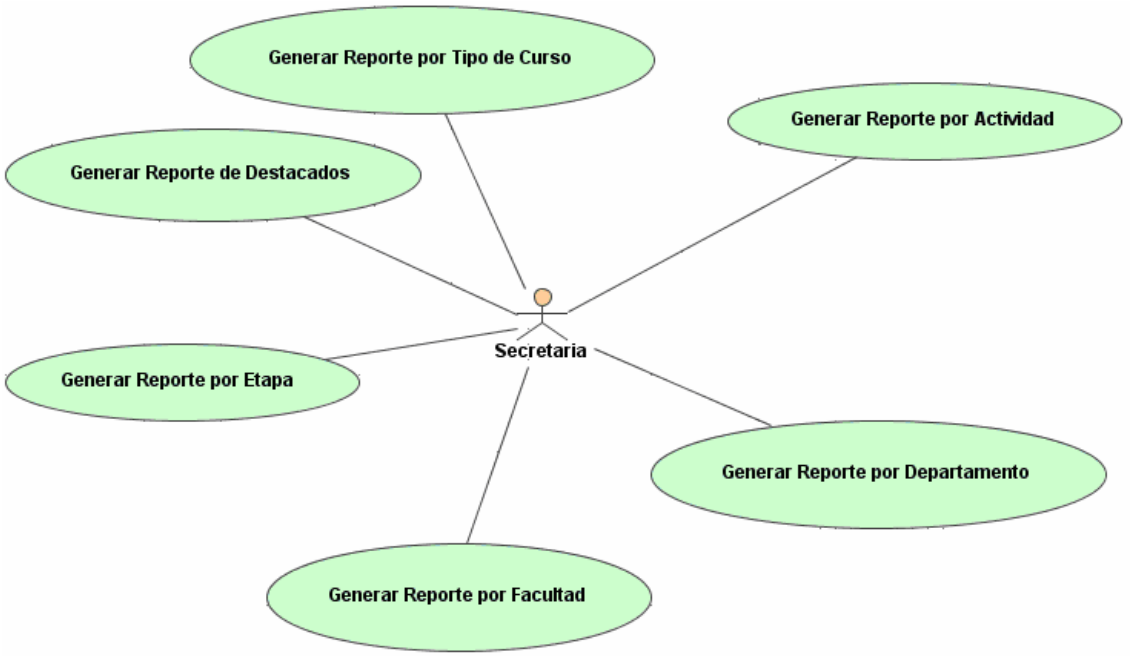


Figura 2.11 Casos de uso para las Secretarias (Relacionados con los reportes que pueden hacer de los Alumnos Ayudantes).

Nombre del caso de uso	**Generar Reporte por Facultad**
Actores	Secretarias.
Precondiciones	La secretaria se autentica correctamente y selecciona la opción de Generar Reporte por Facultad.
Propósito	Obtener un reporte de todos los Alumnos Ayudantes de su facultad.
Resumen:	
La secretaria entra al sistema con el propósito de obtener un reporte completo de todos los estudiantes que son Alumnos Ayudantes en su facultad. Para ello solo tiene que elegir la opción de genera reporte por facultad.	
Poscondiciones	Es generado en formato PDF un reporte de los estudiantes que realizan labor de ayudantía en esa facultad.

Después de hacer el análisis de los actores que intervendrían en el sistema y sus casos de uso correspondientes, pasamos a la definición de los espacios de nombres, que cumpliera con los convenios de nombres establecidos y nos brindara una mejor

organización y claridad del trabajo. A continuación se muestran los espacios de nombres establecidos:

Espacios de nombres para el ambiente Web.

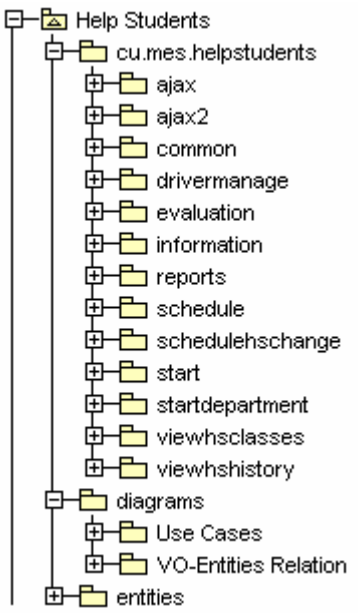


Figura 2.12

cu.mes.helpstudents

Raíz del espacio de nombres para la aplicación en el ambiente Web.

cu.mes.helpstudents.ajax

Paquete que se encarga de la actualización dinámica de las carreras dada la facultad seleccionada.

cu.mes.helpstudents.ajax2

Paquete que se encarga de la actualización dinámica del grupo después de haber seleccionado la facultad y la carrera.

cu.mes.helpstudents.common

En este paquete se encuentra todo lo que es común para las demás operaciones (*value object, enumerativos, etc.*) en el ambiente Web.

cu.mes.helpstudents.drivermanage

Paquete donde se encuentran todas las clases y diagramas para las operaciones de agregar, eliminar, modificar y ratificar los alumnos ayudantes. (Estas operaciones se efectúan sobre la entidad temporal)

cu.mes.helpstudents.evaluation

Clases y diagramas relacionados con la gestión de las evaluaciones de los alumnos ayudantes.

cu.mes.helpstudents.information

Clases y diagramas relacionados con toda la información relacionada con los Alumnos Ayudantes.

cu.mes.helpstudents.reports

Clases y diagramas relacionados con los reportes que pueden hacer los Jefes de Departamento.

cu.mes.helpstudents.schedule

Paquete donde se encuentran todas las clases y diagramas para que un Jefe de Departamento introduzca el horario docente a un Alumno Ayudante.

cu.mes.helpstudents.schedulehschange

Paquete donde se encuentran todas las clases y diagramas para que un Jefe de Departamento modifique el horario docente de un Alumno Ayudante.

cu.mes.helpstudents.start

Paquete que contiene el caso de uso de la página de bienvenida y el diagrama de actividad relacionado con él.

cu.mes.helpstudents.startdepartment

Paquete que se encarga de la autenticación de los jefes de departamento para mostrarle a cada uno su sesión correspondiente.

cu.mes.helpstudents.viewhsclasses

Paquete donde se encuentran todas las clases y diagramas para que un Alumno Ayudante pueda obtener la información de su horario docente.

cu.mes.helpstudents.viewhshistory

Paquete donde se encuentran todas las clases y diagramas para que un estudiante pueda obtener su historial como Alumno Ayudante.

diagrams

Raíz del espacio de nombres para los diagramas.

diagrams.Use Cases

Paquete donde se especifican los actores que intervienes en el ambiente Web del sistema y los casos de uso de cada uno de ellos.

diagrams.VO-Entities

Paquete donde se encuentran los diagramas de clases que relacionan los *value objects* con las *entities*.

entities

Paquete donde se encuentran los diagramas de clases con las relaciones entre nuestras entidades.

Espacios de nombres para el ambiente Desktop.

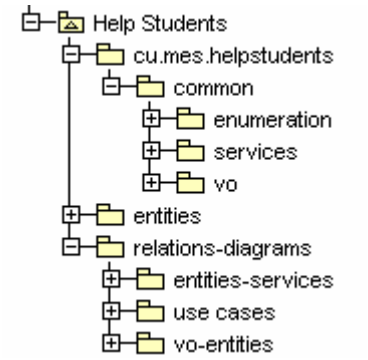


Figura 2.13

cu.mes.helpstudents

Raíz del espacio de nombres para la aplicación en el ambiente Desktop.

cu.mes.helpstudents.common

En este paquete se encuentra todo lo que es común para las demás operaciones en el ambiente Desktop.

cu.mes.helpstudents.common.enumeration

En este paquete se encuentran todas las clases que tienen estereotipo *enumerative.*

cu.mes.helpstudents.common.services

En este paquete se encuentran todas las clases que tienen estereotipo *service.*

cu.mes.helpstudents.common.vo

En este paquete se encuentran todas las clases que tienen estereotipo *value object.*

entities

Aquí se encuentran todas las clases y diagramas que están relacionados con las entidades de nuestro sistema. (Como es lógico tienen que concordar con las clases y diagramas del ambiente *Web*)

relations-diagrams

Paquete donde se encuentran todos los diagramas de clases y casos de uso.

relations-diagrams.entities-services

Paquete donde se especifican todas las relaciones entre las entidades y los servicios.

relations-diagrams.use cases

Paquete donde se especifica el actor (Secretaria) que interviene en el ambiente *Desktop* de nuestro sistema y sus casos de uso.

relations-diagrams.vo-entities

Paquete donde se especifican todas las relaciones entre las entidades y los *value objects*.

Teniendo en cuenta las especificaciones y normas del MES se incorporaron al sistema una serie de entidades y de codificadores ya predefinidos en otros módulos y se definieron las entidades propias de nuestro sistema. Debido a la necesidad de independizar estas aplicaciones (Alumnos Ayudantes y Estipendio) de las ya implementadas, y con el propósito de proporcionar una mayor facilidad a la hora de sus posteriores mantenimientos e implantaciones, nos dimos a la tarea de crear un nuevo proyecto (***aminotaur*** → interpretar como acceso a Minotaur) que funcionara como capa intermedia, y fuera capaz de comunicar nuestra capa de negocio con el módulo de Minotaur, así entonces, a este nuevo proyecto le creamos un ***aminotaur-1.0.jar*** para incluirlo en las aplicaciones de ***helpstudents*** y ***stipend***. De esta forma ganamos en funcionalidad y garantizamos que con solo recompilar estos módulos se puedan hacer

las modificaciones deseadas. Seguidamente damos más detalles para un mejor entendimiento.

Codificadores que se utilizaron de otros módulos (accedidos por ***aminotaur***):

Course: Especifica el curso en que se encuentra el alumno.

Figura 2.14 Codificador del Curso.

Valores Posibles: 2001-2002, 2002-2003, 2003-2004 y así sucesivamente.

CourseType: Especifica el tipo de curso.

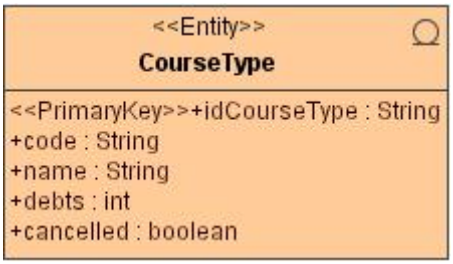


Figura 2.15 Codificadores del Tipo de Curso.

Valores Posibles: diurno, vespertino nocturno, por encuentros, enseñanza a distancia, continuidad de estudios.

El codificador definido por nosotros es:

Activity: Especifica el tipo de actividad realizada por el alumno ayudante.

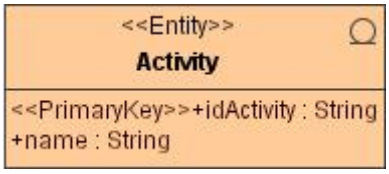


Figura 2.16 Codificadores de la Actividad.

Valores posibles: docencia, investigación y servicio.

A continuación pasamos a describir todas las entidades utilizadas en el sistema. Comenzaremos entonces por las definidas por nosotros. Como ya se explicó anteriormente, nuestras entidades son totalmente independientes a las de otros módulos, por tanto, sus relaciones solo se establecen con las que están en nuestra capa de negocio. Para comunicarse con las demás lo logran mediante las consultas que implementa ***aminotaur***. Es bueno aclarar también que los métodos que aparecen en nuestras entidades y comienzan con *@find…* son las consultas necesarias para acceder a los datos de nuestras tablas para ser manipulados en algún momento.

Datos de los alumnos ayudantes: La principal entidad y el centro de nuestro sistema es *HelpStudent* en la cual se definen y controlan una serie de datos de los alumnos ayudantes.

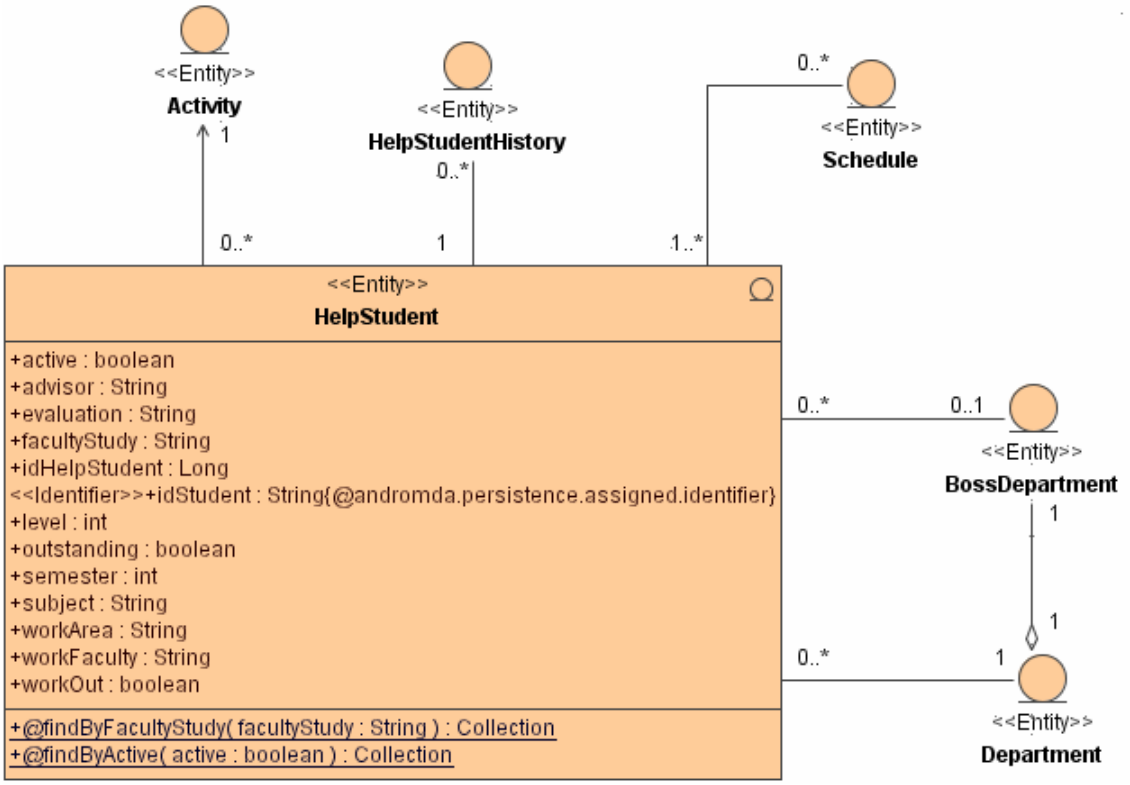


Figura 2.17 Definición de un alumno ayudante.

Atributos	Tipos	Descripción
idStudent	*String*	Identificador del alumno ayudante, mediante este campo se establecen las relaciones.
semester	*int*	Semestre que está cursando el alumno ayudante.
level	*int*	Etapa en la que se encuentra el alumno ayudante.
active	*boolean*	Indica si el alumno ayudante está activo o no.
outstanding	*boolean*	Indica si el estudiante es destacado.
workArea	*String*	Área de trabajo del estudiante, solamente tiene sentido en caso de que la actividad no sea docencia.
advisor	*String*	Profesor que atiende al estudiante.
workOut	*String*	Indica si el estudiante trabaja fuera de la universidad o no.
workFaculty	*String*	Indica la facultad en que trabaja el estudiante.
subject	*String*	Asignatura que asignatura imparte el Alumno Ayudante.
evaluation	*String*	Evaluación del estudiante como Alumno Ayudante.
facultyStudy	*String*	Facultad en la que estudia el estudiante.
idHelpStudent	*Long*	Número de carné de identidad de estudiante.

Historial de los alumnos ayudantes: Entidad en la cual se almacena toda la historia del alumno ayudante durante su labor. Mantiene una relación directa con la entidad *helpstudent* de donde se obtienen los datos a la hora de ser guardada la historia.

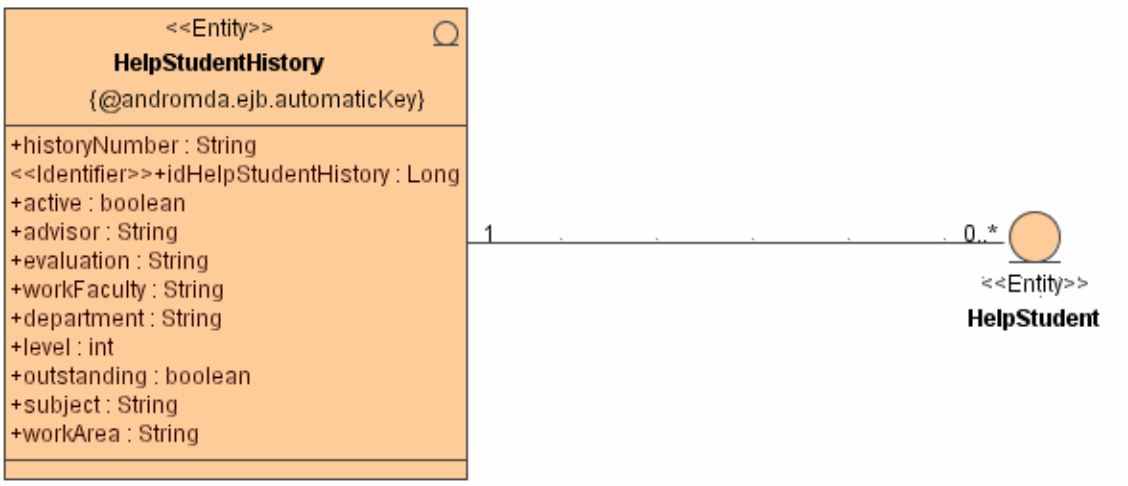


Figura 2.18 Definición de la historia de un alumno ayudante.

Atributos	Tipos	Descripción
idHelpStudentHistory	*Long*	Identificador de la historia del alumno ayudante, mediante este campo se establecen las relaciones.
historyNumber	*String*	Número del historial de cada estudiante.
level	*int*	Etapa del alumno ayudante.
active	*boolean*	Define si el historial es creado porque el estudiante causó baja del movimiento de alumnos ayudantes o no.
department	*String*	Departamento por el cual fue atendido el alumno ayudante.
advisor	*String*	Profesor que estuvo al frente del alumno ayudante.
workFaculty	*String*	Facultad donde trabajó el alumno ayudante.
subject	*String*	Asignatura que impartió el alumno ayudante.
evaluation	*String*	Evaluación obtenida por el alumno ayudante.
outstanding	*boolean*	Indica si fue o no destacado.
workArea	*String*	Área de trabajo del estudiante.

Datos de los departamentos: Entidad donde se controlan los datos del departamento.

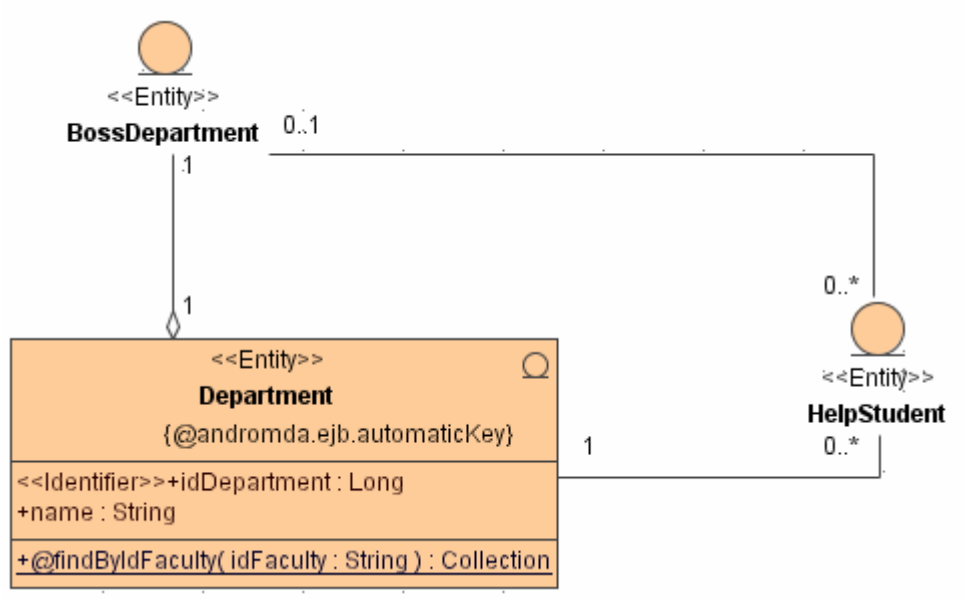


Figura 2.19 Definición de un departamento.

Atributos	Tipos	Descripción
idDepartment	*Long*	Identificador del departamento, mediante este campo se establecen las relaciones.
name	*String*	Nombre del departamento.

Datos de los jefes de departamento: Entidad donde se controlan los datos de los jefes de departamento.

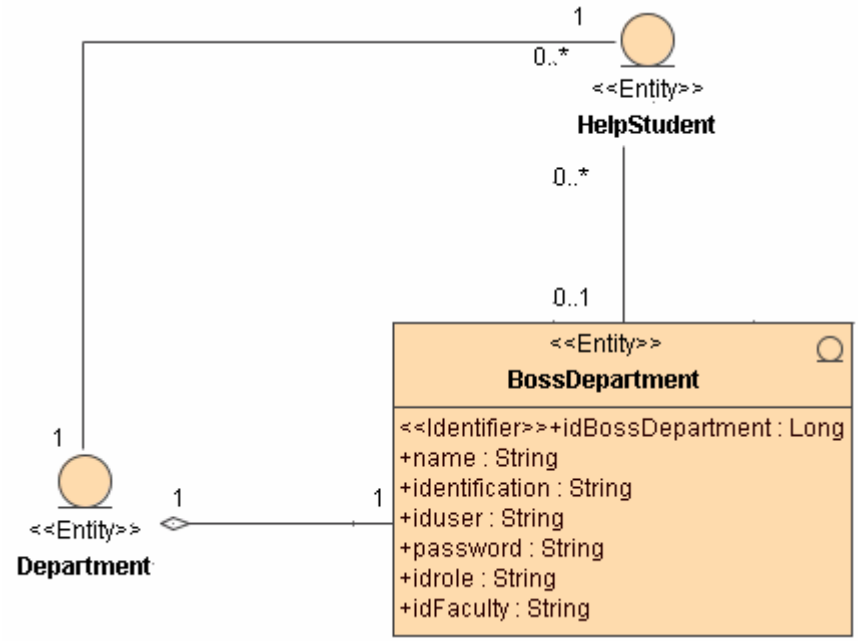


Figura 2.20 Definición de un jefe de departamento.

Atributos	Tipos	Descripción
idBossDepartment	*Long*	Identificador del jefe de departamento, mediante este campo se establecen las relaciones.
name	*String*	Nombre del jefe de departamento.
identification	*String*	Carné de identidad del jefe de departamento.
iduser	*String*	Nombre de usuario del jefe de departamento para entrar al sistema.
password	*String*	Contraseña del jefe de departamento para entrar al sistema.
idrole	*String*	Privilegios de cada usuario en el sistema.
idFaculty	*String*	Facultad a la que pertenece el jefe de departamento.

Horario docente de los Alumnos Ayudantes: Entidad que controla y almacena el horario docente de todos los alumnos ayudantes.

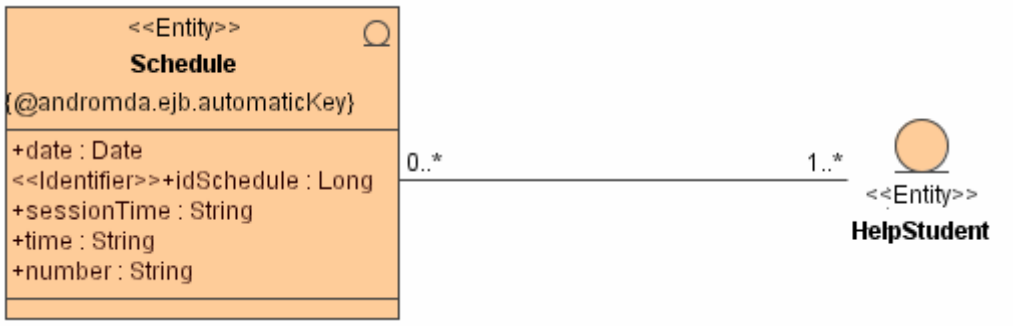


Figura 2.21 Definición de un horario docente.

Atributos	Tipos	Descripción
idSchedule	*Date*	Identificador de un horario docente.
date	*Long*	Día de clase de un alumno ayudante.
sessionTime	*String*	Sesión del día en que es la clase.
time	*String*	Turno en el que toca la clase.
number	*String*	Número de horario para cada estudiante.

Entidad temporal por la que pasan los Alumnos Ayudantes: Entidad en la que insertan los jefes de departamento a los alumnos ayudantes especificando cuál es la operación que se quiere hacer sobre ellos. Luego la secretaria los inserta en *HelpStudent* realizando la operación especificada.

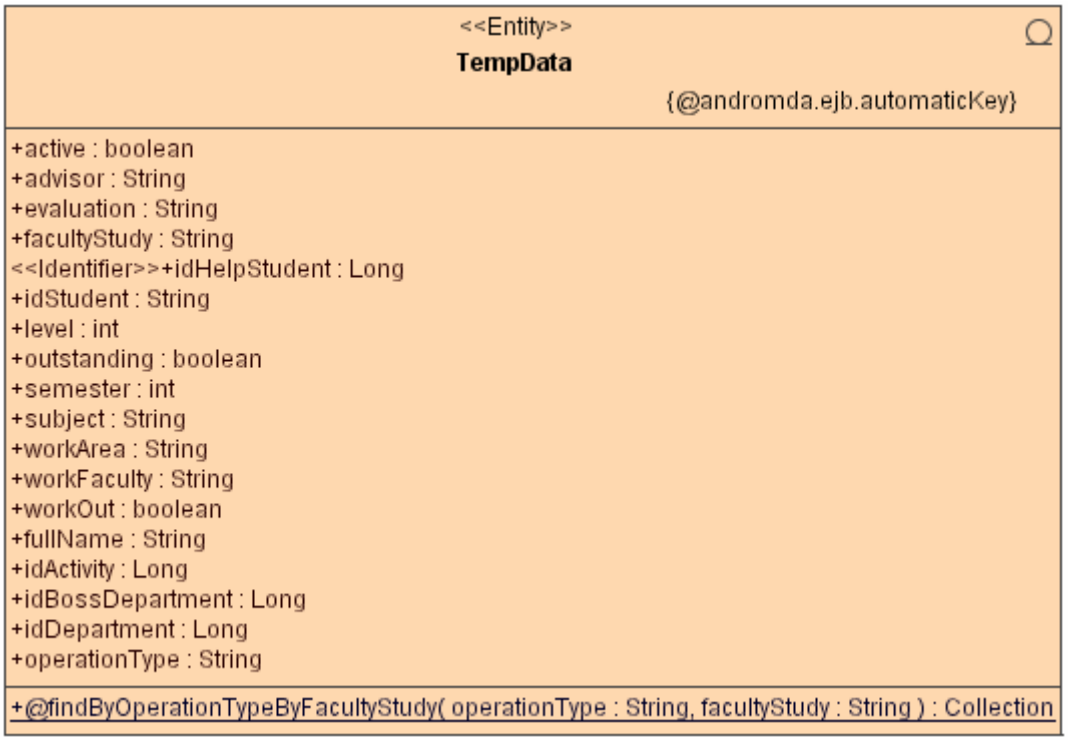


Figura 2.22 Definición de los datos de un Alumno Ayudante y de la operación a realizar por la secretaria con este estudiante.

Nota: Se guardan los datos que después serán insertados en la entidad de los alumnos ayudantes por lo que solo se detallan los atributos que no se especificaron en la entidad de *HelpStudent.*

Campo	Descripción
fullName	Nombre completo del alumno ayudante.
idActivity	Actividad que realiza o realizará el estudiante.
idBossDepartment	Jefe de departamento que o lo atiende o atenderá.
idDepartment	Departamento al que pertenece o pertenecerá.
operationType	Operación que se realizará con el estudiante especificado.

Una vez detalladas las entidades creadas por nosotros, pasamos a dar una breve explicación (sin entrar mucho en las especificaciones) de las utilizadas en nuestro sistema y que ya estaban definidas en otros módulos. Estas entidades son accedidas por ***aminotaur*** para poder manejar su información.

Datos de la facultad:

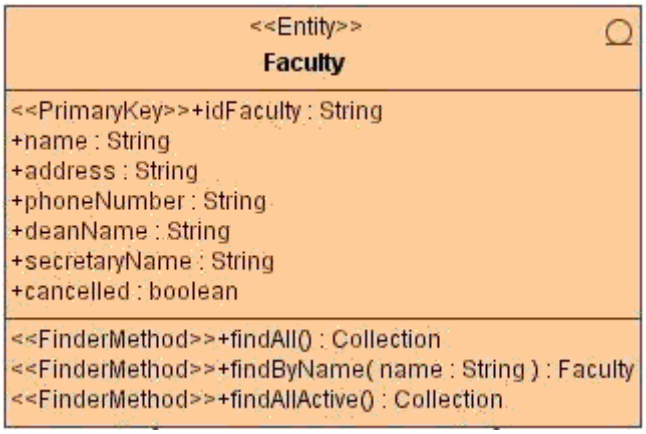


Figura 2.23 Entidad en la que se manejan los datos relacionados con la facultad.

Datos de las asignaturas:

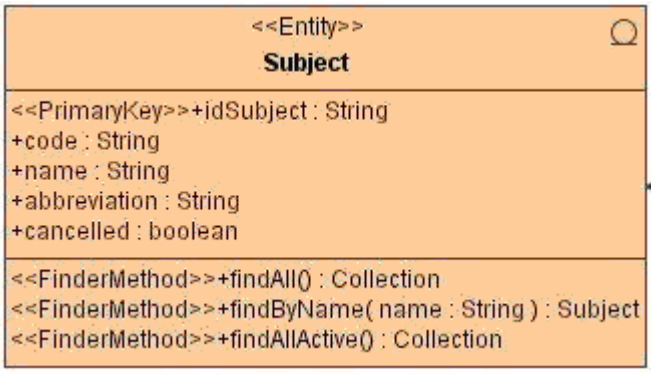


Figura 2.24 Entidad en la que se manejan los datos relacionados con la asignatura.

Datos de los estudiantes:

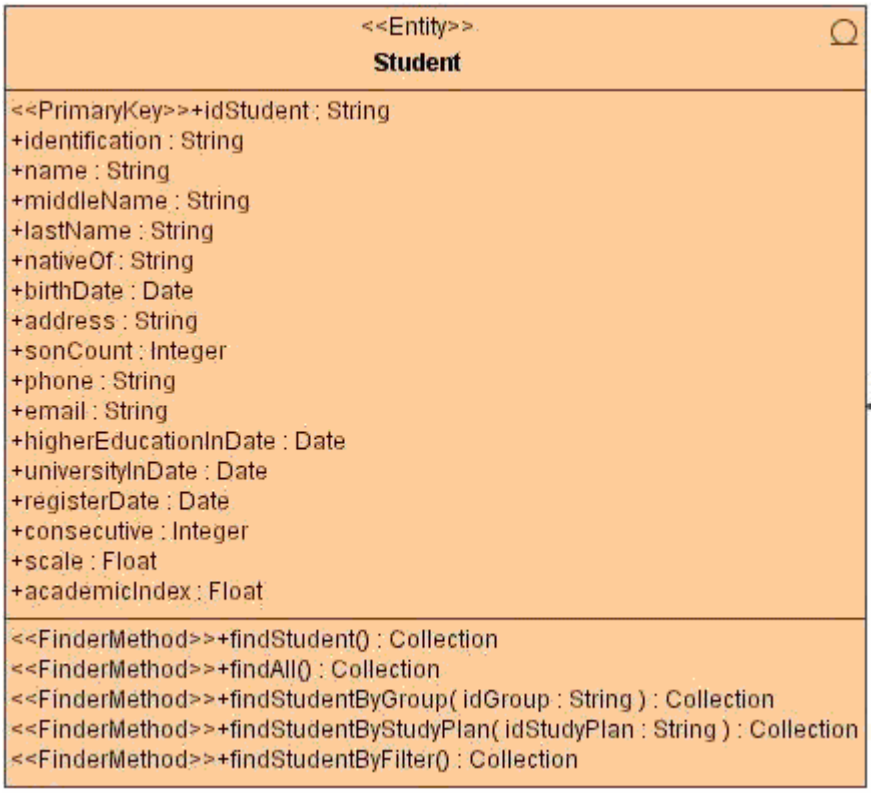


Figura 2.25 Entidad en la que se manejan los datos de todos los estudiantes.

Datos del grupo:

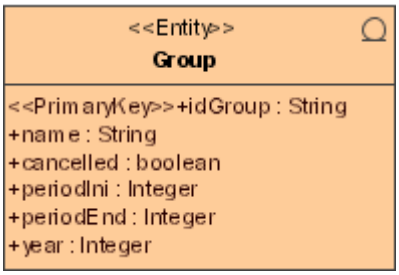


Figura 2.26 Entidad que maneja la información relacionada con el grupo de los estudiantes.

Una vez definidas todas las entidades concluimos el modelado del sistema y ya estábamos preparados para pasar al último paso: la implementación del sistema.

2.4 Implementación del sistema

Una vez terminado el diseño de las entidades persistentes (o *beans*), pasamos a la modelación de las clases que formarían parte de la lógica del negocio.

2.4.1 Capas del sistema

A continuación describiremos cómo quedaron las diferentes capas de nuestra aplicación:

Capa de datos o *Entity Beans*: Está formada por las clases anteriormente comentadas y su función es el almacenamiento o recuperación de información. En las clases pertenecientes a esta capa se han incluido buscadores o *finders* para un mejor rendimiento de la aplicación.

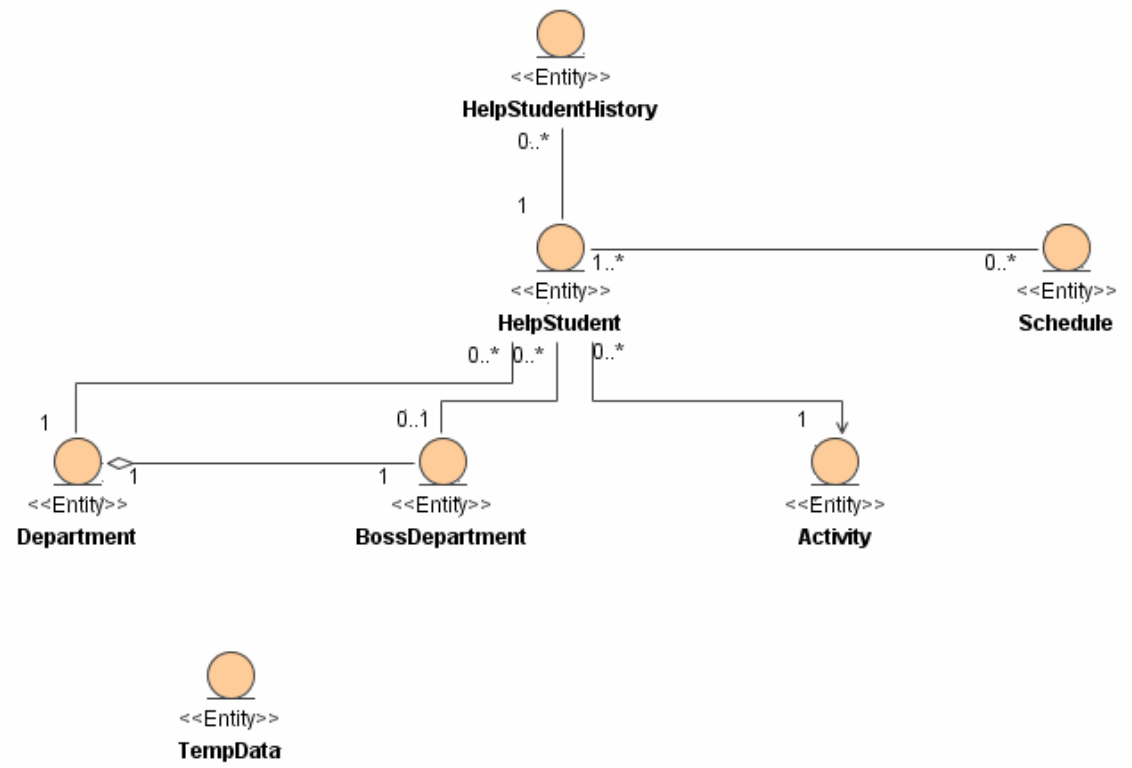


Figura 2.27 Diagrama de clases persistentes.

Capa de gestores de datos: Está conformada por *session beans* los cuales representan una conversación temporal con un cliente. La principal funcionalidad de esta capa es el acceso a los *entity beans*, implementando así el patrón de diseño llamado *session facade*. Cuando el cliente finaliza su ejecución, el *session bean* y sus datos desaparecen.

Los *session beans* implementados en esta capa son los encargados de la recuperación o almacenamiento de la información mediante una transferencia de datos sobre la red. Para optimizar esta transferencia se implementó el patrón de diseño *value object* el cual consiste en tener un objeto que encapsule todos los datos a transmitir a través de la red o entre las capas del sistema.

Todas estas clases y objetos, al igual que sus relaciones fueron modelados, para luego poder ser generados por AndroMDA.

Capa de controladores: En toda aplicación *Web* cada vista proporciona sus propios servicios del sistema, lo que normalmente resulta en duplicación de código. Debido a esto se hace necesario definir una serie de clases llamadas controladores que proporcionen un punto de entrada centralizado que controle y maneje las peticiones *Web*. El controlador también ayuda a reducir la cantidad de código Java. Por cada caso de uso es necesario modelar una clase controladora, la cual contendrá todas las operaciones que serán llamadas en el diagrama de actividad del caso de uso. Es necesario asignar cada controlador a un diagrama de actividad específico.

Por otra parte, existe la necesidad de almacenar algunos datos que podrían ser mostrados en otras vistas y esto se puede lograr con los *session objects*. Estos representan objetos persistentes que se mantienen mientras dure la sesión del usuario, cada *session objects* debe ser asociado a un controlador aunque un controlador podría estar asociado a más de un *session objects.* Como ejemplo ilustrativo mostramos a continuación tres de los diagramas de clases diseñados para la implementación del sistema.

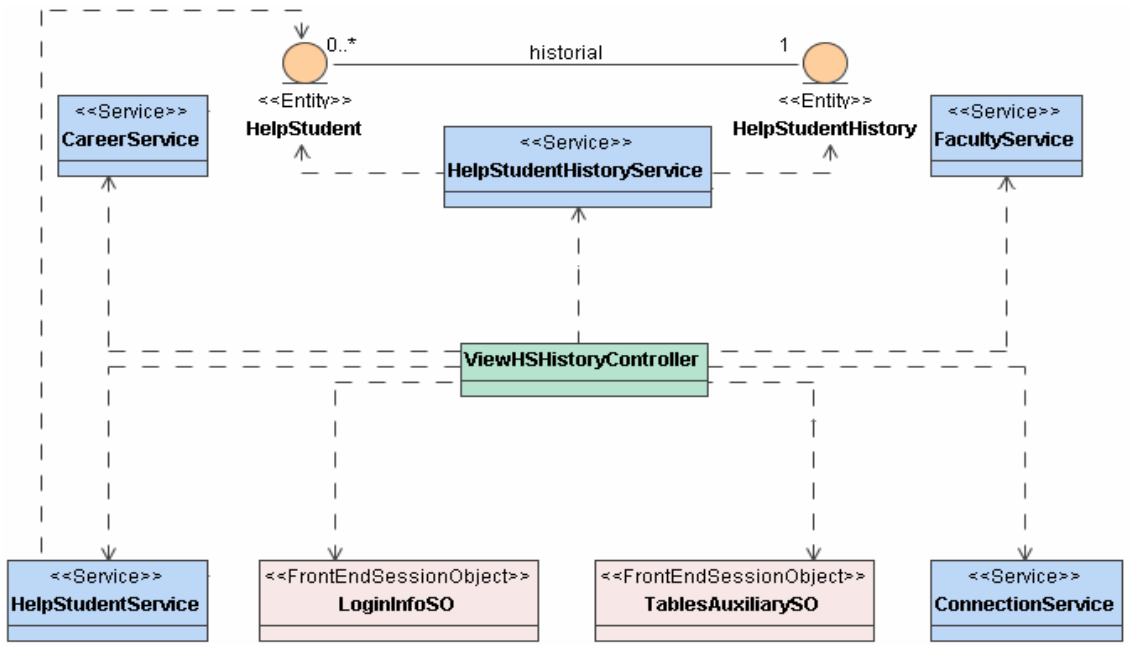


Figura 2.28 Diagrama de clases para visualizar el historial de un Alumno Ayudante.

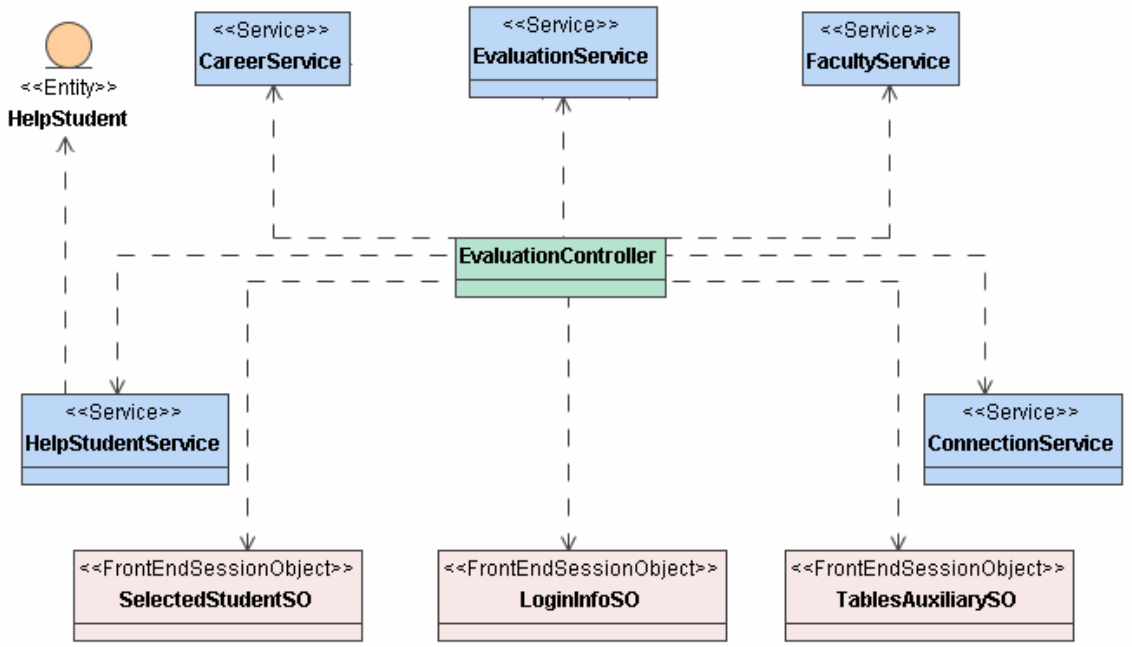


Figura 2.29 Diagrama de clases para introducir la evaluación de un Alumno Ayudante.

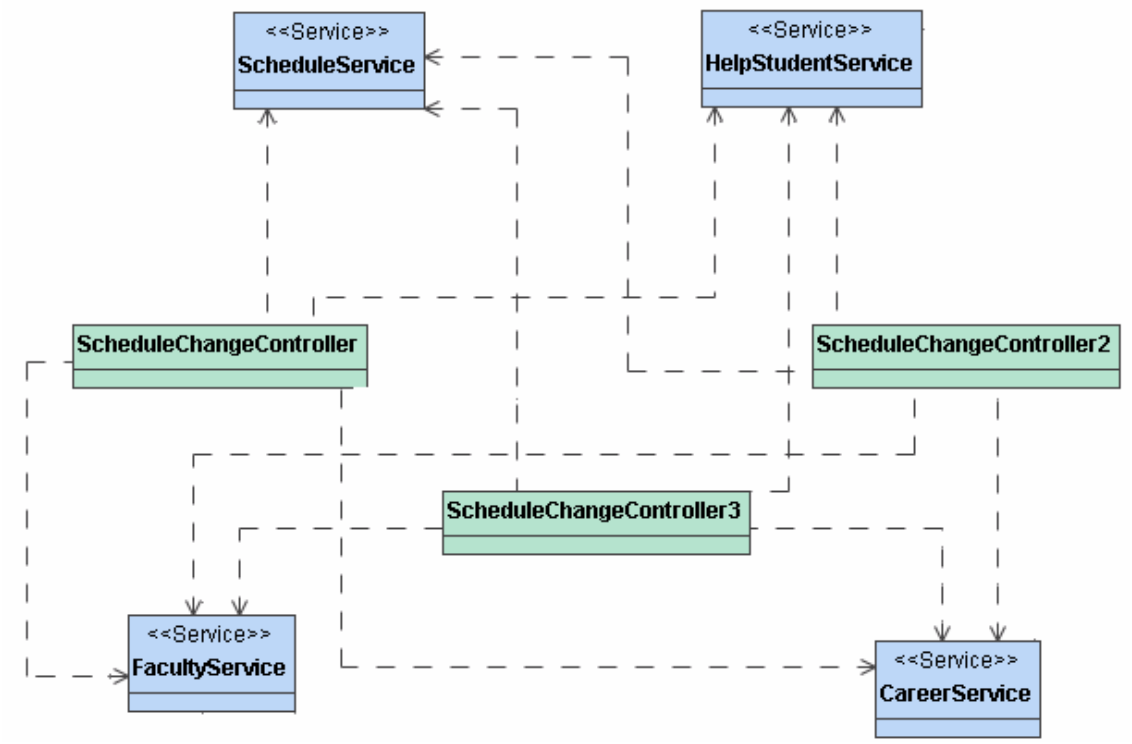


Figura 2.30 Diagrama de clases para modificar el horario docente de un Alumno Ayudante

Capa de vistas para el ambiente *Web* del sistema: En esta capa se modeló cómo sería el flujo del usuario a través de todo el sistema, así como la interacción del mismo en cada una de las vistas. Esta capa se modeló usando diagramas de actividad, donde las actividades en cada uno de los diagramas fueron divididas en dos grupos: las actividades del sistema, y las del usuario, para las cuales AndroMDA generó un fichero jsp.

A continuación presentamos un diagrama de actividad cuya función es permitirle al jefe de departamento insertar un nuevo estudiante, para que luego pase a formar parte del Movimiento de Alumnos Ayudantes cuando la secretaria crea pertinente.

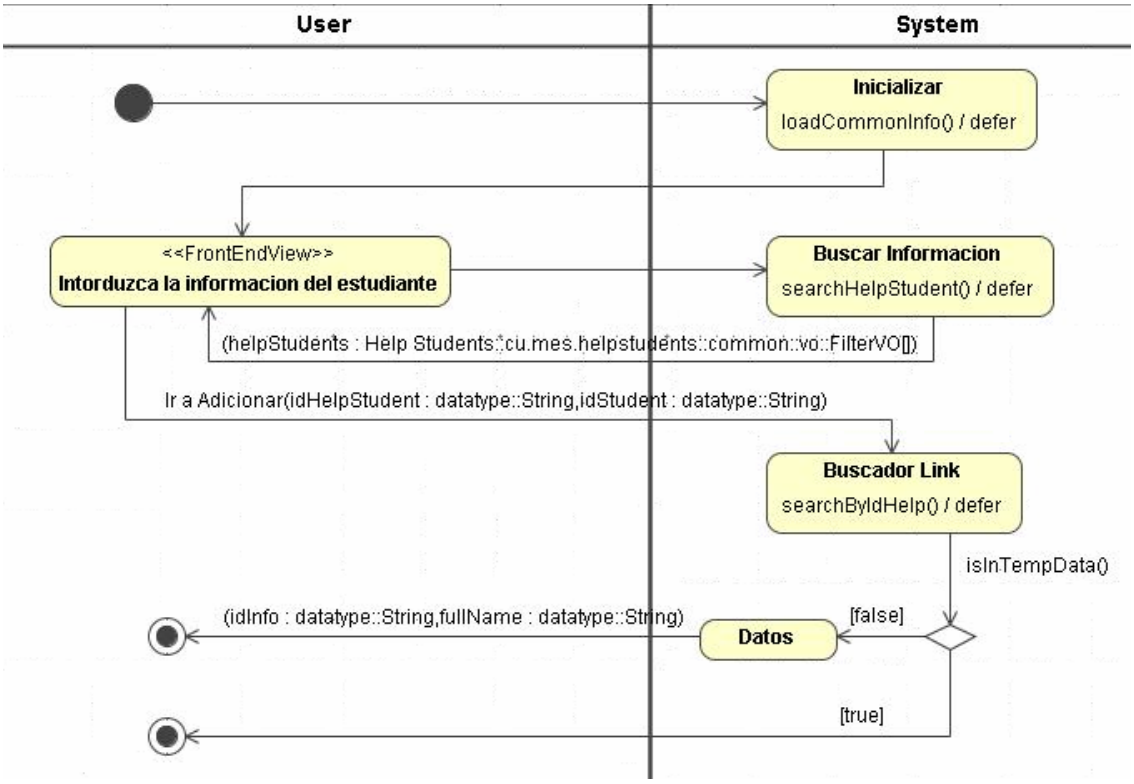


Figura 2.31 Diagrama de actividad para insertar un nuevo Alumno Ayudante.

En las transiciones entre las actividades mostradas en la figura anterior se modelaron los datos que serían mostrados en una actividad del usuario (*page variables***)** y los

datos que serían transmitidos hacia una actividad del sistema (*action parameters***).** De esta forma se le presenta un formulario al Jefe de Departamento para seleccionar el estudiante que luego va a ser adicionado como Alumno Ayudante, se cargan sus datos generales y él introduce los específicos de su labor en la ayudantía.

Sobre el código generado por AndroMDA se implementaron las diferentes funciones definidas en las clases anteriormente expuestas, esto se hizo siguiendo el paradigma de aplicaciones J2EE™.

2.5 Reportes del sistema

Además de modelar las capas del sistema y de implementar las clases pertenecientes a cada capa, se diseñaron los reportes ayudados de la herramienta visual iReport de forma que fueran agradables a la vista del usuario y a la vez de gran utilidad.

Después del diseño, los reportes son compilados por las clases de JasperReports creando un fichero *.jasper* por cada reporte, que luego se muestran en nuestra aplicación. Para mostrar los reportes en el sistema es necesario buscar el *.jasper*, se le pasan los parámetros que fueron modelados en el mismo y se devuelve al cliente el reporte en formato pdf.

2.6 Seguridad propuesta

Otro aspecto a tener en cuenta en la implementación del sistema es la seguridad que va a ser utilizada para ambos módulos.

2.6.1 Ambiente *Web*

Para el manejo de la seguridad en este módulo usamos la plataforma JAAS (*Java Authentication and Authorization Service*) que brinda un mecanismo para que las

aplicaciones J2EE autentifiquen y autoricen grupos de usuarios a usar sus funcionalidades, además de permitir que la aplicación quede independiente de los mecanismos de seguridad usados de forma tal que en algún momento pueda cambiar la forma de autenticación sin tener que cambiar la aplicación.

Con JAAS existían dos variantes para el manejo de la seguridad: declarativa y programática, en nuestro caso se usó la declarativa, que explicaremos a continuación, aunque se utilizaron algunas funcionalidades de la programática para saber los datos del usuario autenticado y poder filtrar la información a manejar por éste.

La seguridad declarativa consiste en expresar la estructura de seguridad de la aplicación en descriptores de despliegue que luego serán usados por el servidor de aplicaciones. AndroMDA cuando genera el código asocia a cada actor un rol, permitiendo a este rol acceder solamente a sus casos de uso, creando automáticamente el descriptor de despliegue de seguridad de nuestra aplicación, de esta forma solo tenemos que especificar en el servidor de aplicaciones la política de autenticación a usar, y este se encargará del resto. En nuestro caso, la autenticación fue realizada como se explicamos ahora. Para especificarle al servidor de aplicaciones la política de autenticación a usar se hizo lo siguiente:

En el fichero *login-config.xml* que se encuentra dentro del Jboss siguiendo el camino *jboss-x.x\server\default\conf* se introdujo el código que mostramos a continuación:

```
<!-- Dominio de seguridad para la aplicación de alumnos ayudantes -->

<application-policy name = "helpstudentsystem">
    <authentication>
        <login-module code ="org.jboss.security.auth.spi.DatabaseServerLoginModule"
                                    flag = "required">
        <module-option name = "unauthenticatedIdentity">guest</module-option>
        <module-option name = "dsJndiName">java:/HelpStudentsDS</module-
```

```
                                    option>
        <module-option name = "principalsQuery">SELECT PASSWORD FROM
                                BOSS_DEPARTMENT WHERE IDUSER=?</module-option>
        <module-option name = "rolesQuery">SELECT IDROLE,'Roles' FROM
                                BOSS_DEPARTMENT WHERE IDUSER=?</module-option>
        </login-module>
    </authentication>
</application-policy>
```

Dentro de *helpstudents\mda\conf\andromda.xml* es necesario agregar la propiedad que se muestra en la siguiente figura:

```
<namespace name="bpm4struts">
    <properties>
        <property name="forms">${maven.andromda.web.generated.dir}</property>
        <property name="pages">${maven.andromda.web.generated.dir}</property>
        <property name="actions">${maven.andromda.web.generated.dir}</property>
        <property name="controllers">${maven.andromda.web.generated.dir}</property>
        <property name="controller-impls">${maven.andromda.web.manual.java.dir}</property>
        <property name="decorators">${maven.andromda.web.manual.java.dir}</property>
        <property name="messages">${maven.andromda.web.generated.dir}/WEB-INF/classes</property>
        <property name="configuration">${maven.andromda.web.generated.dir}/WEB-INF</property>
        <property name="securityRealm">helpstudentsystem</property>
```

Figura 2.32 *Namespace* del *bpm4struts*

Además se creó dentro del *deploy* del Jboss un fichero llamado *helpstudents-ds.xml* especificando el *datasourse* con la conexión a la base de datos, nombre de usuario, etc. (Figura 2.33)

```
<datasources>
  <local-tx-datasource>
    <jndi-name>HelpStudentsDS</jndi-name>
    <connection-url>jdbc:postgresql://localhost:5432/helpstudentsTest</connection-url>
    <driver-class>org.postgresql.Driver</driver-class>
    <user-name>postgres</user-name>
    <password>postgres</password>
        <!-- sql to call when connection is created
        <new-connection-sql>some arbitrary sql</new-connection-sql>
        -->

        <!-- sql to call on an existing pooled connection when it is obtained from pool
        <check-valid-connection-sql>some arbitrary sql</check-valid-connection-sql>
        -->

  </local-tx-datasource>

</datasources>
```

Figura 2.33 *Datasources* de *helpstudents-ds.xml*

2.6.2 Ambiente *Desktop*

Debido a que nuestra aplicación *desktop* será insertada como un *plug-in* en el módulo de *Skeleton* y este proyecto ya trae definida su propia seguridad, el Sistema de Control de Alumnos Ayudantes para la secretaria se subordina a las políticas de seguridad del sistema general. Este mecanismo de acceso al sistema se garantiza mediante la utilización de usuarios y roles asociado a los mismos.

2.7 Despliegue del sistema

Los diagramas de despliegue muestran las relaciones físicas entre los componentes *hardware* y *software* en un sistema terminado, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución (procesos y objetos que se ejecutan en ellos) estarán formados por instancias de los componentes *software* que representan manifestaciones del código en tiempo de ejecución.

Diagrama de despliegue para nuestra aplicación.

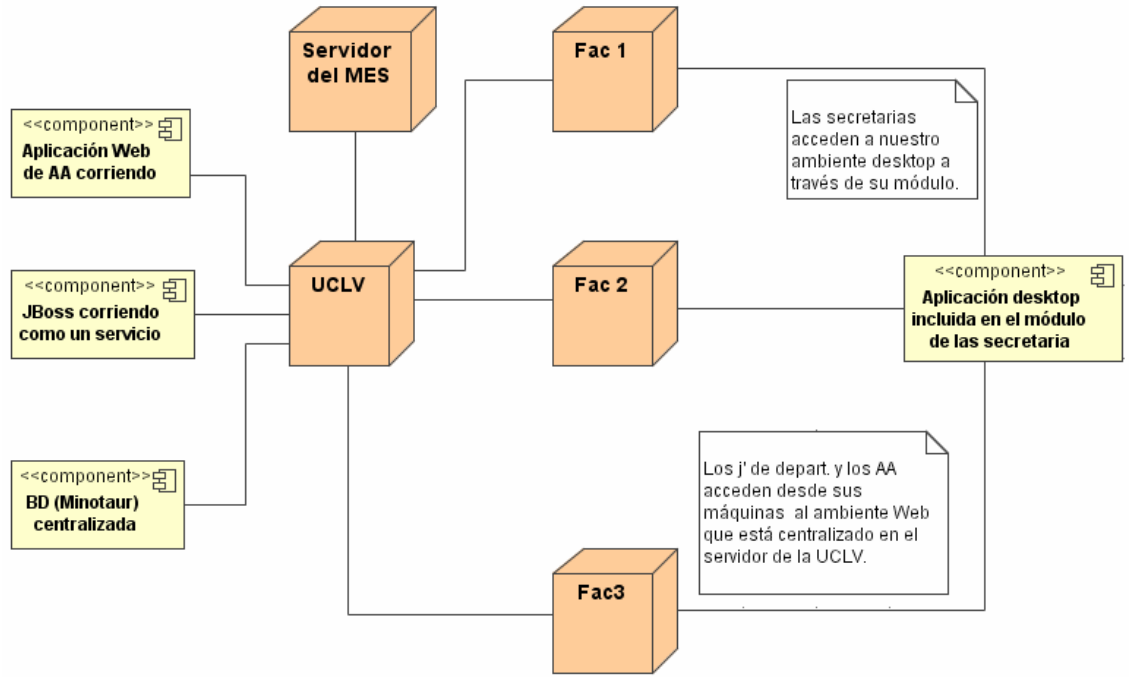


Figura 2.34 Forma en que se efectuará el despliegue de los dos ambientes de la aplicación.

Proponemos pues se haga esta distribución en todos los CES ya que tienen la misma estructura de conectividad.

2.8 Conclusiones del Capítulo

En el presente capítulo se desarrollaron los flujos de trabajo del diseño e implementación del sistema, además se definieron los principales actores y casos de uso de la aplicación, brindándose con ellos una breve descripción e incluyendo tablas de eventos a algunos que se tomaron como ejemplo. Se describieron también todas las entidades utilizadas, tanto las importadas de otros módulos como las definidas por nosotros. Dejamos claro las capas diseñadas en la aplicación, así como los patrones implementados en cada una de ellas y damos una breve explicación de la forma en que se trabaja con los reportes y la seguridad del sistema.

Es válido comentar además, que en el desarrollo del presente trabajo se abarcó en gran medida el contenido de varias de las asignaturas recibidas durante la carrera, algo que sin lugar a dudas contribuyó de forma determinante al rápido desarrollo del mismo.

CAPITULO III.
MANUAL DE USUARIO

El módulo de Control de Alumnos Ayudantes es el único que implementa un ambiente *Web* de los desarrollados hasta ahora por el Sistema de Gestión de la Nueva Universidad (SISGENU). En la primera versión solo se disponía de este ambiente, pero ya en esta nueva etapa de desarrollo, se cuenta además con un módulo *Desktop* que abarca un conjunto de funcionalidades relacionadas con los alumnos ayudantes dentro de las que se incluyen reportes, los cuales garantizan una mejor información y actualización de los datos.

3.1 Elementos de control de acceso al sistema (Ambiente Web)

A diferencia de la versión anterior, el sistema de Control de Alumnos Ayudantes actual tendrá un mecanismo de acceso que actuará de forma genérica para todos los usuarios (ambiente *Web*) sin valorar quién está accediendo al sistema, o esa, la página de inicio puede ser accedida por cualquier tipo de usuario sin necesidad de especificar un *username* y un *password.* Aclarar que esta página es solo para que los Alumnos Ayudantes puedan visualizar toda la información referente a ellos. En caso que sea un Jefe de Departamento el que esté accediendo, entonces sí necesita autenticarse después de estar en esta página para que se le muestren todas las operaciones que puede realizar sobre los Alumnos Ayudantes que pertenecen a su departamento, (en caso que ya estén trabajando en la ayudantía) o los que son estudiantes de su facultad (en caso que vaya a insertar uno nuevo). De esta forma se trata que la información almacenada sea confiable y que se preserve la integridad de la misma.

3.2 Requisitos para la explotación del módulo

Para que el módulo de Control de Alumnos Ayudantes funcione correctamente es necesario que previamente se haya cargado la información correspondiente a los codificadores y los datos de los estudiantes, esta información es almacenada por el Cliente SIGENU que se explota en todas las secretarías docentes. También es necesario que se introduzca la información en la entidad *Activity* de nuestro sistema y que actúa a su vez como otro codificador interno de la aplicación, además de introducir los datos que están relacionados con los departamentos de cada facultad y de sus respectivos jefes.

Nota: Estos requisitos son válidos tanto para el ambiente *Web* como para el ambiente *Desktop.*

3.3 Opciones principales del módulo *Web* del sistema

Con el ambiente *Web* de nuestro sistema se relacionan dos actores, el Alumno Ayudante, y el Jefe de Departamento. Para facilitar la comunicación usuario – sistema, en el presente epígrafe se ofrece una explicación amplia de cómo se debe actuar para lograr un alto grado de beneficio en su uso.

3.3.1 Alumno Ayudante

El Alumno Ayudante solo puede realizar tres operaciones: *visualizar información*, *visualizar historial* y *visualizar horario docente,* las cuales tienen el objetivo de brindar información acerca de la labor que realizan como alumnos ayudantes. Con la característica que pueden ser datos propios, o de otro estudiante que también pertenezca a este movimiento.

Antes de pasar a detallar estos casos de uso, pensamos que es bueno hacer la siguiente aclaración. En toda aplicación siempre se busca un *modus operandi* genérico, de forma tal que cuando un usuario trabaje con el sistema y realice cualquier operación, intuitivamente este sea capaz de manipular otra acción sin necesidad de haberla realizado con anterioridad. Teniendo en cuenta esta filosofía de trabajo, todos los casos de uso en que se necesitan hacer búsquedas, (excepto para generar reportes y gestionar adición) el sistema actúa de igual forma, por tanto solo se explicará esto una vez. A continuación mostramos cómo se debe proceder:

Nota: Los campos que se marcan con un asterisco son obligatorios.

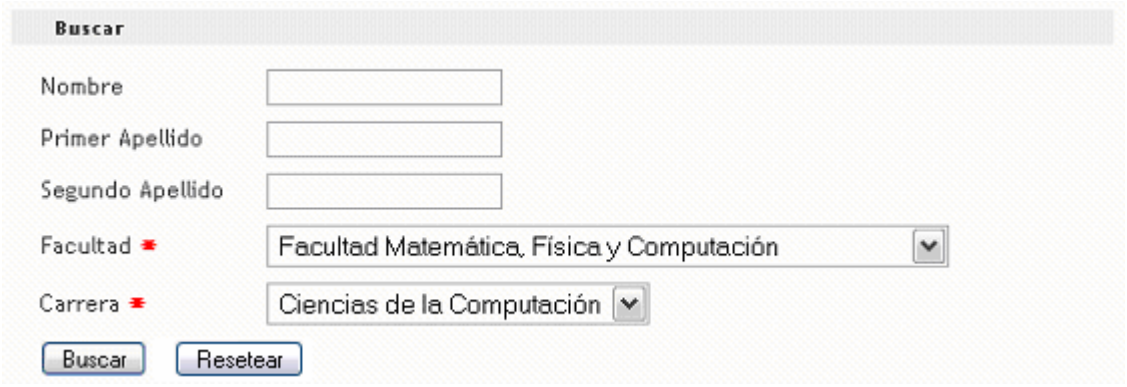
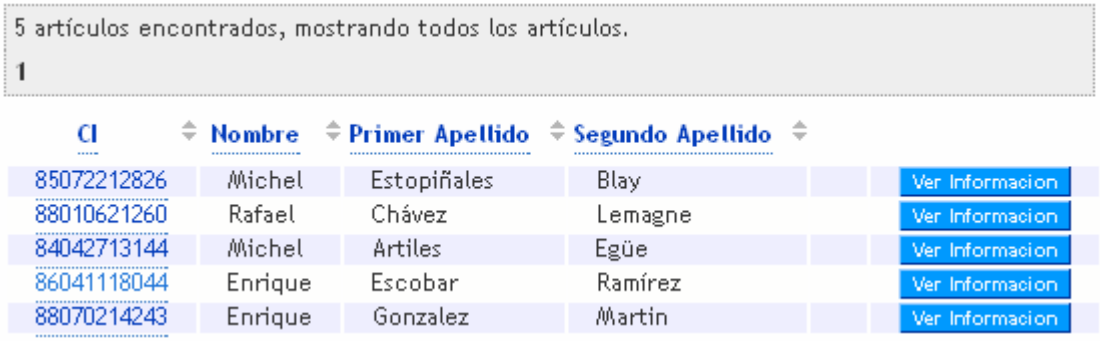


Figura 3.1 Búsqueda de estudiantes

Para buscar un estudiante sin dificultades, primero que todo este debe ser alumno ayudante, de lo contrario nunca se encontrará. Ahora, como muestra la figura anterior, la facultad de estudio y la carrera que cursa el estudiante buscado son obligatorias para efectuar la operación. En caso de no saber sus otros datos, no necesariamente hay que llenarlos. Si no se llenan, el buscador devuelve todos los alumnos ayudantes que pertenecen a esa facultad y estudian esa carrera, para que el usuario seleccione de una tabla el que le interesa. En caso de que se llene alguno, se retornan todos los estudiantes que estudian en esa facultad, cursan esa carrera y por ejemplo, se llamen Juan, en caso que ese haya sido el nombre que se introdujo. El resultado de una búsqueda *x* que cumpla los requisitos de la figura 3.1 se muestra en la siguiente figura:


5 artículos encontrados, mostrando todos los artículos.

1

CI	Nombre	Primer Apellido	Segundo Apellido	
85072212826	Michel	Estopiñales	Blay	Ver Informacion
88010621260	Rafael	Chávez	Lemagne	Ver Informacion
84042713144	Michel	Artiles	Egüe	Ver Informacion
86041118044	Enrique	Escobar	Ramírez	Ver Informacion
88070214243	Enrique	Gonzalez	Martin	Ver Informacion

Figura 3.2 Ejemplo de retorno de una búsqueda

3.3.1.1 Visualizar Información

Para obtener los datos de algún estudiante que sea alumno ayudante, se debe seleccionar la opción *Visualizar Información*, y después de realizar la búsqueda, sobre la fila en que se encuentra el estudiante deseado, dar clic en el botón que aparece a la parte derecha de la tabla o sobre el *link* del carné de identidad. Aparecerá entonces toda la información de ese alumno en su labor de ayudantía durante ese curso (Figura 3.3).


Información del estudiante seleccionado

Buscar Estudiante

Nombre	Michel Artiles Egüe
CI	84042713144
Asignatura	FILOSOFIA
Evaluación	R
Departamento	MATEMATICA
Tutor	CARIDAD
Nivel	4
Destacado	Sí
Colaborador	Sí
Facultad de Ayudantía	DERECHO
Área de Trabajo	IPI

Buscar Estudiante

Figura 3.3 Datos de un Alumno ayudante

3.3.1.2 Visualizar Historial

Para tener acceso a los historiales de un alumno ayudante, se debe dar clic en la opción *Visualizar Historial*. En esta operación, cuando se selecciona un estudiante, (una fila de la tabla) lo que se muestra es otra tabla que guarda la relación de todos los historiales de ese alumno en su labor de ayudantía. De ahí entonces se escoge cuál es el que interesa ver (Figura 3.4).

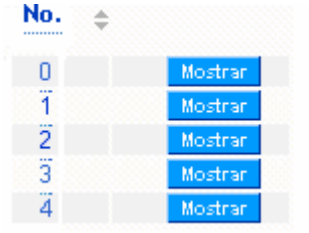


Figura 3.4 Historiales de un Alumno Ayudante

Cuando se selecciona uno de estos, lo que se muestra es algo como lo que aparece en la Figura 3.3 con la información de un Alumno Ayudante en un determinado curso de ayudantía.

3.3.1.3 Visualizar Horario Docente

Para ver cuándo un alumno ayudante tiene que impartir un turno de clases, se debe seleccionar la opción *Visualizar Horario Historial.* En esta acción, cuando se selecciona un estudiante después de realizar la búsqueda, se muestran algunos datos de este y debajo su horario docente como alumno ayudante (Figura 3.5).

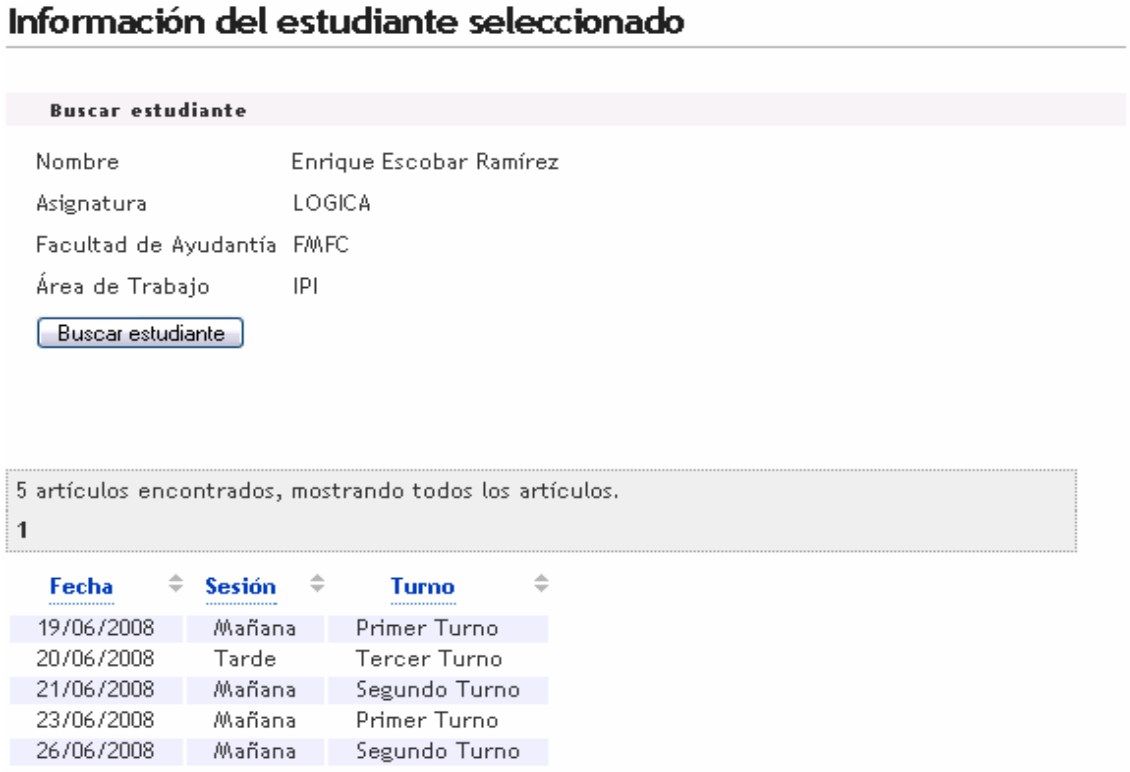


Figura 3.5 Horario de un Alumno Ayudante

Aquí se muestra el día, la sesión y el turno que debe impartir el estudiante seleccionado, además de la asignatura que imparte y la facultad donde está prestando los servicios de docencia.

3.3.2 Jefe de Departamento

El jefe de departamento es el usuario de mayores privilegios --por así llamarlo-- en nuestro ambiente *Web*, por tanto es quien más operatividad tiene en él y será de hecho, el punto de entrada del sistema, pues los demás usuarios no tendrán funcionalidad hasta que este actor no informe de sus alumnos ayudantes a la secretaria (insertado en la tabla temporal).

A continuación describiremos sus posibles acciones y la forma de trabajar con cada una de ellas. Debemos tener en cuenta que el jefe de departamento puede tener acceso a las

operaciones que se detallaron en los puntos anteriores, teniendo en cuenta esto, centraremos la atención solamente en sus casos de uso.

3.3.2.1 Gestionar Evaluación

Cuando el jefe de departamento va a evaluar a un alumno ayudante (se debe hacer solo al terminar cada semestre) debe seleccionar la opción *Gestionar evaluación*. Después de realizar la búsqueda (Figura 3.1) y seleccionar el estudiante deseado, se muestra una página con el siguiente formato:

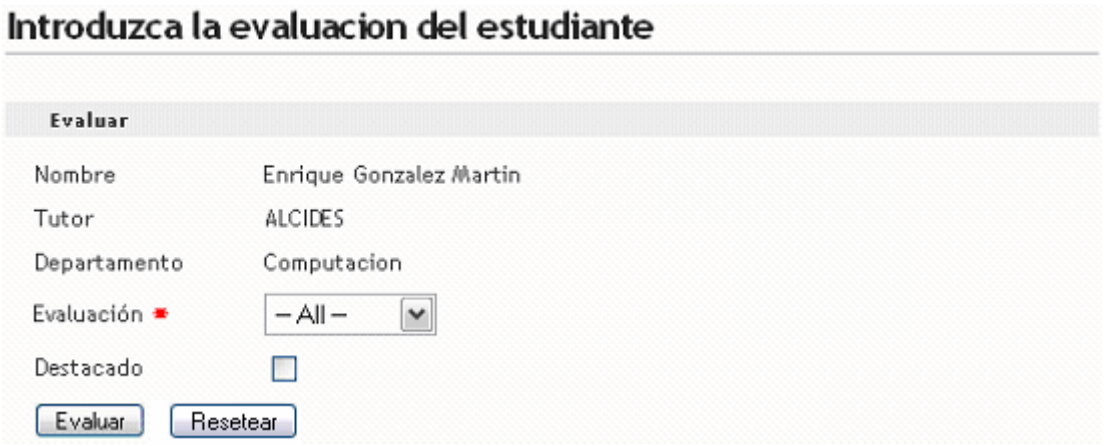


Figura 3.6 Evaluar un estudiante

Aquí el jefe de departamento debe seleccionar la evaluación que quiere asignarle al alumno, y si considera que su labor de ayudantía fue relevante durante ese semestre, entonces debe marcar también el campo de destacado. Al dar clic en *Evaluar*, se guardan estos datos y queda evaluado el estudiante.

3.3.2.2 Introducir Horario Docente

Antes que todo se elige *Introducir Horario Docente*, se efectúa la búsqueda que explicamos con anterioridad y se selecciona el estudiante al que se le quiere introducir su horario. La forma de introducir la información de los turnos de clases que debe impartir este alumno ayudante es como sigue (Figura 3.7):

Figura 3.7 Introducir horario a un estudiante que sea alumno ayudante

Aquí se muestra el nombre, el tutor y el departamento al que pertenece el estudiante y debajo los campos para seleccionar el día de la clase, (calendario que aparece desplegado) la sesión en que tiene que impartirla y el turno que le corresponde. Se oprime *Insertar* y la información queda recogida en la base de datos.

3.3.2.3 Modificar Horario Docente

Esta opción es muy parecida a la detallada en el epígrafe anterior. Su función es permitirle al jefe de departamento modificar datos en el horario de un estudiante que hayan cambiado por algún motivo. Para realizar esta acción se elige *Modificar Horario Docente* y se llega a una página similar a la de la Figura 3.7 pero con los datos editados y listos para ser modificados.

3.3.2.4 Gestionar Eliminación

Cuando el jefe de departamento decide que un estudiante no va a continuar su labor como Alumno Ayudante, debe seleccionar la opción de *Gestionar Eliminación,* nuevamente se hace la búsqueda y se escoge el alumno que pasará a este estado. La página siguiente mostrará entonces los datos de ese estudiante y dará la posibilidad de eliminarlo (Figura 3.7).

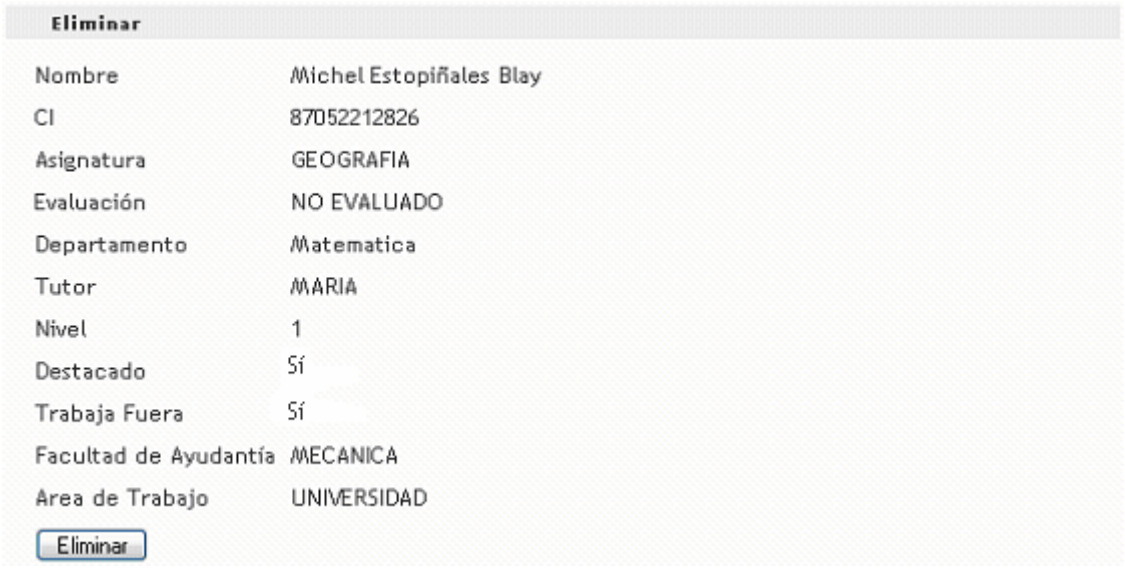


Figura 3.8 Eliminar a un estudiante de su labor como Alumno Ayudante

3.3.2.5 Gestionar Ratificación

En caso de que un estudiante vaya a continuar como alumno ayudante y se mantengan todos sus datos (tutor que lo atiende, asignatura que imparte, facultad de trabajo… etc.) el jefe de departamento debe seleccionar la opción de *Gestionar Ratificación.* De forma general, este proceso funciona igual que el que se explicó en el epígrafe anterior, o sea, se muestran los datos de ese estudiante y se brinda la oportunidad para que se ratifique. Desde el punto de vista visual, solo cambia el nombre del botón, que ahora es *Ratificar*. Para entender esto ver Figura 3.8.

3.3.2.6 Gestionar Modificación

Si por algún motivo durante el curso cambian los datos de un estudiante, el jefe de departamento tiene la posibilidad de actualizarlos seleccionando la opción *Gestionar Modificación*. Hecha la búsqueda y teniendo el alumno a modificar la secuencia de trabajo es como sigue:

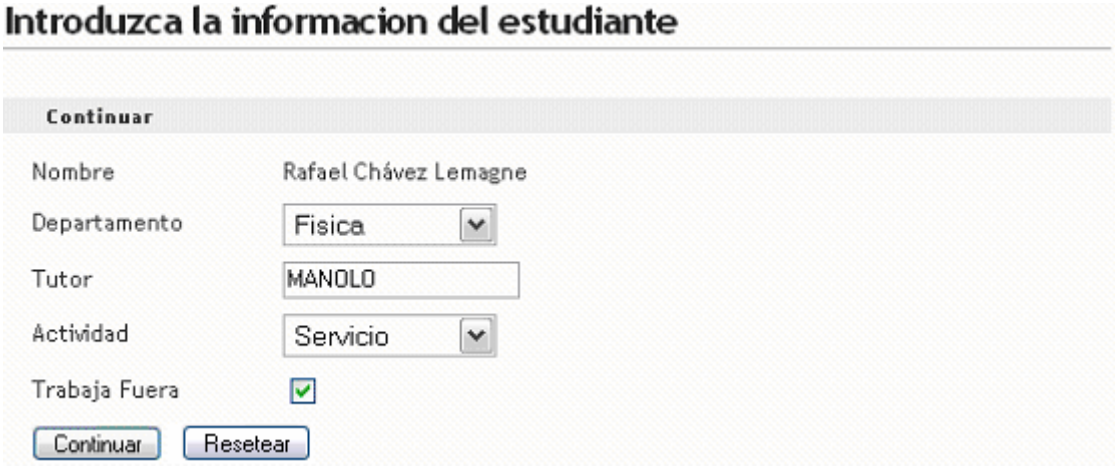


Figura 3.9 Datos editados del estudiante seleccionado

En esta primera página se muestran algunos de los datos del alumno ayudante seleccionado, de manera que pueden ser cambiados en caso de estar erróneos. Después de tener esta información se da clic en *Continuar* y navegamos a la siguiente página:

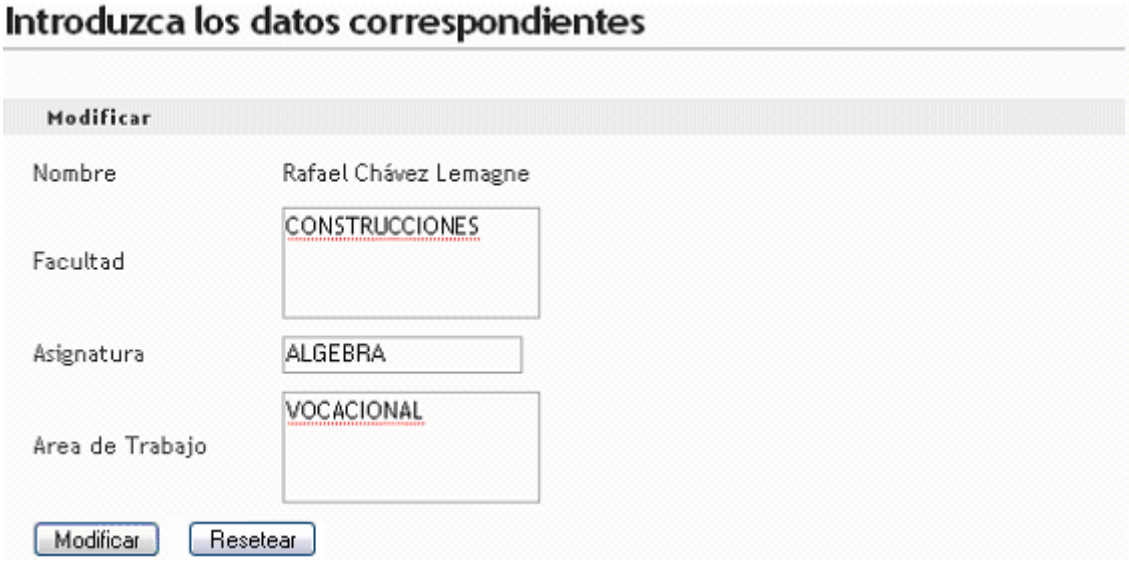


Figura 3.10 Restantes datos editados del estudiante seleccionado

Al igual que en la Figura 3.9, en esta nueva página se muestran los restantes datos para que sean modificados si es necesario. Después de hacer los cambios pertinentes se da clic en *Modificar*, y las actualizaciones son hechas en la base de datos.

3.3.2.7 Gestionar Adición

Como se dejó claro en el epígrafe 3.3.1, de todas las acciones que necesitan establecer una búsqueda, el único caso de uso que la realiza de forma distinta es este que estamos analizando. Es bueno recordar que esta es la única operación encargada de leer de la entidad *Student* del módulo de *Minotaur*, por lo que en aras de ganar en eficiencia, no solo filtramos los datos por facultad y carrera, sino que además se le exige al usuario que especifique el grupo en que estudia el alumno para reducir aún más el espacio de búsqueda, de lo contrario se tornaría demasiado engorrosa y lenta la búsqueda. Debido a esto, cuando se selecciona *Gestionar Adición* se presenta la página que mostramos a continuación:

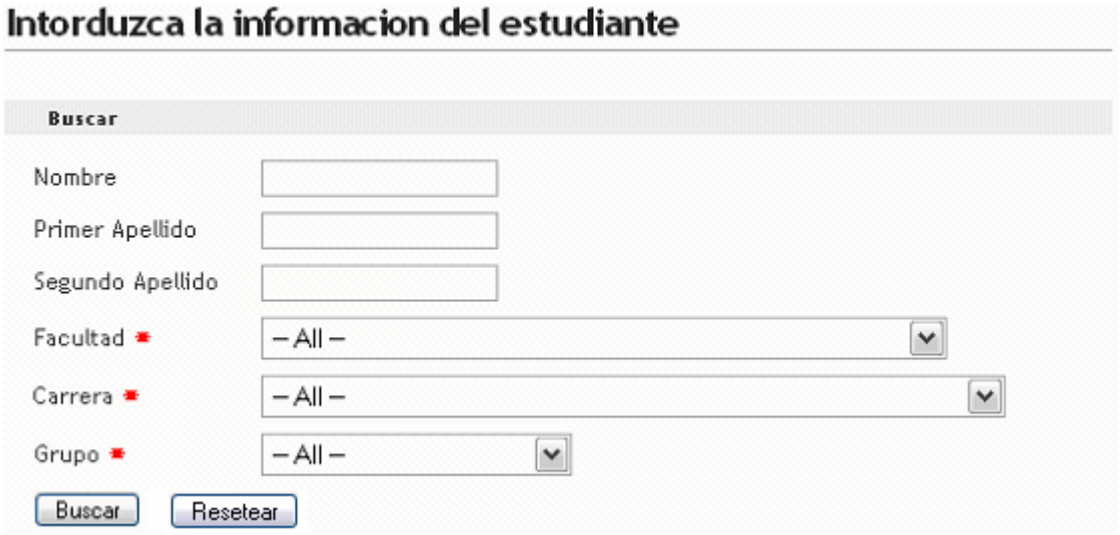


Figura 3.11 Búsqueda para insertar un nuevo Alumno Ayudante

La lógica de esta búsqueda es básicamente igual que la que se explicó anteriormente y es obligatorio introducir el grupo para que esta surta efecto. Cuando se da clic en *Buscar,* aparece una tabla con los estudiantes que cumplen los requisitos especificados para que el usuario seleccione el que desee. Luego se pasa a la fase de introducir los datos como mostramos en las siguientes figuras:

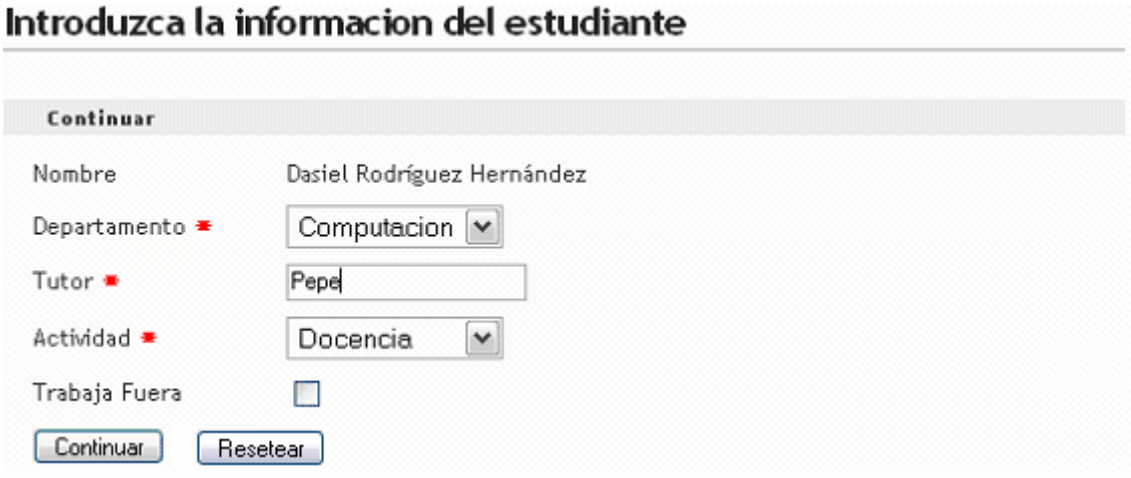


Figura 3.12 Primeros datos que debe introducir el Jefe de Departamento

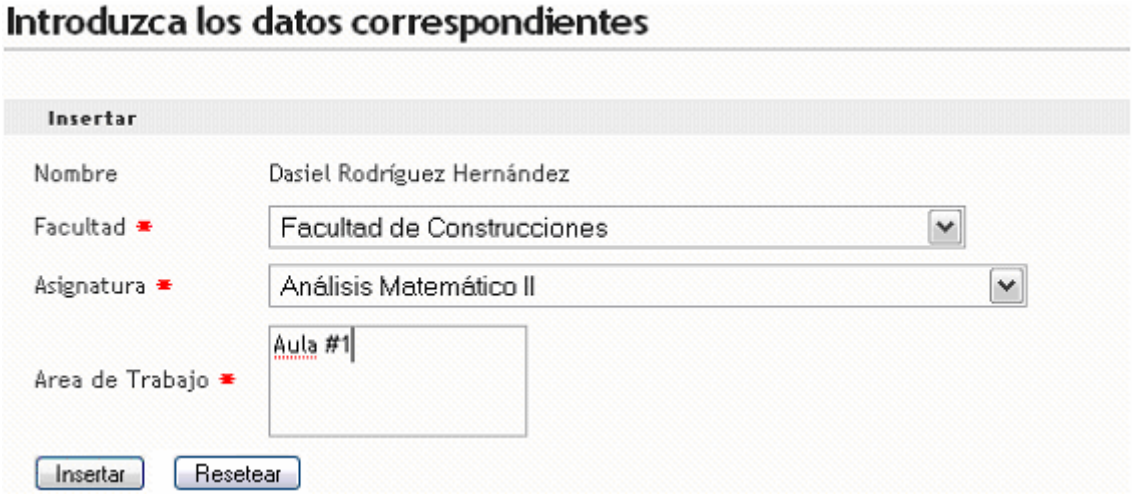


Figura 3.13 Datos restantes

Cuando ya se tienen todos los datos se deba oprimir el botón *Insertar* y si la operación se realiza satisfactoriamente se muestra un mensaje de confirmación (Figura 3.14).

Figura 3.14 Mensaje de confirmación.

3.3.2.8 Generar Reportes

Dentro de las facilidades que brinda este módulo *Web* al jefe de departamento, está la de obtener reportes de los alumnos ayudantes que trabajan en su departamento. La idea que se siguió para lograr esta operación es la misma para todos los tipos de reportes que se implementaron, por lo que se tomará el reporte por actividad como ejemplo general. Para acceder a este caso de uso se selecciona la opción *Generar Reporte por Actividad*, y se carga una página con la información siguiente:

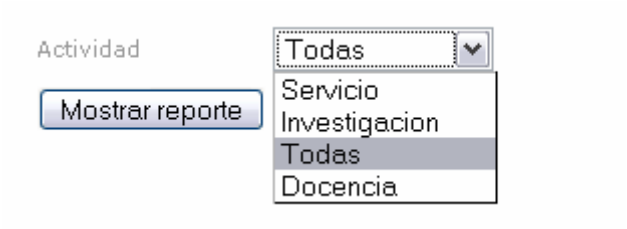


Figura 3.15 Generar reporte por actividad

Después de seleccionar la actividad se da clic en *Mostrar reporte* y se genera un reporte similar al de la Figura 3.30.

3.4 Opciones principales del módulo Desktop del sistema

Con el ambiente *Desktop* de nuestro sistema solo interactúa el actor secretaria, por tanto, todos los detalles que se dan a continuación referentes a las opciones que brinda este módulo son importantes para este único actor.

3.4.1 Secretaria

La secretaria tendrá acceso a dos funciones principales desde este módulo:

i. Realizar operaciones sobre los estudiantes.
ii. Obtener reportes de los Alumnos ayudantes.

Antes de entrar en los detalles de cada opción, mostramos a continuación la siguiente figura que sirve para dar una panorámica del entorno con el que se relacionará la secretaria.

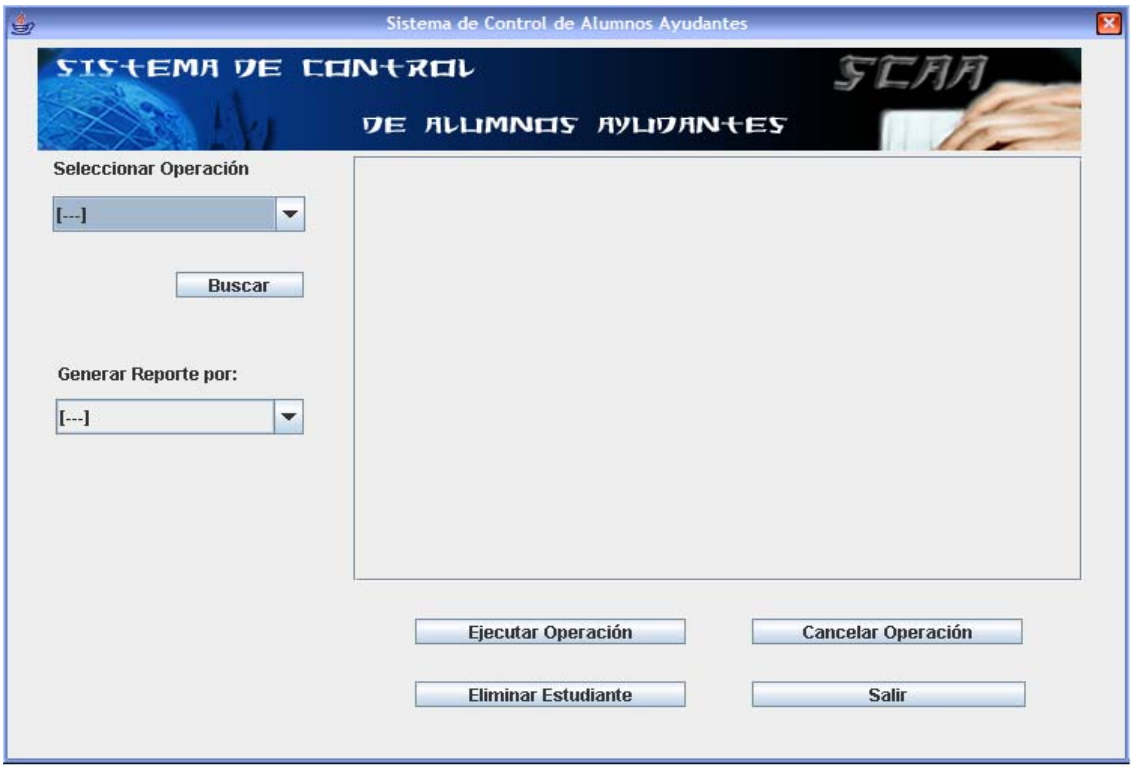


Figura 3.16 Ventana principal de la aplicación *Desktop* del sistema.

Centrándonos en la figura anterior, se puede llegar a la conclusión que la aplicación está dividida en cuatro partes fundamentales.

Sección donde se selecciona la operación a realizar.

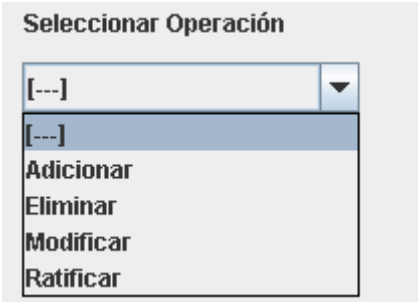


Figura 3.17 Operaciones posibles

Sección donde se selecciona el tipo de reporte que se desea obtener.

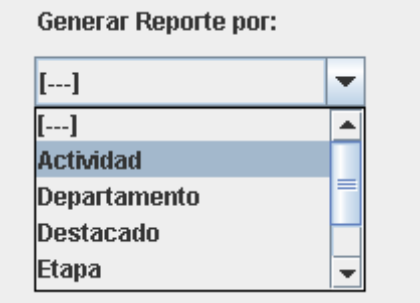


Figura 3.18 Posibles tipos de reportes

Estas dos secciones se encuentran en la parte izquierda de la aplicación y son el punto de entrada para que el sistema pueda trabajar sin ningún tipo de dificultad. Es bueno aclarar, que una no depende de la otra y esto es válido en ambos sentidos, es decir, si se desea realizar una operación no es necesario seleccionar un tipo de reporte, y de igual manera, si se desea obtener un reporte, no es necesario ni seleccionar ni ejecutar una operación.

Tabla donde se muestra los datos de acuerdo a la operación seleccionada.

Figura 3.19 Datos de los estudiantes

Botones de ejecución.

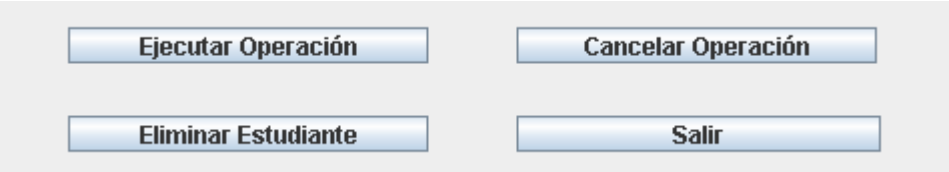


Figura 3.20 Botones encargados de manipular los datos de la tabla

Una vez mostradas las secciones en que se divide la aplicación, pasamos a explicar detalladamente cómo se trabaja con las mismas.

3.4.1.1 Operaciones a realizar con los estudiantes.

Las cuatro posibles operaciones que puede llevar a cabo la secretaria sobre los estudiantes (de su facultad) son:

1. Adicionar
2. Eliminar
3. Modificar
4. Ratificar

El modo de operar para llevar a cabo una de estas acciones de forma satisfactoria es el siguiente. Primero que todo, para poder ejecutar una operación, es necesario haber seleccionado anteriormente una de ellas, como se muestra en la siguiente figura:

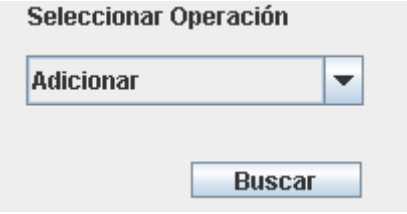


Figura 3.21 Selección de operación

Una vez seleccionada la operación, se debe dar clic en el botón *Buscar*. Esta acción, es la encargada de mostrar en la tabla los estudiantes que están disponibles en ese

momento para que la secretaria opere con ellos. La aplicación pasará entonces al estado siguiente:

Nombre	Carnet de Identidad	Departamento	Operación
Liyam Paz Estevez	85082416016	Física	Adicionar
Enrique Escobar R...	86041118044	Física	Adicionar
Nayle Delgado Arce	84110112436	Matemática	Adicionar
Michel Artiles Egüe	84042713144	Computación	Adicionar

Figura 3.22 Estudiantes que serán adicionados por la secretaria si esta lo determina así.

Después que estamos en este punto, la secretaria debe determinar qué hará con cada estudiante. En caso de dar clic sobre el botón *Cancelar Operación*, se limpia la tabla, si por el contrario no desea hacer la operación que seleccionó con un estudiante determinado, lo marca y oprime *Eliminar Estudiante,* esto lo eliminará de la lista de la tabla, si la secretaria determina que sí realizará una operación con un estudiante, debe entonces dar clic en *Ejecutar Operación* y se pueden presentar los casos que explicamos a continuación.

Nota: Es responsabilidad total de la secretaria, determinar (en caso que la operación sea *Adicionar*, *Modificar* o *Ratificar*) si el estudiante debe o no incrementar su etapa de ayudantía. Para esto, siempre que se ejecuta una de estas operaciones, se muestra una ventana de diálogo que brinda la posibilidad de aumentársela o no, según estime la secretaria.

⇒ Si el estudiante va a ser añadido al movimiento de Alumnos Ayudantes por primera vez, entonces se muestra el siguiente diálogo:

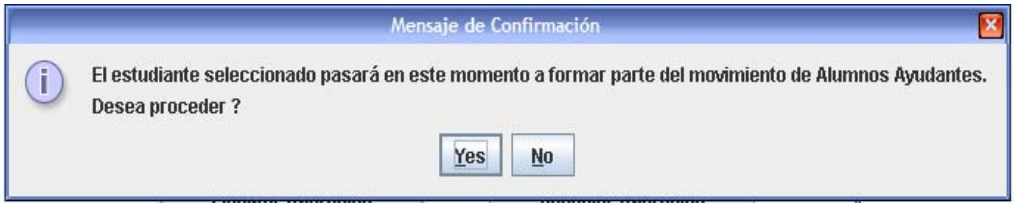


Figura 3.23 Insertar estudiante por primera vez

Si se selecciona *Yes,* se inserta en la primera etapa de ayudantía, en caso contrario no se hace nada.

⇒ Una nueva variante es que el estudiante no se incorpore a este movimiento por primera vez (causó baja por "*x*" motivo y era Alumno Ayudante). En este caso se presenta el siguiente diálogo:

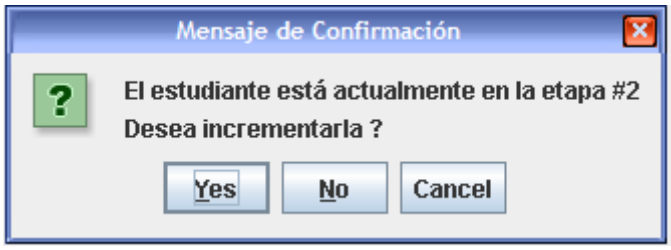


Figura 3.24 Insertar estudiante que en algún momento fue Alumno Ayudante

⇒ En caso que lo que se quiera sea *Modificar* o *Ratificar,* la filosofía que se sigue es la misma, solo que cuando se va a efectuar una de estas acciones el estudiante ya tiene que existir como alumno ayudante, por lo que nunca se presentará el caso en que se muestre el primero de los diálogos anteriores (Figura 3.23).

⇒ Si al tratar de efectuar una de las operaciones, por alguna causa se manipula de forma incorrecta la aplicación o no existen estudiantes para llevar a cabo la acción que se desea se muestran los mensajes siguientes:

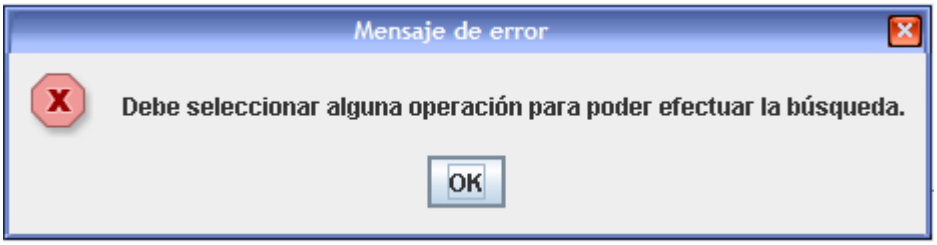


Figura 3.25 Mensaje de error

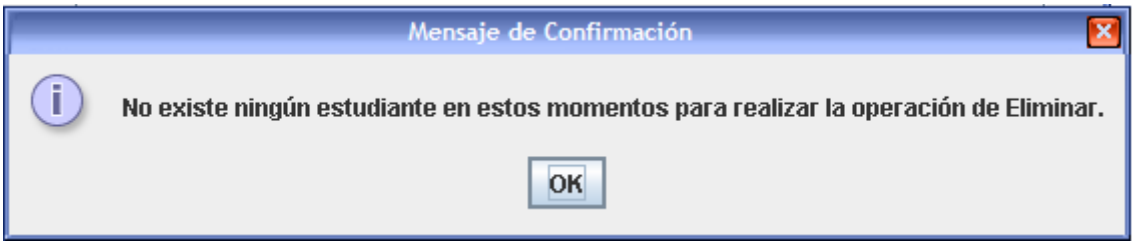


Figura 3.26 Mensaje de advertencia

3.4.1.2 Obtener reportes de los Alumnos Ayudantes

Cuando el objetivo de la secretaria sea obtener información de los Alumnos Ayudantes de su facultad, antes que todo debe seleccionar el tipo de reporte que desea. Aquí pueden presentarse dos variantes, si su finalidad es reportar por *Facultad* o por *Destacados,* la apariencia de la ventana es la que se muestra en la siguiente figura, pues siempre este reporte se hará a nivel de facultad, por lo tanto no es necesario preguntar nada más.

Figura 3.27 Reporte por destacados

Si el reporte que se pretende es de otro tipo, entonces la vista se presenta como se muestra a continuación:

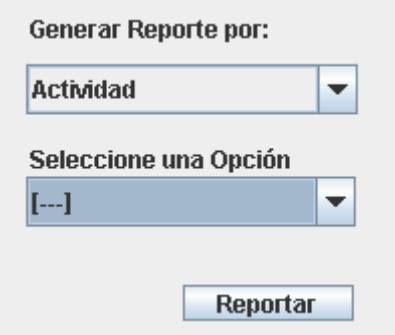


Figura 3.28 Reporte por actividad que realiza

Debe entonces seleccionar una opción de las posibles para ese tipo de reporte, por ejemplo, para este caso anterior sería:

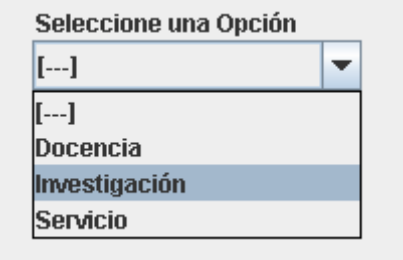
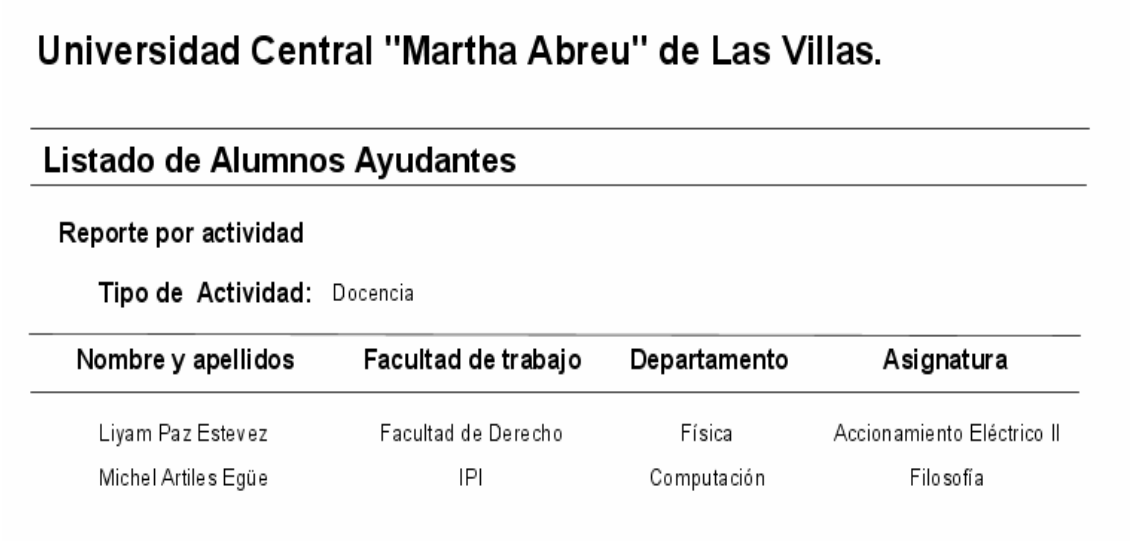


Figura 3.29 Actividades posibles para efectuar un reporte de este tipo

Para ambas variantes, después que se realicen estos pasos, se debe dar clic en el botón *Reportar* y en dependencia del tipo de reporte que se quiera obtener, se mostrará la información en formato *.pdf* como se puede ver en la próxima figura:



Universidad Central "Martha Abreu" de Las Villas.

Listado de Alumnos Ayudantes

Reporte por actividad

Tipo de Actividad: Docencia

Nombre y apellidos	Facultad de trabajo	Departamento	Asignatura
Liyam Paz Estevez	Facultad de Derecho	Física	Accionamiento Eléctrico II
Michel Artiles Egüe	IPI	Computación	Filosofía

Figura 3.30 Reporte por actividad de docencia.

3.5 Análisis de los resultados

Se puede afirmar que este sistema contribuirá de manera significativa al aumento de la calidad del trabajo en la gestión de los alumnos ayudantes en el MES, permitiendo una organización de los datos relacionados con los procesos.

Con este módulo se automatizan una serie de operaciones que antes se hacían de forma manual, agilizando así procesos como:

- Gestión de los alumnos ayudantes.
- Gestión de las evaluaciones de los alumnos ayudantes.
- Generación de reportes.

Resulta importante señalar que la elección del uso de AndroMDA y en general la metodología utilizada, garantiza la novedad y el carácter científico de la presente investigación, ya que esto fue el resultado de un estudio de las tendencias actuales en cuanto al desarrollo de las aplicaciones empresariales, herramientas que presentan una serie de ventajas explicadas con anterioridad en este trabajo.

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

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

Conclusiones Generales

A partir de los resultados obtenidos en el presente trabajo podemos concluir que:

1. Se automatizaron los procesos en cuanto a la gestión de los alumnos ayudantes en la Universidad Central "Martha Abreu" de la Villas y otros Centros de Educación Superior obteniendo una mejor calidad y eficiencia.

2. Se desarrolló el sistema utilizando como herramientas J2EE™ y AndroMDA respondiendo así a una plataforma *Open Source* y compatible con el Sistema de Gestión de la Nueva Universidad, además de ser herramientas idóneas para el desarrollo de aplicaciones empresariales.

3. Debido a las actuales necesidades del Ministerio de Educación Superior, a diferencia de la primera versión que se desarrolló del presente módulo, se creó además de la aplicación *Web*, una aplicación *desktop* que facilita el trabajo de las secretarias y garantiza la seguridad del sistema. Por otra parte, los mayores beneficios que proporciona el módulo *Web,* son sin lugar a dudas , su forma de acceso, pues puede ser utilizado desde cualquiera de los Centros de Educación Superior o Sedes Universitarias Municipales, respondiendo así a las necesidades de todos los posibles usuarios que intervienen en el proceso.

Recomendaciones

1. Adicionar en el Sistema de Matrícula y Control Docente (Cliente SIGENU) los codificadores necesarios para facilitar el control del acceso de los jefes de departamento al sistema(s), concretamente en el Menú Codificadores, agregar el departamento docente.

2. Perfeccionar la parte visual de nuestra aplicación *Web* utilizando el *Eclipse Web Tools Platform (WTP),* herramienta que entre sus facilidades brinda la edición gráfica de páginas *Web*, de manera que se logre un ambiente más amigable y acogedor para los posibles usuarios que interactúen con el mismo.

Referencias bibliográficas

ARMSTRONG, E., BALL, J. & BODOFF, S. (2005) ***The J2EE™ 1.4 Tutorial,*** **California, U.S.A., Sun Microsystems ®, Inc.**

BOHLEN, M., BRANDON, C. & ZOONS, W. (2005) AndroMDA Model Driven Architecture Framework.

DANCIU, T. (2005) JasperReports.

GARCÍA, L. (1995) Sistema automatizado de Control de Estudiantes en Red.

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

JBOSS™ INC. (2005) ***The Jboss™ 4 Application Server Guide*****.**

MINISTERIO DE EDUCACIÓN SUPERIOR (2007) Resolución Ministerial 210/07 Reglamento para el Trabajo Docente y Metodológico en la educación superior.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP (2004) PostgreSQL 8.0.0beta5 Documentation.

TOFFOLI, G. (2005) A design tool iReport for JasperReports.

WIKIPEDIA, E. (2007) Hibernate

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

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

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

SOLEY, R. (2006). Model Driven Architecture, OMG.

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

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

Rational Software Corporation (2006). Online Help.

Bibliografía

Enterprise application. http://en.wikipedia.org/wiki/Enterprise_application

Sistema de gestión de base de datos.
http://es.wikipedia.org/wiki/Sistemas_gestores_de_bases_de_datos

AndroMDA 3.2 http://galaxy.andromda.org

http://java.sun.com/javaee

GRUPO DE GESTIÓN DE LA NUEVA UNIVERSIDAD (2005) Seguridad en J2EE y JBOSS SX.

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

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

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

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

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

FLEITES, E. & BOLAÑOS, T. (2005) Módulo de captación de datos del Sistema de Gestión Centralizado para el control del postgrado en la UCLV.

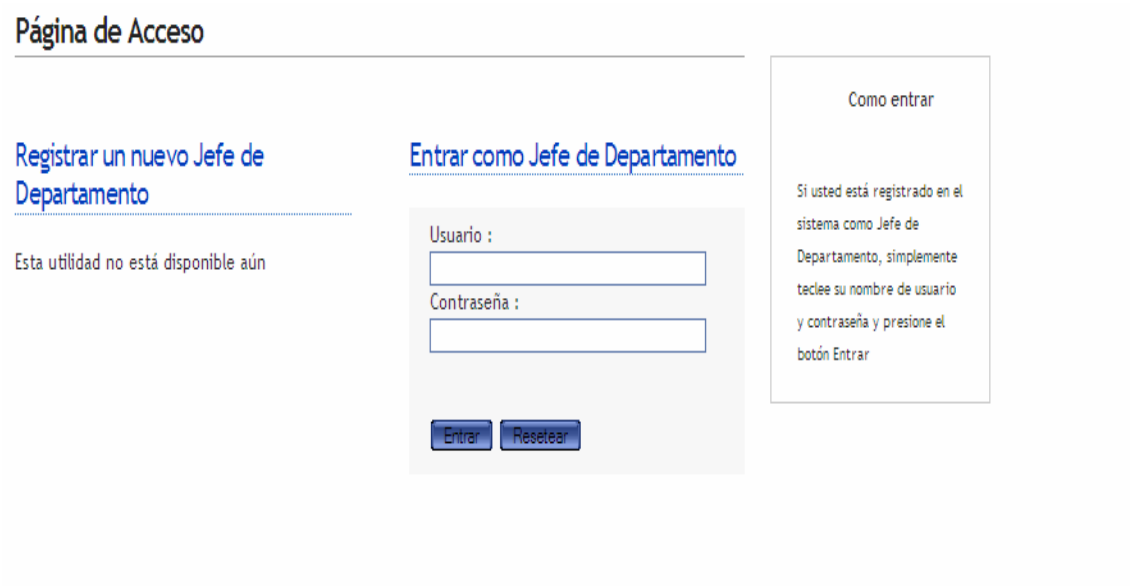
VALERA, Y. & QUIRCE, A. (2006) Módulo de Control de Alumnos Ayudantes del SISGENU

PÉREZ, D. & LORENZO, JJ. (2006) Módulo de Estipendio

TILLY, J. & BURKE, E. (2005) Ant: The Definitive Guide O'Reilly.

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

Anexos



Anexo 1 Página de autenticación para el jefe de departamento