HERRAMIENTA DE SOFTWARE PARA LA ASIGNACIÓN DE LAS AULAS Y ESPACIOS FÍSICOS REQUERIDOS PARA LA PROGRAMACIÓN DE ASIGNATURAS Y GRUPOS OFRECIDOS POR CADA UNA DE LAS ESCUELAS SEMESTRALMENTE EN LA UNIVERSIDAD INDUSTRIAL DE SANTANDER.

Javier Alberto Moreno González

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2011

HERRAMIENTA DE SOFTWARE PARA LA ASIGNACIÓN DE LAS AULAS Y ESPACIOS FÍSICOS REQUERIDOS PARA LA PROGRAMACIÓN DE ASIGNATURAS Y GRUPOS OFRECIDOS POR CADA UNA DE LAS ESCUELAS SEMESTRALMENTE EN LA UNIVERSIDAD INDUSTRIAL DE SANTANDER.

JAVIER ALBERTO MORENO GONZÁLEZ

Proyecto de grado presentado como requisito parcial para optar el título de Ingeniero de Sistemas

Director

Msc. Luis Ignacio González Ramírez

Codirector

Msc. Jorge Villamizar Morales

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2011

DEDICATORIA

A Dios que me ha dado la vida, la paciencia, sabiduría y fortaleza,

A mis padres que siempre se han esmerado por hacer de mí un ser humano de bien, recto, justo y preparado para afrontar cualquier obstáculo,

A mi hermano que se ha convertido en el reflejo de mis acciones,

A todos aquellos que han pasado por mi vida dejando una enseñanza.

Javier Alberto Moreno González

AGRADECIMIENTOS

Un agradecimiento especial a Dios, Padre todo poderoso, que me ha servido de apoyo y camino para sobrellevar cada prueba que afronto en mi vida.

A mis padres, adorables seres humanos que han puesto toda su voluntad para enseñarme, guiarme y apoyarme en el buen camino de mi formación como persona, ser humano e individuo de la sociedad.

A todos y cada uno de mis amigos quienes me han acompañado en cada uno de los momentos que han dejado una enseñanza en mi.

A la División de Servicios de Información de la Universidad Industrial de Santander por brindarme la oportunidad de demostrar mis habilidades como profesional.

A la Dirección de Admisiones y Vicerrectora Académica de la Universidad Industrial de Santander, por la confianza depositada en mí para la realización de éste proyecto.

Y en general a la comunidad académica de la Universidad Industrial de Santander por haber hecho de mí un profesional.

TABLA DE CONTENIDO

LISTADO DE FIGURAS

LISTADO DE TABLAS

RESUMEN

TITULO: HERRAMIENTA DE SOFTWARE PARA LA ASIGNACIÓN DE LAS AULAS Y ESPACIOS FÍSICOS REQUERIDOS PARA LA PROGRAMACIÓN DE ASIGNATURAS Y GRUPOS OFRECIDOS POR CADA UNA DE LAS ESCUELAS SEMESTRALMENTE EN LA UNIVERSIDAD INDUSTRIAL DE SANTANDER[*].

AUTOR: MORENO GONZÁLEZ JAVIER ALBERTO[**]**

PALABRAS CLAVE: Sistema de Información Web, Asignación automática, Aulas, Administración de recursos, Aplicaciones Cliente Servidor, Java, Modelo de Construcción de Prototipos.

DESCRIPCION:

El proceso de creación y gestión de grupos en la Universidad Industrial De Santander se hace de forma manual semestralmente, este es un proceso tedioso que implica mucho tiempo y esfuerzo de parte de los directivos de cada escuela para configurar los horarios y aulas de cada grupo dependiendo de los recursos disponibles.

Ante las necesidad de las escuelas por obtener un sistema de información que brinde soporte en la programación, asignación y administración de grupos semestralmente, ha surgido también la necesidad de un sistema con el que se pueda gestionar la asignación automática de aulas a dichos grupos, esta tesis está basada en la investigación de la administración optima de recursos por medio de artefactos automáticos basados en restricciones, como resultado del desarrollo de la investigación, se ha generado una herramienta colaborativa basada en java 5, que permite a los directivos de la Universidad Industrial De Santander administrar de manera automática los recursos como físicos como aulas, laboratorios y auditorios a nivel de escuela, y a nivel de sede.

El desarrollo de esta herramienta se encuentra encapsulado en el estándar de programación de la división de servicios de información de la Universidad Industrial De Santander comprendiendo las tecnologías de programación orientada a objetos, la estructura de modelo, vista, controlador y la metodología de desarrollo por prototipos.

[*] Trabajo de Investigación

[**] Facultad de Ingenierías Físico – Mecánicas, Escuela de Ingeniería de Sistemas e Informática. Director: Msc. Luis Ignacio González Ramírez

SUMMARY

TITLE: TOOL FOR THE ASSIGNMENT AND ORGANIZATION OF PHYSICAL RESOURCES REQUIRED FOR PROGRAMMING COURSES AND GROUP OFFERED FOR EACH OF THE SEMESTER SCHOOLS IN THE INDUSTRIAL COLLEGE OF SANTANDER[*]. *

AUTHOR: MORENO GONZÁLEZ JAVIER ALBERTO[**]

KEYWORDS: Web Information System, Automatic Mapping, Classroom, Resource Management, Client Server Applications, Java, Model Prototypes Construction.

DESCRIPTION:

Process of creating and managing groups in the Universidad Industrial De Santander is done manually twice a year, this is a tedious process that involves much time and effort on the part of managers from each school to set timetables and classrooms in each group depending available resources.

Given the need for schools to obtain an information system that provides support for the planning, allocation and management of every six months groups, has also come the need for a system that can automatically manage the allocation of classrooms to these groups, this thesis research is based on optimal management of resources through automatic constraint-based devices as a result of research development, has created a collaborative tool based on Java 5, which allows managers of the Universidad Industrial De Santander automatically manage physical resources such as classrooms, laboratories and auditoriums at the school level and headquarters level.

Development of this tool is encapsulated in the standard programming information services division of the Universidad Industrial De Santander, technology understanding object-oriented programming, the structure of Model View Controller and methodology for developing prototypes.

[*] Project

[**] Physique-Mechanics Sciences Department, Computer Science Engineering, Msc. Luis Ignacio González Ramírez.

TERMINOS Y DEFINICIONES

DSI : División de Servicios de Información.

Asignatura : Cada una de las materias que se enseñan en el centro educativo y posee un código especifico.

Grupo : Cada uno de los conjuntos que se programan semestralmente para las Asignaturas. Con un código y horarios de clase específicos.

Director escuela: Persona encargada de dirigir una escuela o departamento en la UIS.

Administrador: Persona encargada de administrar las transacciones dentro del modulo de administrador del sistema de información.

UAA : Unidad Académica y Administrativa de la Universidad Industrial de Santander.

Interfaz : La idea fundamental en el concepto de interfaz, es el de mediación. La interfaz es lo que “media”, lo que facilita la comunicación, la interacción, entre dos sistemas de diferente naturaleza, típicamente el ser humano

y una máquina como el computador. Esto implica, además, que se trata de un sistema de traducción, ya que los dos se comunican con lenguajes diferentes: verbo-icónico en el cado del hombre y binario en el caso de la máquina.

Java : Lenguaje de Programación que se caracteriza por tener una arquitectura que permite que el código escrito funcione en multitud de sistemas operativos sin ser modificado.

Método : Es una operación que define como se comporta un objeto.

Módulo : Se utiliza como sinónimo de Subsistema.

Sistema de Información: Aplicación comercial para el computador. Está constituida por la base de datos, los programas de aplicación, los procedimientos manuales y automatizados, e incluye los sistemas computacionales que realizan procesamiento.

Clase : Una clase de objetos describe un grupo de objetos con propiedades similares, con relaciones comunes entre otros y con una semántica común.

INTRODUCCION

En la actualidad, la administración de entidades públicas requiere gran número de personal y tecnologías que permitan a los funcionarios dar soporte sobre el manejo y gestión de la amplia gama de recursos con los que cuentan estas entidades.

El proceso de asignación de aulas es uno de estas necesidades en la cual la inversión de tiempo y recursos humanos es muy alta, este proceso se lleva a cabo al menos dos veces al año por cada una de las escuelas y posteriormente para concluir con la completa asignación de aulas interviene la División de Admisiones de la Universidad.

La asignación de aulas generalmente se desarrolla en un ámbito donde la demanda de aulas y horarios específicos tiende a crecer, como también las necesidades de los directores de escuela de utilizar aulas particulares para funciones especiales durante el semestre.

Este proyecto aborda el problema de asignación de aulas a nivel general para cada una de las escuelas de la Universidad Industrial de Santander, teniendo como principal cuerpo de estudio la sede principal que alberga la totalidad de escuelas, incluyendo la sede de medicina. Abordar esta problemática se enfoca en dos principales factores: Primero, la asignación de aulas actualmente presenta altos niveles de complejidad y demanda grandes cantidades de tiempo y esfuerzo administrativo, Segundo: La disposición de las tecnologías que utiliza la DSI y la colaboración de la División de admisiones permite generar una herramienta que comprenda todos y cada uno de los departamentos que pueden ser beneficiados con la generación de este sistema.

PARTE I. FUNDAMENTOS

1. PRESENTACION DEL PROYECTO

1.1 DESCRIPCION DEL PROYECTO.

1.1.1 Objetivo General. Desarrollar e implementar una herramienta de software que facilite la asignación de aulas semestralmente, según la demanda y asignación de cursos, horarios y profesores ofrecidos por cada escuela.

1.1.2 Objetivos Específicos. El sistema de gestión de aulas tiene como objetivos específicos:

- Levantamiento de requerimientos y Análisis, de la herramienta de software para la organización y la asignación de aulas que contemple los siguientes insumos:
    - Información recolectada tal como la asignación estipulada en los semestres anteriores.
    - Datos de los recursos como la capacidad de las aulas y el tipo de infraestructura.
    - Datos básicos de la totalidad de los grupos, asignaturas y horarios establecidos.
- Diseñar una herramienta que cumpla con las siguientes funcionalidades:
    - Permitir a las escuelas asignar grupos en aulas especiales tales como audiovisuales, aulas reservadas y demás.
    - La asignación automatizada de las aulas de escuela a los grupos de la escuela por parte de la División de Admisiones.

    - Permitir a las escuelas administrar o gestionar la ocupación de aulas o espacios específicos para que estos no se vean involucrados en la asignación automática.
- Ofrecer la programación de aulas desde cero según los parámetros asignados.
- Análisis y diseño de casos de uso aplicados o involucrados directamente con la herramienta de asignación de recursos físicos de la universidad.
- Generar informes con la información detallada sobre nivel de uso de los recursos físicos de cada escuela como de cada una de sus aulas.
- Componer los respectivos manuales de uso de la herramienta para brindar soporte a los usuarios finales.

1.2 JUSTIFICACION.

1.2.1 Antecedentes y descripción del problema.

La asignación apropiada de los recursos físicos en la Universidad Industrial de Santander permite que la comunidad universitaria obtenga todo el potencial de las instalaciones a su disposición, es por esto que es necesario para la universidad hacer uso de las herramientas informáticas que permitan agilizar y optimizar el proceso de asignación de aulas dando soporte consistente que minimice el margen de error. Adicionalmente la Universidad Industrial de Santander, pionera en ciencia y tecnología, está en la capacidad de implementar procesos tecnológicos de administración de recursos.

Los servicios que presta la División de Sistemas de Información de la universidad deben estar en constante cambio y mejoramiento, a su vez debe prestar soluciones a los problemas y necesidades que se presenten por parte de los

usuarios del sistema incrementando la productividad, rendimiento y fiabilidad de los procesos y servicios prestados.

1.2.2 Impacto.

La División de Sistemas de Información se esmera por que los servicios brindados a la comunidad universitaria sean lo más robustos y útiles. A su vez, se espera que los procesos que son sistematizados sean ágiles, dinámicos, seguros y eficientes, es pues por esta razón que se crean nuevos servicios para responder a las necesidades de la comunidad universitaria.

Esta herramienta permite tanto a los directores de escuela como a los administradores del sistema, hacer gestión de la asignación de las aulas otorgadas a cada escuela de manera automática, previniendo cualquier tipo de cruce o manejo indebido de las instalaciones de la universidad.

1.2.3 Viabilidad.

La generación del servicio de administración de recursos físicos de la universidad es viable pues la estructura organizacional de los sistemas de información de la universidad soporta la implementación de nuevos servicios gracias a su modularidad, el manejo de plataformas web y la coordinación de servicios o procesos categorizados por metas especificas.

Además se cuenta con la supervisión por parte del director del proyecto y la colaboración del equipo de trabajo, agentes de gran.

Para la realización de éste proyecto se cuenta con herramientas de libre distribución como el Jboss Developer Studio 3.0 G.A, el servidor de aplicaciones Jboss-seam productos patrocinados por la compañía Red Hat Software Inc, la cual tiene un contrato de soporte con la Universidad Industrial de Santander. Esto conlleva muchas facilidades durante su desarrollo, éstas herramientas

evolucionan en términos de calidad y seguridad, puesto que hay miles de personas en todo el mundo trabajando en ellas a través de internet, con la formación de grupos de discusión que brindan información y capacitación a los desarrolladores a través del intercambio de conocimiento acerca de los distintos productos,

Aquellas herramientas que no son de libre distribución ya han sido adquiridos y licenciados por la Universidad Industrial de Santander.

En cuanto a hardware se cuenta con instalaciones adecuadas, con los equipos requeridos y el soporte tecnológico necesario para el desarrollo del mismo.

2. MARCO TEÓRICO

La asignación de horarios y cursos es un ejemplo común para el problema de programación y se ha manifestado de diferentes formas a través del tiempo, una forma particular de la calendarización de horarios se muestra en la programación de cursos de las universidades, esta forma conlleva un manejo adecuado de los recursos y denota la necesidad de un artefacto o herramienta que permita hacer de esta tarea menos tediosa y prolongada.

La programación de cursos conlleva a la tarea de asignación de aulas, pues un grupo debe tener entre sus requisitos para su propio desarrollo, el horario de clases, un profesor para cada horario, y un lugar donde se desarrolle la asignatura.

Desde los años 70, el tema de asignación de horas y salones ha sido objeto de constante estudio, la programación de actividades o eventos acordes con la disponibilidad de recursos y sus requerimientos [4] es una tarea que deben enfrentar por lo menos dos veces al año los establecimientos educativos particularmente las Universidades, la mayoría de estas instituciones emplea una heurística simple, asistida por un ordenador que consiste en una asignación basada en el orden de llegada de los listados de requerimientos de aulas para cada sesión que se desea impartir, carente del análisis de datos que permita medir el cumplimiento de los objetivos propuestos por las partes involucradas en la asignación de los recursos [5].

Estos problemas pueden variar de gran forma entre las instituciones, y estas meta heurísticas son, por definición, tipos de algoritmos de “alto nivel” de propósito general que pueden ser usados en un amplio rango de diferentes tipos de problemas [3]

Para que este problema sea considerado de tipo NP o NP-Completo es necesario establecer la cantidad de factores y restricciones que se involucran en las exigencias horarias [6]. Llamado usualmente como Timetabling la asignación sujeta a restricciones de un conjunto de recursos a objetos específicos, siendo ubicados en espacios de tiempo, de tal forma que satisfaga en el mayor grado posible un conjunto de objetivos deseables.

Según Carter y Laporte [7] este problema se descompone en 5 subproblemas de los cuales (Classroom Assigment) o asignación de aulas tendra cabida en este proyecto, pues consiste en asignar las sesiones a las salas con un horario ya determinado.

2.1 ESPECIFICACION DEL PROBLEMA

2.1.1 Tipo de problema

La clase de problemas que se pueden resolver mediante algoritmos eficientes se denota como P, indicando tiempo polinomial. Se considera que este tipo de problemas se pueden resolver eficientemente, y se encuentra una solución exacta en tiempo razonable según González [8].

El problema de asignación de cursos y profesores en relación a la eficiencia computacional se clasifica como un problema NP- completo. Pero la asignación de aulas se presenta como un problema de menor complejidad computacional.

2.1.2 Restricciones

La programación de cursos en las instituciones educativas universitarias es un aspecto muy importante a tener en cuenta y el problema de asignación de aulas a los cursos requiere un especial cuidado en cuanto el buen manejo de los recursos físicos de la entidad, pues es necesario cumplir con todos los objetivos propuestos en la asignación cursos para evitar cualquier tipo de inconsistencias al momento

de ejercer los horarios establecidos en cada uno de los recintos programados para tal fin.

En busca de obtener un resultado óptimo se deben declarar un set de restricciones en las cuales basarse para el desarrollo de la asignación. Algunas de estas restricciones son sugeridas por Corne, Ross, and Fang [9] y categorizadas de la siguiente manera:

Restricción Única: se refiere a un solo evento que se representa como restricción “el evento a no se puede programar los jueves”, o la restricción “el evento a de suceder en el slot b”.

Restricción Binaria: Concierne a un par de eventos, donde la restricción “el evento a debe suceder después del evento b”.

Restricción de capacidad: Todos los eventos deben ser asignados en aulas con la suficiente capacidad.

Restricción de evento de extensión: concierne a los requerimientos donde los eventos de extensión o agrupación de estudiantes son necesarios para hacer más fácil el flujo de trabajo de maestros o estudiantes, o para coincidir con las políticas de programación de la universidad.

Restricción por agentes: son aquellas restricciones impuestas por las preferencias de las personas que necesitan especificar en qué tiempo pueden o no, ser utilizados los recursos.

De otra manera burke[2] también expresa la necesidad de categorizar las limitaciones o restricciones que se deben tener en cuenta en un caso u otro para la debida asignación y programación de aulas, las 7 categorías de burke son:

- Restricciones de asignación de recursos. Limitan la asignación de un recurso a un evento o recurso de otro tipo. Por ejemplo, los laboratorios deben ser asignados en salones con características particulares.

- Restricciones asociadas a la asignación en el tiempo. Establecen que un evento, sesión de clase por ejemplo, debe ser asignado a un periodo especifico del tiempo.

- Restricciones asociadas a la relación temporal entre eventos. Establecen relaciones de precedencia o simultaneidad entre eventos.

- Distribución de los eventos. Establecen la distribución apropiada en el tiempo de los diferentes eventos. Por ejemplo, los estudiantes no deben tener más de dos clases consecutivas.

- Coherencia de los eventos. Este tipo de restricciones operan en sentido opuesto a la anterior y busca programaciones mas ordenadas y compactas. Por ejemplo, los profesores preferirían dictar sus clases en un mismo día.

- Capacidad de las Aulas.

- Continuidad. Establecen que ciertas características de la programación permanezcan constantes. Por ejemplo, todas las sesiones de clase de un grupo de cierta asignatura, deben dictarse en la misma aula.

En particular encontramos una restricción general que especifica que no se deben encontrar cruce alguno entre los cursos, profesores o aulas, específicamente un aula no debe ser programada para dos grupos en un mismo horario.

Las restricciones, sin importar las categorías en las que son clasificadas, son usualmente divididas en dos grandes grupos: restricciones fuertes y restricciones débiles.

Las restricciones fuertes tiene una prioridad más alta y su cumplimiento es obligatorio de no serlo esta solución no se considera factible. A este subconjunto de restricciones pertenecen aquellas que evitan la asignación de unos recursos a más de un evento o un lugar en un mismo periodo de tiempo, como [10]:

- Conflicto de aula: Situación en la cual más de una actividad ha sido asignada para realizarse en dicha aula durante un mismo periodo de tiempo.

Las restricciones débiles [21] son aquellas cuyo cumplimiento es deseable pero no esencial, son satisfechas de ser posible y en la mayoría de los problemas reales no pueden ser cumplidas todas simultáneamente; el incumplimiento de este tipo de restricciones penaliza la función objetivo pero las soluciones generadas son factibles; dentro de este grupo se encuentran las restricciones que pretenden la mejor distribución de los eventos, el cumplimiento de las preferencias de profesores y la generación de horarios lo más compactos posibles.

2.1.3 Procedimiento de Programacion de cursos UIS

El procedimiento de asignación de aulas a grupos se ve reflejado en varias etapas de la programación de cursos en la Universidad Industrial De Santander, a continuación se explicara en breve las etapas que se llevan a cabo para cumplir con éxito la programación y asignación de grupos para un semestre.

Oferta

En cada una de las escuelas los directores tienen la tarea de hacer un calculo aproximado de la cantidad de grupos que será necesario ofertar para cada materia, estos calculos se basan en la cantidad de grupos ofertados el semestre anterior, la disponibilidad del profesorado, y los niveles de mortandad[1] de cada materia.

En está etapa el director de escuela debe organizar y calendarizar los grupos ofertados asignando profesor, espacio de tiempo y aula, evitando cualquier tipo de cruce, todo este procedimiento se realiza de manera manual, lo que hace de está tarea algo tedioso, dificil de realizar y puede tardar alrededor de un mes.

Luego de organizar y calenradizar toda la información de manera manual en pliegos de papel, cada escuela debe hacer la actualización pertinente en los sistemas de la universidad para que se pueda proceder con el periodo de demanda vía web.

Demanda

El periodo de demanda está dispuesto para que cada estudiante realice su matrícula vía web.

Ajuste

[1] Refiérase al promedio de estudiantes que no lograron aprobar materias.

Luego de finalizar el proceso de demanda por parte de los estudiantes, se precisa determinar si los grupos ofertados fueron suficientes o si existe la necesidad de borrar o crear más grupos.

Cabe indicar que al terminar el periodo de demanda no todos los grupos usados tienen un aula asignada y ubicar estos grupos evitando cualquier tipo de cruce es una tarea que conlleva mucho tiempo, dedicación, y es una de las principales razones de ser de este proyecto.

2.1.4 Estrategias de solución

Tipos de asignación

Para poder expresar cual es la metodología que se desarrollo para el uso de la herramienta primero debemos describir los tipos de asignación que serán utilizados y por cuales actores pueden ser ejecutados.

- **Asignación Total en Aulas del Campus:** Este tipo de asignación solo se puede ejecutar por la dirección de admisiones, realiza una asignación de aulas sin discriminación de escuela o edificio a todos los grupos que no tengan asignada aula vigente**.**

- **Asignación Total en Aulas de Escuela:** Asignación ejecutada solo por la dirección de admisiones, realiza una asignación de aulas de escuela a grupos de la misma escuela que no tengan un aula vigente asignada, este procedimiento se realiza a nivel general para todas las escuelas.

- **Asignación parcial de Aulas del Campus:** Este tipo de asignación busca cumplir con la asignación de aulas para los grupos de una escuela específica a cualquier aula o edificio dentro del campus. Este procedimiento solo lo puede realizar la dirección de admisiones.

- **Asignación parcial de Aulas de Escuela:** La asignación de aulas de escuela a grupos de la misma escuela para una escuela específica, realizada por la dirección de admisiones.

- **Asignación local total:** Este tipo de asignación es realizado por la dirección de escuela, realiza una asignación de aulas a los grupos ofertados o a aquellos grupos que no tuvieron una asignación de aula manual.

Para poder realizar una asignación automática de aulas cada uno de los directores de escuela debe autorizar de manera pertinente las aulas que pueden entrar en el proceso de asignación, y hacer la reserva de aquellas aulas que usará para fines específicos durante el semestre[2], para tal fin, es necesario desarrollar un modulo que preste tal función. Dicho modulo tiene cabida a finales del proceso de oferta y precisamente antes de correr cualquier algoritmo de asignación automática.

A continuación describiremos la metodología que será utilizada por parte de las directivas de admisiones y cada una de las direcciones de escuela para el uso apropiado de la herramienta de asignación automática

[2] A estas aulas se les da el nombre de aulas especiales de escuela.

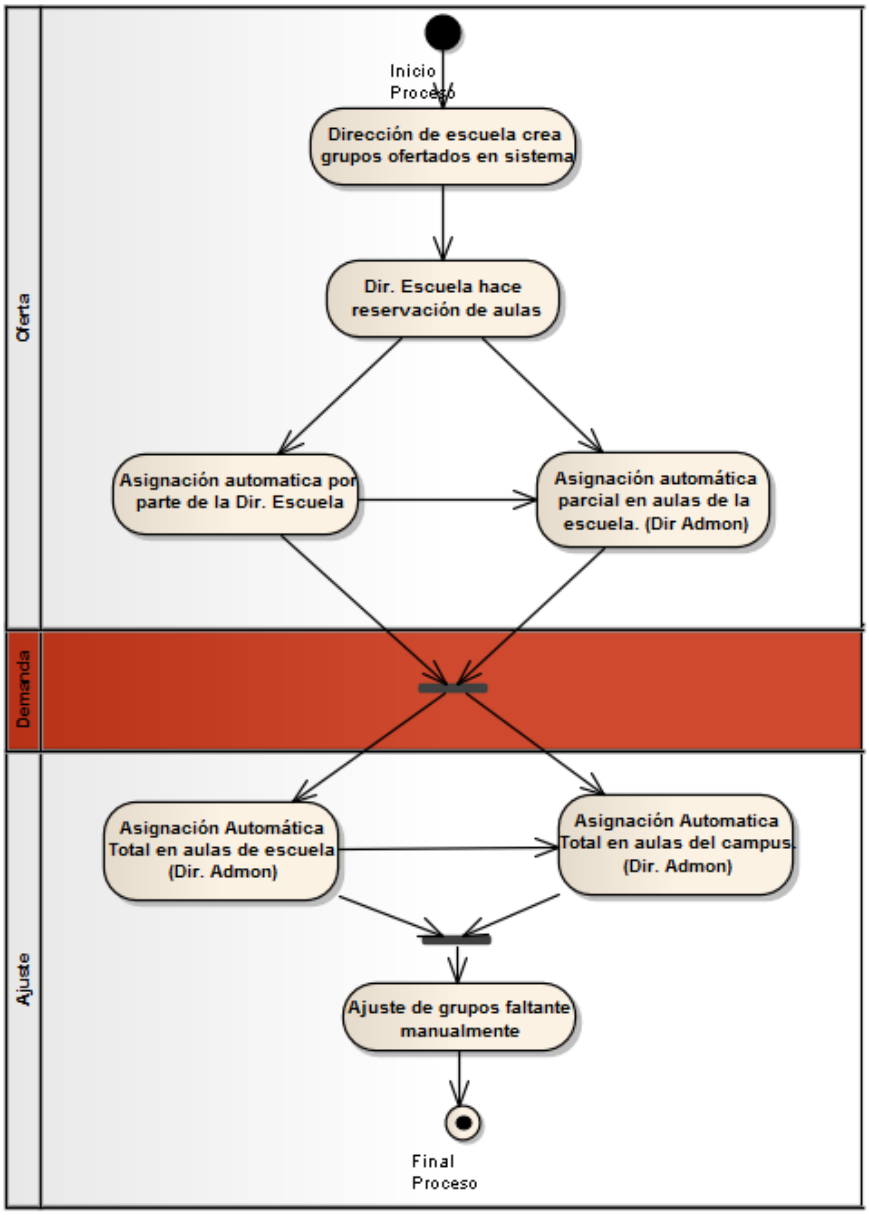


Figura: 1 Diagrama Actividades De los principales actores, en el proceso de asignación automática de aulas durante los periodos de oferta, demanda y ajuste

El anterior diagrama es el resultado de la necesidad de diseñar una metodología que cubra en su totalidad las diferentes necesidades de los principales usuarios del sistema de asignación automática de aulas, brindándoles las pautas a seguir en el transcurso del proceso de matrícula y sus distintas fases. El diseño de este modelo de actividades ha sido elaborado conjuntamente con el desarrollador en jefe de la División de Servicios de Información y la Dirección de Admisiones como principales interesados.

2.2 GENERALIDADES DEL ENTORNO DE DESARROLLO

2.2.1 Aplicaciones orientadas a la Web

En los primeros días de la Web, los sitios Web consistían de páginas estáticas, permitiendo una interacción limitada con el usuario. Al comienzo de los años 90, estas limitaciones fueron superadas cuando los servidores Web fueron reemplazados para permitir comunicaciones a través del desarrollo de fragmentos de código que eran ejecutados del lado del servidor. A partir de entonces las aplicaciones dejaron de ser estáticas y solamente editadas por aquellos "gurúes" del HTML y se permitieron a usuarios normales interactuar con las aplicaciones por primera vez.

Este fue un paso fundamental para llegar a la Web que hoy en día conocemos. Sin la interacción no existiría el comercio electrónico, el Web-mail , Internet-banking, blogs, comunidades online.

2.2.1.1 La Web como Sistema de Información.[3]

La evolución de Internet como red de comunicación global y el surgimiento y desarrollo de la Web como servicio imprescindible para compartir información, creó un excelente espacio para la interacción del hombre con la información hipertextual, a la vez que sentó las bases para el desarrollo de una herramienta integradora de los servicios existentes en Internet. Los sitios Web, como expresión de sistemas de información, deben poseer los siguientes componentes:

[3] Rodríguez Perojo Keilyn, Ronda León Rodrigo. El Web como Sistema de Información. http://bvs.sld.cu/revistas/aci/vol14_1_06/aci08106.htm

•Usuarios.

•Mecanismos de entrada y salida de la información.

•Almacenes de datos, información y conocimiento.

•Mecanismos de recuperación de información.

Se pudiese definir sistema de información como el conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su posterior uso, generados para cubrir una necesidad (objetivo). Dichos elementos formarán parte de alguna de estas categorías:

- Personas
- Datos
- Actividades o técnicas de trabajo.
- Recursos materiales en general (típicamente recursos informáticos y de comunicación, aunque no tienen por qué ser de este tipo obligatoriamente).

Todos estos elementos interactúan entre sí para procesar los datos (incluyendo procesos manuales y automáticos) dando lugar a información más elaborada y distribuyéndola de la manera más adecuada posible en una determinada organización en función de sus objetivos.

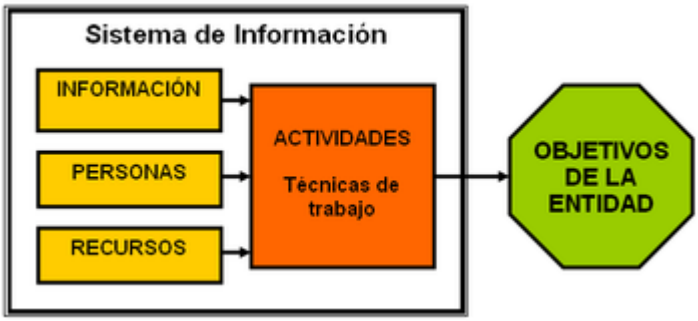


Figura: 2 Elementos de un Sistema de Información[4]

[4] Esquema de los elementos que componen un sistema de información / 2008-02-20 /Autor= Jesuja

Normalmente el término es usado de manera errónea como sinónimo de sistema de información informático, en parte porque en la mayoría de los casos los recursos materiales de un sistema de información están constituidos casi en su totalidad por sistemas informáticos, pero siendo estrictos, un sistema de información no tiene por qué disponer de dichos recursos (aunque en la práctica esto no suela ocurrir). Se podría decir entonces que los sistemas de información informáticos son una subclase o un subconjunto de los sistemas de información en general.

Actualmente, los sistemas de información se encuentran al alcance de las grandes masas de usuarios por medio de Internet; así se crean las bases de un nuevo modelo, en el que los usuarios interactúan directamente con los sistemas de información para satisfacer sus necesidades de información.

2.2.1.2. Tecnologías Web.[5]

Inicialmente, era difícil la construcción de aplicaciones sofisticadas. La primera generación de aplicaciones Web era primitiva, en general basada en formularios con información y aplicaciones de búsqueda. Incluso estas aplicaciones básicas requerían de un alto conocimiento para su construcción.

A través del tiempo, el conocimiento necesario para construir aplicaciones ha sido reducido. Hoy en día, es relativamente sencillo construir aplicaciones sofisticadas utilizando las modernas plataformas y lenguajes, como pueden ser PHP, .NET o Java.

[5] López Gustavo, Blanco Ignacio. Tesis en Ingeniería Informática. http://materias.fi.uba.ar/7500/blanco-tesisingenieriainformatica.pdf

Primera generación – CGI

Common Gateway Interface (CGI) fue la tecnología reinante desde aproximadamente 1993 hasta fines de los '90 cuando los lenguajes de scripting comenzaron a ganar importancia.

CGI trabaja encapsulando la información provista por el usuario en variables de ambiente. Estas luego son accedidas por scripts o programas desarrollados comúnmente en Perl o C. Estos programas procesan la información provista por los usuarios, y luego envían código HTML con la información procesada a la salida estándar, que a su vez es capturada por el servidor Web y pasada al usuario.

Scripting.

La falta de manejos de sesiones y control de autorización por parte de CGI impidió el desarrollo de aplicaciones Web comerciales con esa tecnología.

Los desarrolladores Web comenzaron entonces a utilizar lenguajes de script, como JavaScript o PHP para resolver esos problemas. Básicamente los lenguajes de script son ejecutados en el servidor Web y como son no compilados son desarrollados e implementados más fácilmente.

Los lenguajes de script tienen algunas desventajas:

- La mayoría de los lenguajes no son tipados y no promueven buenas prácticas de programación.
- Son más lentos en comparación con los lenguajes compilados (a veces hasta 100 veces más lentos).
- Es difícil (no imposible) escribir aplicaciones de múltiples capas porque en general las capas de presentación, aplicación y datos residen en la misma máquina, limitando de esta forma la escalabilidad y seguridad.

- La mayoría no soporta nativamente métodos remotos o llamadas a Web services, lo que hace difícil la comunicación entre servidores de aplicación y con Web services externos.

De cualquier manera a pesar de las desventajas aplicaciones grandes y frecuentemente accedidas han sido desarrolladas utilizando lenguajes de script, como eGroupWare (egroupware.org), que está escrita en PHP. Además muchas aplicaciones de Internet banking han sido desarrolladas en ASP.

Los lenguajes de script incluyen, ASP, Perl, Cold Fusion y PHP. De cualquier manera, muchos de esos podrían ser considerados como lenguajes interpretados híbridos, en particular las últimas versiones de PHP y Cold Fusion.

2.2.1.3 Plataformas de desarrollo web.

Una vez que los lenguajes de script alcanzaron los límites de performance y escalabilidad, los proveedores más grandes evolucionaron hacia la plataforma de Sun J2EE y cuya evolución se encuentra en JEE6 y la de Microsoft .NET.

J2EE

- Utiliza el lenguaje Java para producir aplicaciones Web.
- Permite la creación de grandes aplicaciones distribuidas.
- Provee un buen control de sesión y manejo de autorización.
- Permite la creación de aplicaciones de múltiples capas.

Una de las desventajas de J2EE es que posee una curva de aprendizaje importante, lo que provoca una difícil inserción de diseñadores Web y programadores en sus primeros pasos.

.NET

Microsoft actualizó su tecnología ASP a ASP.NET que imita a J2EE en muchas maneras.

- Simplifica la creación de aplicaciones pequeñas a programadores que se están iniciando y a diseñadores gráficos.
- Permite la creación de grandes aplicaciones distribuidas.
- Provee un buen control de sesión y manejo de autorización.
- Permite a los programadores la utilización de su lenguaje de programación favorito, el que es compilado a código nativo.

La elección entre J2EE y .NET es dependiente de la plataforma. Las aplicaciones J2EE teóricamente pueden ser ejecutadas en la mayoría de las plataformas, desde Linux a AIX, MacOS X o Windows.

2.2.1.4 Web 2.0.

El concepto original de la Web (en este contexto, llamada Web 1.0) eran páginas estáticas

HTML que no eran actualizadas frecuentemente. El éxito de las punto com dependía de webs más dinámicas (a veces llamadas Web 1.5) donde los CMS servían páginas HTML dinámicas creadas al vuelo desde una actualizada base de datos. En ambos sentidos, el conseguir hits (visitas) y la estética visual eran considerados como unos factores muy importantes.

Los propulsores de la aproximación a la Web 2.0 creen que el uso de la Web está orientado a la interacción y redes sociales, que pueden servir contenido que explota los efectos de las redes con o sin crear webs interactivas y visuales. Es

decir, los sitios Web 2.0 actúan más como puntos de encuentro, o webs dependientes de usuarios, que como webs tradicionales.

2.2.2 Diseños con el estándar UML.[6]

2.2.2.1 Introducción a UML

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido concebido por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh.

Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa: Booch, OMT y OOSE (Object-oriented software engineering). UML ha puesto fin a las llamadas “guerras de métodos” que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

[6] Ferré Grau Xavier, Sánchez S. María Isabel. Desarrollo Orientado a Objetos con UML. Facultad de Informática - UPM

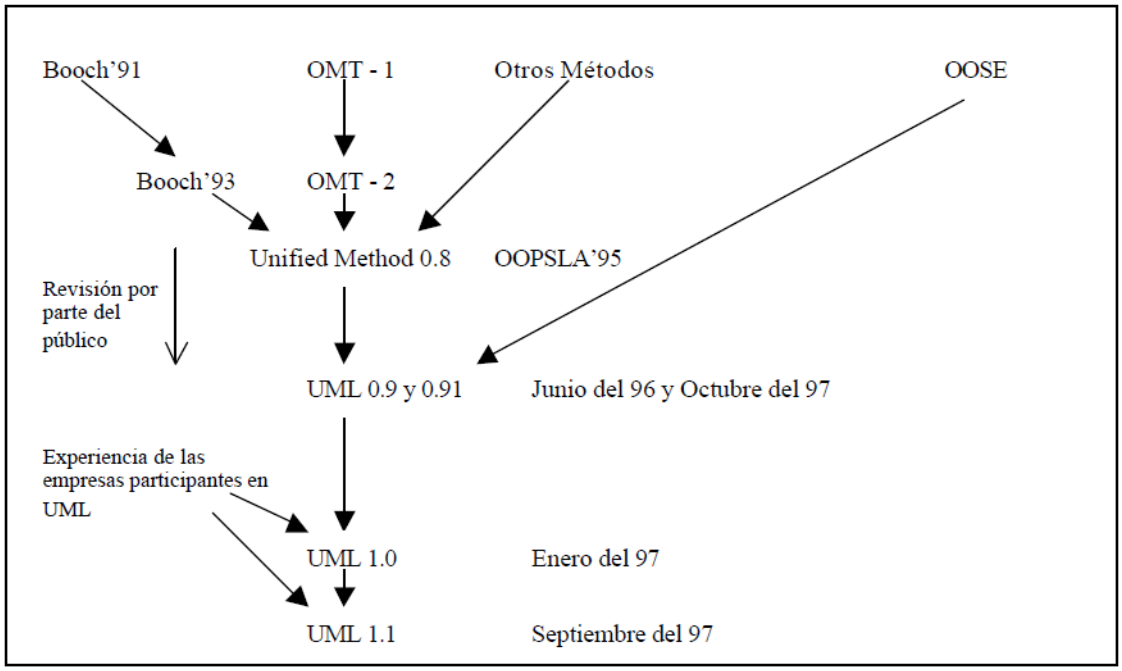


Figura: 3 Historia de UML

2.2.2.2 Elementos comunes a todos los diagramas.

Nota

Una nota sirve para añadir cualquier tipo de comentario a un diagrama o a un elemento de un diagrama. Es un modo de indicar información en un formato libre, cuando la notación del diagrama en cuestión no nos permite expresar dicha información de manera adecuada.

Una nota se representa como un rectángulo con una esquina doblada con texto en su interior.

Puede aparecer en un diagrama unida a un elemento por medio de una línea discontinua. Puede contener restricciones, comentarios, el cuerpo de un procedimiento, etc.

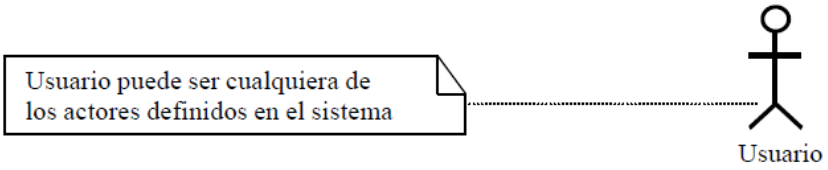


Figura: 4 Ejemplo de una Nota UML

Dependencias

La relación de dependencia entre dos elementos de un diagrama significa que un cambio en el elemento destino puede implicar un cambio en el elemento origen (por tanto, si cambia el elemento destino habría que revisar el elemento origen).

Una dependencia se representa por medio de una línea de trazo discontinuo entre los dos elementos con una flecha en su extremo. El elemento dependiente es el origen de la flecha y el elemento del que depende es el destino (junto a él está la flecha).

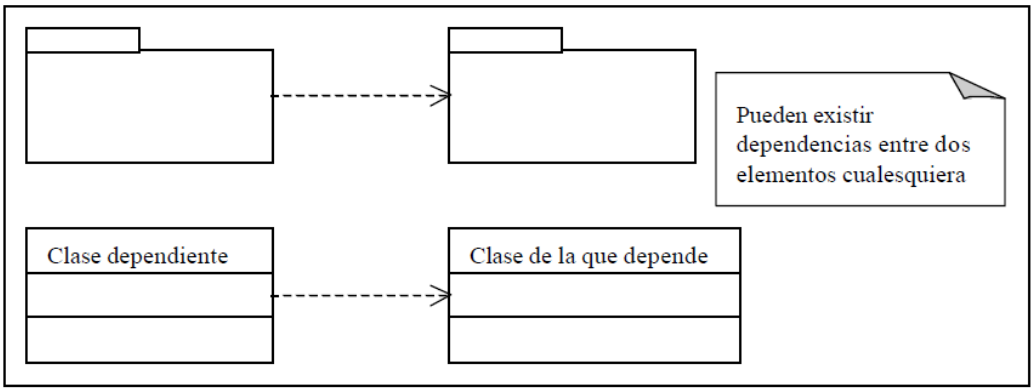


Figura: 5 Ejemplo de una Dependencia UML

2.2.2.3 Diagramas de Casos de Uso.

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

Elementos del diagrama de casos de uso.

Los elementos que pueden aparecer en un Diagrama de Casos de Uso son: actores, casos de uso y relaciones entre casos de uso.

Actores

Un actor es una entidad externa al sistema que realiza algún tipo de interacción con el mismo.

Se representa mediante una figura humana dibujada con palotes. Esta representación sirve tanto para actores que son personas como para otro tipo de actores (otros sistemas, sensores, etc.).

Casos de Uso

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el Diagrama de Casos de Uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

Relaciones entre Casos de Uso

Entre dos casos de uso puede haber las siguientes relaciones:

· Extiende: Cuando un caso de uso especializa a otro extendiendo su funcionalidad.

· Uso: Cuando un caso de uso utiliza a otro.

Se representan como una línea que une a los dos casos de uso relacionados, con una flecha en forma de triángulo y con una etiqueta <<extiende>> o <<uso>> según sea el tipo de relación.

En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea.

En la Figura se muestra un ejemplo de Diagrama de Casos de Uso para un cajero automático.

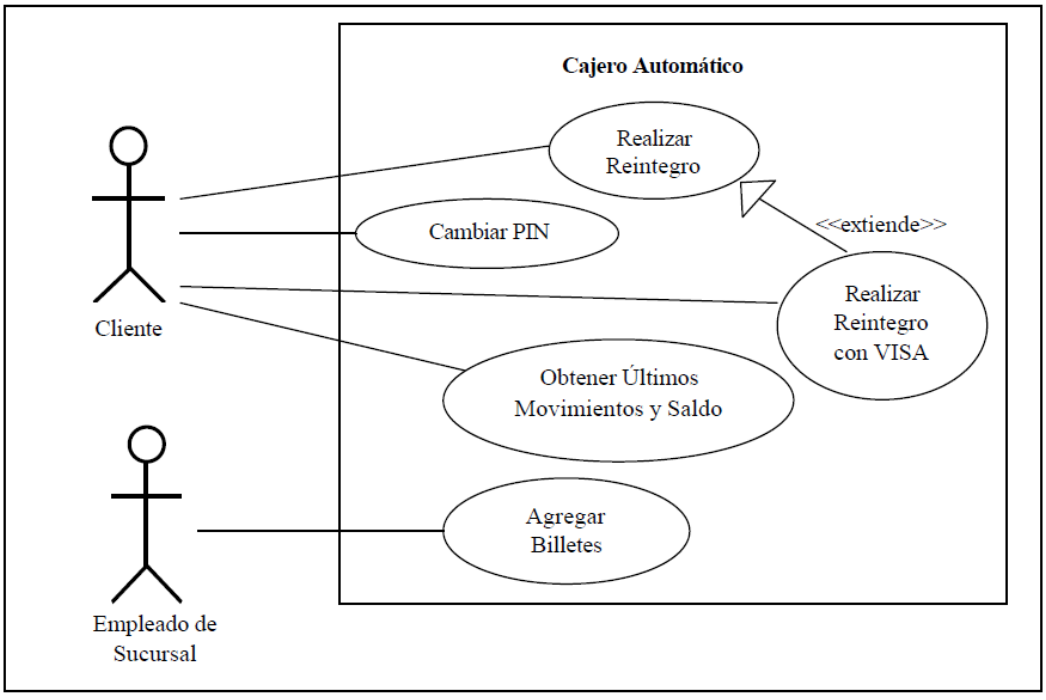


Figura: 6 Ejemplo de Diagrama de Caso de Uso

2.2.2.4 Diagramas de Secuencia.

Un diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo.

El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo.

Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

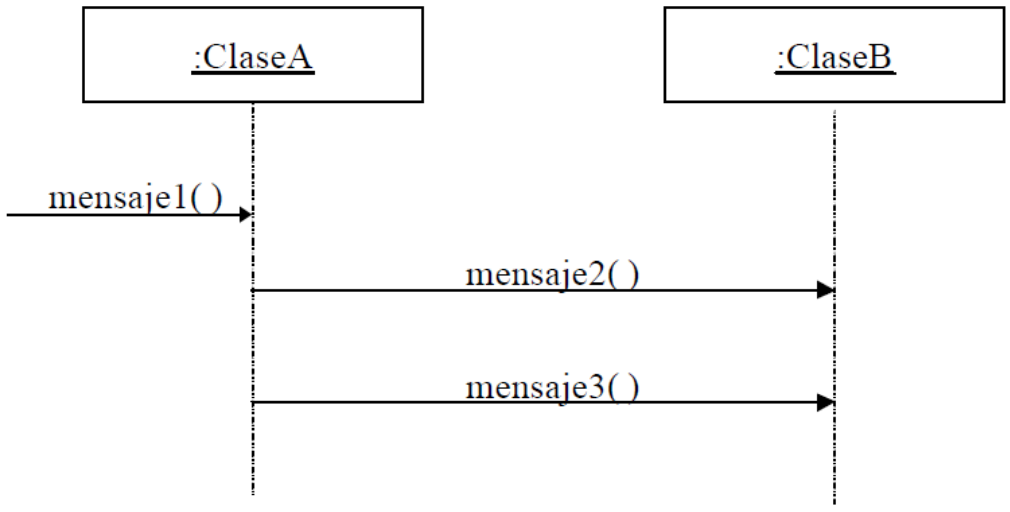


Figura: 7 Ejemplo de Diagrama de Secuencia

2.2.2.5 Diagrama de Actividades

La idea es generar una especie de diagrama Pert, en el que se puede ver el flujo de actividades que tienen lugar a lo largo del tiempo, así como las tareas concurrentes que pueden realizarse a la vez. El diagrama de actividades sirve para representar el sistema desde otra perspectiva, y de este modo complementa a los anteriores diagramas vistos.

Gráficamente un diagrama de actividades será un conjunto de arcos y nodos.

Desde un punto de vista conceptual, el diagrama de actividades muestra cómo fluye el control de unas clases a otras con la finalidad de culminar con un flujo de control total que se corresponde con la consecución de un proceso más complejo.

Por este motivo, en un diagrama de actividades aparecerán acciones y actividades correspondientes a distintas clases. Colaborando todas ellas para conseguir un mismo fin.

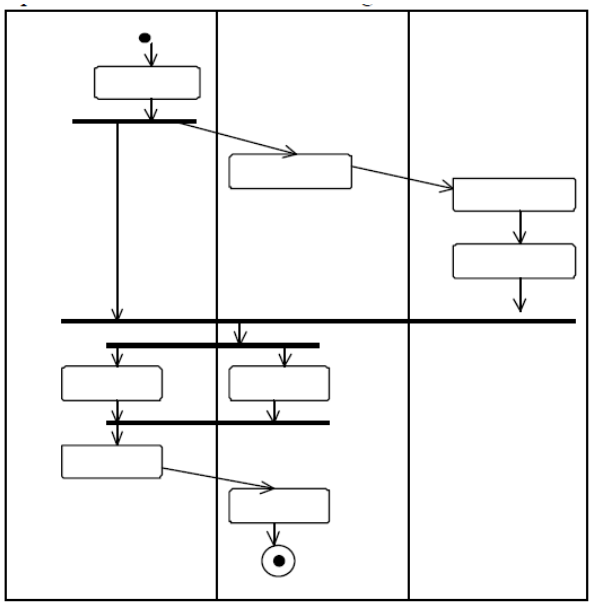


Figura: 8 Ejemplo Diagrama de Actividades

2.2.2.6 Diagrama de Clases[7].

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

Representación de: - Requerimientos en entidades y actuaciones. - La arquitectura conceptual de un dominio - Soluciones de diseño en una arquitectura - Componentes de software orientados a objetos

- Propiedades también llamados atributos o características, son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. Suponiendo que el objeto es una puerta, sus propiedades serían: la marca, tamaño, color y peso.
- Operaciones comúnmente llamados métodos, son aquellas actividades o verbos que se pueden realizar con/para este objeto, como por ejemplo abrir, cerrar, buscar, cancelar, acreditar, cargar. De la misma manera que el nombre de un atributo, el nombre de una operación se escribe con minúsculas si consta de una sola palabra. Si el nombre contiene más de una palabra, cada palabra será unida a la anterior y comenzará con una letra mayúscula, a excepción de la primera palabra que comenzará en minúscula. Por ejemplo: abrirPuerta, cerrarPuerta, buscarPuerta, etc.
- Interfaz es un conjunto de operaciones que permiten a un objeto comportarse de cierta manera, por lo que define los requerimientos mínimos del objeto. Hace referencia a polimorfismo.

[7] UML: Lenguaje de Modelado Unificado, Francisco mora 2002

- Herencia se define como la reutilización de un objeto padre ya definido para poder extender la funcionalidad en un objeto hijo. Los objetos hijos heredan todas las operaciones y/o propiedades de un objeto padre. Por ejemplo: Una persona puede especializarse en Proveedores, Acreedores, Clientes, Accionistas, Empleados; todos comparten datos básicos como una persona, pero además cada uno tendrá información adicional que depende del tipo de persona, como saldo del cliente, total de inversión del accionista, salario del empleado, etc.

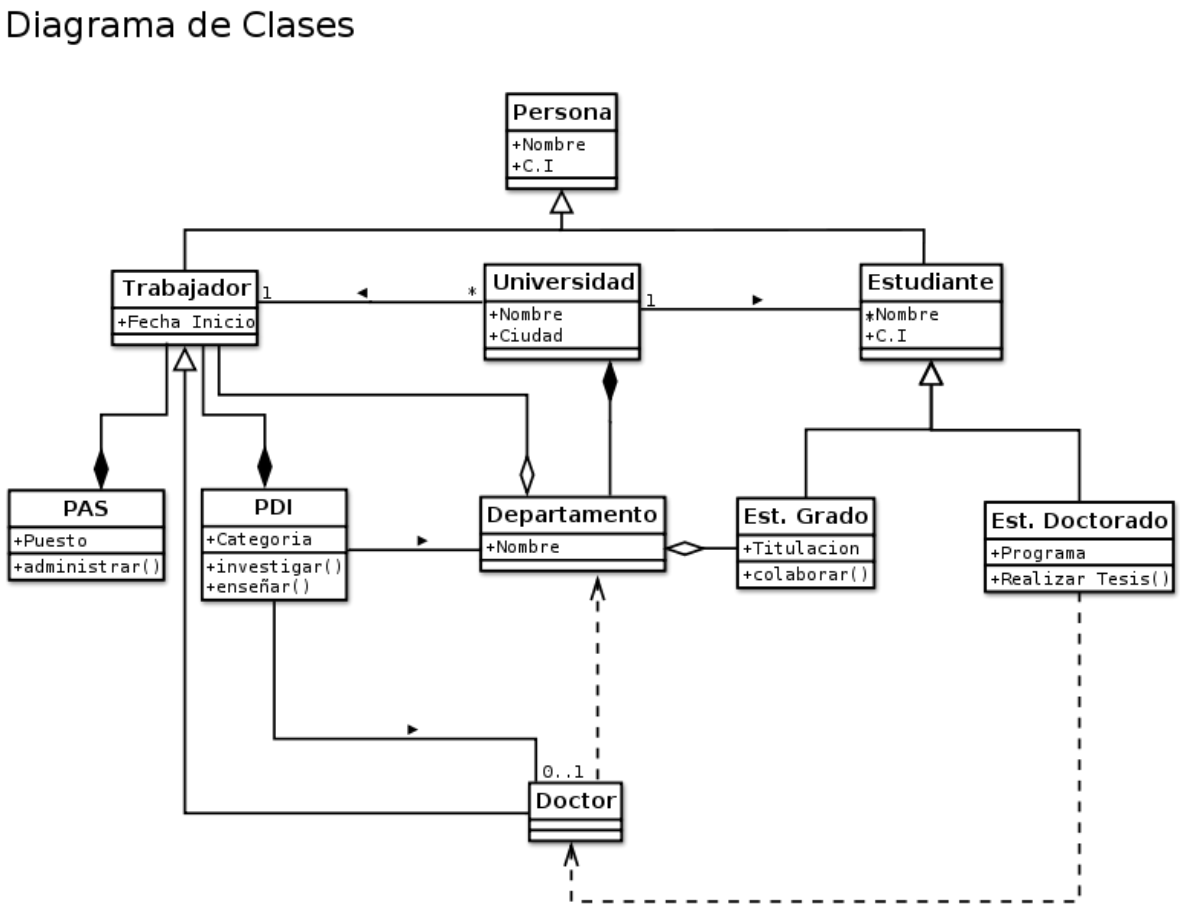


Figura: 9 Ejemplo de Diagrama de Clases

Al diseñar una clase se debe pensar en cómo se puede identificar un objeto real, como una persona, un transporte, un documento o un paquete. Estos ejemplos de clases de objetos reales, es sobre lo que un sistema se diseña. Durante el proceso del diseño de las clases se toman las propiedades que identifican como único al objeto y otras propiedades adicionales como datos que corresponden al objeto.

2.2.2.7 Diagrama Entidad – Relación[8]

Un diagrama o modelo entidad-relación (a veces denominado por su siglas, E-R "Entity relationship", o, "DER" Diagrama de Entidad Relación) es una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información así como sus interrelaciones y propiedades.

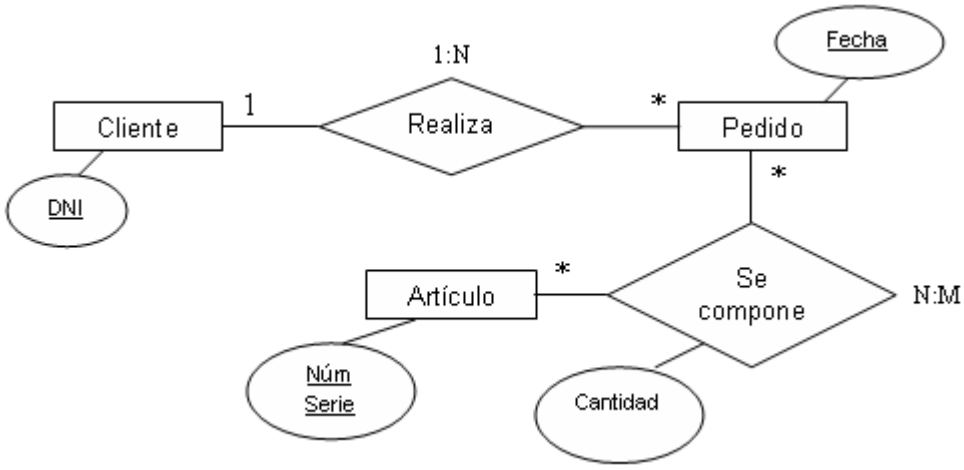


Figura: 10 Ejemplo Diagrama Entidad-Relación

[8]. UML: Lenguaje de Modelado Unificado, Francisco mora 2002

El Modelo Entidad-Relación.

- Se elabora el diagrama (o diagramas) entidad-relación.
- Se completa el modelo con listas de atributos y una descripción de otras restricciones que no se pueden reflejar en el diagrama.

Dado lo rudimentario de esta técnica se necesita cierto entrenamiento y experiencia para lograr buenos modelos de datos.

El modelado de datos no acaba con el uso de esta técnica. Son necesarias otras técnicas para lograr un modelo directamente implementable en una base de datos.

Brevemente:

- Transformación de relaciones múltiples en binarias.
- Normalización de una base de datos de relaciones (algunas relaciones pueden transformarse en atributos y viceversa).
- Conversión en tablas (en caso de utilizar una base de datos relacional).

2.2.3 Java Enterprise Edition 5.0 (JEE 5)

Introducción a Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de

ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

Java EE5

Java Enterprise Edition 5 (Java EE 5) se centra en hacer más fácil el desarrollo, sin embargo, conserva la riqueza de la plataforma J2EE 1.4. Ofrece funciones como Java Server Faces (JSF) y la tecnología de servicios web API, Java EE 5 hace que la codificación sea más simple y directa, pero mantiene el poder que ha establecido a Java EE como la primera plataforma para servicios web y desarrollo de aplicaciones empresariales.

El SDK de Java EE 5 y Java Application Platform SDK prestan apoyo a las especificaciones Java EE 5, y el SDK características adicionales, tales como tiempo de ejecución de Open ESB, Portlet Container, y Sun Java System Access Manager.

2.2.4 Entorno de Desarrollo

2.2.4.1 Java Server Faces (JSF)

Java Server Faces es una solución integral al problema de proveer una experiencia de usuario rica y a la vez sencilla en aplicaciones web. Para los desarrolladores de software, JSF provee una API estandarizada, fácil de usar y orientada a objetos, permitiendo crear interfaces de usuario con componentes reutilizables. Esto también repercute en la consistencia de tales interfaces, brindándole al usuario una experiencia de máxima calidad.

Su principal ventaja consiste en que el desarrollador sólo debe aprender el modelo de interfaces de usuario de JSF una sola vez, lo que le permitirá usar cualquier componente que cumpla los estándares de este modelo, aún si tales componentes provienen de terceros.

2.2.4.2 Object Relational Mapping / JPA

Más conocida por su sigla JPA, es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos, como sí pasaba con EJB2, y permitir usar objetos regulares.

2.2.4.3 EJB 3.0

Enterprise JavaBeans (EJB) es una arquitectura de componentes para la construcción de aplicaciones empresariales ejecutadas en servidores. Tiene por propósito proveer una forma estándar de implementar este tipo de aplicaciones, haciéndose cargo de aspectos comunes y repetitivos como la persistencia, la integridad transaccional y la seguridad, permitiendo que el desarrollador pase a preocuparse exclusivamente por la lógica del negocio en sí.

JBoss AS fue de los primeros servidores de aplicaciones en adoptar las especificaciones de EJB 3.0. Este modelo de EJB simplifica el desarrollo eliminando la necesidad de una interfaz “Home” y descriptores de despliegue, reemplazándolos por anotaciones. Facilita la implementación de la persistencia por medio del api JPA.

2.2.5 Servidor de Aplicaciones – Jboss

El servidor de aplicaciones JBoss es una herramienta certificada para el desarrollo de aplicaciones empresariales Java. Su madurez y el esfuerzo de muchos desarrolladores, e incluso las sugerencias que han realizado muchos usuarios, han permitido que JBoss AS (Application Server) se popularice ampliamente y sea común en los currículos de muchos desarrolladores. Encuestas recientes muestran que es el servidor de aplicaciones más popular actualmente.

Es reconocido por soportar los estándares más recientes. De hecho, es el primer servidor de aplicaciones en alcanzar la certificación J2EE 1.4 cuando salió su versión 4.0 (actualmente va en la versión 5, liberada a finales de 2008). Pero JBoss no sólo marca la pauta en la adopción de estándares con su servidor de aplicaciones, sino en la imposición de los mismos. Recientemente se le eligió para hacer parte del Java Community Process (JCP). Además en los últimos años ha estado a la cabeza del desarrollo de Java Enterprise llegando a establecerse en todas las especificaciones de requerimientos de Java (Java Specification Requests, JSRs).

2.2.6 Enterprise Architect

Enterprise Architect es una herramienta desarrollada por Sparx Systems que ofrece la capacidad de realizar el modelado de un proyecto y apoyar el desarrollo del mismo durante todo su ciclo de vida. Enterprise Architect logra esto usando UML, pero también puede generar código en varios lenguajes de acuerdo a algunos de estos diagramas.

Su valor como herramienta radica en la capacidad de permitir a los ingenieros desarrolladores comunicar sus ideas y su visión sobre los proyectos facilitando la administración y la redistribución de esta información.

2.2.7 Construcción de Prototipos de Software.

Un prototipo es una versión inicial de un sistema de software que se utiliza para demostrar los conceptos, probar las opciones de diseño y, enterarse más acerca del problema y sus posibles soluciones. Un prototipo de software apoya a dos actividades del proceso de ingeniería de requerimientos:

- Obtención de requerimientos: les permite adquirir nuevas ideas para los requerimientos y encontrar áreas fuertes y débiles del software.
- Validación de requerimientos. El prototipo puede revelar errores y omisiones en los requerimientos. La construcción de prototipos puede utilizarse como un análisis de riesgo y una técnica de reducción. Un modelo iterativo del proceso, como el desarrollo incremental, se utiliza junto con un lenguaje diseñado para el desarrollo rápido de aplicaciones. Por lo tanto, las técnicas utilizadas para desarrollar un prototipo para validar los requerimientos también se utilizan para desarrollar el sistema de software mismo.

Ventajas:

- Al demostrar las funciones del sistema se identifican las discrepancias entre los desarrolladores del software y los usuarios.
- Durante el desarrollo del prototipo el personal del desarrollo de software puede darse cuenta de que los requerimientos son inconsistentes y/o están incompletos.
- Se dispone rápidamente de un sistema que funciona y demuestra la factibilidad y usabilidad de la aplicación a administrar.

En sistemas grandes y complejos una forma de resolver la dificultad de evaluación es utilizar un enfoque evolutivo para el desarrollo de sistemas. Esto significa proporcionar al usuario un sistema incompleto y después modificarlo y aumentarlo en el momento en que los requerimientos del usuario sean claros.

El enfoque de construcción de prototipos desechables es para ayudar a refinar y clasificar la especificación del sistema. El prototipo se escribe, evalúa y modifica. La evaluación del prototipo informa del desarrollo de la especificación detallada del sistema que se incluye en el documento de requerimientos de este. Una vez que se ha redactado la especificación, el prototipo ya no es útil y se desecha.

Prototipo No funcional [9]

Es un modelo no funcional a escala configurado para probar ciertos aspectos de diseño. Un ejemplo de este enfoque es un modelo a escala completa de un automóvil que se usa para pruebas en un túnel de viento. El tamaño y forma del automóvil son precisos, pero el automóvil no es funcional. En este caso solo se incluyen las características del automóvil que son fundamentales para la prueba en el túnel de viento.

Un modelo no funcional a escala de un sistema de información podría producirse cuando la codificación requerida por las aplicaciones es demasiado extensa para incluirse en el prototipo, pero se puede conseguir una idea útil del sistema a través de la elaboración de un prototipo de la entrada y la salida. En este caso, el procesamiento, debido al excesivo costo y el tiempo requerido, no podría incluirse en el prototipo. Sin embargo, aún se podrían tomar algunas decisiones sobre la utilidad del sistema con base en la entrada y la salida incluidas en el prototipo.

[9] Fuente: Kendall, Julie E. Analisis y Diseño de sistemas. Capitulo 6. Página 153

3. METODOLOGÍA DE DESARROLLO

3.1 CICLO DE VIDA DEL PROYECTO.

A continuación se hace una descripción de las diferentes actividades que se llevaron a cabo durante el transcurso del proyecto, buscando conceptualizar la metodología que se aplicó en el desarrollo de la herramienta de Asignación automática de aulas para la Universidad Industrial de Santander.

3.1.1 Análisis de Requerimientos:

El análisis de requerimientos es la tarea que plantea la asignación de software a nivel de sistema y el diseño de programas.

Todo el proceso de análisis de requerimientos se especificó, junto con los funcionarios de la Dirección de escuela del departamento de Matemáticas, Ingeniería de Sistemas, El desarrollador en jefe de la División de Servicios de información y la Dirección de admisiones, las funcionalidades y comportamiento que debía tener la herramienta en el transcurso del proyecto, se definió la interfaz y se establecieron los estándares de diseño que el sistema debe mantener.

El análisis de requerimientos permite la representación de la información y las funciones que pueden ser traducidas en datos, arquitectura y diseño procedimental. Finalmente, la especificación de requerimientos suministra los medios para valorar la calidad de los programas, una vez que se haya construido.

En esta etapa se hicieron reuniones periódicas con los funcionarios de la Dirección de escuela, para que fuesen ellos los que definan las necesidades del software que se desea implantar y que se mantenga dentro de los parámetros.

3.1.2 Diseño

El diseño del software es realmente un proceso de muchos pasos que se centra en cuatro atributos distintos de programa: estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental (algoritmo).

En esta etapa de diseño se hace una traducción de los requisitos a una representación del software donde se pueda evaluar su calidad antes de que comience la codificación.

El diseño se efectuó, mediante modelos UML (Lenguaje de Modelado Unificado) dónde se incluyeron los Diagramas de Clases, de Actividades, de Entidad-Relación, utilizando la herramienta Enterprise Architect licenciada por la Universidad Industrial de Santander a través de División de Servicios de Información.

3.1.3 Implementación de la Aplicación.

El diseño se debe traducir en una forma legible por la máquina. El paso de generación de código se llevó a cabo en esta tarea.

En esta etapa se procede a generar el software que se ha diseñado. Se realizó teniendo en cuenta los parámetros establecidos por la División de Servicios de Información en cuanto a los estándares técnicos y de calidad que caracterizan las aplicaciones que son generadas para el servicio de la Universidad, teniendo como base el Lenguaje de programación JAVA 5, frameworks como: seam, java server faces (JSF), Enterprise Java Beans (EJB 3.0) e Informix como motor de base de datos.

3.1.4 Pruebas del software.

Son los procesos que permiten verificar y revelar la calidad de un producto software.

Las pruebas de software se integran dentro de las diferentes fases del ciclo de desarrollo del software establecidas en la ingeniería de software.

Una vez generado el código, comienzan las pruebas, proceso utilizado para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. Las pruebas se centran en los procesos lógicos internos del software, asegurando que todas las sentencias sean probadas y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos.

Estas pruebas se aplicaron de forma permanente a lo largo del desarrollo del proyecto, y se abrió un espacio dónde los usuarios finales interactuaron con la aplicación con el objetivo que detectaran posibles fallos que hayan sido omitidos en el momento del desarrollo.

3.1.5 Ajustes.

Es el proceso de mejora del sistema tomando como base las sugerencias y observaciones que se plantearon en el periodo de pruebas, aquí también se incluyen las correcciones a los posibles fallos detectados, mejoras en la interfaz y el diseño de la misma.

Este proceso está incluido dentro del proceso iterativo de refinamiento que se le dio al sistema, para adaptarse plenamente a las necesidades del cliente que en el presente caso es la Dirección de admisiones y cada una de las Direcciones de escuela.

3.2 METODOLOGIA DE DESARROLLO DEL PROYECTO.

3.2.1 Modelo de Construcción de Prototipos.

Teniendo como base las actividades anteriormente descritas, se dispuso como metodología de desarrollo el MODELO DE CONSTRUCCION DE PROTOTIPOS.

Se eligió esta metodología debido a que los usuarios finales como lo son cada una de las Direcciones de escuela y la Dirección de Admisiones, definan un conjunto de objetivos generales para el software, pero no identifican los requisitos detallados de entrada, proceso o salida.

En otros casos, el responsable del desarrollo del software puede no estar seguro de la eficacia de un algoritmo, o no haber comprendido plenamente el requerimiento del usuario.

Para éstas y otras muchas situaciones, un paradigma de construcción de prototipos puede ofrecer el mejor enfoque, ya que la entrega de prototipos, que hacen parte integral del proyecto en su conjunto, permitirán la corrección temprana de errores o la redefinición del sistema en caso de ser necesario, y los prototipos tanto funcionales como no funcionales permitirán la familiarización del usuario con el sistema que se está desarrollando.

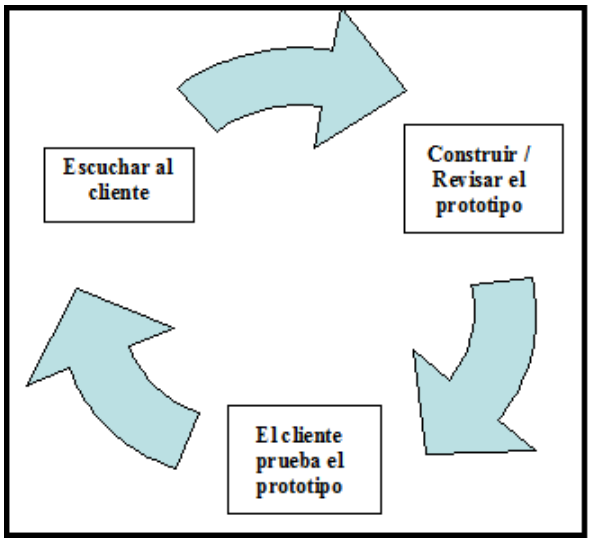


Figura: 11 Modelo de Construcción de Prototipos

3.2.2 Estructura del modelo de construcción de prototipos.

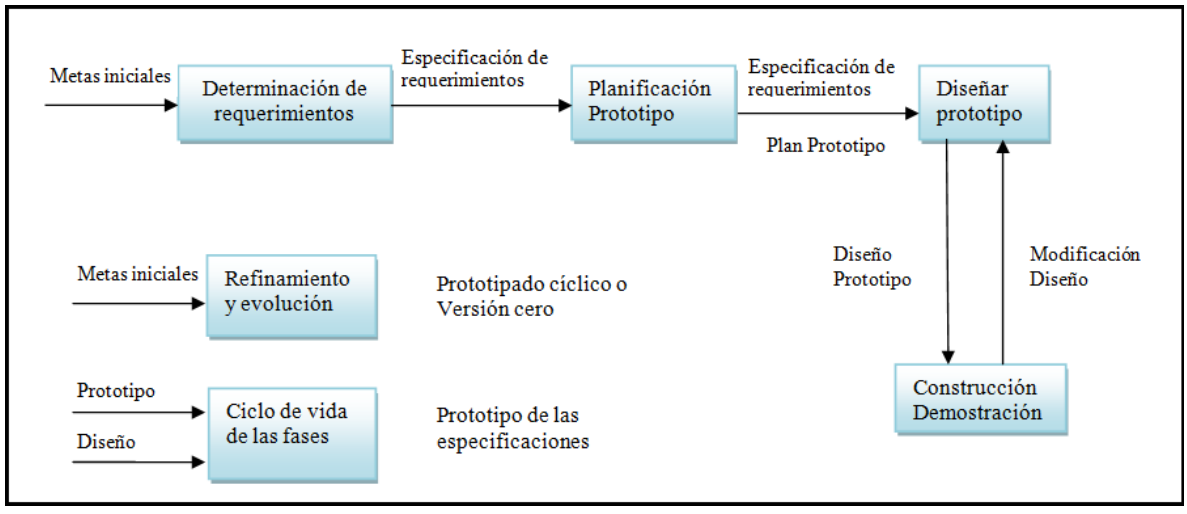


Figura: 12 Estructura del Modelo de Construcción de Prototipos

Ésta metodología planteada, permitió que el proyecto se llevara a cabo de manera eficiente, debido a:

- En la creación de los prototipos iníciales se tomaron en cuenta los requerimientos generales o principales para cubrir la necesidad de dar a comprender a los usuarios finales el comportamiento de la herramienta, esto dio como resultado un producto o prototipo inicial, el cual evolucionó dando como resultado un prototipo más maduro, que cumple con todos los requerimientos del cliente.
- Con el uso del modelo de prototipos se dio la facilidad de mejorar, de manera temprana los prototipos, teniendo en cuenta las sugerencias del usuario solicitante del proyecto, de manera que se pudieran cubrir a cabalidad las necesidades por las cuales se desarrolló el proyecto.
- Es importante conocer de forma temprana si los diferentes prototipos de la herramienta, cumplen a satisfacción con los requerimientos expuestos por los funcionarios de la Dirección de admisiones y las Direcciones de escuela.
- En este sistema de desarrollo no se libera un prototipo sin haber realizado todas las pruebas necesarias del mismo, pero los inconvenientes resultan inevitables y se da la facilidad de detectarlos y corregirlos de manera oportuna.

3.2.3 Procedimiento para la metodología planteada.

a. La construcción de prototipos comienza con la recolección de los requisitos. La Dirección de Admisiones y Direcciones de escuela de la Universidad Industrial de Santander (quienes son los usuarios administradores del sistema) cuentan de manera verbal lo que desean alcanzar con la nueva versión y los problemas de la actual versión que inevitablemente deben ser subsanados.

b. El desarrollador y usuario se reúnen y definen los objetivos globales para el software, identifican todos los requisitos conocidos y perfilan las áreas en donde será necesaria una mayor definición.

c. Teniendo la abstracción del sistema por parte del desarrollador, se realiza todo el diseño de los objetos mediante los diagramas UML: Casos de uso, de clases, de secuencia, entidad – relación y actividades.

d. Luego se produce el Diseño del Prototipo que se enfoca sobre la representación de los aspectos del software visibles al usuario (por ejemplo, métodos de entrada y formatos de salida), en ésta etapa se presenta al usuario el diseño tanto de los diagramas UML como el prototipo NO funcional, para que éste lo apruebe y se adecúe plenamente a su solicitud.

e. Se procede con la construcción del prototipo que se ha propuesto.

f. El prototipo es evaluado por el usuario y se utiliza para refinar los requisitos de tal forma que el usuario identifique aspectos que deban ser replanteados o mejorados.

g. Se produce un proceso iterativo en el que el prototipo es “afinado” (Refinamiento del prototipo) para que satisfaga las necesidades del usuario, al mismo tiempo que facilita al que lo desarrolla una mejor comprensión de lo que hay que hacer y poder entregar el producto final requerido.

PARTE II. DESARROLLO DEL SISTEMA.

4. APLICACIÓN DE LA METODOLOGÍA

4.1 LEVANTAMIENTO DE REQUERIMIENTOS

Se presentan los requerimientos, generados después de las primeras reuniones con el cliente, en donde se pudo abstraer la idea principal de lo que se quiere alcanzar durante el desarrollo del proyecto, que sirvió como base para las primeras etapas del análisis y diseño del nuevo sistema de asignación automática de aulas.

4.1.1 Descripción General.

En vista de la evidente necesidad de remplazar el sistema de asignación manual de aulas, la concepción de este proyecto debe encontrar la forma de suplir a cabalidad esta necesidad, dándole la posibilidad a los usuarios de realizar dicha tarea de forma ágil y sin mayor complicación.

Para suplir dicha necesidad se debe generar una contextualización optima del dinamismo completo que se lleva a cabo en la actualidad para la realización de el proceso de asignación de aulas, teniendo en cuenta los distintos entes o entidades que intervienen en el proceso, la manera en que lo hacen y en que periodos de tiempo. De esta forma poder definir un diagrama de estados y actividades en el que se puedan involucrar todos los usuarios finales y brinde un mayor entendimiento de la metodología del proceso y las diferentes necesidades que se puedan presentar a nivel general.

La generación del diagrama de estados y actividades presentado en la *Figura 1*, detalla algunos conceptos o reglas que se deben tener en cuenta para el desarrollo de la herramienta tales como:

- Es necesario darle la posibilidad a la dirección de escuela como ente administrativo sobre los recursos que le fueron asignados, de hacer cualquier tipo de reserva sobre las aulas que están a su disposición para la totalidad del periodo académico, antes de ejecutarse cualquier proceso de asignación automática, con el fin de preservar la disponibilidad de aulas especiales para fines específicos como clases extracurriculares, de maestría, doctorado, u otras actividades.

- Darle a conocer a la dirección de escuela el listado de asignaturas que intervienen en el proceso de asignación automática y cuáles no. Pues es necesario que aquellas asignaturas que no tienen una relación directa con un tipo de aula especifico tengan una asignación manual.

- Permitirle a la Dirección de admisiones tener control sobre la asignación total de aulas a nivel de todo el campus universitario, logrando definir el tipo de asignación según lo requiera: Asignación de aulas de escuela a grupos de escuela o Asignación de aulas del campus a grupos de escuela. Para todas las escuelas en una sola iteración.

- De igual manera también es necesario darle control a la Dirección de admisiones sobre la asignación automática de aulas de manera parcial sobre cada una de las escuelas.

- Luego de cada ejecución del sistema de asignación automática el usuario podrá analizar el listado de horas grupo que no tuvieron lugar a asignación,

pudiendo revisar si existe algún espacio en el que se pueda asignar cada una de estas horas a nivel de sede.

- El usuario debe tener la posibilidad de ver el listado de horarios grupo que tuvieron una exitosa asignación por medio del sistema, con la respectiva información de la asignación.

- Luego de terminado el proceso de asignación automática el usuario tiene la posibilidad de analizar las estadísticas de la asignación en cuanto disponibilidad de salones y las necesidades de cada escuela.

Con la anterior información determinamos los requerimientos del sistema en cuanto a dinamismo y prestación de servicios, por otra parte es necesario determinar el nivel de seguridad del sistema, el nivel de acceso de cada uno de los usuarios y demás información que pueda ser extraída de la ejecución de la herramienta.

4.1.2 Métodos Meta-Heurísticos

La idea más genérica del término *heurística* está relacionada con la tarea de resolver inteligentemente problemas reales usando el conocimiento disponible. En Investigación Operativa, el término heurístico se aplica a un procedimiento de resolución de problemas de optimización con una concepción diferente. Se califica de *heurístico* a un procedimiento para el que se tiene un alto grado de confianza en que encuentra soluciones de alta calidad con un coste computacional razonable, aunque no se garantice su optimalidad o su factibilidad.

Las heurísticas para resolver un problema de optimización pueden ser más *generales o específicas* que otras. Los métodos heurísticos específicos deben ser diseñados a propósito para cada problema, utilizando toda la información

disponible y el análisis del modelo, por el contrario las heurísticas más generales se caracterizan por su adaptabilidad, sencillez y robustez.

El término meta-heurísticas se obtiene de anteponer a *heurística* el sufijo *meta* que significa “más allá” o “a un nivel superior”. Las meta-heurísticas son estrategias inteligentes para diseñar o mejorar procedimiento heurísticos muy generales con un alto rendimiento.

En una primera aproximación, se ha decidido separar las meta-heurísticas en dos grandes bloque: trayectoriales y poblacionales. Las meta-heurísticas trayectoriales manejan en todo momento una sola solución y deben su nombre a que el proceso de búsqueda que desarrollan se caracteriza por una trayectoria en el espacio de soluciones. Es decir, partiendo de una solución inicial, generan un camino en el espacio de búsqueda mediante operaciones de movimiento. Dentro de estas meta-heurísticas se pueden destacar por su interés: búsqueda tabú, métodos multi-arranque, búsqueda local iterativa.

Las meta-heurísticas poblacionales implementan el proceso de búsqueda manteniendo simultáneamente un conjunto de soluciones Las meta-heurísticas basadas en poblaciones o meta-heurísticas poblacionales son aquellas que emplean un conjunto de soluciones (población) en cada iteración del algoritmo, en lugar de utilizar una única solución como las meta-heurísticas trayectoríales. Estas meta-heurísticas proporcionan de forma intrínseca un mecanismo de exploración paralelo del espacio de soluciones, se contemplan operadores para generar nuevas soluciones a partir de las existentes, y su eficiencia depende en gran medida de como se manipule dicha población [11]. Ejemplos: algoritmos evolutivos, genéticos y meméticos, búsqueda dispersa, optimización por colonia de hormiga.

Sin embargo, los tipos de meta-heurísticas también se pueden clasificar en función del tipo de procedimientos a los que hace referencia de esta manera existen meta-heurísticas de:

- ***Relajación:*** se refieren a procedimiento de resolución de problemas que utilizan relajaciones del modelo original (es decir modificaciones del modo que lo hacen más fácil de resolver), cuya solución facilita la solución del problema original.

- ***Constructivas:*** se orientan a los procedimientos que tratan de la obtención de una solución a partir del análisis y selección paulatina de las componentes que la forman.

- ***Búsqueda:*** guían los procedimientos que usan transformaciones o movimientos para recorrer el espacio de soluciones alternativas y explotar las estructuras de entornos asociadas.

- ***Evolutivas:*** enfocada a los procedimientos basados en conjuntos de soluciones que evolucionan sobre el espacio de soluciones [12]

4.1.3 Definiciones y siglas.

DA: Dirección de admisiones.

DSI: División de Servicios de Información.

UAA: Unidad Académica y Administrativa de la Universidad Industrial de Santander.

Asignatura: Cada una de las materias que se enseñan en el centro educativo y posee un código especifico.

Aula: Recinto, donde se imparten las actividades académicas dentro de la universidad.

Campus: Conglomerado de edificios pertenecientes a la universidad.

Hora Grupo: Horario establecido en un lapso de tiempo para un grupo de una asignatura en particular.

Grupo: Cada uno de los conjuntos que se programan semestralmente para las Asignaturas. Con un código y horarios de clase específicos.

Dirección de escuela: Ente administrativo encargado de la administración de una escuela o departamento UIS.

Director escuela: Persona encargada de dirigir una escuela o departamento en la UIS.

4.2 ESTÁNDARES DE LA DIVISIÓN DE SERVICIOS DE INFORMACIÓN.[10]

4.2.1 Aspectos Generales

Interfaz de desarrollo

El IDE de desarrollo a utilizar es el JBoss Developer Studio, el cual debe ser instalado en la carpeta por defecto del instalador.

Este y todos los programas necesarios se pueden descargar, por el personal autorizado, del equipo establecido para tal fin.

Servidor de aplicaciones

En cuanto al servidor de aplicaciones de desarrollo se debe utilizar el mismo que se encuentra en los servidores de producción y desarrollo, el cual debe ser instalado en la carpeta C:\ jboss-eap-5.0

JAVA

La versión del compilador de JAVA debe ser la 1.5, la cual debe ser instalada en el directorio C:\java1.5.

SEAM

La versión del seam a utilizar es la 2.1.2.GA. (jboss-seam-2.1.2.GA.zip).

Espacio de trabajo

El espacio de trabajo se debe crear en C:\workspace.

[10] Fuente: Estandares de la División de Servicios de Información de la Universidad Industrial de Santander.

El nombre del proyecto para los JPA debe estar conformado de la siguiente manera:

[Sistema]Entidades

Por ejemplo: AcademicoEntidades (La primera letra de cada palabra en mayúscula).

El repositorio para los JPA debe estar conformado de la siguiente manera:

[Sistema]JPA

Por ejemplo: AcademicoJPA (La primera letra de cada palabra en mayúscula).

El Server name en el IDE de desarrollo se debe llamar JBoss 5 y el nombre del JBoss Runtime Enviroment se debe llamar JBoss 5.

Servidor de versiones

Para su configuración se debe instalar en el JBoss Developer Studio los siguientes paquetes:

- subclipse-site-1.4.7
- ajdt_1.6.1a_for_eclipse_3.4
- org.tmatesoft.svn_1.2.1.eclipse

Una vez instalados se debe solicitar el nombre de usuario y la contraseña al Ingeniero encargado.

Cualquier inquietud acerca de la instalación y configuración del servidor de versiones, dirigirse con el Ingeniero encargado.

También se debe instalar el siguiente programa: TortoiseSVN-1.5.8.15348-win32-svn-1.5.5.msi para conectarse con el servidor de versiones de la documentación.

Plantillas, estilos, imágenes y formateador

Descargar del servidor de versiones de documentación las carpetas Estilos, Plantillas e Imágenes y copiarlas en la careta view de la aplicación.

Patrones a utilizar

Los patrones a utilizar corresponden a cada una de las capas implementadas:

- Capa de presentación: Modelo Vista Controlador, el cual es implementado por Java Server Faces (JSF).
- Capa de lógica de negocio: Session Façade
- Capa de persistencia: Entity Access Object (EAO)

Rich Faces

La versión a utilizar es la incluida en el jboss-seam-2.1.2.

Código HTML

Está prohibido el uso de etiquetas HTML en las páginas.

Anotaciones en la entidad

Las anotaciones en la entidad se deben colocar antes del método get correspondiente y no en la declaración del atributo.

Llaves compuestas

Se debe utilizar la anotación EmbeddedId, la cual obliga a declara un objeto del tipo de la llave dentro del EJB de entidad.

HashCode e Equals en EJB de entidad

Se debe utilizar únicamente los campos que conforman la llave primaria. En el caso de las llaves compuestas, el atributo que hace referencia a ésta.

JPA externos

Para utilizar los JPA externos (academico.jar, recursos-humanos.jar, etc), éstos deben copiarse en la carpeta raíz. En el caso de Windows C:\, para Linux \).

En cada uno de los archivos persistence.xml de su aplicación, se debe utilizar <jar-file>file:/jpa.jar</jar-file>.

Por ejemplo:

<jar-file>file:/academico.jar</jar-file>

<jar-file>file:/recursos-humanos.jar</jar-file>

<jar-file>file:/general-UIS.jar</jar-file>

Servicios

Para crear un servicio se debe tener en cuenta las siguientes reglas:

- La interfaz debe contener la anotación Remote
- La implementación de la interfaz debe contener la anotación RemoteBinding con su respectivo nombre jndi (igual al utilizado por la anotación Name). Por ejemplo:

@RemoteBinding(jndiBinding = "ConsultarGenerales")

Para utilizar el servicio se debe utilizar la anotación EJB indicando el nombre jndi. De acuerdo al ejemplo anterior sería:

@EJB(mappedName = "ConsultarGenerales")

4.2.2 Documentación de los Diagramas de Diseño

Casos de Uso.

Para el desarrollo del modelo de casos de uso, se debe realizar un diagrama de casos de usos por módulo del sistema a implementar siguiendo el estándar propuesto por el Lenguaje Unificado de Modelado 2.1 (UML). Se deben tener en cuenta los siguientes puntos:

Identificación de Actores

Se identifican con el rol que desempeñan en el sistema.

Diagrama de Casos de Uso

El diagrama de casos de uso se tiene que realizar con la herramienta Enterprise Architect. Cada caso de uso constituye un flujo completo de eventos especificando la interacción que toma lugar entre el actor y el sistema.

Casos de Uso

Se deben identificar con una acción. Los verbos que se pueden utilizar se encuentran al final del documento.

La información mínima requerida por cada caso de uso es la siguiente:

- Descripción completa
- Precondiciones y postcondiciones
- Descripción del escenario básico y alternos
- Diagrama de clases
- Si el caso de uso es complejo se debe incluir:
- Diagrama de secuencia y/o diagrama de actividades

4.2.3 Sintaxis de Nombres en Java

Reglas de sintaxis generales

- En esta sección se especifican las reglas de sintaxis generales para todos los identificadores (nombres de variables, clases, métodos, etc.)
- Todos los nombres de los identificadores deben estar en español.
- Siempre se deben utilizar nombres que sean claros, concretos y libres de ambigüedades. Usando palabras completas evitando acrónimos y abreviaturas.
- Los nombres deben estar definidos sin espacios en blanco, sin guiones (_ , -), ni comillas (“,’), sin operadores (+, - , / , *), sin tildes, utilizar la n en vez de la ñ y sin caracteres especiales.
- No se debe utilizar la mayúscula para diferenciar entre identificadores distintos. Ejemplo: contador y Contador.
- No se deben diferenciar dos identificadores solo con numerales en cualquier posición. Ejemplo: contador1, contador2, 1contador, 2contador.
- Las siguientes partículas están prohibidas en la declaración de los nombres de identificadores: artículos (el, la, los, unos, unas, un), determinantes demostrativos (este, ese, aquel, aquellos), cardinales (uno, dos, etc.), pronombres de cualquier tipo (yo, tu, el, me, te, se, este, ese, mi, tu, su, etc.).
- Se deben utilizar máximo 5 palabras por nombre, las 3 primeras palabras van completas, a partir de la cuarta palabra se quitan las vocales a la palabra exceptuando la última vocal y la primera si la palabra empieza por vocal. No se utiliza ningún separador entre las palabras, se separa cada palabra utilizando su primera letra en mayúscula. Ejemplo: hacerMantenimientoConsultaUsros, hacerMantenimientoAsignaturasCntxto.

Clases

- Los nombres de las clases deben iniciar siempre en mayúscula, deben ser simples y descriptivos.
- Los nombres de las clases de Entidad deben ser sustantivos en singular.
- Para las clases de Entidad siempre se debe definir la anotación @Table que indica la tabla de la base de datos relacionada para su persistencia. Además se debe utilizar el parámetro name de la anotación @Entity para identificar el EJB (@Entity(name = “xxx”)). El nombre de la clase puede ser distinto al nombre de la tabla y debe seguir el estándar de identificadores mencionado en el numeral anterior.
- Los nombres de los EJB utilizados por el patrón Session Façade está conformado por un verbo autorizado (Ver anexo de verbos). Además debe incluir al final las letras “EJB”. Ejemplo: RegistrarMatriculaEstudianteEJB.
- La interfaz asociada al EJB debe llevar el mismo nombre del EJB sin el sufijo “EJB”. Ejemplo: RegistrarMatriculaEstudiante.
- Para los EJB de entidad, cuando la clase representa una relación entre dos entidades, el nombre se forma uniendo los nombres de las entidades involucradas.
- Los nombres de las clases que representan excepciones terminaran siempre con el sufijo “EXCEPCION”.
- El nombre de la clase no contendrá detalles sobre la implementación interna de la misma. Por ejemplo ArrayEstudiantes no es nombre válido.

Métodos

- Se deben utilizar los verbos autorizados para su codificación.
- El nombre de los métodos debe iniciar con minúscula la primera palabra, las siguientes palabras inician en mayúscula, sin separadores.

- La primera palabra del nombre de los métodos debe ser un verbo en infinitivo y debe representar una acción o comportamiento de la clase.
- El nombre del método debe describir claramente el comportamiento del mismo.
- En lo posible no se deben usar verbos genéricos aplicables a todo como: procesar, gestionar, manejar. Ejemplo: procesarEstudiante(), gestionarCliente(), en este caso el verbo no aclara el cometido real del método.

Paquetes

- El nombre de los paquetes debe iniciar con minúscula la primera palabra, las siguientes palabras inician en mayúscula, sin separadores.
- La estructura de los paquetes es la siguiente:

  co.edu.uis.[sistema].[aplicación].[módulo].[caso de uso]

  Ejemplo:
  co.edu.uis.financiero.egresos.contratacion.ordenarPrestacionServicio.

- La estructura para el paquete donde van a estar las entidades comunes para todos los sistemas es el siguiente:

  co.edu.uis.sistema.entidades

- La estructura para el paquete donde van a estar los servicios comunes para todos los sistemas es el siguiente:

  co.edu.uis.sistema.servicios

Variables

- Para el nombre de las variables utilizar palabras completas en singular (máximo tres palabras). El nombre deber ir en plural cuando la variable representa una lista o un conjunto de elementos.
- Si las palabras no son suficientes para la descripción, se debe hacer un comentario, al frente de la variable.
- Las constantes (final) van en mayúscula sostenida separando las palabras por guión de piso (_).
- Para los argumentos, deben iniciar siempre con la letra “a” y posteriormente el nombre según lo establecido anteriormente.
- Para las instancias de las clases, las entidades llevan el mismo nombre de la clase, solo que la palabra inicial va en minúscula. Si se necesita más de una instancia, para diferenciarlas se debe adicionar una palabra que identifique el rol que desempeña. Ejemplo: Estudiante estudiantePregrado, Estudiante estudiantePostgrado.
- La variable del EntityManager se debe llamar em.
- La variable de tipo FacesMessages se debe llamar facesMessages.

Librerías (JAR)

El nombre de las librerías no sigue el mismo estándar de las clases. Éstas se deben escribir en minúscula, separando cada palabra con un guión (-). Ejemplo: recursos-humanos.jar.

Nombres de archivos

Se sigue el mismo estándar descrito en las reglas de sintaxis generales.

4.2.4 Documentación

La documentación relacionada con el diseño del sistema reside en la base de datos de Enterpirse Architect.

La documentación del código fuente se debe hacer en cada una de las clases, siguiendo el estándar de JAVADOC y documentando la definición de la clase, descripción de los métodos get y set y descripción de los parámetros de entrada y salida de cada uno de los métodos que componen la clase.

4.2.5 Capa de Presentación

Plantilla principal

La plantilla principal de una página es la siguiente:

Figura: 13 Plantilla Principal División de Servicios de Información

Contenido

Esta sección se refiere al caso de uso implementado. La estructura de esta sección es:

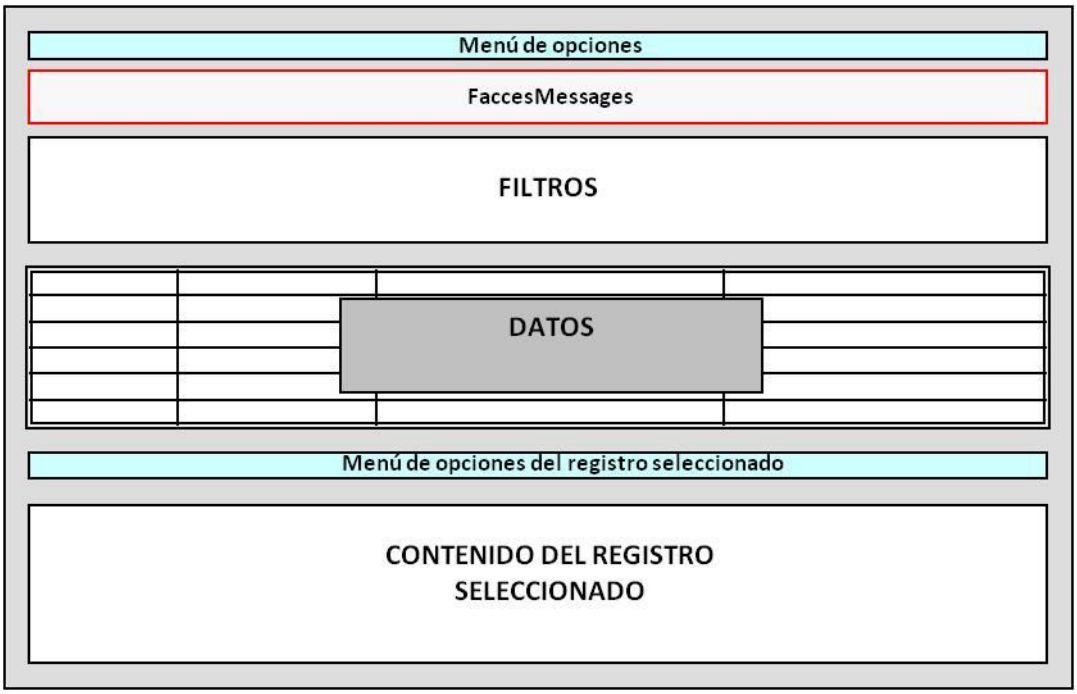


Figura: 14 Plantilla de Contenido División de Servicios de Información.

Para las páginas que muestran formularios:

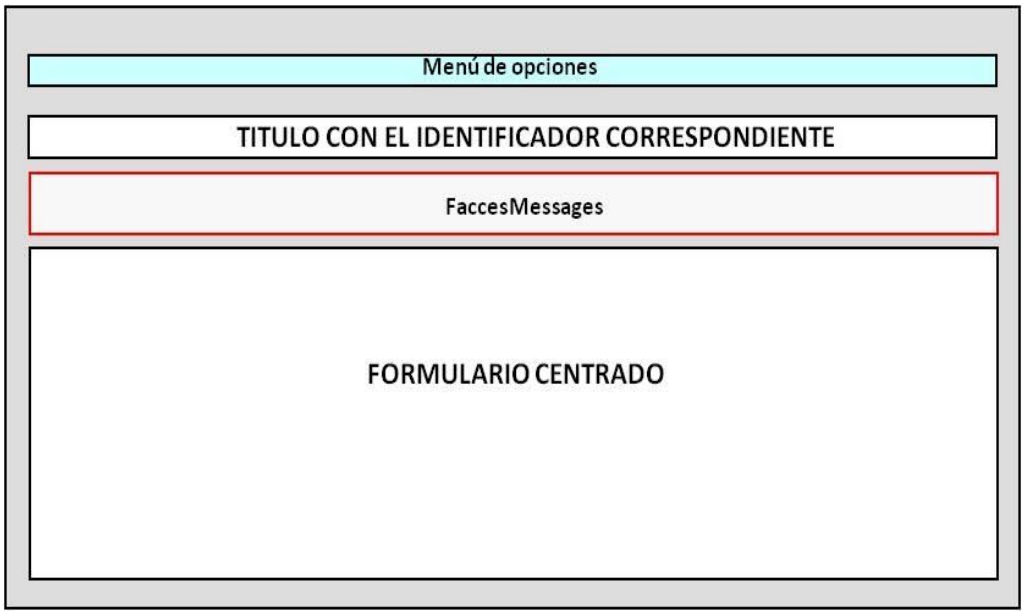


Figura: 15 Plantilla de Contenido con Formulario. División de Servicios de Información.

Paginación

Para la paginación se debe tener en cuenta el tipo de consulta que se va a realizar y la memoria consumida por ésta en el servidor de base de datos. De acuerdo a esto la aplicación decide el número de registros máximos que debe retornar la consulta.

Por ejemplo, se quiere consultar los estudiantes de la sede Bucaramanga. De acuerdo al análisis realizado, no hay un consumo alto de memoria, por lo que se decide retornar 100 estudiantes máximos. Esto indica que cuando se implemente el método de consulta en JPQL, el parámetro setMaxResults debe tener el valor de 100.

Texto

Existen cinco tipos de plantillas para mostrar texto en la página. Éstas se encuentran dentro de la carpeta Plantillas.

- etiqueta.xhtml: Texto o datos.
- etiquetaColumna.xhtml: El título de la columna en una tabla
- titulo.xhtml: Titulos
- etiquetaNavegacion.xhtml: Utilizado en la barra de navegación
- mostrar.xhtml: Permite visualizar dos etiquetas (etiqueta, dato) al mismo tiempo. Su objetivo es la de visualizar datos como si fuera un formulario.

La sintaxis para utilizar las plantillas son:

```
<s:decorate template="../Plantillas/[nombrePlantilla]">
     <ui:define name="label">
          #{FormatoWeb.mostrarMensaje('primerNombre', true)}
     </ui:define>
</s:decorate>
```

Donde nombrePlantilla puede ser etiqueta.xhtml, etiquetaColumna.xhtml, etiquetaTitulo.xhtml o etiquetaNavegacion.xhtml.

Para la plantilla mostrar.xhtml:

```
<s:decorate template="../Plantillas/mostrar.xhtml">
     <ui:define name="label">
          #{FormatoWeb.mostrarMensaje('primerNombre', true)}
     </ui:define>
     <h:outputText
     value="#{formatoWeb.usuario.primerNombre}"/>
</s:decorate>
```

Edición

Para capturar cualquier tipo de dato en una caja de texto se debe utilizar la plantilla edicion.xhtml de la siguiente manera:

```
<s:decorate id="dcr[identificador]"
      template="/Plantillas/edicion.xhtml">

      <ui:define name="label">
            #{FormatoWeb.mostrarMensaje('primerNombre', true)}
      </ui:define>

      <h:inputText id="txt[nombreCajaDeTexto]" value="[valor]"
            required="true">

            <a:support event="onblur"
      reRender="dcr[identificador]"/>
      </h:inputText>

</s:decorate>
```

dcr[identificador] es el nombre del elemento decorate. Por ejemplo: dcrPrimerNombre.

txt[nombreCajaDeTexto] es el nombre de la caja de texto. Por ejemplo: txtPrimerNombre.

Tablas estáticas

Se debe usar el control panelGrid de Java Server Faces.

Tablas dinámicas

Se debe usar el control dataTable de Rich Faces, agregando las siguientes líneas de programación en su definición:

onRowMouseOver="this.style.backgroundColor='#E1E1E1'"

onRowMouseOut="this.style.backgroundColor='#{a4jSkin.tableBackgroundColor}'

Las cuales indican el color de la fila cuando el puntero del mouse esta sobre ésta.

Listas desplegables

Para cualquier tipo de lista se debe utilizar el control de Java Server Faces.

Mensajes del sistema

Los mensajes del sistema son textos enviados por los EJB de sesión, ya sea de confirmación de una acción o errores en el procedimiento. Dichos errores se deben mostrar en la etiqueta FacesMessages la cual debe estar definida al comienzo de la página después del menú si existiera éste. El código es el siguiente:

```
<h:messages globalOnly="true" styleClass="message"/>
```

Para mostrar errores de validación en los formularios, éstos deben aparecer al frente de cada control y no globalmente.

4.3 DIAGRAMAS UML

Dentro del proceso de análisis y diseño del sistema de información se generaron los diagramas UML, que se presentarán a continuación, es de destacar que durante el transcurso del proyecto el diseño se fue adaptando a medida que se fue perfeccionando el prototipo inicial.

Aquí se presentarán los esquemas básicos de cada uno de los diagramas que se generaron, producto del diseño, con el fin de ilustrar el fondo y la forma como fueron producidos.

La documentación completa del diseño, se encuentra en el CD anexo, que hace parte de este informe.

4.3.1 Diagrama de Casos de Uso.

4.3.1.1 Identificación de los Actores

Dentro del sistema de Asignación automática se pueden identificar los siguientes actores, cada uno con roles, acciones y permisos distintos:

- **Dirección Admisiones**: Son los funcionarios de la dirección de admisiones que tienen acceso al caso de uso de Asignación automática total o parcial, obteniendo un control sobre la asignación de cualquier tipo de aula en el campus.
- **Dirección Escuela:** Son los funcionarios de la Dirección de Escuela que tienen acceso al caso de uso de reservación de aulas, para reservar aulas propias de la escuela, y al caso de uso de asignación automática, en cuanto a la asignación de grupos de escuela en aulas propias de la escuela.

4.3.1.2 Casos de uso por Actor

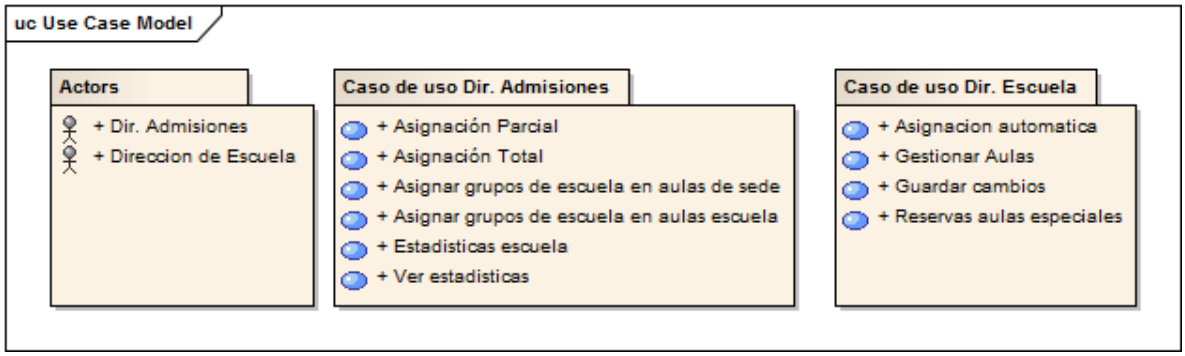


Figura: 16 Paquetes Casos de Uso

Actor: Dirección Admisiones

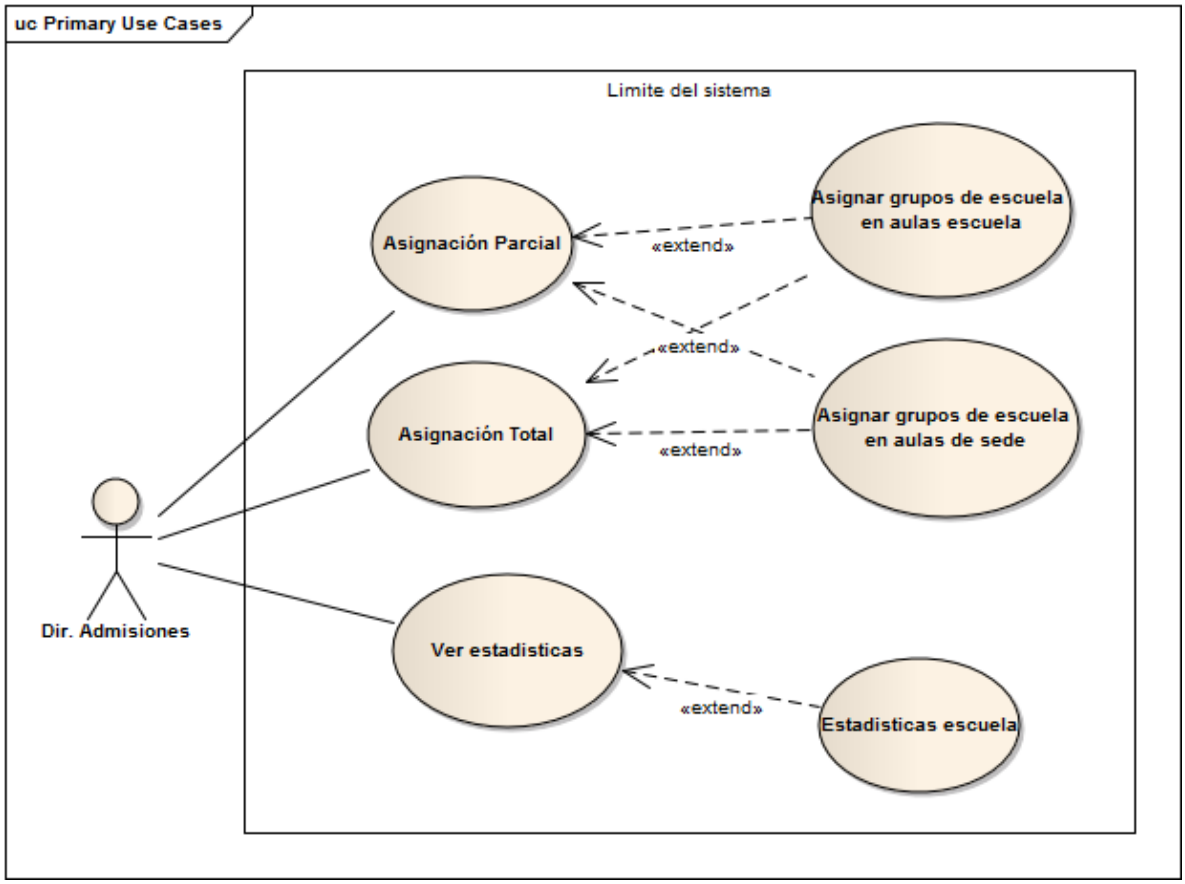


Figura: 17: Casos de Uso Dir. Admisiones

Se documenta la información del caso de uso: Registrar Solicitud, los demás casos de uso se pueden encontrar documentados en el CD que hace parte de éste documento.

Caso de uso: Asignación parcial

El funcionario de la Dirección de Admisiones tiene la posibilidad de seleccionar una de las escuelas o departamentos y proceder con una asignación automática.

CASO DE USO	**ASIGNACIÓN PARCIAL**	
Objetivo	El funcionario de la Dirección de Admisiones tiene la posibilidad de seleccionar una de las escuelas o departamentos y proceder con una asignación automática.	
Actores	Funcionario de la Dirección de Admisiones	
Pre condiciones	Haber ingresado al sistema.	
Post Condiciones	Generar estadísticas de la asignación.	
Pasos	Acción del Actor	Respuesta del Sistema
	Ingresa al Sistema	
		Presenta el listado de Escuelas
	Selecciona una escuela de la lista.	
		Presenta listados de asignaturas
	Selecciona tipo de asignación. (en aulas de la escuela seleccionada o en aulas del campus). Procede Asignación.	
		Inicia el algoritmo de asignación, muestra estadísticas de asignación y listados de horarios.
	Aceptar	
Variaciones		
Extensiones		

Tabla 1 Descripción Caso de uso: Asignación Parcial

Caso de uso: Asignación Total

El funcionario de la Dirección de Admisiones tiene la posibilidad de seleccionar un tipo de asignación como lo son *asignación de grupos de escuela en aulas de escuela o asignación de grupos de escuela en aulas de campus* y proceder con la asignación automática total para cada una de las escuelas o departamentos.

CASO DE USO	**ASIGNACIÓN TOTAL**	
Objetivo	El funcionario de la Dirección de Admisiones tiene la posibilidad de proceder con la asignación automática global para todas las escuelas en un solo procedimiento.	
Actores	Funcionario de la Dirección de Admisiones	
Pre condiciones	Haber ingresado al sistema.	
Post Condiciones	Generar estadísticas de asignación.	
Pasos	Acción del Actor	Respuesta del Sistema
	Ingresa al Sistema	
		Presenta los tipos de asignación posibles.
	Selecciona una tipo de asignación y proceder.	
		Inicia el algoritmo de asignación, muestra estadísticas de Asignación y listados de horarios para cada una de las escuelas.
	Aceptar	
Variaciones		
Extensiones		

Tabla 2 Descripción Caso de uso: Asignación total

Actor: Dirección de Escuela

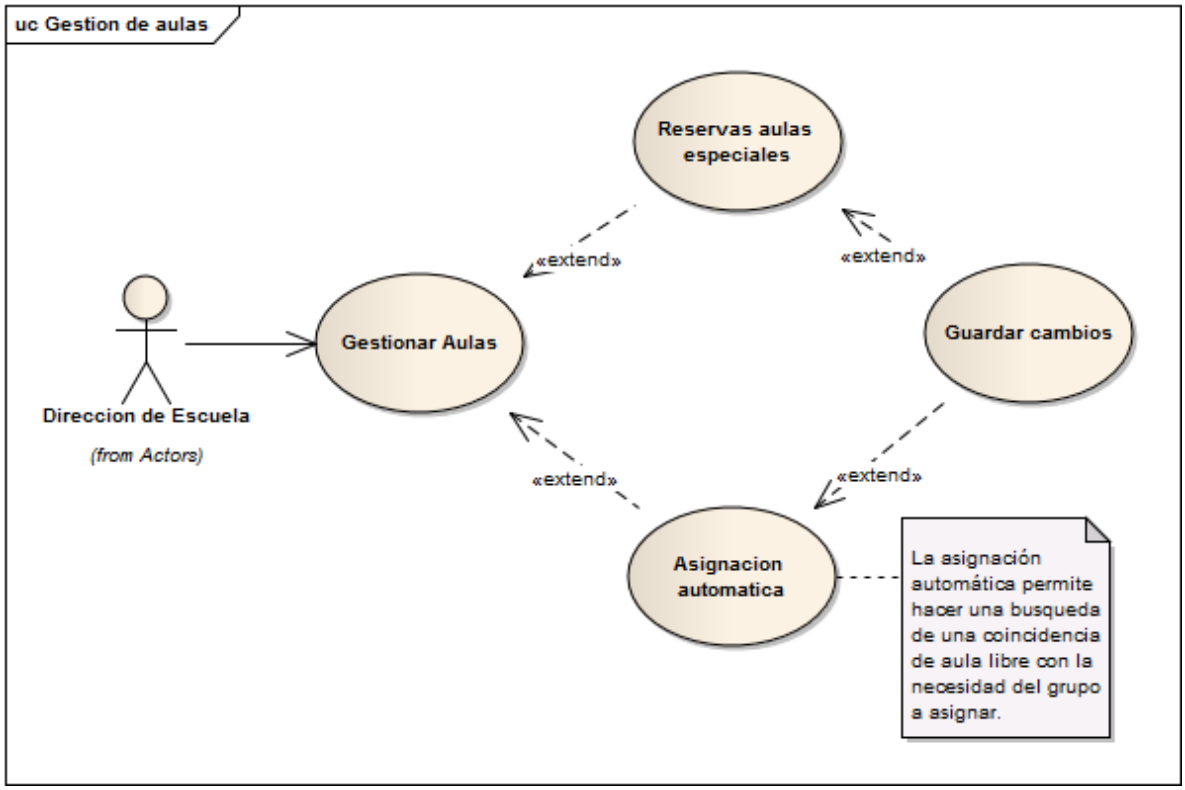


Figura: 18 Casos de Uso Dirección Escuela

Caso de uso: Asignación automática

El funcionario de la Dirección de escuela tiene la posibilidad de hacer una asignación completa de los grupos de escuela en las aulas libres de las que dispone la escuela.

CASO DE USO	**ASIGNACIÓN AUTOMATICA (Dir. ESCUELA)**	
Objetivo	El funcionario de la Dirección de Escuela tiene la posibilidad de hacer una asignación completa de los grupos de escuela en las aulas libres de las que dispone la escuela.	
Actores	Funcionario de la Dirección de Escuela	
Pre condiciones	Haber ingresado al sistema.	
Post Condiciones	Generar estadísticas de asignación.	
Pasos	Acción del Actor	Respuesta del Sistema
	Ingresa al Sistema	
		Presenta listas de asignaturas que intervienen en el proceso.
	Proceder con la asignación automática.	
		Inicia el algoritmo de asignación, muestra estadísticas de Asignación y listados de horarios.
	Aceptar	
Variaciones		
Extensiones		

Tabla 3 Descripción Caso de uso: Asignación automática (Dir. Escuela)

Caso de uso: Reservar Aulas Especiales

El funcionario de la Dirección de escuela tiene la posibilidad de las aulas especiales con el fin de que estas no entren en el proceso de asignación automática.

CASO DE USO	RESERVAR AULAS ESPECIALES	
Objetivo	Reservar aulas especiales de la escuela para que no entren en el proceso de asignación automática	
Actores	Funcionario de la Dirección de Escuela	
Pre condiciones	Haber ingresado al sistema.	
Post Condiciones		
Pasos	Acción del Actor	Respuesta del Sistema
	Ingresa al Sistema	
		Presenta listado de aulas bajo el control de la escuela.
	Seleccionar Aula.	
		Presentar horario de ocupación del aula.
	Seleccionar la nueva ocupación o reserva del aula. Aceptar.	
		Presentar mensaje con éxito de la transacción.
Variaciones		
Extensiones		

Tabla 4 Descripción Caso de uso: Reservar Aulas Especiales

4.3.2 Diagrama de Clases

Aquí se presenta el esquema general del diagrama de clases, resultante en la etapa de análisis del proyecto.

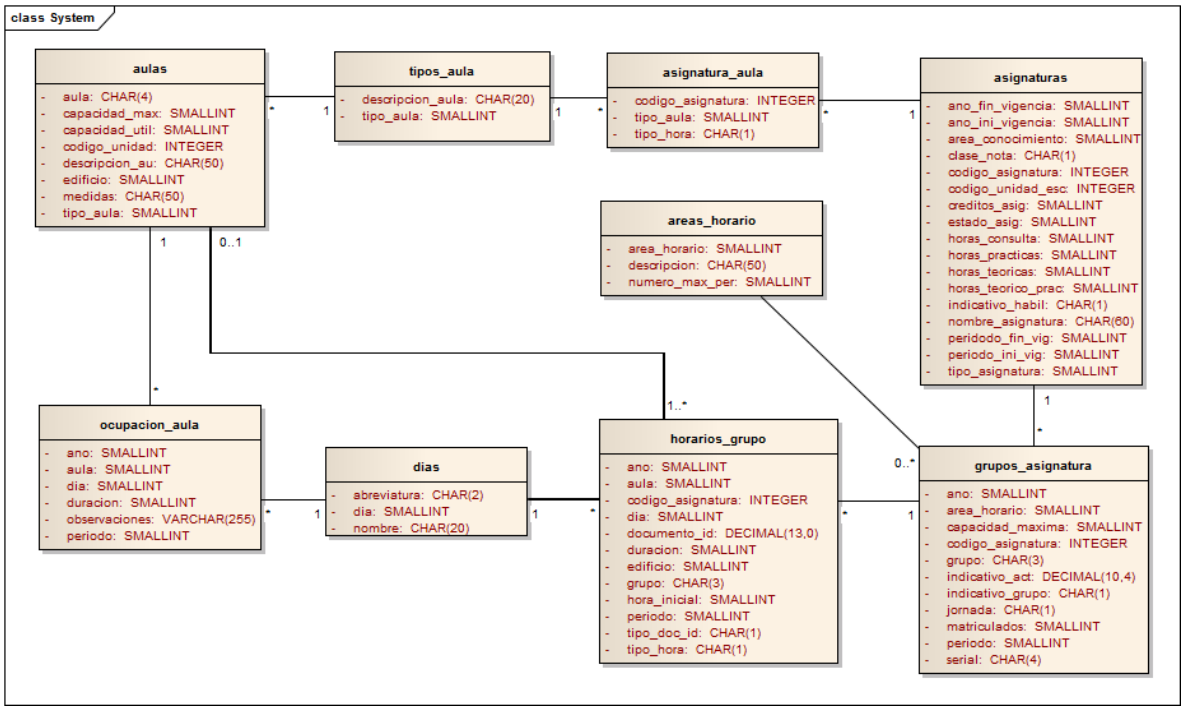


Figura: 19 Diagrama de Clases sistema Asignación automática

La documentación completa de este diagrama nombres de las clases, atributos de ellas y características se encuentran documentadas en el CD anexo a este documento.

4.3.3 Diagrama de Datos

Aquí se presenta un esquema general del diagrama de datos resultante en la etapa de análisis del proyecto.

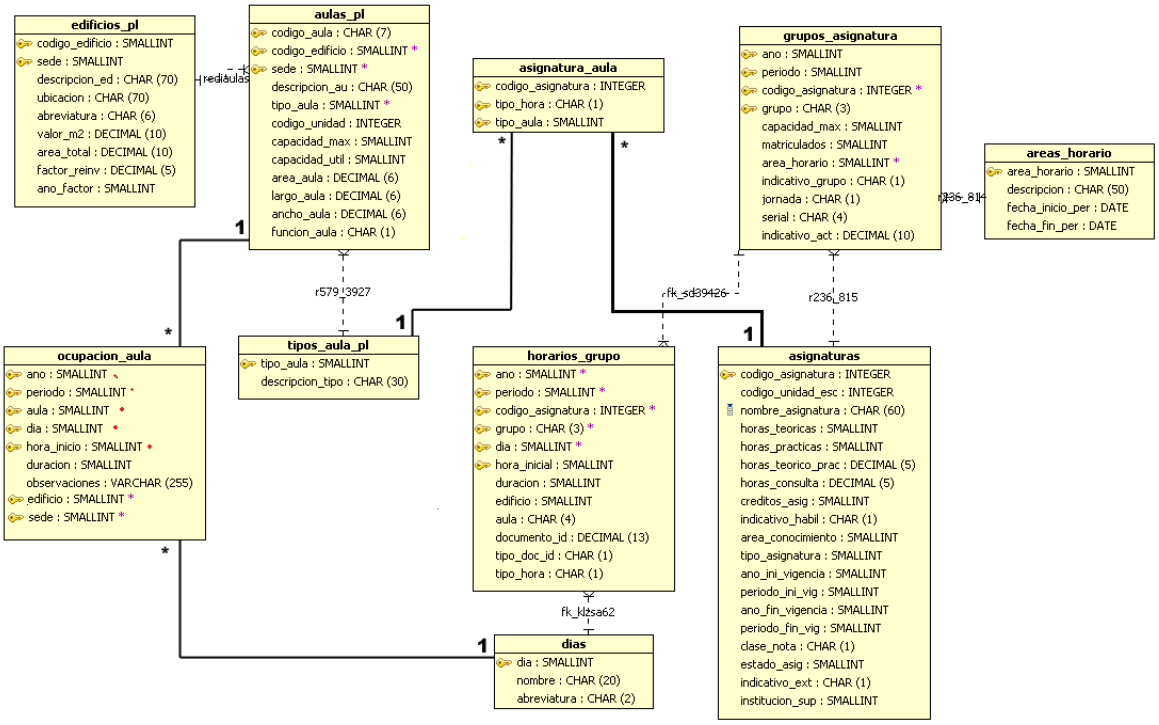


Figura: 20 Diagrama de Datos del Sistema de Asignación Automática de aulas.

La mayoría de las entidades, no son propias del sistema pero se utilizan para el funcionamiento, hacen parte de otros sistemas de información de la Universidad Industrial de Santander, éstas son:

- edificios
- aulas
- asignatura
- areas_horario
- tipos_aula
- horarios_grupo
- grpos_asignatura
- días

En cuanto a las entidades Ocupacion_aula y Asignatura_aula, fueron creadas para el funcionamiento del sistema y tienen las siguientes funciones:

Ocupacion_aula: Esta entidad está encargada de listar la ocupación de cada una de las aulas del campus universitario con una descripción que especifica su labor. Los datos o información que colman esta entidad son generados por el sistema de creación de grupos y por el modulo de reservación de aulas.

Asignatura_aula: La entidad asignatura aula es la encargada de relacionar cada una de las asignaturas con un tipo especifico de aula para las clases teóricas y las practicas, esta entidad juega un papel importante durante el proceso de asignación automática, pues es una entidad de referencia para identificar las asignaturas que pueden entrar en el proceso, y de esta forma identificar de manera breve que tipo de aulas se deben conceder a cada asignatura.

Otro aspecto importante de la anterior entidad es que básicamente es una tabla de control y los datos de referencia son gestionados por la División de servicios de información.

Para la alimentación de esta tabla se desarrollaron scripts que utilizan la información de los periodos anteriores al 2011 en cuanto a la asignación de aulas que se ha venido llevando, teniendo en cuenta la asignación teórica y practica.

La documentación completa de este diagrama nombres de las clases, atributos de ellas y características se encuentran documentadas en el CD anexo a este documento.

4.3.5 Diagrama de Actividades

Se definieron diagramas de actividades como soporte a los casos de uso, considerados fundamentales en el desarrollo del sistema.

Con el fin de ampliar la información del procedimiento que debe desarrollar esos casos de uso, aquí se especifican cada uno de ellos.

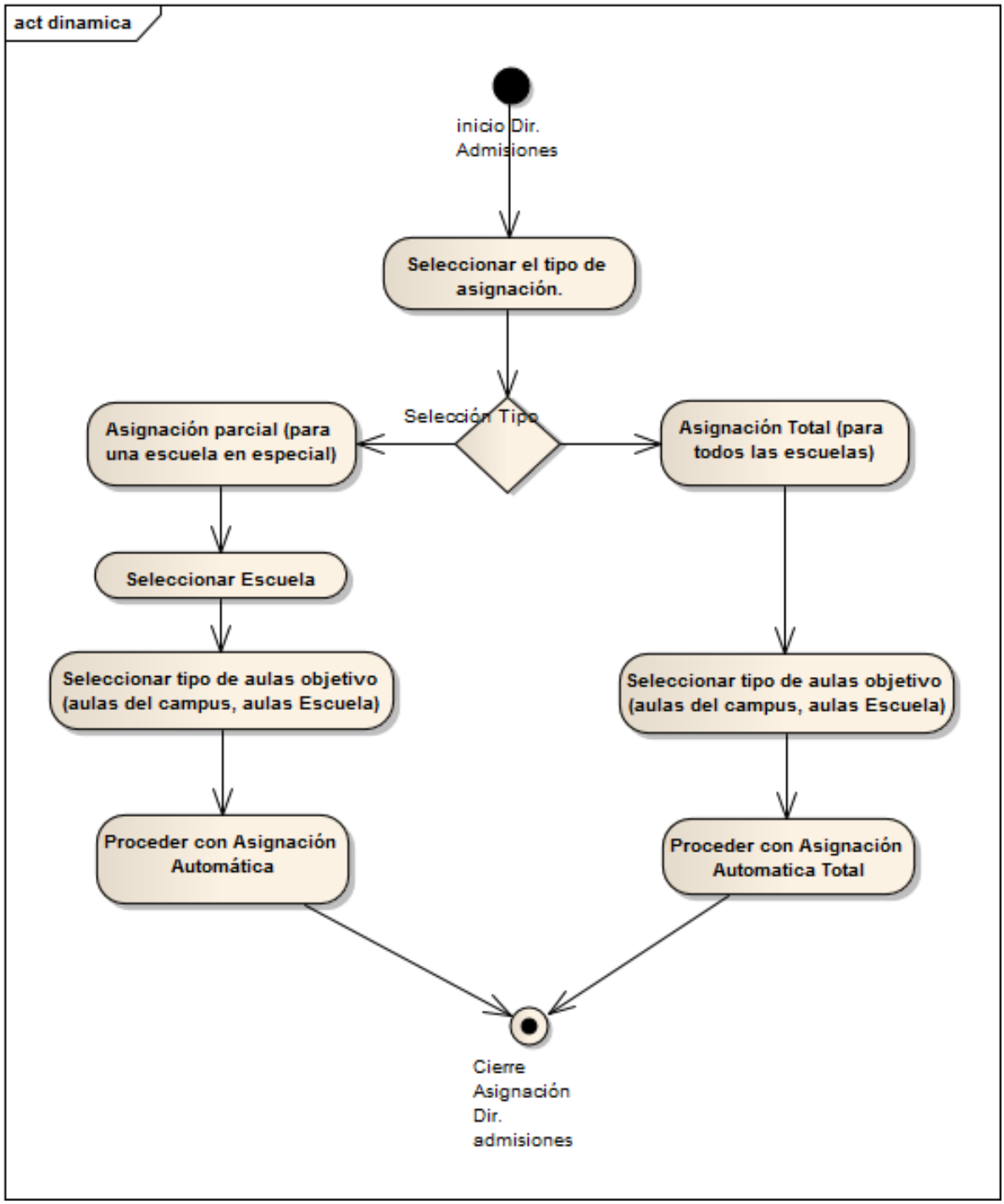


Figura: 21 Diagrama de Actividades para casos de uso, Dir. Admisiones

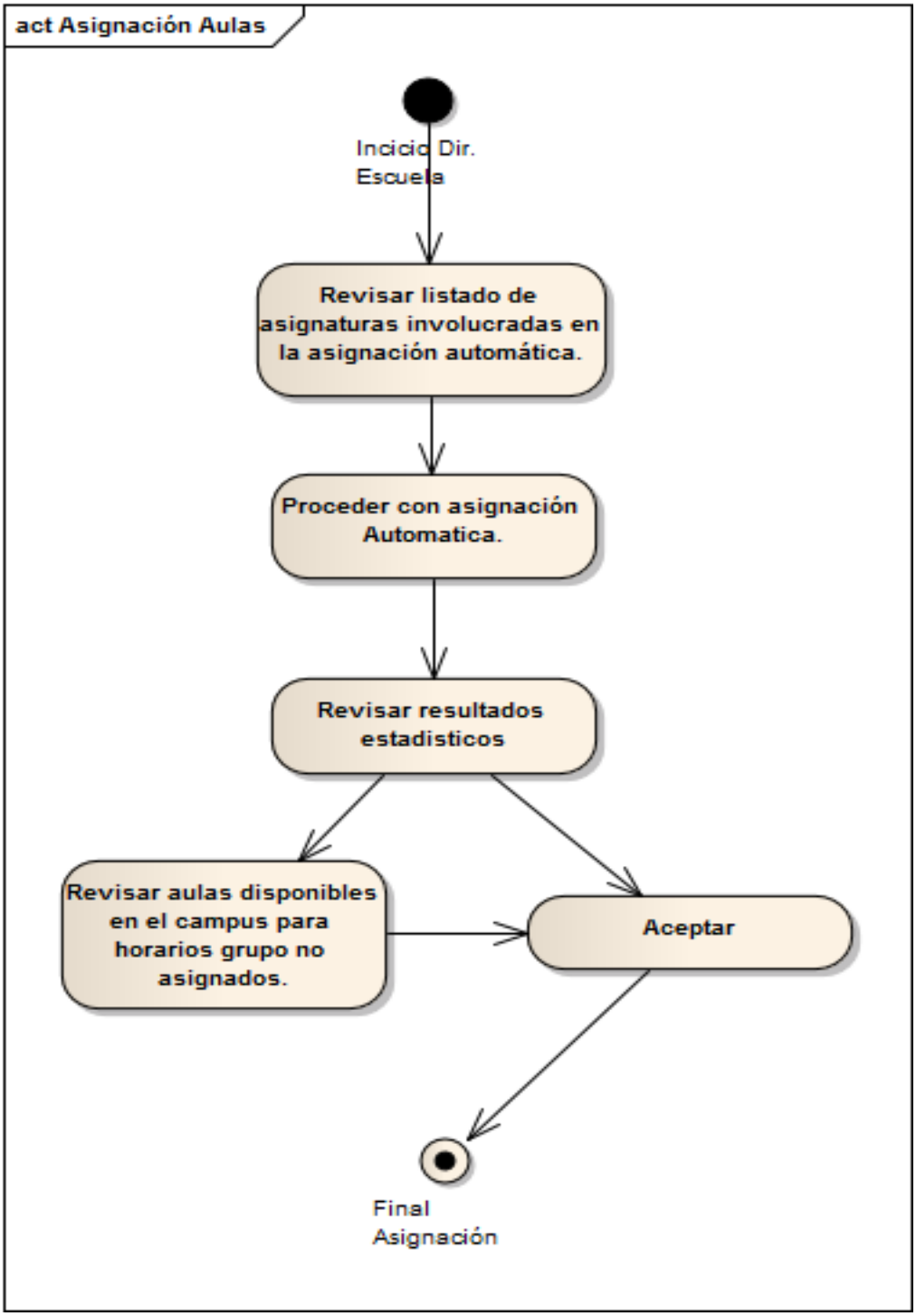


Figura: 22 Diagrama de Actividades para casos de uso, Dir. Escuela

4.4 PROTOTIPO INICIAL

4.4.1 Generalidades del prototipo inicial

El proceso de construcción del prototipo inicial comenzó desde el momento en el que la escuela de matemáticas manifestó el interés de desarrollar el proyecto, continuando con una serie de encuentros periódicos con Gilberto Arenas Director de escuela de matemáticas, Robinson Rojas Ingeniero encargado de controlar el desarrollo de aplicativos en la División de Servicios De Información de la Universidad Industrial de Santander.

En éstos encuentros se escuchó y analizó cada uno de los requerimientos deseados para el sistema y se procedió a generar el diseño del prototipo no funcional que básicamente determina las distintas posibilidades que tiene cada uno de los usuarios para usar la herramienta de asignación automática, tal y como se describió en los diagramas de casos de uso.

El prototipo no funcional se creó como una página web básica sin ningún tipo de programación de fondo, que mostraría gráficamente el esquema de navegación del sistema, de ésta manera se definieron detalles de alta importancia tales como la información que el usuario quiere ver, la que desea ocultar, el contenido de cada una de las vistas y los datos que se le van a solicitar a cada uno de los usuarios que interactuarían con el sistema.

Se trabajó un esquema cíclico de perfeccionamiento y refinamiento de tal manera que se llegara a un nivel de acercamiento detallado de lo que quiere alcanzar el usuario con el sistema, de esta manera se logra que el proceso de refinamiento del prototipo funcional sea mínimo agilizando de esta manera el ciclo de vida del proyecto.

El tiempo destinado a esta etapa fue bastante amplio, ya que del resultado obtenido dependería la eficiencia del desarrollo y programación del sistema.

Durante el transcurso de la construcción del prototipo se definió todo el diseño integrado por los diagramas de casos de uso, de clases, de datos y de actividades.

4.4.2 Interfaz resultado del prototipo inicial

Se presenta a continuación apartes de la interfaz resultante de la construcción del prototipo inicial (no funcional).

Figura: 23 Vista de asignación automática genérica

Figura: 24 Vista de información inicial genérica

Esta figura es un ejemplo del prototipo inicial (no funcional) que permite entender algunos de los aspectos más importantes que debe tener la herramienta de asignación automática con el fin de desarrollar una interfaz más rica, que le permita al usuario saber cuál es el procedimiento que está ejecutando y cuáles son las entidades que se afectan con la ejecución de dicho proceso.

4.5 PROTOTIPO FINAL

4.5.1 Generalidades del Prototipo Final

Al finalizar el desarrollo del proyecto de asignación automática de aulas tomando como punto de partida el prototipo no funcional, se logró consolidar el producto definitivo, que pretende actuar como una herramienta de soporte o ayuda en el proceso de matrícula de grupos semestralmente en la Universidad Industrial de Santander.

El producto cumple todos y cada uno de los parámetros que se definieron en el documento de requerimientos que se menciona en el apartado 4.1 de este documento, y después de un proceso de evaluación de parte del usuario, ha sido aceptado por éste, manifestando su plena conformidad con el resultado.

A continuación se describirá de forma general las funcionalidades más importantes, clasificadas por módulos, propias del sistema de asignación automática de aulas de la Universidad Industrial de Santander

Modulo de reservación de aulas.

Usuario Final: Dirección de escuela

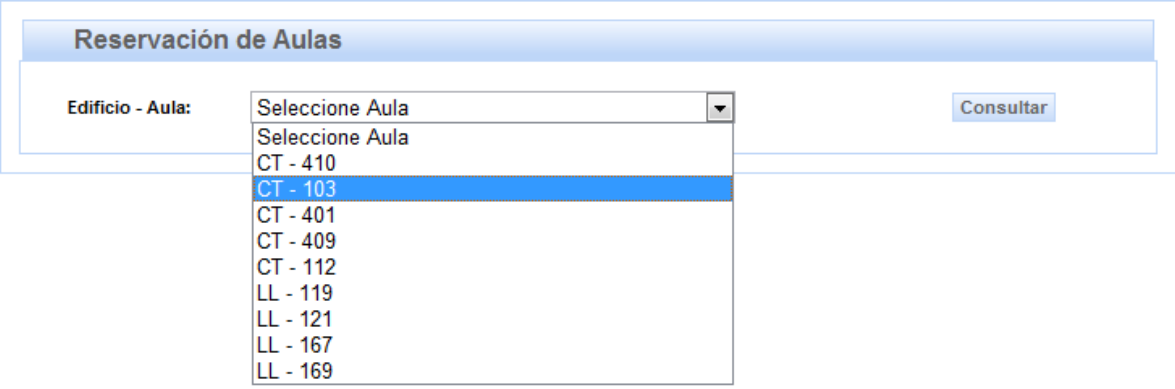


Figura: 25 Selección del aula

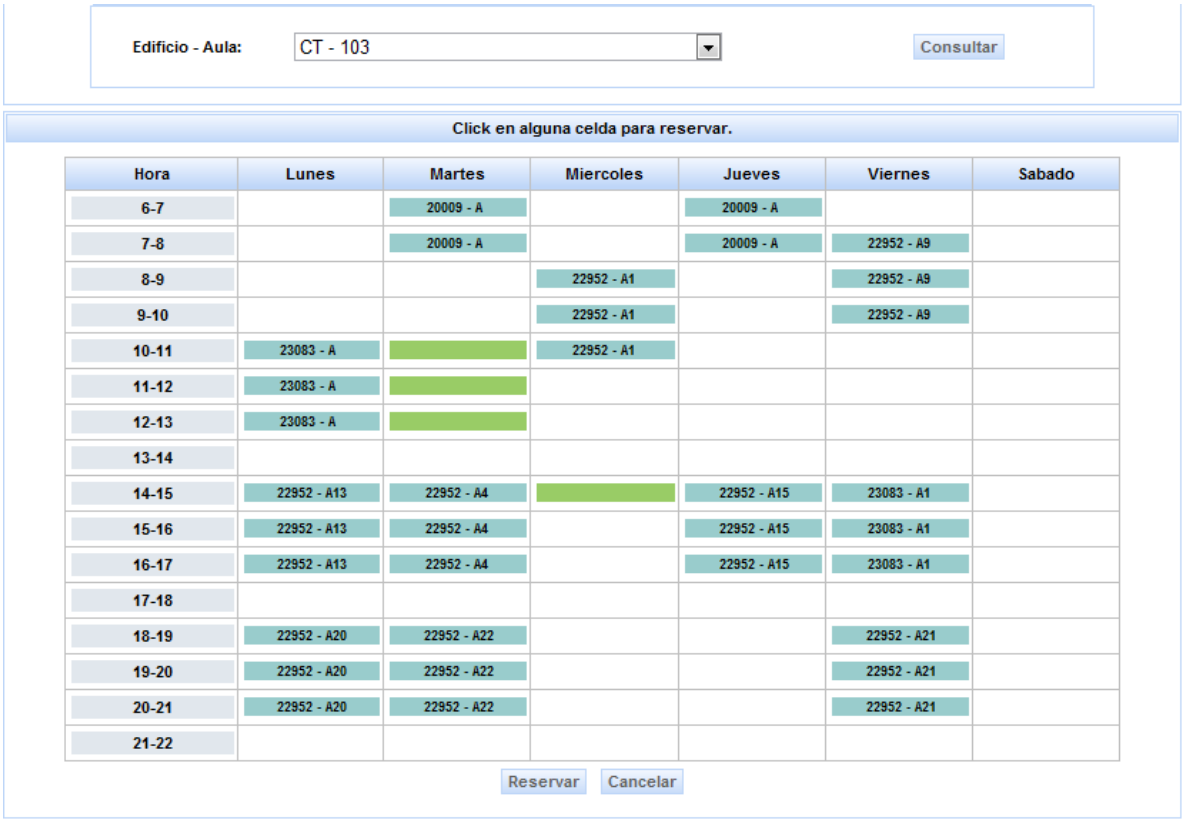


Figura: 26 Reservación del aula

El modulo de reservación de aulas se enfoca en permitirle al usuario consultar la ocupación total de un aula mediante una tabla de horario, diseñada según los horarios de trabajo específicos de la Universidad Industrial de Santander, esta tabla dinámica se presta para seleccionar aquellos espacios donde el usuario desea hacer una reserva del aula para algún propósito especial durante el semestre.

Este modulo se desarrolló especialmente para darle la oportunidad al usuario perteneciente a cada dirección de escuela de reservar sus aulas de forma manual, impidiendo de esta manera que en estos horarios se asigne un grupo automáticamente.

Modulo de Asignación de aulas por parte de la dirección de escuela.

- Cada director de escuela tiene la oportunidad de usar la herramienta para proceder con la asignación automática de los grupos de su escuela, esta vista pretende dar información al usuario sobre el listado de asignaturas que tienen relación con algún tipo de aula y pueden ser asignadas automáticamente y aquellas que no necesitan de un aula para su desempeño o no tienen estipulada una relación con un tipo de aula en especial.

Figura: 27 Vista inicial asignación automática dirección de escuela.

- Luego de examinar aquellas asignaturas que entrarán en el proceso de asignación, el usuario al proceder, desplegará un mensaje de advertencia que le indica sobre el inicio del proceso que asignará a los grupos de la escuela aulas que se encuentren en poder de la misma escuela.

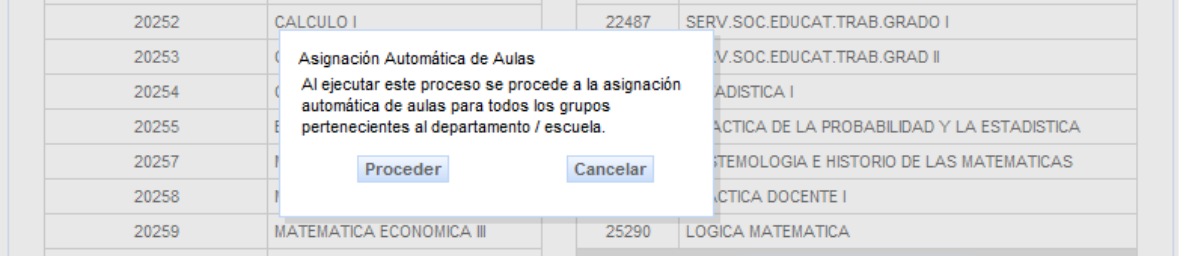


Figura: 28 Dialogo de advertencia sobre el proceso.

- Al ejecutar el proceso de asignación automática el usuario puede apreciar una barra de progreso que le permite verificar el avance del proceso.

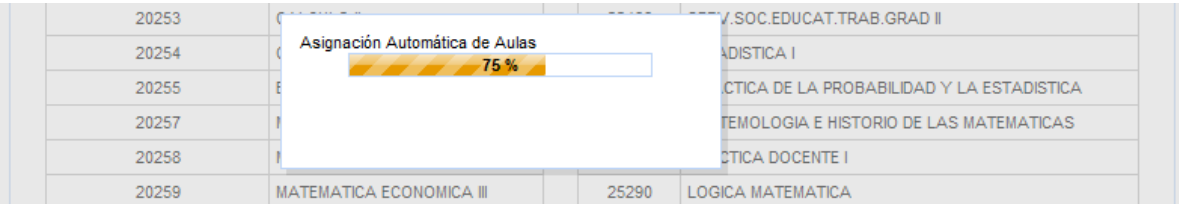
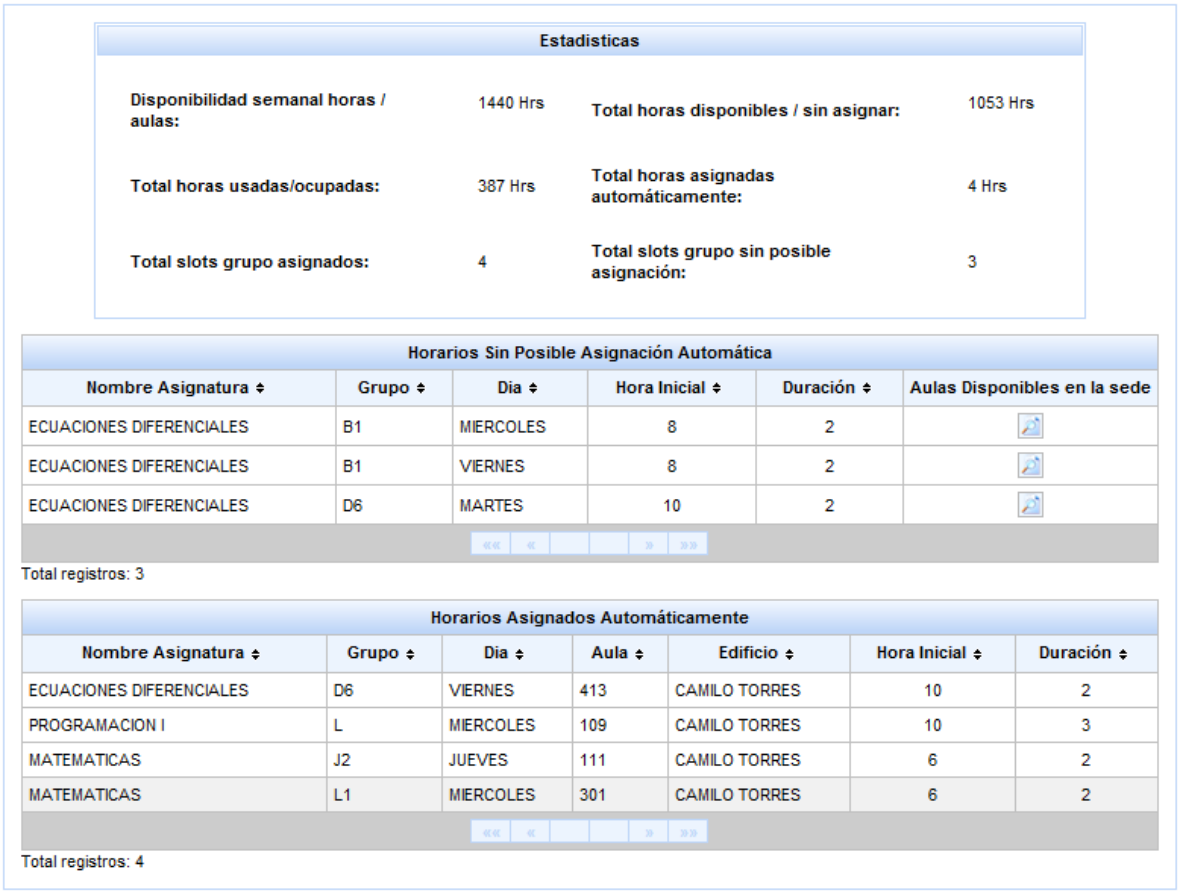


Figura: 29 Vista del progreso del proceso.

- Al finalizar el procedimiento de asignación automática el usuario puede revisar un conjunto de resultados que se conforman de una tabla de estadísticas con datos generales sobre los recursos físicos de los que depende la escuela en cuanto a horas semanales, el total de uso de estos recursos y el total de horas y grupos que fueron asignados durante el proceso de asignación automática.



Estadisticas

Disponibilidad semanal horas / aulas:	1440 Hrs	**Total horas disponibles / sin asignar:**	1053 Hrs
Total horas usadas/ocupadas:	387 Hrs	**Total horas asignadas automáticamente:**	4 Hrs
Total slots grupo asignados:	4	**Total slots grupo sin posible asignación:**	3

Horarios Sin Posible Asignación Automática

Nombre Asignatura	Grupo	Dia	Hora Inicial	Duración	Aulas Disponibles en la sede
ECUACIONES DIFERENCIALES	B1	MIERCOLES	8	2	
ECUACIONES DIFERENCIALES	B1	VIERNES	8	2	
ECUACIONES DIFERENCIALES	D6	MARTES	10	2	

Total registros: 3

Horarios Asignados Automáticamente

Nombre Asignatura	Grupo	Dia	Aula	Edificio	Hora Inicial	Duración
ECUACIONES DIFERENCIALES	D6	VIERNES	413	CAMILO TORRES	10	2
PROGRAMACION I	L	MIERCOLES	109	CAMILO TORRES	10	3
MATEMATICAS	J2	JUEVES	111	CAMILO TORRES	6	2
MATEMATICAS	L1	MIERCOLES	301	CAMILO TORRES	6	2

Total registros: 4

Figura: 30 Vista de los resultados del proceso.

- En algunos casos la asignación automática no es suficiente para satisfacer al 100 % las necesidades de una escuela, es por esta razón que también se debe denotar la lista de aquellos horarios sin posible asignación, como se puede observar en el ejemplo de la figura 27 existen tres horarios que no fueron asignados en aulas de la escuela, con el fin de darle soporte a los directores de escuela en cuanto estos “horarios problema”, se ha implementado un procedimiento que le permite al usuario hacer una búsqueda a nivel de sede, de aquellas aulas que pueden satisfacer la necesidad de cada uno de estos horarios.

- En la última columna de la tabla de horarios sin posible asignación comprendida en la figura 27 podemos observar un elemento que le permite al usuario ejecutar el proceso de búsqueda de un listado de aulas de la sede que puede satisfacer las necesidades de cada horario, y el ejemplo del resultado se puede apreciar en la siguiente figura.

- El cuadro de dialogo de la figura 27 presenta una tabla con el listado de aulas disponibles dentro de la sede que pueden suplir la necesidad del horario seleccionado, esta información es de soporte para cada director de escuela y pretende prestar una ayuda al usuario para que este se pueda remitir al responsable del aula que podría satisfacer su necesidad y entablar la petición del préstamo de la misma. El trámite del préstamo del aula es y su respectiva asignación es gestionado ante la dirección de admisiones, ente administrativo que tiene total control sobre la asignación de las aulas.

  La administración de aquellos grupos que al finalizar todo el proceso de matricula no tienen aun un aula asignada lo realiza la dirección de admisiones y hasta el momento no tienen una herramienta digital que les permita gestionar estas aulas de manera digital.

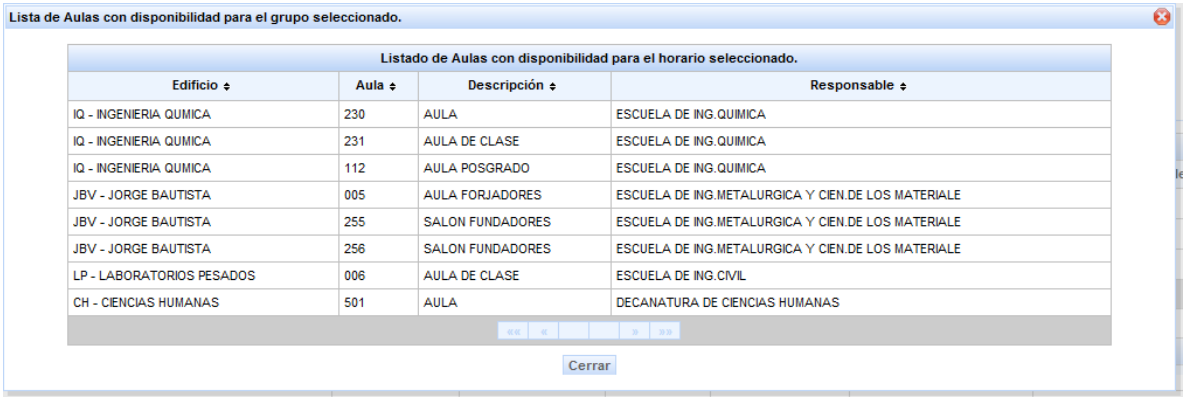


Figura: 31 Vista del listado de aulas disponibles en la sede

Modulo de Asignación de aulas por parte de la dirección de admisiones.

El modulo de asignación de aulas a nivel de escuela se replica en el modulo de asignación para el usuario de dirección de admisiones, la diferencia radica en que este usuario puede hacer el procedimiento en cualquiera de las escuelas de la sede que se encuentre administrando, en este caso la sede principal.

A diferencia de la asignación automática a nivel de escuela, la asignación a nivel de sede requiere tener en cuenta demás reglas de asignación que fueron provistas por el proceso de entrevista con el usuario final, en este caso la dirección de admisiones de la Universidad.

Otra de las cualidades que adquiere el usuario de la dirección de admisiones es poder escoger que tipo de asignación se ejecutara en cuanto a la administración de las aulas, como lo puede ser, asignación de grupos en aulas de la sede, sin ningún tipo de discriminación y asignación de grupos de la escuela seleccionada en aulas pertenecientes a la misma.

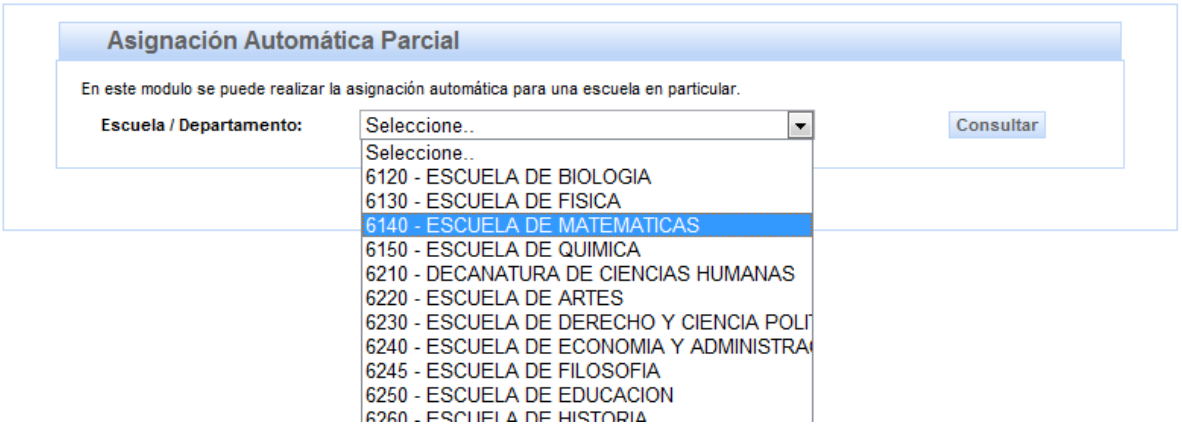


Figura: 32 Vista de selección de escuela.

- Al seleccionar una escuela el usuario puede ver información sobre las asignaturas que entran dentro del procedimiento de asignación automática.

Figura: 33 Vista asignaturas y tipo de asignación.

- Advertencia sobre el proceso que se va a llevar a cabo.

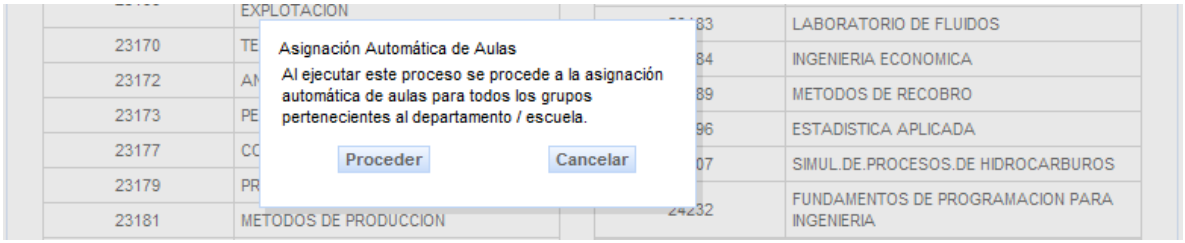


Figura: 34 Dialogo de advertencia.

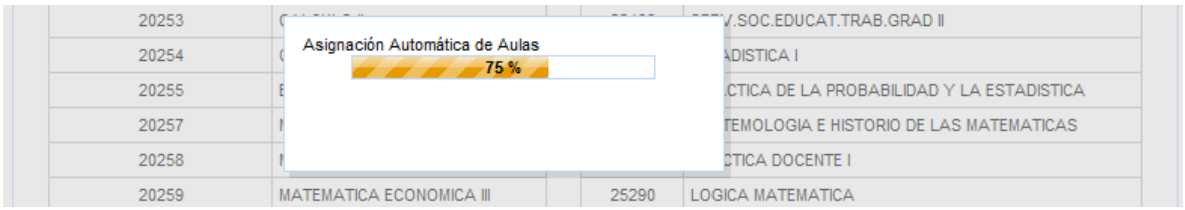
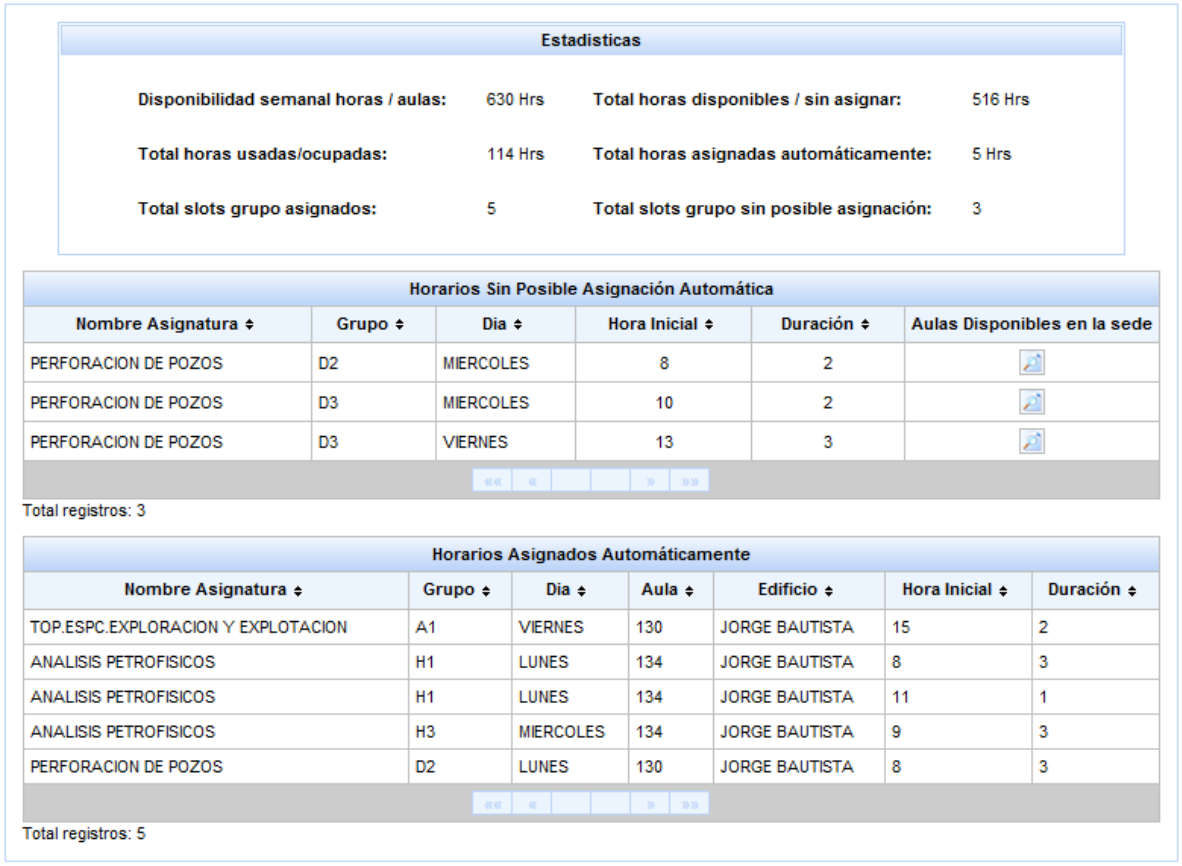


Figura: 35 Dialogo de progreso.

- Vista de resultados de la asignación.



Estadisticas

Disponibilidad semanal horas / aulas:	630 Hrs	**Total horas disponibles / sin asignar:**	516 Hrs
Total horas usadas/ocupadas:	114 Hrs	**Total horas asignadas automáticamente:**	5 Hrs
Total slots grupo asignados:	5	**Total slots grupo sin posible asignación:**	3

Horarios Sin Posible Asignación Automática

Nombre Asignatura	Grupo	Dia	Hora Inicial	Duración	Aulas Disponibles en la sede
PERFORACION DE POZOS	D2	MIERCOLES	8	2	
PERFORACION DE POZOS	D3	MIERCOLES	10	2	
PERFORACION DE POZOS	D3	VIERNES	13	3	

Total registros: 3

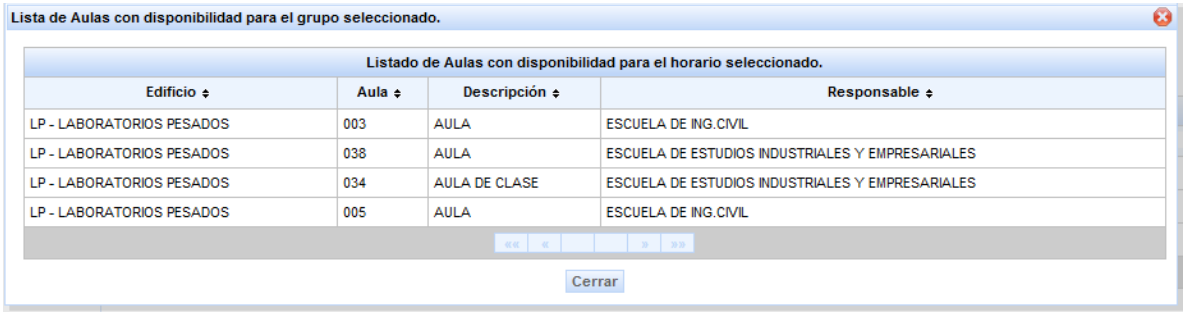
Horarios Asignados Automáticamente

Nombre Asignatura	Grupo	Dia	Aula	Edificio	Hora Inicial	Duración
TOP.ESPC.EXPLORACION Y EXPLOTACION	A1	VIERNES	130	JORGE BAUTISTA	15	2
ANALISIS PETROFISICOS	H1	LUNES	134	JORGE BAUTISTA	8	3
ANALISIS PETROFISICOS	H1	LUNES	134	JORGE BAUTISTA	11	1
ANALISIS PETROFISICOS	H3	MIERCOLES	134	JORGE BAUTISTA	9	3
PERFORACION DE POZOS	D2	LUNES	130	JORGE BAUTISTA	8	3

Total registros: 5

Figura: 36 Vista de resultados.

- Vista de listado de aulas disponibles para horarios no asignados.



Lista de Aulas con disponibilidad para el grupo seleccionado.

Listado de Aulas con disponibilidad para el horario seleccionado.

Edificio	Aula	Descripción	Responsable
LP - LABORATORIOS PESADOS	003	AULA	ESCUELA DE ING.CIVIL
LP - LABORATORIOS PESADOS	038	AULA	ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
LP - LABORATORIOS PESADOS	034	AULA DE CLASE	ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
LP - LABORATORIOS PESADOS	005	AULA	ESCUELA DE ING.CIVIL

Cerrar

Figura: 37 Vista de aulas disponibles en la sede para un horario seleccionado.

Modulo de Asignación automática total.

Usuario: Dirección de Admisiones.

En el caso de asignación a nivel de sede, es necesario tener en cuenta como se ven beneficiadas las diferentes escuelas al cabo del proceso, para esto se establecieron diferentes estrategias con el fin de encontrar la más adecuada a implantar. Entre estas estrategias encontramos la asignación total a nivel de sede atendiendo escuela por escuela dando prioridad a las escuelas con más necesidad, con esta estrategia encontramos que aquellas escuelas que son atendidas primero satisfacen todas o la mayoría de las necesidades de aula para la totalidad de sus grupos. Pero aquellas escuelas que son posteriormente atendidas, mantienen algunas necesidades de aulas. Por esta razón se decidió implementar una estrategia que colmara las necesidades de aulas a nivel general, sin discriminación, pero teniendo en cuenta la duración del horario de clase a satisfacer, con lo cual se desprenden los siguientes dos casos:

- Asignación automática de horarios grupo, con ordenación descendente, esto quiere decir que aquellos grupos que necesitan un slot o espacio para el desarrollo de la clase con mayor duración en horas, serán atendidos en primera medida.
- Asignación automática de horarios grupo, con ordenación ascendente, esto quiere decir que aquellos grupos que necesiten un slot o espacio para el desarrollo de la clase con menor duración en horas, serán atendidos primero.

Al realizar las pruebas de asignación automática a nivel de sede con estas dos estrategias, encontramos los siguientes resultados.

	% total ocupación	Promedio ocupación por escuela
Asignación ascendente	82.49%	77.45%
Asignación descendente	82.40%	81.85%

Tabla 5 Resultados de pruebas: Asignación total

Es pues comprendido que la asignación descendente cubre un porcentaje total a nivel de sede muy similar al de la asignación ascendente, pero en promedio por escuela satisface mejor las necesidades de las escuelas en particular. Es por esto, que la estrategia seleccionada para la implementación del prototipo final es la asignación automática de horarios grupo, con ordenación descendente según duración de la clase.

- El modulo de asignación automática total tiene como propósito ejecutar el proceso de asignación automática para todas las escuelas de una sede en un mismo evento.

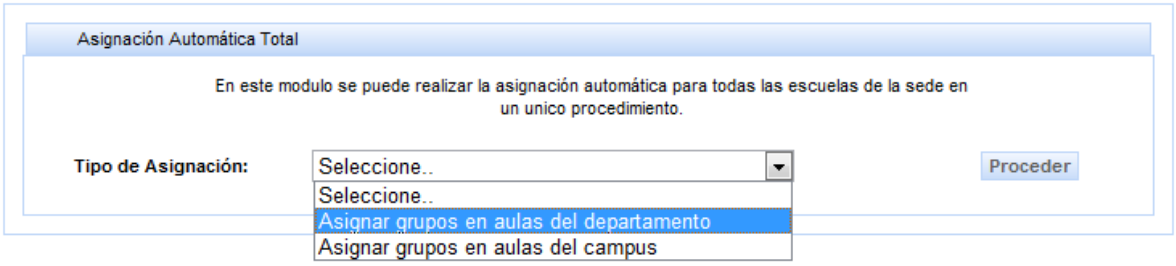


Figura: 38 Selección de tipo de asignación.

- El dialogo de advertencia le muestra al usuario que el tipo de asignación y el nivel de asignación que se dispondrá a ejecutar.

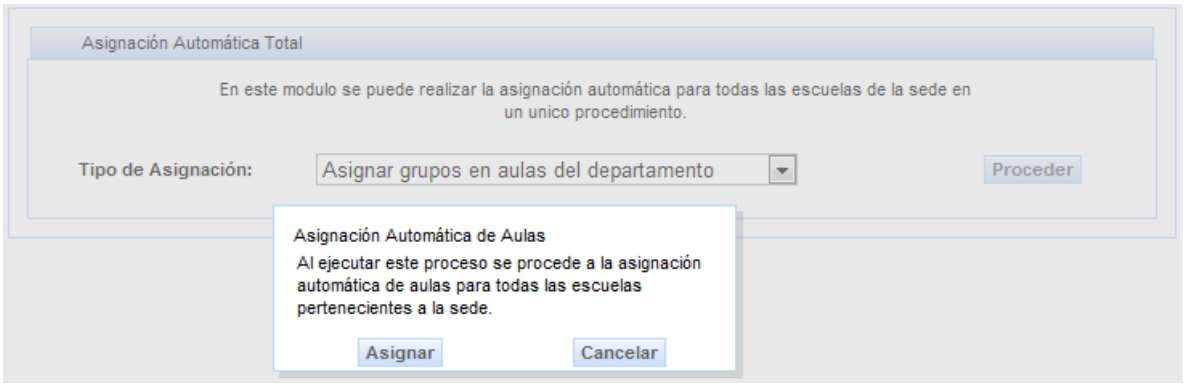


Figura: 39 Dialogo de advertencia.

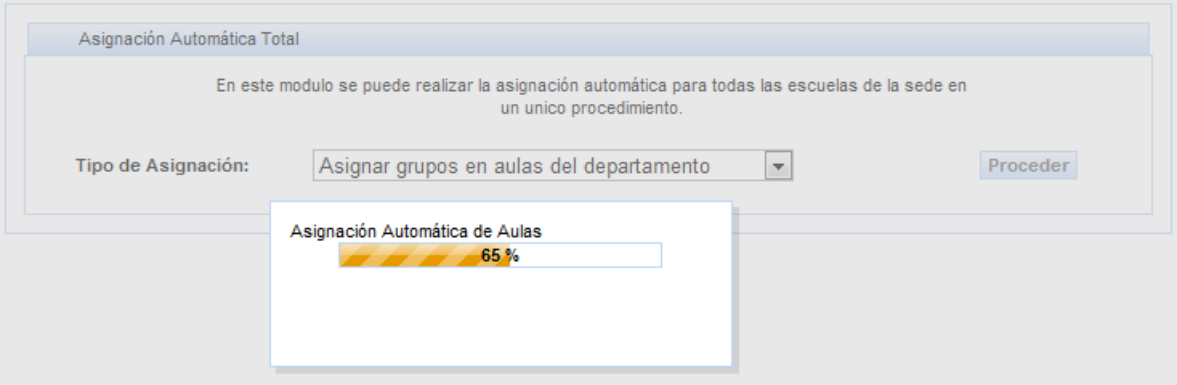


Figura: 40 Dialogo de progreso.

- Luego de terminado el procedimiento de asignación de aulas para todas las escuelas de la sede, la herramienta le presenta al usuario el listado de escuelas tramitadas en el proceso, cada uno de los ítems de la tabla presenta información sobre los recursos físicos representados en horas posibles a usar durante el ciclo semanal y dentro de los horarios para su uso, establecidos por la universidad, la cantidad de estos recursos que serán utilizados durante el semestre, el total de horas que fueron asignadas en la asignación automática, el numero de horarios grupo que intervinieron en la asignación automática y aquellos que no lo fueron asignados.

- Cada una de las escuelas tiene un botón de detalles dispuesto en la última columna para estudiar detalladamente los resultados de la asignación, en estos resultados el usuario tiene acceso a los listados de los horarios asignados y el listado de horarios que no tuvieron posible asignación para la escuela seleccionada, como se puede apreciar en la figura 38.

Listado de Escuelas / Departamentos

Código Escuela	Nombre Escuela	Horas/Aula de Escuela	Horas/Aula Disponibles	Horas Utilizadas	Horas Asignadas Automáticamente	Slots Asignados	Slots sin asignación	Detalles
6220	ESCUELA DE ARTES	1800	1636	164	17	11	2	
6570	ESCUELA DE ING.DE SISTEMAS	1620	1464	156	2	1	2	
6560	ESCUELA DE ING.MECANICA	1530	1213	317	5	2	0	
6130	ESCUELA DE FISICA	1440	1376	64	2	1	2	
6140	ESCUELA DE MATEMATICAS	1440	1053	387	9	4	3	
6530	ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES	1440	1299	141	11	4	0	
6550	ESCUELA DE ING.ELECTRICA, ELECTRONICA Y TELECOMUNI	1350	1222	128	44	22	1	
6520	ESCUELA DE DISENO INDUSTRIAL	1260	1120	140	0	0	1	
6540	ESCUELA DE ING.CIVIL	1170	1099	71	0	0	0	
6150	ESCUELA DE QUIMICA	990	794	196	4	3	4	

« « 1 2 3 Siguiente » »»

Total registros: 33

Figura: 41 Tabla de resultados de asignación de escuelas.

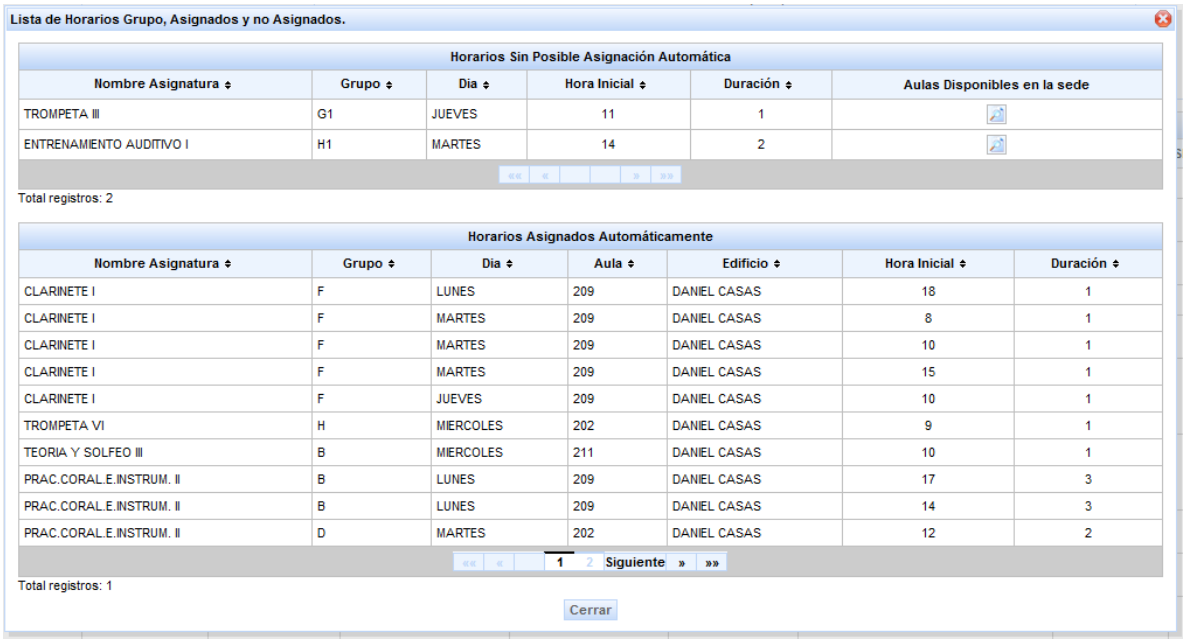


Figura: 42 Dialogo de resultados por escuela.

- El usuario en este modulo también tiene la posibilidad de acceder al listado de aulas posibles dentro de la sede para un grupo en particular que no fue asignado automáticamente.

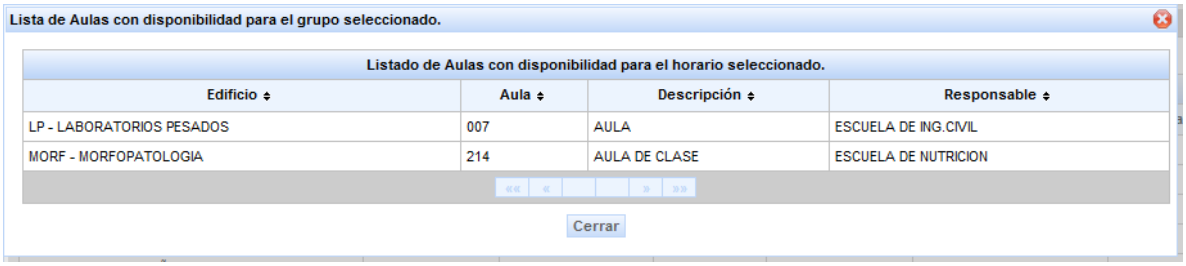


Figura: 43 Listado de posibles aulas en la sede.

4.5.3 Proyecto enmarcado en el esquema de Seguridad de la UIS.

Para este proyecto se utiliza el esquema de seguridad definido por la División de Servicios de Información para los diferentes sistemas de información que apoyan la gestión de la Universidad Industrial de Santander, el cual está basado en la estructura de roles – usuarios.

Los roles se establecen en cada una de las unidades académico administrativas, UAA, responsables de cada sistema, de acuerdo a las actividades que realizan. A cada uno de los roles definidos se le asocian los usuarios de acuerdo a las funciones que desempeñen.

4.5.3.1 Estructura de la Base de Datos soporte

La base de datos que soporta el esquema de seguridad contempla básicamente las siguientes tablas:

Sistema: Contiene información de los sistemas de información de la universidad. Para cada sistema se específica: Nombre, descripción del sistema, fecha y hora de creación en la base de datos, fecha y hora de inicio de vigencia del sistema, fecha y hora de cierre de vigencia del sistema.

Rol: contiene información de los diferentes roles definidos para cada sistema de información, como: Nombre asignado al rol, descripción del rol, fecha y hora de creación, fecha y hora de inicio de vigencia del rol, fecha y hora de cierre de vigencia del rol.

Usuario: Contiene información de los posibles usuarios de los sistemas de información. Entre esta información está: tipo y número de documento de identidad del usuario, fecha y hora de creación del usuario, fecha y hora de inicio de vigencia del usuario, fecha y hora de cierre de vigencia del usuario.

Sistema–rol: Contiene los roles definidos para cada uno de los sistemas de información, indicando: rol, sistema, fecha y hora de creación del rol – sistema, fecha y hora de inicio de vigencia del rol en el sistema, fecha y hora de cierre de vigencia del rol en el sistema.

Rol–usuario: Contempla los usuarios asociados a cada uno de los roles definidos, considerando: Rol, usuario, fecha y hora de creación del rol – usuario, fecha y hora de inicio de vigencia del usuario en el rol, fecha y hora de cierre de vigencia del usuario en el rol.

Menú–rol–sistema: Contiene los menús asociados a los roles en los distintos sistema de información, contemplando: Sistema de información, nombre del menú, descripción del menú, fecha y hora de creación del menú, fecha y hora de inicio de vigencia del menú asociado al rol, fecha y hora de cierre de vigencia del menú asociado al rol.

Opción–menú–rol: Contempla las opciones definidas para cada una de los posibles menús establecidos para cada sistema de información. Contiene: Nombre de la opción, descripción de la opción, nombre del menú superior, nombre del menú que contiene la opción, nombre del programa a ejecutar cuando la opción es la de más bajo nivel, fecha y hora de creación de la opción del menú, fecha y hora de inicio de vigencia de la opción, fecha y hora de cierre de la opción.

Tabla–sistema: Contiene información de las tablas que conforman la base de datos que soporta cada uno de los sistemas de información. Considera: Sistema de información, nombre de la tabla, descripción de la tabla.

Tipo–permiso: Establece para cada tabla de un sistema de información, los roles que tienen permisos para incluir registros, para modificar registros o para eliminar registros en ella. Contiene: Sistema de información, nombre de la tabla, clase de permiso (inclusión, modificación, eliminación de registros), fecha y hora de

creación del permiso, fecha y hora de inicio de vigencia del permiso, fecha y hora de fin de vigencia del permiso.

Acceso–tabla: Define para las tablas de un sistema de información si un rol tiene permiso sobre toda la información de la tabla o sobre una parte de esta. Considera: Sistema, nombre de la tabla, clase de acceso (total, parcial), fecha y hora de creación del permiso, fecha y hora de inicio de vigencia del permiso, fecha y hora de fin de vigencia del permiso.

Atributo–tabla: Establece los atributos sobre los cuales se debe controlar el acceso a una tabla, cuando a un rol se le concede permiso para hacer uso parcial de la información existente en una tabla. Contiene: Sistema de información, nombre de la tabla, nombre del atributo sobre el cual se controla el acceso a la tabla, descripción del atributo, fecha y hora de creación del atributo, fecha y hora de inicio de vigencia del atributo, fecha y hora de fin de vigencia del atributo.

Valor–atributo-proceso: Contiene los valores que deben tener los atributos definidos en cada tabla en la tabla atributo – tabla que permiten el acceso a la información asociada a estos valores. Específica: Sistema de información, nombre de la tabla, nombre del atributo, valor del atributo, descripción, fecha y hora de creación del valor del atributo, fecha y hora de inicio de vigencia del valor del atributo, fecha y hora de fin de vigencia del valor del atributo.

Acceso-sistema: Contempla el histórico de acceso que un usuario ha realizado a un sistema, identificando las opciones que ha seleccionado. Contiene: Login de usuario, rol, identificación de la sesión, sistema, opción seleccionada, fecha y hora de ingreso, fecha y hora de salida.

4.5.3.2 Entorno de Navegación

Para cada sistema de información, la UAA responsable define los roles necesarios para el adecuado uso del sistema de información de acuerdo a las funciones que realice y establece los usuarios asociados a cada uno de ellos.

Para cada rol se define el menú de inicio, el cual le permite a cada usuario que hace parte de este rol, empezar la navegación por las distintas opciones que le ofrece el sistema, hasta llegar al nivel más bajo en el cual se ejecuta el proceso que soporta la actividad que desea realizar.

Este entorno está soportado por las siguientes tablas de las base de datos del esquema de seguridad: Rol, usuario, sistema, sistema-rol, usuario-rol, menú-rol, opción-menú rol, descritas en “ESTRUCTURA DE LA BASE DE DATOS SOPORTE”.

4.5.3.3 Entorno de Control de Datos

Para los roles definidos en cada uno de los sistemas de información se especifican las tablas a las cuales puede acceder, el tipo de transacción que puede realizar sobre estas tablas (inclusión, modificación o eliminación de registros), si tiene acceso total o parcial a la información que contiene la tabla.

Para el acceso a la información de la tabla de manera parcial, se debe establecer el atributo o atributos seleccionados, los valores que estos atributos deben tener para autorizar el acceso solicitado.

Este entorno está soportado por las siguientes tablas de las base de datos del esquema de seguridad: Rol, usuario, sistema, sistema-rol, usuario-rol, tabla–sistema, tipo-permiso, acceso-tabla, atributo-tabla, valor atributo proceso, descritas en “ESTRUCTURA DE LA BASE DE DATOS SOPORTE”.

4.5.3.4 Auditoría

Todas las tablas que conforman la base de datos soporte del esquema de seguridad tienen el historial de las transacciones realizadas sobre cada una de ellas.

El historial de las transacciones de cada tabla contiene información de los registros incluidos en la tabla, de los registros modificados y de los registros eliminados. Adicionalmente, en cada transacción se especifica: Fecha de la transacción, hora de la transacción, tipo de transacción (I/U/D), tipo y número de documento de identidad del usuario que realizó la transacción, login, rol asociado, dirección IP y MAC del equipo desde el cual llevó a cabo la transacción.

4.5.4 Organización de Directorios.

Consideraciones iniciales

Para cada uno de los sistemas principales se debe crear un proyecto que siga los estándares para la estructura de directorios y nombres. En estos proyectos van las entidades propias de cada sistema, así como los componentes comunes a varias aplicaciones.

Las entidades van, en cada sistema, empaquetadas en un archivo con extension .jar, de tal manera que el desarrollo de una nueva aplicación no implica la implementación de éstas.

En el paquete general van los elementos que son utilizados por todos los módulos.

Las entidades van a estar todas en co.edu.uis.[Sistema].entidades.

Los servicios van a estar todos en co.edu.uis.[Sistema].servicios.

Para la creación de las entidades se toma la estructura de la base de datos. Por lo tanto se hace indispensable comenzar por este paso, seguido de la creación de los componentes para posteriormente desarrollar las demás aplicaciones.

Carpetas

Deben comenzar con minúscula y están basados en la estructura de directorios que se genera mediante “Seam-Gen” cuando se crea un nuevo proyecto. Las carpetas que se crean son las siguientes:

- bootstrap
- classes
- dist
- exploded-archives
- lib
- nbproject
- resources
- src
    - hot
    - main
    - test
- test-build

- view

A continuación se hace una breve descripción de las carpetas con las que el desarrollador tiene contacto directo.

Carpeta src

En esta carpeta se guardan los archivos fuentes de las clases del sistema distribuidos en dos subcarpetas de la siguiente manera:

Carpeta main

Contiene las clases de apoyo al proceso. No incluye a las entidades, pues estas estarán disponibles en otro lugar que se describe más adelante en este documento.

Carpeta hot

Contiene los casos de uso reflejados en los EJBs y sus interfaces.

Carpeta resources

Contiene los archivos de configuración de la aplicación.

Carpeta view

Contiene las páginas, imágenes, estilos y todo lo referente a aspecto de la aplicación.

Carpeta dist

Contiene los archivos jar, war y ear de la aplicación. Estos se generan automáticamente.

Carpeta lib

Contiene las librerías necesarias para la ejecución de la aplicación.

Nombres de archivos

Se sigue el mismo estándar dado para el nombre de variables descrito en el apartado 4.2.3 de éste documento.

Organización del código fuente dentro de la estructura de directorios.

Paquetes

La composición del nombre de los paquetes se sigue teniendo en cuenta la siguiente sintaxis:

co.edu.uis.[sistema].[aplicación].[módulo].[caso de uso]

Ejemplo:

co.uis.edu.co. sipqrs.administracion.mantenimiento.TipoSolicitud

En el caso de los proyectos principales (los que empaquetan entidades y componentes para cada sistema) los paquetes serán los siguientes:

Administración (src/action): co.edu.uis.[Sistema].[Aplicación].administracion

Clases generales (src/action): co.edu.uis.[Sistema].[Aplicación].general

Paginas (carpeta view)

Dentro de esta carpeta se tendrán la siguiente estructura:

- Plantillas : Plantillas del sitio
- imágenes
- estilos
- scripts (Si son necesarios. No es obligatorio y se deben evitar en la medida de lo posible).
- Administración : Todo lo referente a la administración

- generales : Páginas de uso global en la aplicación
- ayudas
- módulos

4.5.5 Documentación de programas fuente[11].

Nombre de archivos, variables, constantes, atributos, métodos y parámetros.

Los nombres dentro del código fuente siguen los parámetros establecidos en el estándar general de nombres, con las siguientes particularidades.

- Los nombre de los parámetros de los métodos empiezan con guión de piso, el resto del nombre sigue el estándar para nombres de variables. Una vez dentro del código fuente, se deben asignar a una nueva variable con el mismo nombre pero sin el guión de piso.
- Los nombres de constantes o variables finally, se escriben en mayúscula sostenida separando las palabras por guiones de piso.

Comentarios Java:

Los programas en Java pueden tener dos tipos de comentarios: comentarios de implementación y comentarios de documentación. Los comentarios de implementación están delimitados por /*.. */ cuando se trata de varias líneas y // cuando se trata de una línea. Los comentarios de documentación (conocidos como "comentarios JavaDoc") son específicos de Java y están delimitados por :

/** ... */. Los comentarios de documentación se pueden extraer a ficheros HTML usando la herramienta javadoc.

[11] Fuente: Estándares de la División de Servicios de Información de la Universidad Industrial de Santander

Los comentarios de implementación están destinados a comentar el código o para comentarios sobre la implementación en particular, buscan dar una orientación y hacer aclaraciones sobre la implementación a quien observa el código fuente. Los comentarios de documentación están destinados a describir la especificación del código, desde una perspectiva independiente de la implementación, están hechos para ser leídos por desarrolladores que pueden no tener necesariamente el código fuente a mano.

Los comentarios se deben usar para dar una visión general del código y para proporcionar información adicional que no esté disponible fácilmente en el propio código.

Orden dentro de los archivos de código fuente:

Cada archivo de código fuente Java debe contener una única clase o interfaz pública y deben seguir el siguiente orden:

- Sentencias package.
- Sentencias import.
- Comentario de documentación de la clase/interfaz.
- Declaraciones de clase e interfaz.
- Comentario de la implementación de la clase/interfaz si fuera necesario.
- Declaración de constantes (solo se declaran dentro de interfaces).
- Declaración de atributos. (la declaración se debe hacer uno por línea y al frente debe ir un comentario de implementación de ser necesario.)
- Declaración de variables. (Según los estándares no se deben declarar variables públicas, para ello se definen los métodos gets() y sets()). Las variables aquí declaradas serán privadas y se utilizarán como indicadores de estado de la clase. Las variables se deben declarar en el siguiente orden:
    - Variables estáticas: organizadas por ámbito o accesibilidad de la siguiente manera: públicas, protegidas, de paquete (sin modificador) privadas.
    - Variables de instancia: organizadas de la misma manera que las estáticas.

- Al igual que los atributos se debe declarar una por línea para facilitar los comentarios de implementación frente a ellas en caso de necesitarse.
- Comentario de documentación sobre cada uno de los constructores (no aplica a entidades).
- Declaración de constructores. (Siempre se declarará el constructor sin parámetros, se requiera o no una acción dentro de este. No aplica a entidades).
- Comentario de documentación sobre cada uno de los métodos que lo requieran.
- Declaración de métodos del la lógica del negocio: estos deben ir agrupados por funcionalidad en lugar de por ámbito, siendo el objetivo de esta organización el hacer el código de mas fácil lectura y compresión.
- Declaración de métodos gets() , sets() e is().
- Sobre escritura del método equals().
- Sobre escritura del método hashCode().
- Sobre escritura del método compare().
- Sobre escritura del método compareTo().

Especificaciones sobre el formato del código fuente:

Tabulación:

La unidad de tabulación será de 2 espacios.

Longitud de línea

Se deben evitar líneas de 80 o más caracteres ya que algunas herramientas no las manejan bien, además que líneas demasiado extensas hacen difícil la lectura del código.

Ruptura de líneas

- Cuando una expresión no cabe en una única línea, se debe romper de acuerdo a estos principios generales:
  - Romper después de una coma.
  - Romper antes de un operador.
  - Preferir las rupturas de alto nivel a las de bajo nivel. (por ejemplo no romper por operador dentro de paréntesis.
  - Alinear la nueva línea con el principio de la expresión al mismo nivel que la línea anterior y dar cuatro tabulaciones.

Declaraciones y sentencias

Sentencias package e imports:

Deben seguir las siguientes reglas de formato.

- Estas sentencias no van tabuladas.
- Se debe declarar una por línea.
- No utilizar comodines.

Variables locales:

Todas las variables locales deben inicializarse en el sitio en donde se declaran.

Las variables deben declararse al inicio de cada método y no esperar a declararlas hasta su primer uso. La única excepción son las variables de bucles que en Java se pueden declarar dentro de la sentencia for. Apropósito de estas variables, se utilizarán las letras de la i a la z en la medida que se vayan necesitando.

Clases, interfaces y métodos:

En las declaraciones se deben seguir las siguientes reglas de formato:

- Ningún espacio entre el nombre del método y el paréntesis "(" que abre su lista de parámetros.

- La llave de apertura "{" aparece al final de la misma línea que la sentencia de declaración.
- La llave de cierre "}" comienza una línea nueva tabulada para coincidir con su sentencia de apertura correspondiente, excepto cuando es un bloque vacío que la llave de cierre "}" debe aparecer inmediatamente después de la de apertura "{".
- Los métodos están separados por una línea en blanco.
- Las clases e interfaces principales no van tabuladas
- Los bloques de código que pertenecen a una clase o método van tabulados.

Sentencias simples:

Se debe escribir solo una sentencia por línea, y no escribir más de una separadas por punto y coma en la misma línea sin importar lo cortas que puedan ser.

Sentencias compuestas:

Las sentencias compuestas son sentencias que contienen una lista de sentencias encerradas entre llaves "{" y "}" y deben seguir las siguientes reglas de formato.

- Las sentencias internas deben estar tabuladas un nivel más que la sentencia compuesta.
- La llave de apertura debe estar al final de la línea que comienza la sentencia compuesta; la llave de cierre debe estar en una nueva línea y estar tabulada al nivel del principio de la sentencia compuesta.
- Las llaves se usan en todas las sentencias compuestas, incluidas las sentencias únicas, cuando forman parte de una estructura de control, como una sentencia if-else o un bucle for. Esto hace más fácil introducir nuevas sentencias sin provocar errores accidentales al olvidarse añadir las llaves.

Líneas en blanco

Las líneas en blanco mejoran la legibilidad resaltando secciones de código que están relacionadas lógicamente.

- En las siguientes circunstancias, siempre se deben usar dos líneas en blanco:
- Entre secciones de un archivo de código fuente.
- Entre definiciones de clases e interfaces.
- En las siguientes circunstancias, siempre se debería usar una línea en blanco:
- Entre métodos.
- Entre las variables locales de un método y su primera sentencia.
- Antes de un comentario de bloque o de una sola línea.
- Entre las secciones lógicas de un método, para mejorar la legibilidad.

Espacios en blanco:

Los espacios en blanco deberían usarse en las siguientes circunstancias:

- Una palabra reservada seguida por un paréntesis Por ejemplo:

```
while (true) {
        sentencias;
}
```

- En las listas de argumentos, debe haber un espacio después de cada coma.
- Todos los operadores binarios, excepto el operador punto (.) deben estar separados de sus operandos por espacios. Los operadores unarios (incremento ++, decremento --, negativo -) nunca deben estar separados de sus operandos. Por ejemplo: var++;

- Las expresiones de una sentencia for deben estar separadas por espacios en blanco. Por ejemplo:

  for (expr1; expr2; expr3);

- Las conversiones de tipo (cast) deberían estar seguidas de un espacio en blanco. Por ejemplo:

  variableEntera = (int) variableCadena;

5. CONCLUSIONES

- La herramienta de asignación automática de aulas permite reemplazar la antigua metodología manual que se ha estado utilizando para asignar los recursos físicos a los grupos que cursan cualquier tipo de asignatura en la Universidad Industrial De Santander.

- El uso de la herramienta de asignación automática reduce notablemente los tiempos empleados para la programación y gestión de cursos durante los periodos de oferta y demanda de cursos para la matricula que se lleva a cabo semestralmente.

- Teniendo la oportunidad de hacer la asignación de aulas automáticamente, los entes administrativos de la Universidad Industrial de Santander, como cada una de las directivas de escuela o departamento obtienen de manera fácil y transparente estadísticas sobre la capacidad de uso de sus recursos, con el fin de tener una mejor gestión sobre ellos.

- El problema de organización u ordenamiento de recursos en periodos de tiempo se ha presentado en la mayoría de instituciones universitarias, desde los años 70 se han estudiado soluciones que comprendan la gestión de los recursos incluyendo la programación de los cursos y la asignación de el profesorado que imparta las asignaturas. La herramienta de asignación automática de aulas es un componente que puede acoplarse fácilmente a un sistema más completo que permita a sus usuarios una gestión

automática de los cursos previniendo cualquier tipo de percance que se pueda presentar de manera manual.

- Con una ejecución completa del procedimiento de asignación automática de los recursos físicos para cada una de las escuelas, y la apropiada asignación total por parte de los directivos administrativos reduce notablemente la cantidad de grupos sin asignación de aula.

- Los directivos administrativos de la Universidad Industrial de Santander obtienen mayor control sobre la gestión de sus recursos físicos y del modo en que manejan cada una de las escuelas sus propios recursos.

- La gestión de los recursos es una tarea que conlleva un esfuerzo considerable pero siempre puede ejecutarse con mayor eficiencia y control. Dándole al administrador la posibilidad de maximizar la utilidad de sus recursos en base a procedimientos anteriores.

- Los procedimientos de asignación automática difieren en todos los casos en la metodología que se usa y los resultados que se obtienen, pero el refinamiento de una metodología permite extrapolar su uso a demás ambientes y necesidades.

6. RECOMENDACIONES

- Asignar un funcionario de la División de Servicios de Información de la Universidad Industrial de Santander que pueda dar soporte permanente a los usuarios del sistema.

- Inducir deseos de investigación sobre el tema de timetabling en estudiantes para perfeccionar el sistema y añadir funcionalidades que permitan un mejor rendimiento y mejores resultados.

- Dar a conocer a los directores de escuela las cualidades de la herramienta de asignación automática, con el fin de que encuentren viable el uso de la misma y se gestionen de manera más eficiente y equivalente los recursos de la Universidad Industrial de Santander.

BIBLIOGRAFÍA

Burke E. y Petrovic S. Recent research directions in automated timetabling. European Journal or Operational Research. Vol. 140, No. 2 (2002); p. 266-280.

Burke E., Jackson K., Kingston J., Ware R. Automated university timetabling: The state of the art. The Computer Journal. Vol. 40, No. 9 (1997); p. 565-571.

Carter M. y Laporte G. Examination timetabling: Algorithmic strategiaes and applications. Journar of the Operational Reserarch Society. Vol.47, No 3(1996); p.373-383

Díaz J. y Pinto K. Diseño y validación de un modelo matemático para el proceso de asignación de salones en el edificio camilo torres de la universidad industrial de santander. Tesis de Ingenieria (2009)

D. Corne, P. Ross, and H. Fang. Evoling timetables. In Lance C. Chambers, editor, the practical Handbook of Genetic Agorithms, volume 1, pages 219-276. CRC press, 1995.

FERRÉ GRAU, Xavier - SÁNCHEZ S. María Isabel. Desarrollo Orientado a Objetos con UML. Facultad de Informática - UPM

González J. Sánchez A. y Velásquez J. Galve, J. Algoritmica. Diseño y análisis de algoritmos funcionales e imperativos. Addison Wesley Iberoamericana, 1993.

Juan Jose Pantrigo Fernandez Resolución de Problemas de Optimización Dinamica mediante la Hibridacion entre Filtros de Particulas y Metaheuristicas Poblacionales. Universidad Rey Juan Carlos. Escuela Superior de Ciencias Experimentales y Tecnología.Departamento de Informatica, Estadística y Telematica. Tesis Doctoral,2005

Julio Brito Santana, Clara Campos Rodríguez, Félix C. García López, Miguel García Torres, Belén Melian Batista, José A. Moreno Pérez, J. Marcos Moreno Vega. Metaheurísticas: una revisión actualizada Grupo de Computación Inteligente. Universidad de La Laguna, 2004.

Lewis R. A survey of metaheuristic-based techniques for university timetabling problems. OR Spectrum, (2007).

Marte M. Models and Algorithms for School Timetabling-A Constraint- Programming Approach. Tesis de Doctorado, Universidad de Munich (2002).

Mulvey J. A classroom/time assignment model. European Journal of Operational Research. Vol. 9, (1984); p.64-70.

PRESSMAN, Roger. Ingeniería del Software. Un enfoque práctico. Quinta Edición. McGraw Hill. España. 2005.

Wren, A. Scheduling, timetabling and rostering – A special relationship? In: Burke and Ross (1996) pp. 46–75

REFERENCIAS BIBLIOGRAFICAS

[1] Burke E. y Petrovic S. Recent research directions in automated timetabling. European Journal or Operational Research. Vol. 140, No. 2 (2002); p. 266-280.

[2] Burke E., Jackson K., Kingston J., Ware R. Automated university timetabling: The state of the art. The Computer Journal. Vol. 40, No. 9 (1997); p. 565-571.

[3] Lewis R. A survey of metaheuristic-based techniques for university timetabling problems. OR Spectrum, (2007).

[4] Marte M. Models and Algorithms for School Timetabling-A Constraint-Programming Approach. Tesis de Doctorado, Universidad de Munich (2002).

[5] Díaz J. y Pinto K. Diseño y validación de un modelo matemático para el proceso de asignación de salones en el edificio camilo torres de la universidad industrial de santander. Tesis de Ingenieria (2009)

[6]Wren, A. Scheduling, timetabling and rostering – A special relationship? In: Burke and Ross (1996) pp. 46–75

[7] Carter M. y Laporte G. Examination timetabling: Algorithmic strategiaes and applications. Journar of the Operational Reserarch Society. Vol.47, No 3(1996); p.373-383

[8] González J. Sánchez A. y Velásquez J. Galve, J. Algoritmica. Diseño y análisis de algoritmos funcionales e imperativos. Addison Wesley Iberoamericana, 1993.

[9] D. Corne, P. Ross, and H. Fang. Evoling timetables. In Lance C. Chambers, editor, the practical Handbook of Genetic Agorithms, volume 1, pages 219-276. CRC press, 1995.

[10] Mulvey J. A classroom/time assignment model. European Journal of Operational Research. Vol. 9, (1984); p.64-70.

[11] Juan Jose Pantrigo Fernandez Resolución de Problemas de Optimización Dinamica mediante la Hibridacion entre Filtros de Particulas y Metaheuristicas Poblacionales. Universidad Rey Juan Carlos. Escuela Superior de Ciencias Experimentales y Tecnología.Departamento de Informatica, Estadística y Telematica. Tesis Doctoral,2005

[12] Julio Brito Santana, Clara Campos Rodríguez, Félix C. García López, Miguel García Torres, Belén Melian Batista, José A. Moreno Pérez, J. Marcos Moreno Vega. Metaheurísticas: una revisión actualizada Grupo de Computación Inteligente. Universidad de La Laguna, 2004.

[13] PRESSMAN, Roger. Ingeniería del Software. Un enfoque práctico. Quinta Edición. McGraw Hill. España. 2005.

[14] FERRÉ GRAU, Xavier - SÁNCHEZ S. María Isabel. Desarrollo Orientado a Objetos con UML. Facultad de Informática - UPM

SITIOS WEB

- www.wikipedia.org
- http://www.sparxsystems.com.ar/
- http://download.oracle.com/docs/cd/E17802_01/j2ee/javaee/javaserverfaces/2.0/docs/pdldocs/facelets/index.html
- http://livedemo.exadel.com/richfaces-demo/richfaces/comboBox.jsf?tab=usage&cid=170149