UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

Desarrollo de un prototipo de aplicación web para automatizar y optimizar los procesos académicos de la Coordinación de Postgrado de la Facultad de Ciencias de la Universidad Central de Venezuela

Trabajo Especial de Grado presentado ante la ilustre
Universidad Central de Venezuela
por los bachilleres:
Andrés Ramírez Ruiz.
Gabriel Plaza Lamuño.
para optar por el título de
Licenciado en Computación

Tutor: Andrés Sanoja.

Caracas, Venezuela
Julio, 2009

Acta

Quienes suscriben miembros del jurado designado por el Consejo de Escuela de Computación de la Facultad de Ciencias, para examinar el Trabajo Especial de Grado presentado por los bachilleres Gabriel Plaza Lamuño portador de la Cédula de Identidad No. V-17.704.354, Andrés Ramirez Ruiz portador de la Cédula de Identidad No. V-16.658.670, con el título "*Desarrollo de un prototipo de aplicación web para automatizar y optimizar los procesos académicos de la Coordinación de Postgrado de la Facultad de Ciencias de la Universidad Central de Venezuela*", a los fines de optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue, este trabajo por cada uno de los miembros del jurado, se fijó el día *17 de Julio de 2009* a las *9:30am*, para que sus autores lo defendieran en forma pública en el *Auditorio Manuel Bemporad* de la Escuela de Computación, Facultad de Ciencias de la Universidad Central de Venezuela, mediante una presentación oral de su contenido, luego de lo cual respondieron a las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente Acta, en Caracas el Décimo séptimo día del mes de julio del año dos mil nueve, dejándose también constancia de que actuó como Coordinador del Jurado el profesor Andrés Sanoja.

- -	- -
Profa. Jossie Zambrano (Principal).	Prof. Antonio Silva (Principal).

- -

Prof. Andrés Sanoja (Tutor).

Dedicatorias y Agradecimientos

Antes que nada, quiero agradecer a mis padres Raul y Thais, por haberme guiado por el mejor camino, por estar siempre presente en los buenos y malos momentos, gracias a ellos me encuentro ante ustedes, celebrando la finalización de una etapa que si bien ha sido larga y con muchos obstáculos, no ha dejado de ser gratificante y sin ustedes no hubiera podido llegar a ser la persona que soy hoy en día.

A mi hermano Alejandro que ha estado presente siempre en todo momento, que me ha dado el afecto y apoyo para seguir siendo una persona mejor día a día.

A mi sobrino Alejandro para quien he tratado de ser el modelo a seguir, dandole todo el mejor apoyo y consejo para que siga por el mejor camino.

A mis primos y primas que han sido como mis mejores amigos en todo los sentidos, siendo compañeros y hasta cómplices en todas mis vivencias, contarán conmigo para toda la vida.

A mi buen amigo y compañero durante toda esta aventura Gabriel, por todas esas palabras de aliento, consejos y confianza que siempre haz tenido en mi, y bien sabes por todo los acontecimientos que hemos pasado para llegar a este punto tan gratificante y especial.

A mis amigos y amigas con quien comparti durante toda mi carrera académica, laboral y emocional, gracias por estar siempre presentes y contar con ustedes cada día.

A nuestro tutor Andrés Sanoja por indicarnos siempre el mejor camino a seguir y prestarnos toda su colaboración para poder finalizar este proceso.

A todos los mencionados y faltantes, mis más sinceras gracias.

Andrés Ramírez

Ante todo doy gracias a mí Dios Todopoderoso, mí maestro, a mís padres Miguel e Irene, mís mentores, y en especial a mí hermano Miguel, aunque tu partida muy pronta fue, en mi corazón estarás siempre presente. Ustedes son mís pilares, sin sus enseñanzas, nada de esto podría ser posible.

A mís abuelos, mís tios y primos siempre atentos y dando su apoyo incondional en las buenas y malas.

A Luz, por ser la mejor amiga, compañera y pareja, gracias por tener la paciencia y brindar ese apoyo cuando más lo necesitaba.

A mi gran amigo y compañero Andrés, fiel desde el incio de esta fascinante aventura, gracias por tu compresión y tolerar todos mis locuras, ser esa mano amiga que está cuando otros te olvidan, orgulloso de ser tu compañero.

A nuestro profesor y tutor Andrés Sanoja, guía y orientador, su apoyo, confianza y sinceridad depositada es un sentimiento muy apreciado por nosotros.

A los profesores Jossie, Sergio y Antonio, por su apoyo en la logística del desarrollo de este proyecto y cooparticipes en el desarrollo como profesionales, su carisma y respeto no podrán ser olvidados.

A mis compañeros y amigos, gracias amigos por compartir junto a mi esta bella experiencia.

Finalmente y no menos importante a mi querida Universidad, tus valores y lo que representan fueron estimulos para continuar cuando las adversidades y contratiempos se avecinaban, en mi espíritu vivirá por siempre el sentimiento Ucevista.

"Seguir venciendo las sombras será mi cometido..." *Gabriel Plaza*

Resumen

En la unidad académica de la Coordinación de Postgrado de la Facultad de Ciencias de la Universidad Central de Venezuela intervienen un determinado número de procesos donde se destaca su complejidad y flujo de información así mismo el como afectan de manera directa o indirecta el tiempo de respuesta para llevar á cabo la realización efectiva de los diversos trámites internos y externos que demanda, y se demandan en la coordinación antes mencionada. Éste Trabajo Especial de Grado tiene como Objetivo desarrollar un prototipo que permita sistematizar los procesos académicos de la Coordinación de Postgrado y ayudar a la comunidad que la satisfacen. Dicha comunidad está conformada por estudiantes, docentes y personal administrativo que laboran en dicha organización. Por esa razón debe estar enfocada en adaptarse a las necesidades y dar soporte a todas las solicitudes estudiantiles, del personal administrativo, asi como también a los procesos académicos para los docentes.

El proceso de desarrollo utilizado para la gestión de esta aplicación fue una adaptación de la metodología Programación Extrema (XP), facilitando la realización de la parte práctica del presente trabajo de investigación de manera rápida, sencilla, eficiente y documentada. Dentro de los requerimientos contemplados en la aplicación se tienen los siguientes: La gestión de los procesos de preinscripción o admisión de estudiantes, selección e inscripción de estudiantes, gestión de solicitudes estudiantiles. Apoyo a los procesos académicos de la planta docente, entre otros. La investigación contempló el estudio e incorporación del conocimiento previamente desarrollado en los sistemas referentes a: el Sistema de Control de Estudios (CONEST) y el Sistema de Gestión de Postgrado (SIGEPOST). Del primero se tomaron el modelo de datos, los lineamientos generales y se incorporaron como elementos fundamentales en el análisis y el diseño del sistema; del segundo se tomaron aspectos del modelo de datos y los registros. En este Trabajo Especial de Grado se presenta el primer prototipo que muestra la confluencia de estos dos enfoques.

Palabras Clave: Coordinación de Postgrado, académico, automatización, aplicación Web, postgrados, Ruby on Rails, estudiantes, docentes.

Índice general

Índice de figuras

Índice de cuadros

Introducción

La sociedad actual se ha visto influenciada por el uso de Internet, donde un creciente número de organizaciones cuentan con aplicaciones Web que les permiten difundir su información institucional y dar a conocer sus actividades a un grupo de usuarios en crecimiento como son los usuarios Web. Estas aplicaciones son conocidas como Sitio Web y su calidad es muy variada, considerándose más exitosas aquellas que presentan contenidos relevantes y útiles, que son constantemente actualizados, y que facilitan la visita de los usuarios. Al ser la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela una institución actualizada y a la par en el uso de tecnologías de Internet se considera necesario la ayuda para que la Coordinación de Postgrado de dicha institución cuente con una aplicación eficiente y eficaz que refleje las actividades y procesos para sus usuarios.

Actualmente los diferentes postgrados que se dictan en la Facultad de Ciencias y que son regulados a través de la dirección de postgrado de esta Facultad, cuentan con un sistema que cubre con ciertos aspectos de los procesos. Se observó que aún hay tareas que se llevan a cabo de forma manual o semi-automática, lo que impacta directamente el tiempo de respuesta en comparación a los tiempos de respuesta de un sistema automatizado que tenga una mayor cobertura y de acuerdo a sus necesidades. Por otro lado se tiene el sistema de Control de Estudios (CONEST) que ha tenido una experiencia en pregrado, y es de ínteres de la comunidad de la Facultad estandarizar y unificar los dos sistemas.

En tal sentido, el objetivo fundamental de este Trabajo Especial de Grado consiste en la implementación de una aplicación Web para la Coordinación de Postgrado que tenga como función automatizar su información institucional de procesos académicos y administrativos. Esta aplicación se denomina como Sitema Conest Postgrado que estaría compuesta en procesos académicos estudiantiles como: preinscripcion, inscripcion, selección, solicitud de constancias, solicitud de retiros de materia, solicitud de desincorporación y procesos académicos docentes como: ofertar materias, planificación docente, calificación de forma definitiva y calificación de forma continua.

Para concretar lo planteado y cumplir con el objetivo mencionado anteriormente se dará una visión de cada uno de los capítulos, presentados a los largo de este documento,

el cual se ha estructurado de la siguiente manera:

- **Capítulo I: Marco Referencial** que comprende la descripción en cuanto a la Coordinación de Postgrado, desde su perfil hasta cómo se realiza actualmente las tareas internas en su forma más general, así como también los postgrados por escuela, enfocándonos en la unidad académica.

  Adicionalmente se describe para esta investigación lo relevante e importante sobre el sistema CONEST, contemplando el enfoque al cual está dirigido, los servicios que ofrecen, características, las descripciones de cada uno de sus módulos, tecnologías y estado actual del sistema.

- **Capítulo II: Marco Metodológico** donde se explican y documentan todos los pasos realizados para lograr el desarrollo del sistema siguiendo el ciclo de los procesos ágiles adaptado a la programación extrema: planificación, diseño, codificación y pruebas, para así poder realizar la parte práctica del presente trabajo de investigación de manera rápida, sencilla, eficiente y documentada. También se presenta el análisis inicial del sistema de forma global, mostrando las historias de usuario, la metáfora del sistema y sus especificaciones técnicas.

- **Capítulo III: Marco Aplicativo** se precisan y detallan todos los pasos efectuados por medio de iteraciones para lograr el desarrollo del sistema. También se precisa el contexto de desarrollo en el que se basa la aplicación, se especifica el proyecto en el que se va a trabajar y el plan a tomar para evaluar cada iteración.

- **Capítulo IV: Conclusiones** donde se dará muestra de los resultados, recomendaciones y aportes de la aplicación desarrollada para futuros trabajos relacionados.

- Finalmente se presentan las Referencias Bibliográficas consultadas durante el desarrollo del documento y aplicación.

Objetivo General:

Analizar, diseñar y construir un sistema automatizado y orientado a la Web adjunto con el sistema CONEST, que permita llevar el control de los procesos académicos que se llevan a cabo en la Coordinación de Estudios de Postgrado logrando integrar en un único sistema la gestión de pregrado y postgrado, contribuyendo en la disminución de los costos y los tiempos de respuesta inmersos en el desarrollo de sus procesos y las actividades relacionadas.

Objetivos Específicos:

A continuación se enumeran los objetivos expecifícos del presente trabajo expecial de grado:

- Adaptar la metodología XP para la implantación e implementación de la aplicación Web que gestiona los aspectos relacionados a la ejecución de los procesos académicos de la Coordinación y los postgrados.
- Diseñar e implementar el modelo de la base de datos que permita reflejar la información concerniente a la Coordinación de Postgrado en cuanto a los aspectos académicos.
- Implementar funcionalidades que permitan llevar a cabo procesos inherentes a la admisiones estudiantiles.
- Implementar funcionalidades que permitan llevar a cabo procesos inherentes a la selecciones estudiantiles.
- Implementar funcionalidades que permitan llevar a cabo procesos inherentes a la inscripciones estudiantiles.
- Implementar funcionalidades que permitan llevar a cabo los procesos inherentes a las solicitudes estudiantiles.
- Implementar funcionalidades que permitan llevar a cabo los procesos inherentes a ofertar materia por docente.
- Implementar funcionalidades que permitan llevar a cabo los procesos inherentes a la programación docente.
- Implementar funcionalidades que permitan llevar a cabo los procesos inherentes a la calificación docente.
- Someter a pruebas de verificación a la aplicación y comprobar su correcto funcionamiento.
- Realizar peticiones de servicio en cuanto a consultas al sistema caja .

Capítulo 1

Marco Referencial

La finalidad de este capítulo es presentar las bases conceptuales que sirvieron de fundamento en el desarrollo de esta investigación. El mismo se divide en dos secciones, las cuales se describen a continuación.

En la primera sección, se hace una descripción de la organización a la cual estará destinada el sistema a implementar, en este caso estará orientado a la Coordinación de Postgrado. Se tocaran puntos referentes a esta coordinación, su estructura organizacional y los procesos involucrados en el sistema.

En la segunda sección de este capítulo, se hace referencia al sistema CONEST de la División de Control de Estudio de la Facultad de Ciencias de la Universidad Central de Venezuela, destacando los módulos que lo componen, su plataforma tecnológica y funcionamiento.

1.1. Coordinación de Postgrado [1]

En este capítulo se describe el perfil de la Coordinación de Postgrado, las unidades en la cual está estructurada y las funciones de cada una, como también los procesos de gestión académica que intervienen en dicha coordinación.

1.1.1. Perfil de la Coordinación de Postgrado

La Coordinación de Postgrado de la Facultad de Ciencias de la Universidad Central de Venezuela (UCV), tiene por objetivo primordial coordinar y velar por el funcionamien-

to de todas las actividades concerniente a cualquiera de los postgrados que están bajo su responsabilidad, para ello las funciones principales que son llevadas a cabo por la coordinación son:

- Dirigir y representar al postgrado de la Facultad de Ciencias y velar por su buen funcionamiento.
- Convocar y presidir las reuniones de la comisión de estudio de postgrado de la Facultad de Ciencias.
- Preparar la orden del día de cada reunión de la comisión y la minuta del acta de la sesión precedente.
- Informar periódicamente a la comisión de las gestiones propias de su cargo
- Asistir a las reuniones de consejo de estudio de postgrados de la UCV y a las del consejo de Facultad
- Elaborar el presupuesto para el funcionamiento del postgrado de la Facultad.
- Ejecutar el presupuesto ordinario del postgrado de la Facultad.
- Informar y divulgar a la comunidad universitaria y nacional, todo lo relativo a los programas de postgrado de la Facultad que ameriten difusión.
- Asistir al coordinador de postgrado en las reuniones de comisión de mesa y en la comisión de estudio de postgrado
- Elaborar minutas ordinarias, complementarias, puntos varios y de admisión inherentes a las reuniones de la comisión de estudios de postgrado así como también, las dirigidas al Consejo de Facultad
- Organizar y preparar el material que se utiliza en cada reunión de la comisión.
- Elaborar las actas resultantes de las reuniones de la comisión.
- Elaborar oficios dirigidos al Consejo de Facultad solicitando nombramientos de jurados, inclusión de profesores a la planta profesoral, aperturas de asignaturas, aprobación de proyecto entre otros.
- Emitir oficios para el consejo de estudios de postgrado solicitando la designación de jurados para tesis doctorales, reconocimientos de créditos y correspondencia en general
- Atender a coordinadores, profesores, estudiantes, secretarias y público en general, para brindar información relacionada con la unidad académica.

- Poner en práctica programas destinados a la consecución de recursos financieros
- Representar los postgrados de la Facultad de Ciencias ante el Consejo de Facultad

1.1.2. Estructura Organizacional

La Coordinación de la Facultad de Ciencias esta a cargo de un coordinador y a su vez esta dividida en cinco unidades de trabajo y el CIAD (Centro de Información y Automatización de Datos). Vease figura 1.1

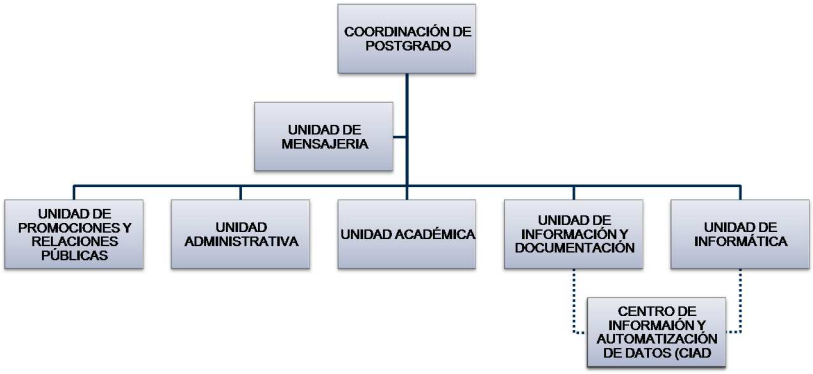

Figura 1.1: Estructura Organizativa de la Coordinación de Postgrado

Cada una de estas unidades tiene unas funciones específica dentro de la coordinación, la cual a grandes rasgos se indican a continuación:

- **Unidad Administrativa:** Esta unidad se encarga de las siguientes actividades:
  - Controlar y ejecutar las acciones necesarias que permitan prestar los servicios y atención adecuada a cada uno de los postgrados de la Facultad de Ciencias.
  - Asignar ordinarias entregadas por el C.N.U.(Centro Nacional de Universidades) e ingresos propios que controla la Coordinación a cada postgrado de la Facultad de Ciencias.
  - Manejar todos los ingresos y egresos generados en la misma.
  - Llevar un control de la cuenta de ingresos propios de la Facultad de Ciencias y de la cuenta del consejo de estudio de postgrado de la Facultad de Ciencias.
  - Elaborar solicitudes de cheques, viáticos, órdenes de pedidos, órdenes de pagos y requisiciones.

- Elaborar minutas y actas de la comisión de presupuesto, adquisición y compra de materiales, equipos y suministros.
- Establecer seguimiento y actualización de los pagos de estudiantes, seguimiento de subvenciones de matriculas y becarios, control de caja chica, atención de estudiantes y profesores.

• **Unidad Académica:** Esta unidad se encarga de las siguientes actividades:

- Elaboración de minutas y actas de las reuniones de la CmEPG (Comisión de Estudios de Postgrado).
- Realizar seguimiento de los trámites tratados en las reuniones de la CmEPG.
- Preparar y montar las reuniones del CmEPG.
- Realizar las minutas de las reuniones de la Comisión de Mesa.
- Enviar trámites al Consejo de Estudio de Postgrado y al Consejo de Facultad.
- Elaborar oficios para la Coordinación de Postgrado.

• **Unidad de Información y Documentación:** Esta unidad se encarga de las siguientes actividades:

- Proveer a los postgrados de la Facultad de Ciencias de los recursos tecnológicos, materiales y humanos que permitan la organización y difusión de los fondos documentales de sus archivos.
- Diseñar productos y servicios que permitan el acceso y difusión de la información de postgrado.
- Servir de interfaz en la transferencia de información entre la CmEPG, Consejo de Estudios de Postgrado y el Consejo de Facultad.
- Llevar memoria institucional de los postgrados de la Facultad de Ciencias.
- Aperturar, organizar, preservar y controlar los expedientes de los estudiantes regulares, irregulares, egresados y profesores de los postgrados de la Facultad de Ciencias.
- Realizar informes de gestión.
- Coordinar proyectos archivísticos.
- Apoyar a los postgrados de la Facultad de Ciencias en todo lo relacionado con la información.

• **Unidad de Informática:** Esta unidad se encarga de las siguientes actividades:

- Realizar mantenimiento a la red informática de la Coordinación del Postgrado.

- Asesorar a los postgrados de la Facultad de Ciencias en las adquisiciones y montaje de equipos informáticos.
- Programar, diseñar y mantener las bases de datos de la Coordinación de Postgrado.
- Diseñar y mantener las páginas Web de los postgrados de la Facultad de Ciencias.
- Servir de intermediario entre la Coordinación de Postgrado, el Centro de Computación de la Facultad de Ciencias y la Dirección de Informática de la Universidad Central de Venezuela.
- Promocionar los postgrados de la Facultad de Ciencias vía Internet.
- Prestar servicios de atención al público.
- Dictar cursos de computación al personal de la Coordinación de Postgrado.
- Controlar la seguridad de la red de la Coordinación de Postgrado.

• **Unidad de Promociones y Relaciones Públicas:** Esta unidad se encarga de las siguientes actividades:

- Promocionar los postgrados de la Facultad de Ciencias a nivel nacional e internacional.
- Preparar los actos de graduación de los postgrados.
- Atender a los graduandos, asesorándolos y facilitándoles la realización de los trámites para la graduación.
- Diagramar y publicar el material de difusión de la Coordinación de Postgrado y los actos de graduación.
- Realizar el proceso de pre-inscripción de los postgrados.
- Atender al público y servir de intermediario en relaciones públicas con personalidades y entidades públicas o privadas.

• **Unidad de Mensajería:** Esta unidad se encarga de las siguientes actividades:

- Distribuir y entregar la correspondencia tales como: oficios, memorandos, libros, periódicos, folletos y cheques entre las distintas dependencias a las cuales van dirigidas.
- Reproducir material académico y administrativo en la fotocopiadora.
- Preparar y clasificar correspondencia general.
- Manejar equipos de oficina como: encuadernadora, plastificadora, fotocopiadora, guillotina, engrapadora industrial, entre otros.
- Instalar equipos de computación.

- Trasladar artículos de oficina de una dependencia a otra, dentro de las instalaciones de la institución.
- Colaborar en cualquier eventualidad que se presente en las otras unidades de la Coordinación de Postgrado siguiendo instrucciones dada por el supervisor inmediato.
- Cumplir con las normas y procedimientos de seguridad integral establecidos por la organización.

• **Centro de Información y Automatización de Datos(CIAD):** Es un centro que brinda a la comunidad de Postgrado de la Facultad de Ciencias de la UCV, servicios informáticos que faciliten las labores académicas y de investigación, compuesto por 8 puntos de conexión y conectados con el SIDEP-UCV (Sistema de Información Digital de Estudios de Postgrado de la UCV)

1.1.3. Procesos de la Unidad Académica

En esta sección se estudiará, analizará y detallará los procesos académicos que son llevados a cabo en conjunto con los trece (13) postgrados de la Facultad de Ciencias de la Universidad Central de Venezuela.

1.1.3.1 Descripción del servicio

Controlar y ejecutar los trámites académicos que permiten prestar servicio y atención a los estudiantes pertenecientes a los distintos postgrados de la Facultad.

1.1.3.2 Descripción del proceso

Manejar y controlar la información de los estudiantes durante su permanencia en el postgrado. También, se encarga de la elaboración de minutas y actas de las reuniones de la Comisión de Estudios de Postgrado (CmEPG), seguimiento de los trámites tratados en las reuniones, preparación y montaje de las reuniones, envío de trámites al Consejo de Estudios de Postgrado y al Consejo de Facultad, elaboración de oficios para la Coordinación de Postgrado, entre otros.

Cabe destacar que los procesos internos que se manejan en esta unidad, involucran la intervención de otras dependencia externas a la Coordinación, pero que están en constante comunicación, como son el control de estudio de cada postgrado de la Facultad de Ciencias, siendo en total trece (13) control de estudio. Vease figura 1.2

A continuación se muestran los procesos internos. La representación de los mismos fue mediante diagramas de actividades, permitiendo describir y establecer la secuencialización de las actividades involucradas en cada uno de los procesos e identificar sus actores y/o responsables [13]. Así mismo se presenta con una breve descripción de cada uno.

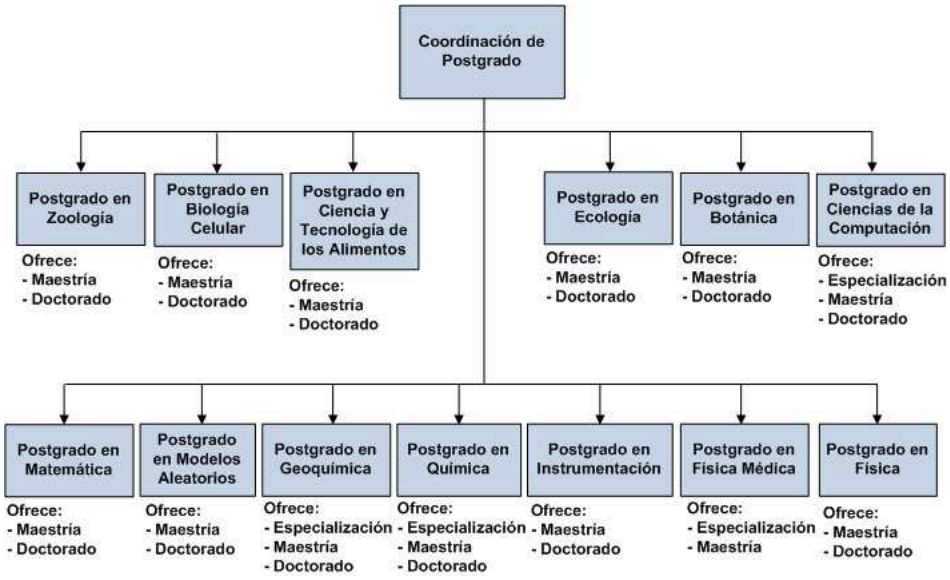

Figura 1.2: Postgrados de la Facultad de Ciencias

Proceso de Admisión.

En el proceso de admisión, el aspirante al postgrado, cancela en caja un monto establecido por la Coordinación para obtener el material de preinscripción. Este llena una planilla de solicitud que se obtiene desde la página Web del postgrado, y la entrega junto con otros recaudos en la coordinación de postgrado de la Facultad de Ciencias. En la recepción de la Coordinación de Postgrado se revisa la documentación, esta se registra en una hoja de cálculo, se hace un duplicado, se archiva el original y se envía copia al Control de Estudio del Postgrado solicitado.

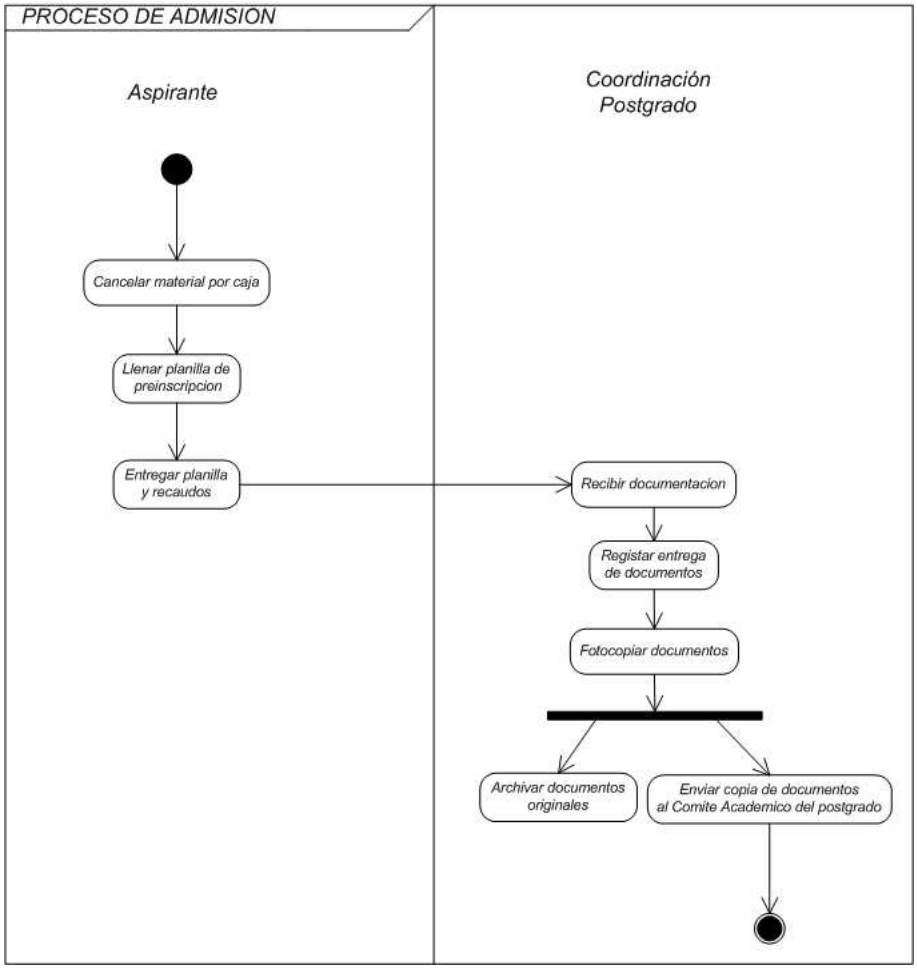

Figura 1.3: Diagrama de actividades del proceso de admisión

Proceso de Selección.

En el proceso de selección, el aspirante realiza una prueba de suficiencia de un idioma extranjero, evaluación de credenciales y entrevista. El comité académico del postgrado respectivo evalúa los resultados obtenidos. Se crea un oficio de admisión indicando si el mismo fue aceptado o no y se envía a la recepción de la Coordinación de Postgrado, en donde se registra el documento. En el caso que el aspirante fuese aceptado, se le asigna un profesor guía y/o tutor, se crea el expediente del aspirante y un oficio-respuesta, enviando una copia al comité académico del postgrado y otra al aspirante quien además retira en la Coordinación de Postgrado una planilla para formalizar su inscripción por Secretaría Central, mientras que el comité académico del postgrado archiva el oficio-respuesta. En el caso que el aspirante no fue aceptado, se desecha la documentación.

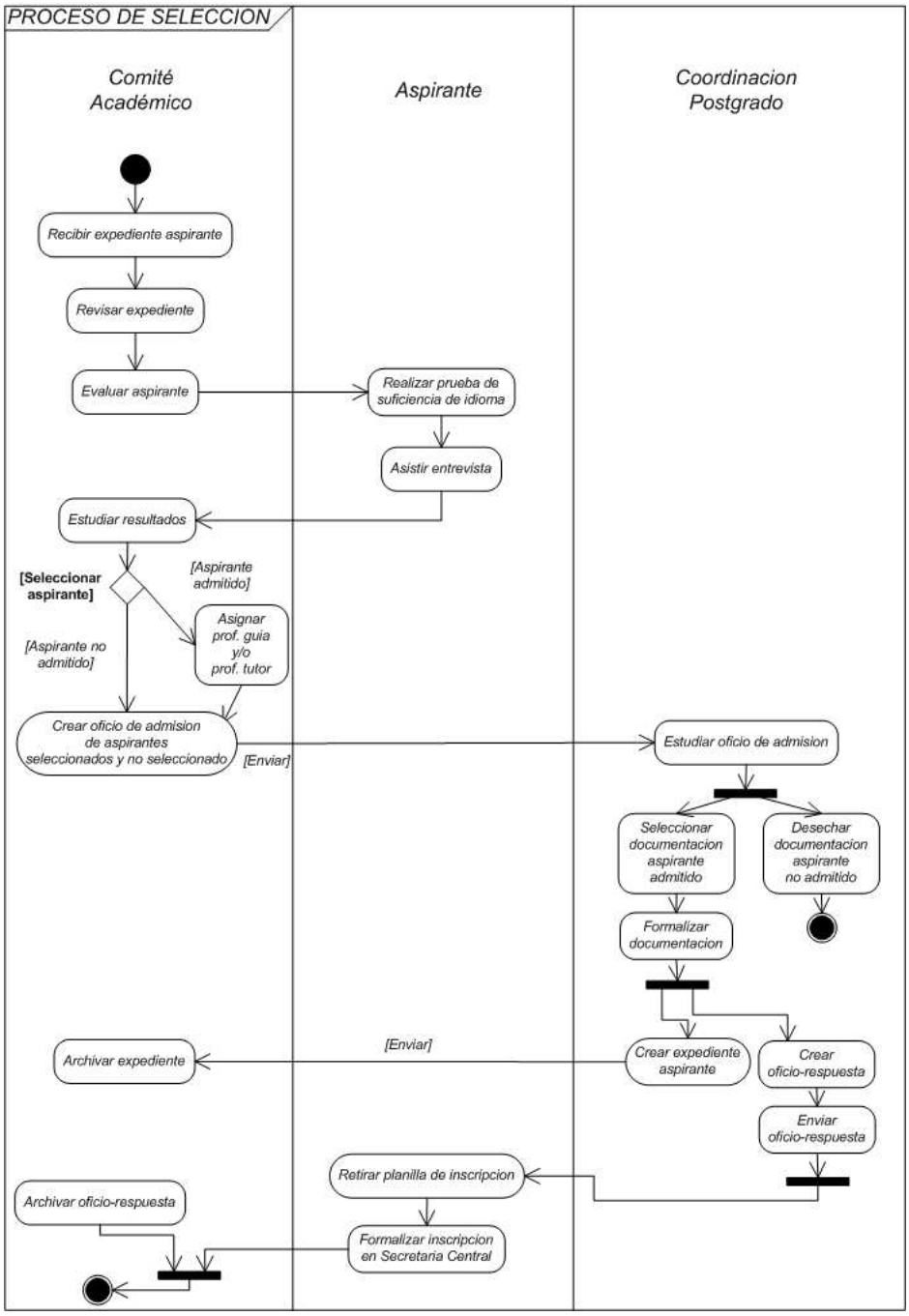

Figura 1.4: Diagrama de actividades del proceso de selección

Proceso de Inscripción.

En el proceso de inscripción, el estudiante revisa la planificación docente y en conjunto a su profesor guía planifica y organiza las materias a inscribir. En caso de financiamiento propio, cancela por caja el arancel respectivo que es variable debido a que depende de las Unidades Tributarias del municipio o en su defecto si es por terceros, solicita una carta de financiamiento (CDCH, beca, convenio). En el control de estudio del postgrado, entrega el recibo de pago o carta de financiamiento según sea el caso, y la planilla con la firma del profesor guía y la firma del profesor tutor si es requerido. En caso de ser estudiante regular se chequea el expediente para verificar que no tenga deudas. De poseer deuda se retiene la planilla, se crea un oficio indicando que el estudiante presenta deuda, y se cancela la inscripción. Se registra los datos de la planilla en sistema. Se le informa al estudiante las materias abiertas en la programación docente del período a comenzar. El mismo menciona cuales materias desea a inscribir y se procede con el registro de esta. Para inscripción de materias de otros postgrado (ampliación) se registra y se envía un oficio al postgrado correspondiente. Se le entrega una constancia de inscripción indicando las asignaturas inscritas. Se envía un oficio y todos los originales del estudiante inscrito a la Coordinación de Postgrado.

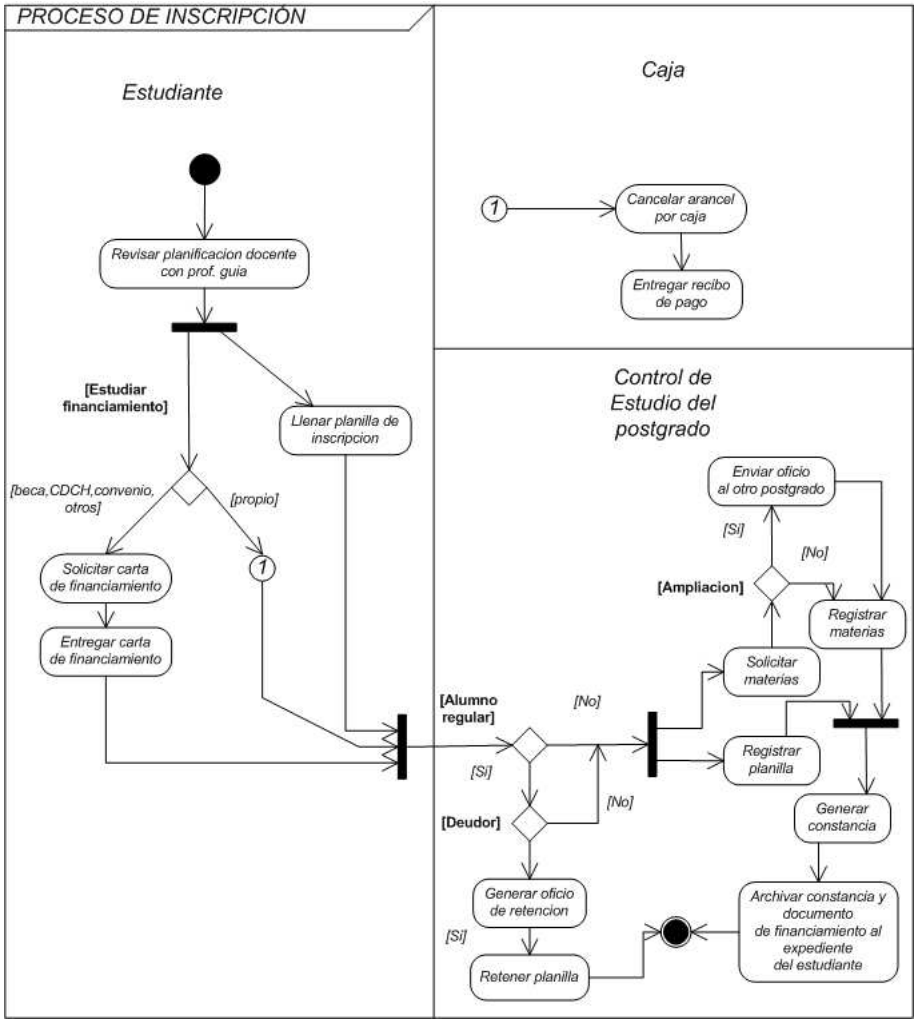

Figura 1.5: Diagrama de actividades del proceso de inscripción

Proceso de Calificación.

En el proceso de calificación, el control de estudio del postgrado, se genera las planillas de notas a mitad de semestre y se reparte a cada profesor. El profesor califica los estudiantes y envía planilla al control de estudio, en donde se carga las notas en el expediente curricular de notas de cada uno y se anexa constancia en su expediente.

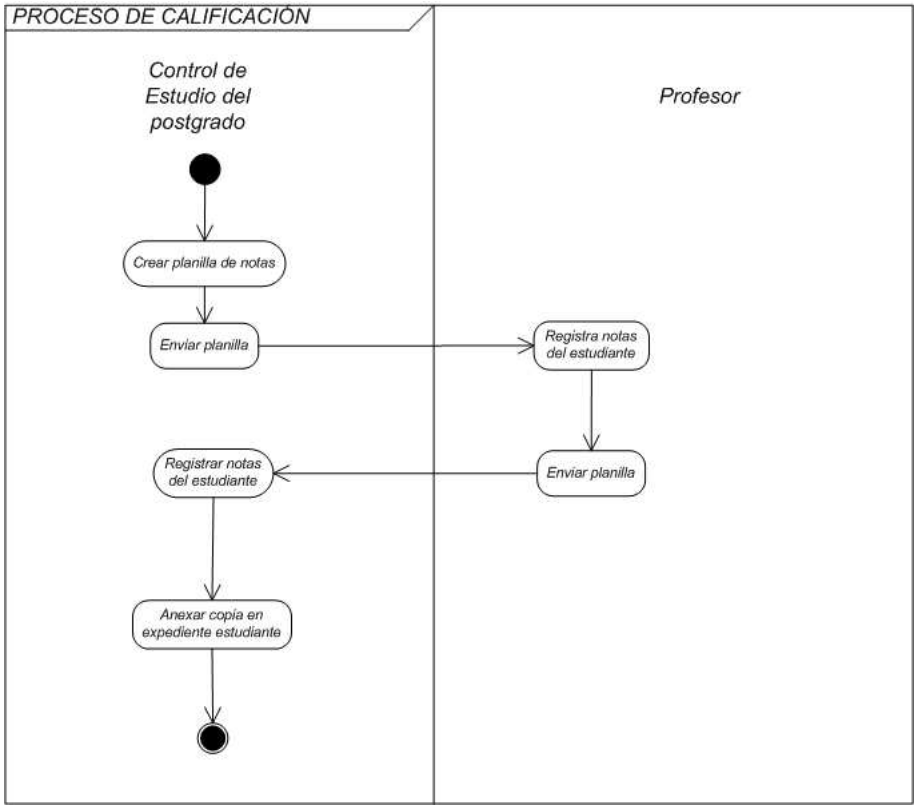

Figura 1.6: Diagrama de actividades del proceso de calificación

Proceso de Retiro de Materia.

En el proceso de retiro de una materia, el estudiante realiza una solicitud indicando la materia a retirar, entrega la carta a la comisión académica, la cual la estudia y se procede a retirar la materia, se crea un oficio el cual se envía una copia a la Coordinación de Postgrado y otra se anexa al expediente del estudiante.

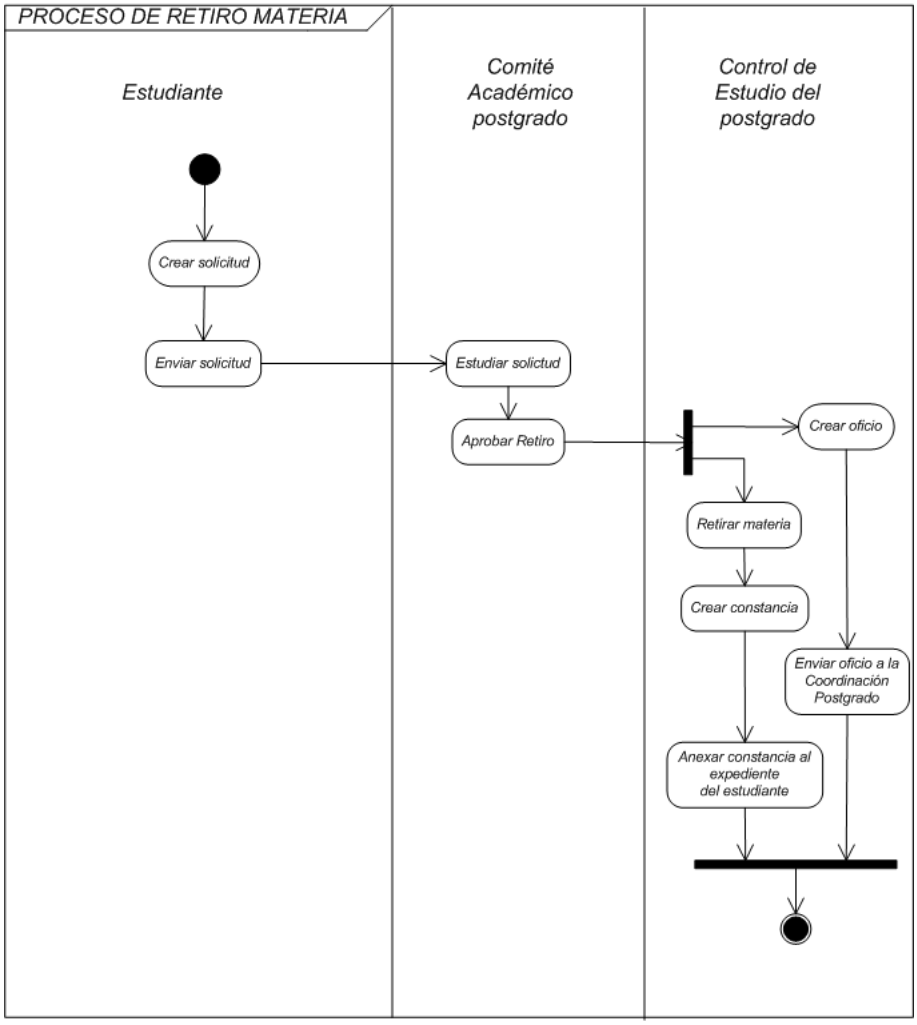

Figura 1.7: Diagrama de actividades del proceso de retiro de materia

Proceso de Desincorporación.

En el proceso de desincorporación, el estudiante realiza un oficio indicando los motivos, avalado por su tutor o profesor guía entrega el oficio a la comisión académica en donde se evalúa la solicitud, de aprobar la desincorporación se cambia el estado del estudiante y se crea un oficio enviándolo a la Coordinación de Postgrado. Este tipo de desincorporación generalmente es voluntaria por el estudiante, se describe como Tipo A.

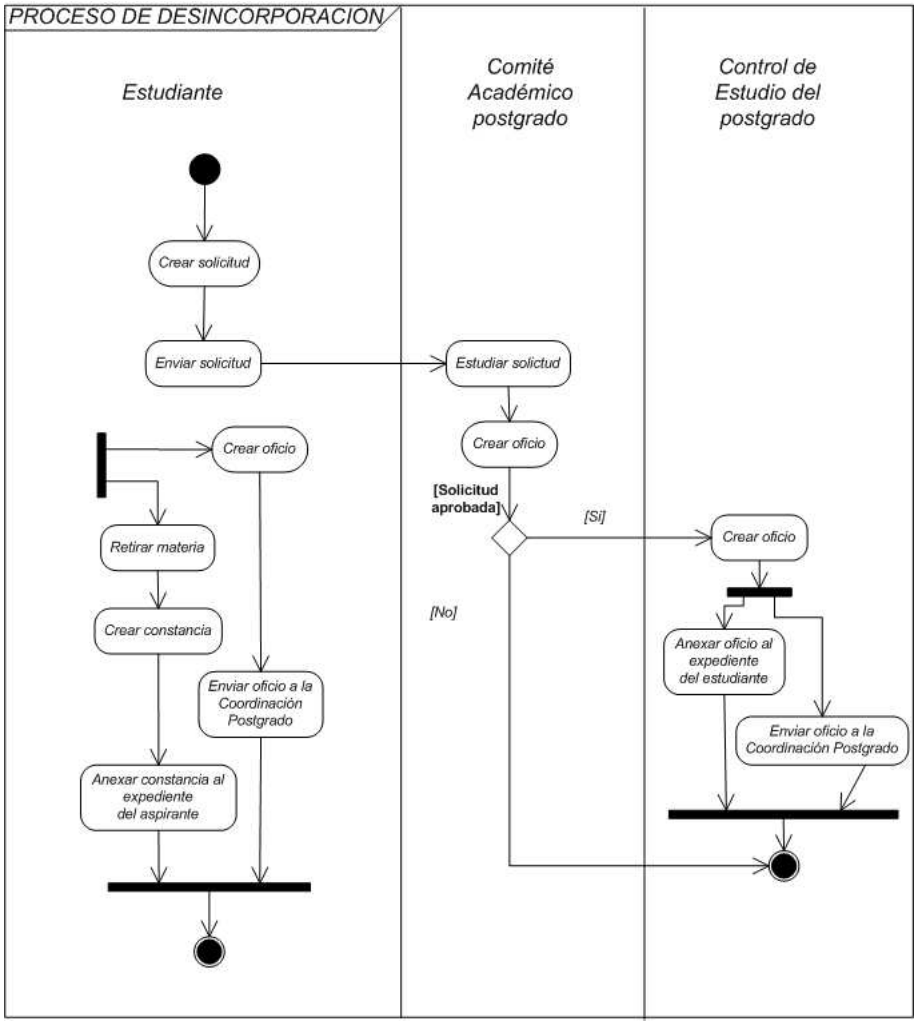

Figura 1.8: Diagrama de actividades del proceso de desincorporación

Proceso de Solicitud de Constancias.

En el proceso de solicitud de constancias, el estudiante cancela un arancel por la constancia en caja, que es variable debido a que depende de las Unidades Tributarias del municipio, entrega el recibo al control de estudio en donde se redacta la constancia, cuyo formato esta predefinido en un documento digital y se entrega al estudiante.

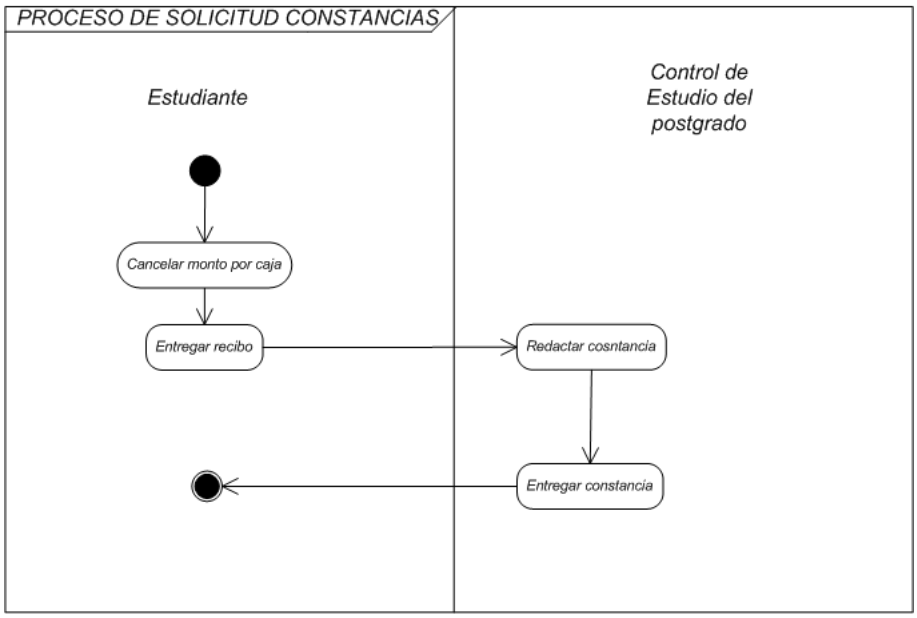

Figura 1.9: Diagrama de actividades del proceso solicitud de constancias

Proceso de Programación Docente.

En el proceso de programación docente, el profesor que desee proponer una materia envía un comunicado dirigido al comité académico, este propone una planificación docente por ejecutar, la cual envía a la coordinación de estudios de postgrados. Este estudia dicha programación, crea un oficio con las revisiones y lo envía al comité donde se hacen los ajustes descritos en las revisiones como programación docente ejecutada, es enviado una copia a la coordinación donde se verifica, archiva y finalmente se publica.

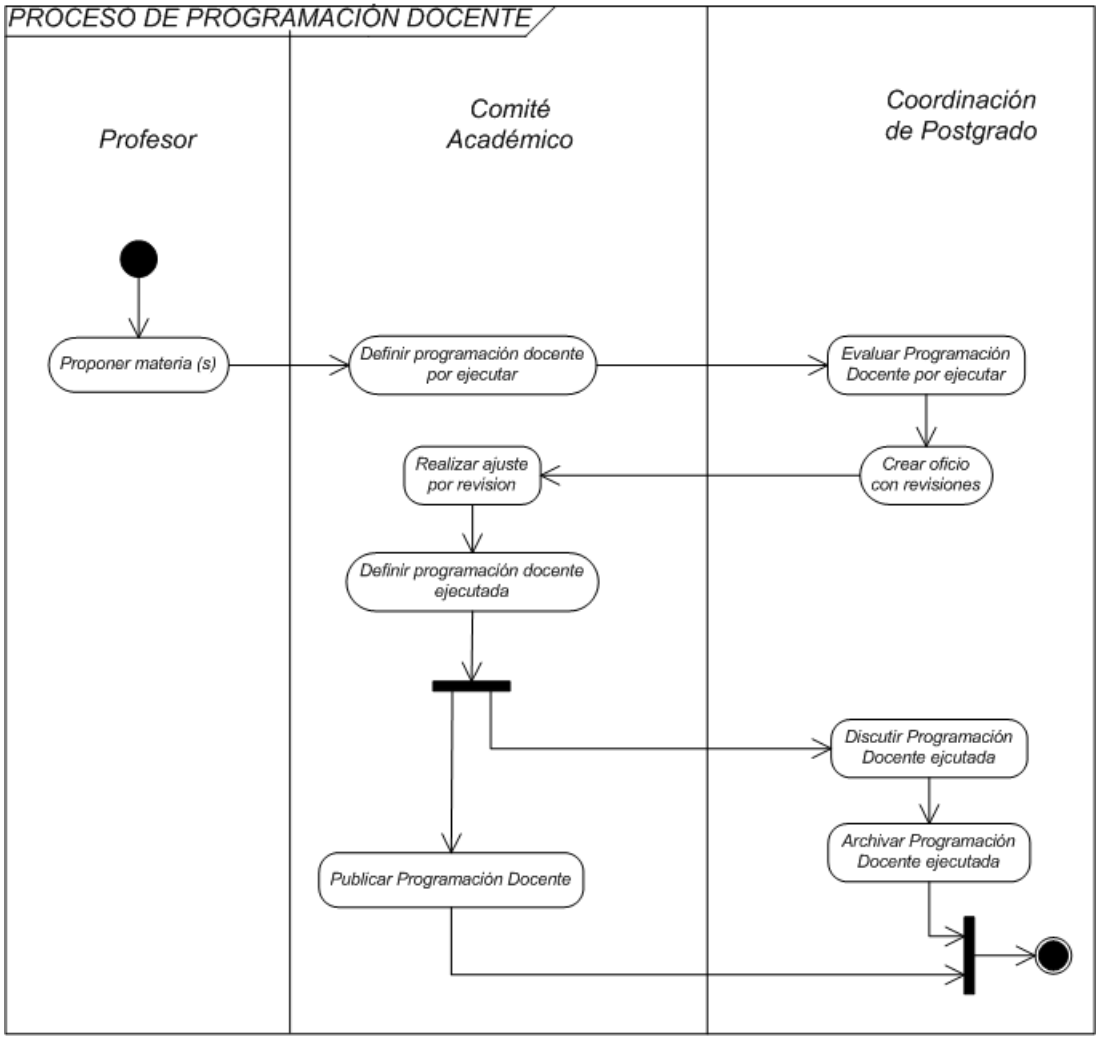

Figura 1.10: Diagrama de actividades del proceso de programación docente

1.2. CONEST

Está parte del capítulo se describe la experiencia del desarrollo y uso del Sistema de Control de Estudios CONEST[1] a nivel general, su estructura, sus módulos, las tecnologías implementadas y la situación actual de dicha aplicación.

1.2.1. Introducción a CONEST [2]

Conest es el Sistema de Gestión Académica de la División de Control de Estudios de la Facultad de Ciencias, cuyo objetivo principal es automatizar los procesos administrativos de la gestión académica. Se encuentra operativo desde el año 2006 con la participación de estudiantes, docentes y personal administrativo de esta comunidad. Es un desarrollo con tecnologías web software libre y como repositorio de datos MySql, haciéndolo modificable y adaptable a las necesidades del usuario. En la Figura 1.11 es posible apreciar la interfaz del menú principal del módulo que presta servicio a los estudiantes. El acceso a esta aplicación es a través de cualquier navegador web.

CONEST administra, gestiona y resguarda los datos de aproximadamente 3600 estudiantes de las licenciaturas de Biología, Computación, Física, Geoquímica, Matemática y Química. Es un sistema orientado a servicios de usuarios y cuenta con un modelo de datos de 5 millones de registros, aproximadamente 120 tablas que contienen la información de más de 37 mil estudiantes que cursan y han cursado estudios en la Facultad de Ciencias. En este desarrollo han participado más de 60 personas (Ver Figura 1.12) las cuales a través de diferentes formas de trabajo como pasantías, seminarios, trabajos especiales de grado, laboratorios prácticos, materias electivas y labor diaria del personal administrativo, han generado como resultado no sólo un software sino la participación activa de diferentes grupos de esta comunidad (docentes, estudiantes, autoridades, personal administrativo). Se han realizado actividades de investigación docente, que aportan experiencias reales a los alumnos que participan en el proyecto, a través del aprendizaje situado.

Actualmente surgen muchos términos y conceptos alrededor de la enseñanza académica, señalando claramente que para garantizar una formación integral no es suficiente la enseñanza tradicional, donde la transferencia de conocimiento tiene lugar en el aula de clase y el profesor es el gran generador de conocimiento. El hecho de involucrar tanto al estudiante como al docente en la resolución del problema de la gestión académica ha enriquecido el proceso de enseñanza aprendizaje, para ambas partes, lo que se evidencia en la alta motivación e incorporación de nuevos grupos al desarrollo.

[1]No hay documentación de CONEST ofical, sin embargo se hizo referencia de las tesis de grado relacionadas

Figura 1.11: Interfaz del Meú Princiapal de Estudiantes

Esta experiencia permitió insertar a los estudiantes en un contexto social, donde existen problemas que ellos, con las herramientas adquiridas a lo largo de su carrera, son capaces de resolver, les permite vivir una experiencia en el trato con usuarios, enfrentar y manejar los conflictos del sistema en producción en condiciones reales, y su participación en grupos de trabajo numerosos, generándoles responsabilidad y compromiso en la entrega de sus asignaciones (Leguizamo & Montaño, 2004).

Para el personal administrativo de la División de Control de Estudios ha sido una experiencia que aporta motivación que mejora la calidad y tiempo de respuestas de su trabajo, a los usuarios que requieran de su atención, como consecuencia se produjo además de un acercamiento importante a la nuevas tecnologías de información y comunicación, la redefinición de los procedimientos mejorando los servicios de la organización.

Esta aplicación se encuentra en constante crecimiento y evolución, ya que se mantiene los desarrollos de nuevas funcionalidades y mejoras en las existentes.

El enfoque de CONEST ha sido el renovar los servicios que ofrece Control de Estudio. Algunas características de CONEST son:

- Implementado en Ruby on Rails bajo la arquitectura Modelo Vista Controlador

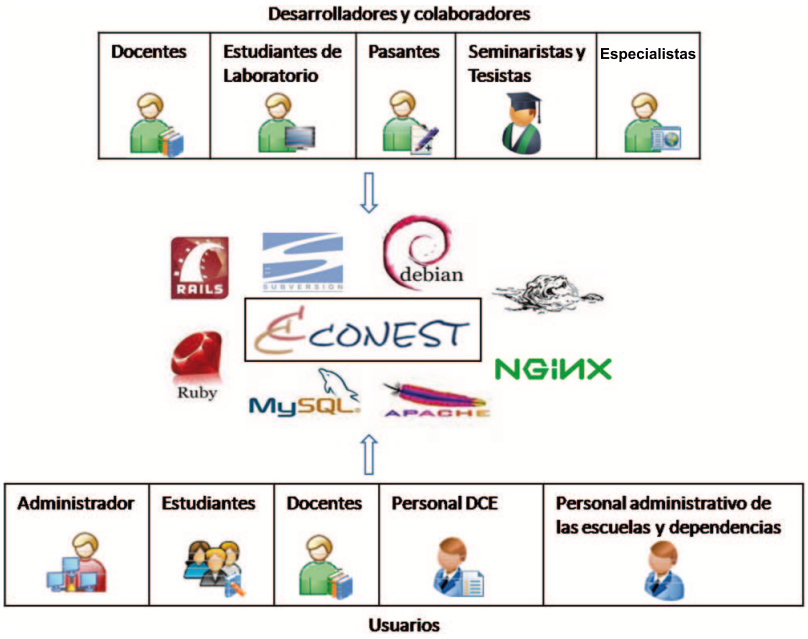

Figura 1.12: Ambiente y usuarios de CONEST

(MVC), permitiendo una buena organización del trabajo debido a la independencia de los componentes que la conforman y mucha flexibilidad en el proceso de realización de cambios.

- Manejado con la herramienta Subversión (SVN) que permite a los desarrolladores establecer sincronización de versiones del código y que trabajen en ambientes colaborativos donde pueda aprender del trabajo realizado por otros integrantes del grupo.

- Posee una interfaz de usuario que es común para todos los módulos en cuanto a la diagramación, colores, banners, fotos, estilo, etc., cumpliendo con los principios, lineamientos y estándares definidos con anterioridad y a los cuales deben adaptarse los desarrolladores.

- Tiene establecida una estructura y organización para su fácil comprensión como por ejemplo estándares en la base de datos (nombre de tablas, atributos, etc.), haciendo que el mantenimiento del sistema sea sencillo.

- Sus funcionalidades pueden ser accedida de forma remota desde cualquier navegador Web las 24 horas del día, los 365 días del año. Entre sus funcionalidades no solo está los procesos de inscripción, grado y calificación, sino también ver el estado de dichos procesos y generar reportes relacionados en distintos formatos, además del envío de información a estudiantes y docentes a través de sus correos electrónicos como por ejemplo la fecha de inicio y fin del proceso de inscripción.

- Incentiva a la participación de forma activa de Estudiantes, Docentes y personal Administrativo que forman parte de la comunidad de la Facultad de Ciencias.

1.2.2. Módulos de CONEST

La aplicación CONEST cuenta con un conjunto de módulos cada uno independiente del otro que ofrecen sus servicios según los roles que cumplan los usuarios en el sistema de pregrado de la Facultad de Ciencias. A continuación se describen los módulos que se encuentran operativos y que han ayudado a satisfacer los compromisos administrativos de la DCE.

- **Módulo servicio al estudiante:** Su primera versión comenzó en el semestre 01-07 y presta servicio a todos los estudiantes que conforman la matrícula de la Facultad de Ciencias, las funcionalidades que tiene son:

  1. La inscripción de materias en un período académico tomando en cuenta las restricciones en cuanto a los requisitos para cursar las materias, además de restringir a los estudiantes incursos en el reglamento de permanencia y a aquellos con deudas antes dependencias internas como Servicio de Orientación, Biblioteca Alonzo Gamero, Bolsa de Libro y Departamento de Administración.
  2. La consulta del horario del semestre en curso, del expediente curricular y la constancia de inscripción.

- **Módulo servicio al Docente:** Comenzó como una prueba realizada con las escuelas de Biología, Computación, Física y Matemáticas donde se realizó por primera vez el proceso de calificación vía Web, lo que ayudó a recoger y centralizar la información de los docentes, número de cédula, correo electrónico y nombre el cual no se disponia de forma digital ni globalizada, a nivel funcional se desarrollo manteniendo el enfoque anterior recopilar notas. Por otro lado las escuelas faltantes (Química y Geoquímica) para la fecha de realización de mencionado proceso no se disponía de la información básica de la planta docente. Este módulo además de la opción de calificar, permite al docente descargar en formato Excel y OpenOffice los listados de estudiantes inscritos en las materias de las licenciaturas correspondientes, al igual que consultar el expediente curricular de dichos estudiantes. Este módulo además contempla otras funcionalidades diferenciadas según los roles que pueda tener un docente como:

- Director(a) de Escuela.
- Jefe de Departamento quien tiene acceso a funcionalidades como programación docente ejecutada, estado de las inscripciones y calificación.
- Unidad de Asesoramiento Académico quienes se encargan de inscribir aquellos estudiantes incursos en el art. 3 de las normas de permanencia y aquellos que se reincorporan.
- Coordinador, entre otros.

- **Módulo de Ingreso:** Fue puesto en producción el semestre 02-07 y este gestiona el proceso de inscripción de los nuevos estudiantes en la Facultad de Ciencias. Ellos a través de un formulario introducen los datos personales y académicos necesarios para formalizar la inscripción, finalizado esto, el nuevo estudiante es inscrito en el primer semestre de su licenciatura.

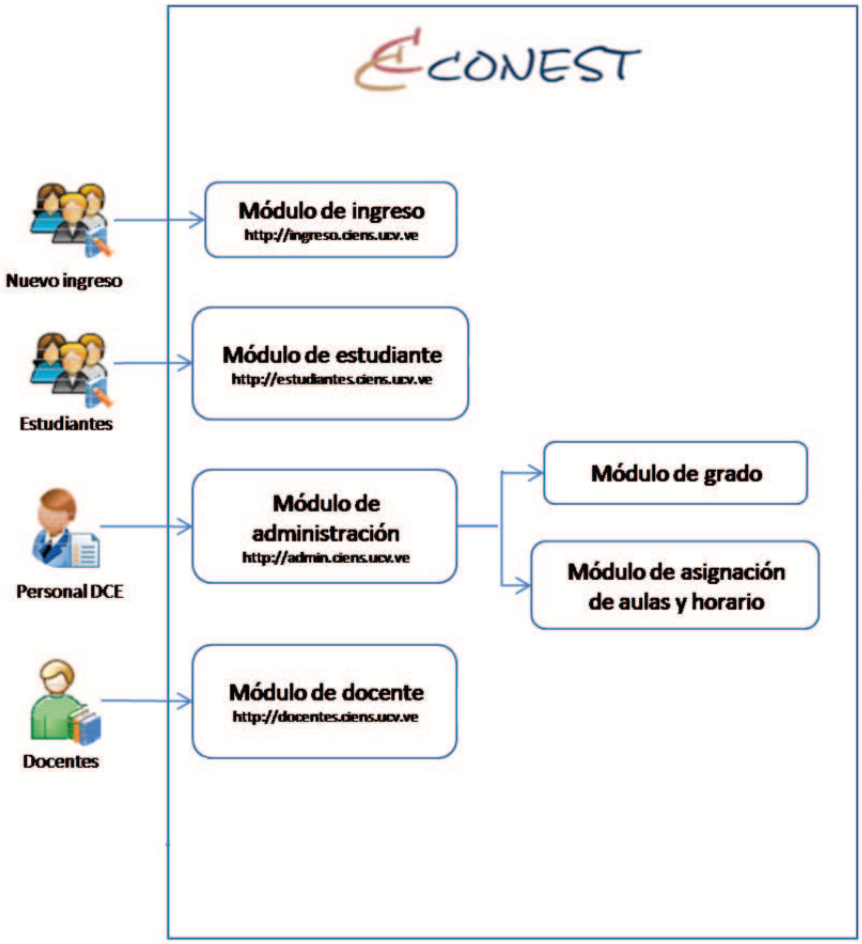

Figura 1.13: Estructura general de CONEST

- **Módulo de Administración:** Este módulo es utilizado por el personal administrativo de la DCE, consolidando todas las funciones de las instancias que han sido desarrolladas, es decir, se integra la administración de todos los módulos. Se encuentra en constante actualización y ofrece funcionalidades que dependiendo del rol que cumpla el personal se despliegan algunas u otras, estás se encuentran organizadas en tres grupos, los cuales contienen procesos relacionados a la asignación y optimización de las planta física de la facultad para impartir clase de pregrado, los relacionados con los actos de grado y los relacionados con actividades administrativas en general. A continuación la descripción de cada grupo:

- **Módulo Admin:** Este módulo posee funcionalidades como consultar, insertar, eliminar y modificar datos de estudiantes, docentes y materias, aplicar el reglamento de permanencia a los estudiantes incursos en determinados artículos, modificar y verificar la calificación de los estudiantes, enviar correos de manera masiva, crear, consultar y eliminar horarios y secciones de una materia, entre otras. Además permite generar reportes tanto individuales como por lotes. Los reportes individuales son aquellos que presentan los datos de un estudiantes en particular y su generación se realiza a través del ingreso de la cédula de identidad, entre los ejemplos de este tipo de reportes se tienen el expediente curricular y diferentes constancias como la de estudios, de notas aprobadas, de inscripción, de horarios, etc. El reporte en lote mas usado es la emisión de planillas de acta final de notas, es un reporte de todas las materias por sección con las calificaciones (incluso retiro) de los alumnos en un período, esta siendo el soporte fisico del cual se genera 3 copias, una destinado a la Dirección de Control de Estudio, la escuela involucrada y la secretaria central.

- **Módulo de Asignación de Horarios y Aulas:** Básicamente se encarga de la asignación de horarios de las materias ofertadas en la programación docente en el período actual, evitando en lo posible el solapamiento de los mismos, luego la asignación de aulas dependiendo de su capacidad y disponibilidad. Otras funcionalidades importantes además de asignar son las de consultar y modificar horario, tanto a los estudiantes como a las materias, y otros reportes.

- **Módulo de Grado:** Se encarga de la gestión administrativa de los graduandos de pregrado que egresan de la Facultad de Ciencias. Entre los servicios se tienen emitir planillas de seminarios y tesis, listados de graduandos, verificación de requisitos de los estudiantes inscritos en TEG (Trabajo Especial de Grado), premios especiales de graduación, estadísticas de graduandos, fotos de los graduandos, generación de CD de la promoción, cierre de expediente, reportes y listados, entre otros. Los actos académicos se efectúan dos veces al año, en los meses de julio y diciembre.

1.2.3. Tecnologías de CONEST

Para el desarrollo de CONEST se pensó en utilizar herramientas y tecnologías con licenciamiento libre. Entre los lenguajes en el que ha sido desarrollado está XHTML, CSS, JavaScript, XML, y Ruby apoyado por el framework Rails. Tiene también el servidor SMTP Exim para el envío de correos, MySQL como sistema manejador de base de datos y SVN como manejador de versiones. CONEST funciona sobre el sistema operativo Linux Debian.

Las aplicaciones Web han sido desarrolladas con el framework de Rails, se realiza la conexión a la base de datos MySQL a través de su componente ActiveRecord. Adjuntamente Mongrel será el encargado de atender las peticiones HTTP que lleguen del servidor web Apache, esto se da a través de Nginx quien redirige y balancea la carga hacia los Mongrels, cada Mongrel ejecuta la aplicación Rails en el intérprete de Ruby. En Ruby es requerido unos componentes adicionales para ser utilizados en ciertos módulos, como por ejemplo en la generación de reportes.

En la Figura 1.14 se puede apreciar la disposición de cada una de las herramientas que conforman la plataforma CONEST, las cuales están ordenadas según su niveles de instalación y acceso, comenzando por el sistema operativo Linux sobre el cual se instala dichas herramientas que logran el funcionamiento de la aplicación.

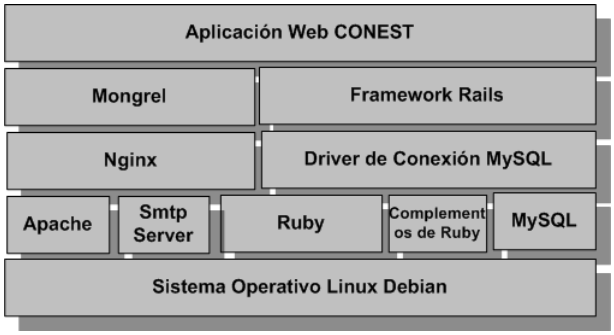

Figura 1.14: Plataforma de CONEST

Para la ejecución de la aplicación es suficiente que el usuario use cualquier navegador Web y haga la petición y conexión al servidor de CONEST para visualizar las páginas y acceder a las distintas funcionalidades. Las peticiones viajan por Internet y son recibidas por el servidor Web Apache.

1.2.4. Estado actual de CONEST

Actualmente el desarrollo del sistema CONEST apunta hacia la creación de una nueva versión llamada CONEST 2, la cual abarca la reingeniería de los procesos internos de la aplicación y donde se plantea en la actualización de algunas herramientas de este sistema a versiones más actuales, contemplando una serie de funcionalidades adicionales y disminuyendo la cantidad de métodos redundantes. También se rediseña y ajusta la estructura de la base de datos, mejorando y creando relaciones y campos necesarios, y se están estableciendo nuevos estándares en las interfaces, en la programación y en la base de datos, todo esto debido a ciertos aspectos donde destacan:

- Redundancia e inconsistencia de datos en la base de datos, comprometiendo la seguridad y fidelidad de la información que se maneja.
- No existía una estandarización en cuanto a los aspectos relacionados a la programación e interfaz.
- Como consecuencia de las irregularidades presentadas en el modelo y estructura de la base de datos, comprometía la eficiencia de las funcionalidades y la codificación de los diversos módulos que lo componen.
- El objetivo o la idea base al desarrollar este proyecto, en la actualidad, no permite la adaptación a las nuevas necesidades y requerimientos suscitados durante el crecimiento del mismo.

En base a la problemática o debilidades que presenta actualemente el sistema CONEST, nació la necesidad de su nueva versión, enfocando el desarrollo a la estandarización y modularidad. Con esta iniciativa, experiencias y por la problemática presente en cuanto a la gestión académica presente en la Coordinación de Postgrado, surge la idea del desarrollo de esta investigación y por ende el desarrollo de una aplicación correlacionada con el nuevo sistema CONEST.

1.3. Resumen del capítulo

En este capítulo, se presentó las bases conceptuales para el desarrollo de este Trabajo Especial de Grado. En el mismo se hizo referencia a la Coordinación de Postgrado de la Facultad de Ciencias, ente u organización bajo el cual se centra el desarrollo de la presente investigación, describiendo su perfil, estructura organizacional y los procesos académicos que son efectuados y coordinados en la mencionada organización. Por otro lado se hizo referencia al sistema CONEST de la División de Control de Estudios de

la Facultad de Ciencias, en donde se exponen los distintos módulos que lo componen, plataforma tecnológica y funcionamiento, así como también su estado actual referente a los cambios y procesos de reingeniería que se le esta aplicando.

Capítulo 2

Marco Metodológico

En este capítulo se presenta la metodología utilizada para el desarrollo del sistema, la cual fue una adaptación de Programación Extrema (XP), metodología ágil en donde los usuarios e interacciones son tan importantes como los procesos y herramientas, y que el funcionamiento del software sea más importante que una documentación exhaustiva.

En el marco metodológico de este Trabajo Especial de Grado se presenta la descripción del proceso de desarrollo a seguir y su adaptación, seguidamente las etapas del proceso, actividades, resultados y estrategias involucradas en el desarrollo de la aplicación.

2.1. Adaptación del Proceso de Desarrollo Programación Extrema

En esta oportunidad, el proceso de desarrollo guía fue *Programación Extrema* (XP), que es metodología de desarrollo ligera basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas [6]. Así mismo, en conjunto a la Modelacion ágil, permitió la realización de manera rápida, sencilla, eficiente y documentada del presente trabajo de investigación. A continuación se describe como se aplican las 4 fases de la metodología ágil en la Programación Extrema para el desarrollo del sistema [8].

- **Planificación:**
Como primer paso fue la de realizar el análisis global del sistema a desarrollar, creando una representación sencilla de las partes que la conformarían y cómo se comunicarían entre ellas. Esto sigue la idea de XP de crear una metáfora del sistema que represente de forma general cuál es el resultado que se persigue en el desarrollo.

Se estableció una bitácora de desarrollo para planificar, en base a días o semanas, todos los requerimientos obtenidos y convertirlos en tareas, así mismo considerar un conjunto de estas tareas para el desarrollo de los módulos del sistema. Esta bitácora consta de una fecha inicio y una fecha fin tanto por módulo como por tarea, la precedencia de la tarea, el porcentaje de culminación, días estimados y días realizados. El formato tomado en consideración se observa en el cuadro 2.1.

No.	Tarea	Precede	Fecha Inicio	Fecha Fin	Días Estimados	Días Realizados
–	–	–	–	–	–	–

Cuadro 2.1: Formato seleccionado para la Bitácora de desarrollo

Además se creó una lista de historias de usuario durante la conversación inicial con el cliente, rol desempeñado por el tutor a cargo del desarrollo de esta investigación, quien fue responsable de observar y añadir todo tipo de modificación para su implementación siendo una prioridad determinada para el aporte de la historia de usuario a la funcionalidad principal del sistema: manejo de información referente a los procesos académicos llevados a cabo en los postgrados de la facultad de ciencias.

Se decidió trabajar en función del tiempo y así, cada historia de usuario tendría una cantidad de días estipulados (no mayor a 1 semana) para realizar actividades que en conjunto dieran como resultado la implementación de la funcionalidad descrita en la historia en cuestión. El formato de presentación de las historias de usuario se observa en el cuadro 2.2:

Número: –	**Nombre:** –
Prioridad: –	**Estimación:** –
Decripción: –	

Cuadro 2.2: Formato seleccionado para manejar Historias de Usuario

Se realizó la planificación de iteraciones en las cuales se dividiría la implementación de las historias de usuario definidas, cada una con una duración de entre 2 a 3 semanas; y las entregas al cliente para obseravaciones, correcciones, cambios y realización de pruebas que se llevarian a cabo al término de cada iteración.

- **Diseño:**
Para seguir la práctica de modelación agil en cada iteración de la etapa de diseño se establecieron la agregación de diagramas o modelos que se tomaron como necesarios para el fácil entendimiento y documentación del sistema. En tal caso de que hayan

sido creados en alguna iteración anterior, entonces, se procedió a actualizarlos y así desechando las versiones anteriores.

- **Codificación:**
Antes de comenzar a implementar se acordó entre los dos programadores un estandar sencillo para codificar en lenguaje ruby y el modo de implementarlo, tomando en cuenta con otra serie de estandares propuestos por el grupo de desarrollo de CONEST:

- Como primer punto a destacar fue, desarrollar de manera modular el sistema, para que cada programador pueda ubicarse y entender el código programado y así ser más rápido a la hora de hacer algun cambio necesario. En esta etapa también se propuso documentar cada método implementado para un fácil entendimiento y a aquellos métodos que fueron creados llamarlos de una forma sencilla y legible.
- Con respecto a la nomenclatura y variables, se optó por desarollo en español. Para variables con dos o mas palabras deben estar separadas por _. Si la palabra tiene $\tilde{n}$ se sustituye por n, si tiene acentos se omite, siempre en minúscula.
- Con respecto a los métodos, se definen siempre en minúscula, si son varios nombres, estaran separados por separados por _, si el método retorna un lógico (boolean) debe terminar en `?`, si es un método que modifica el objeto debe terminar en `!` y si este método muestra una excepción al error, éste debe terminar en `!`.
- Para la base de datos, las tablas tipo o tablas para clasificar algo, los nombres deben estar precedidos por el prefijo *tipo_*, los *id* son preferiblemente char(10) y no son auto-numéricos, las tablas tipo tienen un campo descripción varchar(255), los campos String son varchar(255) a menos que sean largos los cuales son longtext y se aceptaron claves compuestas, además, se declaran siempre en minúscula y en singular. Para las relaciones se mantiene las convenciones de Rails, *nombre_tabla_nombre_campo*.

Para la etapa de codificación de cada iteración se asignaban diferentes tareas por historias de usuario a cada programador, al finalizar la tarea se registraban en el manejador de versiones.

- **Pruebas:**
En esta etapa de la modelación agil, las pruebas se realizan tras la culminación de un método del módulo en el que se estaba trabajando, para validar la funcionalidad del mismo.

Luego de la culminación del desarrollo de cada módulo planteado, se contemplaron pruebas unitarias escribiendo los casos de prueba y aquellos casos que no se tomaron en cuenta se anotaban para una prueba final de dicho módulo.

No. Caso Prueba	Módulo	Descripción del Caso de Prueba	Resultado Esperado	Resultado Obtenido	Motivo de la Falla
–	–	–	–	–	–

Cuadro 2.3: Formato de registro de pruebas del lado del cliente

2.2. Análisis Global del Sistema

Este analisis inicial proviene desde el levantamiento de información que fue realizada al inicio del documento, que conlleva a todos los requerimientos necesarios para el desarrollo de la aplicación. A partir de esto se obtienen las historias de usuario desde la primera reunión con el cliente y se construye un modelo del esquema de funcionamiento del sistema. Los resultados se presentan a continuación:

2.2.1. Historias de Usuario

Número: 1	**Nombre:** Desarrollar método `guardar` en el modelo Aspirante
Prioridad: Alto	**Estimación:** 10 días
Descripción: Guardar la información referente a un aspirante, así como también establecer las relaciones con modelos asociados, (Admisión, Pago, Transacción).	

Número: 2	**Nombre:** Desarrollar interfaz de usuario para modelo Aspirante
Prioridad: Medio	**Estimación:** 2 días
Descripción: Desarrollar interfaz de usuario, donde se solicite la información necesaria para poder asociar un aspirante a un postgrado, y así crear una admisión. En dicho formulario, se requiere la cancelación de un arancel por pre-inscripción.	

Número: 3	**Nombre:** Desarrollar validaciones para formulario del modelo Aspirante
Prioridad: Bajo	**Estimación:** 1 días
Descripción: Validar de la información ingresada por el aspirante y proceder con su postulación al postgrado.	

Número: 4	**Nombre:** Generar archivo PDF y comprimido de preinscripción
Prioridad: Medio	**Estimación:** 5 días
Descripción: Generar, de acuerdo a la información del formulario registrada por el aspirante, un archivo PDF con dicha información y ademas un archivo comprimido del material de preinscripción.	

Número: 5	**Nombre:** Desarrollar modelo que permite asociar documentos a un aspirante
Prioridad: Medio	**Estimación:** 3 días
Descripción: Asociar a un aspirante, el conjunto de documentos necesarios como parte de sus requisitos.	

Número: 6	**Nombre:** Desarrollar componente Javascript para listar usuarios y materias
Prioridad: Alto	**Estimación:** 3 días
Descripción: Desarrollar un componente Javascript, tipo agenda, que permita listar los usuarios (aspirantes, estudiantes, docentes) y materias de acuerdo a varios estatus.	

Número: 7	**Nombre:** Desarrollar vista para módulo Aspirante
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Desarrollar una vista donde se muestre la información relevante de un aspirante seleccionado y poder asociar el conjunto de documentos.	

Número: 8	**Nombre:** Desarrollar componente ajax para módulo Aspirante
Prioridad: Medio	**Estimación:** 1 día
Descripción: Desarrollar un componente ajax, que permita subir imagenes al servidor y asociarlo con un documento.	

Número: 9	**Nombre:** Desarrollar componente javascript de paginación para módulo Aspirante
Prioridad: Medio	**Estimación:** 1 día
Descripción: Desarrollar un componente javascript que permita paginar los documentos del aspirante.	

Número: 10	**Nombre:** Desarrollar modelos para el módulo Solicitudes
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Desarrollar modelos que permita consultar los distintos tipos de solicitudes (constancia, desincorporación, retiro de materia).	

Número: 11	**Nombre:** Establecer relaciones para los modelos del módulo Solicitudes
Prioridad: Bajo	**Estimación:** 1/2 día
Descripción: Establecer relaciones entre el modelo descrito en la historia de usuario 10, con los demás modelos asociados para registrar una solicitud.	

Número: 12	**Nombre:** Desarrollar vistas para el módulo Solicitudes
Prioridad: Bajo	**Estimación:** 2 días
Descripción: Desarrollar vistas donde se visualicen las peticiones solicitadas por el estudiante, actualizando con ajax en la misma página (web 2.0).	

Número: 13	**Nombre:** Desarrollar validaciones para el módulo Solicitudes
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Validar las peticiones hechas por el estudiante y para el caso de retiro de materia, no se podrá el total de estas	

Número: 14	**Nombre:** Desarrollar una adaptación de las vistas del módulo de solicitudes para pago arancel
Prioridad: Bajo	**Estimación:** 1/2 día
Descripción: Realizar modificaciones a las vistas de petición de solicitudes para agregar el pago de arancel.	

Número: 15	**Nombre:** Desarrollar método para el seguimiento de estado de las solicitudes
Prioridad: Bajo	**Estimación:** 1/2 día
Descripción: Incorporar un seguimiento de estado de la solicitud pedida por el estudiante, mostranto los diferentes estados por lo que ha pasado, fecha y por quien fue modificado, ésta informará al estudiante cuando se cambie de estado dicha solicitud en la que amerite.	

Número: 16	**Nombre:** Generar archivo PDF para el módulo de Solicitudes
Prioridad: Bajo	**Estimación:** 1/2 día
Descripción: Generar un archivo PDF al momento de hacer una petición de solicitud de constancia.	

Número: 17	**Nombre:** Notificar vía email a docente por materia asignada
Prioridad: Bajo	**Estimación:** 2 días
Descripción: Enviar via correo electrónico una notificacion a un docente cuando este haya asignado una materia, indicando que la materia se le fue asignada	

Número: 18	**Nombre:** Listar solicitudes ya pedidas por el estudiante
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Mostrar el listado de solicitudes ya pedidas por el estudiante al momento que desea solicitar alguna otra solicitud, con el fin de que pueda observar cuales ha solicitado y cuales no.	

Número: 19	**Nombre:** Visualizar documentación de estudiante con opción a editarla.
Prioridad: Medio	**Estimación:** 2 días
Descripción: Permitir visualizar la documentación asignada o asociada a un aspirante en su proceso de admisión, así mismo poder cambiarla una vez registrada.	

Número: 20	**Nombre:** Listar estudiantes filtrados por estatus
Prioridad: Medio	**Estimación:** 2 días
Descripción: Ofrecer el listado de estudiantes filtrado por tres tipos de estatus (no aceptado, pendiente, aprobado) para el módulo de selección.	

Número: 21	**Nombre:** Generar Reportes
Prioridad: Bajo	**Estimación:** 2 días
Descripción: Generar reportes donde se muestre en detalle, la información referente a las diversas transacciones registradas en el sistema, considerando estatus y motivo.	

Número: 22	**Nombre:** Permitir multiples formatos de respuesta
Prioridad: Bajo	**Estimación:** 3 días
Descripción: Ofrecer diversos formatos de respuesta (xml, json, rss) del servicio web basado en `ActiveResource` para cada una de las consultas o peticones al módulo de caja	

Número: 23	**Nombre:** Generar hoja de cálculo de calificación
Prioridad: Bajo	**Estimación:** 2 días
Descripción: Ofrecer en el módulo de calificación, la opción de obtener un sopote físico de las calificaciones hechas a los estudiantes, sea general o continua.	

Número: 24	**Nombre:** Permitir listar todos los aspirantes
Prioridad: Medio	**Estimación:** 1 días
Descripción: Listar todos los estudiantes, de acuerdo a la organización que pertenece el comité académico evaluador.	

Número: 25	**Nombre:** Visualizar el detalle de solicitud de un estudiante
Prioridad: Medio	**Estimación:** 1/2 día
Descripción: Visualizar el detalle de solicitud, mostrando el estatus de la admisión, docente guía y /o tutor si aplica y datos personales, listado de documentos consignados en la fase de admisión ante la Coordinación de Postgrado.	

Número: 26	**Nombre:** Filtrar lista de aspirantes por estatus
Prioridad: Medio	**Estimación:** 1 días
Descripción: Filtrar la lista por los estatus de la admsión de un aspirante (por evaluar, pendiente, aceptada, no aceptada).	

Número: 27	**Nombre:** Crear un formulario que permita evaluar al aspirante
Prioridad: Alto	**Estimación:** 1 días
Descripción: Permitir evaluará a un aspirante y donde considera la opción de la aceptación o no, un área de texto que permita indicar alguna observación. En el caso de que se acepte al aspirante, asignar el docente tutor y guía, siendo éste último obligatorio.	

Número: 28	**Nombre:** Notificar vía correo electrónico resultado de evaluación
Prioridad: Medio	**Estimación:** 1/2 día
Descripción: Notificar al aspirante del resultado de su evaluación vía correo electrónico.	

Número: 29	**Nombre:** Generar planilla de inscripción de Secretaria Central
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Generar una planilla de inscripción de Secretaria Central personalizada para el aspirante aceptado.	

Número: 30	**Nombre:** Listar materias permitidas para inscribir a un estudiante.
Prioridad: Alto	**Estimación:** 2 días
Descripción: Listar el conjunto de materias posibles para inscripción para un estudiante en particular.	

Número: 31	**Nombre:** Filtrar materias de acuerdo a su tipo.
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Filtrar materias de acuerdo a su tipo (obligatorias, electivas, etc).	

Número: 32	**Nombre:** Solicitar los aranceles por concepto de pago.
Prioridad: Medio	**Estimación:** 4 días
Descripción: Solicitar los aranceles por concepto de pago de matricula o pagos de números de créditos inscritos, si éstos aplica al grado el cual cursa el estudiante.	

Número: 33	**Nombre:** Descargar constancia de inscripción.
Prioridad: Bajo	**Estimación:** 1 días
Descripción: Ofrecer después de haber inscrito el estudiante descarga la constancia de inscripción.	

Número: 34	**Nombre:** Desarrollar método de cancelación de solicitud de constancia.
Prioridad: Alto	**Estimación:** 1 día
Descripción: Permitir seleccionar el el tipo de pago para la cancelación de solicitud de constancia al momento de la petición.	

Número: 35	**Nombre:** Desarrollar método para agregar comentarios a la solicitud.
Prioridad: Medio	**Estimación:** 1 día
Descripción: Agregar comentarios a la solicitud, detallando fecha, y por quién fue realizada.	

Número: 36	**Nombre:** Desarrollar servicio web para consultar transacciones.
Prioridad: Alto	**Estimación:** 1 día
Descripción: Consultar, a través de servicios web, las transacciones efectuadas en la caja y esos formatos de respuestas deben ser soap,xml,json.	

Número: 37	**Nombre:** Desarrollar método para generar reportes.
Prioridad: Medio	**Estimación:** 1 día
Descripción: Generar reportes de acuerdo a los estados y motivos de las transacciones.	

Número: 38	**Nombre:** Desarrollar método para registrar una transacción nueva.
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Registrar una nueva transacción en base al pago de cualquier arancel, producto del pago por algun servicio o solicitud.	

Número: 39	**Nombre:** Desarrollar método listar materias coordinadas por el docente.
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Listar aquellas materias que son dadas o coordinadas por dicho docente, para que este pueda elegir una de ellas y poder calificar.	

Número: 40	**Nombre:** Desarrollar método que permita la calificación continua o definitiva.
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Dar opción para que el docente pueda calificar de manera continua por medio de evaluaciones o de manera definitiva.	

Número: 41	**Nombre:** Desarrollar método que genere listado de estudiantes por materia y por modo de calificación.
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Dar opción para listar estudiantes por materia seleccionada y por modo de calificación (continua o definitiva).	

Número: 42	**Nombre:** Desarrollar método que permita guardar temporalmente o de manera definitiva la calificación (Modo Definitivo).
Prioridad: Medio	**Estimación:** 1 día
Descripción: Dar opción de guardar temporalmente la calificación o guardar definitivamente dicha calificación estando en modo definitiva.	

Número: 43	**Nombre:** Desarrollar método que permita mostrar el acumulado de notas del estudiante cuando se está evaluando de forma continua.
Prioridad: Alto	**Estimación:** 1 día
Descripción: Mostrar el acumulado de notas del estudiantes al momento de realizar una evaluación continua.	

Número: 44	**Nombre:** Desarrollar método que permita notificar al estudiante cuando haya sido calificado y actualizada la nota.
Prioridad: Alto	**Estimación:** 1 día
Descripción: Notificar al estudiante cuando ha sido calificado o actualizada la nota tanto de manera definitiva como continua.	

Número: 45	**Nombre:** Desarrollar método que permita mostrar la nota definitiva o acumulado del estudiante.
Prioridad: Alto	**Estimación:** 1 día
Descripción: Mostrar al docente cuando se está en modo de calificación definitivo, la nota definitiva o el acumulado de ella de un estudiante.	

Número: 46	**Nombre:** Desarrollar método que permita mostrar aquellas materias coordinadas por el docente.
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Mostrar las materias que son coordinadas por un docente, para que al momento de seleccionar pueda asociar tipos de evaluaciones.	

Número: 47	**Nombre:** Desarrollar método que permita asociar los tipos de evaluaciones a materias.
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Asociar a una materia, uno o mas evaluaciones que formarán parte de la evaluación continua de la misma.	

Número: 48	**Nombre:** Desarrollar método que permita definir tema, porcentaje y fecha de evaluación.
Prioridad: Medio	**Estimación:** 1 día
Descripción: Definir tema o temas a evaluar, porcentaje o valor, y fecha de la misma por cada evaluación asociada.	

Número: 49	**Nombre:** Desarrollar método que permita modificar evaluaciones a una materia.
Prioridad: Medio	**Estimación:** 1 día
Descripción: Permitir la modificación de la evaluación que es o son asignadas a una materia en particular.	

Número: 50	**Nombre:** Desarrollar método que permita listar todas las materias asociadas a una organización.
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Listar todas las materias en plan, asociadas a la organización a la cual pertenece el docente.	

Número: 51	**Nombre:** Desarrollar método que permita mostrar el detalle de una materia permitiendo ser ofertada o no.
Prioridad: Medio	**Estimación:** 1 día
Descripción: Mostrar el detalle de la materia elegida y dará opción a que permita ofertar o no la misma.	

Número: 52	**Nombre:** Desarrollar método que permita asociar a un docente formar parte del grupo docente de una materia.
Prioridad: Alto	**Estimación:** 1 día
Descripción: Si una materia se encuentra ofertada por otro docente, se debe indicar al docente actual, pero dará la opción de que él mismo se pueda asociar a dicha materia como parte del grupo docente.	

Número: 53	**Nombre:** Desarrollar método que permita ofertar materias dirigidas.
Prioridad: Medio	**Estimación:** 1 día
Descripción: Permitir a un docente ofertar materias para un grupo de estudiantes, en este caso una materia dirigida.	

Número: 54	**Nombre:** Desarrollar método que permita listar materias por categorías.
Prioridad: Alto	**Estimación:** 1 día
Descripción: Filtrar el listado de las materias en dos categorias, ofertadas y no ofertadas.	

Número: 55	**Nombre:** Desarrollar método que permita listar materias ofertadas por docentes.
Prioridad: Alto	**Estimación:** 1 día
Descripción: Listar las materias ofertadas por los docentes, el cual debe mostrar unicamente aquellas pertenecientes a la organización con la que se inicio la sesión del coordinador de área.	

Número: 56	**Nombre:** Desarrollar método que muestre el detalle de la materia ofertada.
Prioridad: Medio	**Estimación:** 1 día
Descripción: Permitir que al seleccionar una materia, mostrar el detalle de la materia ofertada, asi como también los profesores que la ofertaron.	

Número: 57	**Nombre:** Desarrollar método que permita dar opción de aceptación de la materia para formar parte de la oferta académica del período en curso.
Prioridad: Medio	**Estimación:** 1 día
Descripción: Dar la opción de aceptación de la materia para formar parte de la oferta académica del período en curso, a su vez asignar su coordinador, nombre de la sección a abrir y el número de estudiantes por sección.	

Número: 58	**Nombre:** Desarrollar método notificar al coordinador que es asignado a una materia.
Prioridad: Bajo	**Estimación:** 1 día
Descripción: Notificar al coordinador que es asignado a una materia cuando ésta es aceptada.	

2.2.2. Metáfora

En la Figura 2.1, la metafora del sistema planteado a desarrollar, en donde se destaca el sitio Web, el servidor de aplicaciones, módulo de registro de transacciones (caja api), y los componentes externos que interactuan directa e indirectamente con el sistema, refieriendose a el servidor del Centro de Computación y el sistema de caja de la Faculad de Ciencias respectivamente.

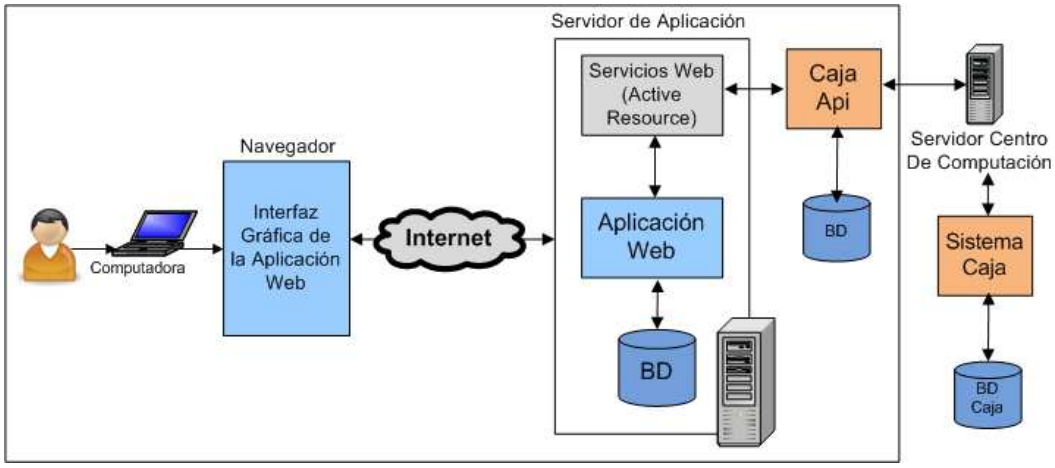

Figura 2.1: Metáfora (Estructura del Sistema)

-Interfaz gráfica de la aplicación Web:
Sitio Web al servicio de los estudiantes de postgrado y personal administrativo de la Coordinación de Postgrado, que se compone de un ambiente de diseño sencillo construida usando tecnología de la Web 2.0. [1]

- Aplicación Web:
Encargado de proveer la eficiencia y eficacia de todas las funcionalidades del sistema, para facilitar todas las peticiones de un estudiante en postgrado, que conlleva desde la postulación de un aspirante al postgrado hasta la notificación de cualquier calificación que amerite, esto con la finalidad de que por medio del sistema pueda gestionar todo tipo de solicitud desde cualquier parte que se encuentre. Esta aplicación Web también tiene un importante desarrollo para la comunidad en el sentido que además de automatizar procesos, disminuye el tiempo de respuesta de dichos procesos como también una comunicación con la caja administrativa de la Facultad de Ciencias para verificar y validar los pagos realizados por los estudiantes.

[1] todas aquellas utilidades y servicios de Internet que se sustentan en una base de datos, la cual puede ser modificada por los usuarios del servicio, ya sea en su contenido (añadiendo, cambiando o borrando información o asociando datos a la información existente), pues bien en la forma de presentarlos, o en contenido y forma simultáneamente.- (Ribes, 2007)

- **Servicio Web (Active Resource):**
Encargado de controlar, verificar y validar diferentes acciones dirijidas hacia la caja de la Facultad de Ciencias con respecto a las solicitudes de constancias realizadas por los estudiantes, quienes deben ingresar un código del ticket de caja a la aplicación Web para poder ser procesada y por medio del Servicio Web hacer la verificación respectiva para obtener así su validéz. También este proceso ayuda a generar todos los reportes necesarios para llevar un registros de los pagos de dichos estudiantes.

- **Caja Api:**
Sistema que se encarga de guardar todos los pagos realizados por la comunidad del postgrado la facultad antes mencionada.

2.2.3. Especificaciones técnicas

El desarrollo del sistema se implantará utilizando las siguientes herramientas tecnológicas:

- Servidor Mongrel de Rails (V-1.1.5)

- Procesamiento de Eventos: JavaScript, Prototype, JQuery y Scriptaculous

- Intercambio de información cliente-servidor: AJAX

- Gestión de usuarios, solicitudes y peticiones: SMBD MySQL (V-5.0)

Una vez establecidos los requerimientos iniciales, seleccionadas las primeras historias de usuario a implementar y haber modelado una idea general de cómo funcionaría el sistema, se planifican 10 iteraciones que se especificarán con mas detalle en el siguiente capítulo.

2.3. Resumen del capítulo

En este capítulo, se presentó una adaptación del proceso de desarrollo de software aplicado a la presente investigación, basado en el enfoque o metodología agil *Programación Extrema* (XP) para la construcción de la aplicación Web. Para tales efectos, se describe el contexto de desarrollo, descripción de cada una de las fases del método de desarrollo utilizado para la automatización de los procesos académicos, definiendose este como todas las activiades relacionadas con los procesos de admisión y selección de aspirates, inscripción de materias, solicitudes estudiantiles y planificación docente, procesos dirigidos en la Coordinación de Postgrado en conjunto a los trece postgrados que lo integran.

Capítulo 3

Marco Aplicativo

3.1. Contexto del desarrollo

El presente proyecto ha sido desarrollado en el contexto del Trabajo Especial de Grado presentando ante la ilustre Universidad Central de Venezuela, Facultad de Ciencias, Escuela de Computación bajo la tutoría del profesor Andres Sanoja y asesoramiento del profesor Sergio Rivas, para optar al título de Licenciado en Computación, por parte de los Bachilleres Gabriel Plaza y Andrés Ramírez.

3.2. Plan de Iteración

Al desarrollar este proyecto bajo el enfoque de desarrollo ágil *Programación Extrema* (XP), es necesario definir el número de iteraciones a desarrollar y para cada una de estas, el lapso o tiempo de desarrollo y fecha de inicio y fecha fin estimada, los casos de pruebas que van ser tomados en cuenta y los criterios de evaluación.

En el desarrollo de este proyecto, cada iteración tendrá una duración de 1 a 2 semamas, donde cada uno de ella sera considera como un módulo funcional sobre el sistema, el punto o fecha de inicio fue a partir del 02 de Abril de 2009 hasta el 02 de Junio de 2009, por lo que este plan de iteración tendrá una duración de aproximadamente 3 meses incluyendo el proceso de documentación.

En cada iteración se van a realizar adaptaciones de las actividades propias del proceso de desarrollo XP, además de incluir las bitácoras de desarrollo día por día.

3.3. Iteración 0

En esta iteración se detalla el planteamiento del problema por el cual se obtuvo la propuesta de solución de forma general, para tener conocimiento teórico de la aplicación que se realizó y con los requerimientos que se obtuvieron en el levantamiento de información. A continuación se explica con detalle cada uno:

- **Planteamiento del problema:** Cada unas de las actividades que se llevan a cabo en los procesos académicos, son en su mayoría manuales siendo el problema principal a considerar. Sumado a esto, por cada flujo de trabajo en las tareas se efectúan un número considerable de pasos repetitivos, lo cual aumenta la posibilidad de incurrir en errores, por consiguiente conllevan a un aumento de los costos en cuanto a recursos físicos y tiempo.

  Otro de los problemas que se presenta, es que en algunas actividades el flujo de trabajo involucra la intervención de diversas unidades, que en cuyo caso debido a la excesiva demanda, conduce a la acumulación de trabajo y por ende retraso en la ejecución de las mismas. Durante la ejecución de estas actividades surge la necesidad de crear informes, oficios, comunicados y/o cualquier otro reporte, lo que representa gastos en material de oficina y tiempo en la elaboración de los mismos.

  La Coordinación de Postgrado la cual es el ente central bajo el cual se basa esta investigación, recibe las diferentes solicitudes, oficios y reportes provenientes de los controles de estudio de los diversos postgrados de la Facultad de Ciencias. Tal documentación es enviada a éste sin un formato estándar, lo que tiene como consecuencia la necesidad de una actividad de transcripción por parte del personal administrativo encargado en dicha coordinación, y por ende un aumento en la carga laboral y el flujo de trabajo. Cierta documentación requiere ser registrada y procesada para de alguna manera llevar datos estadísticos y control para futuros usos dentro de la coordinación.

  En síntesis, se puede evidenciar que como gran parte de estas actividades involucra trabajo manual y más aún, si durante la ejecución de estas, están involucradas otras dependencias donde la comunicación de información es compartida y donde la estandarización juega un papel muy importante, es requerido implantar e implementar una aplicación que automatice y estandarice estas actividades para así facilitar y agilizar los procesos involucrados.

- **Propuesta de solución:** La solución propuesta fue realizar una aplicación Web que utilizando las tecnologías actuales, permita solventar los problemas existentes en la Coordinación de Postgrado en cuanto al manejo de información referente a los procesos académicos y administrativos llevados a cabo en los postgrados de la Facultad de Ciencias. Para tal hecho, se utilizara como base el sistema CONEST,

estableciendo de esta manera un único sistema de gestión de los procesos administrativos de la docencia funcional a nivel de pregrado y postgrado.

A continuación se muestra cada uno de los procesos optimizados de acuerdo al nuevo sistema de Conest para la Coordinación de Postgrado de la Facultad de Ciencias de la Universidad Central de Venezuela:

- **Proceso de admisión:**
  El proceso de admisión inicia con la cancelación del material de pre-inscripción por parte del aspirante. Con el recibo de pago que la caja entrega, el aspirante se registrará en el sistema propuesto, en el cual se le solicitará, datos personales así como también información referente a la carrera por la cual aspira. Dicho sistema generará un comprobante de preinscripción y la opción de descargar el material necesario para formalizar su solicitud. Al momento de consignar estos requisitos ante la Coordinación de Postgrado, el personal administrativos de la mencionada organización, digitalizara los documentos del aspirante en el sistema y manteniendo los originales bajo expediente. Posterior a este hecho se notificará, de manera automática, a la dependecia del postgrado respectivo sobre tal eventualidad.

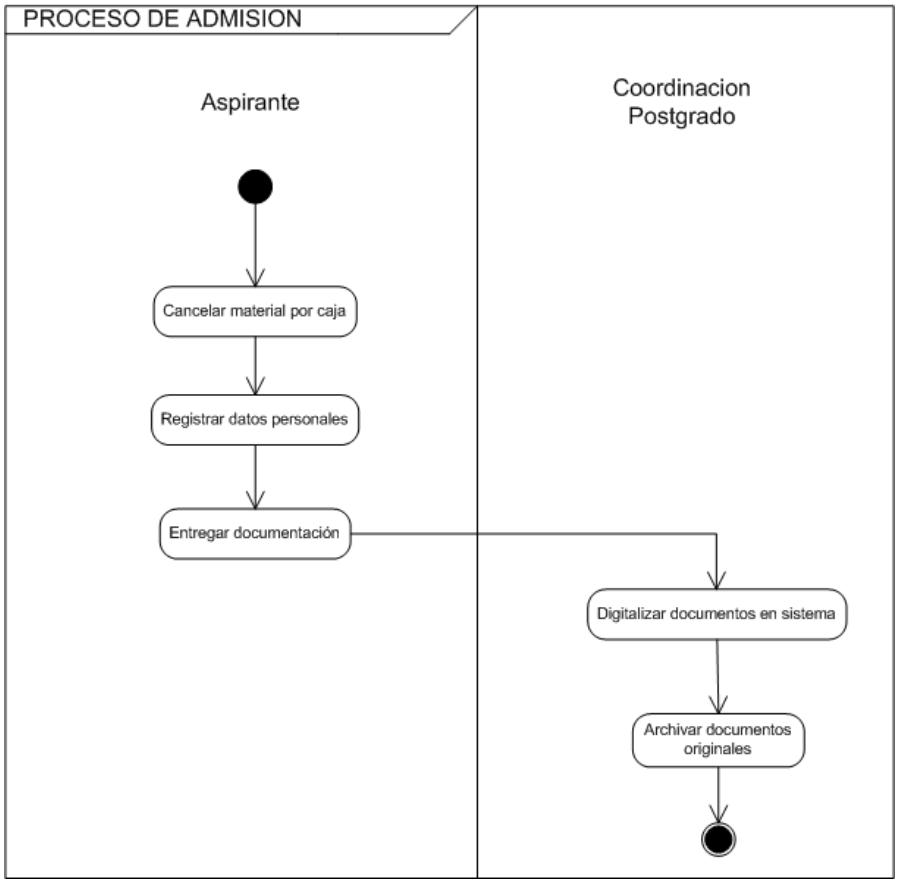

Figura 3.1: Diagrama de actividades de la propuesta del Proceso Admisión.

- **Proceso de selección:**
  Tras el proceso de registro y consignacion de documentos, el comité académico procederá a evaluar a cada aspirante. Para tal efecto, este comité ingresa al sistema y seleccionará del listado el aspirante a evaluar, donde se mostrará el detalle del aspirante y los documentos consignados. Tras la evaluación y entrevista del mismo, el comité podrá emitir alguna observación y la aceptación o no de su admisión, para ambos casos, se notificará los resultados obtenidos más observaciones. Si el aspirante es aceptado, se le asignará el docente guía y de manera opcional el docente tutor, asi mismo y de manera automática, se le asigna una cuenta en el sistema de no poseerla, formarmado así parte de la matrícula estudiantil de acuerdo a la carrera seleccionada.

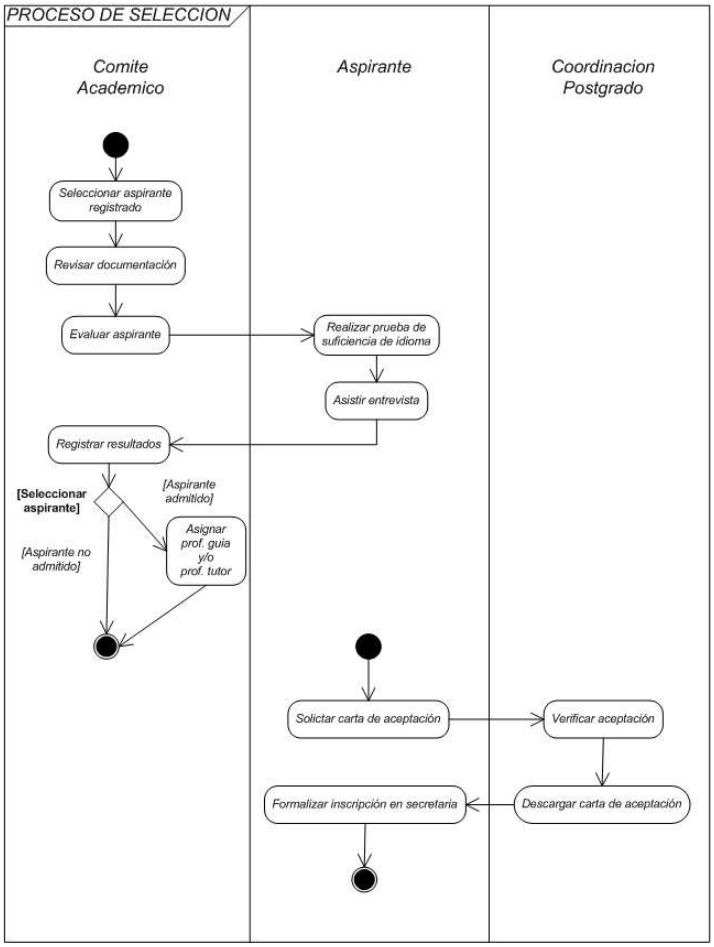

Figura 3.2: Diagrama de actividades de la propuesta del Proceso Selección.

- **Proceso de inscripción:**
  Previo a este proceso, el estudiante revisará en conjunto a su profesor guía, las materias a inscribir. Al ingresar al sistema, el estudiante seleccionará de un listado las materias a inscribir, tras este selección, deberá indicar la forma de pago, propia o financiada, en el caso que sea propia, indicar el número del ticket por concepto de pago de matrícula y pago por número de créditos inscritos, en el caso de que sea financiada, deberá indicar el nombre de la institución u organismo que se encargará de financiar sus estudios, finalmente podra descargar el comprobante de inscripción.

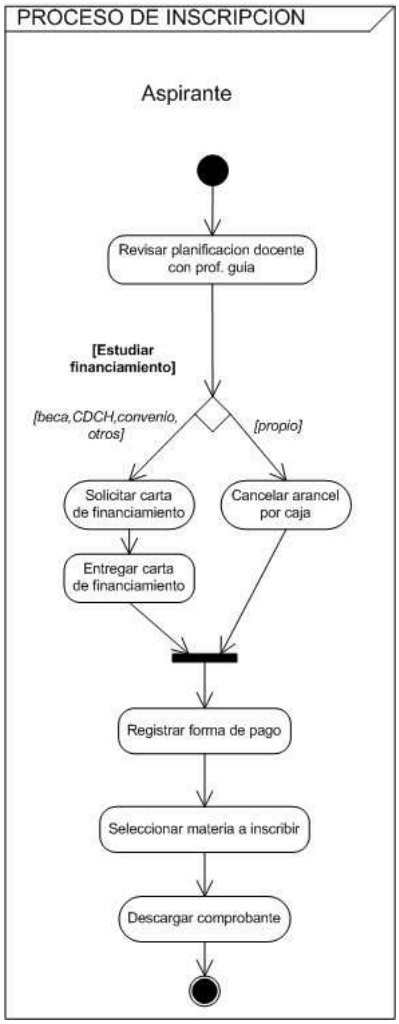

Figura 3.3: Diagrama de actividades de la propuesta del Proceso Inscripción.

- **Proceso de retiro de materia:**
  Al ingresar al sistema, el estudiante detallará el listado de las materias activas. Para retirar alguna de estas, deberá seleccionarla del listado, con la restricción de mantener cursando al menos una. Tras la selección y confirmación de la solicitud de retiro, se le notificará al docente a cargo de la materia sobre tal hecho. Por otro lado, cada solicitud sera evaluada por el comité académico, de ser aceptada, el sistema procederá al retiro formal de la materia, notificando al estudiante vía correo electrónico.

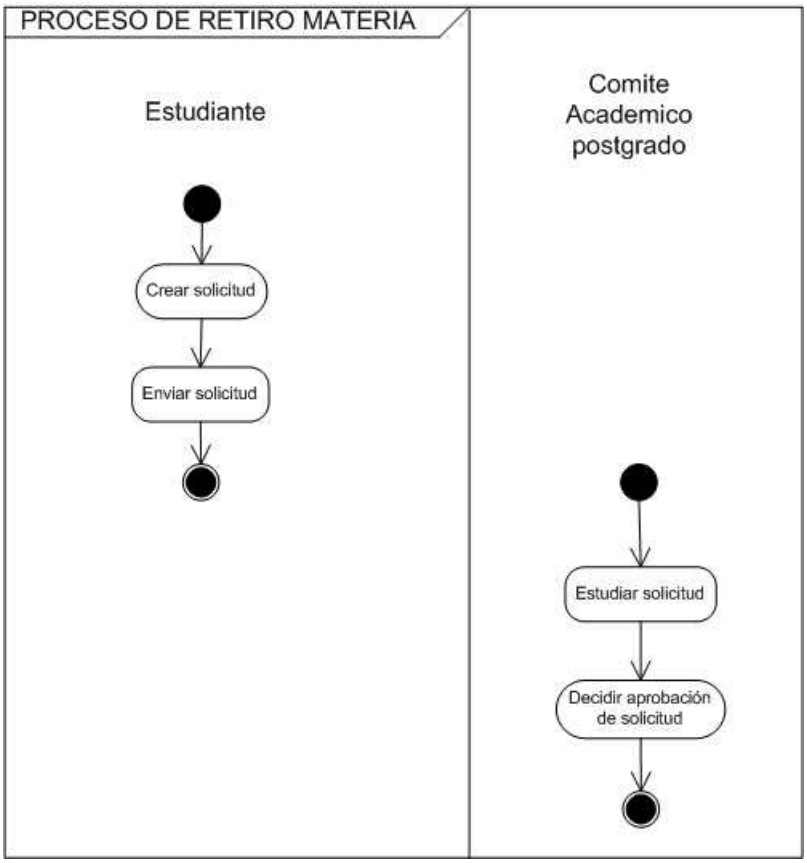

Figura 3.4: Diagrama de actividades de la propuesta del Proceso Retiro Materia.

- **Proceso de desincorporación:**
  Al ingresar al sistema, el estudiante indicará el motivo por el cual desea desincorporarse. Posterior a esto, la solicitud será estudiada por el comité académico, de ser aceptada, el sistema procederá al retiro formal de las materias del estudiante así como también del status del mismo, siendo notificado vía correo electrónico.

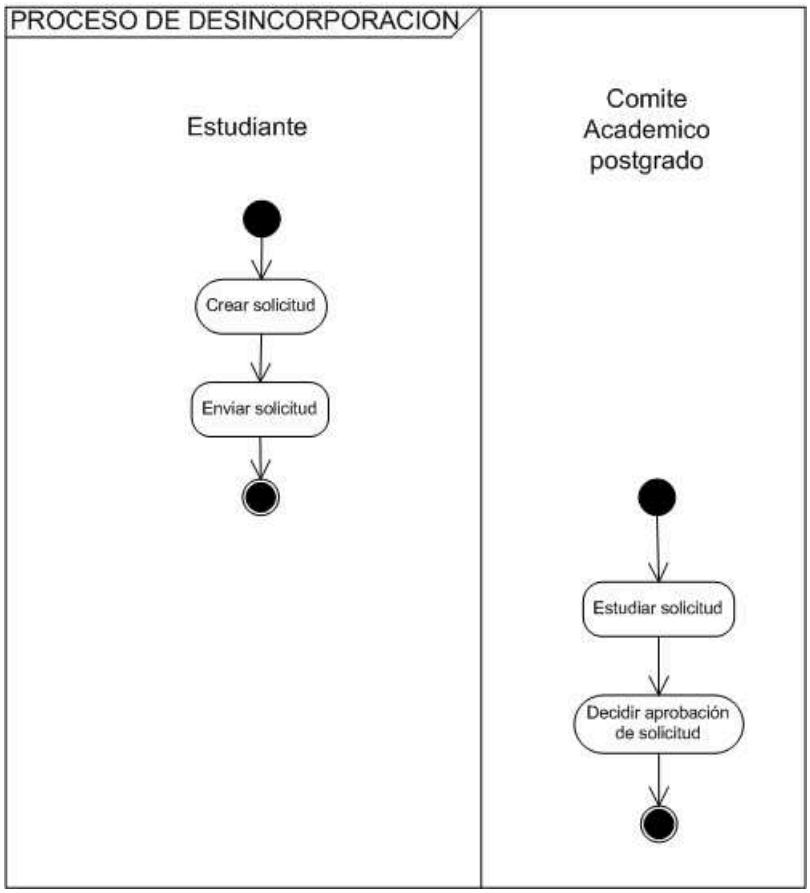

Figura 3.5: Diagrama de actividades de la propuesta del Proceso Desincorporación.

- **Proceso de solicitud de constancia:**
  Al ingresar al sistema, el estudiante seleccionará la constancia a generar y el número de ticket entregado por la caja de la facultad, tras la validación de la transacción, se procederá a la generación y descarga de la constancia solicitada.

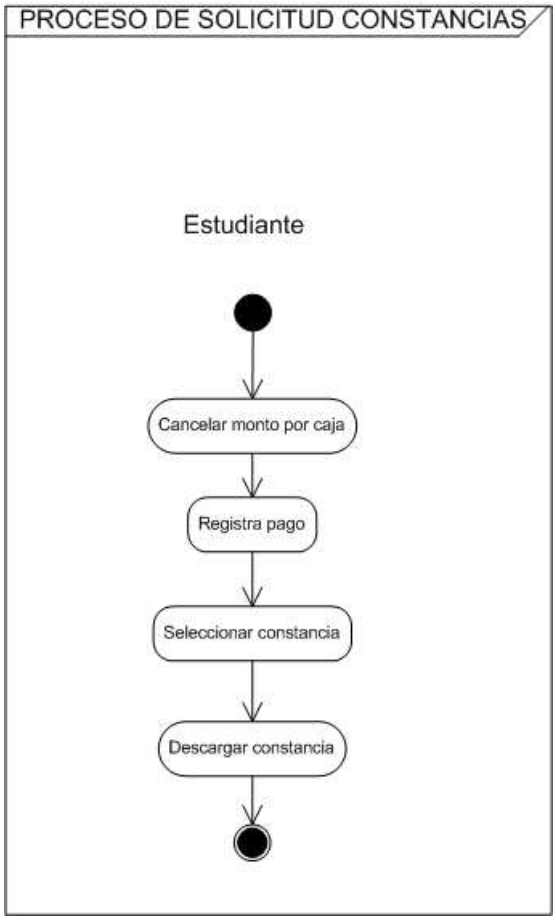

Figura 3.6: Diagrama de actividades de la propuesta del Proceso Solicitud Constancia.

- **Proceso de programación docente:**
  El proceso inicia, con la oferta de materias por parte del docente, quien al ingresar al sistema seleccionará las materias que desee dictar para el período en curso, si desea ofertar una materia que no se encuentre en el listado, podrá proponerla ingresando la información básica de la misma. Con este conjunto de propuestas, el comité académico, seleccionará cuales deben formar parte de la oferta académica o planificación docente del período en curso. Al aceptar una materia, se debera asignar el coordinador a cargo, quien sera notificado vía correo electrónico, el número de secciones a abrir y el número de estudiantes por sección.

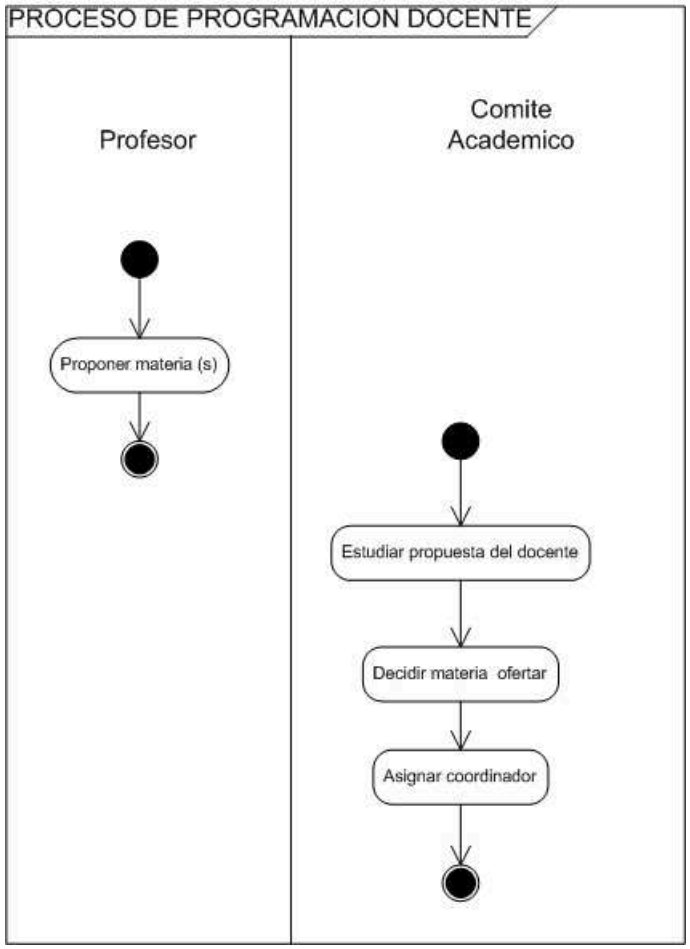

Figura 3.7: Diagrama de actividades de la propuesta del Proceso Programación Docente.

- **Requerimientos:** La aplicación Web cumplió con los requerimientos planteados por el personal encargado de la parte académica de la Coordinación del Postgrado de la Facultad de Ciencias. Para el desarrollo del sistema automatizado se implanto las siguientes funcionalidades:

  - Permitir a los estudiantes llevar a cabo los procesos de admisión, inscripción, retiro y desincorporación vía Web
  - Permitir a los estudiantes consultar el estado de las solicitudes realizadas
  - Permitir la generación en formato digital de determinadas solicitudes que el estudiante demande
  - Proveer mecanismos de modificación de recaudos para optar por algunos de los postgrados
  - Establecer un mecanismo que permitan gestionar la información concerniente a cada postgrado
  - Establecer lineamientos que permitan la unificación de los procesos que se llevan a cabo entre los trece postgrados y su incidencia en los procesos de la coordinación
  - Establecer mecanismos que permitan gestionar las actividades e información referente a los pagos realizados por los estudiantes, en el departamento de administración y presupuesto de la Coordinación de Administración de la Facultad de Ciencias, con el fin de llevar un control de los mismos

3.4. Iteración 1

- Planificación

Iteración 1	
Descripción	Desarrollo del Módulo Admisión, módulo para la postulacion de un aspirante a un postgrado
Historias de Usuario a Desarrollar	1- Desarrollar método `guardar` en el modelo Aspirante 2- Desarrollar interfaz de usuario para modelo Aspirante 3- Desarrollar validaciones para formulario del modelo 4- Generar archivo PDF más comprimido de preinscripción 5- Desarrollar modelo que permite asociar documentos a un aspirante 6- Desarrollar componente Javascript para listar usuarios y materias 7- Desarrollar vista para módulo Aspirante 8- Desarrollar componente ajax para módulo Aspirante 9- Desarrollar componente javascript de paginación para módulo Aspirante
Tiempo Estimado	18 días
Fecha Inicio/Fin	02-04-2009 / 20-04-2009

Tareas por Historia de Usuario

HU 1.- Desarrollar método en el modelo Aspirante

- Crear componente para guardar información del aspirante.
- Establecer las relaciones con los modelos asociados (Admisión, Pago, Transacción) al momento de guardar.

HU 2.- Desarrollar interfaz de usuario para modelo Aspirante

- Montaje de las plantillas (CSS + HTML)
- Diseño base de la página.
- Agregar todo el contenido básico para la generación del formulario.
- Agregar contenido para el proceso de cancelación de arancel

HU 3.- Desarrollar validaciones para formulario del modelo

- Definir las validaciones para los campos de basicos del aspirante, (nombre, apellido, cédula, etc), así como también los mensajes de error respectivos.

HU 4.- Generar archivo PDF y el archivo comprimido de preinscripción

- Diseñar el modelo en html para generar el archivo PDF de preinscripción.
- Crear componente que permita la descarga de documento comprimidos.

HU 5.- Desarrollar modelo que permite asociar documentos a un aspirante

- Asociar la documentación del aspirante

- Permitir modificar la documentación del aspirante con su respectiva asociación.

HU 6.- Desarrollar componente Javascript para listar usuarios y materias

- Generar consulta usuarios o materias por estado
- Despues de hacer la consulta muestre la lista asociada

HU 7.- Desarrollar vista para módulo Aspirante

- Montaje de las plantillas (CSS + HTML)
- Hacer diseño base de la página.
- Agregar contenido para que el aspirante pueda asociar los documentos

HU 8.- Desarrollar componente ajax para módulo Aspirante que permitar subir archivos al servidor

- Permitir enviar archivo digital (imagen o PDF) al servidor como representación de los documentos del aspirante
- Componente que permita asociar los archivos a un documento.

HU 9.- Desarrollar componente javascript de paginación para módulo Aspirante

- Al momento de subir las imagenes de los documentos del aspirante, éstos pueden observarse y paginarse.

Bitácora de desarrollo

Las bitácoras se realizaron generalizando las historias de usuarios y se detallan a continuación:

No.	Tarea	Precede	Fecha Inicio	Fecha Fin	Días Estimados	Días Realizados
1	Arquitectura		02/04/2009	05/04/2009	3	3
2	Montaje de las plantillas (CSS + HTML)		06/04/2009	08/04/2009	2	1.5
3	Análisis y Diseño de la Implementación		13/04/2009	14/04/2009	3	1.5
4	Desarrollo de Interfaz	1	14/04/2009	16/04/2009	2	2
5	Desarrollo de Componentes	3	16/04/2009	20/04/2009	4	3.5

- **Diseño**

En la Figura 3.8 se visualizan las clases de implementación Ruby que dan solución a las historias de usuario mencionadas en la planificación de la iteración

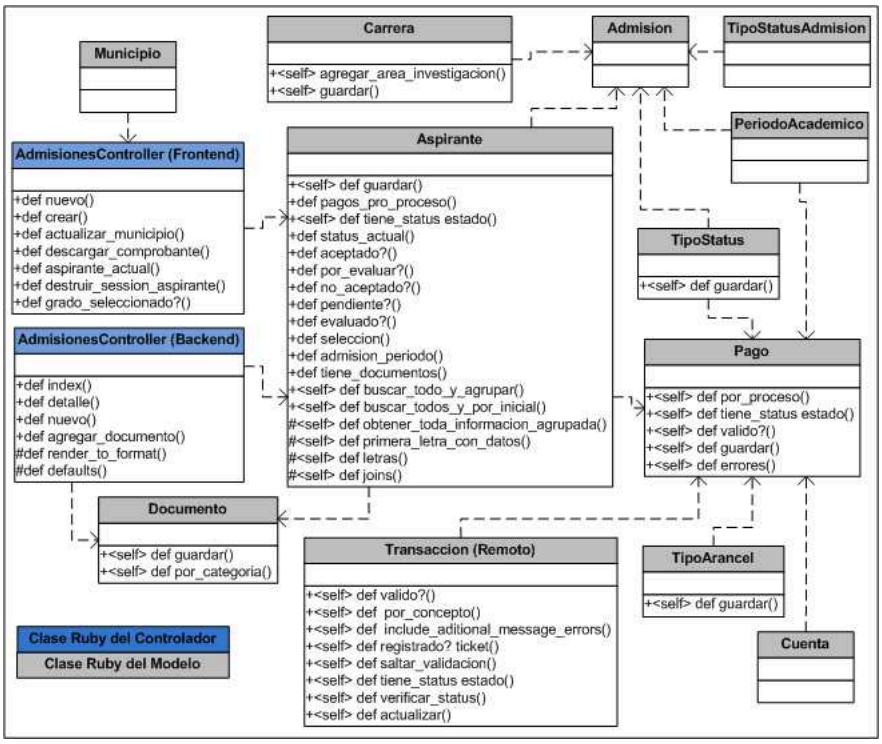

Figura 3.8: Clases de implementación Iteración 1. Módulo Admisión.

- Codificación

En la Figura 3.9 se presenta la implementación del método `crear` ubicado en la clase `AdmisionesController` del frontend encargado en recibir todos los datos incluidos por un aspirante para procesar el registro.

```
# Action que crea al registro nuevo
#
# Parámetros que necesita
# admision: objeto a crear
#
def crear
  options = Hash.new
  options = params
  options = options.merge(:periodo_academico => periodo_academico_actual)
   self.aspirante_actual=params[:aspirante][:cedula]
   result = Aspirante.guardar(params[:aspirante],options)
    @saved = result[:success]
    @models = result[:return]
    if @saved
      flash[:notice] = "Ahora puede descargar su <b>planilla de preinscripción</b>
                        mas la información de requisitos, que debera
                        llevar para la  <b>Coordinacion de Postgrado de la
                        Facultad de Ciencias</b>."

    #GUARDAR USUARIO PARA INGRESAR AL SISTEMA
  end
end
```

Figura 3.9: Método Crear - Clase AdmisionesController. Módulo Admisión.

En la Figura 3.10 se presenta la implementación del método `guardar` ubicado el la Clase Aspirante encargado en registrar un aspirante a un postgrado.

```ruby
def self.guardar(attr,options={})
    objetos = []
    estadocivil = ""
    estadocivil = options[:tipo_estado_civil][:id] unless options[:tipo_estado_civil].nil?
    edo_civil_id = TipoEstadoCivil.find_by_id(estadocivil)

    sexo = ""
    sexo = options[:tipo_sexo][:id] unless options[:tipo_sexo].nil?
    sexo_id = TipoSexo.find_by_id(sexo)

    nacionalidad = ""
    nacionalidad = options[:tipo_nacionalidad][:id] unless options[:tipo_nacionalidad].nil?
    nacionalidad_id = TipoNacionalidad.find_by_id(nacionalidad)

    estado = Estado.find(options[:estado][:id])

    municipio = ""
    municipio = Municipio.find_by_id(options[:municipio][:id]) unless options[:municipio].nil?

    carrera = Carrera.find_by_id(options[:carrera][:id])
    admision = Admision.new(:carrera => carrera)

    aspirante_pendiente = tiene_status("PENDIENTE")

    attr = attr.merge(:tipo_estado_civil_id => edo_civil_id)
    attr = attr.merge(:tipo_sexo_id => sexo_id)
    attr = attr.merge(:tipo_nacionalidad_id => nacionalidad_id)
    attr = attr.merge(:estado_id => estado)
    attr = attr.merge(:municipio_id => municipio)
    aspirante = new(attr)
    aspirante.cedula = attr[:cedula]
    result = Pago.valido?(options,{:tipo_proceso => "admision"})

    success = aspirante.valid?
    success = (admision.valid? and success)
    success = (result[:success] and success)

    if success
       aspirante[:cedula] = attr[:cedula]
       aspirante.save
       Admision.create( :aspirante_cedula => aspirante.id,
                        :carrera => carrera,
                        :periodo_academico_id => options[:periodo_academico],
                        :tipo_status_id => aspirante_pendiente[:id])
      pagos_hash = Hash.new
      pagos_hash[:persona_id] = attr[:cedula]
      pagos_hash[:persona_type] = Aspirante.to_s
      pagos_hash[:pagos] = result[:return]
      objetos.push(Pago.guardar(pagos_hash))
      #Envia un correo al aspirante
      Correo::deliver_enviar_correo_admision(aspirante,options)
    else
      result[:return].each do |obj|
        objetos << obj[:pago]
        objetos << obj[:transaccion]
      end
    end
    objetos << aspirante
    objetos << admision
    return {:success => success, :return => objetos.flatten}
end
```

Figura 3.10: Método Guardar - Clase Aspirante. Módulo Admisión.

- **Pruebas**

Las pruebas se realizaron durante todo el proceso de desarrollo para verificar cada método creado, no obstante las pruebas unitarias y de aceptación relacionadas a las historias asociadas del módulo de Admisiones en esta Iteración fueron las más acertadas y concisas, verificando que cada componente desarrollado funcionara sin problemas. Estas pruebas se realizaron finalizado el módulo, a continuación se describre cada prueba realizada.

No.	**Módulo**	**Descripción del Caso de Prueba**	**Resultado Esperado**	**Resultado Obtenido**	**Motivo de la Falla**
1	Admisión (Frontend)	Aspirante con una postulación a una carrera ya seleccionada	No debe existir dos registros para un aspirante para el mismo período	Se valida sin problemas para que el aspitante no se registre dos o mas veces	
2	Admisión (Frontend)	Aspirante de otra Facultad optando por un postgrado	Se crea un registro de la admisión para la carrera solicitada, se da la opción de descargar constancia de preinscripción, se notifica vía correo	Aspirante de otra Facultad procede con las preinscripción al postgrado notificando por correo	
3	Admisión (Backend)	Verificar ticket de pago de materia de preinscricpion	La coordinación académica podrá registrar los documentos del aspirante para poder ser evaluado por el comite académico	La coordinación académica podrá registrar los documentos del aspirante para poder ser evaluado por el comite académico	

3.5. Iteración 2

- Planificación

Iteración 2	
Descripción	Desarrollo del Módulo Solicitudes, módulo al servicio del estudiante para llevar a cabo solicitudes refrentes a constancias, retiro de materias y desincorporación
Historias de Usuario a Desarrollar	10- Desarrollar modelos para el módulo Solicitudes 11- Establecer relaciones para los modelos del módulo Solicitudes 12- Desarrollar vistas para el modulo Solicitudes 13- Desarrollar validaciones para el módulo Solicitudes 14- Desarrollar una adaptación de las vistas del modulo de solicitudes para pago arancel 15- Desarrollar método para el seguimiento de estado de las solicitudes 16- Generar archivo PDF para el módulo de Solicitudes 18- Listar solicitudes ya pedidas por el estudiante 34- Desarrollar método de cancelación de solicitud de constancia 35- Desarrollar método para agregar comentarios a la solicitud
Tiempo Estimado	36 días
Fecha Inicio/Fin	04-04-2009 / 10-05-2009

Tareas por Historia de Usuario

HU 10.- Desarrollar modelos para el módulo Solicitudes

- Análisis de la base de datos actual y de los requerimientos.
- Establecer los modelos necesarios para las peticiones de solicitudes, incluyendo las dependencias que se necesiten de la base de datos actual.

HU 11.- Establecer relaciones para los modelos del módulo Solicitudes

- Verificación de las dependencias entre modelos.
- Establecer las relaciones necesarias para que las consultas sean rápidas y sencillas.

HU 12.- Desarrollar vistas para el modulo Solicitudes

- Diseño de la arquitectura para previa visualización.
- Montaje de las plantillas (CSS + HTML).
- Hacer diseño base de la página.
- Agregar contenido para que el estudiante pueda realizar las peticiones.

HU 13.- Desarrollar validaciones para formulario del modelo Solicitudes

- Definir las validaciones en el modelo respectivo, donde se consideren inclusión del motivo de solicitud, selección de solicitud, pago de arancel.

HU 14.- Desarrollar una adaptación de las vistas del modulo de solicitudes para pago arancel

- Rediseñar página base de solicitudes.
- Modificar vista para acoplar componente de pago de arancel.

HU 15.- Desarrollar método para el seguimiento del estado de las solicitudes

- Componente que permita cambiar de estatus de la solicitud y guarde dichos cambios.
- Dependiendo del estatus seleccionado, ejecutar las acciones respectivas acorde a la solicitud.

HU 16.- Generar archivo PDF para el módulo de Solicitudes

- Formato de contenido de constancia.
- Descargar las solicitudes de constancias en formato PDF.

HU 18.- Listar solicitudes ya pedidas por el estudiante

- Ver el detalle de las solicitudes que ya han sido pedidas por el estudiante.
- Clasificar o listar las solicitudes por estatus.

HU 34.- Desarrollar método de cancelación de solicitud de constancia

- Ingresar tipo de pago de cancelación de solicitud.
- Validar que el tipo de pago sea unicamente para solicitud de constancia.

HU 35.- Desarrollar método para agregar comentarios a la solicitud

- Ingresar comentario a la solicitud con fecha de registro y por quién fue realizado.

Bitácora de desarrollo

Las bitácoras se realizaron generalizando las historias de usuarios y se detallan a continuación:

No.	Tarea	Precede	Fecha Inicio	Fecha Fin	Días Estimados	Días Realizados
1	Arquitectura		04/04/2009	05/04/2009	1	1
2	Montaje de las plantillas (CSS + HTML)		05/04/2009	05/04/2009	1	0.5
3	Análisis y Diseño de la Implementación		05/04/2009	06/04/2009	1	0.5
4	Desarrollo de Interfaz	1	06/04/2009	08/04/2009	2	2
5	Desarrollo de Componentes Frontend	3	09/04/2009	24/04/2009	15	13
6	Desarrollo de Componentes Backend	3	24/04/2009	10/05/2009	16	16

- **Diseño**

En la Figura 3.11 se visualizan las clases de implementación Ruby que dan solución a las historias de usuario mencionadas en la planificación de la iteración

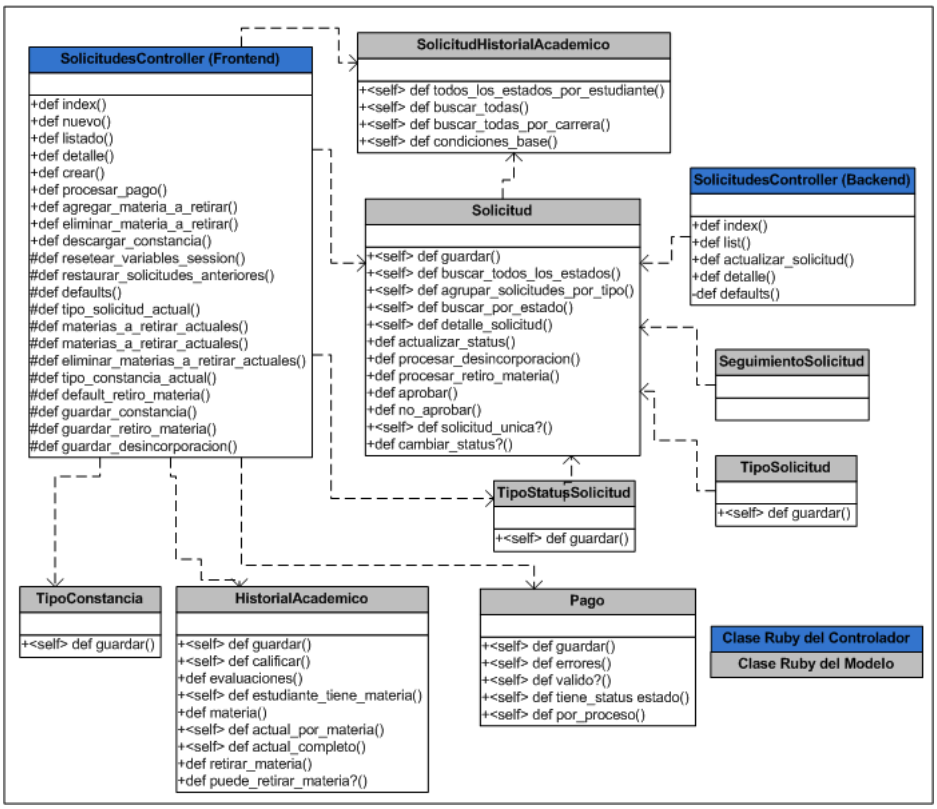

Figura 3.11: Clases de implementación Iteración 2. Módulo Solicitudes.

- Codificación

En la Figura 3.12 se presenta la implementación del método `crear` ubicado en la clase `SolicitudesController` del frontend estudiante encargado crear todas los solicitudes de un estudiante para procesarlas dependiendo del tipo de solicitud.

```
def crear
  case self.tipo_solicitud_actual
    when "SC"
      if TipoConstancia.find_by_nombre(params[:tipo_constancia][:nombre])
        resultado = guardar_constancia
      else
        resultado = {}
        resultado = { :success => false,
                      :return => [],
                      :error_message => "Debe seleccionar una constancia"}
      end
    when "SD"
      resultado =  guardar_desincorporacion
    when "SRM"
      resultado = guardar_retiro_materia
  end
  @objetos = resultado[:retorno] ? resultado[:retorno].flatten : []
  @exito = resultado[:exito]
  if @exito
    unless resultado[:mantener_session]
      resetear_variables_session
    end
      flash[:notice] = resultado[:notice_message] || 'Sus Solicitudes han sido procesadas exitosamente'
  else
    flash[:error] = resultado[:error_message] || 'No se ha podido procesar su solicitud'
  end
end
```

Figura 3.12: Método Crear - Clase SolicitudesController. Módulo Solicitudes.

En la Figura 3.13 se presenta la implementación del método `descargar_constancia` ubicado en la clase `SolicitudesController` del frontend estudiante encargado de permitir la descarga de la constancia solicitada por el estudiante.

```
def descargar_constancia
  #TODO: HACER LOS FORM;ATOS DE CONSTANCA
    begin
    @historial_academico_estudiante  = HistorialAcademico.first(:conditions => {
                                          :periodo_academico_id => periodo_academico_actual,
                                          :estudiante_cedula => estudiante_actual[:estudiante_cedula]})
    tipo_constancia = self.tipo_constancia_actual.downcase.gsub(/ /, "_")
     respond_to do |format|
        format.html { render :partial => "constancia",:layout => false}
        format.pdf  do
          pdf = render_to_pdf(:action => 'constancia.rpdf', :layout => false)
          send_data  pdf, :filename => "constancia_#{tipo_constancia}_#{estudiante_actual[:estudiante_cedula]}",
                                                                    :type => "application/pdf"
        end
      end
    rescue
      redirect_to index_principal_url
    end
  end
```

Figura 3.13: Método Descargar Constancia - Clase SolicitudesController. Módulo Solicitudes.

En la Figura 3.14 se presenta la implementación del método `guardar` definido en la clase `Solicitud`, encargado de registrar todas las solicitudes de un estudiante.

```
def self.guardar(attr,opciones={})
  tipo_solicitud = TipoSolicitud.find_by_id(attr[:tipo_solicitud_id])
  tipo_status_solicitud = TipoStatusSolicitud.first(:conditions=>{
                                                    :tipo_solicitud_id => attr[:tipo_solicitud_id],
                                                    :descripcion => "PENDIENTE"})
  solicitud = Solicitud.new( :tipo_solicitud_id => tipo_solicitud[:id],
                             :tipo_status_solicitud_id => tipo_status_solicitud[:id],
                             :fecha_hora_creacion => Time.now,:fecha_hora_ultimo_estado => Time.now,
                             :motivo => attr[:motivo])
  success =  solicitud.valid? && self.solicitud_unica?(attr[:tipo_solicitud_id],opciones)
  if success
    solicitud.save
    SeguimientoSolicitud.create(:solicitud_id => solicitud[:id],
                                :tipo_status_solicitud_id => tipo_status_solicitud[:id],
                                :usuario_cedula => opciones[:estudiante_cedula],
                                :fecha_creacion => Time.now,
                                :fecha_actualizacion => Time.now)
    solicitud_historial_academico = SolicitudHistorialAcademico.new(
                                      :solicitud_id => solicitud[:id],
                                      :estudiante_cedula => opciones[:estudiante_cedula],
                                      :periodo_academico_id => opciones[:periodo_academico],
                                      :nombre_seccion => opciones[:nombre_seccion],
                                      :materia_codigo => opciones[:materia_codigo],
                                      :plan_nombre => opciones[:plan_nombre],
                                      :carrera_id => opciones[:carrera_id])
    solicitud_historial_academico.solicitud_id = solicitud[:id]
    solicitud_historial_academico.save
  end
  return {:success => success, :return => solicitud}
end
```

Figura 3.14: Método guardar solicitud - Clase Solicitud. Módulo Solicitudes.

- Pruebas

Las pruebas de aceptación con el cliente se realizaron después de culminar la iteración, es decir, después de culminar el módulo, en ellas surgieron observaciones, más no errores debido a que durante el desarrollo del módulo se pudieron observar y corregir, éstas observaciones fueron canalizadas y corregidas para la modificación de la aplicación. A continuación se muestra las pruebas unitarias que se realizaron.

No.	Módulo	Descripción del Caso de Prueba	Resultado Esperado	Resultado Obtenido	Motivo de la Falla
1	Solicitudes (Frontend Estudiante)	Estudiante solicita constancia con ticket válido	Se valída el ticket de pago con éxito y permite solicitar constancia	Estudiente solicita constancia exitosamente con ticket de pago válido	
2	Solicitudes (Frontend Estudiante)	Estudiante solicita constancia con ticket inválido	Se verifica el ticket de pago estando inválido, no se permite solicitar constancia	No se permite realizar la solicitud por ingresar ticket de pago inválido	
3	Solicitudes (Frontend Estudiante)	Estudiante retira una materia o varias materias quedando cursando al menos una	Se permite hacer la solicitud de retirar las materias que desea quedando cursando al menos una	Estudiante retira las materias que desea quedando cursando al menos una	
4	Solicitudes (Frontend Estudiante)	Estudiante retira todas las materias	No se permite el retiro total de materias, debe estar cursando al menos una, sino sería desincorporación	Mensaje de error debido a que se debe estar cursando al menos una materia	
5	Solicitudes (Frontend Estudiante)	Estudiante solicita desincorporación	Se permite la solicitud de desincorporación con el previo motivo de ésta	Se permite la solicitud de desincorporación con el previo motivo de ésta	
6	Solicitudes (Frontend Personal Administrativo)	Comité verifica solicitudes de retiro de materia	El comité académico verifica las peticiones de retiro de materia y se cambia estado de solicitud	El comité académico cambia estado de solicitud aceptando o no la solicitud	
7	Solicitudes (Frontend Personal Administrativo)	Comité verifica solicitudes de desincorporación	El comité académico verifica las peticiones de desincorporación y dependiendo del caso las acepta o no cambiando el estatus de la solicitud, se envía por correo el resultado	El comité académico acepta o rechaza cambiando el estatus de la solicitud, se envía por correo electrónico el resultado al estudiante	

3.6. Iteración 3

- Planificación

Iteración 3	
Descripción	Desarrollo del Módulo Selección, módulo donde se gestiona la evaluación y posterior selección de un aspirante a un postgrado, dicha accion es efectuada por el comité académico del mismo
Historias de Usuario a Desarrollar	24- Permitir listar todos los aspirantes 25- Visualizar el detalle de solicitud de un estudiante 26- Filtrar lista de aspirantes por estatus 27- Crear un formulario que permita evaluar al aspirante 28- Notificar vía correo electrónico resultado de evaluación 29- Generar planilla de inscripcion de Secretaria Central
Tiempo Estimado	5 días
Fecha Inicio/Fin	21-04-2009 / 26-04-2009

Tareas por Historia de Usuario

HU 24.- Permitir listar todos los aspirantes

- Análisis de las organizaciones existentes en la facultad.
- Establecer vista de listado de usuarios (estudiantes).
- Adaptar el componente javascript, tipo agenda que permita listar los aspirantes por organización.

HU 25.- Visualizar el detalle de solicitud de un estudiante

- Consultar información asociada al estudiante.
- Mostrar la información asociada al estudiante.

HU 26.- Filtrar lista de aspirantes por estatus

- Verificar los tipo de estatus existentes por admisión.
- Componente que permita el filtrado de aspirantes por estatus.

HU 27.- Crear un formulario que permita evaluar al aspirante

- Diseñar página base el formulario.
- Definir formulario para evaluar aspirante.
- Establecer las propiedades y eventos que se manejaran en este componente.

HU 28.- Notificar vía correo electrónico resultado de evaluación

- Diseñar formato de contenido para envío de correo electrónico.

- Enviar de correo electrónico con la información de la notificación.

HU 29.- Generar planilla de inscripcion de Secretaria Central

- Diseñar formato de contenido similar a la planilla de inscripción de Secretaria Central física
- Generar la planilla de inscripcion de Secretaria Central.

Bitácora de desarrollo

Las bitácoras se realizaron generalizando las historias de usuarios y se detallan a continuación:

No.	Tarea	Precede	Fecha Inicio	Fecha Fin	Días Estimados	Días Realizados
1	Arquitectura		21/04/2009	21/04/2009	1	1
2	Montaje de las plantillas (CSS + HTML)		22/04/2009	22/04/2009	1	0.5
3	Análisis y Diseño de la Implementación		23/04/2009	24/04/2009	1	1
4	Desarrollo e Integración de Componentes	3	24/04/2009	26/04/2009	2	3

- Diseño

En la Figura 3.15 se visualizan las clases de implementación Ruby que dan solución a las historias de usuario mencionadas en la planificación de la iteración

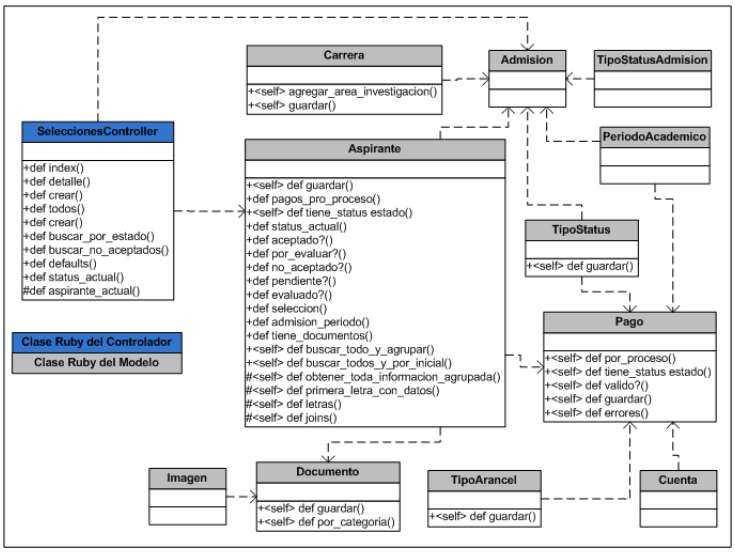

Figura 3.15: Clases de implementación Iteración 3. Módulo Selecciones.

- **Codificación**

En la Figura 3.16 se presenta la implementación del método `crear` definido en la clase `SeleccionesController` donde la comisión académica se encarga de crear aquella selección de un estudiante cuando es aceptado o no aceptado.

```
def crear
  @aspirante =  Aspirante.find(self.aspirante_actual)
  result = @aspirante.seleccion(params[:admision],:periodo_academico => periodo_academico_actual)
  @admision = @aspirante.admisiones.find_by_periodo_academico_id(periodo_academico_actual)
  @status_aceptado = Aspirante.tiene_status("aceptado")
  @status_no_aceptado = Aspirante.tiene_status("no_aceptado")
  @objects = result[:return]
  @exito = result[:success]
  if @exito
    flash[:notice] = "<b>#{@aspirante[:nombre]} #{@aspirante[:apellido]}</b> fue evaluado con exito!!"
  end
end
```

Figura 3.16: Método Crear - Clase SeleccionesController. Módulo Selección.

En la Figura 3.17 se presenta la implementación del método `seleccion` definido en la clase `Aspirante` el cual recibe los parámetros del controlador selecciones cuando se evalúa a un estudiante para guardar los registros del aspirante cuando es aceptado o no aceptado.

```
def seleccion(attr,options={})
  admision = self.admisiones.find_by_periodo_academico_id(options[:periodo_academico])
  success = true
  if attr[:tipo_status_id]
    admision.update_attributes(attr)
    Correo::deliver_enviar_correo_aspirante_resultado(self,options)
  else
    success = false
    admision.errors.add(:tipo_status_id,"Debe selecionar una opci&oacute;n")
  end
  if self.aceptado?(options[:periodo_academico])
    atributos ={}
    atributos[:cedula] = self[:cedula]
    atributos[:municipio_id_vive_actual] = self[:municipio_id]
    atributos[:estado_id_origen] = self[:estado_id]
    usuario = {}
    usuario[:cedula] = self[:cedula]
    usuario[:primer_nombre] = self[:nombre]
    usuario[:primer_apellido] = self[:apellido]
    usuario[:tipo_estado_civil_id] = self[:tipo_estado_civil_id]
    usuario[:tipo_sexo_id] = self[:tipo_sexo_id]
    usuario[:tipo_nacionalidad_id] = self[:tipo_nacionalidad_id]
    usuario[:correo] = self[:correo]
    usuario[:celular] = self[:celular]
    usuario[:telefono] = self[:telefono]
    usuario[:fecha_nacimiento] = self[:fecha_nacimiento]
    options = options.merge(:usuario => usuario)
    Estudiante.guardar(atributos,options)
  end
  {:success => success,:return => [admision]}
end
```

Figura 3.17: Método Selección - Clase Aspirante. Módulo Selección.

- **Pruebas**

Las pruebas de aceptación con el cliente se realizaron después de culminar la iteración, en ellas surgieron observaciones, más no errores debido a que durante el desarrollo del módulo se pudieron observar y corregir, éstas observaciones fueron canalizadas y corregidas para la modificación de la aplicación. A continuación se muestra las pruebas unitarias que se realizaron.

No.	**Módulo**	**Descripción del Caso de Prueba**	**Resultado Esperado**	**Resultado Obtenido**	**Motivo de la Falla**
1	Selección	Aceptar aspirante de acuerdo a los requisitos y resultado de la prueba de insuficiencia	Aspirante aceptado exitosamente	Aspirante aceptado exitosamente se envía correo electrónico informando la situación	
2	Selección	No aceptar al aspirante de acuerdo a los requisitos y resultado de la prueba de insuficiencia	Aspirante no aceptado	Aspirante no aceptado se envía correo electrónico informando la situación	

3.7. Iteración 4

- Planificación

Iteración 4	
Descripción	Desarrollo del Módulo Inscripción, módulo estudiantil para la selección e inscripción de materias a cursar para el período académico actual.
Historias de Usuario a Desarrollar	30- Listar materias permitidas para inscribir 31- Filtrar materias de acuerdo a su tipo 32- Solicitar los aranceles por concepto de pago 33- Descargar constancia de inscripción
Tiempo Estimado	11 días
Fecha Inicio/Fin	28-04-2009 / 09-05-2009

Tareas por Historia de Usuario

HU 30.- Listar materias permitidas para inscribir

- Generar consultas para obtener todas las materias permitidas para un estudiante.
- Establecer vista para el listado de materias.
- Integrar las vistas con las consultas.

HU 31.- Filtrar materias de acuerdo a su tipo

- Definir los tipo de materias existentes.
- Realizar consultas que filtren dichas materias por tipo.

HU 32.- Solicitar los aranceles por concepto de pago

- Modificar vista para agregar formato de arancel de pago.
- Calcular el monto a cancelar de acuerdo a los créditos inscritos y por concepto de matricula dependiendo del grado.
- Consultar pago hecho con el sistema de caja.

HU 33.- Descargar constancia de inscripción

- Diseñar formato de contenido para generar la constancia.
- Permitir la descarga de las constancia de inscripción despues de validar el pago de arancel correpondiente.

Bitácora de desarrollo

Las bitácoras se realizaron generalizando las historias de usuarios y se detallan a continuación:

No.	Tarea	Precede	Fecha Inicio	Fecha Fin	Días Estimados	Días Realizados
1	Arquitectura		28/04/2009	30/04/2009	2	1
2	Montaje de las plantillas (CSS + HTML)		30/04/2009	01/05/2009	1	1
3	Análisis y Diseño de la Implementación	2	02/05/2009	04/05/2009	2	2
4	Desarrollo e Integración de Componentes	3	05/05/2009	09/05/2009	4	4

- **Diseño**

En la Figura 3.18 se visualizan las clases de implementación Ruby que dan solución a las historias de usuario mencionadas en la planificación de la iteración

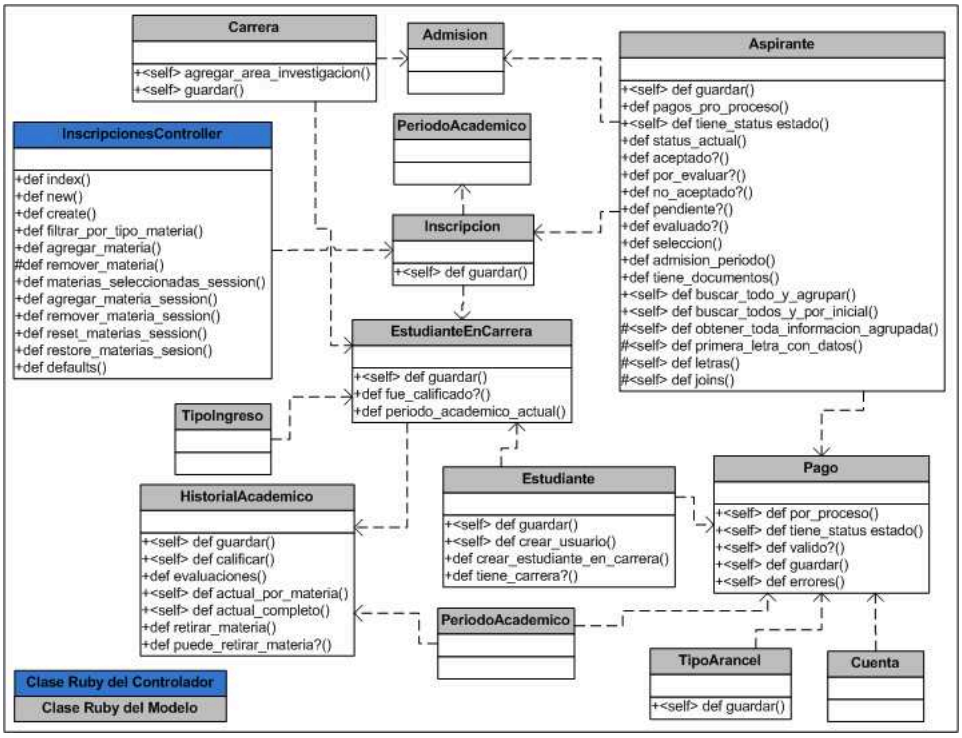

Figura 3.18: Clases de implementación Iteración 4. Módulo Inscripciones.

- Codificación

En la Figura 3.19 se presenta la implementación del método `guardar` definido en la clase `Inscripcion` donde el estudiante realiza el proceso de inscripción seleccionando las materias a cursar más la cancelación del arancel.

```
def self.guardar(attr,options={})
  exito = true
  objetos = []
  options = options.merge(:periodo_academico => attr[:periodo_academico])
  pago_resultado = Pago.valido?(options,:tipo_proceso => "inscripcion")
  exito = (pago_resultado[:success] and exito)

  if exito
      inscripcion_hash = Hash.new
      inscripcion_hash[:carrera_id] = attr[:carrera]
      inscripcion_hash[:periodo_academico_id] = attr[:periodo_academico]
      inscripcion_hash[:estudiante_cedula] = attr[:estudiante_cedula]
      inscripcion_hash[:ip_origen] = attr[:ip_origen]
      inscripcion = Inscripcion.new(inscripcion_hash)
      inscripcion.save

      tipo_status_materia = TipoStatusMateria.find("SC")
      historial_academico_hash = Hash.new
      historiales_academicos = []
      attr[:materias_codigos].each do |materia_codigo|
        historial_academico_hash[:materia_codigo] = materia_codigo
        historial_academico_hash[:carrera] = attr[:carrera]
        historial_academico_hash[:periodo_academico] = attr[:periodo_academico]
        historial_academico_hash[:estudiante_cedula] = attr[:estudiante_cedula]
        historial_academico_hash[:tipo_status_materia] = tipo_status_materia[:id]
        historial_academico_hash[:tipo_examen_id] = TipoExamen.find("SN")
        historiales_academicos << HistorialAcademico.guardar(historial_academico_hash)
      end

      pago_hash = Hash.new
      pago_hash[:pagos] = pago_resultado[:return]
      pago_hash[:persona_id] = attr[:estudiante_cedula]
      pago_hash[:persona_type] = Estudiante.to_s
     objetos.push(Pago.guardar(pago_hash))
  else
    objetos.push(pago_resultado[:return])
  end
  {:exito => exito, :objetos => objetos.flatten }
end
```

Figura 3.19: Método Guardar - Clase Inscripción. Módulo Inscripción.

- Pruebas

Las pruebas de aceptación con el cliente se realizaron después de culminar la iteración, es decir, después de culminar el módulo, en ellas surgieron observaciones, más no errores debido a que durante el desarrollo del módulo se pudieron observar y corregir, éstas observaciones fueron canalizadas y corregidas para la modificación de la aplicación. A continuación se muestra las pruebas unitarias que se realizaron.

No.	**Módulo**	**Descripción del Caso de Prueba**	**Resultado Esperado**	**Resultado Obtenido**	**Motivo de la Falla**
1	Inscripción	Estudiante se inscribe sin materias seleccionadas	No se permite la inscripción, debe elegir al menos una materia	Se le notifica al estudiante seleccionar al menos una materia para proceder con la inscripción	
2	Inscripción	Estudiante se inscribe seleccionando materias sin ingresar pago	No se permite la inscripción, debe ingresar el pago realizado	Inscripción no procede por no ingresar metodo de pago realizado	
3	Inscripción	Estudiante se inscribe seleccionando materias con pago realizado	Estudiante se inscribe exitosamente	Estudiante se inscribe exitosamente	
4	Inscripción	Estudiante se inscribe exitosamente	Se envia notificación de inscripción vía correo electrónico	Se envia notificación de inscripción vía correo electrónico	
5	Inscripción	Estudiante después de inscribirse, descarga constancia de inscripión	Constancia se descarga con la información de inscripción del estudiante	Procede la descarga de la constancia de inscripción	

3.8. Iteración 5

- Planificación

Iteración 5	
Descripción	Desarrollo del Módulo Evaluaciones, módulo donde se podran asociar las evaluaciones a una materia, en tal sentido el docente podra calificar a los estudiantes de manera continua.
Historias de Usuario a Desarrollar	46- Listar aquellas materias coordinadas por el docente 47- Asociar diferentes evaluaciones a una materia 48- Definir tema, porcentaje y fecha a la evaluación 49- Modificar las evaluaciones a una materia
Tiempo Estimado	7 días
Fecha Inicio/Fin	20-04-2009 / 28-04-2009

Tareas por Historia de Usuario

HU 46.- Listar aquellas materias coordinadas por el docente

- Buscar todas las materias coordinadas por el docente actual.
- Desarrollo y diseño de interfaz donde se liste todas las materias coordinadas por el docente actual.
- Para cada materia, agregarle el evento u acción de selección para poder asociarle el conjunto de evaluaciones.
- Mantener en modo activo la materia seleccionada.

HU 47.- Asociar diferentes evaluaciones a una materia

- Desarrollo, diseño y montaje de una interfaz donde se liste las diferentes evalaciones, donde se permita la selección multiple de las mismas.
- Guardar temporalmente las evaluaciones seleccionadas del listado.

HU 48.- Definir tema, porcentaje y fecha a la evaluación.

- Desarrollo, diseño y montaje de una interfaz donde se muestren las evaluaciones seleccionadas, en donde se les pueda asociar tema, porcentaje y fecha de evaluación, todos los elementos anteriores debera ser obligatorios, exceptuando el último.
- Crear en el modelo respectivo, un método, se asocie las evaluaciones seleccionadas más la información antes dicha a la materia que fue seleccionada anteriormente.

HU 49.- Modificar las evaluaciones a una materia.

- Mostrar las evaluciones asociadas a la materia.

- Asociar nuevas evaluaciones o desasociar las existentes.
- Actualizar la información referente a las evaluaciones.
- Guardar los cambios efectuados.

Bitácora de desarrollo

Las bitácoras se realizaron globalizando las historias de usuario de forma general y se clasificaron de la siguiente manera:

No.	Tarea	Precede	Fecha Inicio	Fecha Fin	Días Estimados	Días Realizados
1	Arquitectura		17/05/2009	19/05/2009	2	2
2	Análisis y Diseño de la implementación	1	20/05/2009	21/05/2009	1	1
3	Montaje de Interfaz	2	22/05/2009	24/05/2009	2	2
4	Diseño e implementación de base de datos	3	24/05/2009	25/05/2009	1	2
5	Establecer con el módulo de calificaciones	4	25/05/2009	25/05/2009	0.75	0
6	Modificación de evaluaciones	5	26/05/2009	27/05/2009	2	1
7	Validaciones		27/05/2009	28/05/2009	1	1

- **Diseño**

En esta iteración se desarrolló el módulo que permitirá definir las distintas evaluaciones que seran ejecutadas a lo largo del período académico, de acuerdo a esto en la Figura 3.20 se visualizan las clases de implementación Ruby que dan solución a la *Asociación de Evaluaciones por Materia*.

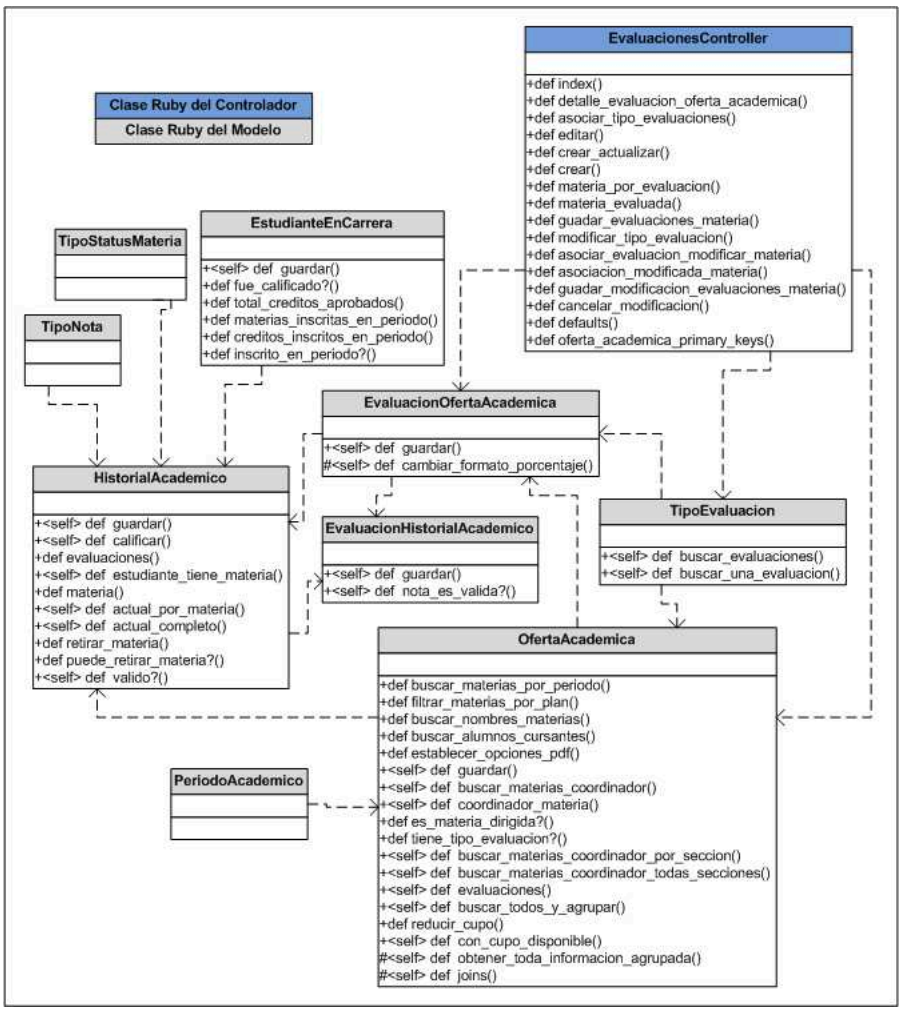

Figura 3.20: Clases de implementación Iteración 7. Módulo Evaluaciones.

- **Codificación**

A continuación en la 3.21 y 3.22 se presentan la implementación del método `crear` y `guardar` respectivamente de la clase `EvaluacionController` y `EvaluacionOfertaAcademica`, los cuales son los encargados de asociar a una materia en particular sus evaluaciones.

```
def crear
  atributos = {}
  atributos = atributos.merge(self.oferta_academica_primary_keys)
  atributos[:evaluaciones_oferta_academica] = params[:evaluacion_oferta_academica]
  resultado = EvaluacionOfertaAcademica.guardar(atributos)
  @exito = resultado[:exito]
  @objetos = resultado[:retorno]
  errores_mensajes = resultado[:errores_mensajes]
  if @exito
    @oferta_academica = OfertaAcademica.buscar_materias_coordinador_por_seccion(self.oferta_academica_primary_keys)
  else
    flash[:error] = errores_mensajes.join("<br>- ")
  end
end
```

Figura 3.21: Método Crear. Clase EvaluacionesController. Módulo Evaluaciones.

En la 3.22 se presenta el código fuente que permite hacer las validaciones, hacer el cálculo y establecer las relaciones con el la materia ofertada y las evaluaciones.

```
def self.guardar(attr,opciones={})
  modelos = []
  errores_mensajes = []
  exito = true
  porcentaje_total = 0
  secciones = OfertaAcademica.buscar_materias_coordinador_todas_secciones(attr)
  evaluaciones_oferta_academica  = attr[:evaluaciones_oferta_academica].keys
  evaluaciones_oferta_academica.each do |eoa|
    tipo_evaluacion = TipoEvaluacion.find_by_id(eoa)
    if tipo_evaluacion
      nombre_secciones = secciones.map(& :nombre_seccion)
       nombre_secciones.each do |seccion|
          evaluacion_oferta_academica = new(
                                    :materia_codigo => attr[:materia_codigo],
                                    :periodo_academico_id => attr[:periodo_academico],
                                    :nombre_seccion => seccion,
                                    :tipo_evaluacion_id => eoa,
                                    :porcentaje => attr[:evaluaciones_oferta_academica][eoa][:porcentaje],
                                    :tema => attr[:evaluaciones_oferta_academica][eoa][:tema],
                                    :fecha => attr[:evaluaciones_oferta_academica][eoa][:fecha])
          self.cambiar_formato_porcentaje(evaluacion_oferta_academica[:porcentaje].to_s)
          exito_aux = evaluacion_oferta_academica.valid?
          exito = (exito_aux && exito)
          unless exito_aux
            if evaluacion_oferta_academica.errors.invalid?(:tema)
              errores_mensajes << (
              "<b>Tema a evaluar de #{tipo_evaluacion[:nombre].titleize}:</b> no puede estar en blanco")
            end
            if evaluacion_oferta_academica.errors.invalid?(:porcentaje)
              errores_mensajes << (
              "<b>Porcentaje de la evaluaci&oacute;n #{tipo_evaluacion[:nombre].titleize}:</b> es invalido")
            end
            evaluacion oferta academica.errors.clear()

          end
          modelos << evaluacion_oferta_academica
      end
  end
  end
  if exito
    evaluaciones_oferta_academica.each do |eoa|
      porcentaje_total += attr[:evaluaciones_oferta_academica][eoa][:porcentaje].to_f
    end
  end
  if(exito && porcentaje_total == 100)
    self.destroy_all(:materia_codigo => attr[:materia_codigo], :periodo_academico_id => attr[:periodo_academic
    modelos.each { |modelo| modelo.save }
  elsif (exito && porcentaje_total!=100)
    errores_mensajes << "El porcentaje total no puede ser #{porcentaje_total}%"
    exito = false
  end

  {:exito => exito,:retorno => modelos,:errores_mensajes => errores_mensajes}
end
```

Figura 3.22: Método Guardar. Clase EvaluacionOfertaAcademica. Módulo Evaluaciones.

- **Pruebas**

Las pruebas realizadas estuvieron enfocadas a validar la integridad de los valores o datos suministrados por el usuario, sobre todo en la sumatoria total de los porcentajes de las evaluaciones y el período de ejecución de las mismas. Así mismo, se verificó las relaciones entre las evaluaciones para un período y materia en particular.

No.	**Módulo**	**Descripción del Caso de Prueba**	**Resultado Esperado**	**Resultado Obtenido**	**Motivo de la Falla**
1	Evaluaciones	Guardar tipo de evaluaciones sin seleccionar alguna	Muestra una notificación indicando que debe elegir al menos un tipo de evaluación	Se muestra una notificación indicando el mensaje de error	
2	Evaluaciones	Guardar tipo de evaluaciones asociadas a una materia	Muestra la información de los tipos de evaluaciones que están asociados a una materia	Muestra la información de los tipos de evaluaciones que están asociados a una materia	
3	Evaluaciones	Formalizar las evaluaciones dada una materia sin definir primero su información	Notificación de error indicando que los campos de tema y porcentaje deben ser obligatorios	Mensaje de error informando de la situación	
4	Evaluaciones	De la materia con evaluaciones asociadas, agregar el porcentaje, fecha y el contenido a evaluar	Se registra la información que estará asociada por evaluación	Se registra la información que estará asociada por evaluación	
5	Evaluaciones	Validación de los porcentajes asignados a las evaluaciones asociadas a las materias	Se notifica si el porcentaje total es menor o mayor a 100 %	Procede sin problemas la validación correspondiente al porcentaje total asociados al conjunto de evaluaciones	

3.9. Iteración 6

- Planificación

Iteración 6	
Descripción	Desarrollo del Módulo Calificaciones, módulo para la calificación de los estudiantes en dos modos, de manera definitiva o por evaluación (calificación contínua).
Historias de Usuario a Desarrollar	39- Desarrollar método listar materias coordinadas por el docente 40- Desarrollar método que permita la calificación continua o definitiva 41- Desarrollar método que genere listado de estudiantes por materia y por modo de calificación 42- Desarrollar método que permita guardar temporalmente o de manera definitiva la calificación (Modo Definitivo) 43- Desarrollar método que permita mostrar el acumulado de notas del estudiante cuando se está evaluando de forma continua 44- Desarrollar método que permita notificar al estudiante cuando haya sido calificado y actualizada la nota 45- Desarrollar método que permita mostrar la nota definitiva o acumulado del estudiante
Tiempo Estimado	10 días
Fecha Inicio/Fin	05-05-2009 / 15-05-2009

Tareas por Historia de Usuario

HU 39.- Desarrollar método listar materias coordinadas por el docente

- Desarrollar vistas para el listado de materias.
- Listar materias dictadas por el docente en sesión.

HU 40.- Desarrollar método que permita la calificación continua o definitiva

- Análisis de la base de datos.
- Permitir la selección de tipo de calificación.
- Componente para poder calificar tanto continua como definitiva.

HU 41.- Desarrollar método que genere listado de estudiantes por materia y por modo de calificación

- Desarrollar interfaz que permita el listado de estudiantes.
- Desarrollar consultas para el listado.
- Integración de la interfaz con las consultas.

HU 42.- Desarrollar método que permita guardar temporalmente o de manera definitiva la calificación (Modo Definitivo)

- Analizar base de datos.

- Validar calificaciones.
- Guardar temporalmente los dos tipos de calificación.

HU 43.- Desarrollar método que permita mostrar el acumulado de notas del estudiante cuando se está evaluando de forma continua

- Establecer consultas para el acumulado de notas.
- Mostrar las notas acumuladas por estudiante.
- Realizar vistas para poder mostrar las notas de los estudiantes.

HU 44.- Desarrollar método que permita notificar al estudiante cuando haya sido calificado y actualizada la nota

- Diseñar formato de contenido para la notificación.
- Permitir la notificación al estudiante cuando es calificado.

HU 45.- Desarrollar método que permita mostrar la nota definitiva o acumulado del estudiante

- Diseñar vista para mostrar notas definitivas o acumulativas.
- Establecer consultas para mostrar las notas de los estudiantes sea definitiva o acumulativa
- Permitir la integración de las consultas con la vista.

Bitácora de desarrollo

Las bitácoras se realizaron globalizando las historias de usuario de forma general y se clasificaron de la siguiente manera:

No.	Tarea	Precede	Fecha Inicio	Fecha Fin	Días Estimados	Días Realizados
1	Desarrollo de interfaz y Componentes - Mostrar		05/05/2009	05/05/2009	1	1
2	Anlisis y Diseo de la implementación	1	06/05/2009	09/05/2009	4	3
3	Montaje de Interfaz		10/05/2009	12/05/2009	2	2
4	Diseo e implementación de base de datos		13/05/2009	13/05/2009	1	2
5	Validaciones		14/05/2009	15/05/2009	2	1

- **Diseño**

En la Figura 3.23 se visualizan las clases de implementación Ruby que dan solución a las historias de usuario mencionadas en la planificación de la iteración

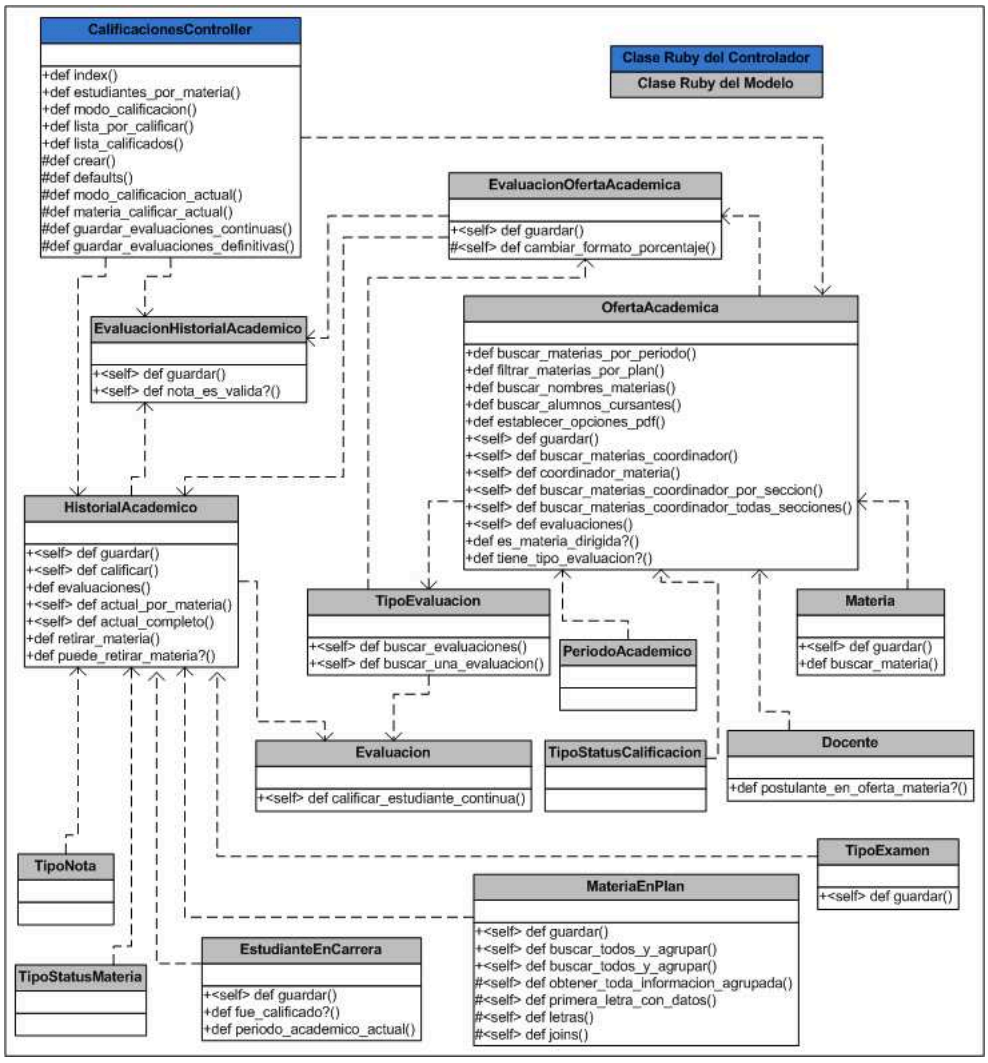

Figura 3.23: Clases de implementación Iteración 6. Módulo Calificaciones.

- **Codificación**

En la Figura 3.24 se presenta la implementación del método `guardar_evaluaciones_continuas` ubicado en la clase `CalificacionesController` el cual se encarga de guardar las notas de cada estudiante por evaluación.

```
def guardar_evaluaciones_continuas
  resultado = EvaluacionHistorialAcademico.guardar(materia_calificar_actual,
                                                   :estudiantes => params[:historial_academico],
                                                   :usuario_actual => usuario_actual,
                                                   :periodo_academico => periodo_academico_actual,
                                                   :docenta_cedula => docente_actual[:docente_cedula] )
  @exito = resultado[:exito]
  @objetos = resultado[:retorno]
  if @objetos.empty?
    @exito = false
    flash[:error] = "Debe evaluar al menos a un estudiante"
  end
  if @exito
    flash[:notice] = "Sus calificaciones fueron efectuadas con exito"
  end
  defaults
end
```

Figura 3.24: Método Guardar Evaluaciones Continuas - Clase CalificacionesController. Módulo Calificación.

En la Figura 3.25 se presenta la implementación del método `guardar_evaluaciones_definitivas` ubicado en la clase `CalificacionesController` el cual se encarga de guardar la nota definitiva de cada estudiante.

```
def guardar_evaluaciones_definitivas
  resultado = HistorialAcademico.calificar(materia_calificar_actual,
                                           :estudiantes => params[:historial_academico],
                                           :usuario_actual => usuario_actual,
                                           :periodo_academico => periodo_academico_actual,
                                           :formalizar_calificacion_activo => params[:formalizar_calificacion_activo] )
  @exito = resultado[:exito]
  @objetos = resultado[:retorno]
  if @objetos.empty?
    @exito = false
    flash[:error] = "Debe evaluar al menos a un estudiante"
  end
  if @exito
    flash[:notice] = "Sus calificaciones fueron efectuadas con exito<br>"
  end
  defaults
end
```

Figura 3.25: Método Guardar Evaluaciones Continuas - Clase CalificacionesController. Módulo Calificación.

Los dos métodos mostrados llaman a la función guardar de la clase `EvaluacionHistorialAcademico` el cual se encarga de recibir las clasificaciones de las evaluaciones de los estudiantes creando los registros y relaciones con el historial del estudiante, y las evaluciones de la materia.

- **Pruebas**

Las pruebas de aceptación con el cliente se realizaron después de culminar la iteración, en ellas surgieron observaciones, más no errores debido a que durante el desarrollo del módulo se pudieron observar y corregir, éstas observaciones fueron canalizadas y corregidas para la modificación de la aplicación. A continuación se muestra las pruebas unitarias que se realizaron.

No.	**Módulo**	**Descripción del Caso de Prueba**	**Resultado Esperado**	**Resultado Obtenido**	**Motivo de la Falla**
1	Calificaciones	Seleccionar materia asociada a un docente	Muestra mensaje de error, debido a que tiene que seleccionar un modo de calificación (continua o definitiva)	Mensaje de error informando sobre la selección del modo de calificación	
2	Calificaciones	Seleccionar materia asociada a un docente con el modo de calificación	Muestra el listado de estudiantes que cursan la materia	Se muestra la lista de estudiantes asociado a la materia seleccionada	
3	Calificaciones	Validar notas correctas	Se indica que debe ingresar notas válidas del 0 al 20 y aprobado (A) aplazado (AP)	Se indica que debe ingresar notas válidas del 0 al 20 y aprobado (A) aplazado (AP)	
4	Calificaciones	Calificar de modo continuo	Muestra la nota de la evaluación y el acumulado en su nota definitiva, se envía un correo electrónico al estudiante con la información de la nota	Muestra la nota de la evaluación y el acumulado en su nota definitiva, no se en'via correo electrónico	Falla en la configuración en correo electrónico
5	Calificaciones	Guardar temporalmente nota definitiva	Se guarda la nota del estudiante temporalmente, en tal caso de que se necesite alguna modificación	Se procede a guardar las calificaciones temporalmente	
6	Calificaciones	Al dar la opción de guardar, al menos debe estar un estudiante calificado	Se notifica que se debe calificar al menos un estudiante, no procede la instrucción	Mensaje de error indicando que debe calificar al menos un estudiante	

3.10. Iteración 7

- Planificación

Iteración 7	
Descripción	Desarrollo del Módulo Oferta Materias, módulo que permite proponer materias para la oferta académica del período en curso, y a la vez postularse al grupo docente en la misma.
Historias de Usuario a Desarrollar	50- Listar todas las materias según las carreras asociadas al docente 51- Mostrar el detalle de la materia seleccionada 52- Proponer una materia del plan de estudio activo 53- Asociar un docente a una materia ya ofertada 54- Ofertar materias dirigidas
Tiempo Estimado	2 días
Fecha Inicio/Fin	20-05-2009 / 22-05-2009

Tareas por Historia de Usuario

HU 50.- Listar todas las materias según las carreras asociadas al docente

- Adaptar el componente javascript, tipo agenda, que muestre el conjunto de materias en plan de estudio asociadas a la carerra relacionadas al docente actual.
- Crear filtros de busqueda con las categorías de materias ya ofertadas y las no ofertadas por otros docentes.
- Desarrollar en el modelo respectivo, un método que reciba como parametro las variables necesarias para poder establecer las condiciones de busqueda.
- Definir las rutas de acuerdo a los filtros de busqueda.

HU 51.- Mostrar el detalle de la materia seleccionada

- Al seleccionar una materia del listado, mostrar la información referente a la misma, contemplando el código y nombre.
- Mostrar el conjunto de docentes ya asociados o hayan ofertado dicha materia.

HU 52.- Proponer una materia del plan de estudio activo.

- Seleccionada una materia, desplegar la opción de proponer la misma.
- Crear en el modelo de datos respectivo, un método que permitar crear la nueva propuesta para oferta académica, con sus respectivas relaciones.
- Asociar el docente a la materia recien propuesta, como opción a formar parte del grupo docente en la misma.

HU 53.- Asociar un docente a una materia ya ofertada.

- Seleccionada una materia ya ofertada o propuesta, mostrar la opción de unirse como alternativa para formar parte del grupo docente en la misma.
- Crear método en el modelo de datos respectivo, que asocie el docente a la materia.

HU 54.- Ofertar materias dirigidas

- Desarrollo, diseño y montaje de interfaz contemplando nombre de la materia que desea dictar, y una breve descripción de la misma.
- Crear un método en el modelo respectivo, que permita crear la nueva materia propuesta
- Asociar la materia como propuesta a la oferta académica

Bitácora de desarrollo

Las bitácoras se realizaron globalizando las historias de usuario de forma general y se clasificaron de la siguiente manera:

No.	Tarea	Precede	Fecha Inicio	Fecha Fin	Días Estimados	Días Realizados
1	Arquitectura		20/05/2009	20/05/2009	1	1
2	Análisis y Diseño de la implementación	1	21/05/2009	21/05/2009	1	0.5
3	Análisis de la base de datos	2	21/05/2009	21/05/2009	1	0.5
4	Permitir la oferta de materias dirigidas	3	22/05/2009	22/05/2009	1	1
5	Establecer relaciones con el modulo de plan docente	4	22/05/2009	22/05/2009	1	1

- **Diseño**

En esta iteración se desarrolló el módulo que permitará a la planta docente, proponer materias para ser consideradas en la planificación docente, de acuerdo a esto en la Figura 3.26 se visualizan las clases de implementación Ruby que dan solución a la *Proponer materias para la planificación Docente*.

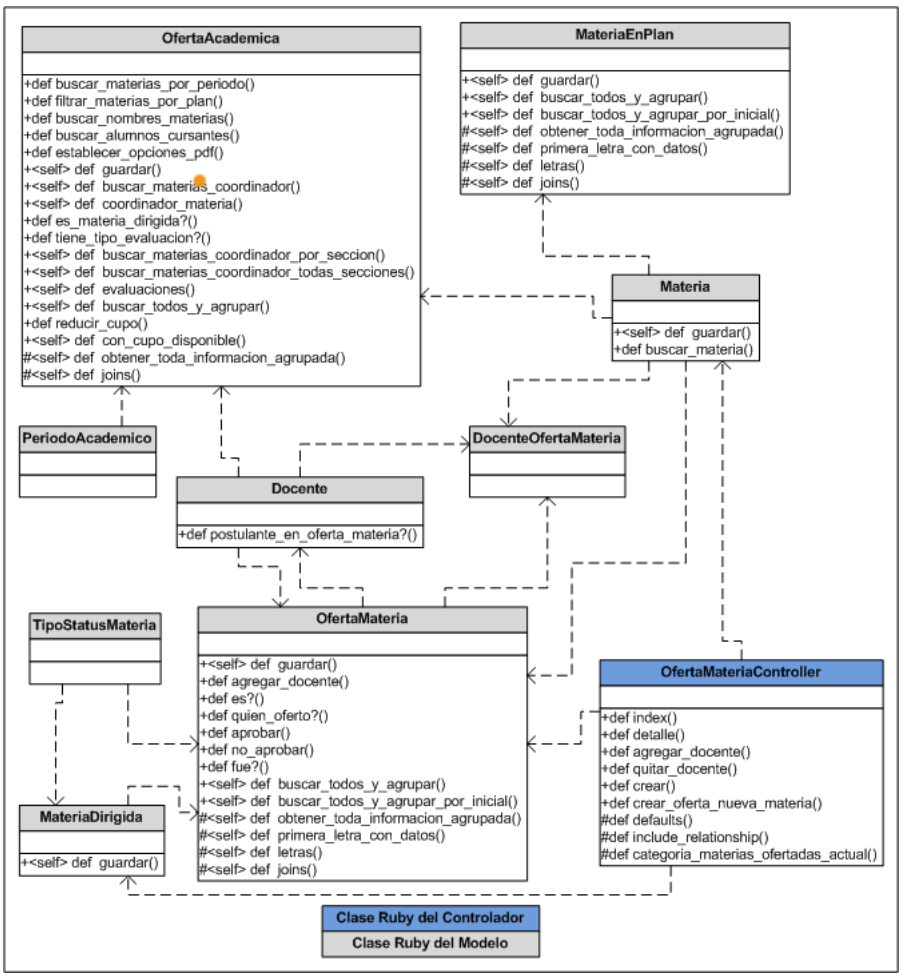

Figura 3.26: Clases de implementación Iteración 8. Módulo Ofertas Materias.

- **Codificación**

En la Figura 3.27, se puede detallar el código implementado para preparar los datos suministrados por el usuario y, así como lo se muestra en la Figura 3.28, realizar sus validaciones en cuanto consistencia e integridad de datos, pudiendo establecer las relaciones entre las entidades.

```
def crear
  atributos = {}
  atributos[:materia_id] = params[:materia_codigo]
  atributos[:materia_type] = Materia.to_s
  atributos[:periodo_academico_id] = periodo_academico_actual
  resultado = OfertaMateria.guardar(atributos,:docente_cedula => docente_actual[:docente_cedula])
  @exito = resultado[:exito]
  @oferta_materia = resultado[:retorno]
  if @exito
    flash[:notice] = "Se ha ofertado la materia <b></b>"
  else
    @materia = Materia.find(params[:materia_codigo])
  end
end
```

Figura 3.27: Método Crear. Clase OfertaMateriaController. Módulo Ofertas Materias.

```
def self.guardar(attr,options={})
  if attr[:materia_type].eql?(Materia.to_s)
      materia = Materia.find_by_codigo(attr[:materia_id])
  else
      materia = MateriaDirigida.find_by_codigo(attr[:materia_id])
  end
  pendiente = TipoStatusOfertaMateria::PENDIENTE
  materia_por_ofertar = new( :materia_id => materia[:codigo],
                             :materia_type => materia.class.to_s,
                             :periodo_academico_id => attr[:periodo_academico_id],
                             :tipo_status_oferta_materia_id => pendiente[:id])
  exito = materia_por_ofertar.valid?
  if exito
    materia_por_ofertar.save
    materia_por_ofertar.agregar_docente(options[:docente_cedula],:oferto => 1)
  end
  {:exito => exito,:retorno => materia_por_ofertar }
end
```

Figura 3.28: Método Guardar. Clase OfertaMateria. Módulo Ofertas Materias.

Así mismo, se muestra en la Figura 3.29, el método que permite crear materias dirigidas, no presentes en el plan activo de la materia.

```
def self.guardar(attr,opciones={})
  materia_dirigida = new(attr)
  exito = materia_dirigida.valid?
  if exito
    materia_dirigida.save
    pendiente = TipoStatusMateriaDirigida::PENDIENTE
    codigo = "DI%04d" % materia_dirigida[:id]
    materia_dirigida.update_attributes(:codigo => codigo)
    atributos = {}
    atributos[:materia_id] = codigo
    atributos[:materia_type] = to_s
    atributos[:periodo_academico_id] = attr[:periodo_academico_id]
    OfertaMateria.guardar(atributos, :docente_cedula => attr[:docente_cedula])
    materia_dirigida
  end
  {:exito => exito,:retorno => [materia_dirigida]}
end
```

Figura 3.29: Método Guardar. Clase MateriaDirigida. Módulo Ofertas Materias.

- **Pruebas**

Para establecer la validez del algoritmo y los resultados, las pruebas realizadas estuvieron basadas en hacer corresponder o verificar las relaciones entre los modelos y su posterior despliegue en el detalle de la materia una vez ofertada. Otras pruebas funcionales hechas fueron en diferenciar entre las materias propuestas por un docente y las materias ya propuestas por otros docentes al momento de mostrar el detalle de las mismas, en donde se debía diferenciar si la presente materia había sido ofertada por el docente actual o por otro, para asociarlo a la propuesta, pero como parte del grupo docente.

No.	Módulo	Descripción del Caso de Prueba	Resultado Esperado	Resultado Obtenido	Motivo de la Falla
1	Oferta Materias	Docente selecciona materia para ofertar	Muestra la información de la materia y se le informa que la puede ofertar	Se muestra por medio de un componente la lista de materias a ofertar	
2	Oferta Materias	Docente oferta materia	Se hace efectivo la oferta de la materia informandole a dicho docente sobre el hecho	Se procede a la oferta de materia, se muestra confirmación para continuar o no la oferta	
3	Oferta Materias	Docente intenta ofertar una materia que ya ha sido ofertada	Notificación indicando que no puede ofertar la materia ya que un docente lo había hecho antes, se le informa de postularse al grupo docente	Notificación negando la oferta, se le ofrece postularse al grupo docente	
4	Oferta Materias	Docente se postula para formar parte del grupo docente	Se efectúa el hecho y aparece en la lista de postulantes	Se le informa sobre la postulación mostrando la lista de docentes postulados	
5	Oferta Materias	Docente se postula en una materia cuando ya lo ha hecho antes	Notificación indicando que ya ha sido postulado y no peude hacerlo más de una vez	Notificación indicando que ya ha sido postulado y no puede hacerlo más de una vez	

3.11. Iteración 8

- Planificación

Iteración 8	
Descripción	Desarrollo del Módulo Planificación Docente, con éste módulo se permitirá aceptar las materias ofertadas para que pertenezcan a la oferta académica del período académico actual, asociar un docente como coordinador de una materia, número de estudiantes que se aceptarán y la sección a la cual pertenecerá.
Historias de Usuario a Desarrollar	55-Listar materias ofertadas por docentes 56- Mostrar el detalle de la materia ofertada 57- Formalizar la oferta de la materia para formar parte de la oferta académica del período en curso 58- Notificar al coordinador que es asignado a una materia
Tiempo Estimado	8 días
Fecha Inicio/Fin	20-05-2009 / 28-05-2009

Tareas por Historia de Usuario

HU 55.- Listar materias ofertadas por docentes

- Adaptar el componente javascript, tipo agenda, que muestre el conjunto de materias ofertadas por los docentes.
- Crear filtros de busqueda con las categorías de materias ofertadas aprobadas, no aprobadas y por revisar.
- Desarrollar en el modelo respectivo, un método que reciba como parametro las variables necesarias para poder establecer las condiciones de busqueda.
- Definir las rutas de acuerdo a los filtros de busqueda.

HU 56.- Mostrar el detalle de la materia ofertada

- Al Seleccionar una materia de listado, buscar, en base de datos, la información referente a esta . Dicha información debe contemplar, nombre, código, si es materia de plan de estudio o si es una materia dirigida, y el conjunto de profesores que la ofertaron.

HU 57.- Formalizar la oferta de la materia para formar parte de la oferta académica del período en curso.

- En el detalle de la materia, crear un formulario, con las opciones de la aceptación o no de la materia, para formar parte de la oferta académica.
- Si la materia es aceptada, mostrar listado de docentes que pueden ser candidatos a coordinador. El listado debe ser mostrado a través de un

autocompletar, considerando nombre, apellido o cédula del docente. Solicitar el número de secciones a abrir y la cantidad de estudiantes por seccion.

- Definir un metodo guardar, en el modelo respectivo, que reciba como parametro la información proveniente del formulario, en donde si la materia es aceptada, incluirla a la oferta académica del período en curso, en caso contrario, cambiar su estatus a no aceptada.

HU 58.- Notificar al coordinador que es asignado a una materia

- Si la materia ofertada es aceptada, enviar al coordinador asignado una notificación a su correo electrónico, en donde se le debe informar, además de que fue asignado como coordinador de la presente materia, que deberá asociar las evaluaciones respectivas.

Bitácora de desarrollo

Las bitácoras se realizaron globalizando las historias de usuario de forma general y se clasificaron de la siguiente manera:

No.	Tarea	Precede	Fecha Inicio	Fecha Fin	Días Estimados	Días Realizados
1	Arquitectura		20/05/2009	23/05/2009	3	3
2	Análisis y Diseño de la implementación	1	24/05/2009	25/05/2009	1	1
3	Montaje de Interfaz	2	25/05/2009	27/05/2009	2	1.5
4	Diseño e implementación de base de datos	3	28/05/2009	28/05/2009	0.5	0.5
5	Validaciones		28/05/2009	28/05/2009	0.5	0.5

- **Diseño**

En esta iteración se desarrollo el módulo que permitirá definir la programación docente en base a las propuestas u ofertas hechas por la planta docente. En la Figura 3.30, se muestran las clases de implementacion Ruby que fueron usadas en la diseño, análisis e implementación del modulo.

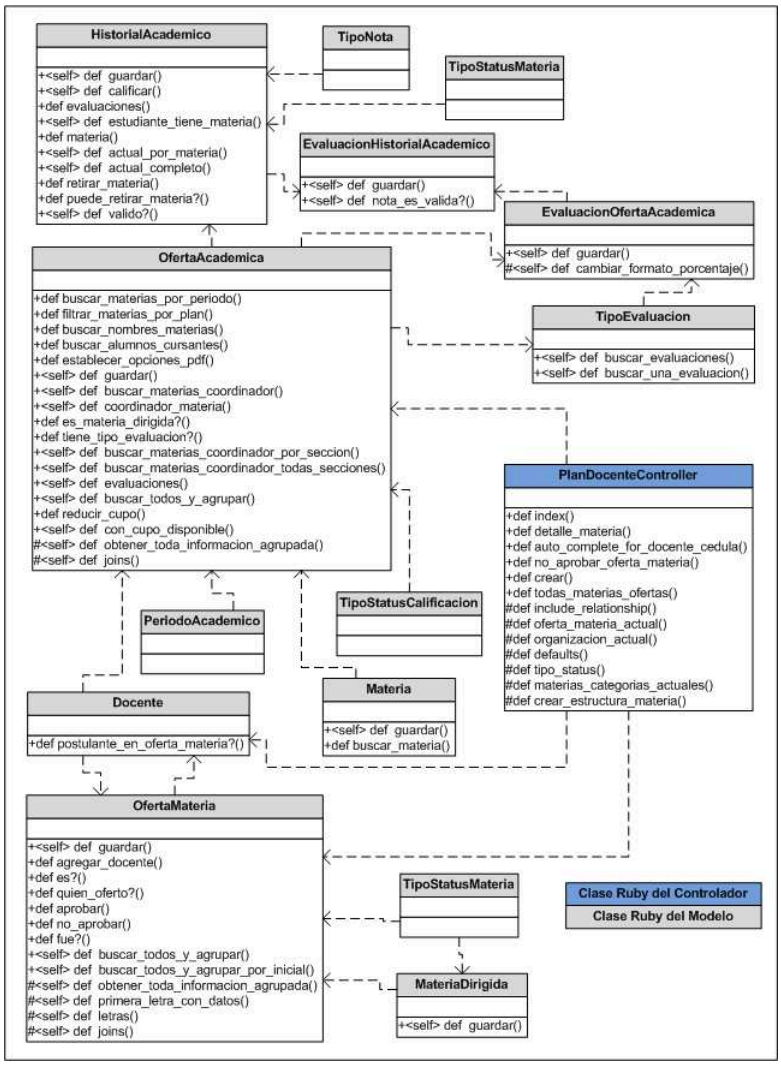

Figura 3.30: Clases de implementación Iteración 9. Módulo Planificación Docente.

- **Codificación**

Luego de la implementación, añalisis y diseño de los métodos y estructuras necesarias que permiten automatizar el proceso de planificación docente, se puede presentar en la Figura 3.31, el código fuente que dio base a la solución.

```
def self.guardar(attr,opciones={})
  modelos = []
  sin_clave = TipoStatusCalificacion.find_by_descripcion("sin clave")
  oferta_acedemica = new( :docente_cedula_coordinador => attr[:docente_cedula_coordinador],
                          :cupo => attr[:cupo],
                          :materia_codigo => attr[:materia_codigo],
                          :nombre_seccion => attr[:nombre_seccion],
                          :periodo_academico_id => attr[:periodo_academico_id],
                          :tipo_status_calificacion_id => sin_clave[:id])
  exito = oferta_acedemica.valid?
  modelos << oferta_acedemica
  if opciones[:registrar_materia_en_plan]
    materia_en_plan = MateriaEnPlan.new(opciones[:materia_en_plan])
    exito = (materia_en_plan.valid? && exito)
  end
  if exito
    if opciones[:registrar_materia]
        materia = Materia.new(opciones[:materia])
        materia.codigo = opciones[:materia][:codigo]
        materia.save
    end
    if opciones[:registrar_materia_en_plan]
      MateriaEnPlan.guardar(opciones[:materia_en_plan])
    end
    modelos.each do |modelo|
      modelo.save
    end
  end
  {:exito => exito,:retorno => modelos}
end
```

Figura 3.31: Método Guardar Oferta Académica - Clase OfertaAcademica. Módulo Planificación Docente.

- Pruebas

Para determinar la validez y correctitud funcional del módulo, se realizaron pruebas que consistieron en verificar la integración del presente módulo con los dos directamente relacionados, el *Módulo de Evaluaciones*, ya descrito previamente, en donde se verificaba si al formalizar la propuesta docente y asignar el coordinador, en el módulo antes mencionado, se reflejaba la materia para el docente asignado, así mismo tal verificacion se realizo con el *Módulo de Inscripciones*, ya que al ser una oferta académica, la materia debía estar presente en el listado de materias para determinados estudiantes acorde a su historial académicio. Con esta pruebas se pudo corroborar y validar la comunicación entre los diversos módulos que compone el sistema.

No.	**Módulo**	**Descripción del Caso de Prueba**	**Resultado Esperado**	**Resultado Obtenido**	**Motivo de la Falla**
1	Planificación Docente	Coordinador selecciona materias de la lista de materia ofertada	Notificación indicando si se acepta o rechaza la oferta de la materia	Se lista las materias ofertadas en un componente notificando al coordinador	
2	Planificación Docente	Coordinador acepta la oferta de la materia	Se le indica ingresar el nombre de la sección que va a pertenecer la materia, la cantidad de alumnos que la verán y ingresar el coordinador de la materia	Se le indica ingresar el nombre de la sección que va a pertenecer la materia, la cantidad de alumnos que la verán y ingresar el coordinador de la materia	
3	Planificación Docente	Coordinador efectua la aceptación de la oferta de la materia	Notificación vía correo electrónico al docente que se asigno como coordinador de la materia sobre el hecho	Se envia correo electrónico al docente indicando que es seleccionado como coordinador de la materia ofertada	
4	Planificación Docente	Coordinador rechaza oferta de la materia	Notificación de oferta rechazada	No se llevo a cabo la notificación	Problemas con configuración de correo electrónico

3.12. Iteración 9

- Planificación

Iteración 9	
Descripción	Desarrollo del Módulo Caja, éste módulo permite consultar a través de servicios web, las transacciones realizadas por caja, con la finalidad de validar pagos realizados por los estudiantes.
Historias de Usuario a Desarrollar	36- Desarrollar servicio web para consultar transacciones 37- Desarrollar método para generar reportes 38- Desarrollar método para registrar una nueva transacción
Tiempo Estimado	5 Días
Fecha Inicio/Fin	01-06-2009 / 06-06-2009

Tareas por Historia de Usuario

HU 36.- Desarrollar servicio web para consultar transacciones

- Investigar los tipos de servicios web existentes.
- Desarrollar un componente que permita la conexión con la caja de la Facultad de Ciencias.
- Desarrollar servicio para validación de transacción.
- Establecer conexión con el sistema de caja de la Facultad de Ciencias.

HU 37.- Desarrollar método para generar reportes

- Analizar los reportes a generar.
- Analizar la base de datos.
- Desarrollar un componente que permita generar reportes de acuerdo a ciertos estados.

HU 38.- Desarrollar método para registrar una transacción nueva

- Desarrollar un componente que permita verificar las nuevas transacciones hechas en caja para poder registrarla.

Bitácora de desarrollo

Las bitácoras se realizaron globalizando las historias de usuario de forma general y se clasificaron de la siguiente manera:

No.	Tarea	Precede	Fecha Inicio	Fecha Fin	Días Estimados	Días Realizados
1	Analisis de la implementación		01/06/2009	01/06/2009	1	1
2	Desarrollo de Interfaz para reportes		02/06/2009	03/06/2009	1	1
3	Análisis de Base de datos		03/06/2009	03/06/2009	0.5	0.5
4	Servicio de consulta de transacciones		04/06/2009	04/06/2009	0.5	0.5
5	Servicio de validación de transacción	4	05/06/2009	05/06/2009	0.5	0.5
6	Servicio de registro de transacción	5	05/06/2009	05/06/2009	0.5	0.5
7	Integración con proyecto CONEST	6	05/06/2009	06/06/2009	1	1
8	Conexión con servidor remoto	7	06/06/2009	06/06/2009	1	0.5

- Diseño

En la Figura 3.32 se visualizan las clases de implementación Ruby que dan solución a las historias de usuario mencionadas en la planificación de la iteración

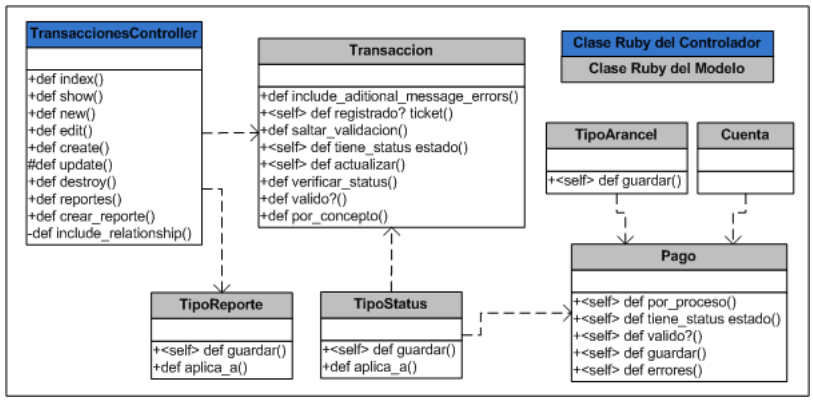

Figura 3.32: Clases de implementación Iteración 5. Módulo Caja.

- Codificación

la Figura 3.33 se presenta la implementación del método `crear_reporte` ubicado en la clase `TransaccionesController`, el cual se encarga de generar el reporte de acuerdo a los estados disponible, por revisar y revisado.

```ruby
def crear_reporte
  @reporte = TipoReporte.find_by_id(params[:tipo_reporte_id])
  if @reporte.nil?
     render :inline => "<h1 style=\"text-align:center;color:red;\">Reporte no encontrado</h1>",:layout => "reportes"
     return
  end
  case @reporte[:nombre]
   when "transacciones_disponibles"
     td = TipoStatus.aplica_a(Transaccion.to_s).find_by_nombre("disponible")
     @transacciones = Transaccion.all(:conditions => {:tipo_status_id => td[:id]})
   when "transacciones_pendientes"
     tp = TipoStatus.aplica_a(Transaccion.to_s).find_by_nombre("por_revisar")
     @transacciones = Transaccion.all(:conditions => {:tipo_status_id => tp[:id]})
   else
     tr = TipoStatus.aplica_a(Transaccion.to_s).find_by_nombre("revisado")
     @transacciones = Transaccion.all(:conditions => {:tipo_status_id => tr[:id]})
  end
  respond_to do |format|
     format.html  {render :action => "reportes/html/#{@reporte[:nombre]}",:layout => "reportes" }
     format.pdf  do
       pdf = render_to_pdf({:action => "reportes/pdf/#{@reporte[:nombre]}.rpdf",:layout => "reportes"})
       send_data  pdf, :filename => "reporte_#{@reporte[:nombre].downcase}.pdf", :type => "application/pdf"
     end
  end
end
```

Figura 3.33: Método Crear Reporte - Clase TransaccionesController. Módulo Caja.

En la Figura 3.34 se presenta la implementación del método `actualizar` ubicado en la clase `Transaccion`, el cual, a traves de una una tarea programada, comprueba las transacciones o pagos llevados a cabo en la caja, verificando y actualizando con los valores de las transacciones llevadas a cabo a traves del sistema.

```ruby
def self.actualizar(attr,options={})
    transaccion = self.registrado?(attr[:numero_ticket])
    if transaccion
      nueva_transaccion = self.registrado?(options[:nuevo_numero_ticket])
      if nueva_transaccion
        if nueva_transaccion.tine_status? "disponible" or
          (nueva_transaccion.tine_status? "pendiente" and nueva_transaccion[:pago_id].zero?)
          nueva_transaccion.update_attribute(:pago_id, transaccion[:pago_id])
        else
         nueva_transaccion.errors.add(:numero_ticket,"Ya se encuentra en uso")
        end
      else
        nueva_transaccion = new(:numero_ticket => options[:numero_ticket],:pago_id => transaccion[:pago_id])
      end
    else
      transaccion.errors.add(:numero_ticket,"No se encuentra registrado")
    end
end
```

Figura 3.34: Método Crear Reporte - Clase TransaccionesController. Módulo Caja.

- **Pruebas**

Las pruebas de aceptación con el cliente se realizaron después de culminar la iteración, en ellas surgieron observaciones, más no errores debido a que durante el desarrollo del módulo se pudieron observar y corregir, éstas observaciones fueron

canalizadas y corregidas para la modificación de la aplicación. A continuación se muestra las pruebas unitarias que se realizaron.

No.	**Módulo**	**Descripción del Caso de Prueba**	**Resultado Esperado**	**Resultado Obtenido**	**Motivo de la Falla**
1	Caja	Conexión servicio web al sistema de caja	Se conecta exitosamente	Se establece la conexión sin problemas	
2	Caja	Generación de reportes de acuerdo a estados existentes	Se crean reportes con los estados existentes	Permite las distintas generaciones de reporte de acuerdo a los estados existentes previsto para ello	

3.13. Iteración 10

- Planificación

Iteración 10	
Descripción	Integrar cada uno de los módulos desarrollados en un único sistema.
Tiempo Estimado	11 Días
Fecha Inicio/Fin	07-06-2009 / 18-06-2009

Bitácora de desarrollo

Las bitácoras se realizaron globalizando las historias de usuario de forma general y se clasificaron de la siguiente manera:

No.	Tarea	Precede	Fecha Inicio	Fecha Fin	Días Estimados	Días Realizados
1	Integrar módulo Admisión		07/06/2009	08/06/2009	1	1
2	Integrar módulo Selección		08/06/2009	09/06/2009	1	1
3	Integrar módulo Inscripción		09/06/2009	10/06/2009	1	1
4	Integrar módulo Solicitudes		10/06/2009	11/06/2009	1	1
5	Integrar módulo Ofertar Materias		11/06/2009	12/06/2009	1	1
6	Integrar módulo Planificación Docente		12/06/2009	13/06/2009	1	1
7	Integrar módulo Evaluaciones		13/06/2009	13/06/2009	0.5	0.5
8	Integrar módulo Calificaciones		13/06/2009	14/06/2009	0.5	0.5
9	Integrar módulo Caja		14/06/2009	14/06/2009	1	0.5
10	Pruebas de funcionalidad	1 al 9	14/06/2009	18/06/2009	4	4

- Diseño y Codificación

En estas etapas no se contemplo diseño de modelos y desarrollo de codificación, debido a que esta iteración solo se definió para la integración de cada módulo desarrollado y verificación su plena funcionalidad.

- **Pruebas**

No.	**Módulo**	**Descripción del Caso de Prueba**	**Resultado Esperado**	**Resultado Obtenido**	**Motivo de la Falla**
1	Ninguno	Integracón de módulos relacionado con el módulo de Caja	Se establece la integración exitosamente en funcionamiento con el servicio web	No se establece el servicio web	Falto configuración del servicio web
2	Ninguno	Integración de todos los módulos	Se integra exitosamente	Algunos módulos dieron fallas de integración	Falta de código desarrollado que no se integró
3	Ninguno	Prueba de funcionalidad de todos los módulos con casos prueba	Prueba realizada con éxito	Se realizo las distintas pruebas con casos prueba sin inconvenientes	

3.14. Resumen del capítulo

En este capítulo, se presentó una adaptación del proceso de desarrollo de software *Programación Extrema* (XP) al caso particular de estudio, para el desarrollo del sistema propuesto. Para tales efectos se describió el contexto de desarrollo, plan de iteración, breve descripción de la problemática presentada y cada una de las iteraciones involucradas en el desarrollo, donde se detalla cada una de las fases de la metodología de desarrollo, para dar solución a la situación actual que presenta la Coordinación de Postgrado de la Facultad de Ciencias, referente a los procesos que conllevan la gestión académica en dicha organización.

Capítulo 4

Conclusiones

El objetivo del presente Trabajo Especial de Grado fue el desarrollo de un sistema enmarcado y correlacionado con el sistema CONEST, el cual apoyará a la gestión académico-administrativa de los procesos internos llevados a cabo en la Cordinación de Postgrado de la mencionada Facultad, el cual fue desarrollado en el lenguaje de programción Ruby sobre el framework de aplicaciones Ruby on Rails. Tras la culminación de la fase de análisis, desarrollo y documentación, a continuación en el presente capítulo se presentarán las conclusiones, resultados, recomendaciones y aportes de la investigación.

4.1. Conclusiones

Al culminar el presente Trabajo Especial de Grado, los objetivos generales propuestos al inicio de la investigación fueron alcanzados con éxito, más sin embargo el objetivo relacionado a la comunicación con el sistema que gestiona los movimientos y transacciones en la caja de Facultad de Ciencias, propio de la Coordinación de Administración, debido a que los involucrados en la administración de dicho sistema, no cumplieron con lo prometido en las discusiones y acuerdos, en donde se habilitaría un servidor dedicado, con acceso al sistema que se desarrolló, en el cual se alojarían copias de las transacciones hechas en la caja de la Facultad, registrados en un archivo texto plano en formato csv, sin embargo el módulo encargado de la gestión y comunicación con el sistema fue desarrollado completamente, en base a las especificaciones. Cabe destacar que dicha razón antes referenciada fué ajena a nuestro interés y alcance y no fué factor negativo para el completo desarrollo del sistema y de los módulos que los componen, estando totalmente operable.

Por otro lado, el análisis de la gestión de procesos académicos llevados en la Coordinación

de Postgrados en conjunto a los trece (13) postgrados de la Facultad de Ciencias, y su posterior modelación en el sistema, fue un proceso complejo. Esto debido a que la mayor parte de las actividades que se llevan a cabo dentro de la coordinación son manuales y en su mayoría, para acelerar los procesos, la redundancía de datos era una opción ineludible, sin embargo con un cuidadoso estudio y apoyo del personal administrativo que laboran tanto en la Coordinación de Postgrado, como en sus trece postgrados, se logró el objetivo planteado.

Utilizar el proceso de desarrollo agil *Programación Extrema* (XP), permitió que la aplicación fuese sencilla de adaptar a los requerimientos y trabajar de manera iterativa e incremental en cuanto al desarrollo e implementación. Dividir el objetivo general del problema en pequeñas partes o iteraciones, permite que la atención se centre en cumplir con la mismas, y luego componer el todo. Integrar a los usuarios finales durante las fase de pruebas, permite la validación de las implementaciones de las historias de usuarios.

La integración de las nuevas librerias o frameworks de desarrollo del lado cliente, como fué el caso de *JQuery*, ofreciendo funcionalidades en javascript, simplificó la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos y el desarrollo de animaciones, enfocando la aplicación a las nuevas tendencias de la web. Con esta integración, se busca la acelerar, agilizar simplificar el desarrollo del lado cliente de una manera mucho más sencilla así mismo simplifica mucho el desarrollo de aplicaciones basadas en AJAX. Sin embargo esta herramienta, además de simplicar la labor del programador, impacta notablemente en el desempeño en tiempo de una aplicación, factor a considerar en cualquier desarrollo orientado a la Web.

Finalmente, el uso de *Ruby on Rails*, como tecnología del lado del servidor agilizó y simplificó de manera notable el proceso de implementación, ya que al apoyarnos de las métodos encapsulados, api's y estructuras de datos que este posee, minimiza en tiempo y esfuerzo la fase de codificación, de depuración y pruebas [4]. El estilo de desarrollo por capas como está basado el lenguaje facilitará a posteriores desarrolladores el entendimiento y estructuración de la aplicación, lo que permite la modularidad y escalabilidad del mismo.

4.2. Resultados

Las pruebas funcionales realizadas en conjunto con el cliente y la parte involucrada con todo lo referente a los procesos académicos de la coordinación, permitió la comprobación del flujo normal de sus actividades, así mismo la corrección oportuna de errores suscitados durante la ejecución de las mencionadas pruebas. Como resultado de dichas pruebas, se pudo evidenciar, no solo la correcta fluidez de los procesos, sino como la automatización de los mismo mejora notable y significativamente su ejecución, en cuanto a tiempo, reducción

de errores humanos y redundancia del manejo de información.

Por otro lado, en todo sistema orientado a la Web, además de las pruebas funcionales, es importamente, como buena práctica, la realización de pruebas de rendimiento, que permitan dar un visión general de los tiempos de respuesta de la aplicación, a fín de determinar el comportamiento en situaciones particulares de trabajo, considerando como variable el número de usuarios concurrentes que realizan un número específico de transacciones. [10]

Para tales efectos, se definieron dos casos de estudio para la realización de las pruebas, cuyas características más resaltantes son la diversidad de procedimientos que realizan y siendo los de mayor demanda, en cuanto a peticiones concurrentes refiere en la aplicación. Los casos de estudio seleccionados son: *Registro de Aspirantes* e *Inscripción de Estudiantes*.

- Contexto

  La unidad de medida usada fue *milisegundo* (ms). El servidor Web empleado fue *Mongrel* y los cálculos de rendimiento Web se realizaron por la herramienta, *HttPerf* de *HPLabs* [11]

  A continuación una breve descricpión de los casos de estudio seleccionados:

  - **Registro de aspirantes:** Forma parte del conjunto de subprocesos del Proceso de Admisión, donde se registra los datos personales y los referentes al postgrado por el cual opta un aspirante, así mismo la validación del pago del arancel, donde tal verificación se hace mediante un servicio Web, usando *REST* y la clase `ActiveResource` de rails y finalmente con el envio al aspirante de una notificación vía correo electrónico, usando la clase `ActionMailer` de rails, informando que se ha registrado una solicitud de admisión a un postgrado.
  - **Inscripción de Estudiantes:** Forma parte del conjunto de subprocesos del Proceso de Inscripción, donde se determinan las materias que un estudiante en particular puede inscribir, así mismo la validación y registro del pago del arancel, que al igual que el caso anterior, se hace mediante un servicio Web, usando *REST* y `ActiveResource`.

  La prueba de rendimiento consistió en realizar 10 iteraciones de 300 peticiones en cada una, de esta manera permitirá evaluar la variación de los valores obtenidos en el tiempo. Esta prueba sera aplicada para cada caso de estudio.

- Representación

A continuación se muestran los resultados obtenidos tras la ejecución de las pruebas:

- **Caso Registro de Aspirantes**: En el cuadro 4.1, se observan los resultados obtenidos tras aplicar la prueba de rendimiento al caso de estudio relacionado al registro de aspirantes.

N	TIEMPO(ms)	MINIMO(ms)	MAXIMO(ms)	MEDIANA(ms)	DESV. ESTANDAR(ms)
1	681596	276,6	5458,1	2740,5	1211,1
2	607974	277,7	5775,6	2214,5	1177,9
3	659633	280,9	5788,5	2706,5	1139,9
4	574644	272,1	4118,4	2194,5	1014,9
5	681596	276,6	5458,1	2740,5	1211,1
6	560022	277,0	4632,4	2190,5	855,5
7	650844	276,3	4731,3	2711,5	1189,8
8	609412	279,4	5285,4	2249,5	1100,5
9	521136	280,2	6189,7	2171,5	943,9
10	542815	283,2	5240,5	2191,5	918,1

Cuadro 4.1: Resultados de rendimiento Web para el caso Registro de Aspirante. 300 peticiones/iteración

En la Figura 4.1, se observa la representación gráfica de los resultados obtenidos en el Cuadro 4.1.

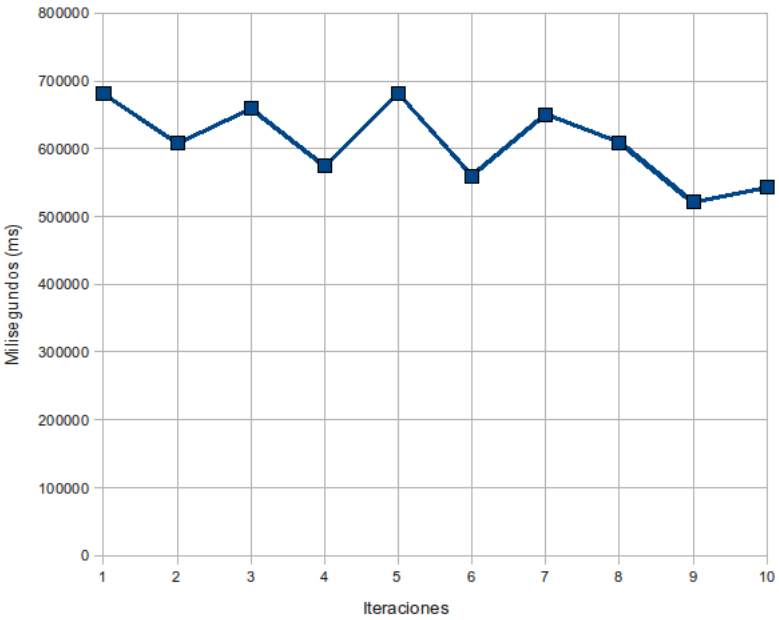

Figura 4.1: Gráfico de resultados del rendimiento Web. Caso Registro de Aspirantes.

- **Inscripción de Estudiantes**: En el cuadro 4.2, se observan los resultados obtenidos tras aplicar la prueba de rendimiento al caso de estudio relacionado al proceso de inscripción de estudiantes.

N	TIEMPO(ms)	MINIMO(ms)	MAXIMO(ms)	MEDIANA(ms)	DESV. ESTANDAR(ms)
1	270620	134,6	5042,8	253,5	2029,3
2	172787	106,3	2425,2	299,5	521,5
3	65963	280,9	5788,5	270,65	1139,9
4	76546	115,1	1036,4	202,5	172,9
5	61283	114,8	714,1	197,5	73,6
6	58711	115,1	985,7	194,5	70,7
7	65084	276,3	4731,3	271,15	1189,8
8	58832	114,0	517,7	201,5	37,8
9	57738	117,3	472,0	197,5	45,2
10	59667	117,7	620,5	199,5	52,1

Cuadro 4.2: Resultados de rendimiento Web para el caso Inscripción de Estudiantes. 300 peticiones/iteración.

En la Figura 4.2, se observa la representación gráfica de los resultados obtenidos en el Cuadro 4.2.

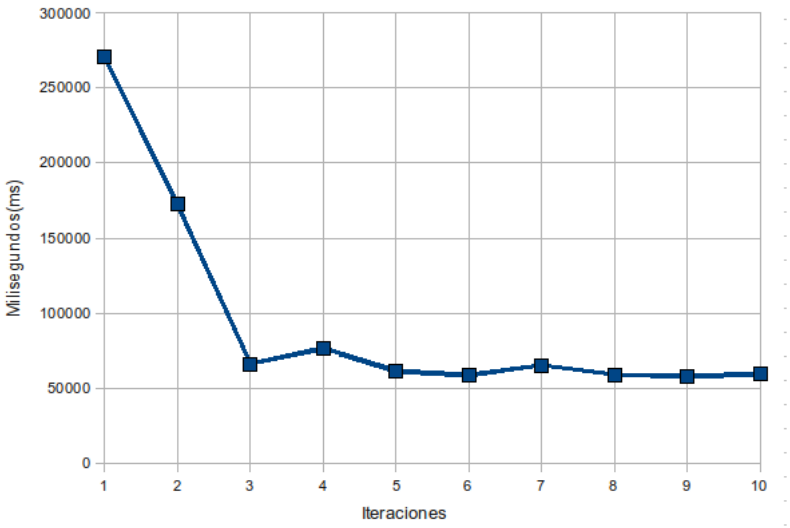

Figura 4.2: Gráfico de resultados del rendimiento Web. Caso Inscripción de Estudiantes.

- Análisis

  A continuación se analizarán los resultados obtenidos en cada uno de los experimentos efectuados.

  - **Caso Registro de Aspirantes**: de acuerdo a el cuadro 4.1 y su gráfica respectiva, se puede interpretar lo siguiente:
    - El tiempo máximo de peticion-respuesta de una solicitud oscila entre los 6 a 7 segundos con alta carga de solicitudes al servidor, tiempo considerable si se atribuye el hecho en que cada petición al servidor, involucra una doble conexión a un servidor remoto para la comprobación y registro de las transacciones, producto del pago del arancel.
    - El envío de correos electrónicos ocupa el 75 % del tiempo de respuesta tras una conexión o transacción.
    - En condiciones de carga normal o moderada, el tiempo de petición respuesta oscila entre 1 a 2 segundos, tiempo esperado considerando la carga y condiciones de la red.
    - En promedio, el tiempo de conexión para una solicitud será aproximadamente de 2.5 segundos.
    - Los tiempos generales oscilan cerca de un mismo valor, por lo que el uso de cache mejoraría el rendimiento de la aplicación, debido a que las peticiones se realizarían sobre recursos ya solicitados, sin embargo esta opción no es viable si los recursos varían constantemente y el *overhead* o carga adicional agregada para revisar la misma, igualaría los tiempos en el caso que no se usara bajo la condición antes mencionada, que para el caso actual, ha de considerarse puesto que los recursos usados para el registro de una aspirante varían constantemente.
  - **Inscripción de Estudiantes**: de acuerdo a el cuadro 4.2 y su gráfica respectiva, se puede interpretar lo siguiente:
    - Para las primeras iteraciones, los tiempos llegaron a ser considerablemente altos, efecto que iba en decremento a medida que aumentaban las iteraciones, debido a que para esta ocasión, para agilizar el rendimiento de la aplicación y considerando las variables presentes en este caso de estudio como estudiantes y carreras, se usó el almacenamiento de recurso en *caché* mediante la clase `ActiveSupport::Cache::MemoryStore` [12], que provee el framework de Rails. El sistema de cache fue usado para disminuir las consultas a la base de datos, almecenando en dicho sistema los resultados obtenidos tras la verificación del conjunto de materias disponibles para un determinado estudiante, consultas que se hacían constantemente a medida que se selccionaban las materias a inscribir.

  - A medida que las trasancciones se almacenaban en cache los tiempos van disminuyendo hasta llegar a un punto donde estos oscilaban bajo un mismo valor, estabilizándo la diferencia entre los tiempos máximos y mínimos, en comparación a las primeras iteraciones.

- Resumen

  Tras la ejecución de las pruebas de rendimiento y en base de los resultados obtenidos, la aplicación mantiene tiempos de conexión relativamente constante y bajas. Así mismo el uso de técnicas como la *caché*, mejora notablemente el redimiento de una aplicación, considerando las condiciones y variables bajo el cual se desee implementar la misma.

4.3. Recomendaciones

A fin de dar continuidad al desarrollo de este proyecto, se recomienda tomar en cuenta los siguientes aspectos:

- El proyecto está separado en dos partes *front-end* y *back-end*, siendo la primera parte la encargada de interactuar con el o los usuarios, mientras que la última procesa la entrada desde el *frontend*. Para tales efectos, al expandir las funcionalidades del sistema, es recomendable discriminar los nuevos módulos en base a esta estructura.
- La definición de rutas en la aplicación, por cada petición HTTP que se vaya a llevar a cabo, permitirá mantener una legibilidad, control sobre la misma, además ser adaptable e integrable a futuros cambios.
- Mantener el uso de las librerias desarrolladas para controlar los accesos y modos de seguridad en el sistema.
- Seguir los lineamientos o estándares de programación y estructuración a nivel de base de datos y de la aplicación.
- Mejorar los aspectos relacionados a la interfaz y el feedback con el usuario.
- Adaptar los componentes desarrollados con el framework de javascript, Prototype, a las nuevas tendencias como JQuery.
- Sobre el uso de XP, sería recomendable que en casos donde los requerimientos y la solución no esten bien definidas, combinar *XP* con *Modelación Ágil*.

4.4. Aporte a la Investigación

Entre los aportes que deja la investigación tenemos:

- La investigacion permitió contextualizar y puntualizar la problemática de los procesos manejados en la Coordinación de Postgrado y alternativas que dan solución a los mismos.
- Cada uno de los módulos que compone el sistema, puede ser usado de manera independiente y de fácil integración por otras aplicaciones, al igual que los componentes *Javascript* desarrollados para este proyecto.
- Permitió al apoyo y crecimiento del Sistema de la División de Estudios de la Facultad de Ciencias, CONEST.
- En aspecto generales, este proyecto será la base para los próximos desarrolladores, tesistas o pasantes que deseen apoyar a su crecimiento y mejoras del mismo.

Bibliografía

[1] Coordinación de Postgrado de la Facultad de Ciencias de la Universidad Central de Venezuela, Manual de Organización, Venezuela, Junio 2005.

[2] Zambrano, J. *Gestion de los procesos Administrativos de la docencia de la UCV soportado por las TICś. Escuela de Computación. Facultad de Ciencias. Universidad Central de Venezuela. Seminario de la Maestría en Ciencias de la Computación.* Venezuela 2009

[3] Boyer, Y. y Méndez, N., *Automatización de procesos relacionados con las solicitudes estudiantiles y actividades administrativas y de docencia de la Facultad de Ciencias, T.E.G.*, Venezuela, Mayo 2008.

[4] González, Z., *Medición y Visualización de Métricas de Software para Frameworks Web. Caso de Estudio: Ruby on Rails y Frameworks en Java. Escuela de Computación. Facultad de Ciencias. Universidad Central de Venezuela, T.E.G.*, Venezuela 2007.

[5] Dávila, Y. y Gesia, F., *Desarrollo de una Aplicación WEB Cliente-Servidor para la Automatización de los Procesos relacionados con los Actos Académicos de Pregrado de la División de Control de Estudios de la Facultad de Ciencias, T.E.G.*, Venezuela, Julio 2007.

[6] Newkirk, J. y Martin, R., *La Programación Extrema en la práctica, Addison-Wesley Iberoamericana Espanya, S.A.*, 2002.

[7] http://www.extremeprogramming.org, *Extreme Programming, XP.*, Febrero 2006.

[8] Ambler, S., http://www.agilemodeling.com/essays/agileModelingXP.htm, *Agile Modeling and eXtreme Programming (XP), Ambysoft Inc.*, 2007.

[9] Bibeault, B. y Katz, Y., jQuery in Action, Manning Publications CO. Segunda Edicion. Estados Unidos, Junio 2009.

[10] Cueva, J., *Métricas de Usabilidad en la Web* España, 2004.

[11] Mosberger, D. y Jin, T., *httperfm: A Tool for Measuring Web Server Performance* Estados Unidos, Diciembre 1998.

[12] http://api.rubyonrails.org/classes/ActiveSupport/Cache/MemoryStore.html

[13] Haya, M. Franch, X., y Mayo, E., *Uso de los Diagramas de Actividades UML y el Lenguaje i* en el Modelado del Proceso de Implantación del Balanced Scorecard* España, Mayo 2001.

Anexos

A continuación, se muestra parte de las capturas de pantalla correspondiente a los diferentes módulos que constituyen el sistema anteriormente descrito.

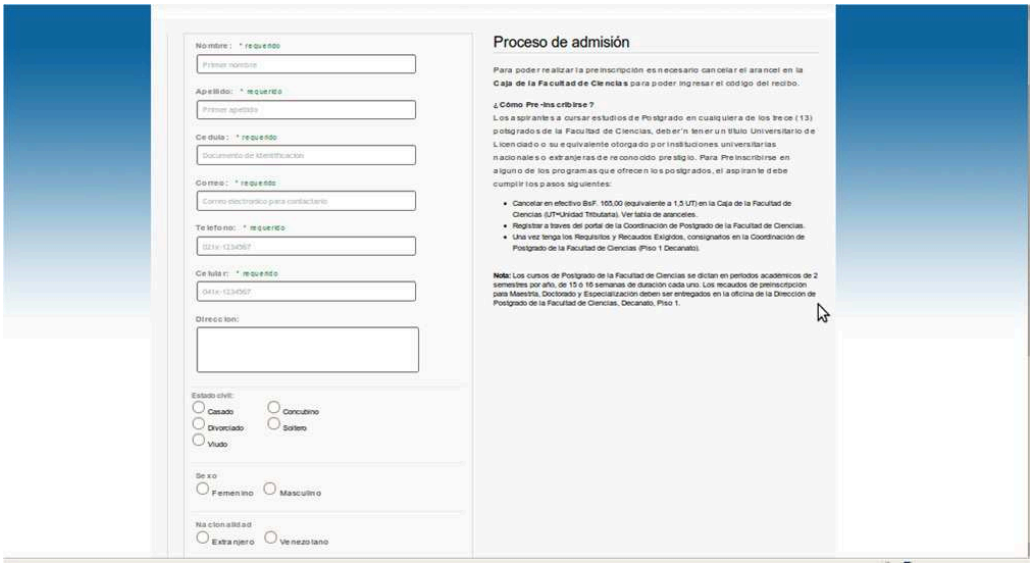

Figura 4.3: Aplicación - Módulo Admisión.

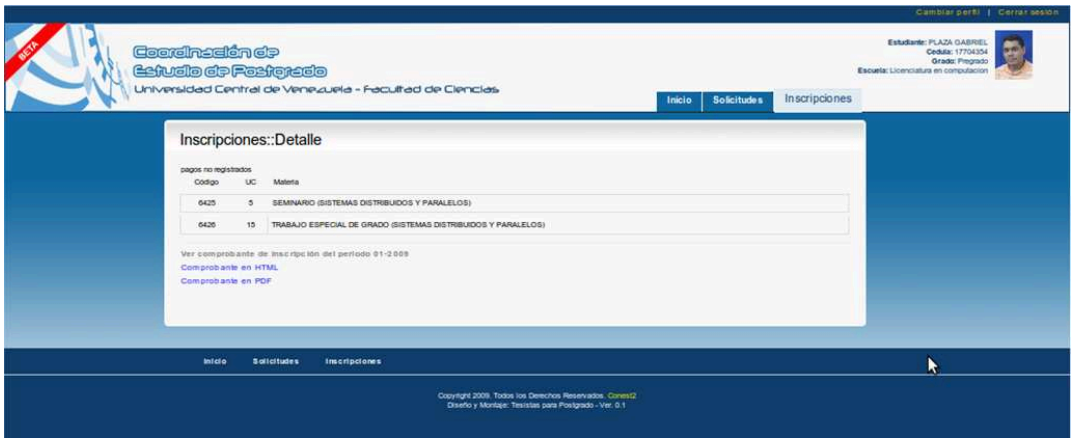

Figura 4.4: Aplicación - Módulo Inscripción.

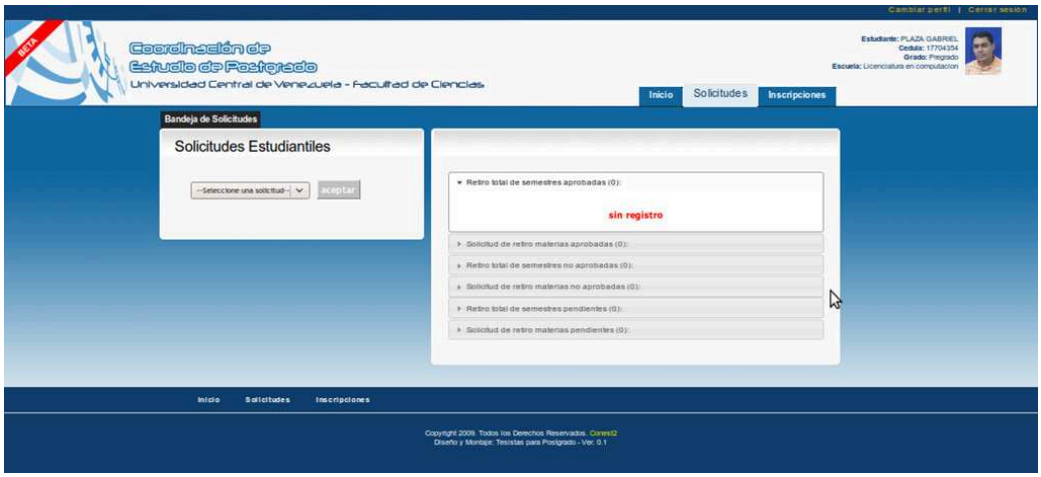

Figura 4.5: Aplicación - Módulo Solicitudes Estudiantiles.

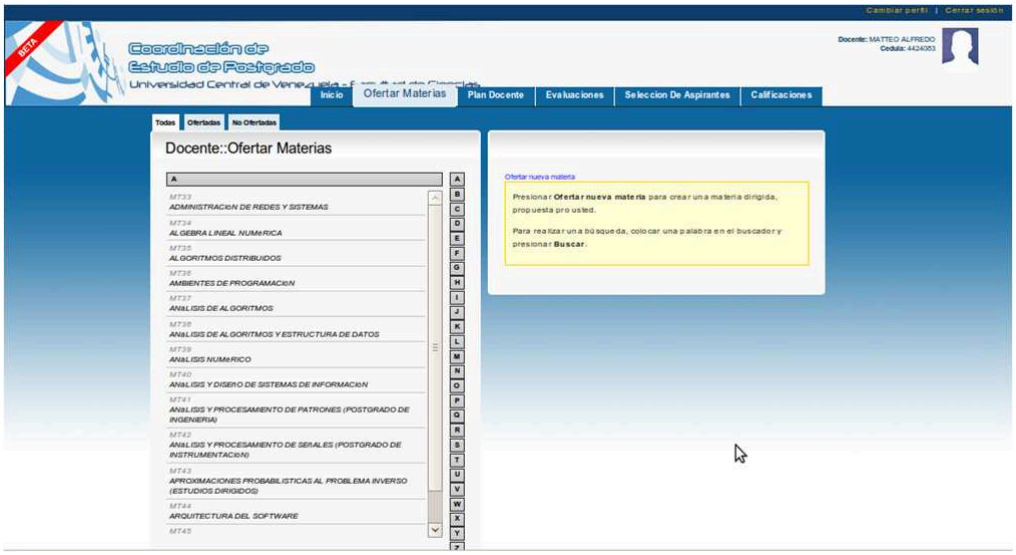

Figura 4.6: Aplicación - Módulo Ofertar Materias.

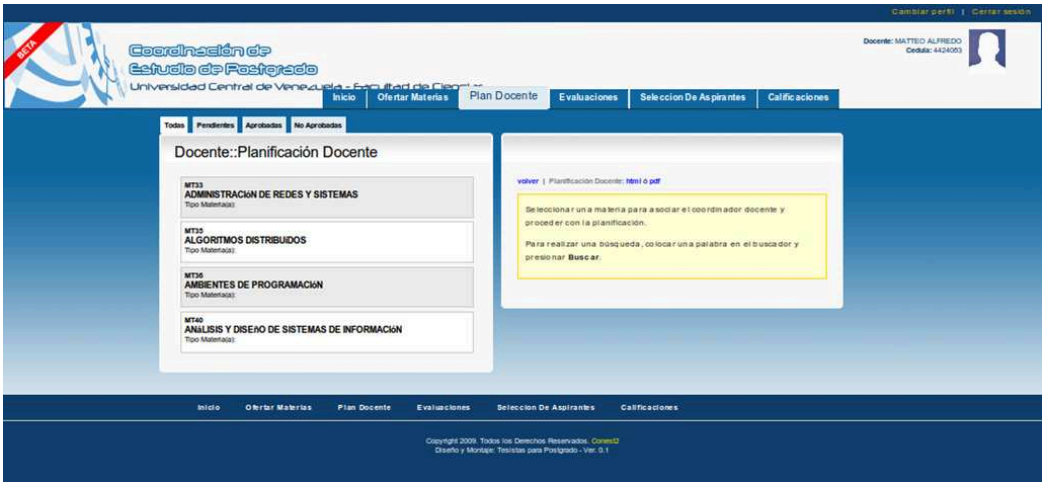

Figura 4.7: Aplicación - Módulo Planificación Docente.

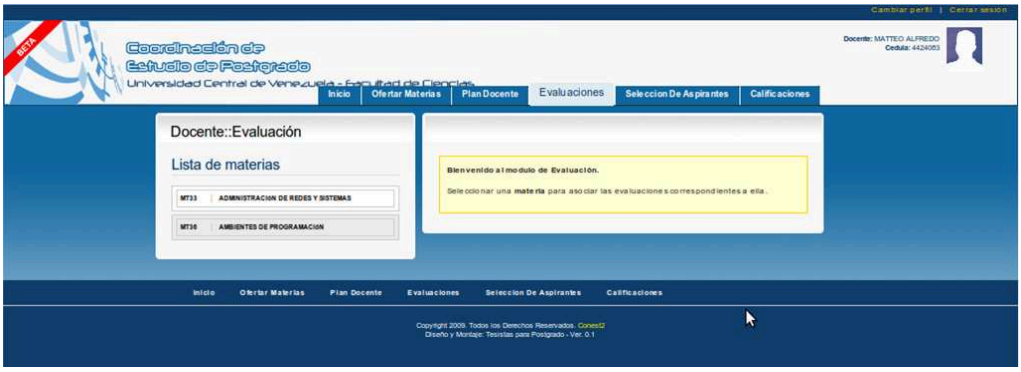

Figura 4.8: Aplicación - Módulo Evaluaciones.

Figura 4.9: Aplicación - Módulo Calificación (Continua y Definitiva).

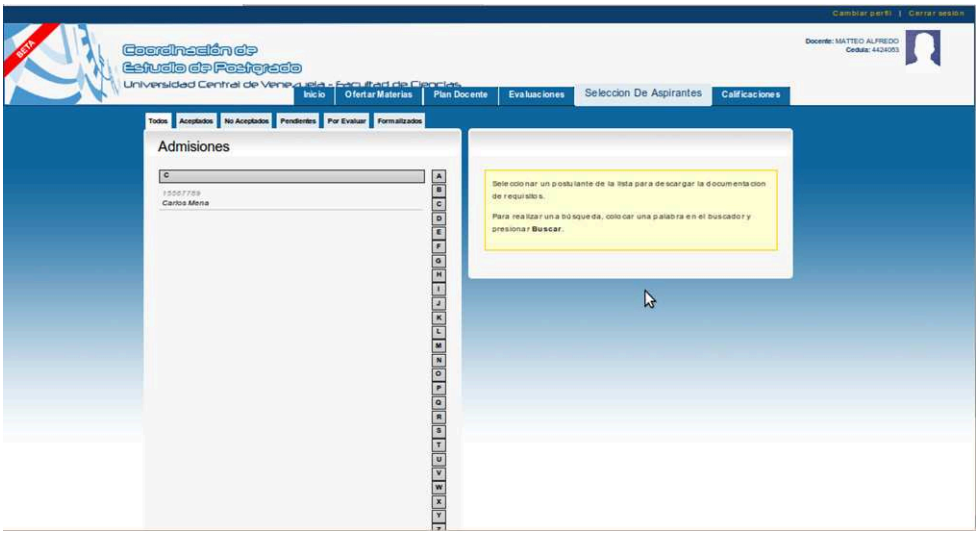

Figura 4.10: Aplicación - Módulo Selección.