UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
LABORATORIO DE SISTEMAS DE INFORMACIÓN Y BASES DE DATOS

SISTEMA DE INFORMACIÓN PARA EL MANEJO DE CORRESPONDENCIA Y PUNTOS DE CUENTA

Trabajo Especial de Grado presentado ante la Ilustre
Universidad Central de Venezuela
por los bachilleres:

Ruiz Duim, Oscar Enrique C.I: 14.988.951
Corredor Calderas, Neldy Esmeralda C.I: 13.385.271

para optar al título de Licenciado en Computación

Tutor Académico: Rangel, Wuilfredo

Caracas, Mayo de 2009

ACTA

Quienes suscriben, miembros del Jurado designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por los Bachilleres Oscar Enrique Ruiz Duim C.I. 14.988.951 y Neldy Esmeralda Corredor Calderas C.I. 13.385.271, con el título: **"Sistema de Información para el manejo de Correspondencia y Puntos de Cuenta"**, a los fines de optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue el presente trabajo por cada uno de los miembros del jurado, se fijó el día 6 de Mayo de 2009 a las 12:15 PM, para que sus autores lo defendieran en forma pública, lo que hicieron en la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondieron 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, a 6 los días del mes de Mayo del año dos mil nueve, dejándose también constancia de que actúo como Coordinador del Jurado el Tutor Wuilfredo Rangel.

Prof. Wuilfredo Rangel
Tutor Académico

Prof. Esmeralda Ramos
Jurado

Prof. Eleonora Acosta
Jurado

AGRADECIMIENTOS Y DEDICATORIA

Oscar Ruiz

Antes que todo, manifiesto un profundo agradecimiento a Dios, sin el cual, nada nos sería posible realizar.

El presente Trabajo Especial de Grado, está dedicado a mi madre Margarita Duim, quien ha velado por mí, desde mi nacimiento, y en particular, me ha dado todo el apoyo posible para llegar hasta aquí.

Gracias a mi abuelo, Pedro Duim, por haberme dado una buena educación en mi niñez, y haber llenado el espacio de un Padre biológico que nunca tuve.

Agradezco también, a la Sra. Dilia Calderas por estar siempre abierta a ayudarnos en el proceso de desarrollo del presente trabajo, y en especial, por haberme atendido como un familiar más.

Neldy Corredor

Agradezco a Dios por permitirme alcanzar esta meta, y por todas las cosas buenas que me ha dado durante toda mi vida.

Dedico este trabajo a mi adorada madre por ese gran esfuerzo y espíritu luchador que tiene para brindarme lo mejor en la vida, por sus constantes consejos, por darme una mano amiga cuando más lo necesitaba y por el infinito amor que me ofrece; y a mi padre por ser de gran ayuda en los momentos difíciles.

Agradezco a toda mi familia, en especial a mis primas Sara Otero y Milagros Otero por ser grandes profesionales y buenos ejemplos a seguir; a mi tía Sara Calderas por sus valiosos consejos y apoyo; a mi querida abuela Amelia Corredor que aunque no está en este mundo se que me guía y me protege; y sobre todo a mi querido tío Pascual Corredor por darme fuerza y apoyarme cuando más lo requería.

Gracias a Oralis García y a Ronald López por el apoyo y colaboración que me brindaron en el desarrollo de este proyecto.

Oscar Duim y Neldy Corredor

Agradecemos la colaboración de nuestro Tutor, Prof. Wuilfredo Rangel, por acompañarnos en el desarrollo de este Trabajo Especial de Grado.

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Trabajo Especial de Grado

Título: Sistema de Información para el Manejo de Correspondencia y Puntos de Cuenta.

Autor(es): Oscar Enrique Ruiz Duim y Neldy Esmeralda Corredor Calderas.

Unidad de Investigación: Laboratorio de Sistemas de Información y Bases de Datos.

Palabras Clave: Sistemas de Información Transaccional, *Workflow*, Correspondencia, Punto de Cuenta, *XP*, MVC.

Tutor(es): Wuilfredo Rangel
Fecha de Presentación: 6 de Mayo de 2009

Resumen

El objetivo del presente Trabajo Especial de Grado (TEG) es desarrollar un Sistema de Información Transaccional que permita el manejo de correspondencia de la presidencia y el seguimiento de puntos de cuenta aprobados en Junta Directiva del BANAVIH. El desarrollo de este sistema se apoya en las siguientes metodologías y herramientas: *XP* (*Extreme Programming*), los patrones MVC y *Singleton*, y el sistema de control de versiones Subversion. El enfoque *XP* permitió al grupo de desarrollo acelerar la velocidad de construcción de la aplicación, así como también, se involucró al representante del Banco en la elaboración del sistema, por lo que tomó confianza en el producto desde la primera iteración. Por otro lado, a través de la implementación del patrón MVC, se intercambiaron los roles de trabajo entre iteraciones; de esta forma, los miembros del equipo de trabajo intervinieron en cada una de las capas de la arquitectura MVC, por lo tanto, cada uno pudo obtener una compresión integral de toda la estructura de la aplicación. El sistema desarrollado permitirá disminuir los tiempos requeridos para la realización de las tareas relacionadas con el manejo de la correspondencia y de los puntos de cuenta del BANAVIH.

ÍNDICE GENERAL

Pág.

REFERENCIAS BIBLIOGRÁFICAS 146

ÍNDICE DE TABLAS Y FIGURAS

Figuras

Tablas

INTRODUCCIÓN

Cada empresa debe ordenar su información para solucionar y administrar eficientemente los procesos que maneje la organización. En tal sentido, se hace necesario que los datos sean estructurados en Sistemas de Información.

Un Sistema de Información "…puede ser cualquier combinación organizada de personas, hardware, software, redes de comunicación, y recursos de datos que recolectan, transforman, y diseminan información en una organización" (O'Brien, James A., 2001, pp. 7).

Los Sistemas de Información permiten entender el entorno, los objetivos, manejar los procesos eficientemente, con la finalidad de facilitar el desenvolvimiento en los negocios, proporcionando ventajas competitivas a las empresas.

Entre los diferentes tipos de Sistemas de Información más importantes se encuentran: Sistemas de Procesamiento de transacciones (*TPS, Transaction Processing Systems*), Sistemas de Trabajo del Conocimiento (*KWS, Knowledge Work Systems*), Sistemas de Información Gerencial (*MIS, Management Information Systems*) y Sistemas de Apoyo a la Toma de Decisiones (*DSS, Decision Support Systems*).

Cuando un Sistema de Información computarizado es creado con el objetivo de "…procesar grandes cantidades de datos relacionadas con transacciones rutinarias de negocios, como las nóminas y los inventarios" (Kendall & Kendall, 2005, pp.2), se está hablando de un **Sistema de Procesamiento de Transacciones.** Sin embargo, muchas veces no basta sólo con automatizar el procesamiento de grandes volúmenes de transacciones; se requiere también asegurar que las mismas se manejen bajo una determinada secuencia de acciones, ejecutadas por

determinadas personas, en un plazo establecido, y bajo ciertas reglas de negocio. Es en este escenario es donde cobra importancia seguir un enfoque de "**Gestión de *WorkFlow***".

En este sentido, el presente caso de estudio, "Sistema de Información para el Manejo de Correspondencia y Puntos de Cuenta", se basa en una Aplicación Web dentro de una Intranet empresarial que debe manejar un número determinado de transacciones, siguiendo una secuencia de acciones manipuladas por distintos tipos de usuarios (roles), los cuales van agregando información relacionada a dos tipos de documentos: correspondencias y puntos de cuenta.

La motivación de este Trabajo Especial de Grado (TEG) es desarrollar un prototipo de sistema de información que pueda automatizar, mejorar e integrar el manejo de los procesos de correspondencia y seguimiento de puntos de cuenta de la presidencia del BANAVIH, mejorando los tiempos de respuesta a requerimientos específicos, así como también, apoyar en posibles rediseños de los procesos anteriormente nombrados.

Cabe destacar, que como antecedentes a este Trabajo Especial de Grado, se tiene al estudio de factibilidad para la automatización o no de los procesos de correspondencia y seguimiento de puntos de cuenta, y por otro lado, la realización de un estudio de la situación actual de los mismos, cuyo resumen se esboza en el capitulo 1 del presente trabajo.

Para lograr cumplir con la motivación, se trazan una serie de objetivos, los cuales pueden resumirse como sigue: diseñar un modelo de datos adecuado a los requerimientos, seleccionar y adaptar un proceso de desarrollo que se ajuste a las necesidades del equipo desarrollador, seguir mejores practicas por medio de patrones de software que garanticen la mantenibilidad de la aplicación, y por último, desarrollar y probar un prototipo de aplicación que cumpla con los requerimientos planteados.

Para tener una idea general de la estructura que tendrá el Trabajo Especial de Grado, se da a conocer una pequeña descripción de los capítulos incluidos en este documento, los cuales son los siguientes:

CAPÍTULO 1: Trata de los aspectos generales que llevaron a la realización del estudio e implantación del referido sistema, tales como: Situación Actual del Banco Nacional de la Vivienda y Hábitat (BANAVIH) y Problemas que presenta; Objetivo General y Específicos; Límites y Alcances que tendrá el desarrollo del sistema; Plataforma Tecnológica a utilizar para implementar la solución, justificación del porqué será aplicada la respectiva metodología en la Institución involucrada y la importancia del desarrollo del proyecto.

CAPÍTULO 2: Se enfoca en el marco conceptual que se utiliza para desarrollar los tópicos que apoyan el desarrollo del sistema, tales como: Sistema de Información, *Workflow*, Programación Extrema (*XP*), Aplicaciones con Tecnología Web, Patrón MVC y *Singleton*, y Subversión, con la finalidad de dar solución al sistema especificado en el Capítulo 1.

CAPÍTULO 3: Este capítulo se enfoca en la aplicación de la metodología, herramientas y plataforma tecnológica utilizada en el desarrollo del Sistema de Información para el Manejo de los Procesos de Correspondencia y Puntos de Cuenta del BANAVIH, el cual se basa en siete iteraciones bajo el enfoque de desarrollo *XP*.

Posteriormente se darán una serie de conclusiones que resaltan los resultados, los aportes y recomendaciones que se emiten con el fin de mejorar el sistema diseñado. Luego referencias bibliográficas que dan un conocimiento más amplio de todo el sistema.

CAPÍTULO I: Problema de Investigación

1.1 Situación Actual

El Banco Nacional de la Vivienda y Hábitat (BANAVIH) es "… un ente de naturaleza financiera, con personalidad jurídica, patrimonio propio, distinto e independiente del Fisco Nacional, autonomía organizativa, funcional y financiera, adscrito al Ministerio del Poder Popular con competencia en materia de vivienda y hábitat" (BANAVIH, (s.f.)).

La visión y misión de BANAVIH es la siguiente:

"Visión

Ser la institución financiera con profunda vocación social que satisfaga las necesidades de vivienda y hábitat dignos de la familia venezolana" (BANAVIH, (s.f.)).

"Misión:

Ejercer una excelente gestión de los recursos humanos, financieros y tecnológicos asociados a los planes de vivienda, con óptima calidad, ética profesional y participación protagónica de la comunidad, a fin de satisfacer las necesidades de vivienda y hábitat de la familia venezolana" (BANAVIH, (s.f.)).

La Presidencia del BANAVIH cuenta con un sistema para el manejo de correspondencias que ingresan al Banco, el cual registra, modifica, consulta y muestra reportes de las mismas. Sin embargo, aún existe información y procedimientos asociados que se realizan manualmente.

Entre la información y acciones que no están automatizadas en el sistema se especifican las siguientes:

- Clasificación de correspondencias por tipo.

- Clasificación de correspondencias por expedientes.
- Clasificación de expedientes por tipo.
- Consultas y reportes de correspondencias por: asunto, rango de fecha, organismo, monto, entidad federal y estatus.
- Toda la información del seguimiento de los puntos de cuenta aprobados por la Junta Directiva. Un Punto de Cuenta refiere un tema o problemática que se trata en reuniones de Junta Directiva, por tanto, es necesario verificar constantemente el status del mismo.

Por otro lado, según estudios realizados, este sistema presenta cuellos de botella en cuanto al tiempo de respuesta en las consultas y reportes, pudiendo llegar, en el peor de los casos, a 20 minutos. Esto trae como consecuencia retraso en el envío de correspondencias, desde la Presidencia hasta las distintas gerencias del Banco.

También puede apreciarse, que no existe un mecanismo de control del acceso a la información confidencial, puesto que cualquier usuario del sistema de correspondencia puede consultar; por ejemplo, instrucciones del Presidente para una problemática en particular, lo cual no debe ocurrir. Aunado a esto, tampoco existe seguimiento para las acciones del usuario dentro del sistema. Con respecto al seguimiento de los puntos de cuenta, cuando la duración de la respuesta de un Gerente con respecto al status de un punto de cuenta es mayor a tres días, el Secretario de Junta Directiva es obligado a utilizar a la Secretaria y un Mensajero, para enviar un comunicado a la gerencia respectiva, teniendo estos últimos, acceso a información que sólo debería conocer el Presidente y sus asesores.

En otro orden de ideas, el modelo de datos del sistema genera diferentes códigos correlativos para la entrada y salida de una misma correspondencia y, en algunos casos, duplicados de la misma.

A continuación se presenta en la figura 1.1 el diagrama de actividad del "*Workflow* correspondiente al manejo de correspondencias de la gerencia de Presidencia".

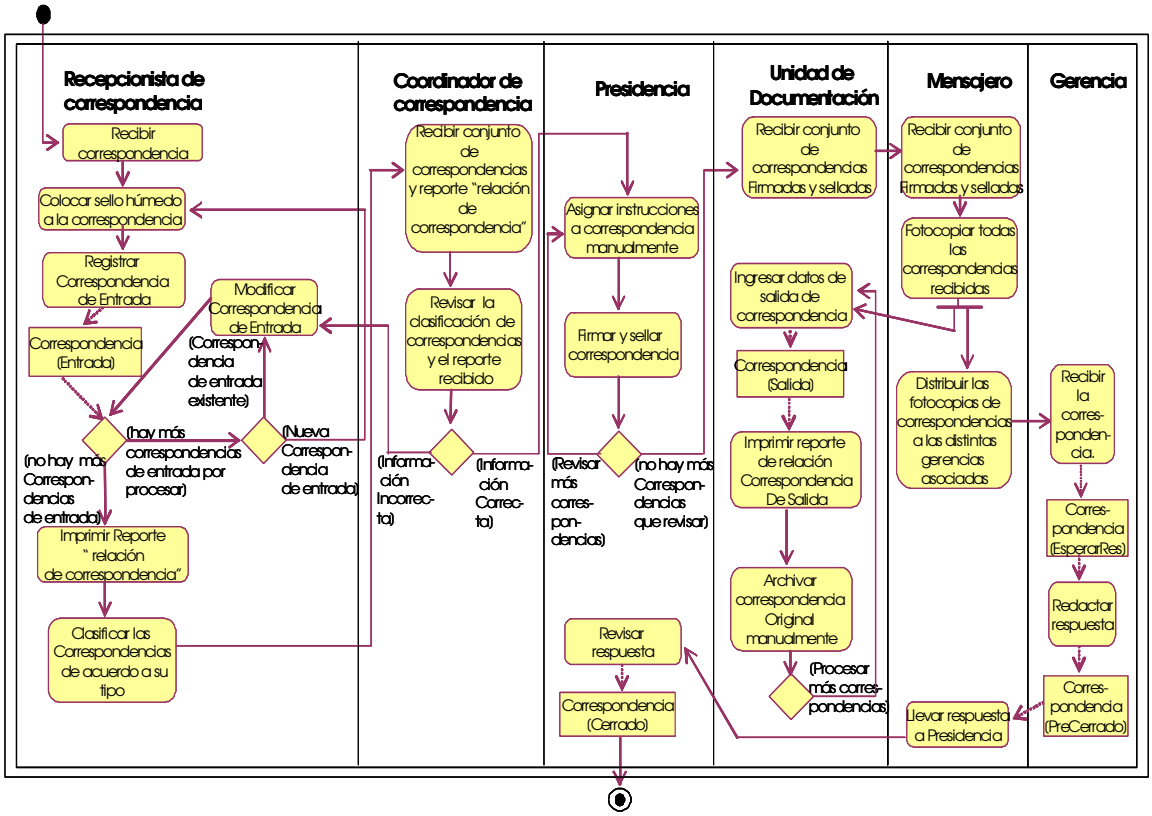

Figura 1.1. Diagrama de Actividad del Workflow correspondiente a la entrega y recepción de correspondencia de la gerencia de Presidencia (Fuente: Elaboración propia).

El diagrama de actividad del *workflow* de la figura 1.1 funciona como sigue: llega una correspondencia al banco (todas las correspondencias provienen de entes adscritos al mismo) y la recepcionista la recibe y le coloca sello húmedo de la institución. Procede entonces a registrar la misma en el sistema, lo que se conoce como darle entrada a la correspondencia. Para el registro, la recepcionista debe leer el asunto e inferir a partir del mismo una clasificación (materia), por ejemplo, "adquisición de inmuebles", "compra de equipos de tecnología", etc.

Busca entonces, en un archivo Excel, el código asociado a la materia que acaba de inferir, y lo introduce en el sistema, para que éste arroje la clasificación. Luego introduce el número del Documento que aparece en el físico y el origen del mismo. Así hará con todas las correspondencias que lleguen al Banco. Este procedimiento, se hace generalmente, dos veces al día (un bloque en la mañana y otro en la tarde). Por cada bloque de correspondencias, debe imprimir el reporte de "relación de correspondencias", que dice cuales han sido las correspondencias a las que se le a dado entrada. Finalmente, procede a ordenar las correspondencias (en físico) por materia y se las entrega al Coordinador de correspondencia, junto con el "reporte de relación de correspondencia".

El coordinador de correspondencia, revisa la clasificación de cada una de las correspondencias recibidas. En caso de haber error de clasificación, las remite de nuevo a la recepcionista para que realice las correcciones pertinentes. Luego de asegurarse de que el proceso de clasificación por materias es correcto, lleva el bloque al Presidente del Banco.

El Presidente, revisa cada una de las correspondencias (generalmente por el asunto), y asigna manualmente las instrucciones, así como también, coloca el sello y la firma, para que de esta forma, sea una correspondencia oficial dentro del banco. Entonces, este bloque es pasado a la Unidad de Documentación, la cual es la encargada de darle "salida" a las correspondencias. Esto consiste principalmente, en asignar dentro del sistema, las instrucciones del Presidente y la gerencia hacia donde va dirigida cada una de las correspondencias, y en caso de ser necesario, asignarlas a un expediente en particular. Pero, antes de darle salida al bloque de correspondencias, el mensajero debe fotocopiar cada

una de ellas, para poder distribuirlas a las distintas gerencias asociadas (la gerencia destino de cada correspondencia forma parte de las instrucciones del Presidente). Una vez que el mensajero haya fotocopiado las correspondencias, la unidad de documentación puede empezar a darles salida. Después de esto, se imprime el reporte de "relación de correspondencia de salida", que análogamente al de entrada, dice cuales fueron las correspondencias que se les ha dado salida en ese bloque. Finalmente, la Unidad de Documentación, procede archivar manualmente las correspondencias en físico en carpetas destinadas para ello.

Se dijo que el mensajero distribuye las fotocopias de las correspondencias a las distintas gerencias, estas la reciben y deben dar una respuesta a la presidencia. Esta respuesta suele darse en varios días. La gerencia, redacta la respuesta en memorando, y la envía a través de un mensajero al Presidente, quien revisa la respuesta. Nótese, los cuellos de botella asociados a:

- La clasificación de correspondencia al momento de la entrada.
- El tiempo de espera de la unidad de documentación para que el mensajero fotocopie las correspondencias, y luego si, comenzar a darles salida.
- El traslado de los mensajeros llevando las fotocopias de las correspondencias de salida a las gerencias, y las respuestas a presidencia.

El siguiente diagrama de actividad de la figura 1.2 representa el *Workflow* "Del Seguimiento de Puntos de Cuenta".

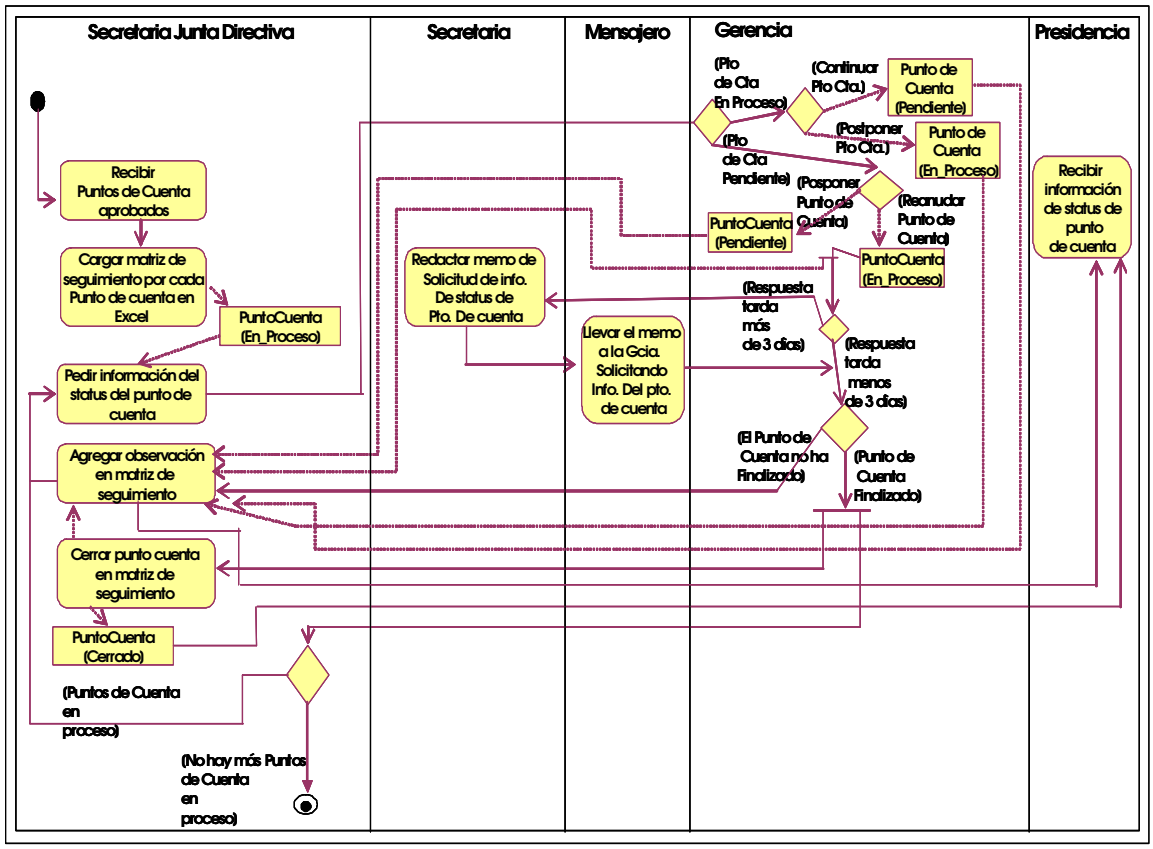

Figura 1.2. Diagrama de Actividad del Workflow del Seguimiento de Puntos de Cuenta

(Fuente: Elaboración propia).

El diagrama de actividad del *workflow* de la figura 1.2 funciona básicamente de la siguiente manera: el Secretario de la Junta Directiva recibe sólo los puntos de cuenta aprobados; carga en Excel lo que se conoce como matriz de seguimiento, la cual se compone de todos los puntos de cuenta, previamente aprobados por la Junta Directiva. En esta matriz se registra la información del punto de cuenta, como por ejemplo, la gerencia y el responsable de su procesamiento o la sesión (número de reunión en que se aprobó). Posteriormente, comienza un ciclo finito, en donde el Secretario debe pedir el estatus de los puntos de cuenta a las distintas gerencias asociadas a través de llamadas telefónicas o las menos veces por correo electrónico. Generalmente, las respuestas tardan varios

días. Cuando el tiempo de respuesta de una gerencia, es superior a los tres días, el Secretario se ve en la necesidad de utilizar a la secretaria y un mensajero para redactar y llevar un memorando respectivamente, siendo esto una especie de amonestación informal que obliga a la gerencia a dar respuesta a la brevedad posible. Cabe destacar, que un momento determinado, un punto de cuenta puede ser postergado momentáneamente, por ejemplo, un punto de cuenta relacionado con la compra de equipos es congelado por una gerencia por falta de presupuesto. Luego posteriormente, puede ser reanudado dicho punto de cuenta cuando la gerencia cuente con el presupuesto. Esto, debe siempre ser notificado al secretario de junta directiva, pero no siempre las gerencias lo hacen, o no por lo menos, con la rapidez que amerita.

Ocurre muchas veces, que un gerente, da informaciones al Secretario (recordemos que estas observaciones siempre son a través de llamadas telefónicas o las menos veces por correo), de las cuales se retracta o desmiente mas adelante, lo cual el Secretario no tiene forma de comprobar. Por ejemplo, un gerente hace la observación por vía telefónica al Secretario, informando que un requerimiento en particular será finalizado esta semana, posteriormente en la fecha pautada el secretario pide la confirmación al gerente de la finalización del punto de cuenta, pero la gerencia le contesta que no será para esa semana sino para otra, alegando que el Secretario a mal interpretado las observaciones del gerente, pero no hay forma de comprobar (al menos por vía telefónica) quien tiene la razón. El workflow finaliza cuando todos los puntos de cuenta cargados en la matriz de seguimiento en Excel, han sido cerrados, es decir, los requerimientos de Junta Directiva se han llevado a cabo por la gerencias respectivas.

El cuello de botella observado, es básicamente el tiempo de respuesta del estatus de puntos de cuenta de las gerencias hacia la secretaria de junta directiva. Además, cuando el secretario utiliza personal para la redacción y traslado del memorando para la solicitud formal del estatus de puntos de cuenta, se esta revelando a estas personas, y posiblemente estas lo revelen a otras mas adelante, información estrictamente confidencia dentro del banco, lo cual es muy grave que suceda. Finalmente, no existe un medio formal, a través del cual, queden plasmadas las observaciones conversadas entre el secretario y las gerencias del banco, pudiendo ocasionar errores de interpretación mas adelante.

1.2 Planteamiento del Problema

Basados en la situación actual, se han logrado identificar los siguientes problemas y áreas de oportunidad del Sistema de Correspondencia de Presidencia y del manejo no automatizado del proceso del Seguimiento de Puntos de Cuenta:

- Tiempo inadecuado en la clasificación de correspondencias y expedientes, ya que se realiza de forma manual, siguiendo criterios de un archivo Excel.
- Tiempo de respuesta inadecuado en las consultas y reportes.
- Falta de criterios (parámetros) importantes en las consultas.
- Duplicados de correspondencias en el sistema.
- Imposibilidad de actualizar información en el sistema concerniente a la denominación de los entes adscritos al Banco (Organismos), así

como también, otros ítems importantes: documentos, estatus, materias y prioridad.

- Personal innecesario en los procesos de seguimiento de puntos de cuenta y manejo de correspondencias.
- Acceso no autorizado a información confidencial del Banco.
- Ralentización del seguimiento manual de los puntos de cuenta.

Con base en los problemas planteados, se establecen en la tabla 1.1 los siguientes criterios para medir el nivel de cumplimiento / resolución de estos problemas.

Criterios del Problema	Descripción
Falta de parámetros de consulta de correspondencias.	Criterios de búsqueda que no incluye el sistema actualmente. Estos son: consultas/reportes por asunto, rango de fecha, organismo, monto, entidad federal y estatus.
Cantidad de Correspondencia Duplicada.	Cantidad de correspondencias duplicadas por día cuando se registran. Actualmente la probabilidad de duplicar una correspondencia es de 100%.
Cantidad de denominaciones de entes adscritos (organismos) y otros ítems no actualizados en las opciones del sistema.	Número de ítems de organismos y otros ítems no ingresados o actualizados en el sistema. Actualmente existen cinco de estos ítems.
Cantidad de Personal que participan en los procesos de	Número de personas que intervienen en el manejo de la

manejo de correspondencia y seguimiento de un Punto de Cuenta.	correspondencia. Participan seis personas: La recepcionista, el coordinador de correspondencia y cuatro personas de la Unidad de Documentación. Número de personas que participan en el proceso de seguimiento de Puntos de Cuenta. Intervienen 4 personas: El Secretario de Junta Directiva y su asesor, que comunican al Presidente. Un mensajero y la Secretaria para la comunicación con las distintas gerencias en caso necesario.
Cantidad de accesos no autorizados a la información de correspondencia y puntos de cuenta	Número de veces que la información es accedida por personal no autorizado. Aunque no hay mediciones estrictas del caso, cabe destacar dos puntos: • De los seis usuarios que manejan la correspondencia dentro del sistema, solo cuatro de ellos deberían tener acceso a inf. de salida de correspondencia. Sin embargo, por no existir manejo de roles, los seis usuarios pueden consultar esta información en cualquier momento.

	• Cuando la duración de la respuesta de un Gerente con respecto al status de un punto de cuenta es mayor a tres días, el Secretario de Junta Directiva se ve en la necesidad de utilizar a la Secretaria y a un Mensajero para enviar un Memorándum a la gerencia respectiva, teniendo estos, acceso a información estrictamente confidencial.
Cantidad de Tiempo para el manejo de un Punto de Cuenta.	Cantidad de tiempo asociado desde que el Presidente requiere información de status de un Punto de Cuenta hasta que recibe la respuesta. Actualmente el tiempo promedio aprox. de este feedback es de 2 días.

Tabla 1.1. Criterios del problema y su descripción (Fuente: Elaboración propia).

Cabe destacar, que la medición de los tiempos de la descripción de los criterios de la tabla 1.1, se calculó en base a un promedio de unas veinte ocurrencias durante la etapa de estudio de la situación actual.

1.3 Objetivo General

Desarrollar un Sistema de Información Transaccional que permita el manejo de correspondencia de la presidencia y el seguimiento de puntos de cuenta aprobados en Junta Directiva del BANAVIH.

1.4 Objetivos Específicos

- Diseñar un modelo de datos que permita representar y almacenar la información acerca del seguimiento de los puntos de cuenta y el manejo de correspondencia del BANAVIH.
- Adaptar el proceso *XP* para el desarrollo de un sistema de información, para la gestión de los procesos de correspondencia y puntos de cuenta.
- Seguir estándares de calidad en el desarrollo de software, tales como los patrones MVC y *Singleton,* para las actividades de codificación del proceso *XP*, a fin de que el Sistema de Información sea mantenible y extensible en el tiempo.
- Desarrollar un prototipo de aplicación, y verificar mediante una prueba piloto contra la data del Banco, su correcto funcionamiento de acuerdo a los requerimientos planteados.

1.5 Límites y Alcances

- Planificación, Diseño, Codificación y Pruebas Unitarias de los módulos de correspondencia, puntos de cuenta y administración para el sistema de manejo de correspondencias y puntos de cuenta.

- Integración de los módulos de correspondencia, puntos de cuenta y administración dentro del sistema.
- Realización de una prueba piloto con el prototipo.

1.6 Plataforma Tecnológica Propuesta del Sistema

La Plataforma Tecnológica propuesta, se presenta desde dos perspectivas: Arquitectura Física y Arquitectura Lógica. A continuación se describe cada una de éstas:

a) Arquitectura Física. Como su nombre lo indica, constituye la disposición física de los elementos de la Arquitectura de la Plataforma Tecnológica Propuesta.

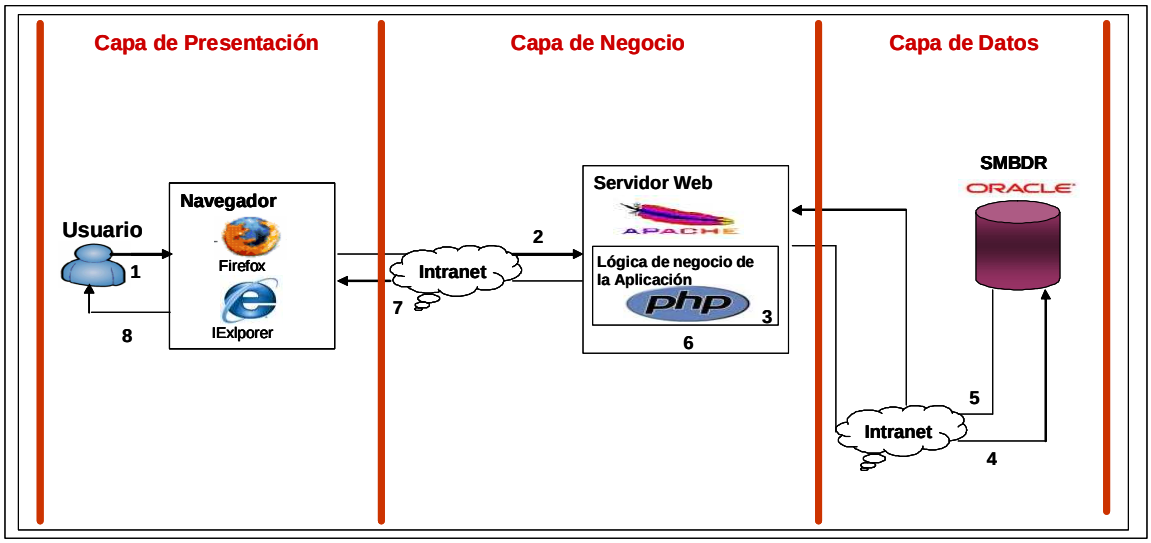

Figura 1.3. Arquitectura Física de la Plataforma Tecnológica propuesta (Fuente: Elaboración propia).

La Arquitectura Física de la Figura 1.3, funciona de la siguiente manera:

1) Un usuario solicita un requerimiento mediante un Navegador (*Internet Explorer o Mozilla Firefox*, los cuales son los únicos utilizados en el Banco), a través de un *URL* asociado dentro de la *Intranet*.

2) El Navegador utiliza el protocolo *HTTP* para buscar un recurso dentro de la *Intranet* del Banco, en este caso, un Servidor *Web*. El navegador localiza el Servidor *Web* en la *Intranet* y le hace una petición, es decir, le solicita un recurso (una página *HTML*) con un resultado esperado.

3) El Servidor *Web* contiene la Aplicación *Web* capaz de dar respuesta a la petición. Se ejecuta entonces, la lógica de negocio de la aplicación bajo lenguaje *PHP*.

4) Según la lógica de negocio ejecutada por la aplicación, ésta solicita un conjunto de registros a la base de datos de la aplicación alojada en el Servidor del Sistema Manejador de Base de Datos Relacional (SMBDR) mediante el lenguaje *SQL*.

5) El SMBDR ejecuta la consulta a la base de datos correspondiente y devuelve un cursor con los datos (registros) esperados por la aplicación del Servidor *Web*.

6) La lógica de negocio de la aplicación recibe los datos esperados de la capa de datos y, con ello, se construye una página *HTML* con los resultados asociados.

7) El Servidor *Web* entrega al Navegador una página *HTML* con los resultados del procesamiento.

8) El Navegador entrega la página *HTML* al usuario con los resultados esperados de la petición.

b) Arquitectura Lógica. Esta Arquitectura, hace abstracción de la ubicación física de los elementos de la plataforma, y muestra la composición lógica de la aplicación en todas sus partes.

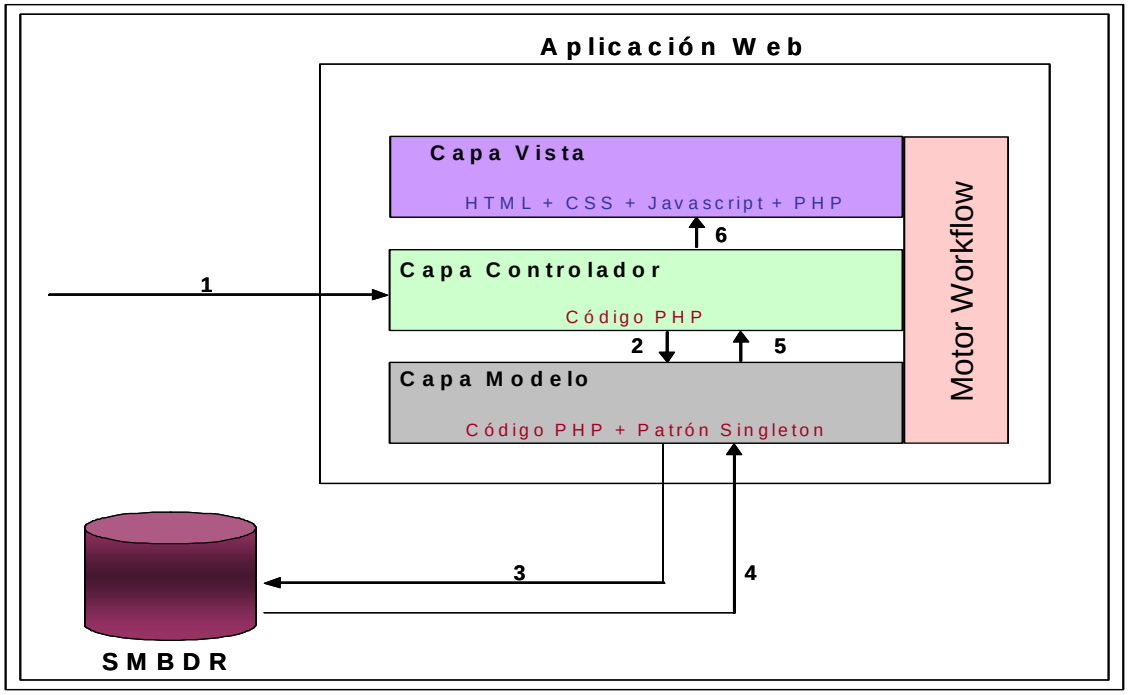

Figura 1.4. Arquitectura Lógica de la Plataforma Tecnológica Propuesta (Fuente: Elaboración propia).

La Arquitectura Lógica de la Figura 1.4, funciona como sigue:

1) La Capa Controlador recibe una petición.

2) Luego se encarga de seleccionar apropiadamente qué parte de la capa modelo puede realizar la lógica de negocio para dar respuesta al requerimiento y le solicita a ésta, un procesamiento en particular.

3) La Capa Modelo realiza el procesamiento para satisfacer la petición y, de acuerdo a ello, solicita un conjunto de datos (registros) al SMBDR mediante el lenguaje *SQL*.

4) El SMBDR selecciona un conjunto de registros según lo requerido por la sentencia *SQL*, y lo devuelve a la Capa Modelo de la aplicación.

5) La Capa Modelo, puede o no, hacer un procesamiento adicional con los datos suministrados por el SMBDR, y retorna el resultado final a la Capa Controlador.

6) La Capa Controlador, de acuerdo al resultado obtenido desde la Capa Modelo, selecciona la vista apropiada y le pasa los datos necesarios para que esta última pueda mostrarse.

Cabe destacar, que el motor *Workflow*, lógicamente está relacionado con cada una de las capas de la Aplicación *Web*, por lo que no se encuentra centralizado como un componente aparte y aislado de las demás capas, y realiza sobre estás, las funciones para lo cual fue construido.

1.7 Justificación e Importancia

El desarrollo de este Trabajo Especial de Grado permitirá:

- Integrar los sistemas de correspondencia y puntos de cuenta, ya que ambos manejan tareas inherentes a la Presidencia del BANAVIH.
- Minimizar los tiempos de respuesta para los procesos de correspondencia y puntos de cuenta.
- Incorporar mecanismos de seguridad (manejo de roles) que permitan limitar el acceso a información de gran importancia.
- Eliminar el manejo manual de los puntos de cuenta e incorporar una estructura organizada que permita el fácil acceso a la información.
- Garantizar la integridad de los datos, para evitar redundancia e incongruencias que lleven a información errónea.

- Aumentar la confidencialidad de la información del seguimiento de los puntos de cuenta.

Con base en lo mencionado anteriormente, es necesaria la elaboración de un Sistema de Información Automatizado que sea usable, en donde se agrupe toda la información (tanto de puntos de cuenta como de correspondencias), que permita la comunicación efectiva entre la Presidencia, la Secretaria de Junta Directiva y las distintas gerencias.

CAPÍTULO II: Marco Conceptual

El presente capítulo tiene como meta suministrar la base teórica que soporte el desarrollo de este trabajo de investigación.

Se comienza haciendo un resumen de los sistemas de información, en donde se hace referencia al tipo de sistema de información implementado en el presente trabajo: sistema de información transaccional.

Se estudia además, lo que se conoce como *workflow*, y los conceptos más importantes para la compresión del mismo.

Se detalla el proceso de desarrollo XP, indicando su definición, características y actividades.

Por otro lado, se provee información de como las aplicaciones con Tecnología Web trabajan dentro de una Intranet, así como también, la forma en que se adaptan al modelo de tres capas.

También se estudian brevemente los patrones MVC y *Singleton.*

Por último, se da un pequeño resumen, acotando las ventajas de una tecnología muy utilizada actualmente por equipos de desarrollo de *software*: Subversión.

2.1 Sistemas de Información

El presente caso de estudio implementa un sistema de información, el cual recibe entradas, procesa y disemina información dentro del banco. Es por esto que se hará un resumen de los sistemas de información.

Un sistema de Información "…puede ser cualquier combinación organizada de personas, hardware, software, redes de comunicación, y recursos de datos que recolectan, transforman, y diseminan información en una organización" (O'Brien, James A., 2001, pp. 7).

2.1.1 Tipos de Sistemas de Información

Los sistemas de información se desarrollan con diversos propósitos, según las necesidades de la empresa. Por lo tanto, existe una diversidad de sistemas de información que podrían desarrollar los analistas, los cuales aparecen de forma jerárquica en la Figura *2.1*. El aporte de estos tipos de sistemas de información y su relación con los niveles de una organización, según (Kendall & Kendall, 2005), es como sigue: "Los sistemas de procesamiento de transacciones (TPS, Trasaction Processing Systems) funcionan al nivel operativo de una organización, los sistemas de automatización de la oficina (OAS, Office Automation Systems) y los sistemas de trabajo del conocimiento (KWS, Knowledge Work Systems) apoyan el trabajo a nivel de conocimiento. Los sistemas de información gerencial (MIS, Management Information Systems) y los sistemas de apoyo a la toma de decisiones (DSS, Decision Support Systems) se encuentran entre los sistemas de alto nivel. Los sistemas expertos aplican el conocimiento de los encargados de la toma de decisiones para solucionar problemas estructurados específicos. Los sistemas de apoyo a ejecutivos (ESS, Executive Support Systems) se encuentran en el nivel estratégico de la administración. Los sistemas de apoyo a la toma de decisiones en grupo (GDSS, Group Decision Support Systems) y los sistemas de trabajo corporativo apoyados por computadora (CSCWS, Computer-Supported Collaborative Work Systems), descritos de manera más general, auxilian la toma de decisiones semiestructuradas o no estructuradas a nivel de grupo".

Figura 2.1. Tipos de Sistemas de Información (Fuente: Kendall & Kendall, (2005), pp. 2).

El presente caso de estudio implementa un Sistema de Procesamiento de Transacciones (*TPS, Transaction Processing System*).

Un *TPS* permite simplificar la complejidad y disminuir el tiempo que se requería en el pasado para manejar transacciones operativas; sin embargo, los usuarios tienen aún que capturar datos en los sistemas computarizados. Es de suma importancia que estos sistemas funcionen sin ningún tipo de interrupción, puesto que los administradores recurren a los datos producidos por los *TPS,* con la finalidad de obtener información actualizada sobre el funcionamiento de sus empresas.

Entre las principales características de estos sistemas, se tienen:

- Automatizan tareas operativas de la organización.

- Con frecuencia, son los primeros tipos de Sistemas de Información que se implantan en las organizaciones.
- Son intensivos en entrada y salida de información; sus cálculos y procesos suelen ser simples y poco sofisticados.
- Tienen la propiedad de ser recolectores de información; es decir, a través de estos sistemas se cargan las grandes bases de información para su explotación posterior.
- Son fáciles de justificar ante la dirección general, ya que sus beneficios son visibles y palpables. Entre las ventajas que pueden medirse se encuentra el ahorro de trabajo manual e insumos como papeles, carpetas, etc.

2.2 Workflow

Es oportuno hacer un pequeño estudio de lo que se denomina *workflow*, ya que el prototipo de aplicación del presente trabajo de investigación, esta orientado a la automatización de procesos de negocio bajo un enfoque *workflow*.

La Coalición de Gestión de *Workflow* (*WfMC, Workflow Management Coalition*) es una organización sin fines de lucro cuyo objetivo es propiciar las oportunidades para la explotación de la tecnología *workflow* a través del desarrollo de estándares y terminología común.

La *WFMC* define un *Workflow* como "La automatización de procesos de negocios, en su totalidad o en parte, en función de cómo sus documentos, información o tareas son pasadas de un participante a otro para realizar su tarea de acuerdo a un conjunto de reglas procedurales" (1999).

2.2.1 Otros conceptos de Workflow

Existen otros conceptos importantes que ayudan a comprender los componentes de un *workflow*, y la forma en como este funciona:

2.2.1.1 Proceso de negocio

Un proceso de negocio consiste en un conjunto de una o más actividades, las cuales colectivamente cumplen un objetivo de negocio previamente planteado por la organización, definiendo roles y relaciones funcionales. Entre las características más resaltantes de un proceso de negocio, se tienen:

- Un proceso de negocio esta típicamente asociado a objetivos operacionales y relaciones de negocios, como por ejemplo, un proceso de reclamo de seguro, o un proceso de desarrollo de ingeniería.
- Un proceso de negocio define condiciones para disparar su inicio en cada nueva instancia (por ejemplo, la llegada de un reclamo) y establece salidas al momento de su finalización.
- Un proceso de negocio puede involucrar interacciones formales o relativamente informales entre los participantes (roles); su duración puede también variar ampliamente.
- Un proceso de negocio puede consistir de actividades automatizadas y/o actividades manuales.

2.2.1.2 Actividad

Consiste en una descripción de una pieza de trabajo que forma un paso lógico dentro de un proceso. Una actividad puede ser manual o puede ser una actividad de *workflow* (automatizada). Una actividad automatizada

puede ser ejecutada por recursos humanos y/o por una máquina (por ejemplo, un agente inteligente).

2.2.1.3 Ítem de Trabajo

Un *ítem* de trabajo no es más que la representación del trabajo a ser procesado (por un participante del *workflow*) en el contexto de una actividad dentro de una instancia de proceso. Haciendo referencia al presente caso de estudio, un ítem puede ser una correspondencia o un punto de cuenta.

2.2.1.4 Participante del Workflow

También llamado rol, constituye un recurso que realiza el trabajo representado por una instancia de actividad. En fin, un rol hace referencia a que clase de usuario ejecuta las actividades sobre los ítems.

2.2.2 Tipos de Workflow

Generalmente se distinguen dos tipos de *workflow*:

- *Workflow* de producción.
- *Workflow* colaborativo.

El presente caso de estudio se enfoca en un *workflow* de producción, el cual es similar a la producción en una línea de ensamble en una fábrica; debe ejecutarse en el menor tiempo posible, es altamente predecible, repetitivo y de alto volumen. Los trabajadores, en la línea de ensamble, pasan su mayor parte del tiempo produciendo objetos; pueden participar en actividades adicionales, pero ellas son secundarias. Debe notarse, además, que el *workflow* de producción se suele circunscribir a un sólo

departamento de la empresa; la escalabilidad, o capacidad de "crecer" no es importante. En un banco, por ejemplo, los individuos a cargo de la aprobación de solicitudes de crédito sólo realizan *workflow* para esa actividad; es improbable que otros funcionarios del banco realicen esa actividad fuera del departamento.

Debido a su naturaleza de "producción", las aplicaciones que gestionen *workflows* de producción deben cumplir con algunos de los siguientes atributos:

- Velocidad de transferencia, esto es, la velocidad con que las tareas pasan de un paso a otro; es muy importante en el *workflow* de producción, ya que es la tarea principal de los participantes. Es improductivo que un miembro del equipo no haga nada mientras espera a que le llegue trabajo.
- La flexibilidad de poder cambiar el proceso no suele ser importante. Una vez establecido el flujo, este permanece sin cambio por largo tiempo.

2.3 Programación Extrema (XP)

Para este caso de estudio se utilizó *XP*, debido a que es el enfoque de desarrollo que se utiliza en BANAVIH.

A continuación se dará un pequeño resumen de los aspectos resaltantes de este enfoque:

2.3.1 Definición

"La programación extrema (XP) es un enfoque de desarrollo de software… que adopta lo que generalmente designamos como prácticas de desarrollo de software aceptable y las lleva al extremo" (Kendall & Kendall, 2005, pp. 165).

"…. la programación extrema intenta definir rápidamente un plan global del sistema, desarrollar y liberar rápidamente el software y posteriormente revisarlo continuamente para incorporarle características adicionales" (Kendall & Kendall, 2005, pp. 165).

2.3.2 Características de XP

Las características de *XP* son las siguientes:

- ***XP* es un proceso de desarrollo ligero**, debido a que los desarrolladores no necesitan manejar mucha documentación y lo importante es entregar el producto al cliente con las características que necesita.
- **La comunicación entre el cliente y el equipo de desarrolladores** es importante para el cumplimiento satisfactorio de los objetivos del sistema, permitiendo establecer una retroalimentación (aportes de los clientes y los desarrolladores), para hacer los cambios necesarios a tiempo, dentro del sistema.
- **Sencillez:** El equipo de desarrollo debe empezar a desarrollar la aplicación, desde lo más simple hasta lo más complejo, y cada problema debe resolverse de manera sencilla, evitando la complejidad.

- **Liberación Limitada:** El equipo de desarrollo debe entregar el producto en la fecha establecida, con las características más importantes del sistema, y luego establecer las mejoras requeridas.
- **Cliente en el sitio:** Para poder cumplir con las metas de la aplicación establecidas, es necesario que el cliente se encuentre en el mismo sitio en el que esté el equipo de desarrolladores, debido a que debe interactuar con los miembros del equipo de desarrollo e indicar cuáles son las características más resaltantes que deben ser incluidas en el sistema.
- **Los clientes utilizan historias de usuarios** escritas por ellos mismos, para poder identificar los requerimientos más resaltantes del negocio.
- **Retroalimentación Rápida:** Es necesario que haya una constante verificación (tanto del cliente como de los desarrolladores) de la aplicación, por lo que debe existir un límite de tiempo corto en el que se detecten los posibles errores, a fin de que el equipo de desarrollo aplique las correcciones a tiempo.
- **Refactorización:** Consiste en la realización de mejoras del código del sistema por parte de los desarrolladores en posteriores entregas del producto.
- **Programación en parejas:** *XP* se basa en el trabajo en parejas; es decir, dos personas se encargan de programar y hacer las respectivas pruebas. Este método evita distracciones, ahorra tiempo y permite estimular la creatividad.
- Es ideal para equipos de desarrollo y proyectos pequeños.
- **Iteraciones:** XP consiste en iteraciones incrementales antes de la entrega de la primera versión, en la que se hacen cambios (fechas

de entrega, historias de usuarios, etc.), ciclos de pruebas, y se lleva a cabo la retroalimentación.

2.3.3 Actividades de XP

Existen cuatro actividades que utiliza la programación extrema, las cuales son:

- **Planificar:** Cada desarrollador debe estar en constante comunicación con el cliente y con los miembros del equipo de desarrollo, para entender correctamente las ideas que se deben diseñar, codificar y probar, con la finalidad de establecer una planificación adecuada del proyecto.
- **Diseñar:** Es necesario estructurar la lógica del sistema, para poder establecer el enfoque adecuado para el desarrollo de la aplicación. A partir de un buen diseño, los programadores pueden hacer los cambios o agregar nuevas extensiones en el sistema; y los clientes pueden tener una idea de cómo será construida la aplicación.
- **Codificar:** A través del código se puede implementar conceptos lógicos que provienen de una idea, a fin de probar y verificar, si ésta última, es la más adecuada a la solución requerida.
- **Probar:** *XP* se basa en pruebas automatizadas que verifican el funcionamiento de la aplicación. También utiliza las pruebas escritas para verificar el funcionamiento, rendimiento y el código del sistema, y ver si los objetivos han sido cumplidos.

2.4 Aplicaciones con Tecnología Web

Si bien es cierto que las Aplicaciones con Tecnología W*eb*, son cada vez más utilizadas dentro del mundo informático, hasta convertirse casi en un estándar de tipo de aplicación, es importante su compresión, debido a que los antecedentes del presente caso de estudio refieren una aplicación tipo *standalone*, la cual automatizaba el proceso de manejo de correspondencia dentro del banco, según un estudio de la situación actual (Capítulo 1, Pág. 1 del presente trabajo). Es importante entonces, resaltar la forma de trabajo y ventajas de las aplicaciones con Tecnología *Web* con respecto a las aplicaciones tradicionales *standalone*.

Se entiende por Aplicaciones con Tecnología *Web* al conjunto de tecnologías implementadas del lado del servidor y del cliente que involucran manejo de persistencia con bases de datos mediante el uso de un navegador *Web*, con el objetivo de dar respuesta a ciertos requerimientos de usuario, a fin de optimizar los procesos del manejo de información en las instituciones y organizaciones, ya que generan de forma dinámica un conjunto de datos públicos o privados a los usuarios que acceden a través de Internet o mediante una Intranet empresarial.

Hay muchos beneficios en la implementación de una aplicación en la *Web*, entre los cuales se tienen los siguientes:

- Una creciente difusión de la disponibilidad de un servicio, producto, industria, persona o grupo.
- La posibilidad de que los usuarios accedan las 24 horas.
- Múltiples usuarios concurrentes, ya que varios usuarios pueden ver e incluso editar el mismo documento de manera conjunta.

- La creación de un sistema que se puede extender a nivel mundial y llegar a la gente en lugares remotos sin preocuparse por la zona horaria en que se encuentren.

2.4.1 Intranet

"Una Intranet es una infraestructura de comunicaciones. Esta basada en estándares de comunicación de Internet y en estándares de contenido de la World Wide Web. Por lo tanto, las herramientas usadas para crear una Intranet son idénticas a las usadas para aplicaciones Web y de Internet. La característica resaltante de una intranet es que el acceso a información publicada en Internet es restringido a los clientes del Grupo de la Intranet. Históricamente esto ha sido cumplido a través del uso de LANs protegidas por Firewalls. " (Amdahl Corporation, 1996).

Las Intranets corren sobre protocolos y tecnologías estándares de Internet (*TCP/IP*, *HTTP*, y *HTML*) que son usados por la *World Wide Web*. Esto permite a las compañías usar los mismos tipos de servidores y navegadores (que estas usan actualmente para tener acceso a la *WWW*) para aplicaciones internas e información distribuida sobre su red de área local (*LAN*, *Local Area Network*), dando origen a lo que se conoce como Aplicaciones de Negocio de Intranet.

2.4.1.1 Aplicaciones de Negocio de Intranet

En términos generales, dado un entorno de una Aplicación de Negocio de *Intranet*, pueden rescatarse las siguientes características:

- Usuarios con un propósito de negocio común.
- Usuarios gozan de Soporte de interacción en tiempo real para el procesamiento de sus solicitudes.

- Transacciones complejas con múltiples diálogos de entrada y salida.
- Una interfaz de usuario presentada y manejada por un navegador *Web*.
- Servidores *Web* alberga la aplicación que contiene la lógica de negocio que procesa los requerimientos provenientes del usuario.
- Comunicaciones sobre la *LAN* interna corporativa, principalmente a través de los protocolos *HTTP* y *TCP/IP*.

El entorno sobre el cual opera una aplicación de negocio de *Intranet*, básicamente consiste en una arquitectura de tres capas, la cual es similar a la arquitectura tres capas de los Sistemas Cliente-Servidor. El nivel inferior es generalmente el Servidor de base de datos. El nivel medio contiene un Servidor *Web* y la lógica de negocio. El nivel superior emplea un navegador *Web* como un *host* (huésped) para la interfaz de usuario. La Figura 2.4 muestra la arquitectura tres niveles de una Aplicación de Negocio de *Intranet*.

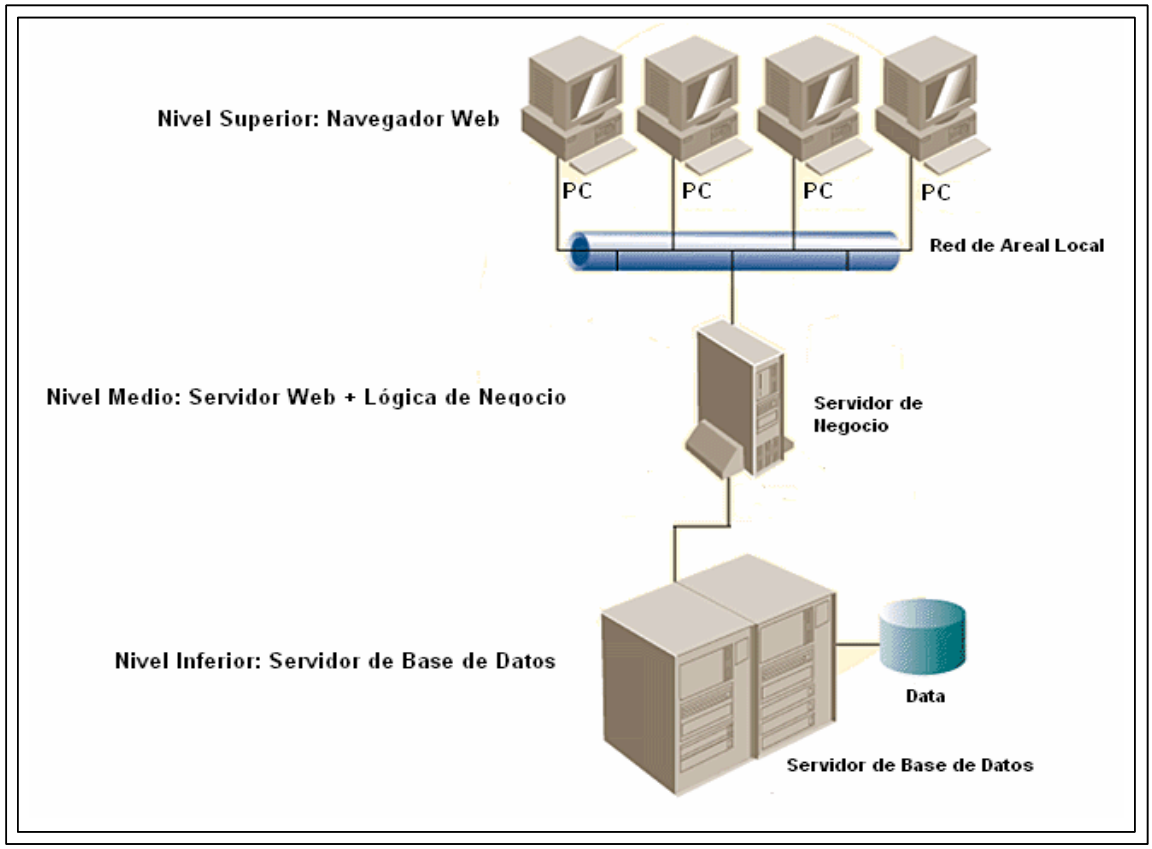

Figura 2.2. Arquitectura por niveles de una aplicación de Intranet (Fuente: International Engineering Consortium, 2007).

2.4.1.1.1 Nivel Superior. El Navegador como Interfaz de Usuario.

El principal propósito de un navegador es la entrada y presentación de los datos relacionados a un usuario particular; el navegador en sí mismo no conoce nada acerca de la aplicación de negocio; sin embargo, el desarrollador puede validar campos antes de que el navegador retorne datos al *host*. El desarrollador puede crear una presentación y mecanismos de navegación de pantallas, que cada vez sean más similares al funcionamiento de las *GUI* de escritorio.

Igual que con una *GUI* de escritorio, una aplicación *Intranet* debe poner atención a los temas de usabilidad. Se deben conocer las

necesidades y capacidades de los usuarios. ¿Cuál es su nivel de experiencia? ¿Cuánto entrenamiento recibirán? ¿Cuán frecuentemente usarán el programa y por cuánto tiempo? El diseñador debe ser consciente de cómo se usarán las características del programa, tanto las más usadas, como las más fáciles y rápidas.

El navegador como interfaz de usuario especifica un localizador de recurso uniforme (*URL, Uniforme Resource Locator*), solicitándole una página al Servidor *Web*, el cual está localizado en el nivel intermedio y alberga la aplicación de negocio, la cual conoce como procesar los requerimientos del negocio.

2.4.1.1.2 Nivel Intermedio. El Servidor Web y la Lógica de Negocio.

El nivel intermedio posee dos componentes claves: el servidor *Web* y la lógica de negocio. El servidor *Web* es el componente que capta y da respuesta a las solicitudes desde los navegadores *Web*. El servidor *Web* entrega una página *html* al navegador como respuesta a sus solicitudes. Puede enviar algunas páginas sin ninguna modificación, pero en una aplicación de negocio, el servidor *Web* usa la lógica de negocio que contiene la aplicación para modificar las páginas antes de enviarlas al navegador.

En teoría, las aplicaciones de negocio deben implementar mecanismos de seguridad para todos los sitios *Web* (aunque en la práctica esto no siempre sucede). La seguridad requiere la cooperación entre los componentes de las capas superior y media para autenticar al usuario y autorizar los servicios de aplicación. Un enfoque común es el uso de una ventana de diálogo en el navegador para ingresar nombre y clave de usuario. El nivel intermedio es el encargado de autenticar al usuario (esto es, validar su autorización para acceder a ciertos servicios de la

aplicación), para luego establecer una sesión asociada al mismo. Todos los pases de mensajes entre el navegador y el servidor *Web* usan un identificador de sesión para mantener una conexión lógica entre los niveles e identificar a un usuario dentro de la aplicación *Web*.

"Una vez que la sesión es establecida, el servidor *Web* posee un mecanismo para almacenar estados de transacción. El estado de una transacción refleja la entrada mas reciente de datos del usuario que no ha sido procesada en la base de datos (*commit*). Por ejemplo, cuando se crea una cuenta de usuario, la información del cliente y sus servicios es reunida, y quizás se prepara un requerimiento para instalar un nuevo servicio. A medida que el usuario ingresa estos trozos de información, los mismos no son actualizados a la base de datos en el mismo momento. Por el contrario, son mantenidos en algún lugar de memoria hasta que la orden completa esta lista. La orden puede entonces ser validada para asegurar consistencia e integridad antes de que los cambios sean actualizados en la base de datos.

El servidor *Web* puede usar varias vías de acceso a la lógica de negocio y colocar los resultados en la página *Web*. La mayoría de los servidores *Web* permiten incluir código embebido directamente en la página *Web*. Este código, escrito en un lenguaje como *PHP* o *Java*, puede directamente acceder a la lógica de negocio" (International Engineering Consortium, 2007).

2.4.1.1.3 Nivel Inferior. El Servidor de base de datos.

El cimiento para las aplicaciones de negocio de *Intranet* es la base de datos. La base de datos es responsable de almacenar información hasta donde el sistema lo requiera. En particular, maneja la persistencia de todos los aspectos del sistema de negocio. Entre los diversos tipos de bases de

datos, tenemos la base de datos relacional, la cual almacena elementos de datos en tablas que contienen filas y columnas; cada tabla posee una o varias columnas que sirven de clave para localizar los datos.

En definitiva, es en este nivel, donde reside la información crítica y estrictamente confidencial de una empresa u organización, por lo tanto, se requiere asegurar control total sobre los datos que pueden ser accedidos en este nivel.

2.5 Patrón Singleton

Singleton es un patrón de diseño de tipo creación, que permite que una clase *Singleton* sólo pueda ser instanciada una vez. Bajo este enfoque, varios objetos pueden compartir el acceso a ésta única instancia. El patrón *Singleton* se usa en este caso de estudio para poder limitar la cantidad de instancias de una conexión a sólo una.

2.5.1 Reglas de Funcionamiento del Patrón Singleton

Las reglas que se cumplen para el correcto funcionamiento del patrón *Singleton* son las siguientes:

- Poseer un constructor de la clase *Singleton* oculto; es decir, debe ser declarado privado o protegido.
- Debe existir una variable declarada que tendrá el contenido de la instancia (privada y estática).
- La clase *Singleton* debe contener un método estático y público que contenga el acceso a la única instancia.

2.6 Patrón MVC (Modelo-Vista-Controlador)

Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software que permite separar el modelo de negocio, la lógica de control y presentación de los datos, con la finalidad de colocarlos en tres componentes: modelo, vista y controlador. El presente caso de estudio, utiliza como arquitectura de software al patrón MVC, debido a que constituye un estándar de desarrollo dentro del banco, y además, permite mantener y escalar una aplicación de forma más fácil, al estar separadas las capas de negocio, control y presentación. Es por ello que se hace necesario realizar un pequeño análisis del funcionamiento de este patrón arquitectónico.

2.6.1 Componentes del Patrón MVC

Los tres componentes del patrón MVC son: Modelo, Vista y Controlador. Estos se describen a continuación:

- **Modelo:** Maneja las reglas del negocio, contiene la representación de la información, y permite el acceso y actualización de los datos.
- **Vista:** Recibe la información y la muestra al usuario final a través de la interfaz de la aplicación.
- **Controlador:** Procesa los eventos de entrada llamando las acciones que hayan sido activadas por los usuarios; por lo que invoca al modelo para que procese los datos y envíe la información a la vista correspondiente, dependiendo de la acción invocada.

2.6.2 Ventajas del Patrón MVC

Las ventajas del patrón MVC son las siguientes:

- Reutilización de los componentes del modelo: Debido a que la vista y el modelo se encuentran separados, es posible que múltiples vistas puedan reutilizar el mismo modelo de negocio.
- Escalabilidad: La separación de componentes facilita incorporar nuevas funcionalidades a la aplicación.
- Proporciona un enfoque más claro del diseño, debido a que se tiene una idea clara del funcionamiento del modelo, vista y controlador.
- Evita el acoplamiento entre capas, debido a que el modelo no conoce cuál es la vista.
- Independencia de Funcionamiento, debido a que cada componente tiene funciones específicas.

2.6.3 Desventajas del Patrón MVC

Las desventajas del patrón MVC son las siguientes:

- Debido a la separación de los componentes y a las múltiples vistas que se crean para el mismo modelo, el sistema es más complejo.
- El número de archivos a desarrollar y mantener aumenta.

2.7 Subversión (SVN)

Subversión es una tecnología utilizada por el equipo desarrollador, que facilitó la construcción del "sistema de información para el manejo de correspondencia y puntos de cuenta" (objeto de estudio de este TEG), por

lo que es imprescindible la compresión de su significado y ventajas asociadas.

Subversión, también conocido con las siglas *SVN*, es un sistema de control de versiones que permite administrar las versiones de los archivos de un proyecto, promoviendo con esto la colaboración entre los miembros de un equipo de trabajo y, de una manera muy cómoda y efectiva, coordinar las tareas entre ellos a lo largo de un proyecto de desarrollo de *software*.

La herramienta permite acceso a los archivos de un proyecto, observar el trabajo, realizar cambios y guardar los mismos en su repositorio, donde se almacenan todos los archivos involucrados. Al finalizar una acción de guardar cambios, se considera que se ha creado una nueva revisión o versión.

2.7.1 Ventajas de Subversion

Entre las ventajas más importantes de la utilización de subversión para un equipo desarrollador, se tienen:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados, las cuales constituyen lo que se conoce como versiones.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- Se envían sólo las diferencias en ambas direcciones (en *CVS* siempre se envían al servidor archivos completos).
- Permite selectivamente el bloqueo de archivos, principalmente cuando conviene que no sean editados por más de una persona a la vez.

- Cuando se usa integrado a *Apache*, permite utilizar todas las opciones que este servidor provee al momento de autentificar archivos (*SQL*, *LDAP*, *PAM*, etc.).
- Posibilidad de revertir una actualización que se ha hecho previamente.

Hasta ahora, se ha visto en este Capítulo II los conceptos teóricos más importantes que proveen la base conceptual para el desarrollo del marco aplicativo, el cual es a continuación detallado en el Capítulo III.

CAPÍTULO III: Marco Aplicativo

En el capítulo I del presente trabajo, se describió un estudio de la situación actual en los procesos de manejo de correspondencia y manejo de puntos de cuenta, el cual determinó que estos procesos se componen de una serie de actividades secuenciales, en donde los participantes van interviniendo en las mismas para agregar información a documentos (correspondencias y puntos de cuenta), dando origen a lo que se conoce como *workflow*. Tal y como se expresó anteriormente, los resultados arrojados del estudio de la situación actual de ambos procesos se pueden resumir mediante los diagramas de actividades correspondientes a las Figuras 1.1 y 1.2 respectivamente.

En tal sentido, la idea es mejorar la eficiencia de ambos *workflows* de actividades, tanto en tiempo de ejecución como en utilización de recursos. Por otro lado, la metodología aplicada para el desarrollo del "Sistema de Información para el manejo de Correspondencia y Puntos de Cuenta" se basa en un proceso de desarrollo denominado Programación Extrema (*XP*), explicado anteriormente en el Capítulo II. Esto significa que a través del *match* (asociación) entre un *workflow* propuesto y las historias de los usuarios se llega a un *workflow* final, los cuales se muestran en las figuras 3.1 y 3.2.

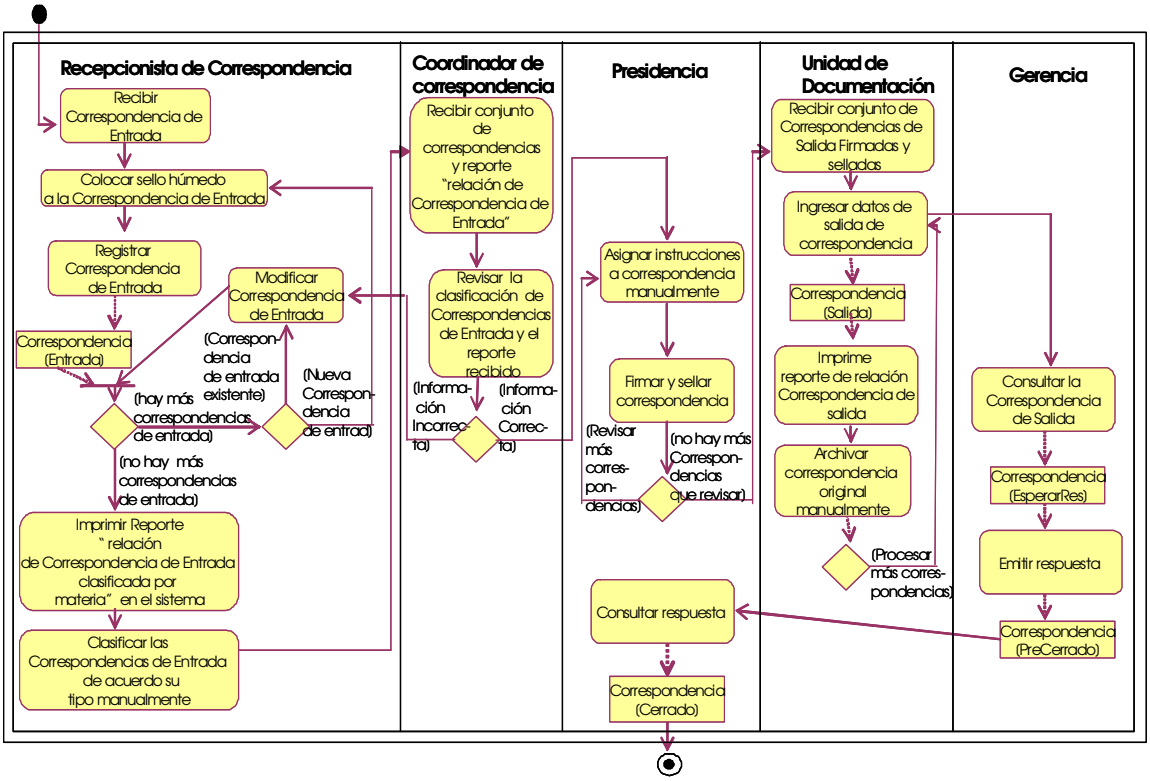

Figura 3.1. Diagrama de actividad del Workflow final correspondiente a la entrega y recepción de correspondencia de la gerencia de Presidencia (Fuente: Elaboración propia).

Entre las mejoras del diagrama de actividad del workflow final del manejo de correspondencia con respecto al workflow actual, se tienen las siguientes:

- La recepcionista, ya no necesita buscar en el archivo Excel los códigos de las materias para introducirlos y clasificar las correspondencias en el sistema, estas clasificaciones ya se encuentran de forma automática dentro del sistema prototipo.
- El reporte de relación de correspondencia de entrada, está ahora ordenado por materias, para de esta forma, facilitar la

clasificación en físico, del bloque de correspondencias antes de ser pasadas al coordinador.

- No hace falta un mensajero que fotocopie y distribuya las copias de las correspondencias a las gerencias, puesto que la unidad de documentación escanea las portadas de las correspondencia (donde esta la firma, sello e instrucciones del presidente), y la cargas en el sistema. De esta forma, cada gerencia, al consultar las salidas de las correspondencias, pueden observar lo que realmente les importa: la firma, sello e instrucciones del presidente.
- La respuesta de las gerencias, y su posterior revisión por el presidente, se hacen dentro del sistema, por lo que no hace falta que el gerente responda a través de un memorando y envíe la respuesta por medio de un mensajero a la presidencia.
- Se puede hacer un seguimiento de las correspondencias, a través de la consulta, dentro del sistema, de todos los estados por los que ellas pasan a lo largo del workflow: entrada, salida, esperar respuesta, precerrado y cerrado.

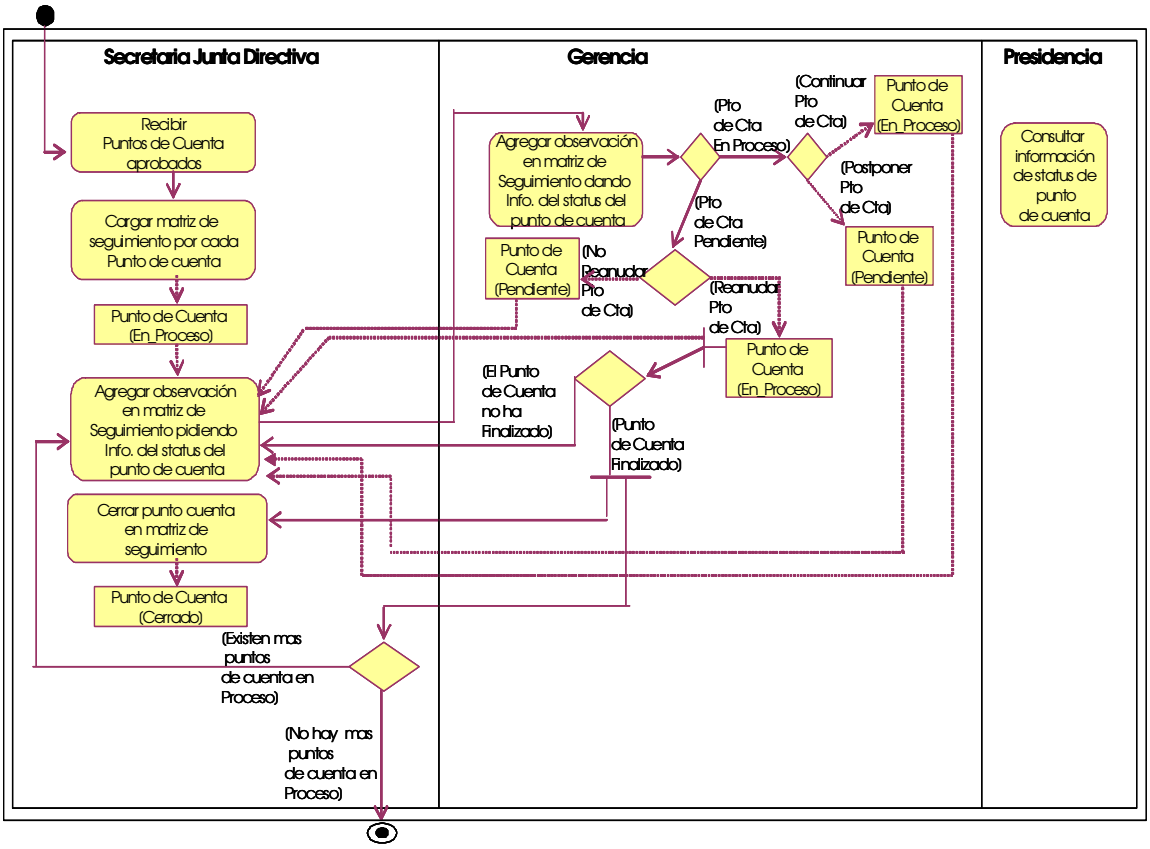

Figura 3.2. Diagrama de Actividad del Workflow final del Seguimiento de Puntos de Cuenta

(Fuente: Elaboración propia).

Entre las mejoras del diagrama de actividad del workflow final del seguimiento de puntos de cuenta con respecto al diagrama de actividad workflow actual, se tienen las siguientes:

- Carga y consulta de los puntos de cuenta de forma automatizada dentro del sistema.
- Las observaciones entre la secretaria de la Junta Directiva y las gerencias, son hechas dentro del sistema, quedando plasmadas una vez que se han incluido; es decir, las observaciones hechas no se pueden borrar, a fin de que quede constancia y compromiso con lo que se ha dicho.

- Disminución del tiempo de respuesta de las gerencias, ya que ahora las mismas deben revisar sus puntos de cuentas asociados diariamente, para observar si hay una solicitud de respuesta del status de un punto de cuenta por parte de la secretaría de la Junta Directiva.
- Ahora la información de los puntos de cuenta solo fluye entre la secretará de la Junta Directiva y los gerentes, no como antes, cuando la secretaria y el mensajero, en algunos casos, observaban información confidencial.
- Se puede hacer un seguimiento del estado de los puntos de cuenta, a través de la consulta, dentro del sistema, de todos los estados por los que ellos pasan a lo largo del workflow: en proceso, pendiente y cerrado.
- El presidente puede en cualquier momento consultar el estado y observaciones de un punto de cuenta, sin necesidad de esperar una respuesta por parte de la secretaría de Junta Directiva.

La idea principal es cubrir el desarrollo del sistema expresado en los *workflows* finales mediante iteraciones. Cada iteración consiste en un conjunto de actividades automatizables relacionadas a cada rol (calle) del *workflow* final. El orden de las iteraciones va de izquierda a derecha en los diagramas de actividades expresados anteriormente. Esto significa, que por ejemplo, para el *workflow* final de correspondencia, la primera iteración corresponde a desarrollar el conjunto de actividades automatizables del rol (calle) "Recepcionista de Correspondencia". Una vez realizadas estas actividades, se procede a realizar la próxima iteración, que no es más que automatizar las actividades del siguiente rol, sí y solo si

tiene al menos una actividad susceptible de ser automatizada dentro del flujo secuencial de actividades. Siguiendo el ejemplo, esto significa que la próxima iteración (la 2da.) corresponde a las actividades del rol "Unidad de Documentación", puesto que como puede apreciarse, las actividades dentro del flujo secuencial de los roles de "Coordinador de Presidencia" y "Presidente" no tienen actividades automatizables (sino manuales). Nótese que la actividad "Consultar respuesta" del rol de "Presidente" si es automatizable, pero no pertenece (en el ejemplo) al flujo secuencial de actividades, por lo que no forma parte de la segunda iteración, sino de otra iteración posterior.

3.1 Desarrollo del Sistema de Manejo de Correspondencia y Puntos de Cuenta basado en el proceso de desarrollo XP

Tal y como todo proceso de desarrollo, *XP* puede ser adaptado a las necesidades de cada desarrollo en particular. En este sentido, el sistema objeto de estudio del presente trabajo, se realizó a través de varias iteraciones, en las que se siguen las actividades propias del proceso *XP*: Planificación, Diseño, Codificación y Pruebas.

Como se expresó anteriormente, el desarrollo del sistema parte de los *workflows* propuestos por los analistas (producto del estudio de la situación actual), más las historias de usuarios del cliente, dando como resultado dos *workflows* finales, relacionados a los procesos de manejo de correspondencia y puntos de cuenta. Es de aquí donde parte la concepción de realizar dos módulos principales:

- Módulo de correspondencia.

- Módulo de puntos de cuenta.

Pero más allá de manejar los requerimientos propios (funcionales) de los procesos de negocio ya citados, existen requerimientos de administración del sistema que deben realizarse; por ejemplo: Manejo de usuarios, auditorias del sistema, actualización de datos que alimentan la aplicación, entre otras. Así como también, la posibilidad de que el sistema posea un pequeño motor de *workflow* que permita el manejo de actividades, manejo de roles, asignación dinámica de actividades según los roles y asignación dinámica de opciones de menú de usuarios, de acuerdo a las actividades asignadas previamente al rol de los mismos. Es entonces cuando surge la necesidad de la adición de un tercer módulo, quedando los módulos del sistema del presente caso de estudio como sigue:

- Módulo de Correspondencia.
- Módulo de Puntos de Cuenta.
- Módulo de Administración.

De lo anterior se desprende que:

Los módulos de Correspondencia y Puntos de Cuenta se componen de varios roles predefinidos (como lo expresan los diagramas de actividad de los *workflows* finales, en las figuras 3.1 y 3.2), cada uno de los cuales, solos o combinados con otros roles, representan una iteración (si es el caso, es decir, si posee actividades automatizables), mientras que el módulo de Administración presenta dos iteraciones para el desarrollo de los requerimientos del *Root* del Sistema.

En la figura 3.3 que se muestra a continuación, se detallan los módulos de correspondencia, puntos de cuenta y administración, con las

iteraciones que pertenecen a cada módulo. Asimismo, se aprecian los roles Recepcionista, Coordinación de Presidencia, Unidad de Documentación, Gerencia, Presidencia, Secretaría de Junta Directiva y Administrador del Sistema, relacionados con cada iteración.

Figura 3.3. Módulos, Iteraciones y Módulos del sistema (Fuente: Elaboración propia).

A continuación se describen los roles mostrados en la Figura 3.3:

- **Recepcionista:** es la encargada de recibir y registrar dentro del sistema la correspondencia que llega a la presidencia del banco.
- **Coordinador de Presidencia:** es el asesor del Presidente, encargado de filtrar y asegurar que las correspondencias estén correctamente clasificadas por materia.
- **Unidad de Documentación:** complementa las correspondencias que han sido registradas previamente por la recepcionista, agregando los datos necesarios para dar salida a las correspondencias hacia las gerencias respectivas; por ejemplo: asigna las instrucciones del

presidente, la gerencia destino, prioridad, responsable, carga la imagen de la correspondencia, la asocia a un expediente si es necesario, etc.

- **Gerencia:** consulta y manipula cierta información sobre los ítems de correspondencias y puntos de cuenta adscritos a su gerencia respectiva (no a otras).
- **Presidencia:** consulta cualquier información de los ítems, tanto de correspondencia, como de puntos de cuenta asociados a las distintas gerencias.
- **Secretaría de Junta Directiva:** se encarga de registrar todos los puntos de cuenta, previamente aprobados por la Junta Directiva e informa al Presidente del estado de los puntos de cuenta, solicitando la información a los gerentes responsables de los mismos.
- **Administrador del Sistema:** se encarga de crear, eliminar o modificar usuarios, roles, actividades, consultar las pistas de seguimiento del sistema (auditorias), entre otras cosas, concernientes a la administración de la aplicación.

Como se puede observar, estos roles y sus actividades están predefinidos en el sistema porque parten de las historias de usuarios. Existe también la posibilidad de que pueda haber una reingeniería de los procesos, pudiendo agregar a distintos roles nuevas actividades, o suprimir cualquiera de las ya asignadas de manera automática dentro del sistema, como veremos más adelante.

Se debe acotar que en la iteración 5, los recordatorios pueden ser manejados por todos los roles del sistema.

3.1.1 Iteraciones Correspondientes al Módulo de Correspondencia

En este apartado se explicará el desarrollo de todas las funcionalidades necesarias para construir el módulo de correspondencia.

3.1.1.1 Iteración 1

La iteración 1 contempla las funcionalidades de la correspondencia de entrada.

3.1.1.1.1 Planificación de la iteración 1

Para la planificación de todas las iteraciones, se utilizan historias de usuarios definidas por el representante de BANAVIH Ing. Carlos Moreno.

En la Tabla 3.1 se especifican las historias de usuarios de la primera iteración:

Historias de Usuarios para la 1era. Iteración		
Fecha	**Usuario**	**Descripción**
01/10/2008 – 08/10/2008	Ing. Carlos Moreno	Registrar una correspondencia de entrada con los campos: Nro. de Documento, Fecha del Documento, Monto, Tipo de Documento, Entidad Federal, Organismo, Materia y Asunto.
09/10/2008 – 24/10/2008	Ing. Carlos Moreno	Consultar Correspondencias de Entrada por los parámetros: campos de registro de la correspondencia. Cabe destacar,

		que todas las búsquedas en el sistema deberán poder mostrar un reporte de los resultados en formato *PDF*.
27/10/2008 – 31/10/2008	Ing. Carlos Moreno	Modificar los datos asociados a una Correspondencia de Entrada, indicando los campos que hayan sido cambiados.

Tabla 3.1. Formato de Historia de Usuarios para la Primera Iteración perteneciente al rol Recepcionista de Correspondencia (Fuente: Elaborado por el representante de BANAVIH).

3.1.1.1.2 Diseño de la iteración 1

El diseño del sistema se basa en una arquitectura MVC.

Para esta y todas las iteraciones, el diseño consiste en el Diagrama de Entidad/Relación, Modelo Lógico y Diagrama de Clases, dependiendo de si es necesario su estudio. Estos diagramas son utilizados como estándares dentro del banco para modelar el negocio.

En todas las iteraciones, los Diagramas de Entidad/Relación y los Modelos Lógicos, serán tratados incrementalmente, incorporando las entidades, relaciones y tablas, de acuerdo a la iteración que se esté manejando, con el fin de obtener los diagramas finales en la última iteración. Con respecto a los diagramas de clases, las clases serán agregadas dependiendo del módulo que se esté tratando, ya sea de correspondencia, puntos de cuenta ó de administración; es decir, si el módulo que se está explicando es puntos de cuenta, no se anexará la clase correspondencia, a menos que sea necesario.

El diseño de la iteración 1 consiste en el Diagrama de Entidad/Relación, Modelo Lógico y Diagrama de Clases.

A continuación se describirán los diagramas de la primera iteración:

- **Diagrama de Entidad/Relación**

El diagrama Entidad/Relación, se enfoca en aquellos aspectos importantes del negocio para desarrollar las funcionalidades pertenecientes a cada iteración, en este caso la primera. En la iteración 1, se requieren las siguientes entidades: Usuario, Correspondencia, Materia y Rol; y las relaciones son: clasifica, maneja y desempeña. En la Figura 3.4 se muestran estas entidades y relaciones.

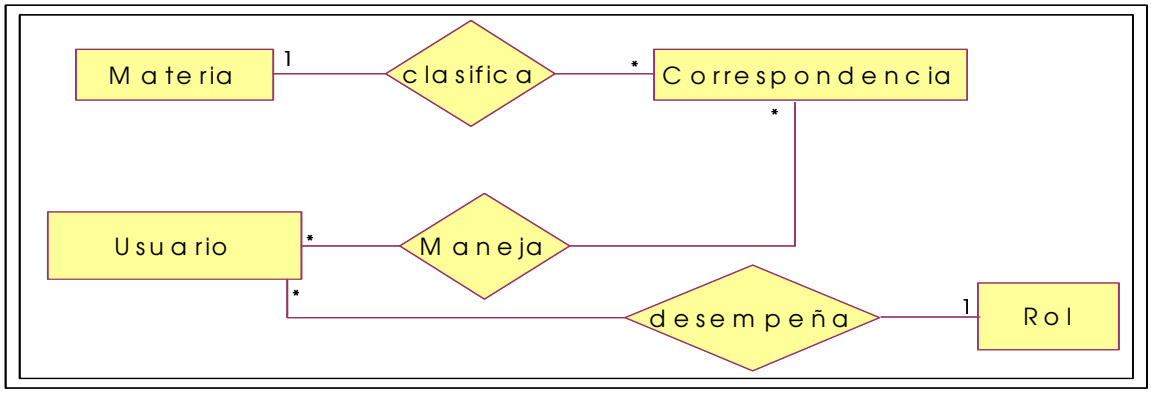

Figura 3.4. Diagrama de Entidad/Relación, perteneciente a la Primera Iteración para los Roles "Recepcionista Correspondencia" y "Coordinador de Presidencia" (Fuente: Elaboración propia).

La Figura 3.4 muestra que la entidad Usuario se utiliza para poder llevar el registro de los usuarios y desempeña un rol que controla el acceso al sistema. Cada usuario maneja múltiples correspondencias, las cuales contienen información necesaria para cumplir con los requerimientos de inserción, consulta y modificación de las mismas. Cada correspondencia se clasifica por materia (por ejemplo, correspondencia de tipo presupuesto) y una materia puede clasificar múltiples correspondencias.

- **Modelo Lógico**

Este modelo muestra la estructura lógica de la base de datos, que permitirá realizar el manejo de la persistencia para el soporte de las funcionalidades especificadas en las historias de usuarios. La estructura lógica de la primera iteración se compone de las siguientes tablas: correspondencia, materia, gerencia, rol, tipoorganismo, usuario, estado y tipodocumento (Ver figura 3.5).

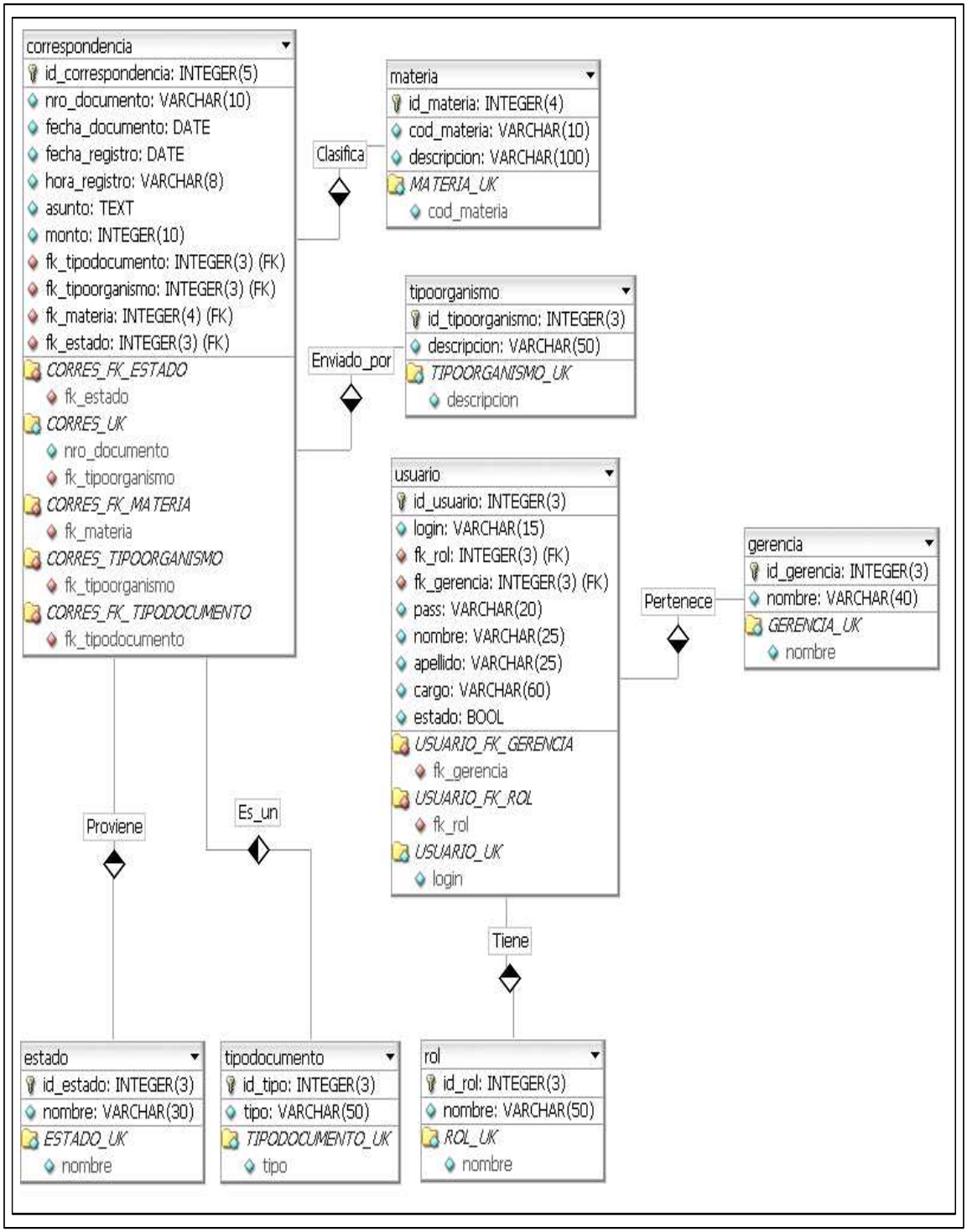

Figura 3.5. Modelo Lógico perteneciente a la Primera Iteración para los Roles "Recepcionista Correspondencia" y "Coordinador de Presidencia" (Fuente: Elaboración propia).

Las tablas mostradas en la Figura 3.5 son:

- correspondencia: Posee la información de todas las correspondencias.

Esta tabla contiene los siguientes campos:

- id_correspondencia: clave primaria que indica que el registro de la correspondencia es único en el sistema.
- Nro_documento: número que está en el físico de la correspondencia.
- Fecha_documento: fecha en la que el ente adscrito elaboró la correspondencia.
- Fecha_registro: fecha en la que se registró la correspondencia en el sistema.
- Hora_registro: hora en la que se registró la correspondencia.
- Asunto: pequeño resumen que indica de lo que trata la correspondencia.
- Monto: cantidad de Bsf. que puede tener asociado o no la correspondencia. Este campo no es obligatorio, sólo se inserta si la correspondencia posee un monto, ya que es utilizado como un nemotécnico para recordarla.
- Claves ajenas: fk_tipoorganismo (identificador del organismo del que proviene la correspondencia), fk_tipomateria (identificador de la materia de la correspondencia), fk_tipodocumento (identificador del tipo de documento de la correspondencia, por ejemplo: oficio, circular, etc.), fk_estado (identificador de la entidad federal de la que provino la correspondencia).

- Índices: CORRES_FK_ESTADO, CORRES_UK (indica que el nro_documento y fk_tipoorganismo, conjuntamente, deben ser únicos en el sistema), CORRES_FK_MATERIA, CORRES_FK_TIPOORGANISMO, CORRES_FK_TIPODOCUMENTO.

o materia: Posee la información de todas las materias.

Esta tabla contiene los siguientes campos:

- id_materia: clave primaria que indica que el registro de la materia es único en el sistema.
- Cod_materia: código de la materia.
- Descripción: nombre de la materia.
- Índice: MATERIA_UK (indica que el código de la materia debe ser único en el sistema).

o tipoorganismo: Posee la información de todos los organismos.

Esta tabla contiene los siguientes campos:

- id_tipoorganismo: clave primaria que indica que el registro del organismo es único en el sistema.
- Descripción: nombre del organismo.
- Índices: TIPOORGANISMO_UK (indica que la descripción del organismo debe ser único en el sistema.)

o gerencia: Contiene la información de las gerencias de la institución.

Esta tabla contiene los siguientes campos:

- Id_gerencia: clave primaria que indica que el registro de la gerencia es único en el sistema.
- Nombre: nombre de la gerencia.

- Índice: GERENCIA_UK (índica que el nombre de la gerencia debe ser único en el sistema).

o usuario: Contiene la información de los usuarios en el sistema.

Esta tabla contiene los siguientes campos:

- Id_usuario: clave primaria que indica que el registro del usuario es único en el sistema.
- Login: nombre del usuario para el acceso al sistema.
- Pass: clave de usuario para entrar al sistema.
- Nombre: nombre del usuario.
- Apellido: apellido del usuario.
- Cargo: cargo que desempeña en la institución.
- Estado: indica si el usuario se encuentra en modo inactivo o activo en el sistema.
- Claves Ajenas: fk_gerencia (identificador de la gerencia del usuario), fk_rol (identificador de la permisología del usuario para el acceso al sistema).
- Índices: USUARIO_FK_GERENCIA, USUARIO_FK_ROL, USUARIO_UK (indica que el login del usuario debe ser único en el sistema).

o tipodocumento: Contiene información sobre el tipo de documento.

Esta tabla contiene los siguientes campos:

- id_tipo: clave primaria que indica que el registro del tipo de documento es único en el sistema.
- tipo: descripción del tipo de documento.
- Índice: TIPODOCUMENTO_UK (indica que el tipo del documento debe ser único en el sistema).

- rol: Posee la permisología de acceso al sistema.

Esta tabla contiene los siguientes campos:

- id_rol: clave primaria que indica que el registro del rol es único en el sistema.
- nombre: descripción del rol.
- índice: ROL_UK (indica que el nombre del rol es único en el sistema).

- estado: Posee la entidad federal de donde proviene la correspondencia.

Esta tabla contiene los siguientes campos:

- id_estado: clave primaria que indica que el registro de la entidad federal es único en el sistema.
- nombre: descripción del estado.
- Índice: ESTADO_UK (indica que el nombre del estado es único en el sistema).

- **Diagrama de Clases**

En el diagrama de clases, de ésta y todas las iteraciones, se mostrarán solamente aquellas clases pertenecientes a la capa "modelo" del patrón MVC, las cuales contienen la lógica de negocio de la aplicación, así como las clases que manejan la conexión con la base de datos. Las clases pertenecientes a otras capas y/o otras clases auxiliares, no serán tomadas en cuenta, debido a que los diagramas se centran sólo en la lógica del negocio del caso de estudio.

La figura 3.6 muestra el diagrama de clases de la iteración 1.

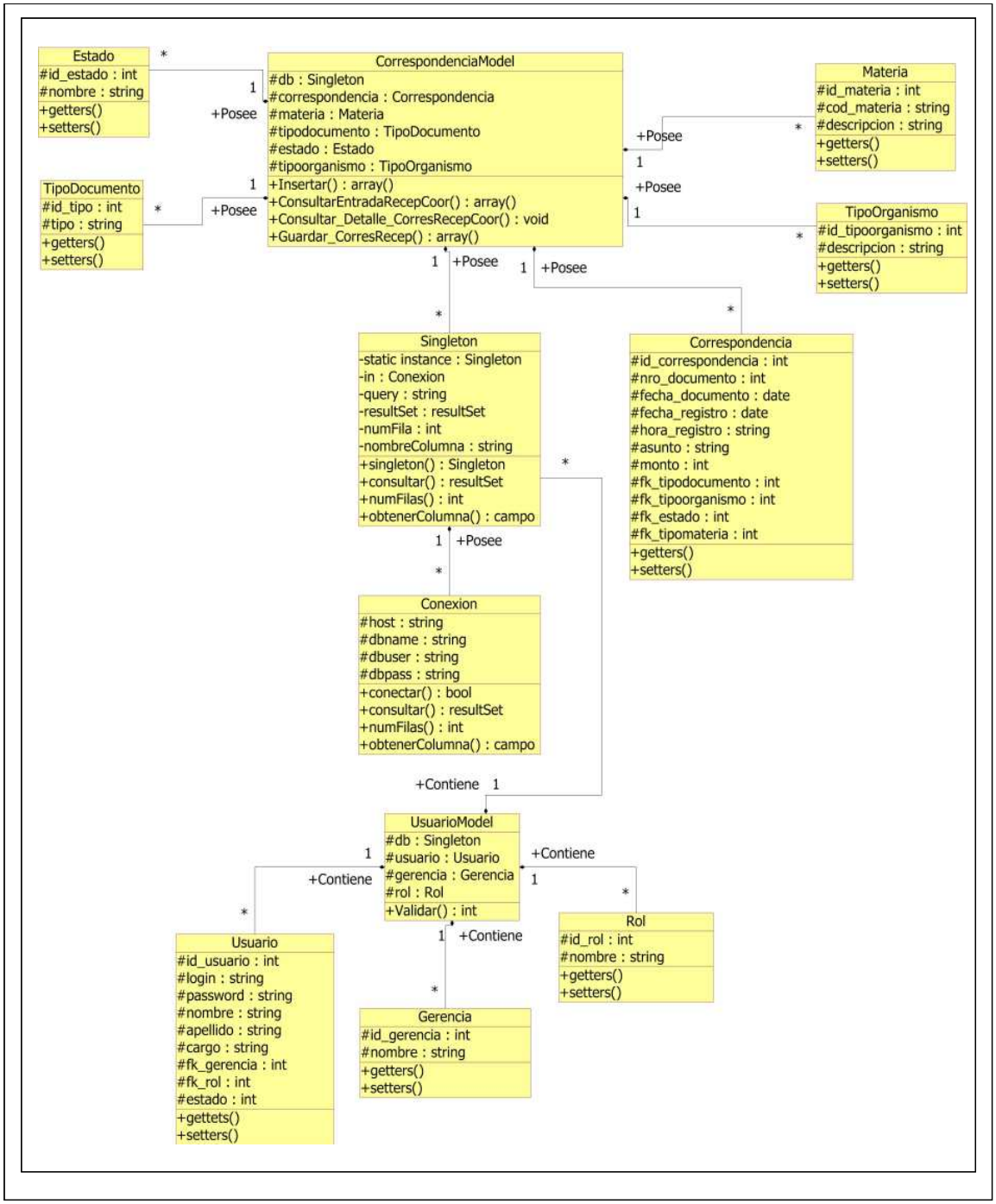

Figura 3.6. Diagrama de Clases, perteneciente a la Primera Iteración del Módulo de Correspondencia para el Rol Recepcionista Correspondencia (Fuente: Elaboración propia).

En la Figura 3.6 se aprecian las clases CorrespondenciaModel y UsuarioModel de la iteración 1, para manejar las correspondencias de entrada y la validación del usuario en el sistema, así como también, las clases que manipulan la conexión con la base de datos (*Singleton* y Conexion). Todas las clases de la capa modelo se encuentran relacionadas con otras clases contenedoras (estas almacenan exclusivamente información de entidades como prioridad, tipodocumento, etc.), a través de una relación de composición (1 a muchos (*)). Esta idea no será especificada en las demás iteraciones para evitar redundancia de explicación.

En este apartado sólo se explicarán las clases del Modelo CorrespondenciaModel y UsuarioModel, *Singleton* y Conexión, debido a que los atributos de las demás clases ya se describieron en el modelo lógico. A continuación se describirán estás clases:

- CorrespondenciaModel: Define la lógica del negocio para Correspondencia, posee las funciones y acciones correspondientes para la solución de los requerimientos del manejo de correspondencia.

Esta clase contiene los siguientes atributos:

- db: instancia de la clase *Singleton*.
- correspondencia: instancia de la clase Correspondencia.
- materia: instancia de la clase Materia.
- tipodocumento: instancia de la clase TipoDocumento.
- estado: instancia de la clase Estado.
- tipoorganismo: instancia de la clase TipoOrganismo.

Los métodos de esta clase son los siguientes:

- Insertar: Registra una nueva correspondencia.
- ConsultarEntradaRecepCoor: Permite consultar la correspondencia de entrada, a través de los criterios de búsquedas indicados por el usuario.
- Consultar_Detalle_CorresRecepCoor: Consulta los campos de la correspondencia de entrada.
- Guardar_CorresRecep: Actualiza los campos de la correspondencia de entrada.

o UsuarioModel: Define la lógica del negocio para el Usuario, posee las funciones y acciones correspondientes para la solución de los requerimientos de la gestión de usuarios.

Esta clase contiene los siguientes atributos:

- db: instancia de la clase *Singleton*.
- usuario: instancia de la clase Usuario.
- gerencia: instancia la clase Gerencia.
- rol: instancia de la clase Rol.

El método de esta clase es el siguiente:

- Validar: Función que permite el acceso del usuario al sistema, si éste está registrado en el mismo.

o Singleton: Mantiene una única instancia de la conexión a la base de datos.

Esta clase contiene los siguientes atributos:

- static instance: instancia privada y estática de la clase *Singleton*.
- in: instancia privada de la clase Conexión.
- query: contiene la consulta que se ejecutará en la base de datos.
- resultSet: contiene la ejecución de la consulta en el sistema.
- numFila: Devuelve el número de filas del resultSet.
- nombreColumna: Contiene el nombre de las columnas de los campos del resultSet.

Los métodos de esta clase son los siguientes:

- *singleton*: Permite acceder a la clase *Singleton*.
- consultar: Permite ejecutar un *query* en la base de datos.
- numFilas: Devuelve el número de filas del resultSet.
- obtenerColumna: Obtiene el valor de la columna indicada del resultSet.

o Conexión: Define el acceso a la base de datos.

Esta clase contiene los siguientes atributos:

- host: nombre o dirección IP de la base de datos.
- dbname: nombre de la base de datos.
- dbuser: nombre de usuario para acceder a la base de datos.
- dbpass: clave de acceso a la base de datos

Las funciones de esta clase son las siguientes:

- conectar: permite la conexión con la base de datos.

- consultar, numFilas y obtenerColumna, funcionan de la misma manera que las funciones que tienen el mismo nombre en la clase *Singleton.*

3.1.1.1.3 Codificación de la iteración 1

En la primera iteración, la codificación se basa en (Ver *anexos A1, A2, A3, A4, A5, A6 y A7*):

- Funciones del modelo de correspondencia que permiten insertar, consultar y actualizar los campos de la correspondencia de entrada.
- Código de los reportes que permiten generar la información en *PDF* de las correspondencias consultadas.
- Función del modelo de Usuario que permite verificar si el usuario se encuentra registrado y acceder al sistema.
- Funciones de la clase *Singleton* y Conexión que permiten instanciar y establecer la conexión con la base de datos.

En el anexo de este documento, se muestra el código de las funciones y acciones más importantes de las iteraciones. Aquellas funciones y acciones que presenten un comportamiento similar a la de iteraciones anteriores, no se mostrarán para no caer en redundancia de ideas.

3.1.1.1.4 Pruebas de Aceptación y Resultados del Sistema de la iteración 1

El equipo de trabajo está conformado por el Líder de Proyecto Ing. Néstor Lugo, el representante de BANAVIH Carlos Moreno y los desarrolladores: Oscar Ruiz y Neldy Corredor.

Para las pruebas de aceptación fueron coordinadas reuniones con el Ing. Carlos Moreno, quien fue el encargado de supervisar las pruebas unitarias ejecutadas por el equipo de desarrolladores, con la finalidad de verificar que la aplicación cumpliera con todos los requisitos especificados en las historias de usuarios.

Se utiliza el formato de realización de las pruebas de aceptación suministrado por BANAVIH. La tabla 3.2 contiene el formato establecido por el representante de BANAVIH.

Nombre del Sistema							
Módulo:							
Historia de usuarios							
Caso de Prueba:							
Versión de caso de prueba:		**Fecha ejecución:**					
Nombre del Probador:							
Precondiciones:							
Para la ejecución del Caso de Prueba:							
Paso	**Condición**	**Valor(es)**	**Resultado esperado**	**Resultado Obtenido**			
Observaciones del Caso de Prueba							
Decisión de Aprobación de Caso de Prueba:		**Aprobó:**		**Falló:**			
Fecha de Aprobación del Caso de Prueba:				**Aprobado por:**			
Ajustado por:			**Fecha de Ajuste:**				
Prueba de Ajuste, en fecha:		**Decisión de Aprobación de Ajuste:**	**Aprobó:**		**Falló:**		

Tabla 3.2. Formato de Caso de Prueba. (Fuente: Formato establecido por el representante de BANAVIH).

El formato de la tabla 3.2 consiste en: Nombre del Sistema, Módulo que se trata en la prueba, Historia de usuarios usada para verificar el cumplimiento de la funcionalidad, Caso de prueba que indica cuál será la prueba a ejecutar, Versión de caso de prueba que indica si se ha hecho o no otra prueba anterior sobre el mismo caso, Fecha de ejecución en la que se ejecutará la prueba, Nombre del probador: nombre de los

desarrolladores que ejecutarán la prueba, Precondiciones: condiciones necesarias antes de ejecutar la prueba, Paso que debe realizarse para ejecutar la prueba, Condición necesaria para que la prueba se ejecute, Valor(es) de la condición, Resultado Esperado en la prueba, Resultado Obtenido al ejecutar la prueba, Observaciones del caso de prueba, Decisión de Aprobación de Caso de Prueba: Si se ejecutó con éxito o no la prueba (esto lo determina el representante de BANAVIH), Fecha de aprobación: fecha en la que se aprobó la ejecución de la prueba, Aprobado por: responsable de aprobar la prueba, Ajustado por: desarrollador que ajustó la prueba, Fecha de ajuste y Prueba de Ajuste, en fecha: indica la fecha en la que se repitió la prueba para su ajuste y Decisión de aprobación de ajuste (Si se aprobó o no la ejecución).

Los formatos se mostrarán en el presente documento, sólo en caso de que alguna funcionalidad arroje errores.

Cabe destacar que los errores encontrados en las funcionalidades de cada una de las iteraciones, se han debido corregir el mismo día de las pruebas unitarias, debido a las políticas de la organización.

Para esta iteración, los desarrolladores realizaron todas las pruebas con las correspondencias que se consideraron pertinentes y el único error que se presentó en el conjunto de la muestra, fue en el momento de registrar una correspondencia con un monto máximo de diez dígitos (ver Tabla 3.3).

Una vez corregido el error de la iteración, el Ing. Carlos Moreno quedó conforme con todos los resultados obtenidos en el sistema.

A continuación, en la tabla 3.3, se presenta la prueba de la funcionalidad Registrar Correspondencia:

- **Registrar Correspondencia**

Nombre del Sistema **Sistema de Información para el Manejo de Correspondencias y Puntos de Cuenta**							
Módulo:	Correspondencia						
Historia de Usuarios	Registrar una correspondencia de entrada con los campos: Nro. de Documento, Fecha del Documento, Monto, Tipo de Documento, Entidad Federal, Organismo, Materia y Asunto.						
Caso de Prueba:	Verificar que el Registro de una Correspondencia de Entrada se realice con los siguientes campos: Nro. de Documento, Fecha del Documento, Monto, Tipo de Documento, Entidad Federal, Organismo, Materia y Asunto.						
Versión de caso de prueba:	Versión 1	**Fecha ejecución:**	03/11/2008				
Nombre del Probador:	Neldy Corredor						
Precondiciones: Los campos de entrada no puede ser vacíos y el monto no puede ser negativo ni un carácter							
Para la ejecución del Caso de Prueba:							
Paso	**Condición**	**Valor(es)**	**Resultado esperado**	**Resultado Obtenido**			
Ingresar los datos correspondientes a la Correspondencia de Entrada	El número del documento y el organismo no pueden repetirse conjuntamente en el sistema	Numero del Documento ϵ (0..9999999999), Monto ϵ (0..9999999999)	Debe mostrar el mensaje: "Se ha insertado una nueva Correspondencia...", y el detalle con la datos que se han insertado: Número del documento, Asunto, Tipo de documento y el organismo del que proviene la correspondencia, en caso de éxito.	La Funcionalidad desplegó el mensaje "Error inesperado del servidor…", cuando se le introdujo un monto de 10 dígitos, por lo que fue necesario corregir el código para que registrara la correspondencia correctamente.			
Observaciones del Caso de Prueba							
Decisión de Aprobación de Caso de Prueba:		**Aprobó:**		**Falló:** √			
Fecha de Aprobación del Caso de Prueba:				**Aprobado por:** Ing. Carlos Moreno			
Ajustado por:	Neldy Corredor			**Fecha de Ajuste:**	03/11/2008		
Prueba de Ajuste, en fecha:	03/11/2008	**Decisión de Aprobación de Ajuste:**		**Aprobó:**	√	**Falló:**	

Tabla 3.3. Formato de Caso de Prueba perteneciente a la funcionalidad Registrar Correspondencia del Rol "Recepcionista Correspondencia" (Fuente: Formato establecido por el representante de BANAVIH).

Las fotos del sistema de todas las iteraciones, contienen sólo las pantallas de las funcionalidades más importantes de cada una de las iteraciones.

La Figura 3.7 muestra la interfaz de acceso al sistema.

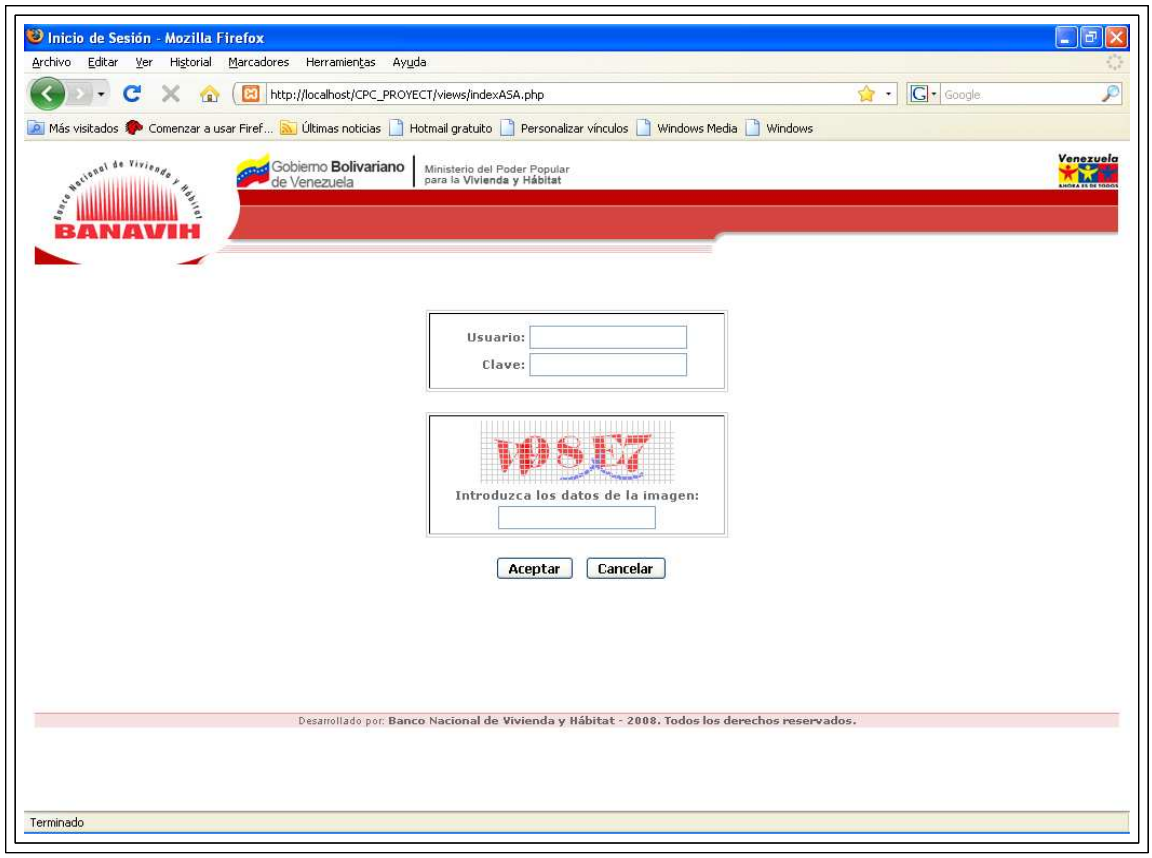

Figura 3.7. Foto de la Interfaz de Acceso al Sistema (Fuente: Elaboración propia).

En la Figura 3.7 se puede apreciar que el usuario debe introducir el nombre de usuario, la clave de acceso y los datos de la imagen.

La Figura 3.8 muestra la ejecución del registro de la correspondencia, la cual ha sido exitosa.

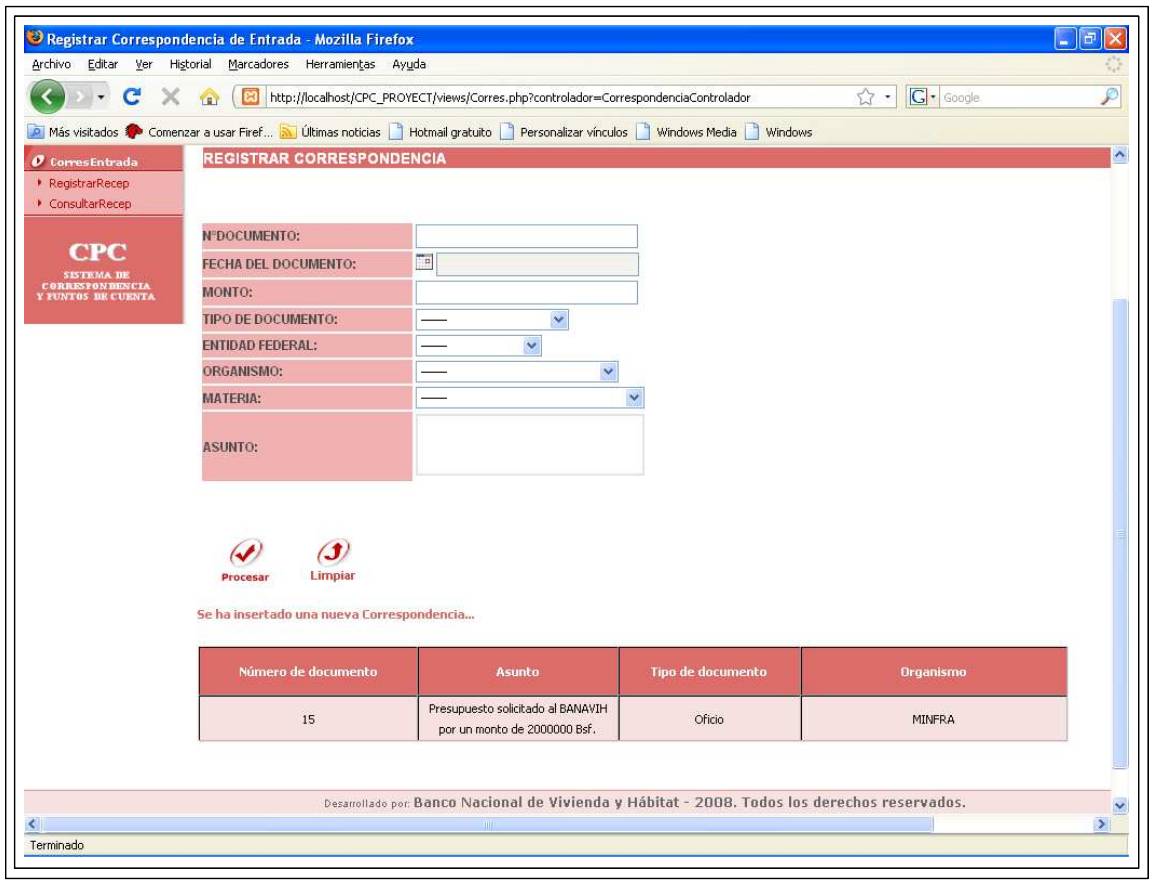

Figura 3.8. Foto de la ejecución de la funcionalidad Registrar Correspondencia
(Fuente: Elaboración propia).

3.1.1.2 Iteración 2

La iteración 2 se basa en las funcionalidades de la correspondencia de salida.

3.1.1.2.1 Planificación de la iteración 2

La planificación de esta iteración, se basa en las historias de usuarios de la tabla 3.4 que se especifica a continuación:

Historias de Usuarios para la 2da. Iteración		
Fecha	**Usuario**	**Descripción**
04/11/2008 – 05/11/2008	Ing. Carlos Moreno	Consultar Correspondencias de Salida.
06/11/2008 - 10/11/2008	Ing. Carlos Moreno	Modificar los datos asociados a una Correspondencia de Salida.
11/11/2008 - 12/11/2008	Ing. Carlos Moreno	Cargar y Mostrar la imagen digitalizada de la página principal de cada Correspondencia de Salida, de manera que se refleje la firma, el sello y las instrucciones de la misma.
13/11/2008 - 14/11/2008	Ing. Carlos Moreno	Registrar expediente con los siguientes campos: Código del expediente y Materia a la que pertenece.
17/11/2008 - 18/11/2008	Ing. Carlos Moreno	Consultar el expediente por los siguientes parámetros: campos y fecha de registro del expediente.
19/11/2008 - 21/11/2008	Ing. Carlos Moreno	Clasificar la correspondencia por expediente (asociar correspondencias a un expediente)

Tabla 3.4. Formato de Historia de Usuarios para la Segunda Iteración perteneciente al Rol "Unidad de Documentación" (Fuente: Elaborado por el representante de BANAVIH).

3.1.1.2.2 Diseño de la iteración 2

El Ing. Carlos Moreno, indicó la necesidad de establecer una diferencia entre las correspondencias de entrada y las de salida, por lo que la consulta de correspondencias será dividida en dos tipos: correspondencias de entrada y de salida, utilizando para ello un estatus de entrada y otro de salida, que indique el estado de la correspondencia. En este caso la funcionalidad Consultar Correspondencia de Entrada perteneciente al Rol "Recepcionista de Correspondencia" fue refactorizada, para poder realizar la búsqueda de las correspondencias con estatus de entrada. Esto será explicado en la sección de codificación de esta iteración.

En ésta, y en posteriores iteraciones, se colocará en un recuadro, resaltado en rojo, las entidades, los campos y los métodos que vayan siendo agregados (cuando sea necesario) a los diagramas, y sólo serán descritas las entidades, campos, tablas, clases y métodos que estén marcados en rojo sobre los diagramas.

El diseño de la segunda iteración se basa en el Diagrama de Entidad/Relación, Modelo Lógico y Diagrama de Clases.

A continuación se describen los Diagrama de Entidad/Relación, Modelo Lógico y Diagrama de Clases de la iteración 2:

- **Diagrama de Entidad/Relación**

El Diagrama de Entidad/Relación de la Iteración 2 muestra las entidades y relaciones de la iteración 1, y adicionalmente incorpora la entidad Expediente.

La Figura 3.9 muestra el diagrama de Entidad/Relación de la iteración 2.

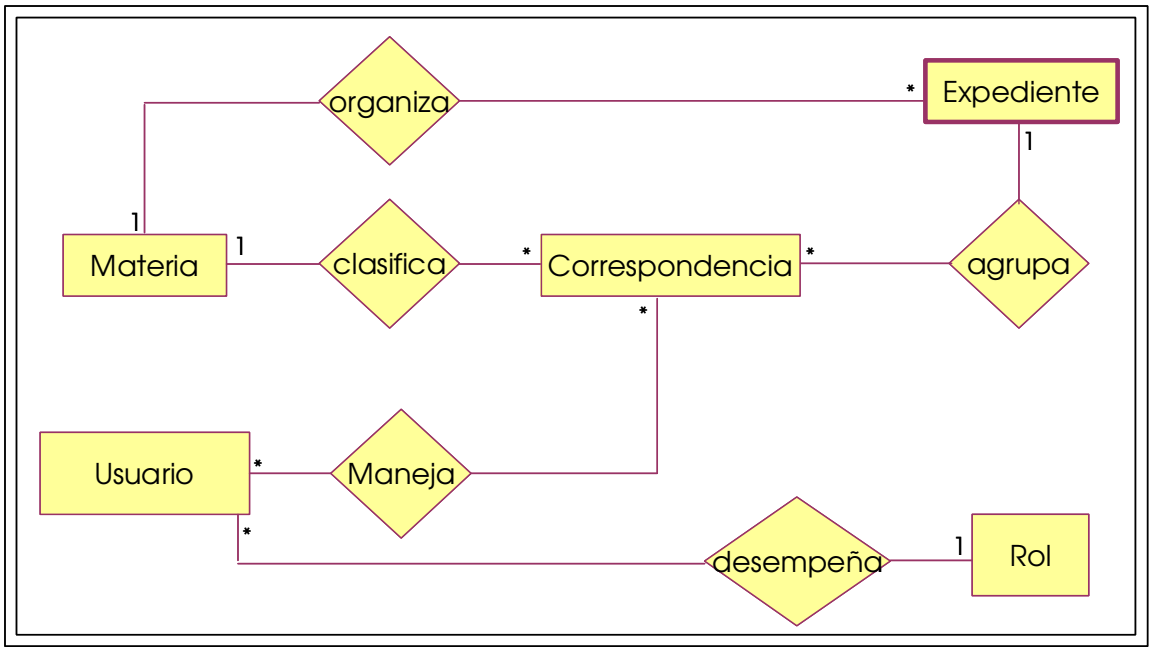

Figura 3.9. Diagrama de Entidad/Relación, perteneciente a la Segunda Iteración para el Rol "Unidad de Documentación" (Fuente: Elaboración propia).

La Figura 3.9 indica que la entidad expediente agrupa un conjunto de correspondencias; a su vez, cada correspondencia es agrupada por expediente; un expediente está organizado por Materia, la que a su vez, organiza múltiples expedientes.

- **Modelo Lógico**

El modelo lógico de la iteración 2 se basa en las tablas mostradas en la iteración 1, con campos de salida adicionales, los cuales son: fk_gerencia_destino, fk_expediente, fk_estatus_correspondencia, fk_prioridad, responsable, dirección_imagen, fecha_registro_salida, hora_registro_salida e instrucciones para la tabla correspondencia. Además se añaden tres tablas: estatus_correspondencia, prioridad y expediente, las cuales serán descritas posteriormente.

La figura 3.10 muestra el modelo lógico de la iteración 2.

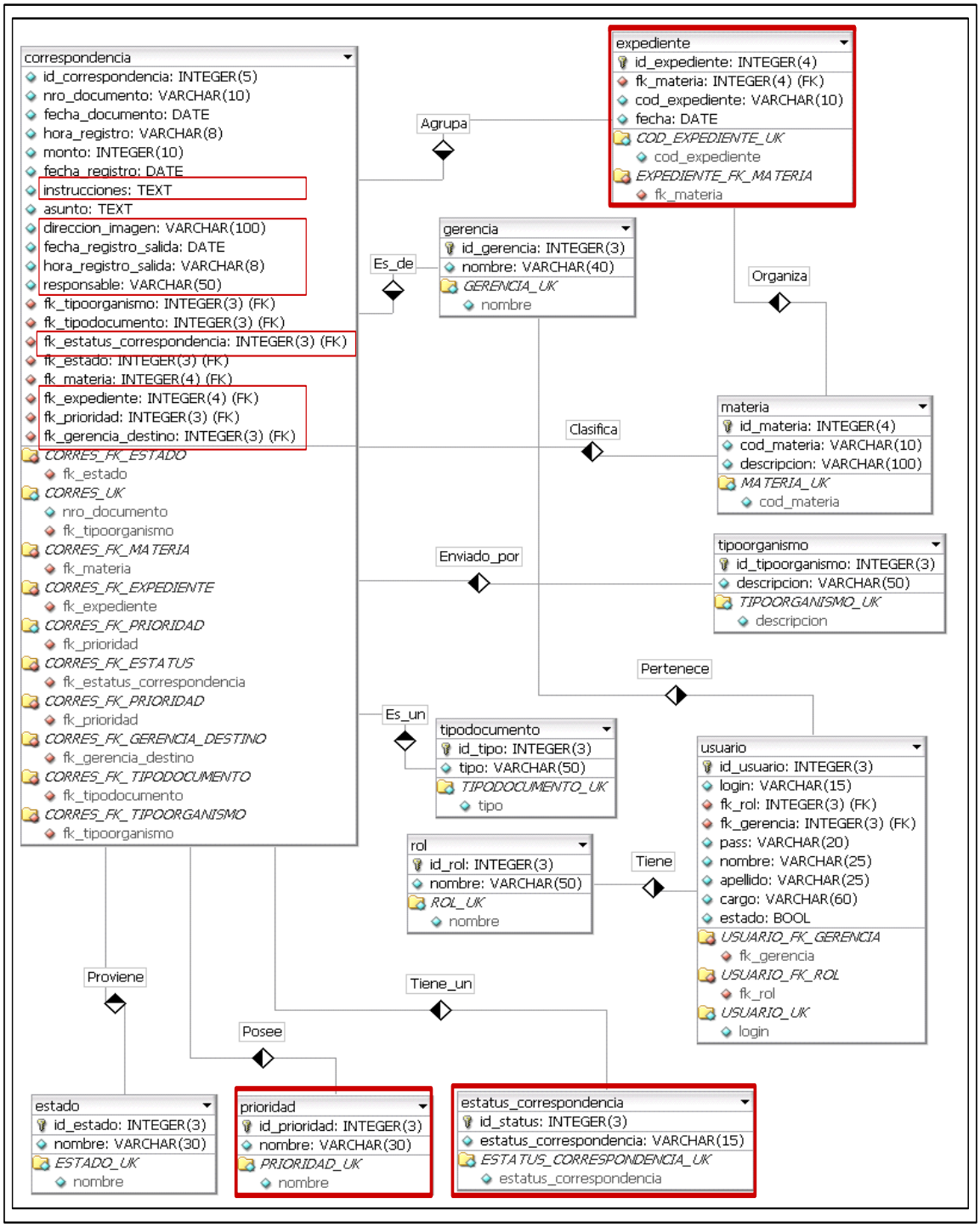

Figura 3.10. Modelo Lógico perteneciente a la Segunda Iteración para el Rol "Unidad de Documentación" (Fuente: Elaboración propia).

A continuación se describirá lo que está resaltado en rojo en la figura 3.10:

- En la tabla correspondencia se incorporan los siguientes campos:
  - Instrucciones: posee las indicaciones del Presidente.
  - Direccion_imagen: contiene la dirección de la imagen de la correspondencia digitalizada.
  - Fecha_registro_salida: fecha de registro de la salida, una vez que la unidad de documentación registre la información de salida en el sistema.
  - Hora_registro_salida: hora de registro de la salida, una vez que la unidad de documentación registre la información de salida en el sistema.
  - Responsable: es el encargado de la correspondencia.
  - Claves ajenas: fk_estatus_correspondencia (identificador del estatus de la correspondencia), fk_expediente (identificador del expediente al que está asociada la correspondencia), fk_prioridad (identificador de la prioridad que tiene la correspondencia) y fk_gerencia_destino (identificador de la gerencia a la que va destina la correspondencia).
  - Índices: CORRES_FK_EXPEDIENTE, CORRES_FK_PRIORIDAD, CORRES_FK_GERENCIA_DESTINO y CORRES_FK_ESTATUS.
- Expediente: Contiene la información de todos los expedientes.
  - Id_expediente: clave primaria que indica que el registro del expediente es único en el sistema.
  - Cod_expediente: código del expediente.
  - Fecha: fecha en la que se registra el expediente.

- Clave ajena: fk_materia(identificador de la materia asociada al expediente).
- Índice: COD_EXPEDIENTE_UK (indica que el código del expediente es único en el sistema).

o Prioridad: Contiene la información de las prioridades, ya sea urgente o normal para las correspondencias de salida.

Esta tabla contiene los siguientes campos:

- Id_prioridad: clave primaria que indica que el registro de la prioridad es único en el sistema.
- Nombre: nombre de la prioridad.
- Índice: PRIORIDAD_UK (indica que la prioridad es única en el sistema).

o Estatus_correspondencia: permite establecer una separación lógica entre las correspondencias de entrada y las de salida, utilizando valores de entrada y salida.

Esta tabla contiene los siguientes campos:

- Id_status: clave primaria que indica que el registro del estatus de la correspondencia es único en el sistema.
- Estatus_correspondencia: nombre del estatus de la correspondencia, que puede ser entrada o salida.
- Índice: ESTATUS_CORRESPONDENCIA_UK (indica que el estatus de la correspondencia es único en el sistema).

- **Diagrama de Clases**

En el Diagrama de Clases de la segunda iteración se tomaron en cuenta las clases pertenecientes a la primera iteración y se agregaron cuatro clases: estatus_correspondencia y prioridad que se incluirán en la capa del modelo del patrón MVC para el módulo de correspondencia, las cuales serán manejadas dentro de la clase CorrespondenciaModel y las clases expediente y ExpedienteModel (maneja la clase expediente).

La Figura 3.11 muestra el diagrama de clases de la iteración 2.

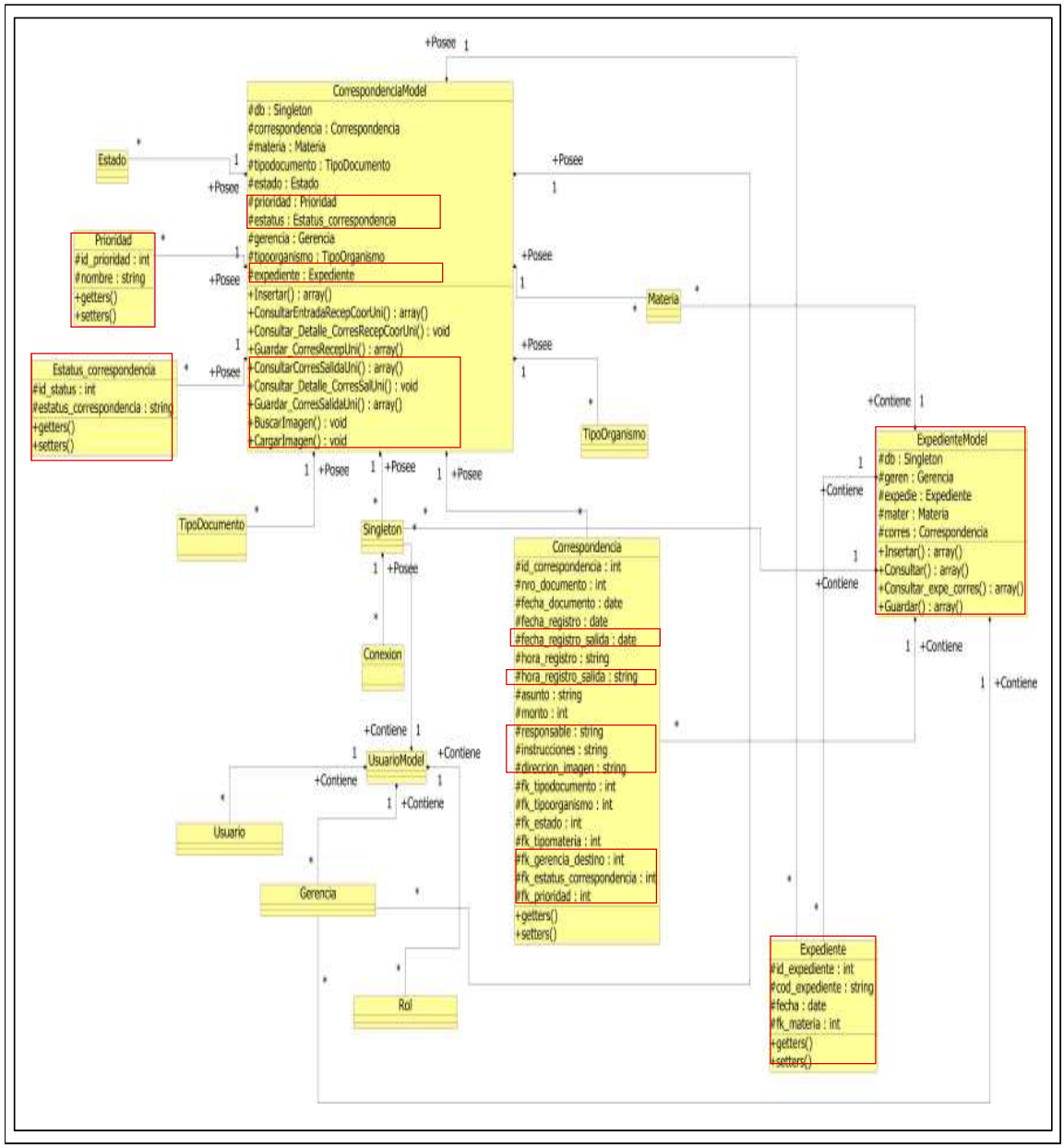

Figura 3.11. Diagrama de Clases, perteneciente a la Segunda iteración del Módulo de Correspondencia para el Rol "Unidad de Documentación" (Fuente: Elaboración propia).

A continuación se describen los métodos de la clase CorrespondenciaModel y la clase ExpedienteModel.

Los métodos agregados son los siguientes:

- ConsultarCorresSalidaUni: Permite consultar la correspondencia de salida.
- Consultar_Detalle_CorresSalUni: Permite consultar el detalle de la correspondencia de salida.
- Guardar_CorresSalidaUni: Permite actualizar la correspondencia de salida.
- BuscarImagen: Consulta la dirección de la imagen de la correspondencia de salida.
- CargarImagen: Inserta la dirección de la imagen en la correspondencia de salida.

La clase ExpedienteModel contiene los siguientes atributos:

- db: instancia de la clase *Singleton*.
- geren: instancia de la clase Gerencia.
- expedie: instancia de la clase Expediente.
- mater: instancia de la clase Materia.
- corres: instancia de la clase Correspondencia.

Los métodos de la clase ExpedienteModel son los siguientes:

- Insertar: registra un nuevo expediente.
- Consultar: consulta el expediente, a través de los criterios de búsqueda que indique el usuario.
- Consultar_expe_corres: consulta el detalle de los campos del expediente y las correspondencias asociadas al mismo.
- Guardar: asigna las correspondencias al expediente.

3.1.1.2.3 Codificación de la iteración 2

Para la iteración 2, se toma en cuenta otras funciones del Modelo de Correspondencia y del Modelo de Expediente, que son importantes para el desarrollo de esta iteración (Ver *anexos A8, A9, A10, A11y A12*).

En la segunda iteración, la codificación se basa en:

- Refactorización de la inserción y consulta de una correspondencia de entrada con el estatus Entrada, debido a que en la primera iteración no se utilizó estatus para la correspondencia.
- Funciones adicionales del modelo de Correspondencia que permiten la consulta y actualización de las correspondencias de salida.
- Acciones que permiten consultar y actualizar la imagen de la correspondencia de salida.
- Funciones del modelo Expediente que permiten registrar, consultar y asociar correspondencias a un expediente.

3.1.1.2.4 Pruebas de Aceptación y Resultados del Sistema para la iteración 2

En estas pruebas se presentó un error cuando al momento de asignar correspondencias a un expediente, y posteriormente al consultar las mismas por el código del expediente mencionado, no arrojaba ningún resultado; es decir, no había *match* (asociación) entre correspondencias y el expediente a las cuales fueron previamente asignadas.

El Ing. Carlos Moreno verificó que el error fue solucionado, por lo que quedó conforme con todos los resultados obtenidos en el sistema.

A continuación se muestran los formatos de pruebas (ver tabla 3.5 y tabla 3.6) utilizados para comprobar el funcionamiento de la consulta de las correspondencias de salida y agregar correspondencias a un expediente.

- **Consultar Correspondencia de Salida**

Nombre del Sistema **Sistema de Información para el manejo de Correspondencias y Puntos de Cuenta**				
Módulo:	Correspondencia			
Historia de usuarios	Consultar Correspondencias de Salida.			
Caso de Prueba:	Verificar que la consulta de una Correspondencia de Salida se haga con los siguientes campos: Número del Documento, Fecha Desde Entrada, Fecha Hasta Entrada, Fecha Desde Salida, Fecha Hasta Salida, Monto, Tipo de Documento, Gerencia Destino, Entidad Federal, Organismo, Materia, Código del Expediente y Estatus de Correspondencia.			
Versión de caso de prueba:	Versión 1	**Fecha ejecución:**	24/11/2008	
Nombre del Probador:	Neldy Corredor			
Precondiciones: El número del documento y el monto debe ser entero.				
Para la ejecución del Caso de Prueba:				
Paso	**Condición**	**Valor(es)**	**Resultado esperado**	**Resultado Obtenido**
Filtrar la Búsqueda de la Correspondencia de Salida por los ítems: Número del Documento, Fecha Desde Entrada, Fecha Hasta Entrada, Fecha Desde Salida, Fecha Hasta Salida, Monto, Tipo de Documento, Gerencia Destino, Entidad Federal, Organismo, Materia, Código del Expediente y Estatus de Correspondencia.	El número del documento y el Monto debe ser numérico.	Numero del Documento ϵ (0..9999999999), Monto ϵ (0..9999999999)	Debe mostrar un mensaje indicando el número de correspondencias encontradas y el detalle de todas las Correspondencias de Salida existentes con los siguientes campos: Identificador, Número del documento, Asunto y Materia, en caso de éxito.	La Funcionalidad no muestra correspondencias de salida con un código de expediente, al que se le han agregado correspondencias.
Observaciones del Caso de Prueba				
Decisión de Aprobación de Caso de Prueba:	**Aprobó:**		**Falló:** √	
Fecha de Aprobación del Caso de Prueba:			**Aprobado por:** Ing. Carlos Moreno	
Ajustado por:	Neldy Corredor		**Fecha de Ajuste:**	24/11/2008
Prueba de Ajuste, en fecha:	24/11/2008	**Decisión de Aprobación de Ajuste:**	**Aprobó:** √	**Falló:**

Tabla 3.5. Formato de Caso de Prueba perteneciente a la funcionalidad Consultar Correspondencia de Salida para el Rol "Unidad de Documentación" (Fuente: Formato establecido por el representante de BANAVIH).

- **Agregar correspondencias a un expediente**

<table>
<tr><td colspan="8">Nombre del Sistema
Sistema de Información para el manejo de Correspondencias y Puntos de Cuenta</td></tr>
<tr><td>Módulo:</td><td colspan="7">Correspondencia</td></tr>
<tr><td>Historia de Usuarios</td><td colspan="7">Clasificar la correspondencia por expediente (asociar correspondencias a un expediente)</td></tr>
<tr><td colspan="2">Caso de Prueba:</td><td colspan="6">Verificar que las correspondencias sean asociadas a un expediente.</td></tr>
<tr><td colspan="3">Versión de caso de prueba:</td><td>Versión 1</td><td>Fecha ejecución:</td><td colspan="3">24/11/2008</td></tr>
<tr><td colspan="4">Nombre del Probador:</td><td colspan="4">Oscar Ruiz</td></tr>
<tr><td colspan="8">Precondiciones: El identificador de la correspondencia no puede ser vacío y debe ser numérico</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td colspan="8">Para la ejecución del Caso de Prueba:</td></tr>
<tr><td>Paso</td><td>Condición</td><td>Valor(es)</td><td colspan="2">Resultado esperado</td><td colspan="3">Resultado Obtenido</td></tr>
<tr><td>Introducir el identificador de la correspondencia.</td><td>Debe consultar el expediente para poder asociarle las correspondencias y el identificador de la correspondencia debe existir en el sistema.</td><td>Identificador ϵ (0..99999)</td><td colspan="2">Debe mostrar un mensaje indicando que la correspondencia ha sido asociada al expediente y mostrar el detalle en la que se muestren: identificador, número del documento, asunto, materia y fecha en la que se registró la correspondencia.</td><td colspan="3">La Funcionalidad no agrega correctamente las correspondencias al expediente seleccionado.</td></tr>
<tr><td colspan="8">Observaciones del Caso de Prueba</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td colspan="4">Decisión de Aprobación de Caso de Prueba:</td><td>Aprobó:</td><td></td><td>Falló: √</td><td></td></tr>
<tr><td colspan="5">Fecha de Aprobación del Caso de Prueba:</td><td></td><td colspan="2">Aprobado por: Ing. Carlos Moreno</td></tr>
<tr><td colspan="2">Ajustado por:</td><td colspan="3">Oscar Ruiz</td><td colspan="2">Fecha de Ajuste:</td><td>24/11/2008</td></tr>
<tr><td colspan="2">Prueba de Ajuste, en fecha:</td><td>24/11/2008</td><td colspan="2">Decisión de Aprobación de Ajuste:</td><td>Aprobó:</td><td>√</td><td>Falló:</td></tr>
</table>

Tabla 3.6. Formato de Caso de Prueba perteneciente a la funcionalidad Agregar correspondencias a un expediente para el Rol "Unidad de Documentación" (Fuente: Formato establecido por el representante de BANAVIH).

Los resultados obtenidos para esta iteración se aprecian en la Figura 3.12.

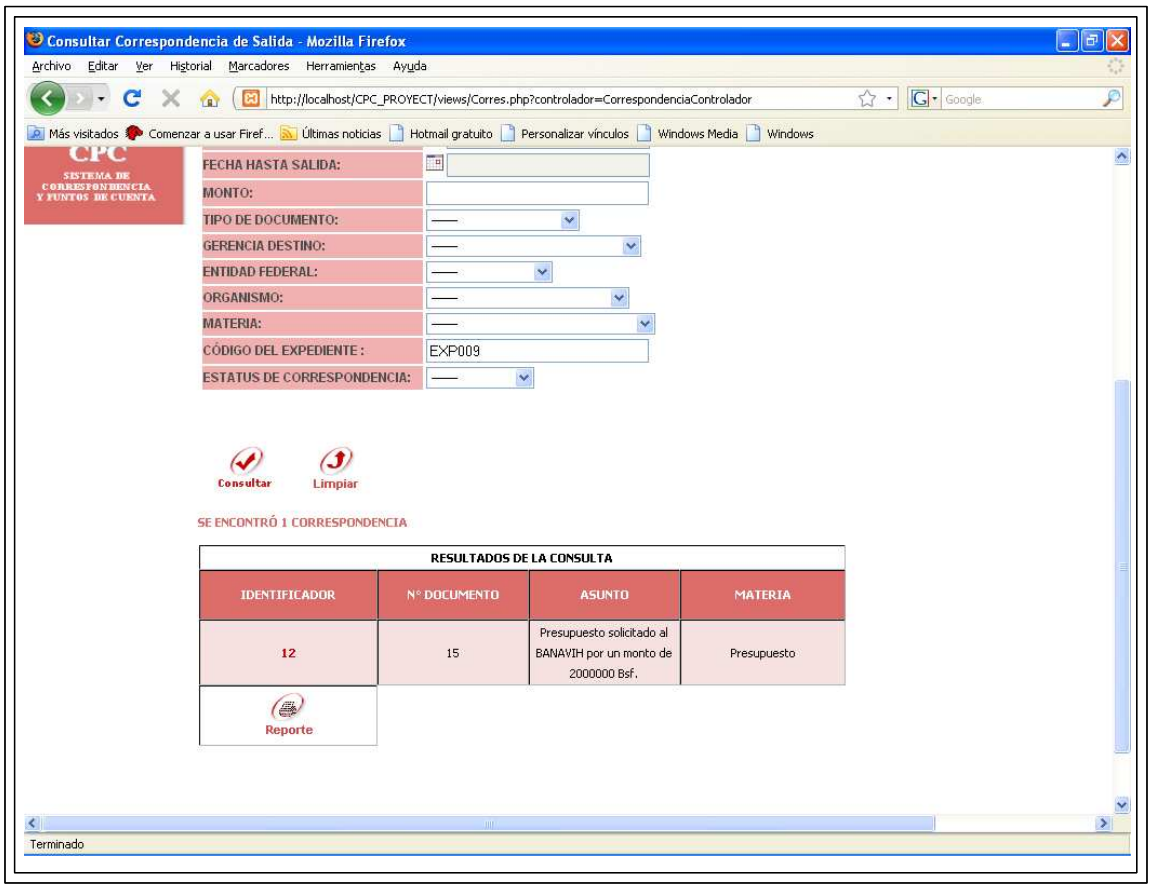

Figura 3.12. Foto de la Consulta de la correspondencia de salida en el sistema (Fuente: Elaboración propia).

La figura 3.12 muestra los resultados de la consulta de la correspondencia de salida, a través del criterio de búsqueda código del expediente.

3.1.1.3 Iteración 3

La iteración 3 se basa en las funcionalidades de consulta de la correspondencia de salida, emisión de la misma por parte de la gerencia y verificación de la respuesta por parte de la Presidencia.

3.1.1.3.1 Planificación de la iteración 3

La planificación de esta iteración, se basa en las historias de usuarios de la tabla 3.7 que se especifica a continuación:

Historias de Usuarios para la 3era. Iteración		
Fecha	**Usuario**	**Descripción**
25/11/2008	Ing. Carlos Moreno	Consultar las Correspondencias de Salida pertenecientes solamente a la gerencia de un usuario.
25/11/2008	Ing. Carlos Moreno	Emitir respuesta de una Correspondencia de Salida, por parte de la gerencia.
26/11/2008	Ing. Carlos Moreno	Consultar Correspondencias de Salida asociadas a todas las gerencias. Este requerimiento es ejecutado por la presidencia.
26/11/2008	Ing. Carlos Moreno	Consultar la respuesta asociada a una correspondencia de salida dada por la gerencia emisora. Este requerimiento es ejecutado por la presidencia.

Tabla 3.7. Formato de Historia de Usuarios para la Tercera Iteración perteneciente a los roles "Gerencia" y "Presidencia" (Fuente: Elaborado por el representante de BANAVIH).

3.1.1.3.2 Diseño de la iteración 3

El diseño de la iteración 3 es planteado a través del Modelo lógico y el Diagrama de Clases, debido a que el Diagrama Conceptual y el Diagrama de Entidad/Relación no presentan ninguna modificación en esta iteración.

- **Modelo Lógico**

El modelo lógico de la iteración 3 se basa en las tablas mostradas en la iteración 2, con campos de salida adicionales, los cuales son respuesta y fecha de respuesta para la entidad Correspondencia, con la finalidad de que el gerente pueda emitir la respuesta pertinente y quede registrada la fecha en la que se dio esta respuesta.

La tabla estatus_correspondencia contiene tres estatus más para la correspondencia, los cuales son: EsperarRes, se activa cuando la gerencia consulta la correspondencia de salida. PreCerrado, se activa cuando la gerencia emite la respuesta de la correspondencia. Cerrado, el cual se activa cuando la Presidencia consulta la respuesta de la correspondencia de salida que ha emitido la Gerencia (Ver Figura 3.13).

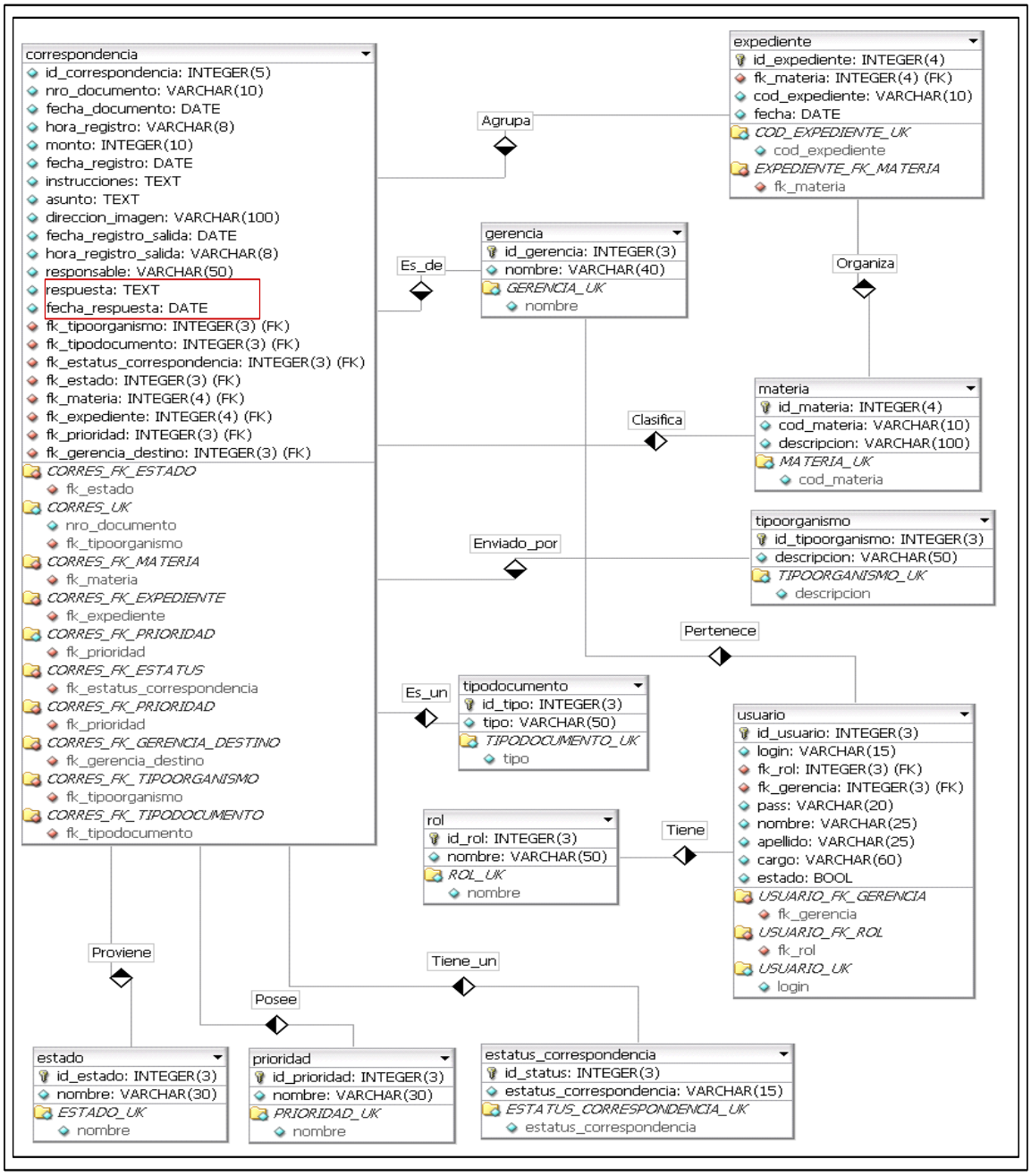

Figura 3.13. Modelo Lógico perteneciente a la Tercera Iteración para los Roles "Gerencia" y "Presidencia" (Fuente: Elaboración propia).

- **Diagrama de Clases**

En el diagrama de clases de la iteración 3 se toman en cuenta las clases pertenecientes a la segunda iteración y se agregan dos campos a la clase correspondencia: respuesta y fecha_respuesta. Además, se incorporan las funciones correspondientes a los roles: Gerencia y Presidencia en la clase CorrespondenciaModel, para poder cumplir a cabalidad las historias de usuarios de esta iteración para el módulo de correspondencia. Las funciones son las siguientes:

- ConsultarCorresSalidaGer: consulta las correspondencias de salida asociadas a la gerencia.
- Consultar_Detalle_CorresSalGer: consulta el detalle de los campos de la correspondencia de salida.
- Guardar_CorresSalidaGer: inserta la respuesta en la correspondencia de salida.
- ConsultarCorresSalidaPresi: consulta las correspondencias de salida asociadas a todas las gerencia.
- Consultar_Detalle_CorresSalPresi: consulta el detalle de los campos de la correspondencia de salida.
- ConsultarEntradaPresi: consulta el detalle de los campos de la correspondencia de entrada.
- Consultar_Detalle_CorresEntradaPresi: consulta el detalle de los campos de la correspondencia de entrada.

La figura 3.14 muestra lo descrito anteriormente.

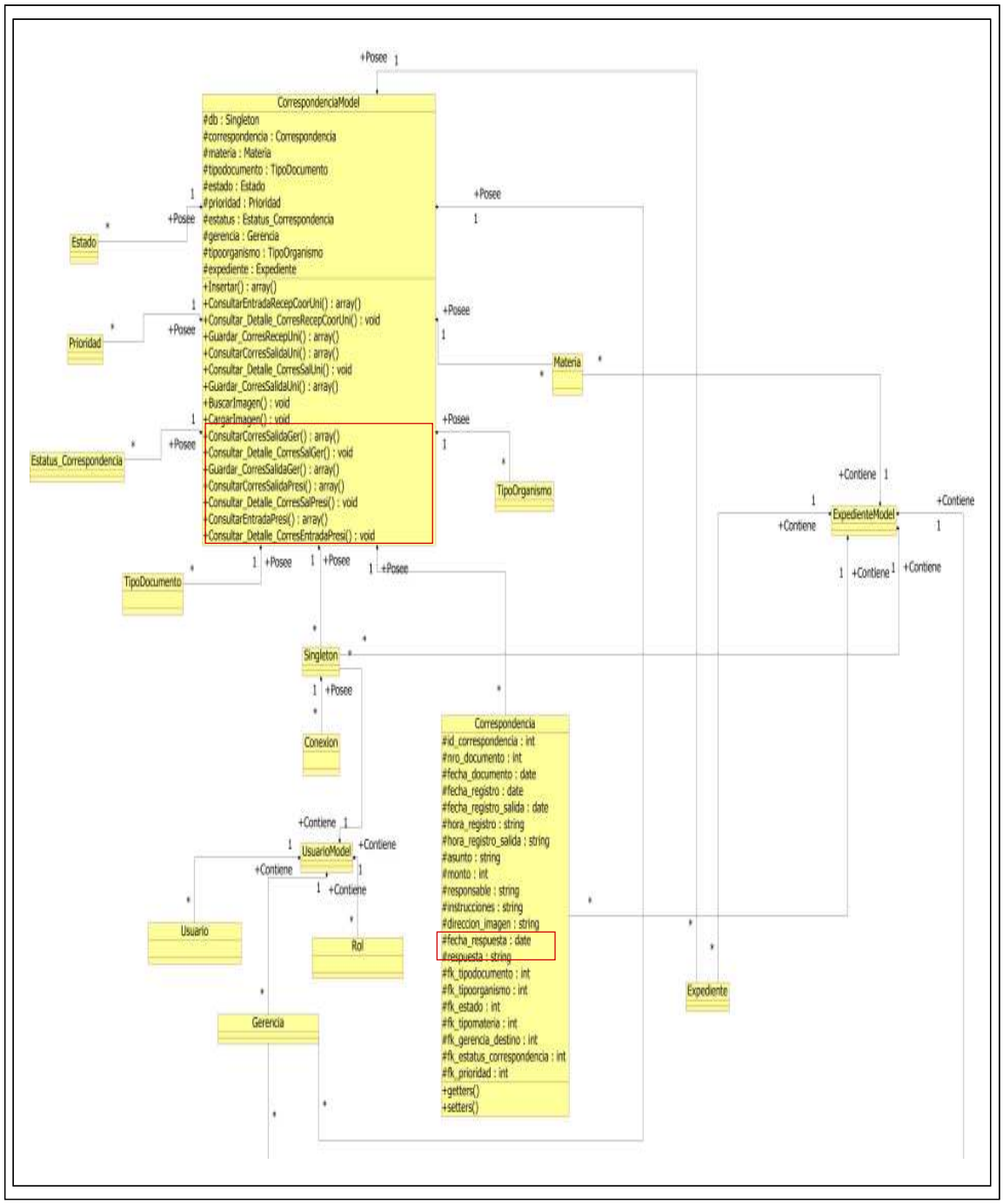

Figura 3.14. Diagrama de Clases, perteneciente a la Tercera iteración del Módulo de Correspondencia para los Roles "Gerencia" y "Presidencia" (Fuente: Elaboración propia).

3.1.1.3.3 Codificación de la iteración 3

Para la iteración 3 se toman en cuenta otras funciones del Modelo de Correspondencia que manejan las historias de usuario para el rol "Gerencia" y "Presidencia". Esta iteración se basa en lo siguiente (Ver *anexos A13, A14, A15, A16 y A17*):

- Refactorización de la función que muestra el detalle de la correspondencia de salida para la Unidad de Documentación, debido a que incorpora dos nuevos campos: respuesta y fecha de respuesta.
- Funciones que permiten consultar la correspondencia de salida y actualizar los campos respuesta y fecha de respuesta, cuando el gerente emite la misma.
- Función que permite consultar todas las correspondencias de salida (con estatus Salida, EsperarRes, PreCerrado o Cerrado) asociadas a todas las gerencias.
- Función que cambia internamente el estatus de una correspondencia de salida, de PreCerrado a Cerrado cuando Presidencia consulta la respuesta que proviene de la gerencia asociada por primera vez.

3.1.1.3.4 Pruebas de Aceptación y Resultados del Sistema para la iteración 3

Todas las funcionalidades fueron comprobadas por los desarrolladores y supervisadas por el Ing. Carlos Moreno, quien estuvo de acuerdo en aprobarlas, debido a que no surgió ningún error en las mismas.

La figura 3.15 muestra los resultados de la emisión de la respuesta a la correspondencia de salida, por parte de la gerencia.

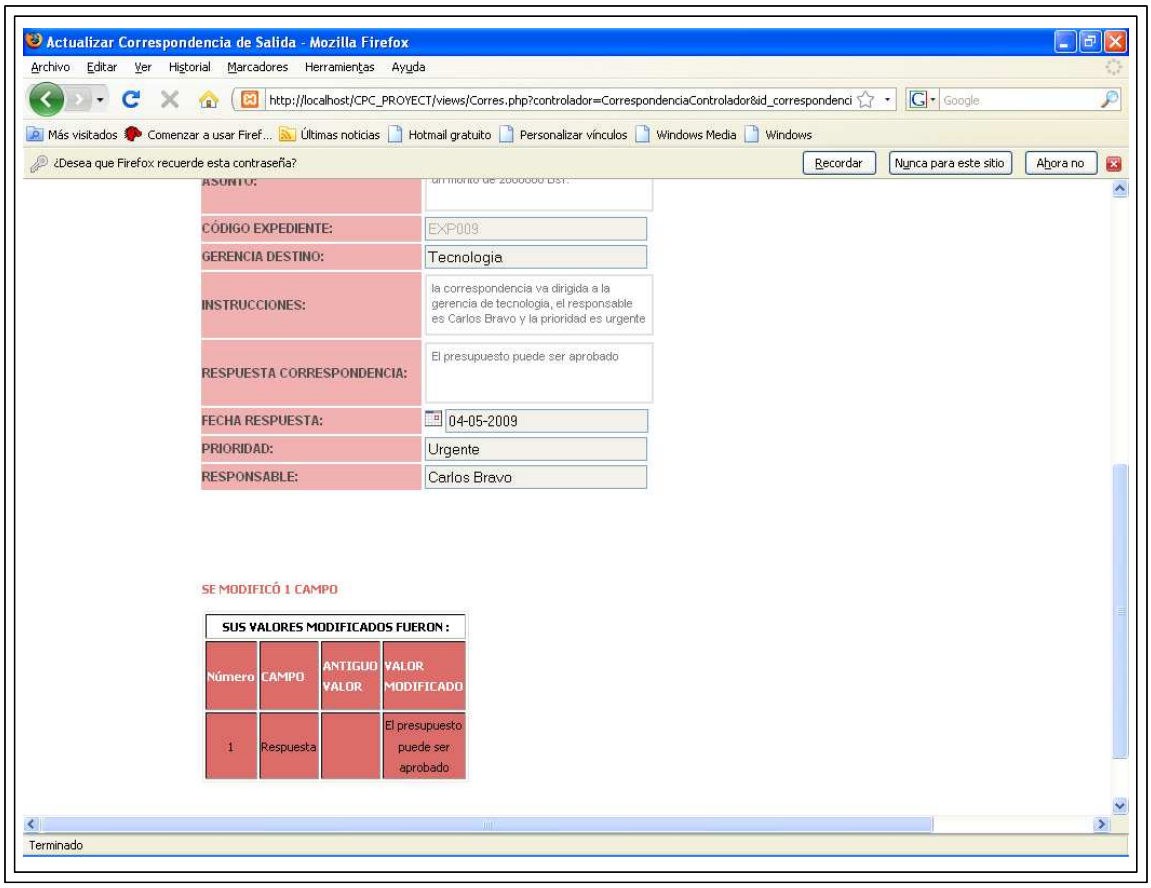

Figura 3.15. Foto de la emisión de la respuesta a la correspondencia, por parte de la gerencia (Fuente: Elaboración propia).

3.1.2 Iteraciones Correspondientes al Módulo de Puntos de Cuenta

En este apartado se detalla el desarrollo de todas las funcionalidades necesarias para construir el módulo de puntos de cuenta.

3.1.2.1 Iteración 4

La iteración 4 se basa en las funcionalidades de los puntos de cuenta, descritos en las historias de usuarios.

3.1.2.1.1 Planificación de la iteración 4

La planificación de esta iteración, consiste en las historias de usuarios de la Tabla 3.8 que se especifica a continuación:

Historias de Usuarios para la 4ta. Iteración		
Fecha	**Usuario**	**Descripción**
27/11/2008 – 28/11/2008	Ing. Carlos Moreno	Registrar Puntos de Cuenta con los siguientes campos: Sesión, Código de Resolución, Gerencia Presentante, Asunto, Fecha, Resultado, Responsable, Resolución e Instrucciones. Esta funcionalidad pertenece solamente a la Secretaría de Junta Directiva.
01/12/2008-02/12/2008	Ing. Carlos Moreno	Consultar Puntos de Cuenta asociados a todas las gerencias. Esta búsqueda solo puede ser realizada por la Secretaría de Junta Directiva y la Presidencia.
03/12/2008 - 05/12/2008	Ing. Carlos Moreno	Agregar observaciones a un Punto de Cuenta. Esto lo puede hacer tanto la Secretaría de Junta Directiva como una Gerencia, para poder establecer una comunicación efectiva acerca del

		estado actual de un Punto de Cuenta.
08/12/2008 – 09/12/2008	Ing. Carlos Moreno	Modificar un Punto de Cuenta. Esto solo debe ser realizado por la Secretaría de Junta Directiva.
10/12/2008	Ing. Carlos Moreno	La Secretaría de Junta Directiva debe poder cerrar un Punto de Cuenta cuando este haya finalizado.

Tabla 3.8. Formato de Historia de Usuarios para la Cuarta Iteración perteneciente a los Roles "Secretaria de Junta Directiva" y "Presidencia" (Fuente: Elaborado por el representante de BANAVIH).

3.1.2.1.2 Diseño de la iteración 4

El diseño de esta iteración consiste en el Diagrama de Entidad/Relación, Modelo Lógico y Diagrama de Clases, debido a que en todos se han incorporado elementos nuevos.

- **Diagrama de Entidad/Relación**

El diagrama de Entidad/Relación de esta iteración consiste en algunos conceptos vistos en el diagrama de Entidad/Relación de la iteración 2 y se agregan los siguientes conceptos: Punto de Cuenta (asuntos tratados y aprobados por la Junta Directiva que necesitan de un seguimiento) y Observación (contiene los comentarios sobre el Punto de Cuenta de parte de la Gerencia relacionada al mismo, y también, de la Secretaría de Junta Directiva). La Figura 3.16 muestra el diagrama Entidad/Relación de la iteración 4.

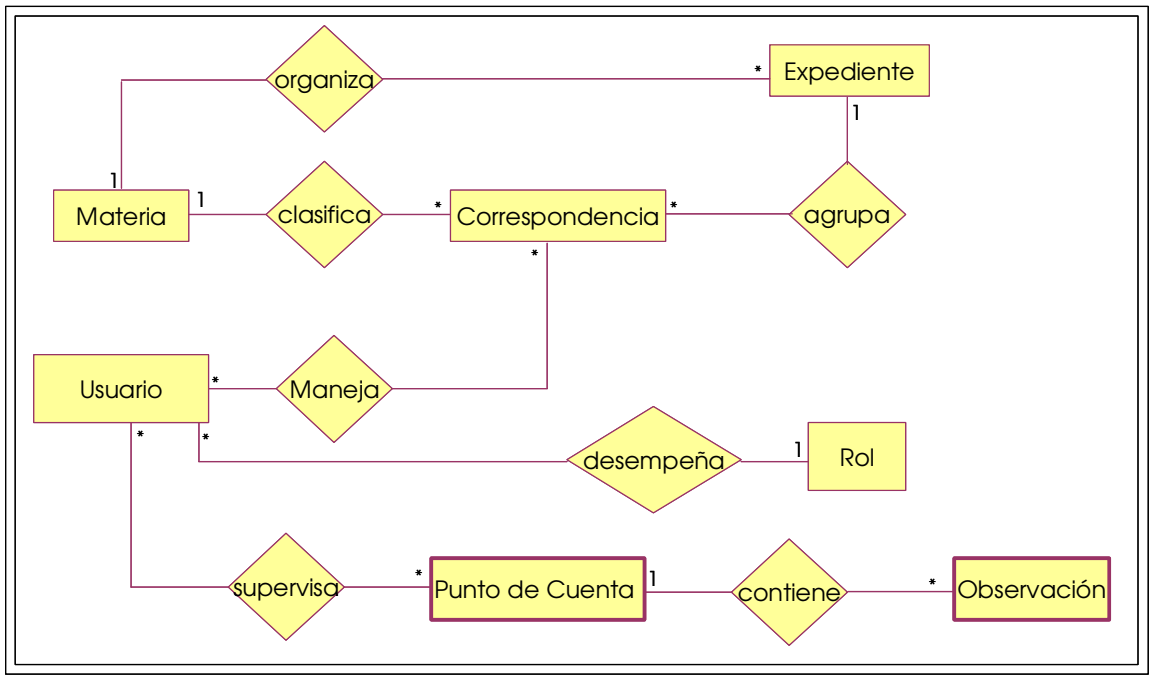

Figura 3.16. Diagrama de Entidad/Relación, perteneciente a la Cuarta Iteración para los Roles "Secretaria de Junta Directiva" y "Presidencia" (Fuente: Elaboración propia).

La descripción de la Figura 3.16 es la siguiente:

El diagrama de Entidad/Relación indica que los puntos de cuenta son supervisados por múltiples usuarios, y un usuario supervisa uno o más puntos de cuenta; cada punto de cuenta contiene múltiples observaciones y una observación solamente puede estar relacionada con un punto de cuenta.

- **Modelo Lógico**

En la iteración 4, la estructura física de la base de datos se compone de las entidades (tablas), atributos (campos), claves primarias, claves ajenas e índices de la iteración 3 y, adicionalmente se incorporan las tablas: ptocuenta, resultado y estatus_pto_cuenta (Ver Figura 3.17).

Las tablas que se describirán de la figura 3.17 son las siguientes:

- ptocuenta: contiene los puntos de cuenta aprobados en la Junta Directiva.

Los campos de la tabla ptocuenta son los siguientes:

  - id_correspondencia: clave primaria que indica que el registro del punto de cuenta es único en el sistema.
  - sesion: número de reunión de la Junta Directiva.
  - cod_resolucion: código de la resolución.
  - resolucion: nombre de la resolución.
  - asunto: resumen de lo que trata el punto de cuenta.
  - fecha: fecha de registro del punto de cuenta en el sistema.
  - responsable: nombre del encargado del punto de cuenta.
  - instrucciones dadas por la Junta Directiva a las gerencias relacionadas a los puntos de cuenta.
  - id_por_gerencia: número que permite establecer una secuencia por gerencia, con la finalidad de mostrar los puntos de cuenta discriminados por gerencia.
  - claves ajenas: fk_resultado (identificador que indica el resultado asociado al punto de cuenta), fk_gerencia (identificador que indica la gerencia que ha presentado el punto de cuenta) y fk_estatus_pto_cuenta (identificador que indica el estatus asociado al punto de cuenta).
  - Índices: PTOCUENTA_UK(indica que el código de la resolución debe ser único en el sistema), PTOCUENTA_FK_ESTATUS, PTOCUENTA_FK_GERENCIA, PTOCUENTA_FK_RESULTADO.

- resultado: contiene la decisión tomada por la Junta Directiva acerca del punto de cuenta, que puede ser aprobado, visto o diferido.

Los campos de la tabla resultado son los siguientes:

- id_resultado: clave primaria que indica que el registro del resultado es único en el sistema.
- resultado: nombre de la decisión tomada por la Junta Directiva acerca del punto de cuenta.
- Índice: RESULTADO_UK (indica que el resultado debe ser único en el sistema).

o estatus_pto_cuenta: contiene el estatus del punto de cuenta, que puede ser: En_Proceso, cuando el punto de cuenta está en ejecución; o Cerrado, cuando el punto de cuenta es finalizado

Los campos de la tabla estatus_pto_cuenta son los siguientes:

- id_status: clave primaria que indica que el registro del estatus del punto de cuenta debe ser único en el sistema.
- estatus_pto_cuenta: nombre del estatus del punto de cuenta.
- Índice: ESTATUS_PTO_CUENTA_UK (indica que el estatus_punto_cuenta debe ser único en el sistema).

o observación: contiene las observaciones hechas por las gerencias y la secretaria de Junta Directiva a los puntos de cuenta.

- id_observacion: clave primaria que indica que el registro de la observación debe ser único en el sistema.
- titulo: nombre del titulo de la observación.
- fecha: fecha en la que se registra la observación.
- cuerpo: resumen de lo que trata la observación.
- hora: hora en la que se registra la observación.
- Claves ajenas: fk_ptocuenta y fk_usuario.

- Índice: OBSERVACION_FK_USUARIO, OBSERVACION_FK_PTOCUENTA y OBSERVACION_UK (indica que el titulo y la fecha de cada observación deben ser únicos en el sistema).

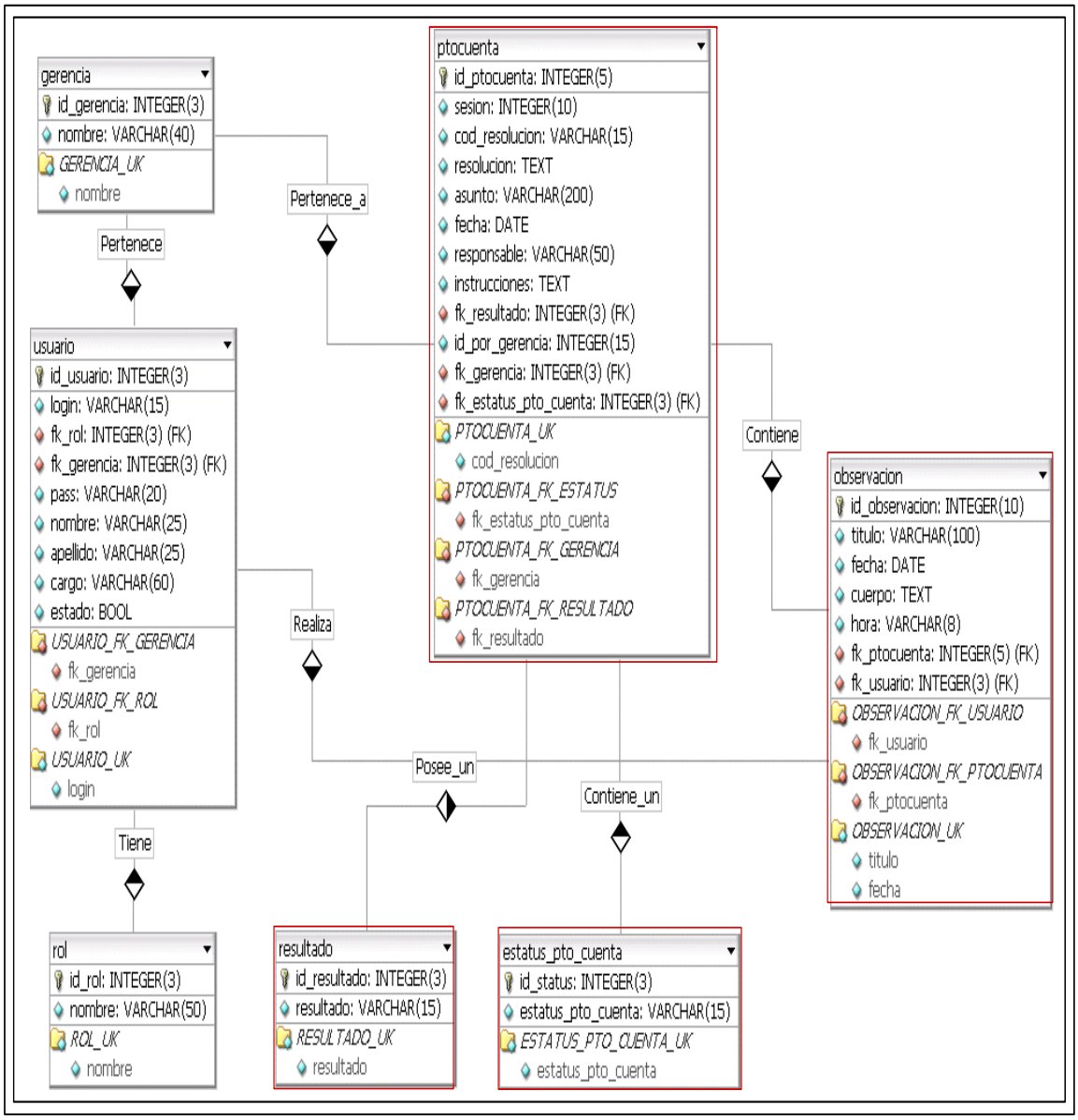

Figura 3.17. Modelo Lógico perteneciente a la Cuarta Iteración para los Roles "Secretaria de Junta Directiva" y "Presidencia" (Fuente: Elaboración propia).

- **Diagrama de Clases**

En el Diagrama de Clases se mostrarán sólo las clases pertenecientes a la iteración 4 del módulo de Punto de Cuenta, entre éstas: Punto de Cuenta,

Resultado, Estatus_Pto_Cuenta, Observación y la clase PtoCuentaModel, perteneciente a la lógica del negocio del Patrón MVC (Ver Figura 3.18).

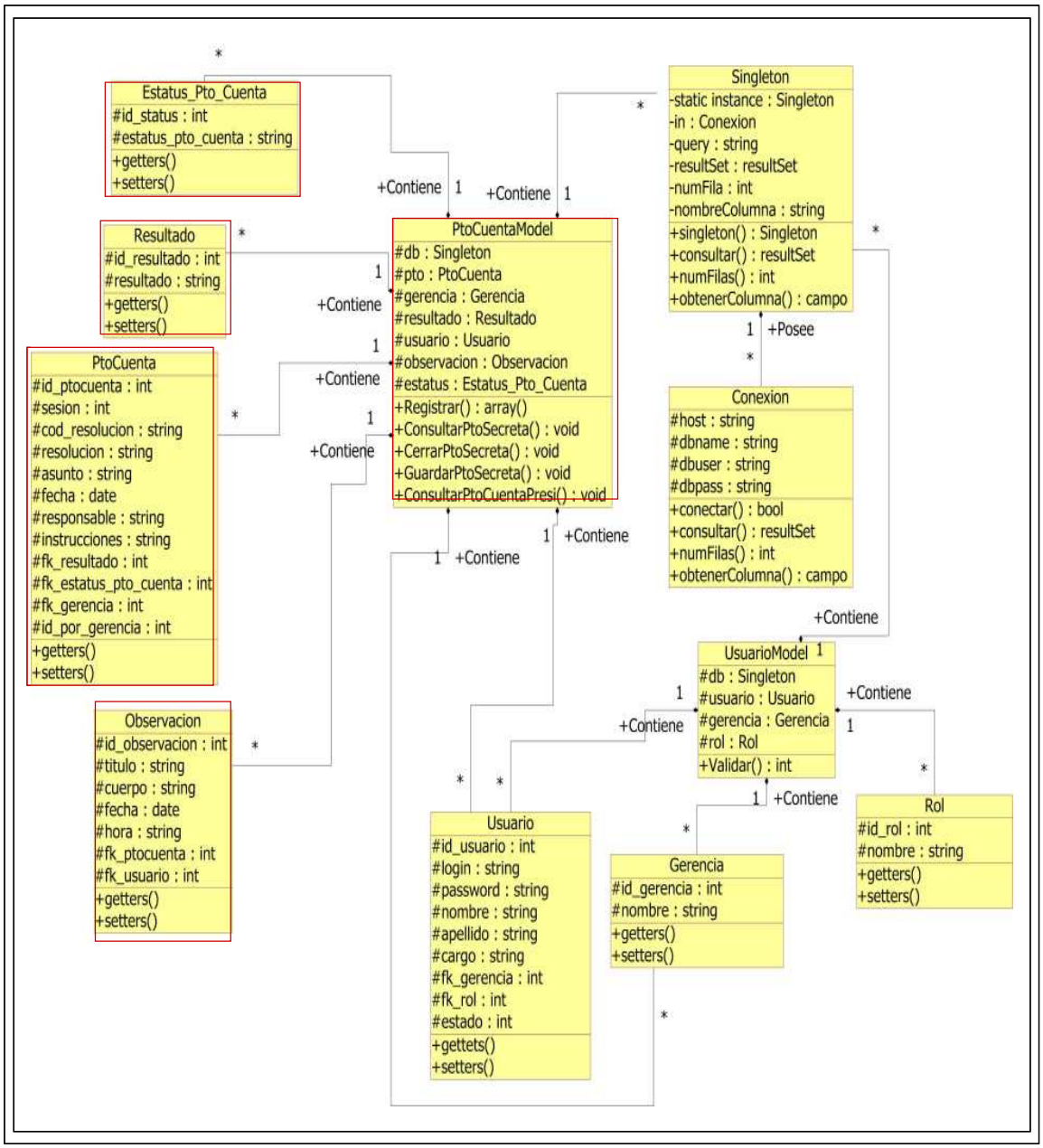

Figura 3.18. Diagrama de Clases, perteneciente a la Cuarta Iteración del Módulo Punto de Cuenta para los Roles "Secretaria de Junta Directiva" y "Presidencia" (Fuente: Elaboración propia).

A continuación se describe la clase PtoCuentaModel, debido a que los atributos de las clases ya fueron descritos por los atributos de las tablas vistas en el modelo lógico de esta iteración:

- PtoCuentaModel: Contiene la lógica de negocio necesaria para manejar los puntos de cuenta.

Los atributos de la clase PtoCuentaModel son:

- db: instancia de la clase Singleton.
- pto: instancia de la clase PtoCuenta.
- gerencia: instancia de la clase Gerencia.
- resultado: instancia de la clase Resultado.
- usuario: instancia de la clase Usuario.
- observación: instancia de la clase Observación.
- estatus: instancia de la clase Estatus_pto_cuenta.

Los métodos de la clase PtoCuentaModel son:

- Registrar: registra un nuevo punto de cuenta.
- ConsultarPtoSecreta: consulta todos los puntos de cuenta asociados a todas las gerencias del banco, de acuerdo al criterio de búsqueda del usuario y el detalle de los campos de los puntos de cuenta.
- CerrarPtoSecreta: cierra el punto de cuenta.
- GuardarPtoSecreta: actualiza el punto de cuenta.
- ConsultarPtoCuentaPresi: consulta todos los puntos de cuenta asociados a todas las gerencias del banco, de acuerdo al criterio de búsqueda del usuario y el detalle de los campos de los puntos de cuenta.

3.1.2.1.3 Codificación de la iteración 4

Para la iteración 4 se toma en cuenta las funciones y acciones del Modelo de PtoCuenta que manejan las historias de usuario para la Secretaría de Junta Directiva y la Presidencia.

La codificación de esta iteración se basa en lo siguiente:

- Funciones y acciones del Modelo PtoCuenta que se encargan de insertar, consultar, agregar observaciones, finalizar y actualizar los puntos de cuenta de todas las gerencias, perteneciente a la Secretaria de Junta Directiva. Esto es ejecutado por la Secretaría de Junta Directiva.
- Acción que permite consultar todos los Puntos de Cuenta asociados a las distintas gerencias, consultar el detalle de los campos y las observaciones de los Puntos de Cuenta. Esto es ejecutado por la Presidencia.
- Reportes que permiten generar la información en *PDF* de las consultas, observaciones y el detalle de los puntos de cuenta.

3.1.2.1.4 Pruebas de Aceptación y Resultados del Sistema para la iteración 4

En esta iteración se presentaron dos errores, los cuales fueron:

- Al registrar un punto de cuenta, se asignaba al mismo la gerencia del usuario y no la gerencia a la que pertenecía el punto de cuenta.
- Al agregar una observación de un punto de cuenta, se repetía la inserción de la misma, tantas veces como se actualizaba el navegador.

Los errores de esta iteración fueron corregidos y supervisados por el Ing. Carlos Moreno, el cual quedó conforme con todos los resultados obtenidos en el sistema.

A continuación se muestran los formatos (ver tabla 3.9 y tabla 3.10) de prueba para las funcionalidades que presentaron errores:

- **Registrar Punto de Cuenta**

<table>
<tr><td colspan="8">Nombre del Sistema
Sistema de Información para el manejo de Correspondencias y Puntos de Cuenta</td></tr>
<tr><td>Módulo:</td><td colspan="7">Punto de Cuenta</td></tr>
<tr><td>Historia de Usuarios</td><td colspan="7">Registrar Puntos de Cuenta con los siguientes campos: Sesión, Código de Resolución, Gerencia Presentante, Asunto, Fecha, Resultado, Responsable, Resolución e Instrucciones. Esta funcionalidad pertenece solamente a la Secretaría de Junta Directiva.</td></tr>
<tr><td colspan="2">Caso de Prueba:</td><td colspan="6">Verificar que el Registro del Punto de Cuenta se efectúe satisfactoriamente con los siguientes campos: Sesión, Código de Resolución, Gerencia Presentante, Asunto, Fecha, Resultado, Responsable, Resolución e Instrucciones.</td></tr>
<tr><td colspan="2">Versión de caso de prueba:</td><td colspan="2">Versión 1</td><td>Fecha ejecución:</td><td colspan="3">11/12/2008</td></tr>
<tr><td colspan="3">Nombre del Probador:</td><td colspan="5">Oscar Ruiz</td></tr>
<tr><td colspan="8">Precondiciones: El código de resolución debe ser Alfanumérico y el campo sesión debe ser un número</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td colspan="8">Para la ejecución del Caso de Prueba:</td></tr>
<tr><td>Paso</td><td>Condición</td><td>Valor(es)</td><td colspan="3">Resultado esperado</td><td colspan="2">Resultado Obtenido</td></tr>
<tr><td>Insertar los datos de un Punto de Cuenta.</td><td>El código de resolución no debe estar registrado en el sistema.</td><td></td><td colspan="3">Debe mostrar un mensaje indicando que el Punto de Cuenta se ha insertado correctamente y el detalle de la inserción del mismo con los siguientes campos: Identificador, Asunto, Sesión, Fecha, Resultado y Gerencia.</td><td colspan="2">La Funcionalidad se ejecutó incorrectamente, debido a que registra el punto de cuenta con la gerencia del usuario, en vez de la gerencia a la que pertenece el punto de cuenta.</td></tr>
<tr><td colspan="8">Observaciones del Caso de Prueba</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td colspan="3">Decisión de Aprobación de Caso de Prueba:</td><td>Aprobó:</td><td></td><td>Falló:</td><td colspan="2">√</td></tr>
<tr><td colspan="4">Fecha de Aprobación del Caso de Prueba:</td><td></td><td colspan="3">Aprobado por: Ing. Carlos Moreno</td></tr>
<tr><td colspan="2">Ajustado por:</td><td colspan="3">Oscar Ruiz</td><td colspan="3">Fecha de Ajuste: 11/12/2008</td></tr>
<tr><td colspan="2">Prueba de Ajuste, en fecha:</td><td>11/12/2008</td><td colspan="2">Decisión de Aprobación de Ajuste:</td><td>Aprobó:</td><td>√</td><td>Falló:</td></tr>
</table>

Tabla 3.9. Formato de Caso de Prueba perteneciente a la funcionalidad Registrar Punto de Cuenta del Rol "Secretaria de Junta Directiva" (Fuente: Formato establecido por el representante de BANAVIH).

- **Agregar las observaciones del Punto de Cuenta.**

<table>
<tr><td colspan="6">Nombre del Sistema
Sistema de Información para el manejo de Correspondencias y Puntos de Cuenta</td></tr>
<tr><td>Módulo:</td><td colspan="5">Punto de Cuenta</td></tr>
<tr><td>Historia de Usuarios</td><td colspan="5">Agregar observaciones a un Punto de Cuenta. Esto lo puede hacer tanto la Secretaría de Junta Directiva como una Gerencia, para poder establecer una comunicación efectiva acerca del estado actual de un Punto de Cuenta.</td></tr>
<tr><td colspan="2">Caso de Prueba:</td><td colspan="4">Verificar que las Observaciones hayan sido almacenadas con el título y cuerpo adecuadamente.</td></tr>
<tr><td colspan="2">Versión de caso de prueba:</td><td>Versión 1</td><td>Fecha ejecución:</td><td colspan="2">11/12/2008</td></tr>
<tr><td colspan="2">Nombre del Probador:</td><td colspan="4">Neldy Corredor</td></tr>
<tr><td colspan="6">Precondiciones: Consultar el detalle del Punto de Cuenta seleccionado</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td colspan="6">Para la ejecución del Caso de Prueba:</td></tr>
<tr><td>Paso</td><td>Condición</td><td>Valor(es)</td><td>Resultado esperado</td><td colspan="2">Resultado Obtenido</td></tr>
<tr><td>Agregar las respectivas Observaciones.</td><td>El punto de cuenta no debe tener el estatus cerrado y los campos titulo y observación no pueden estar vacíos, ni repetirse en el sistema.</td><td></td><td>Debe insertar y Mostrar la observación almacenada y las existentes para el Punto de Cuenta con los siguientes campos: Usuario, Gerencia a la que pertenece, Cargo, Fecha y Hora, Título y la Observación correspondiente.</td><td colspan="2">La Funcionalidad se ejecutó incorrectamente, debido a que al actualizar el navegador se registrada de nuevo la observación, repitiéndose la misma en el sistema.</td></tr>
<tr><td colspan="6">Observaciones del Caso de Prueba</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td colspan="3">Decisión de Aprobación de Caso de Prueba:</td><td>Aprobó:</td><td>Falló: √</td><td></td></tr>
<tr><td colspan="3">Fecha de Aprobación del Caso de Prueba:</td><td></td><td colspan="2">Aprobado por: Ing. Carlos Moreno</td></tr>
<tr><td colspan="2">Ajustado por:</td><td colspan="2">Neldy Corredor</td><td>Fecha de Ajuste:</td><td>11/12/2008</td></tr>
<tr><td>Prueba de Ajuste, en fecha:</td><td>11/12/2008</td><td>Decisión de Aprobación de Ajuste:</td><td>Aprobó:</td><td>√</td><td>Falló:</td></tr>
</table>

Tabla 3.10. Formato de Caso de Prueba perteneciente a la funcionalidad Agregar las Observaciones del Punto de Cuenta del Rol "Secretaria de Junta Directiva" (Fuente: Formato establecido por el representante de BANAVIH).

Los resultados obtenidos para esta iteración se muestran en la figura 3.19, en la que se aprecia el registro de un punto de cuenta en el sistema.

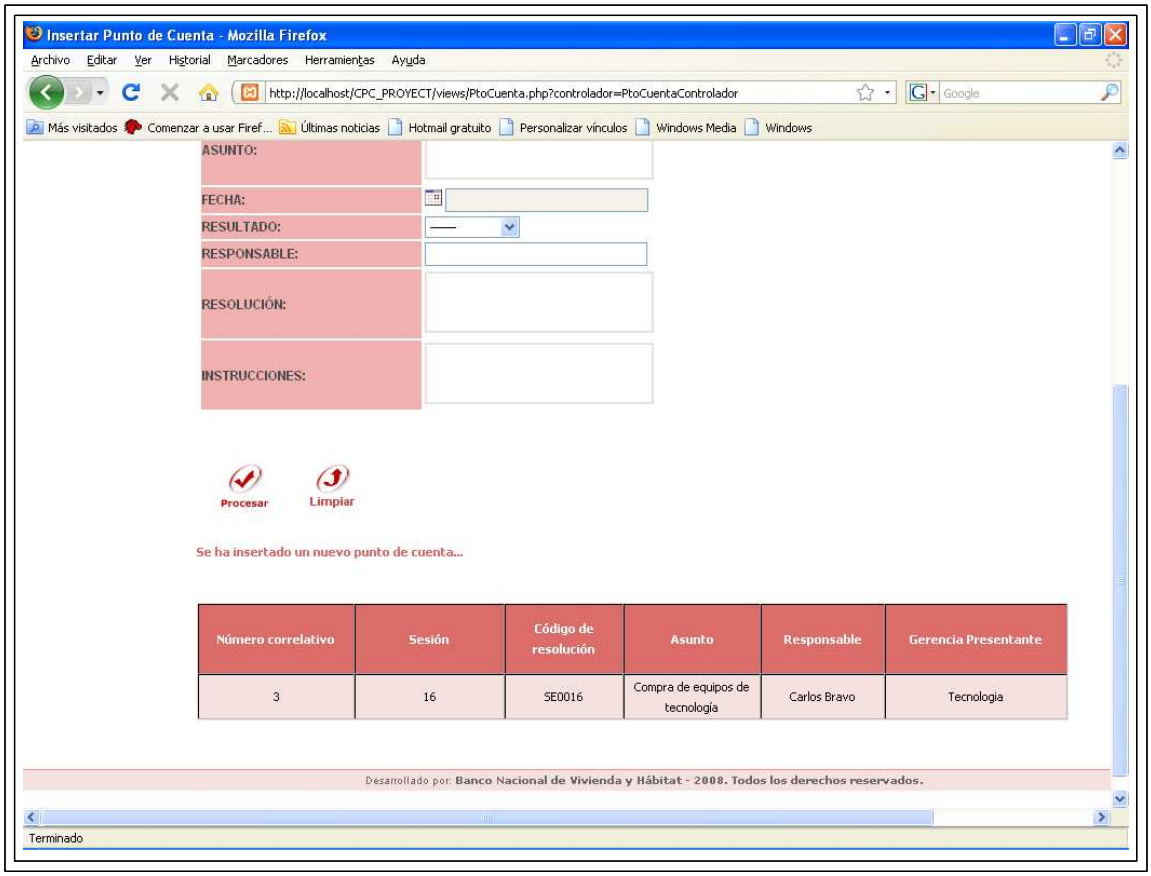

Figura 3.19. Foto del registro del punto de cuenta en el sistema (Fuente: Elaboración propia).

3.1.2.2 Iteración 5

En esta iteración se estudian todas las funcionalidades del rol de "Gerencia" (para el módulo de Puntos de Cuenta). También se desarrollan otras funcionalidades relacionadas con el manejo de recordatorios en el sistema, que pertenecen a todos los roles existentes en la aplicación.

3.1.2.2.1 Planificación de la iteración 5

La planificación de esta iteración, consiste en las historias de usuarios de la Tabla 3.11 que se especifica a continuación:

Historias de Usuarios para la 5ta. Iteración		
Fecha	**Usuario**	**Descripción**
12/12/2008	Ing. Carlos Moreno	Una gerencia debe poder consultar los Puntos de Cuenta asociados a la misma.
12/12/2008	Ing. Carlos Moreno	El gerente debe tener la posibilidad de postergar y reanudar un punto de cuenta.
15/12/2008 – 07/01/2009	Ing. Carlos Moreno	Los usuarios podrán manipular recordatorios en el sistema, así como también, activar alarmas que avisen la proximidad de los mismos.

Tabla 3.11. Formato de Historia de Usuarios para la Quinta Iteración perteneciente al Rol "Gerencia" y Manejo de Recordatorio para todos los roles (Fuente: Elaborado por el representante de BANAVIH).

3.1.2.2.2 Diseño de la iteración 5

El diseño de esta iteración consiste en el Diagrama Conceptual, Diagrama de Entidad/Relación, Modelo Lógico y Diagrama de clases.

- **Diagrama Entidad/Relación**

La Figura 3.20 muestra las entidades y relaciones de la iteración 4 y además incorpora las siguientes entidades:

- Recordatorio: Un recordatorio es un evento que el usuario puede registrar y asociarle una fecha de ejecución, y se encarga de almacenar eventos ya sean particulares, de correspondencia o de puntos de cuenta. Los eventos particulares son eventos propios del sistema, por ejemplo: almuerzo de negocio, cumpleaños, etc. Los eventos de correspondencia son eventos que tienen que ver con la correspondencia, por ejemplo: si una correspondencia tiene que ver con un curso de capacitación, el evento será evento del curso de capacitación. Los eventos de punto de cuenta son eventos que tienen que ver con los puntos de cuenta del sistema, por ejemplo: una reunión para discutir un punto de cuenta, etc.
- Alarma: Maneja las alarmas con los eventos que se muestran al usuario en la página principal (*Home*) del sistema.

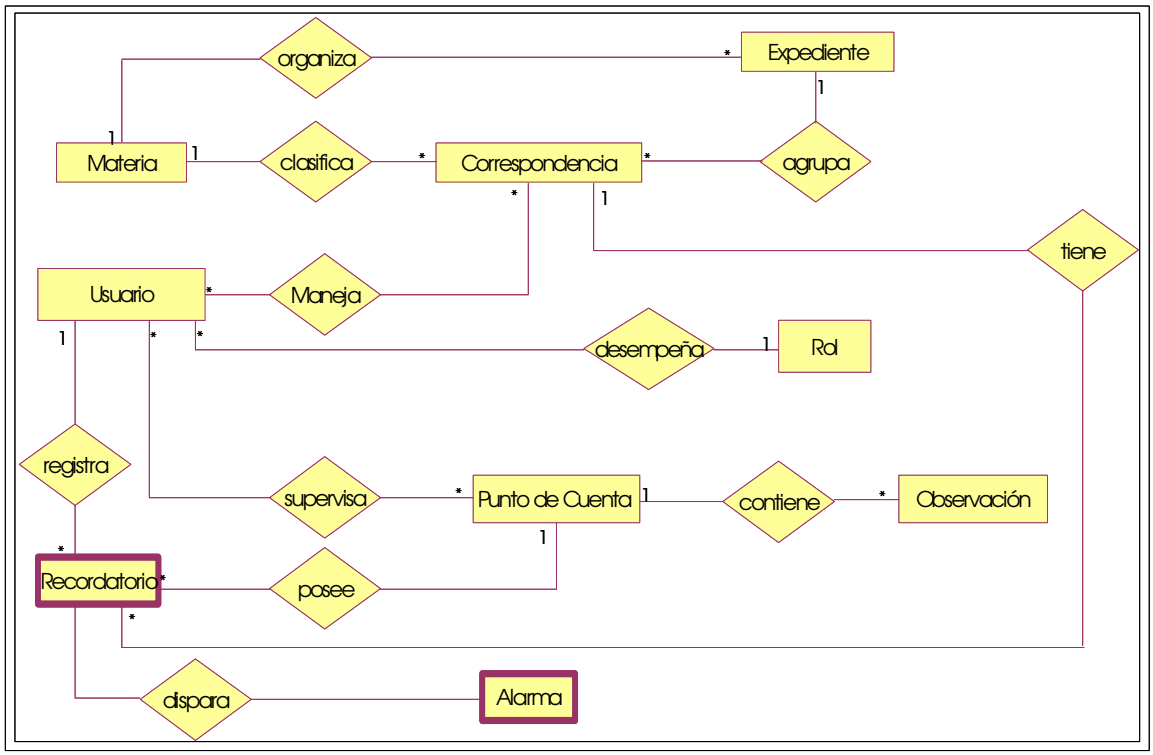

Figura 3.20. Diagrama Entidad/Relación, perteneciente a la Quinta Iteración para el Rol "Gerencia" y el Manejo de Recordatorios (Fuente: Elaboración propia).

En la figura 3.20 se aprecia lo siguiente: un usuario puede registrar múltiples recordatorios, y éste a su vez, puede ser registrado por un sólo usuario; cada recordatorio dispara una alarma y por cada alarma existe un recordatorio.

- **Modelo Lógico**

En la iteración 5, la estructura lógica de la base de datos consiste en las entidades (tablas), atributos (campos), claves primarias, claves ajenas e índices de la iteración 4 y adicionalmente se incorpora la tabla recordatorio para manejar las funcionalidades de recordatorio expuestas en las historias de usuarios de esta iteración (Ver Figura 3.21). Además la tabla estatus_pto_cuenta, contiene el estatus Pendiente, que indica que

un punto de cuenta puede ser postergado por la gerencia de ser necesario.

- Los campos de la tabla recordatorio son:
  - id_recordatorio: clave primaria que indica que el registro del recordatorio es único en el sistema.
  - fecha_inicio: fecha en la que se inicia el evento.
  - fecha_fin: fecha en la que finaliza el evento.
  - evento: descripción del evento.
  - fecha_alarma: fecha en la que se activa la alarma para mostrársela al usuario.
  - Claves ajenas: fk_correspondencia (identificador de la correspondencia que tiene asociado el evento), fk_usuario(identificador del usuario que tiene asociado el evento), fk_ptocuenta(identificador del punto de cuenta que tiene asociado el evento).
  - Índices: RECORDATORIO_FK_USUARIO, RECORDATORIO_FK_PTOCUENTA, RECORDATORIO_FK_CORRES, RECORDATORIO_UK (el evento y la fecha_fin del recordatorio deben ser únicos conjuntamente en el sistema).

La Figura 3.21 muestra los campos descritos anteriormente.

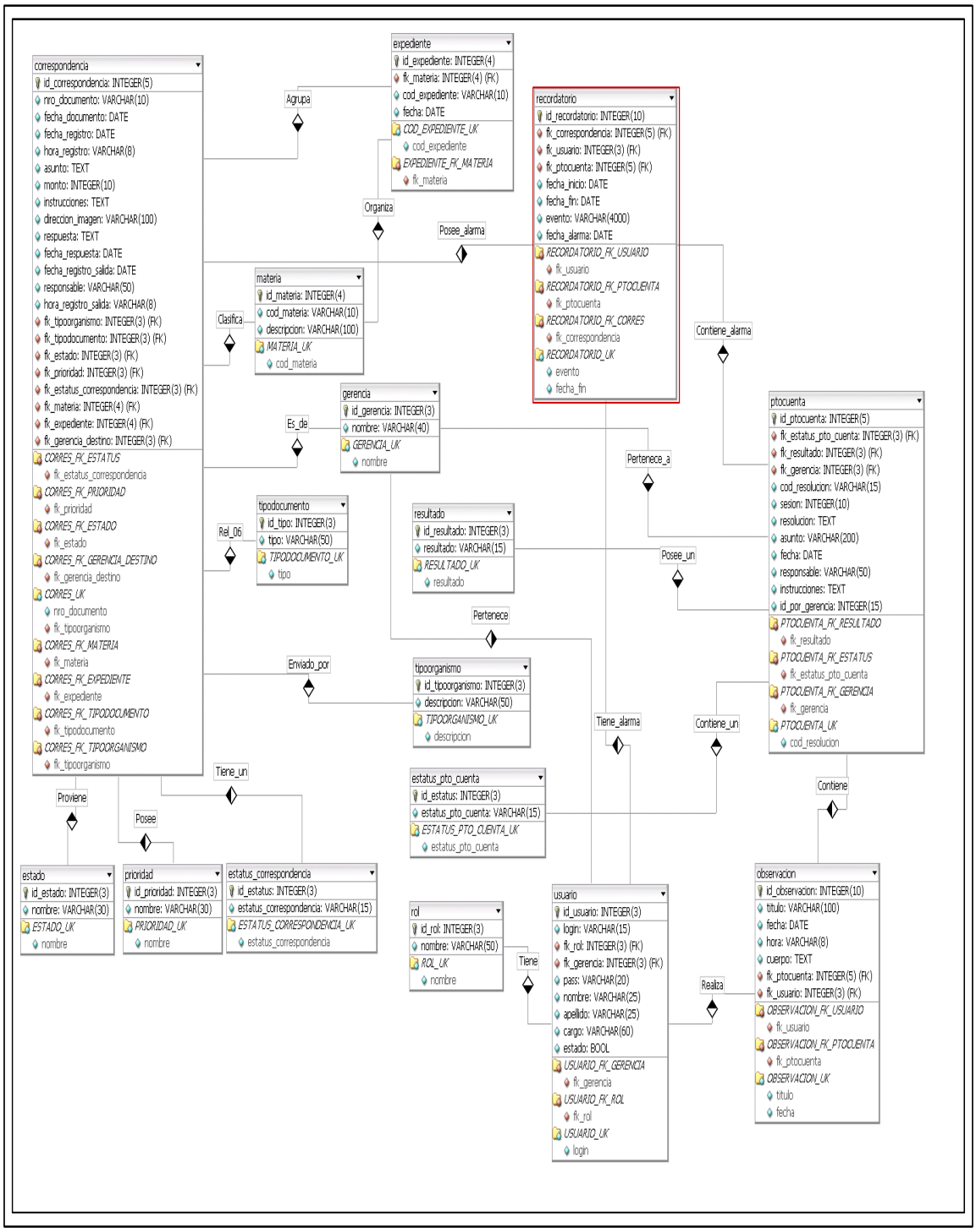

Figura 3.21. Modelo Lógico, perteneciente a la Quinta Iteración para el Rol "Gerencia" y el Manejo de Recordatorios (Fuente: Elaboración propia).

- **Diagrama de Clases**

En el Diagrama de Clases se muestran las clases de la iteración 4 del módulo de Puntos de Cuenta y adicionalmente se incorpora la clase Recordatorio y la clase RecordatorioModel. Además, se anexan los métodos pertenecientes a las funcionalidades del Rol Gerencia en la clase PtoCuentaModel (Ver Figura 3.22).

A continuación se describe la clase RecordatorioModel y los métodos pertenecientes a las funcionalidades del Rol Gerencia en la clase PtoCuentaModel de la figura 3.22:

- RecordatorioModel: Es la clase que maneja la lógica de los eventos asignados a los usuarios. Los atributos de esta clase son:
  - db: instancia de la clase *Singleton*.
  - recordatorio: instancia de la clase Recordatorio.
  - puntocuenta: instancia de la clase PtoCuenta.
  - correspondencia: instancia de la clase Correspondencia.
  - Usuario: instancia de la clase Usuario.

  Los métodos de la clase RecordatorioModel son:

  - Mostrar: despliega la información de los campos del recordatorio.
  - Insertar: registra la información del recordatorio.
  - Eliminar: elimina el recordatorio.
  - Guardar: Modifica la información del recordatorio.

  Los métodos de la clase PtoCuenta agregados en esta iteración son:

- ConsultarPtoCuentaGer: consulta la inf. de los puntos de cuenta asociados a la gerencia.
- GuardarObservacionesGer: inserta la observación del punto de cuenta, por parte de la gerencia.
- PendientePtoCuentaGer: Cambia el estatus del punto de cuenta a Pendiente e inserta una observación interna en el sistema.
- ReanudarPtoCuentaGer: Cambia el estatus del punto de cuenta a En_Proceso e inserta una observación interna en el sistema.

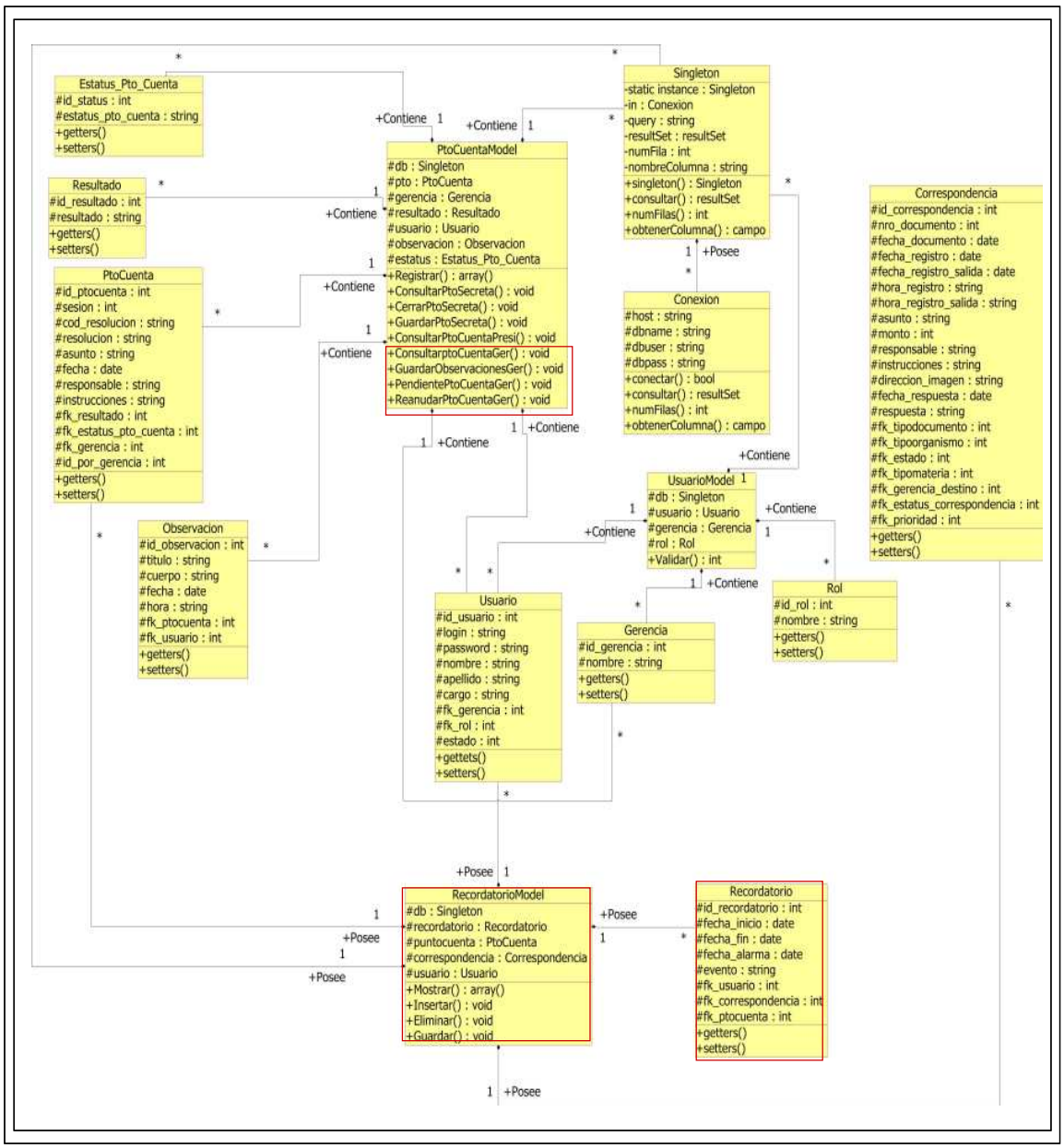

Figura 3.22. Diagrama de Clases, perteneciente a la Quinta Iteración del Modelo de Punto de Cuenta para el Rol "Gerencia" y el Manejo de Recordatorios (Fuente: Elaboración propia).

3.1.2.2.3 Codificación de la iteración 5

Para la iteración 5 se toma en cuenta las acciones del Modelo PtoCuenta que manejan las historias de usuario para el rol "Gerencia" y las acciones del Modelo Recordatorio pertenecientes a la lógica del negocio del patrón MVC.

La quinta iteración se refiere a lo siguiente (Ver *anexos A18 y A19*):

- Acciones para la consulta y cambio del estatus (ya sea para posponerlo o reanudarlo) de los puntos de cuenta de la gerencia a la que pertenece el usuario. Estas acciones están incluidas en el modelo de PtoCuenta.
- Acciones para mostrar, insertar, eliminar y actualizar recordatorios. Estas acciones están incluidas en el modelo de Recordatorio.
- Código de la alarma, la cual se despliega en la página principal (*Home)* de la aplicación, dependiendo si la fecha actual se encuentra entre la fecha de la alarma y la fecha fin de los eventos del usuario.

3.1.2.2.4 Pruebas de Aceptación y Resultados del Sistema para la iteración 5

En esta iteración se encontraron dos errores:

- Al registrar un recordatorio la fecha seleccionada toma valores erróneos, si es una fecha posterior a tres meses de la fecha actual.
- Al modificar los campos de un recordatorio, los eventos pertenecientes a una fecha específica, se actualizaban con la misma información del evento que había sido modificado.

En esta iteración fueron corregidos todos los errores y verificados por el Ing. Carlos Moreno, quien quedó conforme con todos los resultados obtenidos en el sistema.

Los formatos de la tabla 3.12 y tabla 3.13 muestran las pruebas que se realizaron para registrar el recordatorio y modificar el mismo.

- **Registrar Recordatorio**

<table>
<tr><td colspan="6">Nombre del Sistema
Sistema de Información para el manejo de Correspondencias y Puntos de Cuenta</td></tr>
<tr><td>Módulo:</td><td colspan="5">Correspondencia Y Puntos de Cuenta</td></tr>
<tr><td>Historia de Usuarios</td><td colspan="5">Los usuarios podrán manipular recordatorios en el sistema, así como también, activar alarmas que avisen la proximidad de los mismos.</td></tr>
<tr><td>Caso de Prueba:</td><td colspan="5">Verificar que el Registro de un Recordatorio se ejecute con los siguientes campos: Fecha, Evento, Cantidad de días para activar la alarma (No Obligatorio), Identificador y Tipo de Evento.</td></tr>
<tr><td>Versión de caso de prueba:</td><td>Versión 1</td><td colspan="2">Fecha ejecución:</td><td colspan="2">08/01/2009</td></tr>
<tr><td colspan="2">Nombre del Probador:</td><td colspan="4">Oscar Ruiz</td></tr>
<tr><td colspan="6">Precondiciones: Seleccionar fecha en el calendario</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td colspan="6">Para la ejecución del Caso de Prueba:</td></tr>
<tr><td>Paso</td><td>Condición</td><td>Valor(es)</td><td colspan="2">Resultado esperado</td><td>Resultado Obtenido</td></tr>
<tr><td>Insertar los datos del recordatorio</td><td>El evento no puede ser vacío. Si se desea insertar un evento no particular se requiere el identificador de la Correspondencia o el Punto de Cuenta, el cual debe estar registrado en el sistema.</td><td></td><td colspan="2">Debe mostrar un mensaje indicando que el recordatorio se ha insertado correctamente y mostrar los datos del recordatorio registrado.</td><td>La Funcionalidad no se ha ejecutado correctamente, debido a que al tratar de registrar recordatorios la fecha toma valores erróneos.</td></tr>
<tr><td colspan="6">Observaciones del Caso de Prueba</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td colspan="3">Decisión de Aprobación de Caso de Prueba:</td><td>Aprobó:</td><td>Falló: √</td><td></td></tr>
<tr><td colspan="3">Fecha de Aprobación del Caso de Prueba:</td><td></td><td colspan="2">Aprobado por: Ing. Carlos Moreno</td></tr>
<tr><td>Ajustado por:</td><td colspan="2">Oscar Ruiz</td><td colspan="2">Fecha de Ajuste:</td><td>08/01/2009</td></tr>
<tr><td>Prueba de Ajuste, en fecha:</td><td>08/01/2009</td><td>Decisión de Aprobación de Ajuste:</td><td>Aprobó:</td><td>√</td><td>Falló:</td></tr>
</table>

Tabla 3.12.Formato de Caso de Prueba perteneciente a la funcionalidad Registrar Recordatorio para todos los roles (Fuente: Formato establecido por el representante de BANAVIH)

- **Modificar Recordatorio**

<table>
<tr><td colspan="6">Nombre del Sistema
Sistema de Información para el manejo de Correspondencias y Puntos de Cuenta</td></tr>
<tr><td>Módulo:</td><td colspan="5">Correspondencia y Puntos de Cuenta</td></tr>
<tr><td>Historia de Usuarios</td><td colspan="5">Los usuarios podrán manipular recordatorios en el sistema, así como también, activar alarmas que avisen la proximidad de los mismos.</td></tr>
<tr><td colspan="2">Caso de Prueba:</td><td colspan="4">Verificar que en el recordatorio se pueda modificar el evento, cantidad de días para activar la alarma, el identificador y el tipo de evento.</td></tr>
<tr><td colspan="2">Versión de caso de prueba:</td><td>Versión 1</td><td>Fecha ejecución:</td><td colspan="2">08/01/2009</td></tr>
<tr><td colspan="2">Nombre del Probador:</td><td colspan="4">Neldy Corredor</td></tr>
<tr><td colspan="6">Precondiciones: El recordatorio debe existir</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td colspan="6">Para la ejecución del Caso de Prueba:</td></tr>
<tr><td>Paso</td><td>Condición</td><td>Valor(es)</td><td colspan="2">Resultado esperado</td><td>Resultado Obtenido</td></tr>
<tr><td>Modificar en el recordatorio los siguientes campos: Eventos del Día, Cantidad de días para activar la alarma, Identificador y Tipo de Evento.</td><td>Si se desea modificar un evento no particular se requiere el identificador de la Corresponde ncia o el Punto de Cuenta, el cual debe estar registrado en el sistema.</td><td></td><td colspan="2">Debe mostrar el detalle del recordatorio con las modificaciones realizadas.</td><td>Todos los eventos pertenecientes a una fecha específica, han sido actualizados incorrectamente con la misma información del evento que ha sido modificado.</td></tr>
<tr><td colspan="6">Observaciones del Caso de Prueba</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td colspan="3">Decisión de Aprobación de Caso de Prueba:</td><td>Aprobó:</td><td>Falló: √</td><td></td></tr>
<tr><td colspan="3">Fecha de Aprobación del Caso de Prueba:</td><td></td><td colspan="2">Aprobado por: Ing. Carlos Moreno</td></tr>
<tr><td>Ajustado por:</td><td colspan="2">Neldy Corredor</td><td colspan="2">Fecha de Ajuste:</td><td>08/01/2009</td></tr>
<tr><td>Prueba de Ajuste, en fecha:</td><td>08/01/2009</td><td>Decisión de Aprobación de Ajuste:</td><td>Aprobó:</td><td>√</td><td>Falló:</td></tr>
</table>

Tabla 3.13. Formato de Caso de Prueba perteneciente a la funcionalidad Modificar Recordatorio para todos los roles (Fuente: Formato establecido por el representante de BANAVIH)

Los resultados obtenidos para esta iteración se muestran en la figura 3.23, en la que se aprecia el registro de recordatorios de tipo particular, de punto de cuenta y de correspondencia.

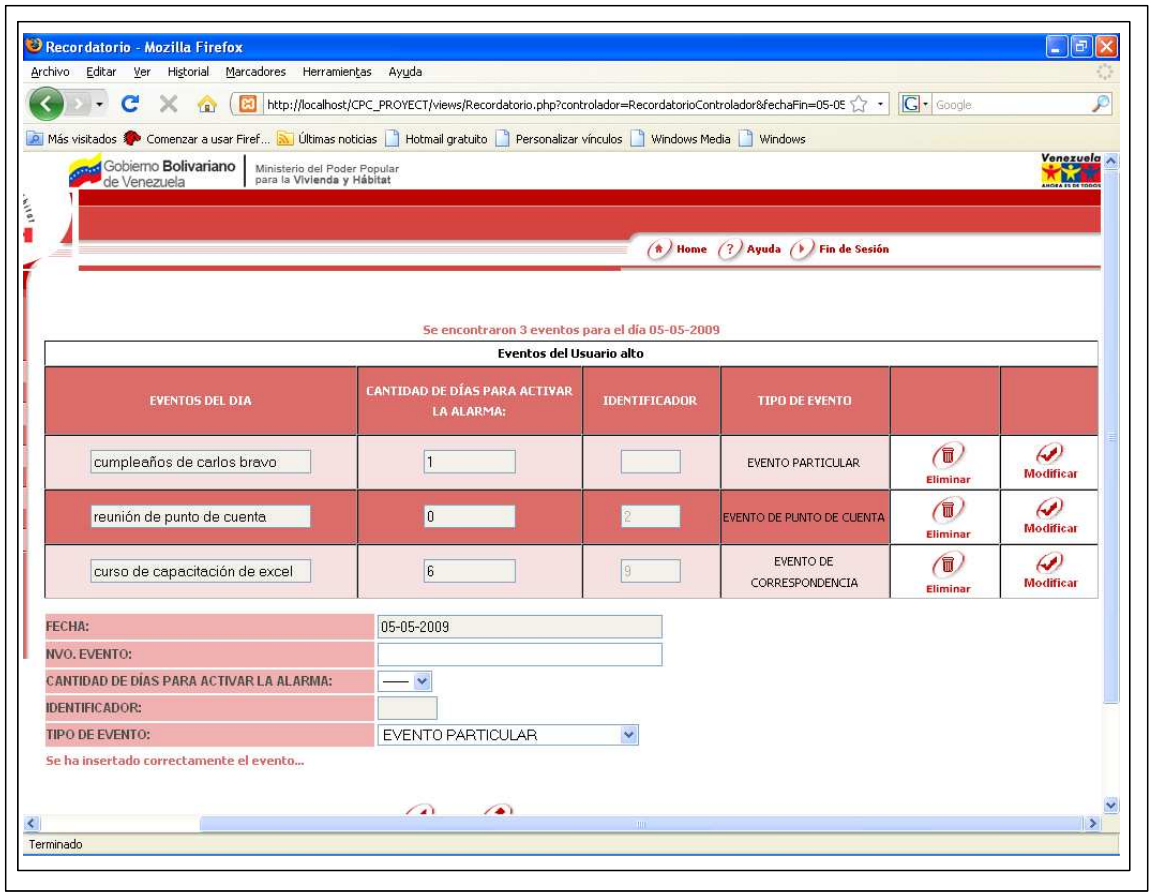

Figura 3.23. Foto del registro de recordatorios en el sistema (Fuente: Elaboración propia**).**

3.1.3 Iteraciones Correspondientes al Módulo de Administración

En este apartado se explica el desarrollo de todas las funcionalidades necesarias para construir el módulo de administración.

3.1.3.1 Iteración 6

En esta iteración se estudian todas las funcionalidades relacionadas con los usuarios, pistas de seguimiento (auditoria) de los usuarios y opciones del sistema.

3.1.3.1.1 Planificación de la iteración 6

La planificación de esta iteración, consiste en las historias de usuarios de la Tabla 3.14 que se especifica a continuación:

Historias de Usuarios para la 6ta. Iteración		
Fecha	**Usuario**	**Descripción**
09/01/2009	Ing. Carlos Moreno	Registrar un usuario con los siguientes ítems: *Login*, *Password*, Nombre, Apellido, Cargo, Gerencia y Rol.
12/01/2009 – 13/01/2009	Ing. Carlos Moreno	Consultar usuarios mediante los siguientes parámetros: campos de registro de usuario; además del estado del mismo en el sistema (activo ó inactivo).
14/01/2009	Ing. Carlos Moreno	Modificar los datos de un usuario.
14/01/2009	Ing. Carlos Moreno	Eliminar un usuario del Sistema. Este borrado no será físicamente, sino que internamente el Administrador podrá activar o desactivar un usuario del sistema. Esto a fin de mantener un histórico de los

		usuarios del Banco.
15/01/2009 - 16/01/2009	Ing. Carlos Moreno	Se debe poder insertar nuevas opciones para los siguientes ítems: Documentos, Organización, Materia, estatus de una Correspondencia, estatus de un Punto de Cuenta y prioridad de una correspondencia.
16/01/2009	Ing. Carlos Moreno	Modificar el *Password* del usuario.
19/01/2009 – 20/01/2009	Ing. Carlos Moreno	El sistema debe poder seguir pistas de auditoria que reflejen las acciones de los usuarios en el mismo.

Tabla 3.14. Formato de Historia de Usuarios para la Sexta Iteración perteneciente al Módulo de Administración (Fuente: Elaborado por el representante de BANAVIH)

3.1.3.1.2 Diseño de la iteración 6

El diseño de esta iteración contiene los siguientes diagramas: Diagrama de Entidad/Relación, Modelo Lógico y Diagrama de clases.

- **Diagrama Entidad/Relación**

El diagrama Entidad/Relación muestra las entidades y relaciones de la iteración 5 e incorpora una nueva entidad que es la auditoria del sistema. Esta entidad almacena pistas de seguimiento o auditoria, la cual se encarga de registrar todas las acciones de escritura que el usuario ejecuta

en el sistema (insertar, modificar y eliminar). La Figura 3.24 muestra lo siguiente: un usuario puede producir múltiples auditorias en la aplicación y una auditoria puede ser producida por un sólo usuario.

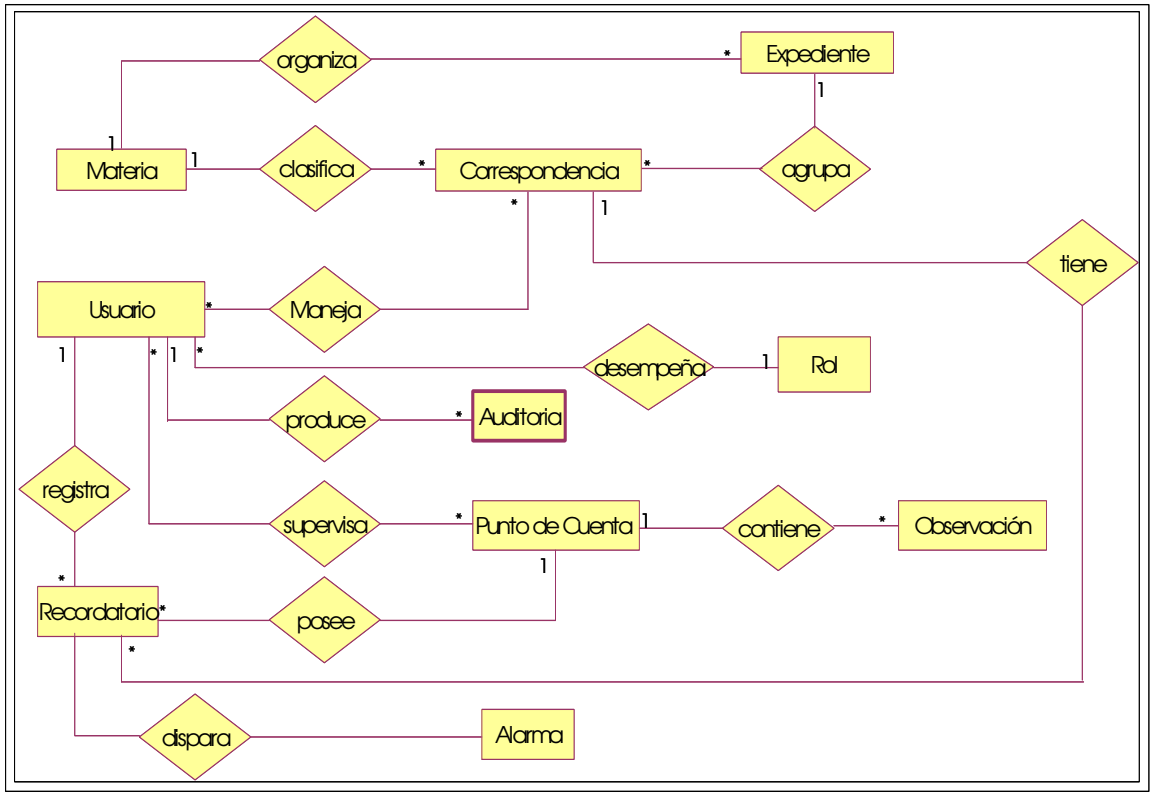

Figura 3.24. Diagrama Entidad/Relación, perteneciente a la Sexta iteración del Módulo de Administración (Fuente: Elaboración propia).

- **Modelo Lógico**

El modelo lógico de esta iteración hace referencia a las tablas, claves primarias, claves ajenas, índices y relaciones de la iteración 5 y adicionalmente se incorpora la tabla auditoria con la finalidad de poder registrar el seguimiento de las acciones del usuario en el sistema (Ver figura 3.25).

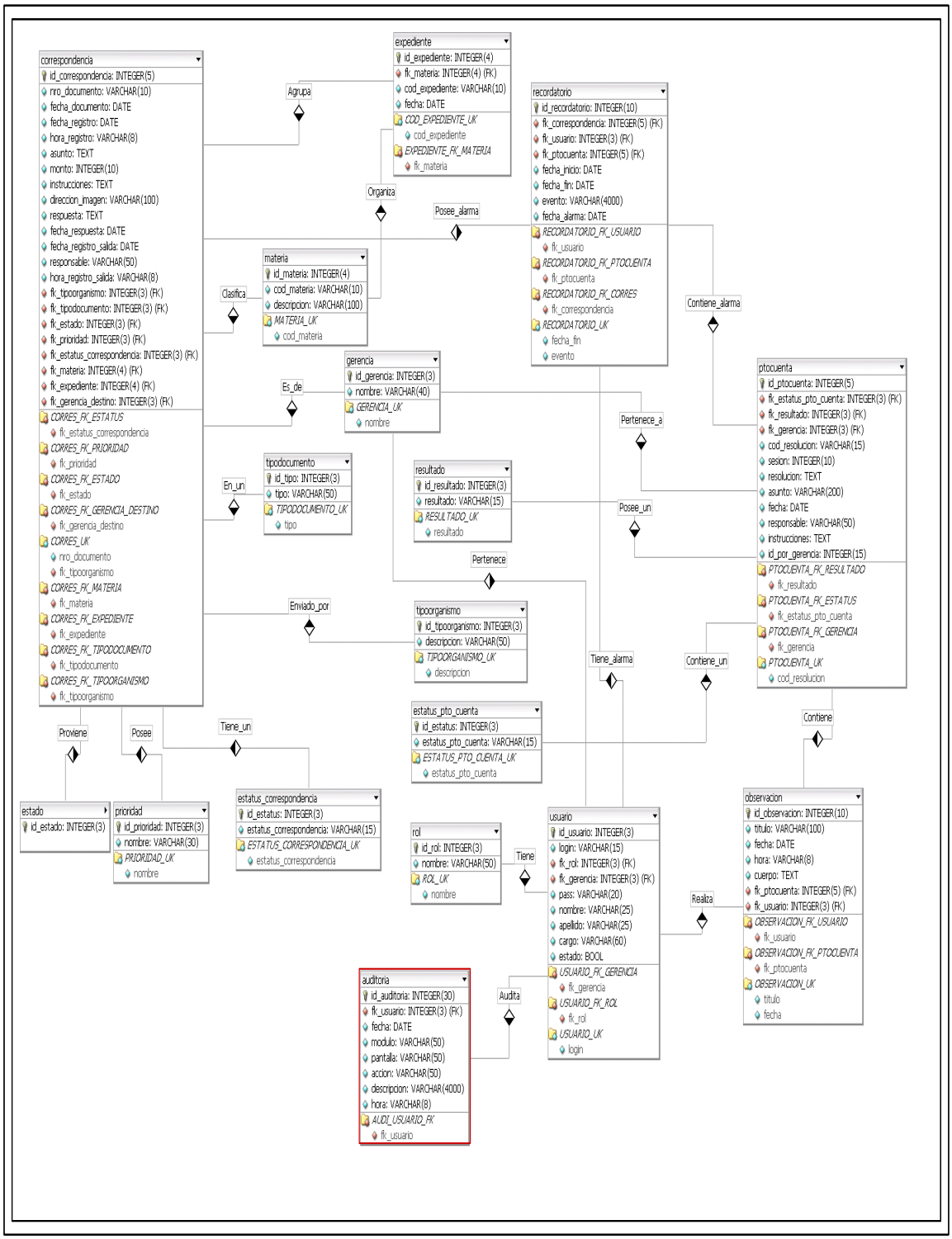

Figura 3.25. Modelo Lógico, perteneciente a la Sexta iteración del Módulo de Administración (Fuente: Elaboración propia).

La figura 3.25 muestra la tabla auditoria anexada en la iteración, con los siguientes campos:

- Id_auditoria: clave primaria que indica que el registro de la auditoria es único en el sistema.
- Fecha: fecha en la que la auditoria se registra en el sistema.
- Modulo: módulo que registró la acción.
- Pantalla: nombre de la interfaz que ejecutó la acción.
- Accion: Ejecución de una acción insertar, modificar o eliminar.
- Descripción: query que indica la acción que se ejecuta en el sistema.
- Hora: hora de registro de la auditoria.
- Clave ajena: fk_usuario (identificador del usuario que ejecutó la acción dentro del sistema).
- Índice: AUDI_USUARIO_FK.

- **Diagrama de Clases**

El diagrama de clases de esta iteración consiste en las clases de los modelos pertenecientes al patrón MVC del módulo de Administración, como lo son: DocumentoModel, PrioridadModel, EstatusModel, AuditoriaModel, TipoOrganismoModel, UsuarioModel, MateriaModel, los cuales permiten ejecutar las funcionalidades especificadas en las historias de usuarios de esta iteración (Ver Figura 3.26).

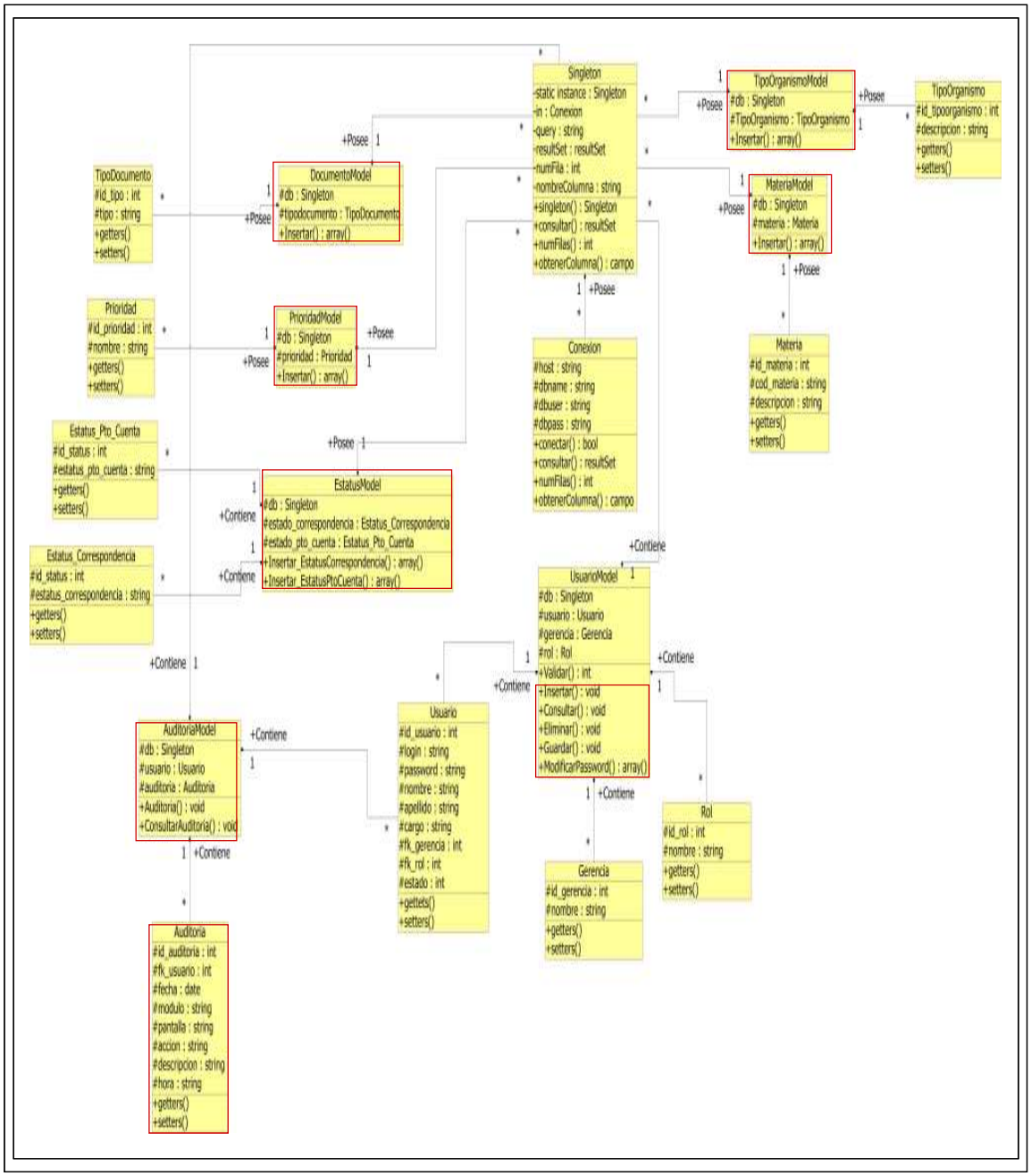

Figura 3.26. Diagrama de Clases, perteneciente a la Sexta Iteración del Módulo de Administración (Fuente: Elaboración propia).

La figura 3.26 muestra lo siguiente:

- DocumentoModel: Contiene la lógica necesaria para desarrollar las funcionalidades del tipo de documento de la correspondencia.

Los atributos de la clase DocumentoModel son:

- db: instancia de la clase *Singleton*.
- tipodocumento: instancia de la clase TipoDocumento.

El método de la clase DocumentoModel es:

- Insertar: registra nuevos documentos.

- MateriaModel: Contiene la lógica necesaria para desarrollar las funcionalidades del tipo de materia de la correspondencia.

Los atributos de la clase MateriaModel son:

- db: instancia de la clase *Singleton*.
- materia: instancia de la clase Materia.

El método de la clase MateriaModel es:

- Insertar: registra nuevas materias.

- PrioridadModel: Contiene la lógica necesaria para desarrollar las funcionalidades de la prioridad de la correspondencia.

Los atributos de la clase PrioridadModel son:

- db: instancia de la clase *Singleton*.
- prioridad: instancia de la clase Prioridad.

El método de la clase PrioridadModel es:

- Insertar: registra nuevas prioridades.

o AuditoriaModel: Contiene la lógica necesaria para desarrollar las funcionalidades de la auditoria del sistema.

Los atributos de la clase AuditoriaModel son:

- db: instancia de la clase *Singleton*.
- usuario. Instancia de la clase Usuario.
- auditoria: instancia la clase Auditoria.

Los métodos de la clase AuditoriaModel son:

- Auditoria: registra las auditorias del sistema.
- ConsultarAuditoria: permite ejecutar las consultas de las pistas de seguimiento (auditoria), de acuerdo a los criterios de búsqueda del usuario.

o TipoOrganismoModel: Contiene la lógica necesaria para desarrollar las funcionalidades del tipo de organismo de la correspondencia.

Los atributos de la clase TipoOrganismoModel son:

- db: instancia de la clase *Singleton*.
- tipoorganismo: instancia de la clase TipoOrganismo.

El método de la clase TipoOrganismoModel es:

- Insertar: registra nuevos organismos.

- EstatusModel: Contiene la lógica necesaria para desarrollar las funcionalidades del estatus de la correspondencia y de los puntos de cuenta.

Los atributos de la clase EstatusModel son:

- db: instancia de la clase *Singleton*.
- estado_correspondencia: instancia de la clase estatus_correspondencia.
- estado_pto_cuenta: instancia de la clase estatus_pto_cuenta.

Los métodos de la clase EstatusModel son:

- Insertar_EstatusCorrespondencia: registra nuevos estatus de correspondencia.
- Insertar_EstatusPtoCuenta: registra nuevos estatus de puntos de cuenta.

Los métodos agregados en la clase UsuarioModel son:

- Insertar: registra la información de los usuarios en el sistema.
- Consultar: consulta la información de los usuarios, a través de los criterios indicados por los usuarios.
- Eliminar: coloca en estado activo o inactivo los usuarios en el sistema.
- Guardar: actualiza la información de los usuarios.
- ModificarPassword: modifica la clave de los usuarios.

3.1.3.1.3 Codificación de la iteración 6

Para la iteración 6 se toma en cuenta las acciones y funciones de los Modelos: UsuarioModel, AuditoriaModel, DocumentoModel, TipoOrganismoModel, MateriaModel, EstatusModel, PrioridadModel que manejan las historias de usuarios para el Módulo de Administración pertenecientes a la lógica del negocio del patrón MVC. Las acciones y funciones de los modelos indicados anteriormente, se resumen en:

- Las acciones y la función del Modelo UsuarioModel permiten registrar, consultar, eliminar y actualizar un usuario. También permiten cambiar la clave de acceso del usuario.
- Las acciones de AuditoriaModel permiten registrar y consultar las pistas de seguimientos de las acciones del usuario en el sistema.
- Función de DocumentoModel para registrar el tipo de documento de la correspondencia.
- Función de TipoOrganismoModel para registrar el organismo que envía la correspondencia.
- Función de MateriaModel para registrar el tipo de materia al que pertenece la correspondencia.
- Funciones de EstatusModel para registrar el estatus de una correspondencia y un punto de cuenta en el sistema.
- Función de PrioridadModel para registrar la prioridad de la correspondencia.

3.1.3.1.4 Pruebas de Aceptación y Resultados del Sistema para la iteración 6

En esta iteración se produjo un error, debido a que las funcionalidades que tienen que ver con la inserción de nuevas opciones para los ítems: Documentos, Organización, Materia, estatus de una Correspondencia, estatus de un Punto de Cuenta y prioridad de una correspondencia, habían sido asignadas a la Unidad de Documentación. En este caso el Ing. Carlos Moreno indicó, que este tipo de información sólo podía ser manejada por el administrador, debido a que el usuario podía insertar cualquier información redundante para las opciones. Por ejemplo, el usuario A inserta "status" como nueva opción del campo estatus, mientras que el usuario B lo hará con "estado". En el ejemplo, el sistema maneja dos opciones con distintas palabras (status y estado) para este campo, cuando realmente significan lo mismo.

En este caso no será colocada la prueba de aceptación, debido a que el error que se presentó fue en el establecimiento de las respectivas funcionalidades que debe tener cada rol (Unidad de Documentación y Administración). Lo único que se resalta, es que las actividades de inserción de las opciones anteriormente mencionadas, fueron removidas del rol "Unidad de Documentación" y asignadas al rol "Admin".

El Ing. Carlos Moreno supervisó que todo se cumpliera a cabalidad, por lo que quedó conforme con todos los resultados obtenidos en el sistema.

Los resultados del sistema para la iteración 6, se aprecian en la figura 3.27. Esta figura muestra la consulta de las pistas de seguimiento de las acciones del usuario en el sistema.

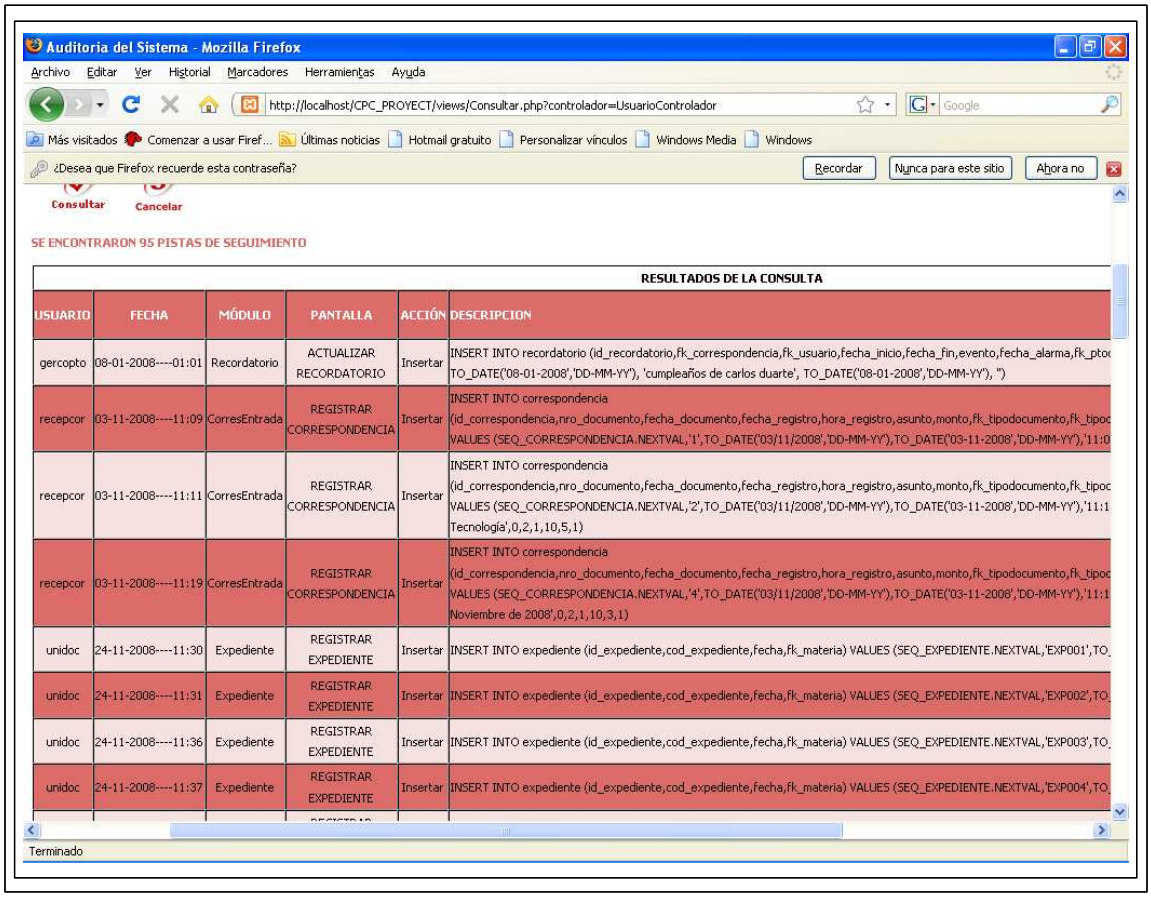

Figura 3.27. Foto de la consulta de auditoria del sistema (Fuente: Elaboración propia).

3.1.3.2 Iteración 7

En esta iteración se especifican las últimas funcionalidades del rol administador.

3.1.3.2.1 Planificación de la iteración 7

La planificación de esta iteración, consiste en las historias de usuarios de la Tabla 3.15 que se especifica a continuación:

Historias de Usuarios para la 7ma. Iteración		
Fecha	**Usuario**	**Descripción**
21/01/2009 – 22/01/2009	Ing. Carlos Moreno	El *Root* debe poder manipular las actividades del *Workflow* del sistema.
22/01/2009 – 23/01/2009	Ing. Carlos Moreno	El *Root* debe poder manipular los roles del *Workflow* del sistema.
23/01/2009 – 28/01/2009	Ing. Carlos Moreno	La aplicación debe tener la posibilidad de asignar y eliminar actividades relacionadas a un rol, permitiendo que el mismo en cualquier momento tenga el comportamiento que la organización requiera (ejecute o no actividades determinadas). De esta manera, se deberá tener un Workflow del sistema dinámico.
28/01/2009 – 30/01/2009	Ing. Carlos Moreno	Al cambiarse las actividades de un rol, se deberán cargar dinámicamente las opciones del usuario en el sistema (Menú Dinámico), sin necesidad de modificar el código de la Aplicación.

Tabla 3.15. Formato de Historia de Usuarios para la Séptima Iteración perteneciente al Módulo de Administración (Fuente: Elaborado por el representante de BANAVIH)

3.1.3.2.2 Diseño de la iteración 7

El diseño de esta iteración se fundamenta sólo en el Diagrama Entidad/Relación, Modelo Lógico y Diagrama de clases.

- **Diagrama Entidad/Relación**

El diagrama de Entidad/Relación muestra todas las relaciones y entidades finales del sistema. Este diagrama contiene las entidades y relaciones de la iteración 6 e incorpora una nueva entidad llamada actividad, que permite el manejo de las actividades y la asignación de las mismas, a los diferentes roles del sistema. Como se observa en la figura 3.28, cada usuario desempeña un rol y éste, a su vez, realiza múltiples actividades. Cada actividad puede ser realizada por múltiples roles y un rol puede ser desempeñado por múltiples usuarios.

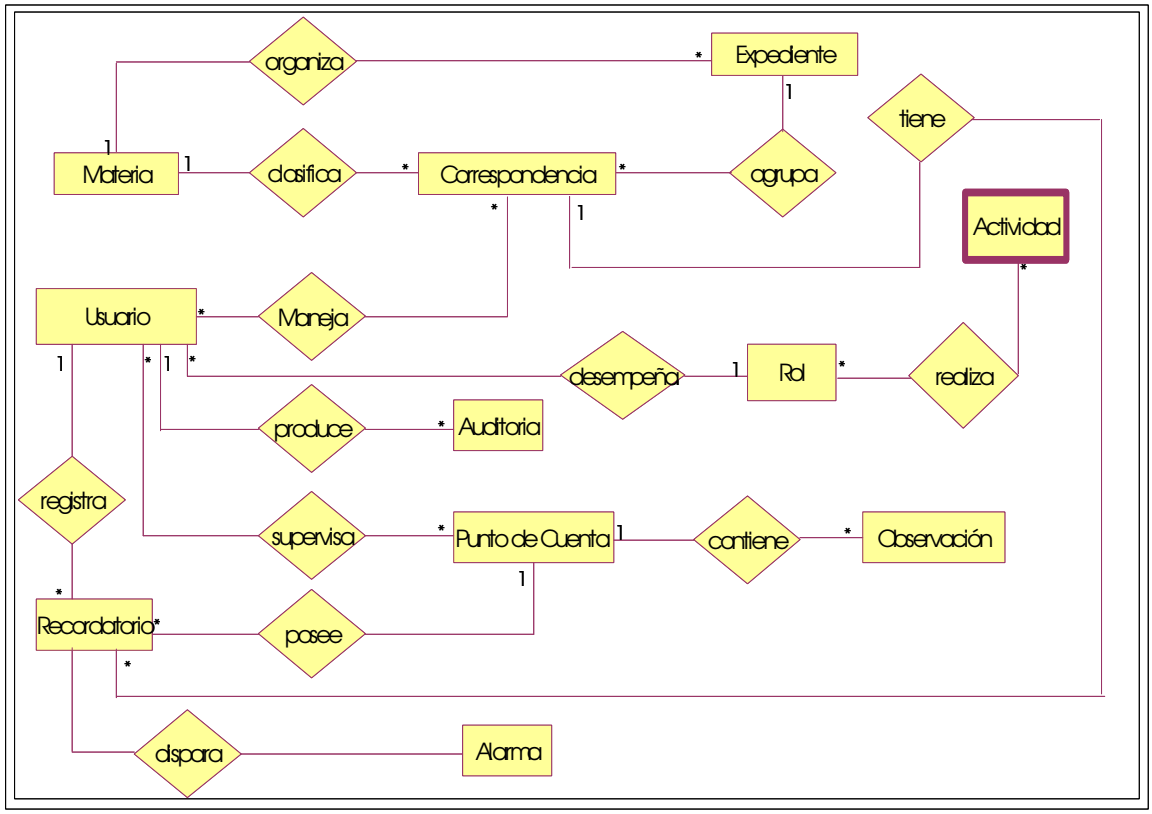

Figura 3.28. Diagrama Entidad/Relación, perteneciente a la Séptima Iteración del Módulo de Administración (Fuente: Elaboración propia).

A través de las actividades es posible que cualquier rol pueda ejecutar una actividad si es asignada por el administrador del sistema, lo que permite mayor flexibilidad y un manejo dinámico de las tareas necesarias en el sistema.

- **Modelo Lógico**

El modelo lógico de esta iteración muestra las tablas, claves primarias, claves ajenas, índices y relaciones de la iteración 6 e incorpora nuevas tablas, como lo son: rol_actividad, actividad y modulo, las cuales permiten el manejo dinámico de las actividades que ejecuta cada usuario en el sistema. La figura 3.29 muestra estas tablas y los campos que se describen a continuación:

- rol_actividad: contiene las actividades asignadas a los diversos roles del sistema.

Los campos de la tabla rol_actividad son:

- claves ajenas: fk_rol(identificador del rol) y fk_actividad (identificador de la actividad).
- Índices: ROL_ACTIVIDAD_FK_ROL y ROL_ACTIVIDAD_FK_ACTIVIDAD.

- actividad: contiene las actividades del sistema.

Los campos de la tabla actividad son:

- id_actividad: clave primaria de la actividad que indica que el registro de la actividad debe ser único en el sistema.
- actividad: nombre de la actividad.
- direccion_actividad: dirección en la que se puede acceder la interfaz de la actividad.
- clave ajena: fk_modulo (identificador del módulo).
- índices: ACTIVIDAD_FK_MODULO y ACTIVIDAD_UK (indica que la actividad debe ser única en el sistema).

- modulo: contiene los módulos del sistema, por ejemplo: CorresEntrada, CorresSalida, Expediente, Rol, etc.

Los campos de la tabla modulo son:

- id_modulo: clave primaria que indica que el registro del módulo debe ser único en el sistema.
- nombre: nombre del módulo.
- Índice: MODULO_UK (indica que el nombre del módulo debe ser único en el sistema).

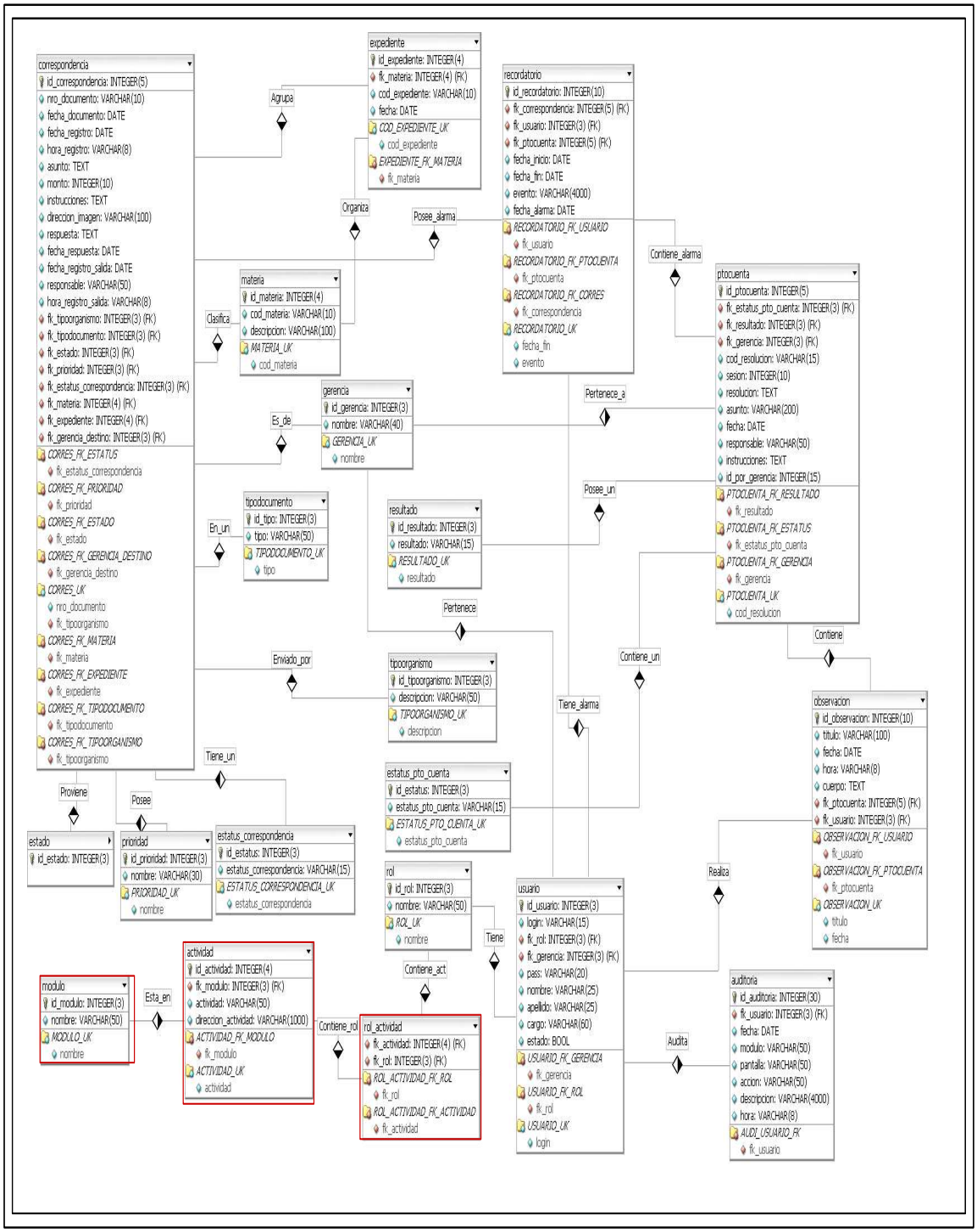

Figura 3.29. Modelo Lógico, perteneciente a la Séptima Iteración del Módulo de Administración (Fuente: Elaboración propia).

- **Diagrama de Clases**

El diagrama de clases (Ver Figura 3.30) de esta iteración, muestra todas las clases pertenecientes al módulo de Administración. Además, muestra nuevas clases como lo son: Modulo, Actividad, Rol_Actividad, ActividadModel y RolModel.

Las clases Modulo, Actividad y Rol_Actividad no son descritas, debido a que análogamente, las entidades relacionadas a éstas, se describieron en el modelo lógico.

A continuación se describen las clases ActividadModel y RolModel de la figura 3.30:

- ActividadModel: Maneja las funcionalidades de las historias de usuarios relacionadas con las actividades.

Los atributos de la clase ActividadModel son:

- db: instancia de la clase *Singleton*.
- rol: instancia de la clase rol.
- actividad: instancia de la clase actividad.
- modulo: instancia de la clase modulo.

Los métodos de la clase ActividadModel son:

- BuscarActividad: consulta las actividades de los roles.
- Registrar: inserta el nombre y las actividades del rol.
- Consultar: consulta la información de las actividades, a través de los criterios de búsqueda definidos por el usuario.
- Eliminar: elimina una actividad.

- Guardar: actualiza la información de los campos (actividad, direccion_actividad y modulo) del rol.

o RolModel: Maneja las funcionalidades de las historias de usuarios relacionadas con los roles.

Los atributos de la clase RolModel son:

- db: instancia de la clase *Singleton*.
- rol: instancia de la clase rol.
- rol_actividad: instancia de la clase rol_actividad.
- actividad: instancia de la clase actividad.

Los métodos de la clase RolModel son:

- Registrar: inserta la información de los roles (rol y actividades asociadas).
- Consultar: consulta la información de los roles, a través de los criterios de búsqueda (rol) definidos por el usuario.
- Eliminar: elimina un rol.
- Guardar: actualiza la información del rol (nombre del rol y sus actividades).

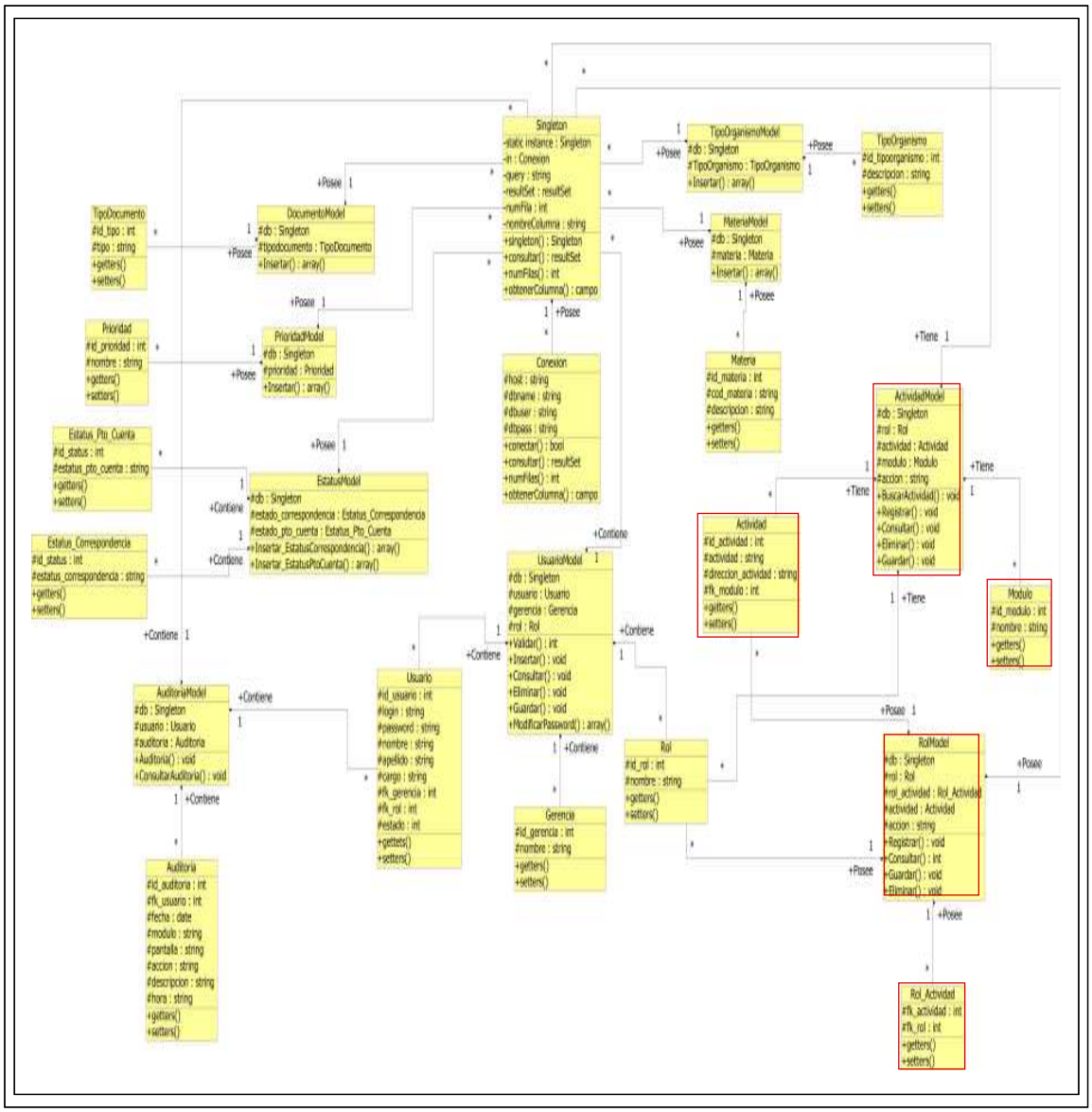

Figura 3.30. Diagrama de Clases, perteneciente a la Séptima Iteración para el Módulo de Administración (Fuente: Elaboración propia).

3.1.3.2.3 Codificación de la iteración 7

Para la iteración 7 se toma en cuenta las acciones y funciones de los Modelos: ActividadModel y RolModel que manejan las historias de usuarios para el Módulo de Administración (Ver *anexos A20, A21 y A22*).

En esta iteración se implementó un menú dinámico, con la finalidad de poder flexibilizar el manejo de actividades para los roles de los usuarios; es decir, se tiene la posibilidad de asignar o eliminar actividades a los roles del usuario, sin tener que modificar el código de las plantillas de las vistas.

El menú dinámico invoca a la acción BuscarActividad, que obtiene la dirección, el nombre y el módulo al que pertenecen las actividades.

Cabe destacar, que anteriormente existía un menú estático para cada rol de la aplicación, con las funcionalidades que debían ser manejadas por el usuario, dependiendo de su rol; por lo que cada vez que se quería modificar las actividades de un rol se debía cambiar las opciones del menú en el código de las plantillas de PHP.

La codificación de esta iteración contempla lo siguiente:

- Acciones del Modelo ActividadModel que permiten buscar las actividades asociadas a los roles del sistema, registrar, consultar, eliminar y actualizar actividades.
- Acciones y función del Modelo RolModel que permiten registrar, consultar, eliminar y actualizar los roles del sistema.
- Refactorización del código del menú: En este caso, cada rol obtiene sus actividades a partir de un solo menú (dinámico) que muestra las opciones que debe desplegar.

3.1.3.2.4 Pruebas de Aceptación y Resultados del Sistema para la iteración 7

En esta iteración, el único error encontrado fue al momento de construir el menú dinámico. Cuando las actividades de un mismo módulo no eran contiguas en base de datos, el motor *Workflow* repetía un mismo módulo en la construcción del menú, agregándolo de nuevo al final del mismo.

El Ing. Carlos Moreno supervisó que el error fue solucionado, por lo que quedó conforme con los resultados obtenidos.

El formato 3.16 describe la prueba que se ejecutó para el menú dinámico.

- **Menú dinámico**

Nombre del Sistema **Sistema de Información para el manejo de Correspondencias y Puntos de Cuenta**					
Módulo:	Administración				
Historia de Usuarios	Al cambiarse las actividades de un rol, se deberán cargar dinámicamente las opciones del usuario en el sistema (Menú Dinámico), sin necesidad de modificar el código de la Aplicación.				
Caso de Prueba:	Verificar que las actividades asignadas a un rol se carguen correctamente.				
Versión de caso de prueba:	Versión 1	**Fecha ejecución:**	03/02/2009		
Nombre del Probador:	Oscar Ruiz				
Precondiciones: El administrador debe asignar las actividades al rol del usuario.					
Para la ejecución del Caso de Prueba:					
Paso	**Condición**	**Valor(es)**	**Resultado esperado**	**Resultado Obtenido**	
Desplegar menú dinámico			Debe mostrar el menú dinámico del lado izquierdo con los módulos y actividades asociadas al rol.	El menú despliega repetido un mismo módulo si las actividades no son contiguas en base de datos, por lo que siempre agrega una repetición al final de la construcción del menú.	
Observaciones del Caso de Prueba					
Decisión de Aprobación de Caso de Prueba:		**Aprobó:**		**Falló:**	√
Fecha de Aprobación del Caso de Prueba:				**Aprobado por:**	Ing. Carlos Moreno
Ajustado por:	Oscar Ruiz		**Fecha de Ajuste:**	03/02/2009	
Prueba de Ajuste, en fecha:	03/02/2009	**Decisión de Aprobación de Ajuste:**	**Aprobó:**	√	**Falló:**

Tabla 3.16. Formato de Caso de Prueba perteneciente a la funcionalidad Menú dinámico al Módulo de Administración (Fuente: Formato establecido por el representante de BANAVIH)

La figura 3.31 muestra las actividades asignadas al rol administrador.

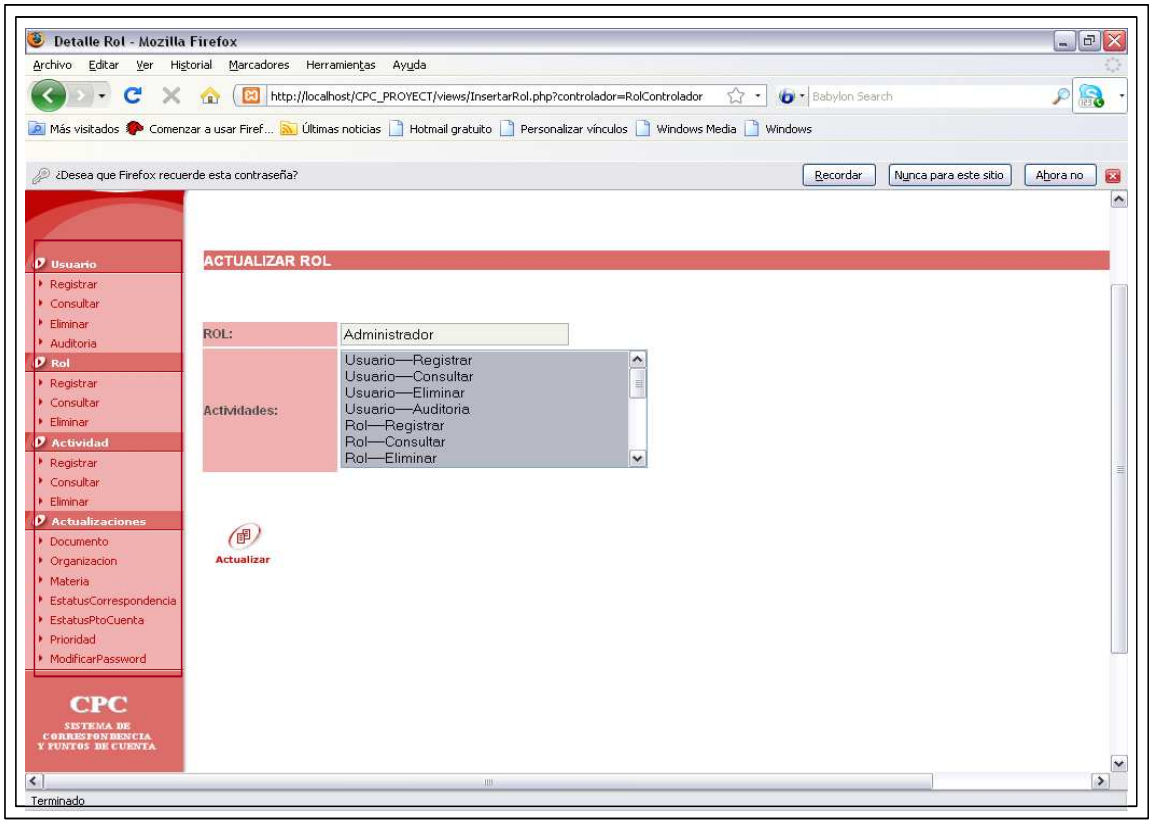

Figura 3.31. Foto anterior de la actualización del rol administrador del sistema (Fuente: Elaboración propia)

En la figura 3.32 se cambia las actividades asignadas al rol administrador. Esto se puede apreciar en el menú dinámico mostrado del lado izquierdo. Nótese, que dentro de las nuevas actividades no se encuentran las actividades relacionadas al módulo de actualizaciones.

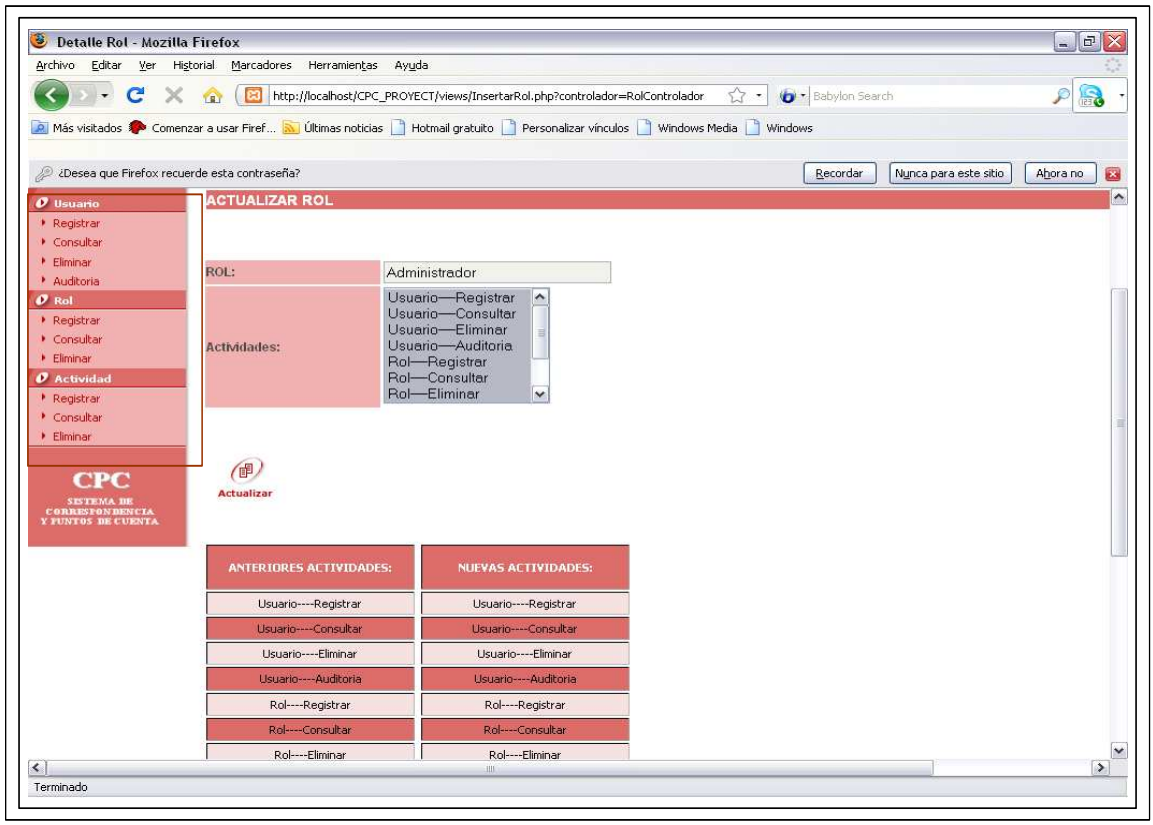

Figura 3.32. Foto posterior a la actualización del rol administrador del sistema (Fuente: Elaboración propia).

Una vez desarrollado los módulos de correspondencia, puntos de cuenta y administración (Ver Figura 3.3), se procedió a la integración de los mismos, dando como resultado al Sistema de Información para el Manejo de Correspondencia y Puntos de cuenta, tal como se aprecia en la Figura 3.33.

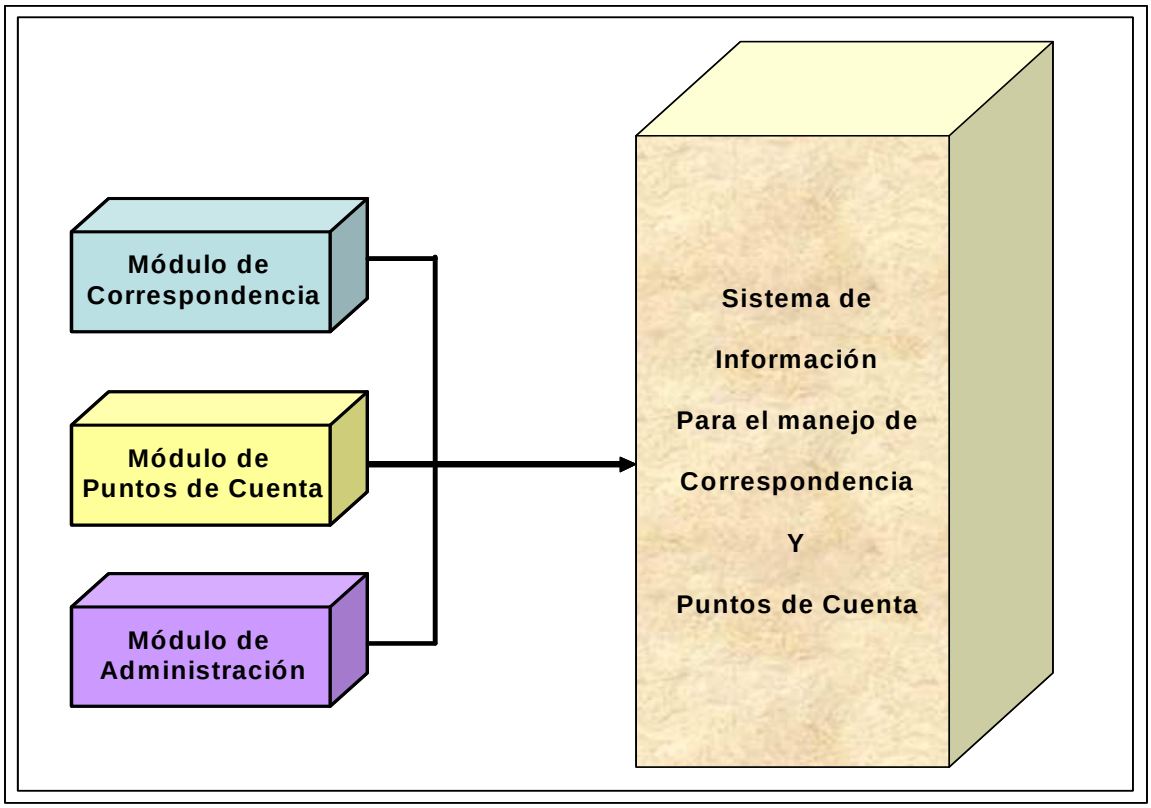

Figura 3.33. Integración de los módulos de puntos de cuenta, correspondencia y administración en el Sistema (Fuente: Elaboración propia).

CONCLUSIONES

Partiendo de los objetivos trazados al inicio del presente trabajo, y a través de la entrega y refinamiento del sistema en cada una de las iteraciones (siete en total) para el cumplimiento de lo especificado en las historias de usuario, podemos afirmar las siguientes conclusiones:

- El proceso de desarrollo *XP*, se adaptó a la construcción de funcionalidades de cada iteración/calle del *workflow.*
- Mediante *XP*, el cliente tomó confianza en el producto a medida que el proyecto avanzó, ya que se involucró en el desarrollo del mismo, desde la primera iteración.
- A través del Patrón MVC y el proceso *XP*, se intercambiaron los roles de trabajo entre iteraciones: mientras un miembro del equipo de desarrollo trabajaba en la presentación de los datos, el otro se enfocaba en la lógica del negocio y viceversa; de esta forma, se aceleró la velocidad de construcción del sistema y permitió a cada desarrollador intervenir en cada una de las capas de la arquitectura MVC, por lo que cada uno obtuvo una compresión integral de toda la estructura de la aplicación.
- El sistema de control de versiones subversion, permitió administrar de forma ordenada y esquemática las versiones del proyecto, debido a que en algunas oportunidades era necesario retomar versiones de archivos anteriores en el ciclo de desarrollo.
- La prueba piloto de toda la aplicación no se realizó, debido a que BANAVIH tiene su propio departamento de pruebas funcionales, a las que el equipo de desarrollo no tuvo acceso.

- La implementación del Patrón *Singleton* permitió establecer una sola instancia de conexión a la base de datos, ya que sin este patrón se podría tener múltiples conexiones innecesarias a la misma; pudiendo sobrecargar el sistema y desmejorando los tiempos de respuestas para la aplicación.

Además, se plantean las siguientes recomendaciones:

- Migrar el motor *workflow* de la aplicación a una solución *BPM* (*Business Process Management*) para extender las funcionalidades del motor *workflow* del presente caso de estudio, en aspectos como: manejo de notificaciones, creación de nuevos procesos, identificación de cuellos de botella en los procesos, entre otros.
- Realizar un estudio de usabilidad a fin de medir que tan fácil es para los usuarios utilizar el sistema y, con base en estos resultados, hacer los correctivos correspondientes. De esta manera sería posible permitir a los usuarios llevar a cabo sus tareas dentro del sistema, de manera simple e intuitiva.

Glosario

Business Process Management (BPM): Es "...la aplicación de técnicas y herramientas software para modelizar, gestionar y optimizar los procesos de negocio de la organización" (Ibermática, (s.f).).

Interfaz gráfica de usuario (*GUI*): Consiste en una interfaz gráfica de usuario que utiliza imágenes y gráficos para representar información y acciones, a fin de establecer una interacción humano computador.

Interfaz de programación de aplicaciones (*API*): Agrupación de funciones y procedimientos ofrecidos por una biblioteca estableciendo abstracción sobre su funcionamiento interno.

Lenguaje de Marcado de Hipertexto (*HTML*): "Sencillo Lenguaje de formateo de documentos de hipertexto que utiliza etiquetas para indicar la forma en que una parte dada de un documento se debería interpretar por una aplicación de visualización, como un navegador web" (Academia de Networking de Cisco Systems, (2004), pp. 914).

Localizador Uniforme de Recurso (*URL*): Es la dirección que se utiliza para acceder a documentos y otros recursos en la *Web*.

Object Management Group (OMG): Es un consorcium que tiene la tarea de desarrollar normas de integración de empresas para varias tecnologías, entre ellas UML, y para un amplio rango de industrias.

Patrón de diseño de tipo creación: Son aquellos utilizados para crear instancias de clases.

Protocolo del control de transporte/Protocolo Internet (*TCP/IP*): Son protocolos desarrollados por el DoD de Estados Unidos en la década de los 70's para desarrollar redes en todo el mundo.

Protocolo de Transferencia de Hipertexto (*HTTP*): "Protocolo utilizado por los navegadores web y los servidores web para transferir archivos, como archivos de texto y gráficos" (Academia de Networking de Cisco Systems, (2004), pp. 914).

Redes de Área Local (LANs): Son redes que interconectan estaciones de trabajo, dispositivos, terminales y periféricos en un edificio; son de alta velocidad, y están limitados a un área geográfica pequeña.

Referencias Bibliográficas

Academia de Networking de Cisco Systems, (2004). Guía del Primer Año CCNA 1 y 2. (Tercera Edición). Madrid: Editorial Pearson Educación.

Banco Nacional de la Vivienda y Hábitat. (s.f). La Institución. (Página Web en Línea). (Consultado el día 12 de octubre de 2008 en la World Wide Web: http://www.banavih.gob.ve/index.php?option=com_content&task=view&id=1&Itemid=6

Kendall, Kenneth E., & Kendall, Julie E. (2005). Análisis y Diseño de Sistemas. (6ta edición). México: Editorial Pearson Prentice Hall.

Ibermática, (s.f). BPM - Business Process Management. (Página Web en Línea). Consultado el día 17 de Abril de 2009 en la World Wide Web: http://www.ibermatica.com/ibermatica/bpm

International Engineering Consortium (2007). Intranet Business Applications. (Documento en Línea). Consultado el día 16 de Diciembre de 2008 en la World Wide Web: http://www.iec.org/online/tutorials/int_bus/topic04.asp

López Rivera, Alejandro (16 de enero de 2008). Sistema asistente para la generación de horarios de cursos, Capítulo 2, Universidad de las Américas Puebla, México. (Tesis de Grado en Línea). Consultada el día 06 de Octubre de 2008 en la World Wide Web: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rivera_l_a/capitulo_2.html#

Microsoft patterns & practices Developer Center (s.f). Singleton. (Librería MSDN en Línea). Consultado el día 07 de octubre de 2008 en la World Wide Web: http://msdn.microsoft.com/es-ve/library/ms998426(en-us).aspx

Molpeceres, Alberto. (Revisado 15.2.2003). Proceso de desarrollo: RUP, XP y FDD. (PDF en Línea). Consultado el día 01 de Octubre de 2008 en la World Wide Web: http://www.willydev.net/descargas/Articulos/General/cualxpfddrup.PDF

O'Brien, James A. (2001), Introduction to Information Systems. (Décima Edición). Boston: Editorial McGraw-Hill.

Sun Microsystems (s.f). Java BluePrints Model – View – Controller. Consultado el día 08 de Octubre de 2008 en la World Wide Web: http://java.sun.com/blueprints/patterns/MVC-detailed.html

Telleen, Steven L. (1996, Junio) The IntraNet Architecture: Managing information in the new paradigm. (Página Web en Línea). Consultado el día 15 de Diciembre de 2008 de la World Wide Web: http://www.iorg.com/papers/amdahl/infra.html#RTFToC5

Welicki, León (s.f). Patrones y Antipatrones: una Introducción – Parte I. (Librería MSDN en Línea). Consultado el día 07 de Octubre de 2008 en la World Wide Web: http://msdn.microsoft.com/es-ve/library/bb972242.aspx#XSLTsection125121120120

Welicki, León (s.f). Patrones y Antipatrones: una Introducción – Parte II. (Librería MSDN en Línea). Consultado el día 08 de Octubre de 2008 en la World Wide Web: http://msdn.microsoft.com/es-es/library/bb972251.aspx

Welicki, León (s.f). El Patrón Singleton. (Librería MSDN en Línea). Consultado el día 07 de octubre de 2008 en la World Wide Web: http://msdn.microsoft.com/es-ve/library/bb972272.aspx#EDAA

Workflow Management Coalition (Febrero, 1999). Workflow Management Coalition, Terminology & Glossary. (PDF en Línea). Consultado el día 13 de Diciembre de 2008 en la World Wide Web: http://www.wfmc.org/index.php?option=com_docman&task=cat_view&gid=35&Itemid=74

Anexos

```
public function Insertar()
{ //Registra las correspondencias
//Obtiene los parámetros de entrada
for($i=0;$i<count($this->arrayelement_p);$i++){
if ($this->arrayelement_p[$i]!='accion' && $this->arrayelement_p[$i]!='accion_x'&& $this->arrayelement_p[$i]!='accion_y'){
if ($this->arrayelement_p[$i]=='nro_doc')
$this->correspondencia->set_nro_documento($_POST[$this->arrayelement_p[$i]]);
...
} //Fin IF
}//Fin For
$nro_doc = $this->correspondencia->get_nro_documento();
...
//Ejecución del Registro de la Correspondencia
$sql = "INSERT INTO correspondencia
(id_correspondencia,nro_documento,fecha_documento,fecha_registro,hora_registro,asunto,monto,fk_tipodocumento,fk_tipoorganismo,fk
_estado,fk_materia) VALUES
(SEQ_CORRESPONDENCIA.NEXTVAL,'$nro_doc',$fecha_documento,$fecha_registro,'$hora_registro','$asunto',$monto,$fk_tipodocume
nto,$fk_tipoorganismo,$fk_estado,$fk_materia)";
$resultado= $this->db->consultar($sql);
...
return $elementos;
}
```

Anexo A1. Código de la función Insertar del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function ConsultarEntradaRecepCoor(){
//Consulta las correspondencias de entrada
//Query de consulta de los valores requeridos de la correspondencia
$query = "SELECT id_correspondencia,nro_documento,asunto,
fk_materia FROM correspondencia
//Extrae los valores de los parámetros de búsqueda para la correspondencia
for($i=0;$i<count($this->arrayelement_p);$i++){
if ($this->arrayelement_p[$i]!='accion_x' && $this->arrayelement_p[$i]!='accion_y'
&& $this->arrayelement_p[$i]!='accion' && $this->arrayelement_p[$i]!='consultar'
&& $this->arrayelement_p[$i]!='limpiar') {
if ($this->arrayelement_p[$i]=="nro_documento" && $_POST[$this->arrayelement_p[$i]]!=""){
$query = $query." AND nro_documento = '".$_POST[$this->arrayelement_p[$i]]."'";
$this->correspondencia->set_nro_documento($_POST[$this->arrayelement_p[$i]]);
}
…
}//Fin IF
}//FIN For
//Realiza la consulta de los parámetros de búsqueda para la correspondencia
$query = $query." ORDER BY fk_materia";
$resultado = $this->db->consultar($query);
$total = $this->db->numFilas($resultado);
…
return $elementos;
}
```

Anexo A2. Código de la función ConsultarEntradaRecepCoor del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function Consultar_Detalle_CorresRecepCoor(){
//Consulta el detalle de la correspondencia de entrada
//Obtiene los parámetros de entrada
for($i=0;$i<count($this->arrayelement_g);$i++){
if ($this->arrayelement_g[$i]=="id_correspondencia"){
$this->correspondencia->set_id_correspondencia($_GET[$this->arrayelement_g[$i]]);
$id_correspondencia = $this->correspondencia->get_id_correspondencia();
}
}//fin de for
//Consulta la correspondencia
$query = "SELECT
id_correspondencia,nro_documento,fecha_documento,monto,fk_tipodocumento,fk_estado,fk_tipoorganismo,fk_ma
teria,asunto FROM correspondencia WHERE  id_correspondencia = $id_correspondencia";
$resultado = $this->db->consultar($query);
…
}
```

Anexo A3. Código de Consultar_Detalle_CorresRecepCoor del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function Guardar_CorresRecepUni(){
//Modifica la correspondencia de entrada
//Se obtienen los datos de entrada - GET
for($i=0;$i<count($this->arrayelement_g);$i++){
if ($this->arrayelement_g[$i]=="id_correspondencia"){
$this->correspondencia->set_id_correspondencia($_GET[$this->arrayelement_g[$i]]);
$id_correspondencia = $this->correspondencia->get_id_correspondencia();
}
…
//Se extraen los datos
$num_documento = $this->correspondencia->get_nro_documento();
…
//Actualiza la correspondencia
$sql = "UPDATE correspondencia SET
nro_documento='$num_documento',fecha_documento=$fecha,asunto='$asunto',monto=$monto,fk_tipodo
cumento=$fk_tipodocumento,fk_tipoorganismo=$fk_tipoorganismo,fk_estado=$fk_estado,fk_materia=$fk_
tipomateria WHERE id_correspondencia=$id_correspondencia";
$resultado = $this->db->consultar($sql);
…
//Si hubo modificaciones
if($fech_s!=$fecha_documento)
{
…
$elementos[$cont] = "Fecha_Documento";
$cont = $cont + 1;
…
$_SESSION["fecha_documento"] = $fecha_documento;
}
return $elementos;
}
```

Anexo A4. Código de la función Guardar_CorresRecepUni del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function Validar(){

//Obtiene los valores de entrada
$cod = $this->codigo;
$valid = $this->img->check($cod);
$log=$this->usuario->get_login();
$cla=$this->usuario->get_password();
...
$query = "SELECT * FROM usuario WHERE ((UPPER(password)=UPPER('$cla')) AND (UPPER(login)=UPPER('$log')) AND
(estado=1))";
$resultado = $this->db->consultar($query);
//Verificamos si el usuario está registrado en el sistema
if ($this->db->numFilas($resultado) && ($valid)){
$this->usuario->set_id_usuario($this->db->obtenerColumna($resultado,0,"id_usuario"));
...
$_SESSION["id_usuario"] = $this->usuario->get_id_usuario();
$_SESSION["gerencia"] = $this->gerencia->get_nombre();
$_SESSION["rol"] = $this->rol->get_nombre();
$_SESSION["login"] = $log;
$permiso=1; //retorno el tipo de permiso en este caso el acceso al sistema es permitido
}
...
return $permiso;
}
```

Anexo A5. Código de la función Validar del Modelo UsuarioModel (Fuente: Elaboración propia).

```
class Singleton
{
static private $instance = NULL;

private function __construct()
{   //Instancia la conexion
$config = Configuracion::singleton();
$this->in = new Conexion($config->get('dbhost'),$config-
>get('dbname'),$config->get('dbuser'),$config->get('dbpass'));
$this->in->conectar();
}


static public function singleton()
{ //Llama a la instancia Singleton
if (self::$instance == NULL) {
self::$instance = new Singleton();
}
return self::$instance;
}

function consultar($query){
//Ejecuta los querys
return $this->in->consultar($query);
}
function numFilas($resultSet)
{//Retorna el número de filas del resultSet
return $this->in->numFilas($resultSet);
}
function obtenerColumna($resultSet, $numFila, $nombreColumna)
{ //Obtiene el campo del resultSet
return $this->in->obtenerColumna($resultSet, $numFila, $nombreColumna);
}
function cerrar()
{//Cierra la conexion
if ($this->in)
$this->in->cerrar();
else
return "";
}
}
```

Anexo A6. Clase Singleton y sus funciones (Fuente: Elaboración propia).

```
function conectar()
{
//se crea la conexion al SMBD
 $this->conn = NewADOConnection("ORACLE");
 $this->conn->Connect($this->nombre_bd,$this->user_bd,$this->passw_bd);

if (!$this->conn)
{
return false;
}
else
{ return true;
}
return $this->conn;
}
```

Anexo A7. Función de la clase Conexión (Fuente: Elaboración propia).

```
public function Insertar()
{ //Registra las correspondencias
//Obtiene los parámetros de entrada
for($i=0;$i<count($this->arrayelement_p);$i++){
if ($this->arrayelement_p[$i]!='accion' && $this->arrayelement_p[$i]!='accion_x'&& $this-
>arrayelement_p[$i]!='accion_y'){
if ($this->arrayelement_p[$i]=='nro_doc')
$this->correspondencia->set_nro_documento($_POST[$this->arrayelement_p[$i]]);
…
} //Fin IF
}//Fin For
$nro_doc = $this->correspondencia->get_nro_documento();
…
//Ejecución del Registro de la Correspondencia
$sql = "INSERT INTO correspondencia
(id_correspondencia,nro_documento,fecha_documento,fecha_registro,hora_registro,asunto,monto,fk_tipodoc
umento,fk_tipoorganismo,fk_estado,fk_materia,fk_estatus_correspondencia) VALUES
(SEQ_CORRESPONDENCIA.NEXTVAL,'$nro_doc',$fecha_documento,$fecha_registro,'$hora_registro','$as
unto',$monto,$fk_tipodocumento,$fk_tipoorganismo,$fk_estado,$fk_materia,fk_status)";
$resultado= $this->db->consultar($sql);
…
$elementos[3] = $nro_doc;
$elementos[4] = $asunto;
$elementos[5] = $tipo_doc;
$elementos[6] = $tipo_org;
...
return $elementos;
}
```

Anexo A8. Refactorización de la función Insertar del Modelo CorrespondenciaModel

(Fuente: Elaboración propia).

```
public function ConsultarEntradaRecepCoorUni(){
//Consulta las correspondencias de entrada
//Consulta el id_status de la correspondencia de entrada
$query2= "SELECT id_status FROM estatus_correspondencia where
UPPER(estatus_correspondencia)=UPPER('Entrada')";
$re = $this->db->consultar($query2);
$fk_status = $this->db->obtenerColumna($re,0,'id_status');
//Query de consulta de los valores requeridos de la correspondencia
$query = "SELECT id_correspondencia,nro_documento,asunto,
fk_materia FROM correspondencia fk_estatus_correspondencia=$fk_status";
//Extrae los valores de los parámetros de búsqueda para la correspondencia
for($i=0;$i<count($this->arrayelement_p);$i++){
if ($this->arrayelement_p[$i]!='accion_x' && $this->arrayelement_p[$i]!='accion_y'
&& $this->arrayelement_p[$i]!='accion' && $this->arrayelement_p[$i]!='consultar'
&& $this->arrayelement_p[$i]!='limpiar') {
if ($this->arrayelement_p[$i]=="nro_documento" && $_POST[$this->arrayelement_p[$i]]!=""){
$query = $query." AND nro_documento = '".$_POST[$this->arrayelement_p[$i]]."'";
$this->correspondencia->set_nro_documento($_POST[$this->arrayelement_p[$i]]);
}
…
}//Fin IF
}//FIN For
//Realiza la consulta de los parámetros de búsqueda para la correspondencia
$query = $query." ORDER BY fk_materia";
$resultado = $this->db->consultar($query);
$total = $this->db->numFilas($resultado);
$i=0;
$j=1;
while ($i<$total) { //Extrae los valores y los deja en un arreglo
…
$elementos[$j+1] = $nro_documento; //obtiene nro_documento
$elementos[$j+2] = $asunto; //obtiene asunto
…
}
return $elementos;
}
```

Anexo A9. Refactorización de la función ConsultarEntradaRecepCoor del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function ConsultarCorresSalidaUni(){
//Consulta la correspondencia de salida
//Se construye el query de consulta
$query = "SELECT id_correspondencia,nro_documento,asunto,fk_materia FROM correspondencia WHERE 1=1";
//Se obtienen los datos de entrada
for($i=0;$i<count($this->arrayelement_p);$i++){
if ($this->arrayelement_p[$i]!='accion_x' && $this->arrayelement_p[$i]!='accion_y'
&& $this->arrayelement_p[$i]!='accion' && $this->arrayelement_p[$i]!='consultar'
&& $this->arrayelement_p[$i]!='limpiar') {
if ($this->arrayelement_p[$i]=="gerencia_destino" && $_POST[$this->arrayelement_p[$i]]!="NADA"){
$query2="Select id_gerencia from gerencia where UPPER(nombre) = UPPER('".$_POST[$this->arrayelement_p[$i]]."')";
$r = $this->db->consultar($query2);
$fk_gerencia_destino = $this->db->obtenerColumna($r,0,"id_gerencia");
$query = $query." AND fk_gerencia_destino = '".$fk_gerencia_destino."'";
$this->correspondencia->set_fk_gerencia_destino($fk_gerencia_destino);
}
...
} //FIN DE IF
}//FIN DE FOR
...
//ejecuta la consulta
$query = $query." ORDER BY id_correspondencia";
$resultado = $this->db->consultar($query);
...
//Se obtienen los valores y se almacenan en un arreglo
while ($i<$total) {
...
$elementos[$j+1] = $nro_documento; //obtiene nro_documento
$elementos[$j+2] = $asunto; //obtiene asunto
...
}
return $elementos;
}
```

Anexo A10. Código de la función ConsultarCorresSalidaUni del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function Consultar_Detalle_CorresSalUni(){
//Consulta el detalle de los campos de la correspondencia de salida
//Se obtienen los datos de entrada
for($i=0;$i<count($this->arrayelement_g);$i++){
if ($this->arrayelement_g[$i]=="id_correspondencia"){
$this->correspondencia->set_id_correspondencia($_GET[$this->arrayelement_g[$i]]);
$id_correspondencia = $this->correspondencia->get_id_correspondencia();
}
...
}//fin de for
//Consulta los campos de la correspondencia de salida
$query = "SELECT
id_correspondencia,direccion_imagen,nro_documento,fk_prioridad,responsable,fk_gerencia_destino,fk_estatus_cor
respondencia,fecha_documento,monto,fk_tipodocumento,fk_estado,fk_tipoorganismo,fk_materia,asunto,fk_expedie
nte FROM correspondencia WHERE  id_correspondencia = $id_correspondencia";
$resultado = $this->db->consultar($query);
//Se obtienen los campos
$this->correspondencia->set_direccion_imagen($this->db->obtenerColumna($resultado,0,"direccion_imagen"));
$this->correspondencia->set_id_correspondencia($this->db->obtenerColumna($resultado,0,'id_correspondencia'));
...
$_SESSION['id_correspondencia']=$id_correspondencia;
$_SESSION['direccion_imagen'] = $this->correspondencia->get_direccion_imagen();
...
}
```

Anexo A11. Código de Consultar_Detalle_CorresSalUni del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function Guardar_CorresSalidaUni(){
//Modifica los campos de la correspondencia de salida
//Obtiene los datos de entrada - GET
for($i=0;$i<count($this->arrayelement_g);$i++){
if ($this->arrayelement_g[$i]=="id_correspondencia"){
$this->correspondencia->set_id_correspondencia($_GET[$this->arrayelement_g[$i]]);
}
...
//Se obtienen los campos
$num_documento = $this->correspondencia->get_nro_documento();
...
//Si el estatus es Entrada
if ($status_s=="Entrada" || $status_s=="ENTRADA"){
$query = "SELECT id_status FROM estatus_correspondencia WHERE UPPER(estatus_correspondencia) = UPPER('Salida')";
$re= $this->db->consultar($query);
$this->correspondencia->set_fk_estatus_correspondencia($this->db->obtenerColumna($re,0,"id_status"));
$fk_status = $this->correspondencia->get_fk_estatus_correspondencia();
$_SESSION["status"] = "Salida":
$cambio_estatus=1;
}
...
//Actualiza la correspondencia
$sql = "UPDATE correspondencia SET
nro_documento='$num_documento',fecha_documento=$fecha,asunto='$asunto',monto=$monto,fk_tipodocumento=$fk_tipodocumento,fk
_tipoorganismo=$fk_tipoorganismo,fk_estado=$fk_estado,fk_materia=$fk_tipomateria,fk_estatus_correspondencia=$fk_status,fk_gerenc
ia_destino=$fk_gerencia_destino,instrucciones='$instruccio',fecha_registro_salida=$fecha_registro_salida,hora_registro_salida='$hora_r
egistro_salida',responsable='$responsable',fk_prioridad=$fk_prioridad  WHERE id_correspondencia=$id_correspondencia";
$resultado = $this->db->consultar($sql);
...
retun $elementos;
}
```

Anexo A12. Código de la función Guardar_CorresSalidaUni del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function Consultar_Detalle_CorresSalUni(){
//Consulta el detalle de los campos de la correspondencia de salida
//Se obtienen los datos de entrada
for($i=0;$i<count($this->arrayelement_g);$i++){
if ($this->arrayelement_g[$i]=="id_correspondencia"){
$this->correspondencia->set_id_correspondencia($_GET[$this->arrayelement_g[$i]]);
$id_correspondencia = $this->correspondencia->get_id_correspondencia();
}
…
}//fin de for
//Consulta los campos de la correspondencia de salida
$query = "SELECT
id_correspondencia,direccion_imagen,nro_documento,fk_prioridad,responsable,fk_gerencia_destino,fk_estatus_cor
respondencia,fecha_documento,monto,fk_tipodocumento,fk_estado,fk_tipoorganismo,fk_materia,asunto,fk_expedie
nte,fecha_respuesta,respuesta FROM correspondencia WHERE  id_correspondencia = $id_correspondencia";
$resultado = $this->db->consultar($query);
//Se obtienen los campos
$this->correspondencia->set_direccion_imagen($this->db->obtenerColumna($resultado,0,"direccion_imagen"));
$this->correspondencia->set_id_correspondencia($this->db->obtenerColumna($resultado,0,'id_correspondencia'));
…
$_SESSION['id_correspondencia']=$id_correspondencia;
$_SESSION['direccion_imagen'] = $this->correspondencia->get_direccion_imagen();
…
}
```

Anexo A13. Refactorización de Consultar_Detalle_CorresSalUni del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function ConsultarCorresSalidaGer(){
//Consulta las correspondencias de salida perteneciente a la gerencia
//Obtiene la gerencia
for($i=0;$i<count($this->arrayelement_s);$i++){
if ($this->arrayelement_s[$i]=='gerencia'){
$this->gerencia->set_nombre($_SESSION[$this->arrayelement_s[$i]]);
}
}
$gerencia = $this->gerencia->get_nombre();
...
//Consulta la correspondencia según la gerencia
$query = "SELECT id_correspondencia,nro_documento,asunto,fk_materia FROM correspondencia WHERE
fk_gerencia_destino = $id_gerencia";
//Obtiene los datos de entrada
for($i=0;$i<count($this->arrayelement_p);$i++){
if ($this->arrayelement_p[$i]!='accion_x' && $this->arrayelement_p[$i]!='accion_y'
&& $this->arrayelement_p[$i]!='accion' && $this->arrayelement_p[$i]!='consultar'
&& $this->arrayelement_p[$i]!='limpiar') {
if ($this->arrayelement_p[$i]=="nro_documento" && $_POST[$this->arrayelement_p[$i]]!=""){
$query = $query." AND nro_documento = '".$_POST[$this->arrayelement_p[$i]]."'";
$this->correspondencia->set_nro_documento($_POST[$this->arrayelement_p[$i]]);
}
...
}
}
...
//Selecciona los identificadores de estatus distintos de entrada
$q1="Select id_status from estatus_correspondencia where
UPPER(estatus_correspondencia)=UPPER('Salida')";
$r = $this->db->consultar($q1);
$est1=$this->db->obtenerColumna($r,0,"id_status");
...
$query = $query." AND (fk_estatus_correspondencia = '".$est1."' or fk_estatus_correspondencia = '".$est2."' or
fk_estatus_correspondencia = '".$est3."' or fk_estatus_correspondencia = '".$est4."')";
//Ejecuta la consulta
$query = $query." ORDER BY id_correspondencia";
$resultado = $this->db->consultar($query);
...
return $elementos;
}
```

Anexo A14. Código de la función ConsultarCorresSalidaGer del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function Consultar_Detalle_CorresSalGer(){
//Consulta el detalle de las correspondencia de salida de la gerencia
//obtiene datos de entrada
for($i=0;$i<count($this->arrayelement_g);$i++){
if ($this->arrayelement_g[$i]=="id_correspondencia"){
$this->correspondencia->set_id_correspondencia($_GET[$this->arrayelement_g[$i]]);
$id_correspondencia = $this->correspondencia->get_id_correspondencia();
}
}//fin de for
//Consulta los campos de la correspondencia de salida
$query = "SELECT
id_correspondencia,nro_documento,fecha_registro_salida,fk_prioridad,respuesta,responsable,fk_gerencia_de
stino,fk_estatus_correspondencia,hora_registro_salida,fecha_respuesta,fecha_documento,monto,fk_tipodocu
mento,fk_estado,fk_tipoorganismo,fk_materia,asunto,fk_expediente FROM correspondencia WHERE
id_correspondencia=$id_correspondencia";
$resultado = $this->db->consultar($query);
...
//Si el estatus es Salida
if ($estatus=='Salida' || $estatus=='SALIDA'){
$query = "SELECT id_status FROM estatus_correspondencia WHERE UPPER(estatus_correspondencia) =
UPPER('EsperarRes')";
$re= $this->db->consultar($query);
$this->correspondencia->set_fk_estatus_correspondencia($this->db->obtenerColumna($re,0,"id_status"));
$fk_status = $this->correspondencia->get_fk_estatus_correspondencia();
$q="UPDATE correspondencia SET fk_estatus_correspondencia=$fk_status";
$r=$this->db->consultar($q);
$estatus = 'EsperarRes';
}
...
}
```

Anexo A15. Código de Consultar_Detalle_CorresSalGer del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function Guardar_CorresSalidaGer(){
//Modifica que el campo respuesta de la correspondencia de la gerencia
//Se obtienen los datos de entrada - GET
for($i=0;$i<count($this->arrayelement_g);$i++){
if ($this->arrayelement_g[$i]=="id_correspondencia"){
$this->correspondencia->set_id_correspondencia($_GET[$this->arrayelement_g[$i]]);
$id_correspondencia=$this->correspondencia->get_id_correspondencia();
}
}
...
//Si el estatus es EsperarRes pasa a PreCerrado
if (($status_s=='EsperarRes' || $status_s=='ESPERARRES') && ($respuesta_s!=$respuesta)){
$query = "SELECT id_status FROM estatus_correspondencia WHERE
UPPER(estatus_correspondencia) = UPPER('PreCerrado')";
$re= $this->db->consultar($query);
$this->correspondencia->set_fk_estatus_correspondencia($this->db-
>obtenerColumna($re,0,"id_status"));
$fk_status = $this->correspondencia->get_fk_estatus_correspondencia();
$esperar=1;
}
...
//Actualiza el estatus de la correspondencia
..
$sql = "UPDATE correspondencia SET
fk_estatus_correspondencia=$fk_status,respuesta='$respues',fecha_respuesta=$fecha_respu
e  WHERE id_correspondencia=$id_correspondencia";
...
return $elementos;
}
```

Anexo A16. Código de la función Guardar_CorresSalidaGer del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function Consultar_Detalle_CorresSalPresi(){
//Consulta los campos de la correspondencia de salida
//Obtiene los datos de entrada
for($i=0;$i<count($this->arrayelement_g);$i++){
if ($this->arrayelement_g[$i]=="id_correspondencia"){
$this->correspondencia->set_id_correspondencia($_GET[$this->arrayelement_g[$i]]);
$id_correspondencia = $this->correspondencia->get_id_correspondencia();
}
}//fin de for
…
//Consulta los campos de la correspondencia
$query = "SELECT
id_correspondencia,direccion_imagen,nro_documento,fecha_registro_salida,fk_prioridad,respuesta,responsable,fk_g
erencia_destino,fk_estatus_correspondencia,hora_registro_salida,fecha_respuesta,fecha_documento,monto,fk_tipod
ocumento,fk_estado,fk_tipoorganismo,fk_materia,asunto,fk_expediente FROM correspondencia WHERE
id_correspondencia = $id_correspondencia";
$resultado = $this->db->consultar($query);
$this->correspondencia->set_id_correspondencia($this->db->obtenerColumna($resultado,0,'id_correspondencia'));
…
//Cambia el estatus si todos los campos estan completos
if ($_SESSION["status"]=="PreCerrado" && $_SESSION['estado']!="" &&
$_SESSION['tipoorganismo']!="" && $_SESSION['tipodocumento']!="" &&
$_SESSION['num_documento']!="" && $_SESSION['fecha_documento']!="" &&
$_SESSION['monto']!="" && $_SESSION['asunto']!="" &&
$_SESSION['descripcion']!="" &&
$_SESSION['fecha_registro_salida']!="" && $_SESSION['fecha_respuesta']!="" &&
$_SESSION['gerencia_destino']!="" && $_SESSION['prioridad']!="" && $_SESSION['respuesta']!="" &&
$_SESSION['responsable']!="" && $_SESSION['instrucciones']!=""){
$query="select id_status from estatus_correspondencia where
UPPER(estatus_correspondencia)=UPPER('Cerrado')";
$r=$this->db->consultar($query);
$status_fk=$this->db->obtenerColumna($r,0,"id_status");
$sql1="update correspondencia set fk_estatus_correspondencia=$status_fk where
id_correspondencia=$id_correspondencia";
$r=$this->db->consultar($sql1);
}
}
```

Anexo A17. Código de Consultar_Detalle_CorresSalPresi del Modelo CorrespondenciaModel (Fuente: Elaboración propia).

```
public function Guardar()
{//Modifica los campos del evento
//Se obtienen los parámetros de entrada
$fecha_fin = $this->recordatorio->get_fecha_fin();
$id_usuario = $this->usuario->get_id_usuario();
$evento = $this->recordatorio->get_evento();
$id_recordatorio = $this->recordatorio->get_id_recordatorio();
$tipo_evento = $this->tipo_evento;
...
//Alarma
$fecha2 = mktime(0,0,0,$fechaSeparada[1],$fechaSeparada[0]-$this->dias_alarma,$fechaSeparada[2]);
$fecha_alarma = date("d/m/y", $fecha2);
$fecha_alarma = "TO_DATE('".$fecha_alarma."','DD-MM-YY')";
...
if ($tipo_evento=="EVENTO DE PUNTO DE CUENTA"){
$fk_ptocuenta=$this->identificador;
...
$fk_correspondencia='null';
$sql = "UPDATE recordatorio SET
evento='$evento',fecha_alarma=$fecha_alarma,fk_ptocuenta=$fk_ptocuenta,fk_correspondencia=$fk_correspondenc
ia WHERE fk_usuario=$id_usuario AND id_recordatorio=$id_recordatorio";
}
else{
    if ($tipo_evento=="EVENTO DE CORRESPONDENCIA"){
       $fk_correspondencia=$this->identificador;
       $fk_ptocuenta='null';
       ...
      $sql = "UPDATE recordatorio SET
evento='$evento',fecha_alarma=$fecha_alarma,fk_correspondencia=$fk_correspondencia,fk_ptocuenta=$fk_ptocuen
ta WHERE fk_usuario=$id_usuario AND id_recordatorio=$id_recordatorio";
    }
    else{
            $fk_ptocuenta='null';
             $fk_correspondencia='null';
            $sql = "UPDATE recordatorio SET
evento='$evento',fecha_alarma=$fecha_alarma,fk_correspondencia=$fk_correspondencia,fk_ptocuenta=$fk_ptocuen
ta WHERE fk_usuario=$id_usuario AND id_recordatorio=$id_recordatorio";
        }
    }
//Actualiza el evento
$result = $this->db->consultar($sql);
}
```

Anexo A18. Código de Guardar del Modelo RecordatorioModel (Fuente: Elaboración propia).

```
$db = Singleton::singleton(); //se trae la instancia
$id_usuario = $_SESSION["id_usuario"]; //Obtiene identificador del usuario
$fecha_actual = date("d")."-".date("m")."-".date("Y"); //Obtiene la Fecha Actual
//Selecciona los eventos con fecha comprendida entre fecha alarma y fecha fin pertenecientes al usuario
$query="SELECT evento,fecha_fin,fecha_inicio,fk_ptocuenta,fk_correspondencia FROM recordatorio WHERE
(TO_DATE('$fecha_actual','DD-MM-YY') BETWEEN fecha_alarma AND fecha_fin) AND fk_usuario =
'".$_SESSION["id_usuario"]."'";
//Ejecuta el query
$resultado = $db->consultar($query);
$total = $db->numFilas($resultado);
```

Anexo A19. Código de la alarma mostrada en el home de la Aplicación (Fuente: Elaboración propia).

```
public function BuscarActividad()
{ //Busca las actividades

//Obtiene el Rol
$rol = $this->rol->get_nombre();
$query = "SELECT id_rol FROM rol WHERE UPPER(nombre) = UPPER('$rol')";
$resultado = $this->db->consultar($query);
$id_rol = $this->db->obtenerColumna($resultado,0,"id_rol");
//Obtiene las actividades del rol
$query = "select fk_actividad from rol_actividad where fk_rol = $id_rol";
$resultado = $this->db->consultar($query);
$total_act = $this->db->numFilas($resultado);
for($i=0;$i<$total_act;$i++)
{//Obtiene los identificadores de las actividades
$ids_actividades[$i] = $this->db->obtenerColumna($resultado,$i,"fk_actividad");
}
for($i=0;$i<count($ids_actividades);$i++)
{//Obtiene los campos asociados a las actividades
$result = $this->db->consultar("SELECT actividad,direccion_actividad,fk_modulo FROM actividad WHERE
id_actividad=$ids_actividades[$i]");
//Extrae las actividades, direcciones y módulos y los coloca en arreglos
$actividades[$i] = $this->db->obtenerColumna($result,$i,"actividad");
$direcciones[$i] = $this->db->obtenerColumna($result,$i,"direccion_actividad");
$fk_modulo       = $this->db->obtenerColumna($result,$i,"fk_modulo");
$result          = $this->db->consultar("SELECT nombre FROM modulo WHERE id_modulo = $fk_modulo");
$modulos[$i]     = $this->db->obtenerColumna($result,$i,"nombre");
}
//Ordena las actividades según los módulos
$cont = 0;
for($i=0;$i<count($actividades);$i++)
{
if($modulos[$i]!="tachado")
{
$aux = $modulos[$i];
$modulos_ordenados[$cont] = $modulos[$i];
$actividades_ordenadas[$cont] = $actividades[$i];
$direcciones_ordenadas[$cont] = $direcciones[$i];
$cont++;
$modulos[$i] = "tachado";
for($j=$i+1;$j<count($actividades);$j++)
{
if($aux==$modulos[$j])
{
$modulos_ordenados[$cont] = $modulos[$j];
$modulos[$j] = "tachado";
$actividades_ordenadas[$cont] = $actividades[$j];
$direcciones_ordenadas[$cont] = $direcciones[$j];
$cont++;
}
}
}
}
for($i=0;$i<count($actividades);$i++)
{
$modulos[$i] = $modulos_ordenados[$i];
$actividades[$i] = $actividades_ordenadas[$i];
$direcciones[$i] = $direcciones_ordenadas[$i];
}
}
```

Anexo A20. Código de BuscarActividad del Modelo ActividadModel (Fuente: Elaboración propia).

```
public function Eliminar(){
//Elimina una actividad del sistema
//Obtiene datos de entrada
$act_s = $this->actividad_s;
if ($act_s!="0"){
//Borra las actividades asociadas a los roles
$sql1 = "DELETE FROM rol_actividad WHERE fk_actividad = $act_s";
$resultado = $this->db->consultar($sql1);
//Borra la actividad
$sql2 = "DELETE FROM actividad WHERE id_actividad = $act_s";
$resultado = $this->db->consultar($sql2);
…
}
…
}
```

Anexo A21. Código de Eliminar del Modelo ActividadModel (Fuente: Elaboración propia).

```
<?php
$moduloAnterior = "";
for($i=0;$i<count($ids_actividades);$i++)
{  //Obtiene las actividades y los módulos
   $actividadActual = $actividades[$i];
$moduloActual = $modulos[$i];
if ($moduloActual != $moduloAnterior)
{
?> <!—Coloca el módulo-->
<tr>
<td width="24" bgcolor="#D76966">
 <p style="margin-top: 0; margin-bottom: 0;"><b>
 <font face="Verdana" size="1" color="#FFFFFF">
 <img src="images/banavih_15.gif" width="24" height="20" alt="" /><img src="../images/banavih_15.gif" width="24" height="20"
alt="" /></font></b></p>                                    </td>
 <?php if ($_SESSION['cambio']!=1) { ?><td width="122" background="images/banavih_17.gif"><?php } else { if
($_SESSION['cambio']==1) { ?><td width="122" background="../images/banavih_17.gif"><?php }} ?>
 <p style="margin-top: 0; margin-bottom: 0"><b>
 <font face="Verdana" size="1" color="#FFFFFF"><?php echo $moduloActual; ?></font></b></p>
</td>
 <?php if ($_SESSION['cambio']!=1) { ?><td width="9" background="images/arte01b1_21.gif"><?php } else { if
($_SESSION['cambio']==1) { ?><td width="9" background="../images/arte01b1_21.gif"><?php }} ?> </td>
 <td width="5"> </td>
 </tr>
<?php
}

$moduloAnterior = $moduloActual;
?>
<tr> <!—Coloca la actividad-->
<td width="24" bgcolor="#EAACAC">
<p style="margin-top: 0; margin-bottom: 0">
<img src="images/banavih_19.gif" width="24" height="18" alt="" /><img src="../images/banavih_19.gif" width="24" height="18" alt=""
/></p>
</td>
<td width="121" bgcolor="#EAACAC">
<p class="linkroll" style="margin-top: 0; margin-bottom: 0"><a href="<?php echo $direcciones[$i];?>"><?php echo $actividadActual;
?></a></p>
</td>
<?php if ($_SESSION['cambio']!=1) { ?><td width="9" background="images/arte01b1_21.gif"><?php } else { if
($_SESSION['cambio']==1) { ?><td width="9" background="../images/arte01b1_21.gif"><?php }} ?> </td>
<td width="5"> </td>
</tr>
<?php
}
?>
```

Anexo A22. Refactorización del Menú (Dinámico) de la Interfaz Principal (Fuente: Elaboración propia).