ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"SISTEMA DE HORAS EXTRAS PARA TELCONET CON MODELOS DE APLICACIÓN DE CAPAS MÚLTIPLES"

INFORME DE MATERIA DE GRADUACIÓN

Previo a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES
ESPECIALIZACIÓN SISTEMAS TECNOLÓGICOS

Presentado por:

Diana Catalina Tobar Lara.
Kenneth Joffre Jiménez Plúas.

GUAYAQUIL – ECUADOR

Año: 2012

AGRADECIMIENTO

Agradecemos a Dios por brindarnos las fuerzas necesarias para lograr nuestras metas. A nuestros padres por ser ese pilar importante de apoyo y fortaleza en nuestras vidas. Al Ing. Matteo Silva y a la Ing. Jessica Suárez por brindarnos su ayuda y conocimientos para el desarrollo de nuestra tesis. A nuestros profesores, especialmente a la Ing. Carmen Vaca, por otorgarnos los conocimientos necesarios para nuestra vida profesional. A la Espol por darnos las herramientas necesarias para formarnos como auténticos profesionales y a todas las personas que de alguna manera nos ayudaron a culminar esta etapa de nuestras vidas.

Catalina Tobar Lara.
Kenneth Jiménez Plúas.

DEDICATORIA

Este trabajo está dedicado a Dios por brindarme las energías y fuerzas necesarias para poder culminar con éxito mi vida universitaria. A mi familia especialmente a mis padres por la tolerancia, amor, fortaleza y apoyo incondicional brindado ante cualquier situación. A mi novio Miguel Gallegos por siempre contar con su amor, compañía, ayuda y protección en todo lo que necesite, recordándome que estará al lado mío apoyándome y dándome fuerzas para lograr mis objetivos. A mis amigos, mi segunda familia, por brindarme su compañía y ayuda desinteresada en los buenos y malos momentos de mi vida, dejándome en claro que siempre podré contar con ellos. Finalmente, pero no menos importante a mi compañero de tesis y mejor amigo Kenneth Jiménez, por el sacrificio y esfuerzo brindado en este proyecto y durante toda nuestra vida universitaria, brindándome sus consejos, conocimientos y ayuda en toda situación académica, profesional y personal.

Diana Catalina Tobar Lara.

DEDICATORIA

Quiero Dedicar este trabajo primeramente a Dios, por darme las fuerzas y sabiduría necesaria para cumplir con esta etapa de mi vida muy importante que es mi formación como profesional. A mis padres, por ser ese pilar en el cual cuento con ese apoyo incondicional, brindándome en todo momento sus sabios consejos para hacer de mí una persona de bien, con valores y ética, impulsándome al éxito en mi vida. A mis amigos, quienes se han convertido en mi familia. Hemos compartido muchos momentos, buenos y malos, pero sobre todo me han ayudado en lo que ha estado a su alcance. Los Llevare siempre en mi mente y corazón. A Catalina mi gran amiga casi hermana, por ayudarme mucho a lo que en este presente soy, brindándome su apoyo y consejos en todo momento tanto en la parte humana como académica. Y finalmente a mi amada esposa Ruth, siendo ella mi fuente de inspiración para todas las cosas que hago. Su Amor e infinito apoyo me dan muchas fuerzas para ir creciendo día a día y cumplir mis metas.

Kenneth Joffre Jiménez Plúas.

TRIBUNAL DE SUSTENTACIÓN

Ing. Matteo Silva

PROFESOR DE LA MATERIA DE GRADUACIÓN

Msc. Vanessa Cedeño

PROFESOR DELEGADO POR LA UNIDAD ACADÉMICA

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este informe, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

Diana Catalina Tobar Lara

Kenneth Joffre Jiménez Plúas

RESUMEN

Este documento muestra el desarrollo del SISTEMA DE HORAS EXTRAS, a través del desarrollo dirigido por modelos, haciendo uso de la herramienta WebRatio.

El objetivo del proyecto es utilizar la herramienta MDD WebRatio, haciendo uso de los lenguajes BPM y WebML para la implementación del sistema de Horas Extras, logrando realizar una aplicación web de calidad y fácil mantenimiento. De esta manera, hemos reducido el esfuerzo y tiempo de desarrollo, debido a la transformación automática de modelos que genera código libre de errores.

La tesis está separada en 3 capítulos donde se tratarán diversos aspectos de la aplicación como el problema sobre el cual parte la aplicación realizada, las herramientas a utilizar, su diseño, desarrollo e implementación.

En el Capítulo 1 se presenta el planteamiento del problema que la presente tesis busca resolver, asimismo da a conocer la solución al problema señalado, además de indicar los objetivos a cumplir y el alcance de la misma.

En el Capítulo 2 se exponen los fundamentos teóricos de la arquitectura dirigida por modelos, indicando los lenguajes y las herramientas a utilizar.

En el Capítulo 3 se explica el diseño, desarrollo e implementación que se realizó para el proceso de horas extras.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ABREVIATURAS

CASE: Computer Aided Software Engineering - Ingeniería de Software Asistida por Computadora.

RRHH: Recursos Humanos.

MDD: Model Driven Development - Desarrollo Dirigido a Modelos.

MDE: Model Driven Engineering - Ingeniería Dirigida para Modelos.

OMG: Object Management Group - Grupo de Gestión de Objetos.

MDA: Model Driven Architecture - Arquitectura Dirigida por Modelos.

IBM: International Business Machines.

HP: Hewlett-Packard.

CIM: Computation Independent Model - Modelo Independiente de la Computación.

PIM: Platform Independent Model - Modelo Independiente de la Plataforma.

PSM: Platform Specific Model - Modelo Específico de la Plataforma.

BPM: Business Process Modeling - Modelado de Procesos de Negocio.

BPMN: Business Process Modeling Notation - Notación para el Modelado de Procesos de Negocio.

WebML: Web Modeling Language - Lenguaje de Modelado Web.

J2EE: Java 2 Platform, Enterprise Edition.

JSTL: JavaServer Pages Standard Tag Library.

JSP: JavaServer Pages.

GTR: Gerencia Técnica Regional.

HE: Horas Extras.

E-R: Entidad-Relación.

HTML: HyperText Markup Language - Lenguaje de Marcado de Hipertexto.

BO: Business Object - Objeto de Negocio

INTRODUCCIÓN

La industria de desarrollo de software busca constantemente mejorar el desempeño, reduciendo el tiempo de desarrollo, con el fin de maximizar las ganancias.

Es por esto que se debe buscar la programación a nivel de Ingeniería de Modelos y requisitos, evitando desgastarse en diseño, codificación y pruebas.

En el desarrollo dirigido por modelos se consigue separar la especificación de la estructura y funcionalidad del sistema de la implementación, logrando abstracción en el desarrollo de software y otorgando, de esta manera, una mayor importancia a los modelos.

De este modo el Sistema de Horas Extras fue desarrollado utilizando la herramienta CASE WebRatio la cual se enfoca en diseñar, construir y mantener aplicaciones web completas, concentrando los recursos en los requerimientos del negocio, la creación del modelo, en lugar de los detalles de la implementación técnica, ya que genera de forma automática la aplicación web.

CAPITULO 1

1. ANÁLISIS DEL PROBLEMA

1.1 ADMINISTRACIÓN ACTUAL DEL REGISTRO Y GENERACIÓN DE LAS HORAS EXTRAS EN TELCONET

En la actualidad en Telconet existe un sistema de Actividades donde se registra las diferentes tareas programadas de los empleados (ya sea como responsable o como colaborador).

De este registro parte el reporte de Horas extras (excel) que el empleado cada mes debe realizar y enviar a sus respectivos jefes departamentales, si y solo si tiene registro de tareas programadas en el sistema mencionado y tareas extras enviadas por su jefe.

Cada Jefe departamental verifica que cada uno de los reporte enviados por los empleados de su área este correcto. Una vez revisados todos los reportes los envía a Gerencia técnica Regional, que vuelve a verificar cada uno de los reportes y los envía a Recursos Humanos para que calcule el número de horas extras totales de cada empleado y pasarlo a Nómina. La Figura 1.1 ilustra la administración actual del Sistema de Horas Extras

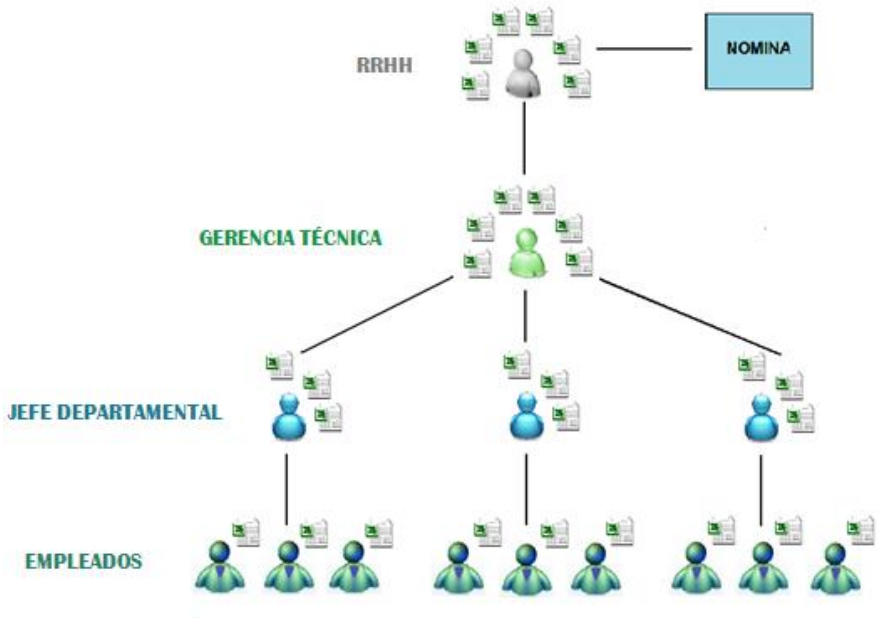

Figura 1.1 Administración Actual del Sistema de Horas Extras

1.2 DESCRIPCIÓN DEL PROBLEMA

Debido a que el Excel es manipulable, el control de las horas extras para los jefes departamentales y para la gerencia técnica regional se complica al momento de verificar y autorizar los reportes emitidos por

los empleados ya que suelen tener inconsistencia, y el tiempo invertido en estas acciones es directamente proporcional al número de empleados por departamento convirtiéndose en una tarea que consume tiempo muy valioso para las demás actividades de los jefes departamentales y Gerencia técnica. Generando también un descontrol de gastos y la falta de optimización de los recursos de la empresa.

1.3 JUSTIFICACIÓN

Un sistema de Horas Extras en Telconet constituye una herramienta de gran importancia en la empresa, ya que en lugar de que el proceso se realice manualmente como se hace en la actualidad, será un proceso automático guiado por humanos, que ayudará a mejorar el tiempo de respuesta de los procesos realizados para la obtención de horas extras de cada empleado.

El presente trabajo nace como una iniciativa que busca controlar los recursos de la empresa, permitiendo obtener un cálculo de horas extras consistente con el que se podrá ajustar costos, controlar los gastos de la empresa, y optimizar el tiempo que los jefes Departamentales, Gerencia Técnica y Recursos Humanos invierten en el proceso anteriormente mencionado.

1.4 OBJETIVOS

- Diseñar e implementar una aplicación web que automatice el proceso de registro de Horas Extras de los empleados de Telconet.
- Utilizar una herramienta MDD que mediante la transformación automática de modelos genere código libre de errores.
- Reducir el esfuerzo, tiempo de desarrollo, y mantenimiento de la aplicación web mencionada.
- Implementar la interfaz gráfica del sistema de horas extras.
- Implementar un módulo de reportes destinado al análisis estadístico de los datos.

1.5 ALCANCE

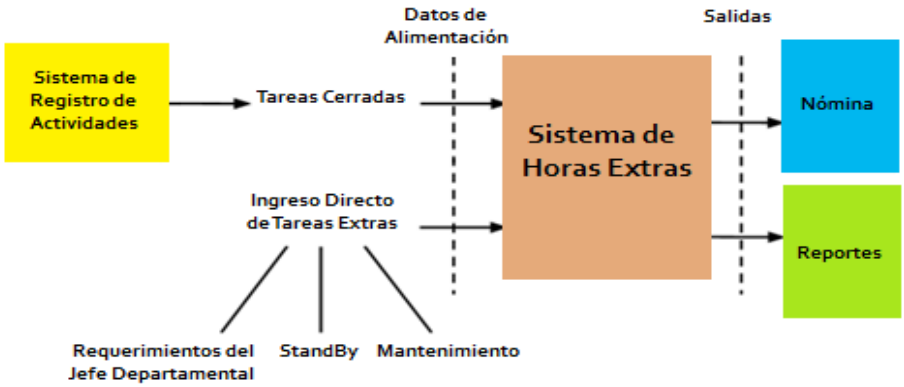

Figura 1.2 Alcance del Sistema de Horas Extras

Como se indica en la Figura 1.2, en este trabajo se realizará la implementación del sistema de Horas Extras el cual permitirá la autenticación de cada usuario. Este sistema tendrá como fuentes de ingreso las tareas del sistema de actividades programadas, así como el ingreso directo de Tareas Extras por parte del usuario.

El sistema le brindará al usuario el mapeo y seguimiento del estado de sus horas extras, así como niveles de autorización tanto para los jefes departamentales y Gerencia Técnica. Estas autorizaciones estarán sujetas a un tiempo límite.

Este sistema realizará el cálculo y clasificación automática del tipo de las horas extras basadas en las jornadas laborales de los empleados y les indicará el acumulado de las mismas.

El sistema contendrá un módulo de reportes, el cual permitirá elaborar reportes de las horas extras por empleado, por área y total. Estos a la vez podrán ser exportados en formato Excel. Adicionalmente contendrá un módulo de gráficos estadísticos, con el fin de analizar las tendencias de las Horas Extras por empleado.

Finalmente el sistema cada mes elaborará el resumen final de horas extras de los empleados y los pasará a nómina.

CAPITULO 2

2. FUNDAMENTOS TEÓRICOS DE LA ARQUITECTURA DIRIGIDA POR MODELOS

2.1 INTRODUCCIÓN AL MDE

En el proceso de desarrollo de Software, debido al rápido crecimiento tecnológico de las plataformas, se vuelve primordial mantener el modelo conceptual del negocio, ya que este es independiente de la plataforma o la tecnología en la cual se implemente.

En la Ingeniería Dirigida por Modelos (MDE), cualquier característica del sistema debe ser modelada de acuerdo al dominio al que

pertenecen, utilizando los motores de transformación, que transforman un modelo a otro, basándose en las reglas de transformación entre meta modelos, generando finalmente código fuente.

Hoy en día el uso de modelos es primordial para el desarrollo de sistemas. La principal característica del desarrollo dirigido por modelos (MDD) es suplir, como herramienta principal en el desarrollo de software, al código de lenguajes de programación, por modelos.

De esta manera, los modelos se convierten en una parte importante, al momento de crear, analizar y manipular sistemas mediante diversos lenguajes y herramientas [1].

2.2 OMG

Object Management Group (Grupo de Gestión de Objetos) es una organización sin fines de lucro encargada de determinar y mantener estándares para aplicaciones que utilicen tecnologías orientadas a objetos. Este grupo está formado por distintas corporaciones como IBM, Apple, Sun Microsystems, HP, entre otras [2].

OMG determinó el framework MDA como una arquitectura para el desarrollo de software, en la cual los modelos son la base del

proceso de desarrollo. Este nuevo paradigma se ha denominado Ingeniería de modelos o Desarrollo basado en modelos.

2.3 MDA

MDA es el acrónimo de Model Driven Architecture (Arquitectura Dirigida por Modelos), es una arquitectura que integra diferentes especificaciones y estándares definidos por la OMG.

MDA separa la lógica del sistema, de los detalles con que el sistema será implementado, teniendo como objetivos la portabilidad, interoperabilidad y reusabilidad independientemente de la plataforma tecnológica a utilizar.

El paradigma MDA abarca el ciclo de un sistema software mediante el siguiente proceso de desarrollo:

Del levantamiento de requisitos se obtiene un Modelo Independiente de la Plataforma (PIM), transformándose en uno a más Modelos Específicos de la Plataforma (PSM), donde cada PSM se transforma en código fuente.

2.3.1 CLASES DE MODELOS

MDA intenta separar la especificación del proceso y la lógica del sistema, con los detalles de la plataforma en la cual el sistema

será implementado. Es por esto que propone tres modelos a utilizarse durante el proceso de ingeniería:

- Modelo Independiente de la Computación
- Modelo Independiente de la Plataforma
- Modelo Específico de Plataforma

2.3.1.1 DEFINICIONES

Modelo Independiente de la Computación (CIM): Un CIM modela los requerimientos que debe cumplir el sistema, independientemente de la plataforma tecnológica, describiendo como el sistema será usado. Un CIM es muy útil ya que facilita la comprensión del problema y proporciona una base para otros modelos.

Modelo Independiente de la Plataforma (PIM): Un PIM es un modelo que representa la lógica del sistema y sus interacciones, sin determinar una plataforma específica (MDE aplicado a BPM) . Un PIM se usa como base del proceso de desarrollo (siendo creado solamente por el desarrollador), además de ser fácil de entender por los usuarios del sistema facilitando la corrección del mismo.

Modelo Específico de Plataforma (PSM): El modelo PSM detalla la plataforma específica con la que el sistema se implementará. Éste se genera en base al PIM, por lo que combina la especificación del sistema hecha en el PIM, con los detalles con que el sistema será implementado en una plataforma determinada.

La herramienta mapping que brinda MDA, mediante transformaciones convierte un PIM a PSM.

2.3.1.2 PROCESO DE DESARROLLO

En el proceso de desarrollo de Software la mayoría de los desarrolladores utilizan el modelado para analizar el problema y proponer una solución. Actualmente las estrategias que se utilizan en la Ingeniería de Software para el desarrollo del mismo usan modelos, dejando en claro que el enfoque actual está dirigido por modelos.

MDD es el acrónimo de Model Driven Development (Desarrollo Dirigido por Modelos) un paradigma de la Ingeniera de Software en el que el desarrollo de software se basa en los modelos y la transformación entre ellos, obteniendo la generación automática de

código a partir de modelos gráficos o especificaciones textuales.

El enfoque MDD garantiza la consistencia, productividad, portabilidad, mantenimiento del sistema, tal como se muestra en la Figura 2.1, ya que la transformación de un modelo a otro es automática así como la generación de código, es decir lo realiza la herramienta utilizada y no el desarrollador como se da en el desarrollo tradicional [3].

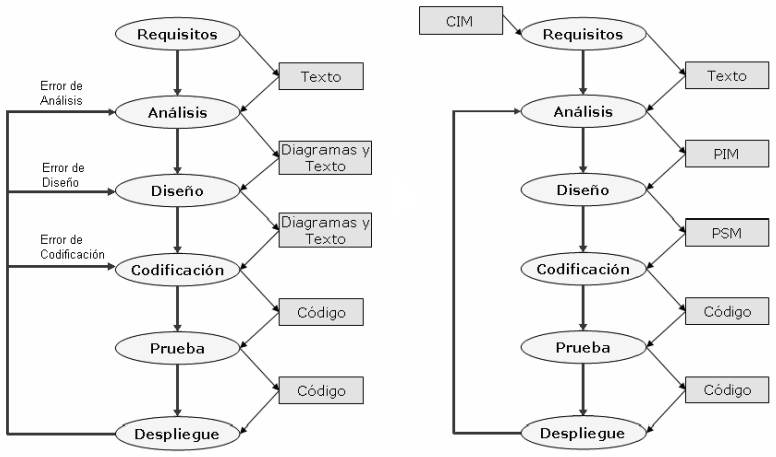

Figura 2.1 Proceso Tradicional y Proceso MDD

Fuente: http://revista.eia.edu.co/articulos8/Art.10.pdf

2.3.2 LENGUAJES

Los lenguajes que se utilizarán para realizar el modelado de nuestro proceso haciendo uso del desarrollo dirigido por

modelos son BPMN y WebML, los cuales serán explicados a continuación:

2.3.2.1 BPMN 2.0

Business Process Modeling Notation (Notación para el Modelado de Procesos de Negocio) es un estándar internacional definido por la OMG para modelar procesos.

BPMN, el cual está basado en un diagrama de flujo para definir un modelo de negocio, consiste en un conjunto de elementos gráficos, los cuales se muestran en la Figura 2.2 y detallados en [4], que facilitan la comprensión de un proceso, indicando los eventos, las actividades, los participantes y los resultados del flujo de proceso, así como las decisiones del negocio y las ramificaciones del flujo.

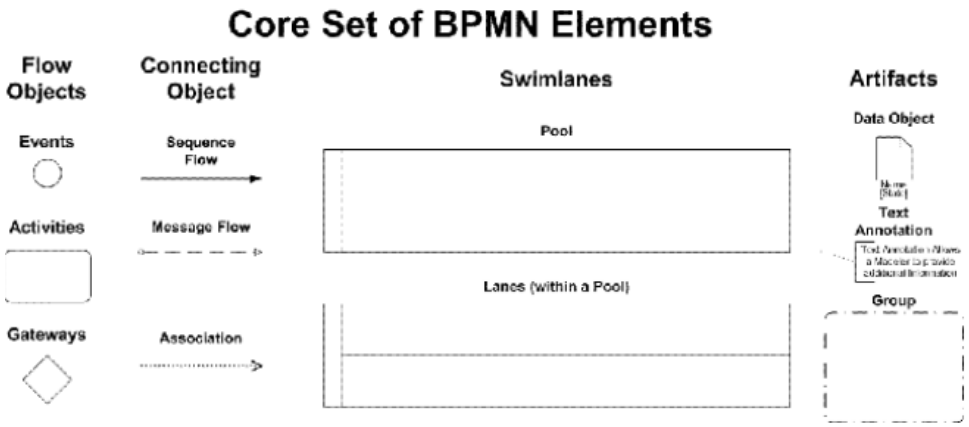

Figura 2.2 Notación BPMN

Fuente: http://wiki.webratio.com/index.php/Getting_started_with_WebRatio_6_BPM

De esta manera, integra la secuencia del proceso y la información q se envía entre los participantes, permitiendo que el flujo y el proceso sean legibles y fáciles de entender para todos los involucrados en el negocio (usuarios, desarrolladores, etc).

2.3.2.2 WEBML

WebML(Web Modeling Language) es un lenguaje de modelado para aplicaciones web, que permite mediante especificaciones graficas formales, el diseño completo de la aplicación web en base a 4 modelos distintos, tal como se muestra en la Figura 2.3, los cuales explicaremos a continuación:

- Modelo estructural: El proceso de desarrollo de WebML inicia con el modelado conceptual en el que se representa la estructura del sistema, a través del levantamiento de requisitos de la aplicación, definiendo el diseño de datos mediante el modelo Entidad-Relación o diagramas de clase UML.
- Modelo del hipertexto: Se realiza el modelo gráfico del proceso, definiendo las vistas del sitio, utilizando la notación propia de WebML. La estructura del hipertexto se realiza mediante el modelo de composición, especificando que páginas utilizaremos en nuestra aplicación web con su contenido, y el modelo de navegación, que como su nombre lo indica define la navegación a través de las páginas.
- Modelo de presentación: Se realiza la interfaz gráfica de la aplicación, el estilo y apariencia de cada página.
- Modelo de personalización: Se definen los permisos a cada grupo y usuario de la aplicación a los distintos módulos del sistema [5].

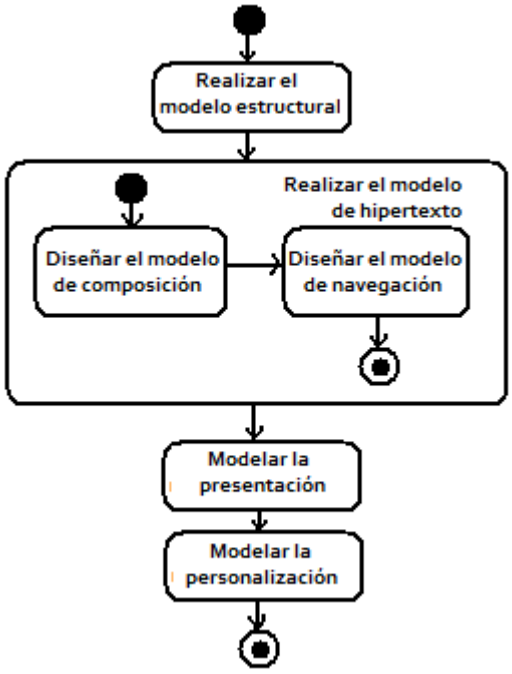

Figura 2.3 Proceso de desarrollo de WebML

Fuente: http://es.scribd.com/doc/63630638/27/Figura-2-10-Proceso-de-desarrollo-de-WebML

2.4 HERRAMIENTAS DE DESARROLLO MDD

WebRatio es una herramienta MDD que nos permite diseñar, construir y mantener aplicaciones web completas, enfocando los recursos en el análisis de los requisitos, la creación del modelo, con el fin de obtener la generación automática de la aplicación web.

Las aplicaciones web construidas con WebRatio cumplen con el estándar Java / J2EE, permitiendo así desarrollar aplicaciones web en el lenguaje de programación Java utilizando librerías estándar

(Hibernate, Struts, JSTL, JSP y Java servlet), y ejecutarlas en cualquier servidor de aplicaciones Java (Tomcat, JBoss, etc). Adicionalmente, gracias a que WebRatio utiliza las librerías Hibernate, la aplicación web puede conectarse a cualquier gestor de base de datos soportada por el mismo framework (MySQL, Oracle, PostgreSQL, etc.).

WebRatio utiliza el entorno de desarrollo dirigido por modelos, de esta manera, los requisitos del negocio son reflejados mediante la construcción de modelos de alto nivel de abstracción e independientes de la plataforma utilizada (PIM): BPMN y WebML, produciendo que el motor de generación transforme estos modelos automáticamente al código Java de la aplicación, permitiendo utilizar las reglas de generación predefinidas, o agregar nuevos componentes según las necesidades, además de definir estilos de presentación obteniendo como resultado una aplicación Java estándar a medida y sin componentes propietarios [7].

CAPITULO 3

3. ANÁLISIS DISEÑO Y DESARROLLO DEL SISTEMA

3.1 REQUERIMIENTOS DEL NEGOCIO

El proceso de desarrollo del Sistema de Horas Extras inicia con la recolección de requisitos, donde se describen los requerimientos de la aplicación tales como, los usuarios que la utilizarán, el contenido a presentar, guías de estilo, personalización y validaciones requeridas, etc.

3.1.1 REQUERIMIENTOS FUNCIONALES

En el sistema de horas extras de Telconet se registrarán las horas extras de todos los empleados del área técnica, las cuales surgen de las Actividades antes mencionadas:

- Tareas programadas (Horario no laboral)
- Tareas extras(standby, peticiones directas del jefe de área, mantenimiento, etc)

Las Horas Extras se clasifican en dos tipos dependiendo de la jornada laboral del empleado:

- Extraordinarias
- Normales

El sistema permitirá registrar las horas extras por la categoría de Tareas Extras (estas tareas no existen en el sistema de actividades de Telconet y por este motivo se registrarán directamente en el sistema).

El sistema le permitirá al responsable de la actividad durante un tiempo de 48 horas el poder realizar un ingreso de hora tentativa de inicio y fin de una tarea. El tiempo tentativo registrado afectará a todos los involucrados en la actividad, cabe recalcar que esto no es obligatorio. Si el empleado responsable de la actividad no registra este tiempo tentativo pasadas las 48 horas, el sistema calculará las horas extras y enviará el reporte de la actividad al jefe departamental para que este lo revise, valide y autorice.

El jefe departamental podrá ver gráficos estadísticos de las tendencias de las Horas Extras de su área y por empleado

Las horas extras autorizadas por el jefe de área se remitirán a la gerencia técnica para su autorización. Una vez autorizadas por la gerencia técnica el acumulado de las mismas estará listo para enviarse a nomina (sistema financiero de la empresa).

Si Gerencia Técnica está en desacuerdo con algún registro de hora extra podrá enviarlo a revisión con el jefe departamental correspondiente.

Tanto los jefes departamentales como la gerencia técnica podrán modificar los tiempos tentativos ingresados en las actividades.

El sistema siempre indicará el totalizado de las horas normales y extraordinarias que tiene el usuario.

Tanto jefe departamental como Gerencia Técnica tendrán un tiempo establecido para autorizar las horas extras. Si el tiempo establecido del jefe departamental se vence, las horas extras serán escaladas a Gerencia Técnica. Por otra parte, si el tiempo establecido para Gerencia técnica se vence, estas horas serán alertadas a GTR como pendientes de aprobación.

3.1.2 GRUPOS DE USUARIOS

Los grupos de Usuarios participantes de la aplicación del sistema de horas extras, que poseen acceso a la información y al hipertexto, son los detallados a continuación:

Rol	Descripción
Empleado	Los usuarios pertenecientes a este grupo pueden ingresar una nueva actividad, escogiendo los responsables y colaboradores de la misma, además de ingresar los tiempos estimados de cada hora extra realizada.
Jefe Departamental	Los usuarios pertenecientes a este grupo están encargados de la revisión y aprobación de las horas extras de cada empleado perteneciente a su departamento.
Gerencia Técnica	Los usuarios pertenecientes a este grupo están encargados de la revisión y aprobación de las horas extras aprobadas por el Jefe departamental.

3.1.2 REQUERIMIENTOS NO FUNCIONALES

Flexibilidad

El sistema de horas extras debe ser multiempresa, es decir, debe poder adaptarse a cualquier empresa del Grupo Telconet que requiera el cálculo de las horas extras. Adicionalmente, el sistema de horas extras debe funcionar correctamente en cualquier navegador usado por los usuarios.

Confiabilidad

El sistema de horas extras debe proporcionar confianza al usuario durante todo el proceso, calculando correctamente el número de horas extras de cada empleado, y ejecutando con eficiencia los niveles de autorización de las horas extras de cada empleado.

Usabilidad

El sistema de horas extras debe mostrar una interfaz gráfica amigable y fácil de usar por todos los usuarios, permitiendo consultas rápidas, navegación intuitiva y manteniendo los colores representativos de la empresa.

Seguridad

El sistema de horas extras debe proteger la información de todos los usuarios, permitiendo ingresar al sistema utilizando

usuario y contraseña, y restringiendo el acceso a los módulos dependiendo del grupo de usuario en sesión.

3.2 DISEÑO DE LA ARQUITECTURA

El Sistema de horas extras será utilizado en la intranet de la empresa, para lo cual se utilizará una Base de Datos implementada en MySQL que almacenará toda la información necesaria para la aplicación por lo que se necesita un servidor linux.

Además, como lo indica la Figura 3.1, se necesita un servidor donde se alojará el sitio web. Este servidor debe contar con Apache Tomcat 5 para permitir la ejecución de las páginas JSP creadas con la herramienta WebRatio.

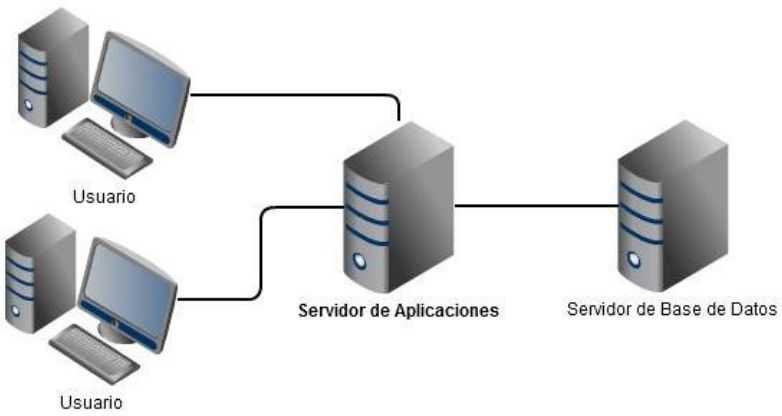

Figura 3.1 Arquitectura del Sistema de Horas Extras

Para una mayor escalabilidad del sistema en un futuro se podría configurar varios servidores para que las peticiones de los usuarios a

los servicios se distribuyan entre estos, tal como se muestra en la Figura 3.2. De esta manera se consigue alta disponibilidad y balanceo de carga, ya que cada servidor se hará cargo de un porcentaje de peticiones permitiendo que el sistema abarque mas usuarios, además en caso de fallo de un servidor se podrá contar con otro servidor disponible para prestar servicio.

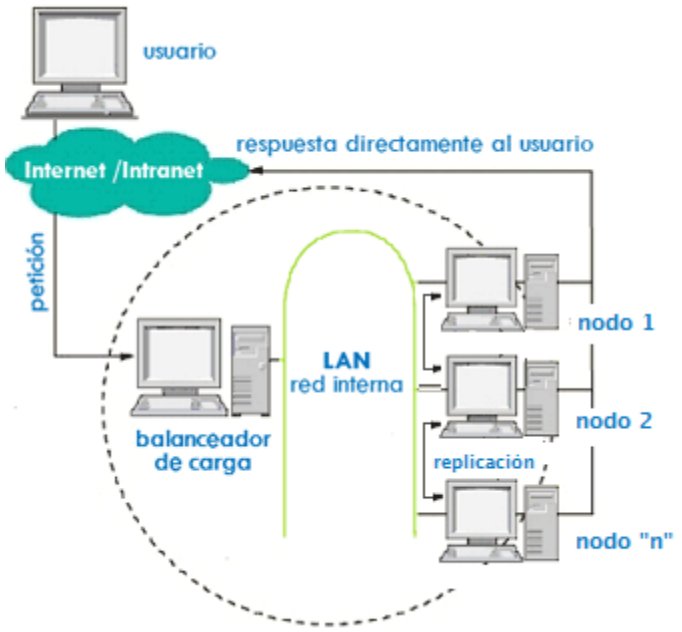

Figura 3.2 Balanceador de carga

Fuente: http://moodle.org/mod/forum/discuss.php?d=116594

La configuración recomendada incluye:

- Apache WebServer como balanceador
- Un cluster JBOSS como application server
- Un cluster a nivel de Base de Datos

3.3 DIAGRAMAS DE CASOS DE USO

Los casos de uso representan el comportamiento de un sistema a través de la interacción entre los usuarios y la aplicación. De esta manera determinan las funcionalidades que un sistema puede realizar. A continuación los casos de uso del sistema de horas extras:

CU_01	
Nombre del Caso de Uso	Autenticación
Actores	Empleados, Gerentes Técnicos, Jefes Departamentales, Administrador.
Flujo de eventos	1.- Iniciar en el Sistema De Horas Extras. 2.- Escribe su nombre de usuario y contraseña en los campos correspondientes. 3.- Clic en el botón Login.
Condiciones de Entrada	Abrir la página principal del sitio Horas Extras
Condiciones de Salida	El Sistema de Horas Extras presentará

	mensaje de confirmación.
Calidad de Requerimientos	Los campos a llenar serán validados. Creación de la sesión de usuario y abertura de la página principal de la site view asociada al grupo específico.

CU_02	
Nombre del Caso de Uso	Cerrar Sesión
Actores	Empleados, Gerentes Técnicos, Jefes Departamentales, Administrador.
Flujo de eventos	1.- Iniciar sesión en el Sistema de Horas Extras. 2.- Hace clic en la opción Logout.
Condiciones de Entrada	Tener una sesión abierta.
Condiciones de Salida	Sesión cerrada con éxito y visualización de un mensaje de confirmación.
Calidad de	

Requerimientos	

CU_03	
Nombre del Caso de Uso	Ingresar Nueva Actividad
Actores	Empleado
Flujo de eventos	1.- Iniciar en el Sistema De Horas Extras. 2.- Hace clic en la opción Ingresar Nueva Actividad. 3.- Llenar todos los campos requeridos. 4.- Hacer clic en el botón Guardar. 5.- El sistema mostrará un mensaje de confirmación.
Condiciones de Entrada	El usuario debe iniciar sesión en el sistema.
Condiciones de Salida	El Sistema de Horas Extras presentará mensaje de confirmación del Ingreso de la Nueva Actividad.

	El Sistema de Horas Extras pasara la Nueva Actividad a un periodo de Ingreso de Tiempos Estimados.
Calidad de Requerimientos	Los campos a llenar serán validados

CU_04	
Nombre del Caso de Uso	Ingreso de Tiempos Estimados de Una Actividad
Actores	Empleado
Flujo de eventos	1.- Iniciar en el Sistema de Horas Extras. 2.- Escoger la Actividad a la cual le desea Ingresar los Tiempos Estimados. 3.- Llenar todos los campos requeridos. 4.- Hacer clic en el botón Guardar. 5.- El sistema mostrará un mensaje de confirmación.
Condiciones de Entrada	El usuario debe iniciar sesión en el

	sistema.
Condiciones de Salida	El Sistema de Horas Extras presentará mensaje de confirmación del Ingreso de Tiempos Estimados en la Actividad. El Sistema de Horas Extras pasara la Horas Extras de la Actividad a la Aprobación del Jefe Departamental Respectivo.
Calidad de Requerimientos	Los campos a llenar serán validados.

CU_05	
Nombre del Caso de Uso	Revisión y Aprobación de Horas Extras de una Actividad por el Jefe Departamental.
Actores	Jefe Departamental
Flujo de eventos	1.- Iniciar el Sistema de Horas Extras. 2.- Escoger las Horas Extras de la Actividad a Aprobar.

	3.- Llenar todos los campos requeridos. 4.- Hacer clic en el botón Finalizar. 5.- El sistema mostrará un mensaje de confirmación.
Condiciones de Entrada	El usuario debe iniciar sesión en el sistema.
Condiciones de Salida	El Sistema de Horas Extras presentará mensaje de confirmación de la Aprobación de Horas Extras de la Actividad por parte del Jefe departamental. El Sistema de Horas Extras pasara la Horas Extras de la Actividad a la Aprobación de la Gerencia Técnica.
Calidad de Requerimientos	Los campos a llenar serán validados.

CU_06	
Nombre del Caso de Uso	Aprobación de Horas Extras de una Actividad por el Gerente Técnico

Actores	Gerente Técnico
Flujo de eventos	1.- Iniciar el Sistema de Horas Extras. 2.- Escoger las Horas Extras de la Actividad a Aprobar. 3.- Llenar todos los campos requeridos. 4.- Hacer clic en el botón Finalizar. 5.- El sistema mostrará un mensaje de confirmación.
Condiciones de Entrada	El usuario debe iniciar sesión en el sistema.
Condiciones de Salida	El Sistema de Horas Extras presentará mensaje de confirmación de la Aprobación de Horas Extras de la Actividad por parte del Gerente Técnico. El Sistema de Horas Extras pasara las Horas Extras de la Actividad a Nomina.
Calidad de Requerimientos	Los campos a llenar serán validados.

3.4 DISEÑO DEL PROCESO

Para modelar el proceso de Horas Extras en BPM, se comenzó analizando, mediante los requerimientos, el objetivo del proceso el cual es obtener un cálculo de horas extras consistente de cada empleado con el que se podrá ajustar costos, controlar los gastos de la empresa, y optimizar el tiempo que los jefes Departamentales, Gerencia Técnica y Recursos Humanos invierten en el proceso.

Definido el objetivo se procedió a localizar los actores del proceso y sus roles. Para el proceso de Horas Extras los roles que interactúan en el sistema son: Empleado, Jefe Departamental, Gerencia Técnica, Sistema, como se muestra en la Figura 3.3.

Figura 3.3 Roles del Proceso BPM

Además se definieron los objetos de negocio (Business Objects) del proceso de Horas Extras, definidos en la Figura 3.4, los cuales son: actividad, Colaborador, HistorialUsuario, HoraExtraActividad.

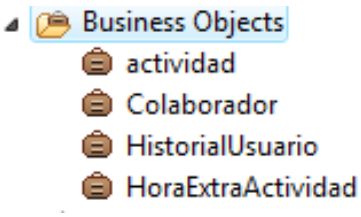

Figura 3.4 Objetos de Negocio del Proceso BPM

Después se definieron las tareas a realizar por cada uno de los roles del proceso. Se inició creando la pool llamada HE_Actividad en la que se ubicó las tareas definidas anteriormente. La primera tarea "Ingresar Nueva Actividad" marca el inicio del proceso. En esta tarea el empleado ingresa una nueva actividad, definiendo el tiempo que tomo realizarla y los colaboradores de la misma, siendo los objetos de negocio asociados a esta tarea actividad y Colaborador, cada uno con sus respectivos parámetros. La Figura 3.5 describe la tarea Ingresar Nueva Actividad.

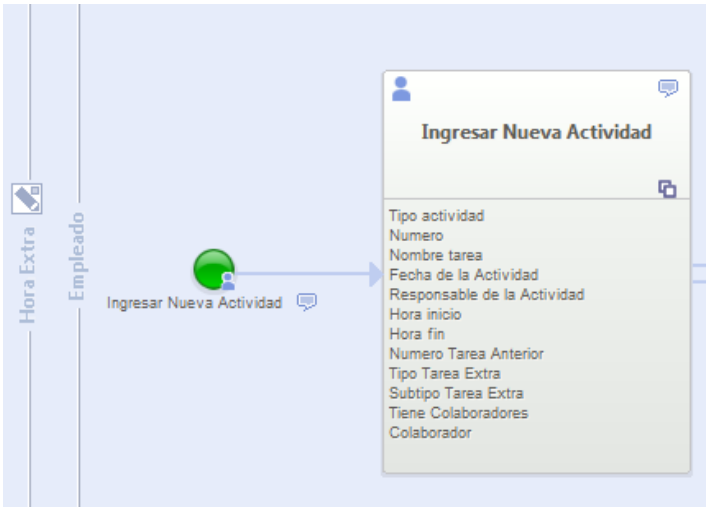

Figura 3.5 Ingresar Nueva Actividad

Luego de que el empleado Ingresa la actividad, mediante una Gateway exclusiva, se compara si la actividad tiene colaboradores. Si tiene colaboradores, se crea el subproceso de Aprobación de Horas Extras para cada uno de los colaboradores de la actividad además del subproceso para el responsable. Si la actividad no tiene colaboradores, solamente se crea el subproceso de Aprobación de Horas Extras para el responsable de la actividad. La Figura 3.6 presenta los subprocesos de Aprobación de Horas Extras para el Responsable y Colaborador.

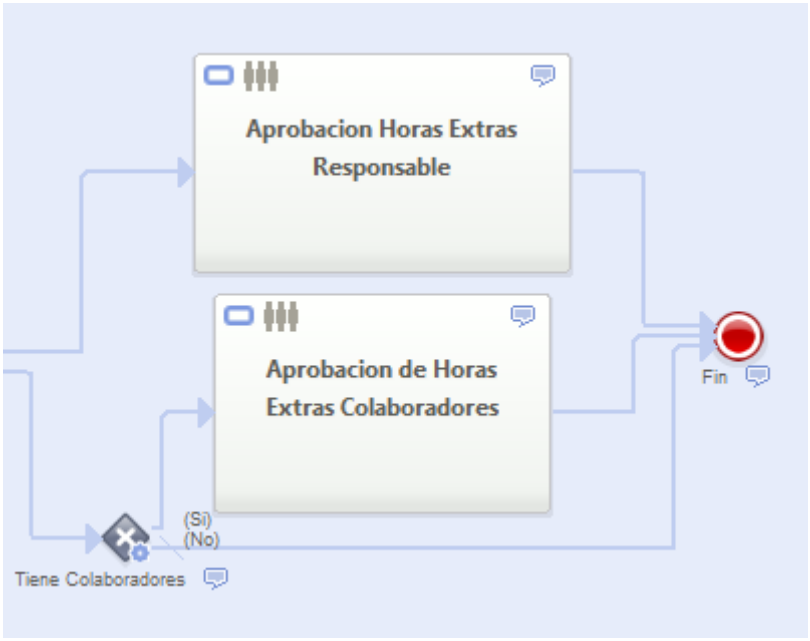

Figura 3.6 Subproceso Aprobación HoraExtra Responsable y Colaborador

El proceso de aprobación de horas extras para cada empleado participante de la actividad (responsable y colaboradores), inicia con

la tarea “Ingresar Tiempos Estimados” en la que el empleado participante de la actividad anteriormente creada, puede ingresar durante 48 horas, el tiempo estimado que le llevo realizar la tarea. Si pasadas las 48 horas el empleado no realiza ningún ingreso, se tomará como tiempo estimado el tiempo de la actividad ingresado al momento de crearla, quedando estos tiempos estimados como los tiempos de duración de la hora extra del empleado, la cual se crea en estado pendiente. La Figura 3.7 muestra el inicio de cada subproceso con la tarea Ingresar Tiempos Estimados.

Figura 3.7 Ingresar Tiempos Estimados

Después el sistema calcula el número de horas extras normales y extraordinarias de la actividad en base a los tiempos estimados obtenidos anteriormente. Una vez calculados, a través de una Gateway exclusiva se compara el estado de la hora extra. Si la hora extra se encuentra en estado pendiente, aprobada, escalada o en

revisión, se compara el total de las horas extras del empleado. Si el acumulado es mayor o igual a 48 horas, el sistema alerta al jefe departamental sobre estos tiempos y procede a notificarle que tiene una hora extra pendiente de revisar y aprobar, caso contrario solamente le notifica la hora extra pendiente de aprobación. La Figura 3.8 ilustra el cálculo de las horas extras ingresadas por el empleado a través de la tarea Calcular Acumulado Horas Extras.

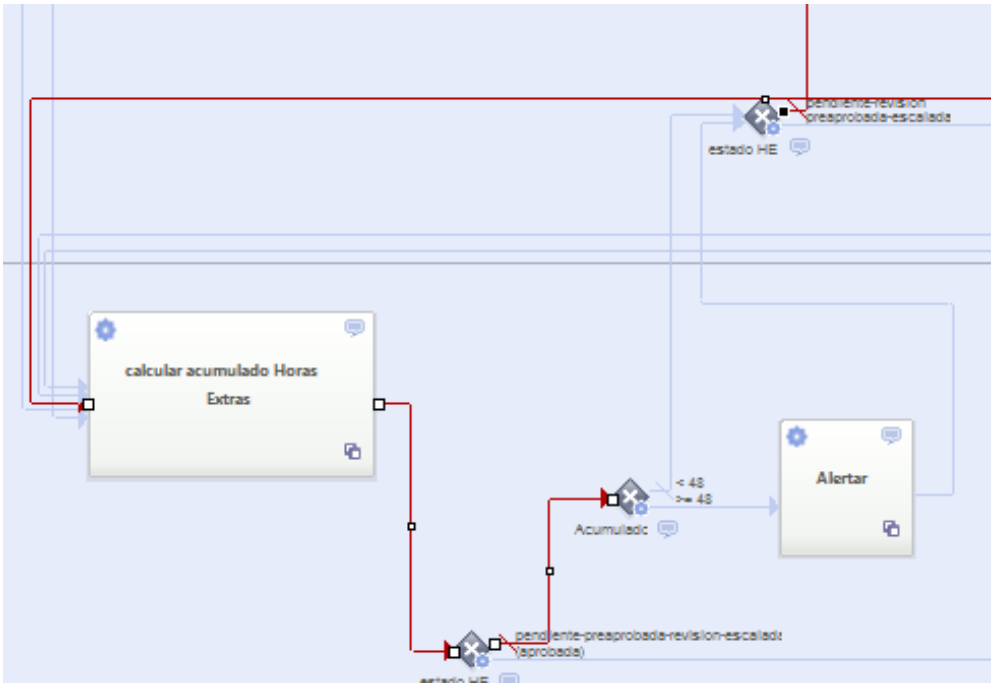

Figura 3.8 Calcular Acumulado Horas Extras Empleado - Alertar

Una vez notificada la hora extra, durante 72 horas, el jefe departamental puede aprobarla, permitiéndole verificar la información de la actividad, y modificar los tiempos estimados de la hora extra en caso de ser necesario, quedando la misma en estado preaprobada.

Si pasadas las 72 horas el jefe departamental no aprueba la hora extra, el sistema automáticamente la escala a gerencia técnica para su respectiva revisión y aprobación. La Figura 3.9 presenta el flujo de la aprobación de una hora extra por parte del Jefe Departamental mediante la tarea Revisión y Aprobación Horas Extras Jefe Departamental.

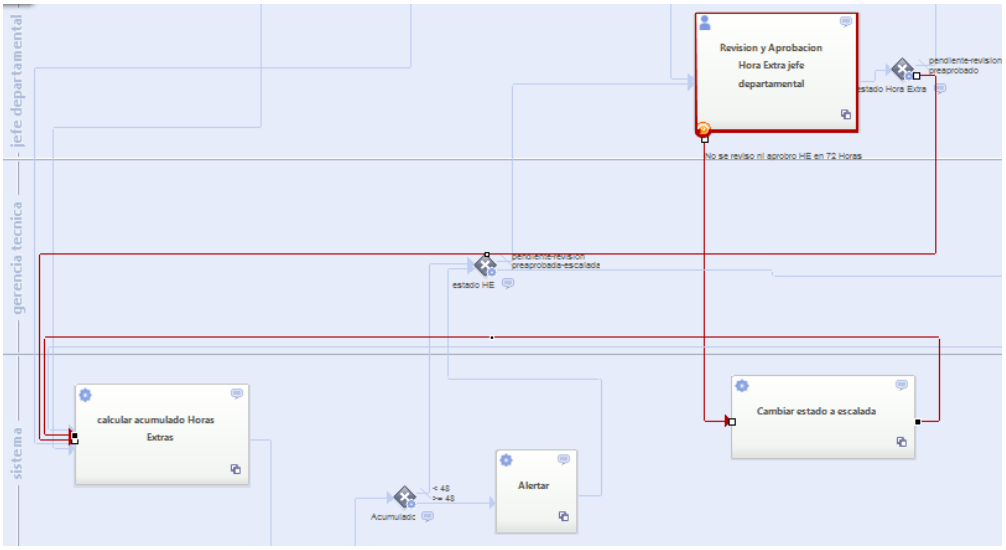

Figura 3.9 Revisión y Aprobación Horas Extras Jefe Departamental

Nuevamente el sistema vuelve a calcular el número de horas extras normales y extraordinarias de la actividad en base a los tiempos estimados de la hora extra. Después se alerta a gerencia técnica sobre estos tiempos, según lo definido anteriormente, y se procede a notificarle que tiene una hora extra pendiente de revisar y aprobar. La Figura 3.10 ilustra el cálculo de las horas extras ingresadas por el

Jefe Departamental a través de la tarea Calcular Acumulado Horas Extras.

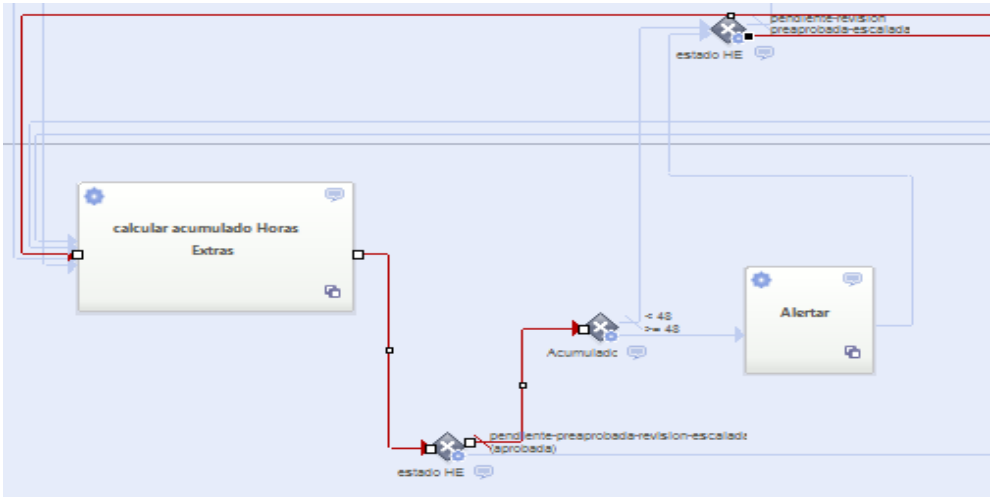

Figura 3.10 Calcular Acumulado Horas Extras Jefe Departamental– Alertar

Una vez notificada, durante 72 horas, gerencia técnica puede aprobar la hora extra o mandarla a revisión con el jefe departamental, permitiéndole verificar la información de la actividad, de la hora extra, y modificar los tiempos estimados de la misma, en caso de ser necesario. Si pasadas las 72 horas gerencia técnica no realiza alguna acción, el sistema nuevamente vuelve a notificarle que la hora extra se encuentra pendiente de revisión y aprobación. La Figura 3.11 presenta el flujo de la aprobación de una hora extra por parte de Gerencia Técnica mediante la tarea Revisión y Aprobación Horas Gerencia Técnica.

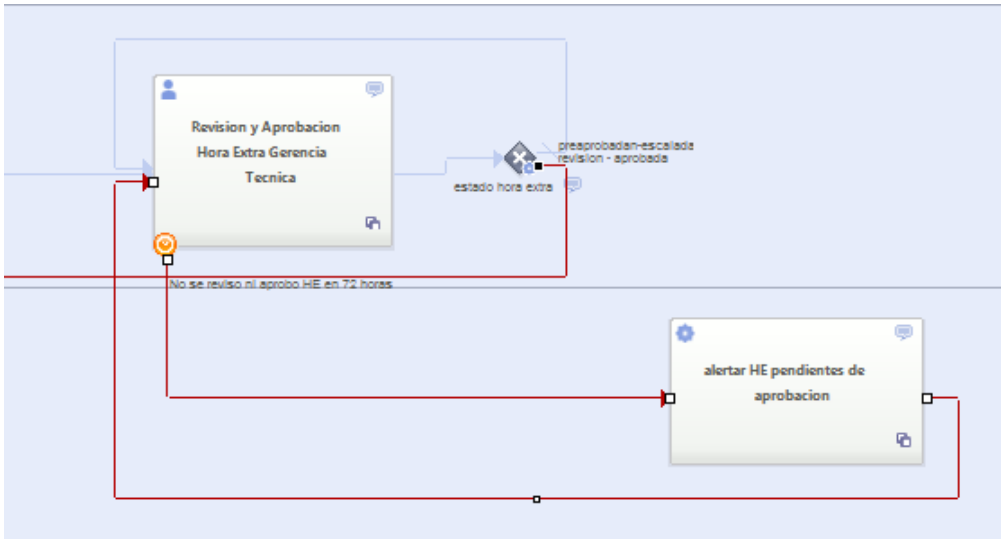

Figura 3.11 Revisión y Aprobación Horas Extras Gerencia Técnica

Si gerencia técnica decide mandar a revisión la hora extra, el sistema procede a notificarle al Jefe Departamental que tiene una hora extra proveniente de gerencia técnica para su revisión y aprobación, siguiendo el proceso ya antes descrito tal como se muestra en la Figura 3.12.

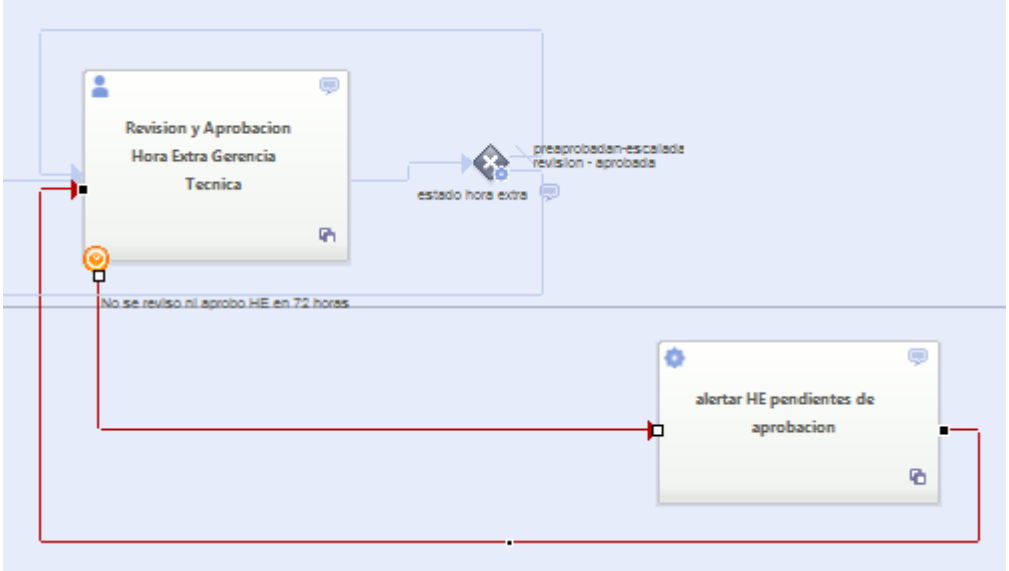

Figura 3.12 Alertar HE pendiente de Aprobación

Si gerencia técnica decide aprobar la hora extra, el sistema vuelve a calcular el número de horas normales y extraordinarias de la hora extra en base a los tiempos estimados, quedando lista para nomina, proceso descrito en la Figura 3.13. En esta etapa el valor calculado, se suma al total de horas de ese mes, obteniendo de este modo el número total de horas extras normales y extraordinarias del empleado en el mes en curso y finalizando de esta manera el proceso. El diagrama completo del Diseño del proceso BPM se presentará en el Anexo A de la presente tesis.

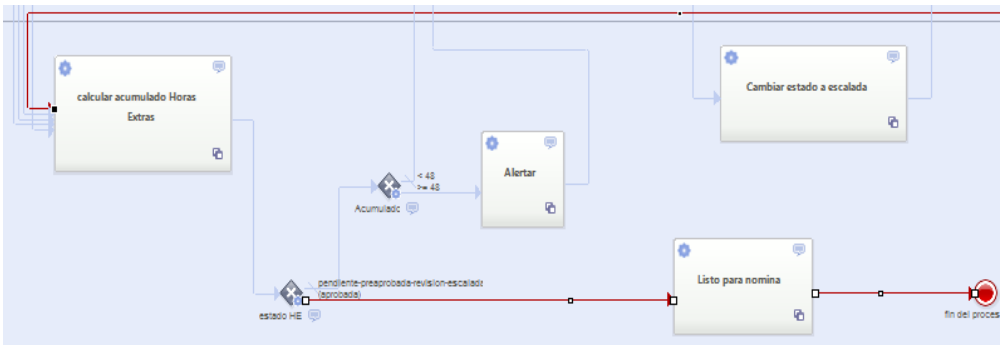

Figura 3.13 Calcular Acumulado Horas Extras Gerencia Técnica – Alertar – Listo para Nómina

3.5 DIAGRAMA DE LA BASE DE DATOS

Una vez realizado el diseño del proceso BPM, gracias al análisis de los requisitos, obtuvimos un prototipo del Sistema de Horas Extras. Por consiguiente, basándonos en el proceso de desarrollo con WebML, nos corresponde la etapa del diseño de datos. Para el diseño de la base de datos utilizaremos el modelo Entidad-Relación, el cual nos permite representar las entidades del sistema, las relaciones entre ellas y propiedades de nuestro modelo.

El diagrama E-R del sistema de horas extras se genera sincronizando el modelado BPM a WebML, de esta manera nuestros objetos de negocio se transforman en entidades junto con sus atributos respectivos. A continuación en la Figura 3.14 se muestra el Diagrama E-R generado:

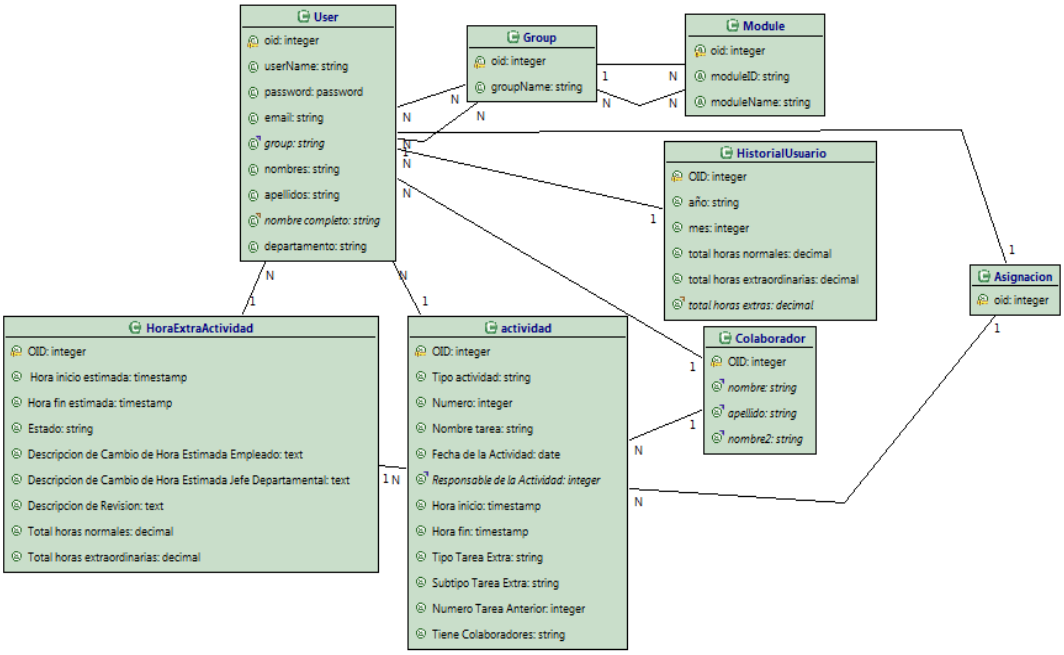

Figura 3.14 Modelo E-R

3.5.1 DETALLES EN EL DISEÑO DE DATOS

A continuación se detallará el diseño de datos del sistema de horas extras, explicando cada una de las entidades y relaciones generadas.

Cada vez que modelamos una aplicación web con WebRatio, la herramienta automáticamente crea 3 entidades importantes: user, group y module. Estas entidades nos ayudan a otorgar los permisos de acceso a los distintos módulos (site views, areas, pages) de la aplicación por parte de los usuarios, dependiendo del grupo al que pertenecen. Para el sistema de hora extras los

grupos de usuario son: Empleado, Jefe Departamental, Gerencia Técnica y Sistema. De esta manera los módulos quedan restringidos, permitiendo que ciertos grupos usuarios accedan a ellos. La Figura 3.15 presenta las relaciones entre las entidades User, Group y Module.

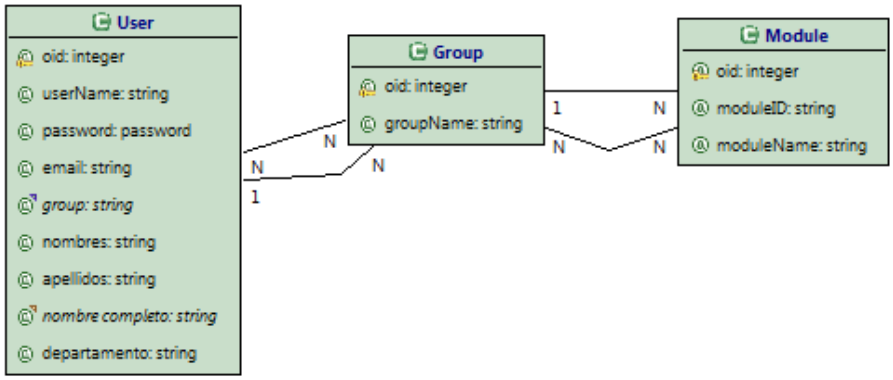

Figura 3.15 Relación User – Group – Module

La herramienta WebRatio adicionalmente crea las entidades Process, ActivityType, ProcessInstance y ActivityInstance, con el fin de gestionar la ejecución del BPM [8]. Representando la entidad Process a los procesos de la aplicación Web: HE_Actividad y HorasExtras, la entidad ActivityType a las actividades que componen cada proceso del sistema, la entidad ProcessInstance a las instancias de procesos creados por la aplicación al ejecutarse, y la entidad ActivityInstance a cada una de las actividades realizadas al ejecutarse las instancias del

proceso. En la Figura 3.16 podemos observar la relación entre las entidades ActivityInstance - ActivityType, y entre las entidades ProcessInstance - Process.

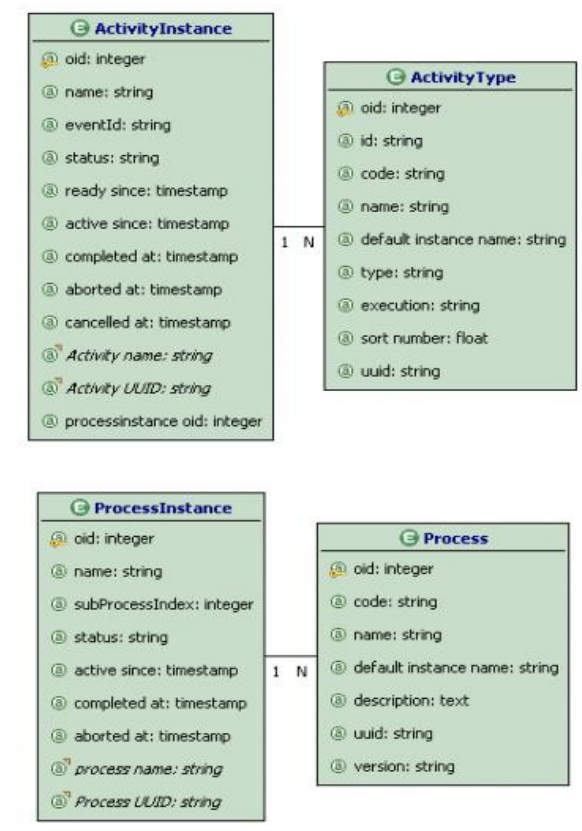

Figura 3.16 Relación ActivityInstance-ActivityType

ProcessInstace-Process

Tenemos la entidad actividad la cual representa la actividad ingresada en el sistema. Una actividad, además de poseer la hora de inicio y la hora de fin de la misma, tiene un solo responsable, y también puede tener uno o muchos colaboradores, siendo un colaborador un usuario del sistema.

La Figura 3.17 muestra las relaciones entre las entidades User, actividad y Colaborador.

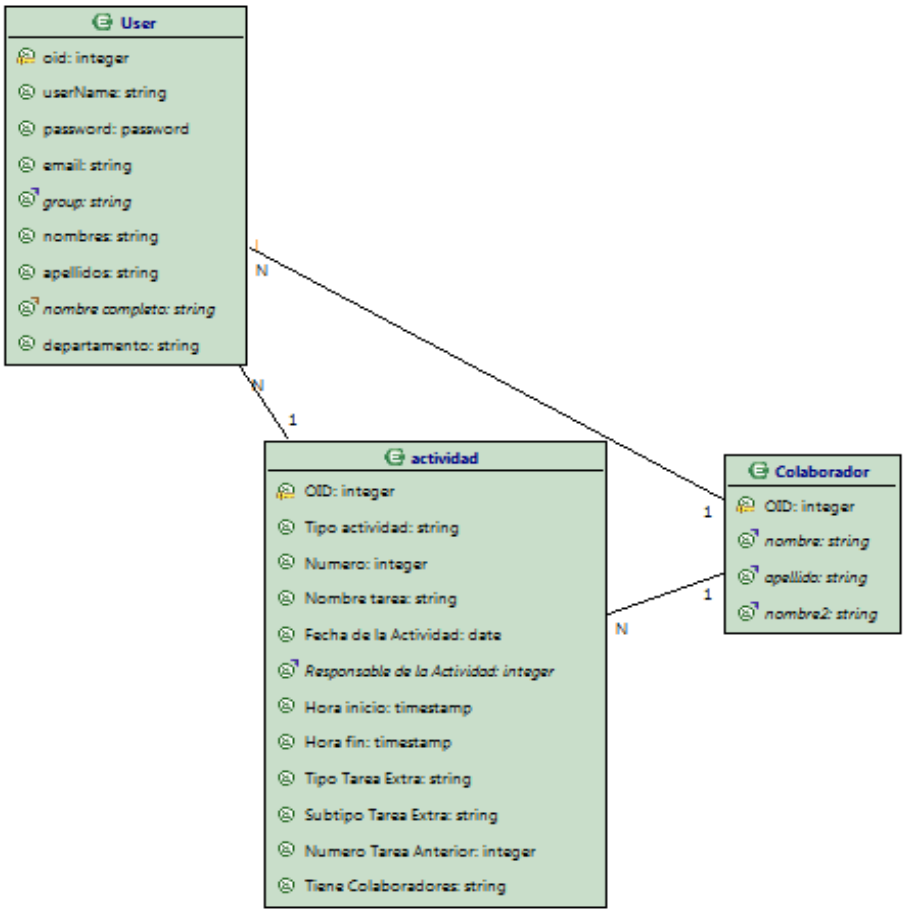

Figura 3.17 Relación User – actividad – Colaborador

Podemos notar que la entidad Usuario tiene un atributo derivado “nombre completo”, el cual se crea mediante la unión del atributo nombres y apellidos, formando el nombre completo del usuario, además posee el atributo departamento el cual representa el departamento al cual pertenece el usuario. Adicionalmente la entidad Actividad tiene un atributo derivado

simple “Responsable de la Actividad", el cual nos permite obtener el nombre del usuario responsable de la actividad, proveniente de la relación entre la entidad Usuario y Actividad. En la referencia [9] podemos encontrar información sobre derivaciones en el modelo de datos, las cuales se transforman automáticamente en vistas SQL.

A partir de una Actividad se generan una o muchas Horas Extras. Una hora extra le pertenece a un usuario del sistema (colaborador o responsable de la actividad) y contiene la hora inicio y fin estimadas, el total de horas normales y extraordinarias, además del estado en la que se encuentra. La Figura 3.18 muestra las relaciones entre las entidades User, actividad y HoraExtraActividad.

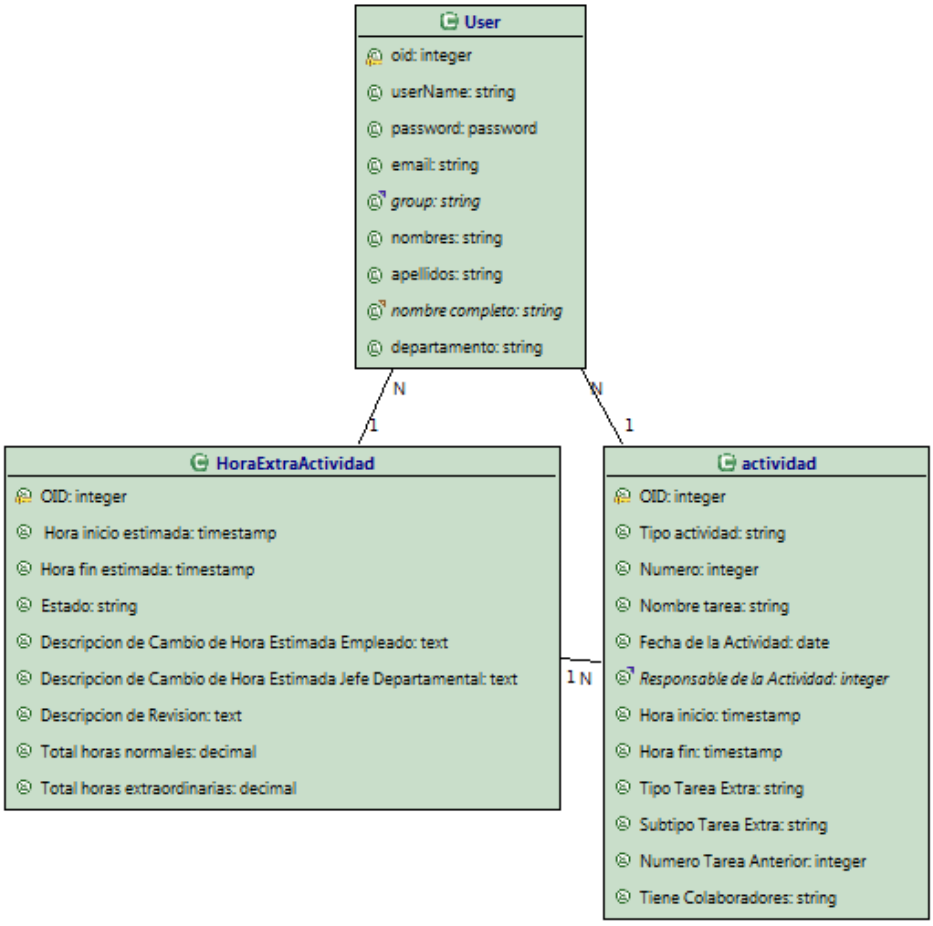

Figura 3.18 Relación User – HoraExtraActividad – actividad

Un usuario posee un Historial, en el cual se especifica el total de horas extras normales y extraordinarias que tiene en cada mes. La Figura 3.19 presenta la relación entre las entidades User e HistorialUsuario.

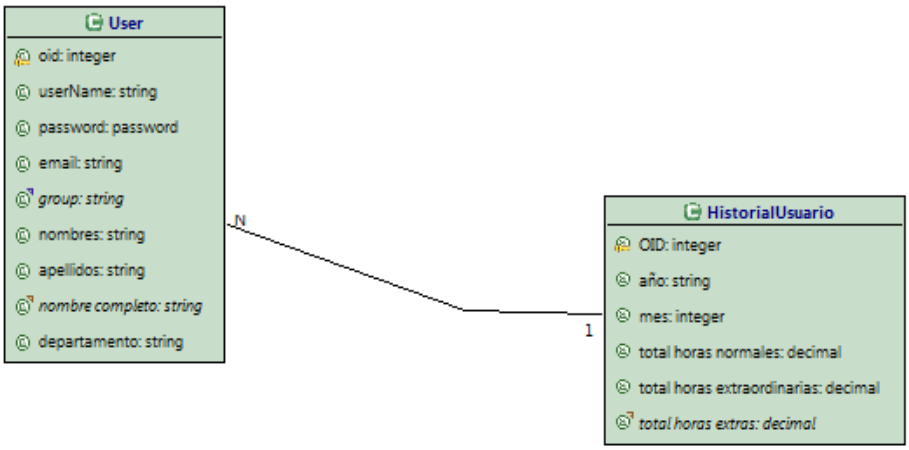

Figura 3.19 Relación User – HistorialUsuario

Podemos notar que la entidad HistorialUsuario tiene un atributo derivado calculado “total horas extras”, el cual se crea mediante la suma del atributo total horas normales y total horas extraordinarias, quedando como resultado el total de horas extras en el mes.

Tuvimos la necesidad de crear la entidad Asignación, para poder indicar si una actividad ya fue asignada a un usuario. En la Figura 3.20 podemos observar la relación entre las entidades User, actividad y Asignación respectivamente.

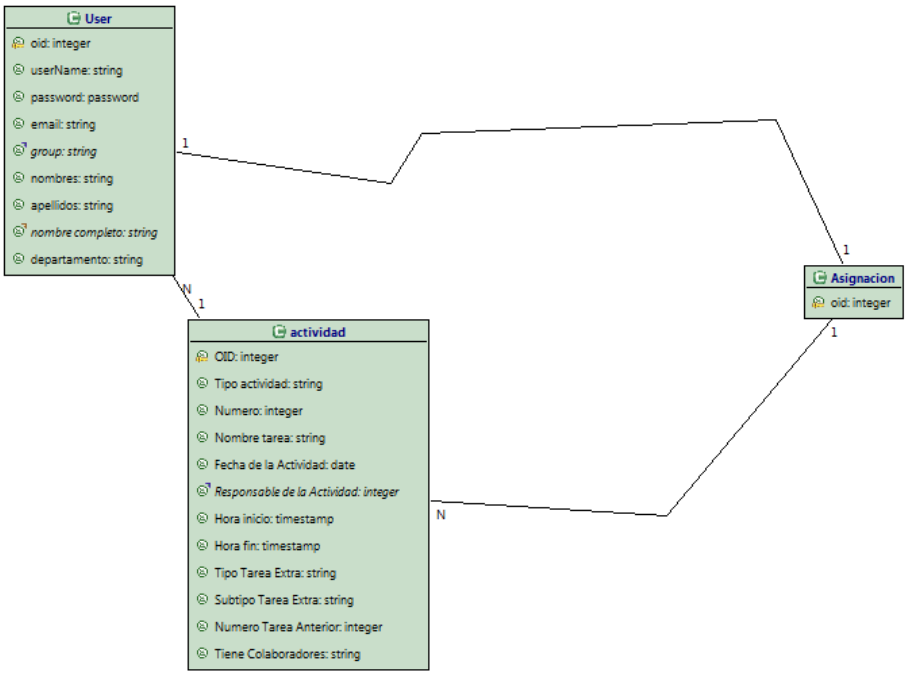

Figura 3.20 Relación User – actividad – Asignación

3.6 DISEÑO DEL HIPERTEXTO

Continuando con en el proceso de desarrollo con WebML nos corresponde el diseño del Hipertexto donde describiremos la composición y navegación del Sistema de Horas Extras. Al referirnos a composición hablamos de las units, páginas, etc. que componen el sitio web. Respecto a navegación nos referimos a cómo se podrá navegar a través de las páginas y contenidos de las units para formar el hipertexto.

3.6.1 SITE VIEWS DEL SISTEMA DE HORAS EXTRAS

El Sistema de Horas Extras presenta una interfaz amigable, sencilla, simple y estándar cumpliendo las funciones básicas y puntuales para lo cual fue desarrollado. Por este motivo el sistema se creó con tres vistas o site views como se puede ver en la Figura 3.21.

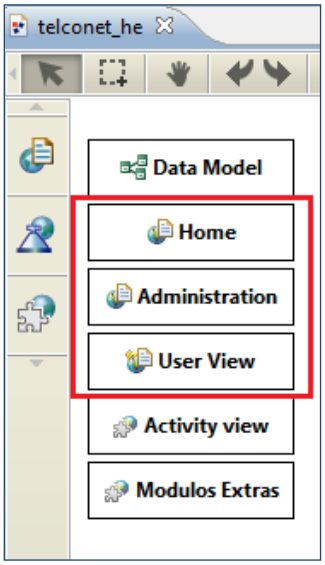

Figura 3.21 Site Views del Sistema de Horas Extras

Una site view contiene un conjunto de páginas, units y enlaces diseñados para cumplir con requisitos bien definidos ya sea el acceso a un dispositivo específico o los intereses de un grupo específico de usuarios.

3.6.1.1 SITE VIEW HOME

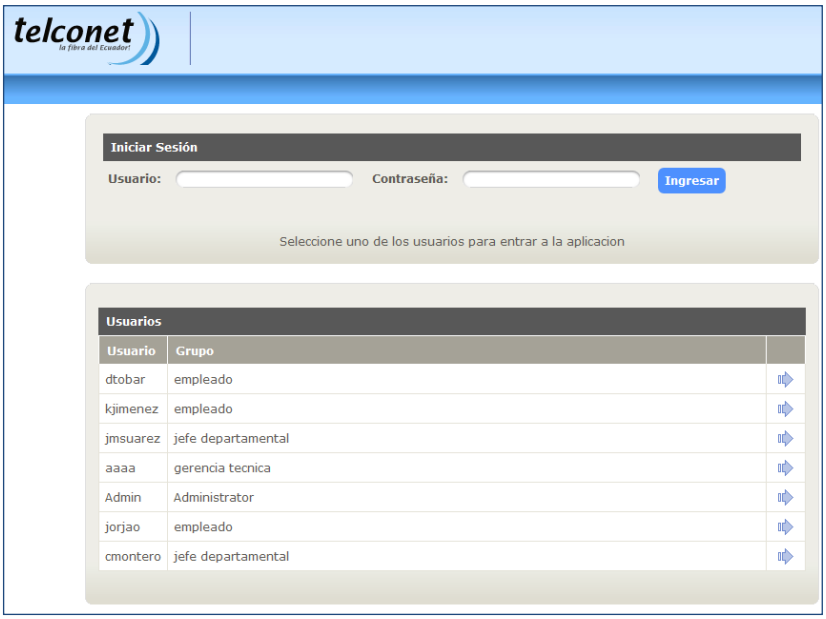

Figura 3.22 Site View Home

Descripción	Es la página de inicio y el punto de acceso a la aplicación web. Esta página forma parte de la site view pública, la cual no está protegida.
Grupos	Administrador, Empleado, Jefe Departamental, Gerencia Técnica.
Áreas Contenidas	Ninguna, tiene solo la página que permite a los usuarios iniciar sesión.

3.6.1.2 SITE VIEW ADMINISTRATION

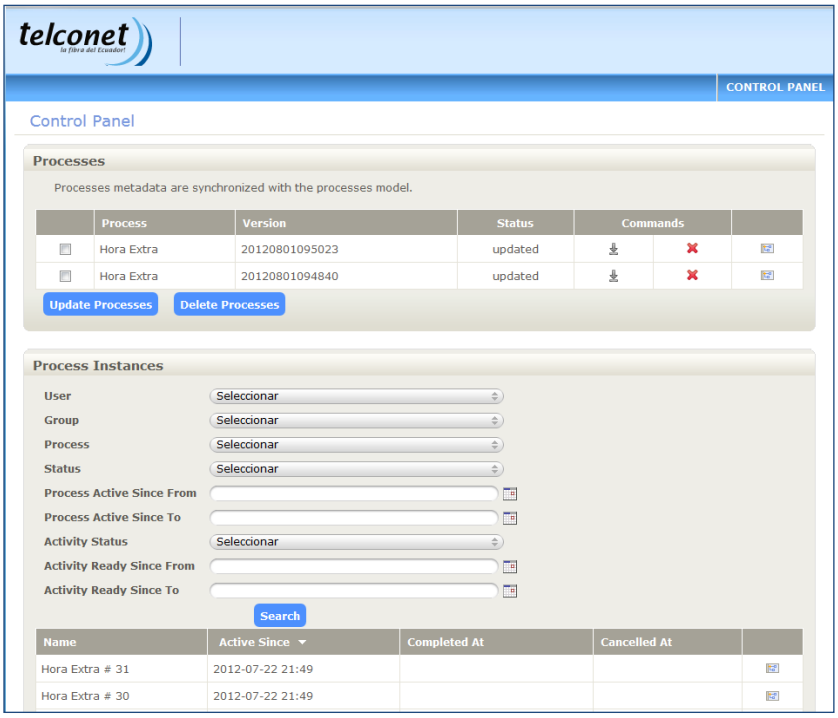

Figura 3.23 Site View Administration

Descripción	Es la site view en la que el administrador puede gestionar el contenido de toda la aplicación.
Grupos	Administrador
Áreas Contenidas	• Panel de Control: Permite conocer el detalle y el estado de todos los ProcessInstance que se creen durante la ejecución de la aplicación, también podrá

	visualizar un diagrama que sigue el proceso de la Aprobación de una Hora Extra.

3.6.1.3 SITE VIEW USER

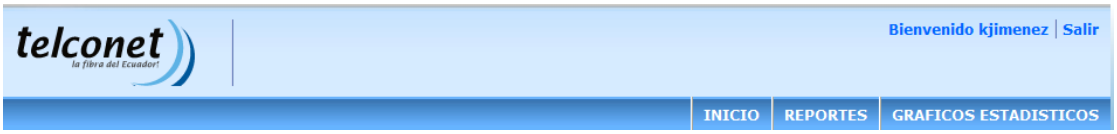

Figura 3.24 Site View User

Descripción	Es la site view dedicada a todos los grupos de usuarios de la aplicación. La site view User se encarga de mostrar los contenidos dependiendo del grupo al que pertenece el usuario en sesión. A través de esta vista se puede ingresar nuevas actividades, ingresar tiempos estimados de cada hora extra, aprobar una hora extra, generar reportes excel y visualizar gráficos estadísticos.
Grupos	Empleado, Jefe Departamental, Gerencia Técnica.
Áreas Contenidas	• Inicio: si el grupo del usuario es: - Empleado se mostrará la lista de las actividades pendientes, la lista actividades en espera de ingreso

de tiempos estimados y la lista de las horas extras aprobadas.

- Jefe Departamental o Gerencia Técnica se mostrará la lista de las Horas Extras pendientes de Aprobación.

- Reportes: Permite exportar en formato Excel las Horas Extras acumuladas y aprobadas de los empleados.
- Gráficos Estadísticos: Mostrara dos gráficos X/Y de los promedios de las horas extras normales y extraordinarias por mes.

3.6.2 DETALLE DE LAS SITE VIEWS DEL SISTEMA

En esta sección se profundizará el detalle de las elecciones de diseño aplicadas en las diferentes site views del Sistema de Horas Extras.

Dependiendo del tipo de usuario en sesión se mostrará la site view correspondiente. Para un usuario empleado observaremos en su página principal, como se muestra en la Figura 3.25, información sobre las horas extras tales como:

- Actividades Pendientes

- Tiempos Estimados
- Ingresar Nueva Actividad
- Horas Extras Aprobadas

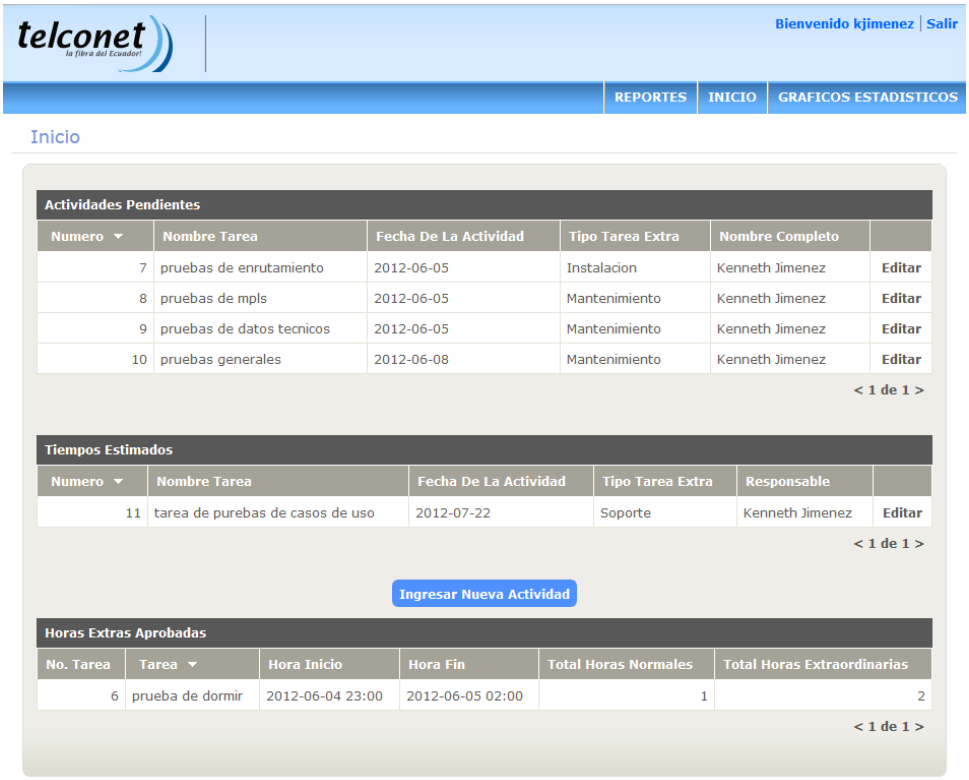

Figura 3.25 Página Principal de la Site View del Empleado

Empezaremos detallando el ingreso de una nueva Actividad. En la Figura 3.26 se presenta la página donde el empleado podrá realizar la creación de la nueva actividad ingresando la información respectiva.

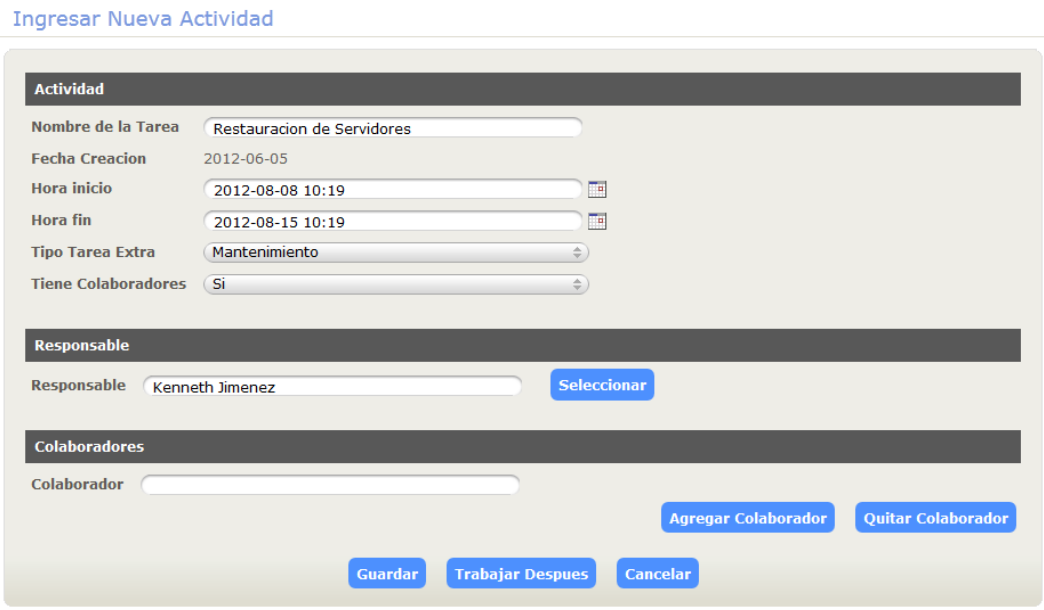

Figura 3.26 Página de Ingresar Nueva Actividad

La *Activity View* que se activa para el Ingreso de una nueva Actividad es "Hora Extra – Empleado - Ingresar Nueva Actividad" la cual está diseñada de la siguiente manera:

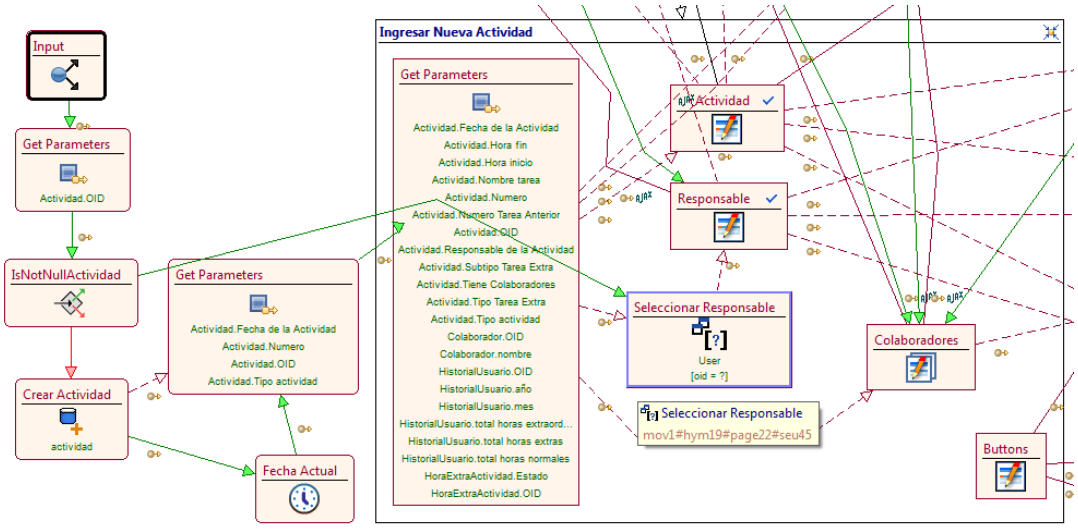

Figura 3.27 WebML de Ingresar Nueva Actividad (parte 1)

En la figura 3.27 vemos claramente como se verifica con la *Activity Parameters Unit* y la Is *Not Null Unit* si la Actividad que el usuario va a trabajar es una guardada anteriormente o es una totalmente nueva. Si la Actividad a trabajar es nueva, se utiliza la *Create Unit* para crear la nueva Actividad y la *Time Unit* para pre cargar la fecha en la cual se está creando. Así también se pre cargaran todos los datos ingresados si la Actividad a trabajar es una guardada con anterioridad.

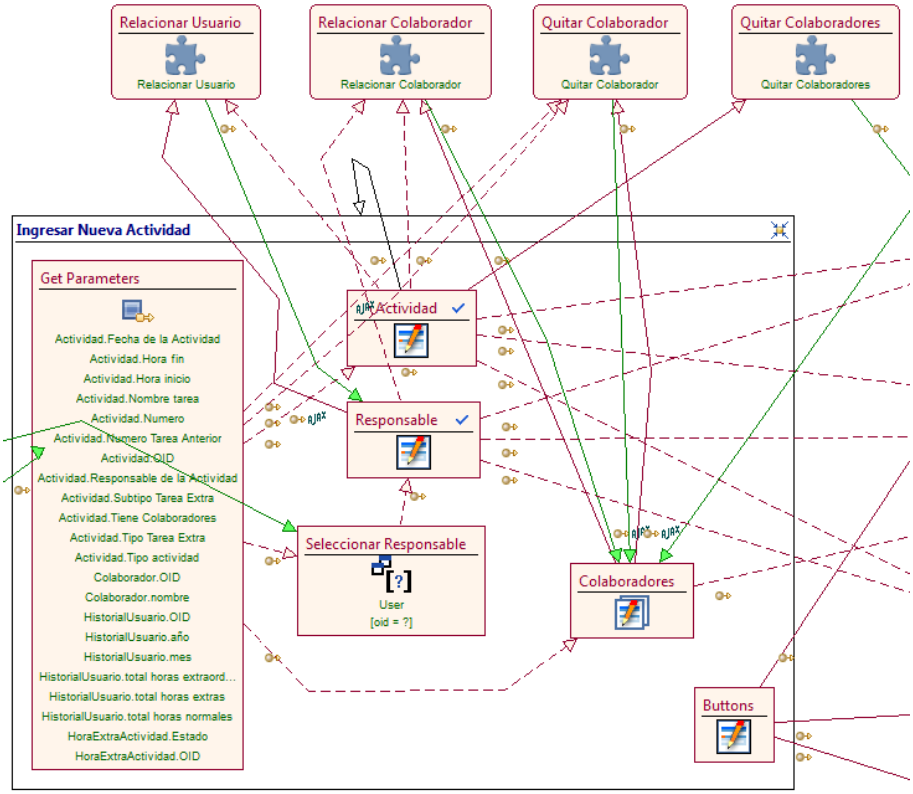

Figura 3.28 WebML de Ingresar Nueva Actividad (parte 2)

En la Figura 3.28 se observan las *Entry Units* y los módulos utilizados para el ingreso de la información necesaria para la creación de la nueva Actividad y las relaciones con los empleados sean estos como responsable o colaboradores. Comenzaremos detallando el ingreso del responsable de la Actividad.

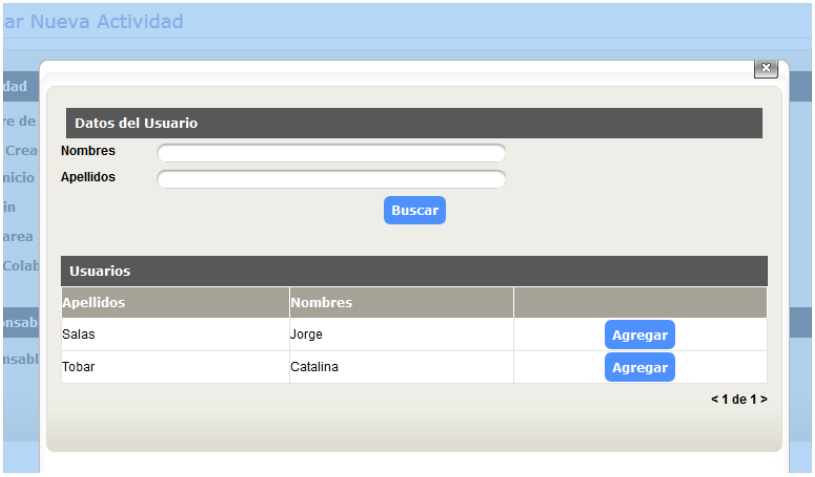

Figura 3.29 Página Ajax de Ingreso del Responsable de la Actividad

En la Figura 3.29 se observa la Página Ajax para la relación de la Actividad con el empleado responsable y obtención de información del mismo. Para este procedimiento se crearon dos Módulos los cuales podemos observar en la Figura 3.30:

Figura 3.30 Módulos para el Ingreso del Responsable

A continuación detallaremos el Módulo “Módulos Extras – Seleccionar Usuario” que es el que contiene la página que observamos en la Figura 3.29.

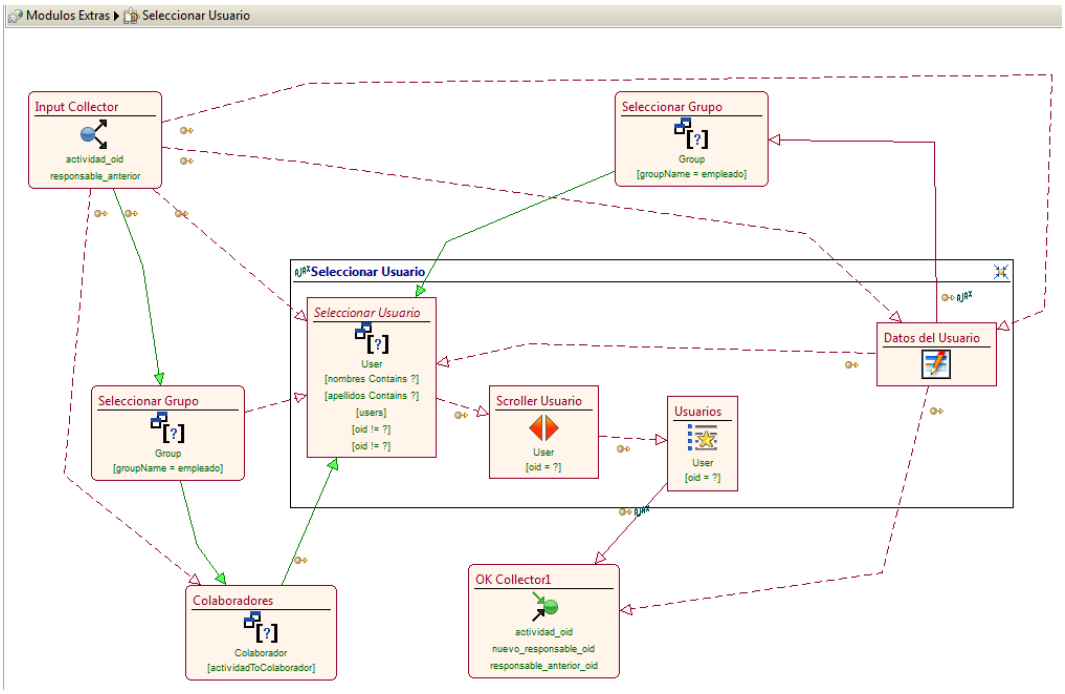

Figura 3.31 WebML del Módulo Seleccionar Usuario

En la Figura 3.31 se muestra como utilizamos la *Selector Unit* para filtrar los empleados a mostrar por Rol que en este caso serán solo los que cumplan con el Rol de “Empleado”. También

se filtran los Empleados que ya constan como responsable o como colaborador para que ya no sean mostrados y puedan ser escogidos. Esta Página Ajax es utilizada tanto para elección del Empleado Responsable de la Actividad como para la elección de los Empleados Colaboradores de la misma.

Una vez descrito el Módulo “Módulos Extras – Seleccionar Usuario” pasaremos al Módulo que crea la relación del Empleado escogido con la Actividad este es el Módulo “Módulos Extras – Relacionar Usuario”.

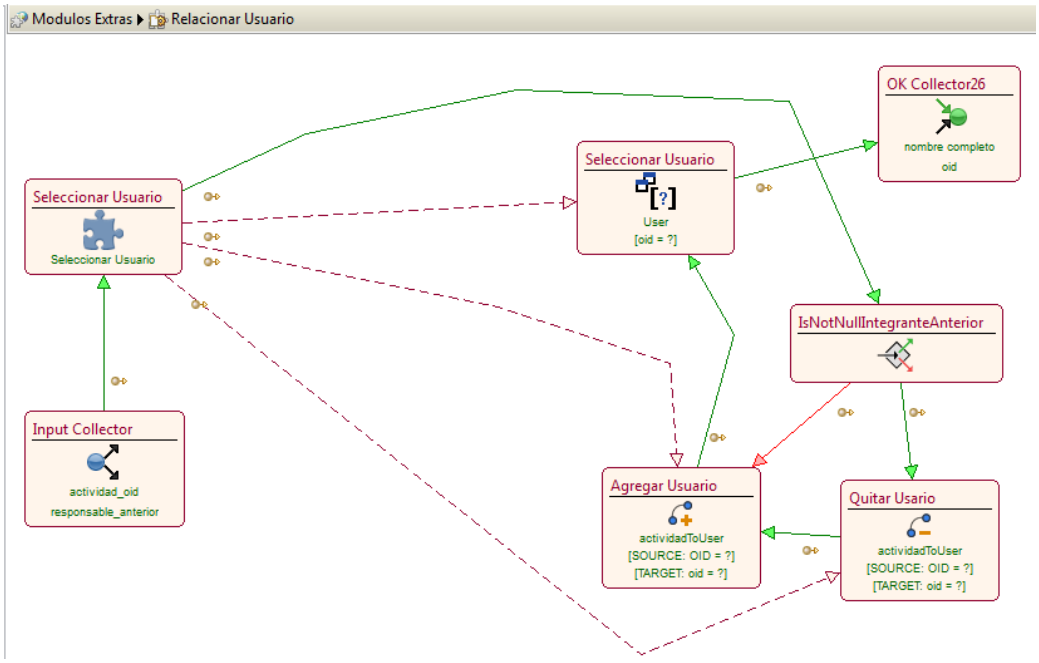

Figura 3.32 WebML del Módulo Relacionar Usuario (parte 2)

En la Figura 3.32 se muestra el módulo de Relacionar Usuario que utiliza la Connect Unit para crear la relación “actividadToUser” y así asignar al Empleado Responsable de la Actividad, también se hace uso de la *Is Not Null Unit* para verificar si la Actividad ya contiene un responsable y luego utilizar la *Disconnect Unit* para eliminar la Relación existente y así poder crearla con el nuevo Empleado escogido.

Como se mencionó anteriormente para agregar o eliminar Colaboradores a la Actividad se hace uso de los mismos módulos, así que por tal motivo no se volverá a detallar el funcionamiento de los mismos.

Lo próximo a describir será como se distribuye la Actividad ingresada para todos los empleados involucrados una vez guardada. Para este Procedimiento se creó el módulo “Activity view – Ingresar Tiempos Estimados [DISTRIBUCION]” el cual se muestra en la Figura 3.33.

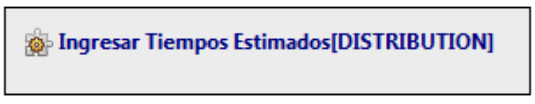

Figura 3.33 Módulo de Distribución de la Actividad

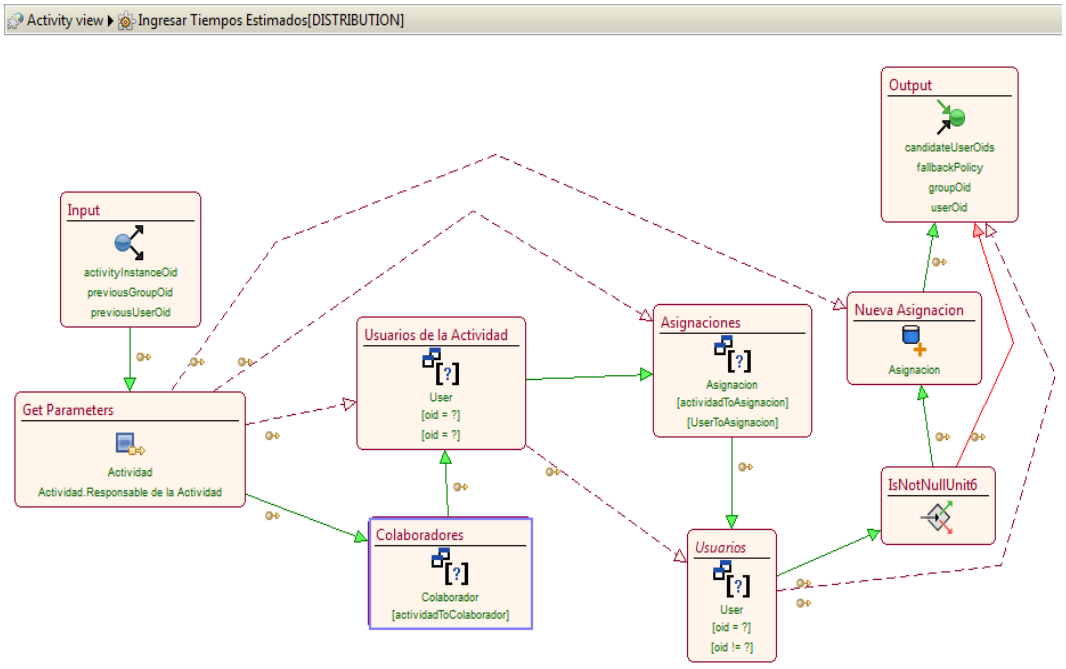

Figura 3.34 WebML del Módulo de Distribución de la Actividad

En la Figura 3.34 se presenta el diseño del modulo de Distribución de la actividad. En este módulo se realizan con las *Select Unit* la Obtención del Responsable y de sus respectivos colaboradores para con la *Create Unit,* ingresar los registros en la Tabla con Nombre “Asignación”. Esta es la Tabla que contendrá la relación de los empleados involucrados en la Actividad.

Este módulo se ejecuta cada vez que el responsable o un Colaborador de la Actividad va a Ingresar los Tiempos Estimados de una Hora Extra. Se controla que no se realice una doble asignación de la actividad excluyendo los usuarios que ya

contengan la relación, eso se lo realiza con una *Select Unit* de la Tabla Asignación y no generando relaciones para los usuarios que ya existan en dicha tabla.

La siguiente Site View que detallaremos será la “Hora Extra – Empleado – Ingresar Tiempos Estimados” la cual podemos observar en la Figura 3.35

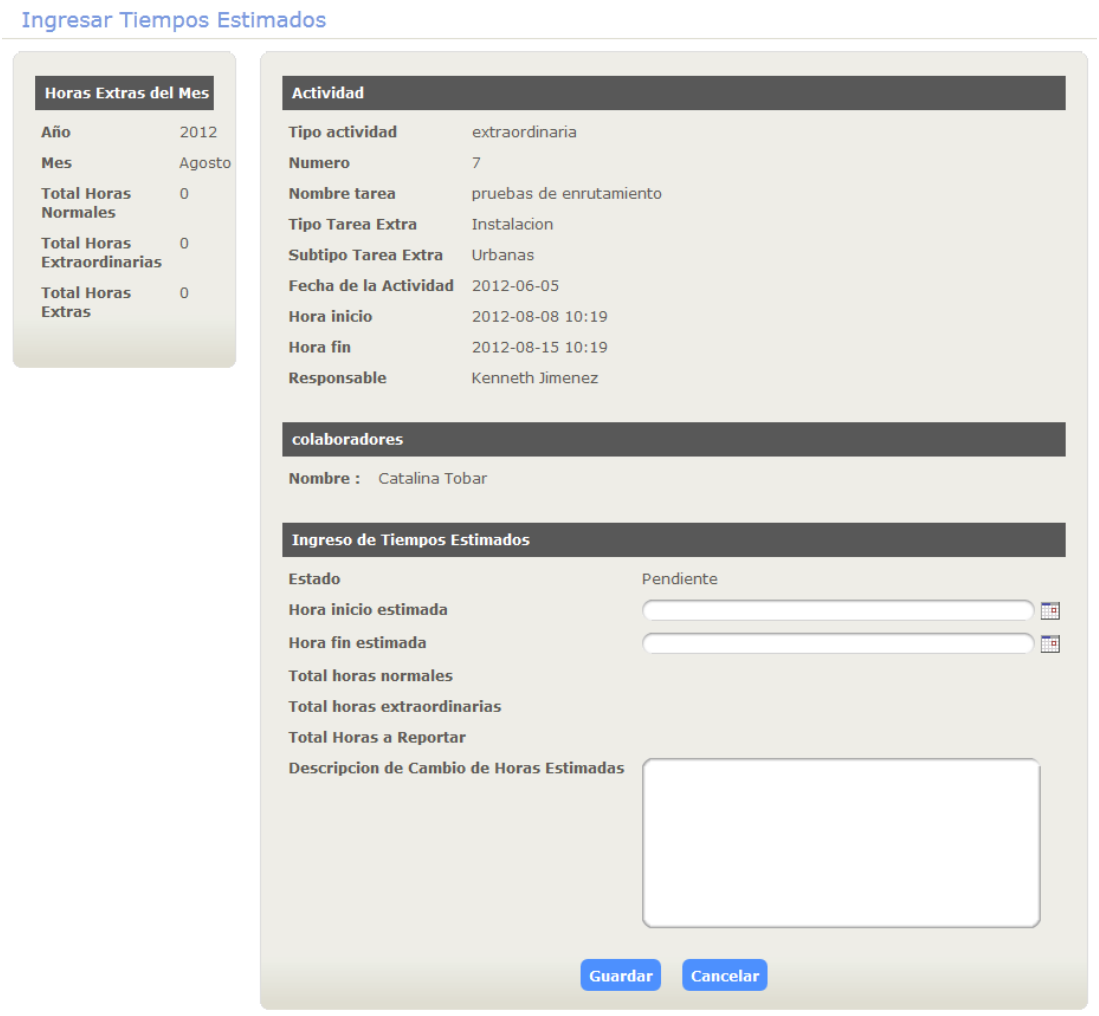

Figura 3.35 Página de Ingresar Tiempos Estimados

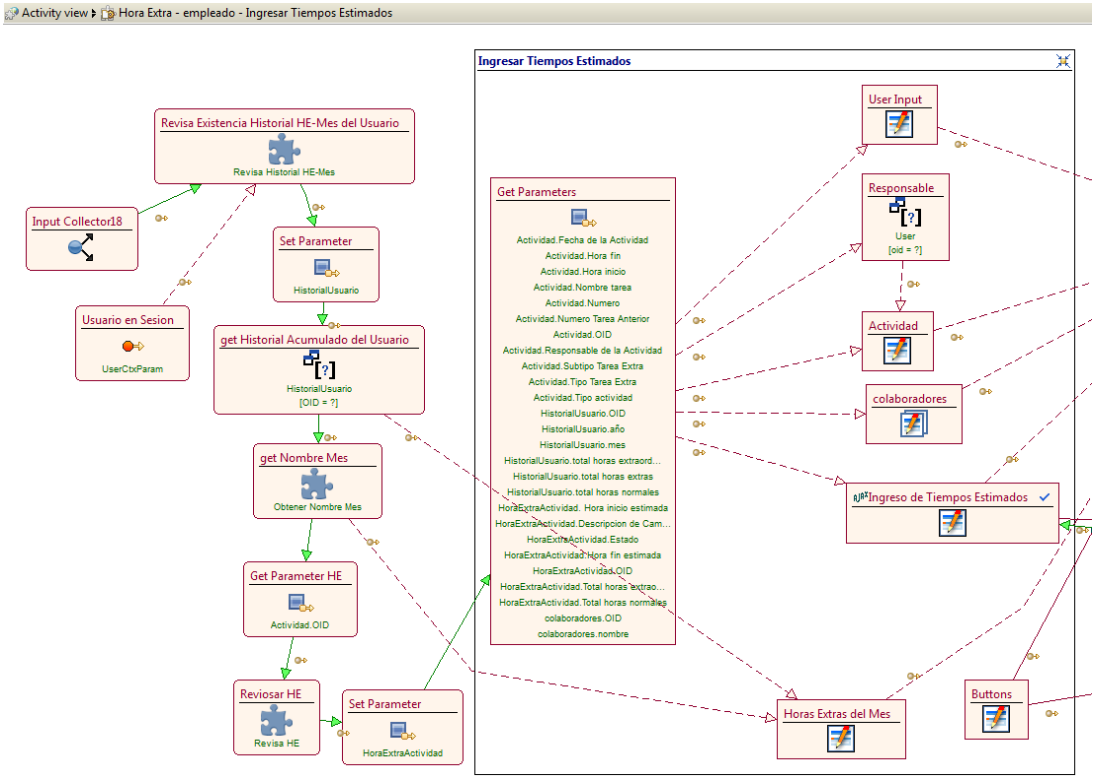

Figura 3.36 WebML de Ingresar Tiempos Estimados (parte 1)

En la Figura 3.36 podemos observar el diseño WebML de la página Ingresar Tiempos Estimados. En este diseño se observa que se hace uso de varias Units y Módulos que recolectan información sobre el Historial y acumulado de las Horas extras del Mes del Empleado, todo esto antes de activar la Site View, veremos uno a uno lo que realiza cada módulo que interviene en este diseño. Empezaremos con el módulo “Módulos Extras – Revisa Historial HE - Mes” el cual se muestra en la Figura 3.37.

Figura 3.37 Módulo de Revisión del Historial de Horas Extras del Mes

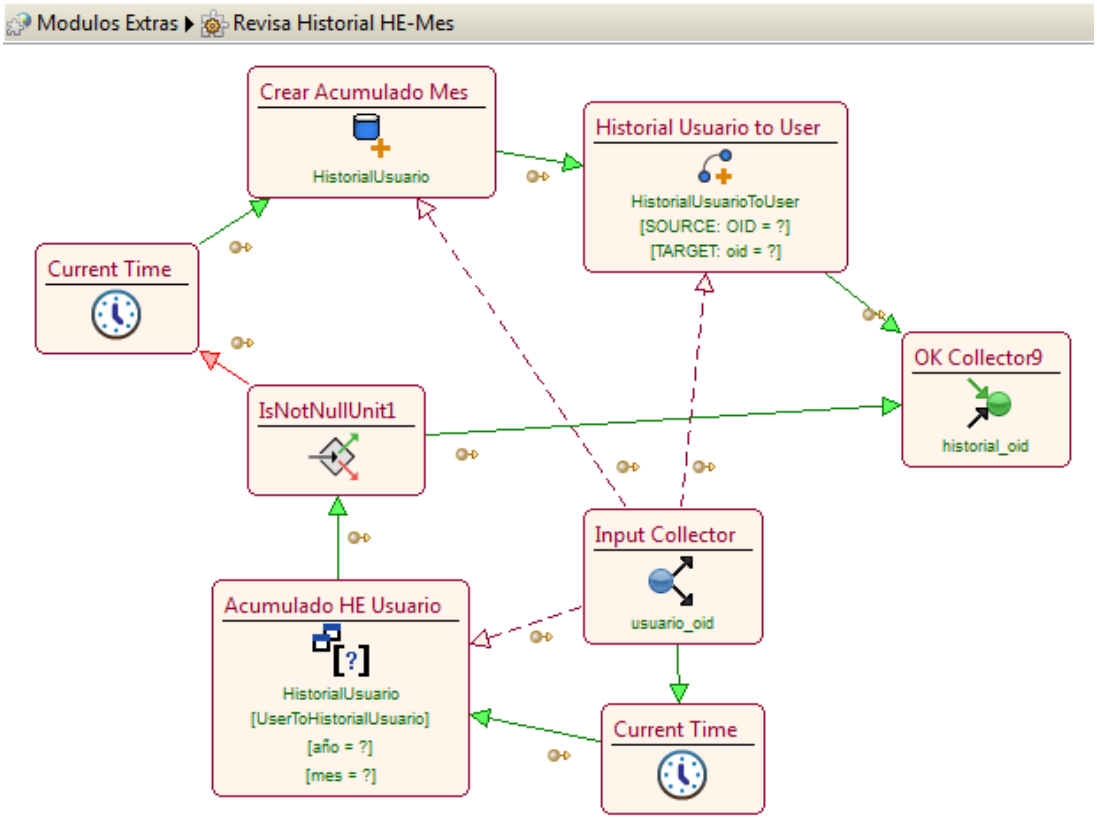

Figura 3.38 WebML del Modulo Revisar Historial HE-Mes

En la Figura 3.38 observamos el diseño del módulo Ingreso de Tiempos Estimados, el cual tiene como entrada el usuario en sesión y en donde podemos observar el uso de la *Time Unit* para obtener tanto el año y el mes actual, y con estos datos poder utilizar una *Select Unit* y seleccionar los datos existentes

del Historial del usuario. También se hace uso de la *Is Not Null Unit* para verificar que si no existe tal Historial de Horas Extras del Empleado del Mes Actual se cree uno como podemos observar involucrando a las *Time, Create y Connect Units.*

Siguiendo con el diseño de la Site View “Hora Extra – Empleado – Ingresar Tiempos Estimados” podemos observar que el siguiente módulo utilizado es el “Módulos Extras – Obtener Nombre Mes” el cual se muestra en a Figura 3.39.

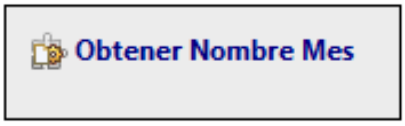

Figura 3.39 Módulo Obtención del Nombre de un Mes

Módulo sencillo creado debido a que la *Time Unit* maneja los meses con números y se necesitan los nombres para poder mostrar a los usuarios. Contiene el siguiente diseño presentado en la Figura 3.40:

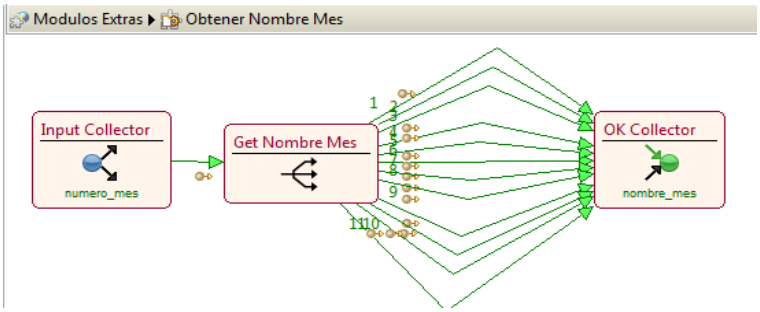

Figura 3.40 WebML del Módulo de Obtención del Nombre de un Mes

Como se puede observar se hace uso de una *Switch Unit* que tiene como entrada el número del mes y dependiendo de esta devolverá el Nombre del Mes correspondiente. Luego de la obtención de estos Datos para la presentación al Empleado de su acumulado de Horas Extras del Mes en el diseño de la Site View “Hora Extra – Empleado – Ingresar Tiempos Estimados” tenemos la utilización de otro Módulo, con nombre “Módulos Extras – Revisa HE” el cual se muestra en la Figura 3.41.

Figura 3.41 Módulo que revisa las Horas Extras por Actividad de los Empleados

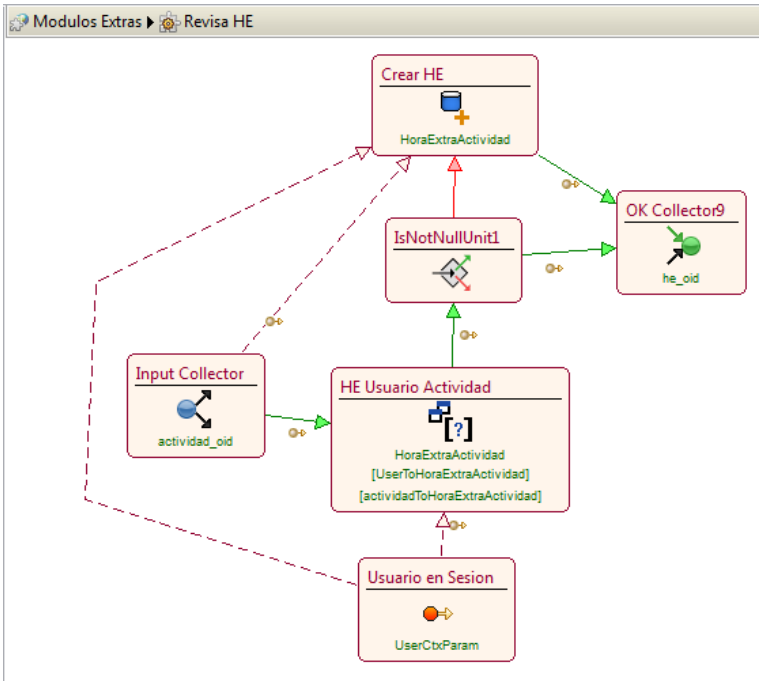

Figura 3.42 WebML del Módulo Revisa HE

En la Figura 3.42 se presenta el diseño del módulo Revisar HE, donde podemos observar que utilizamos la *Select Unit* para obtener las Horas Extras de una Actividad del empleado en sesión. Si no existen las horas extras, se procede a su creación por lo que utilizamos la *Create Unit.*

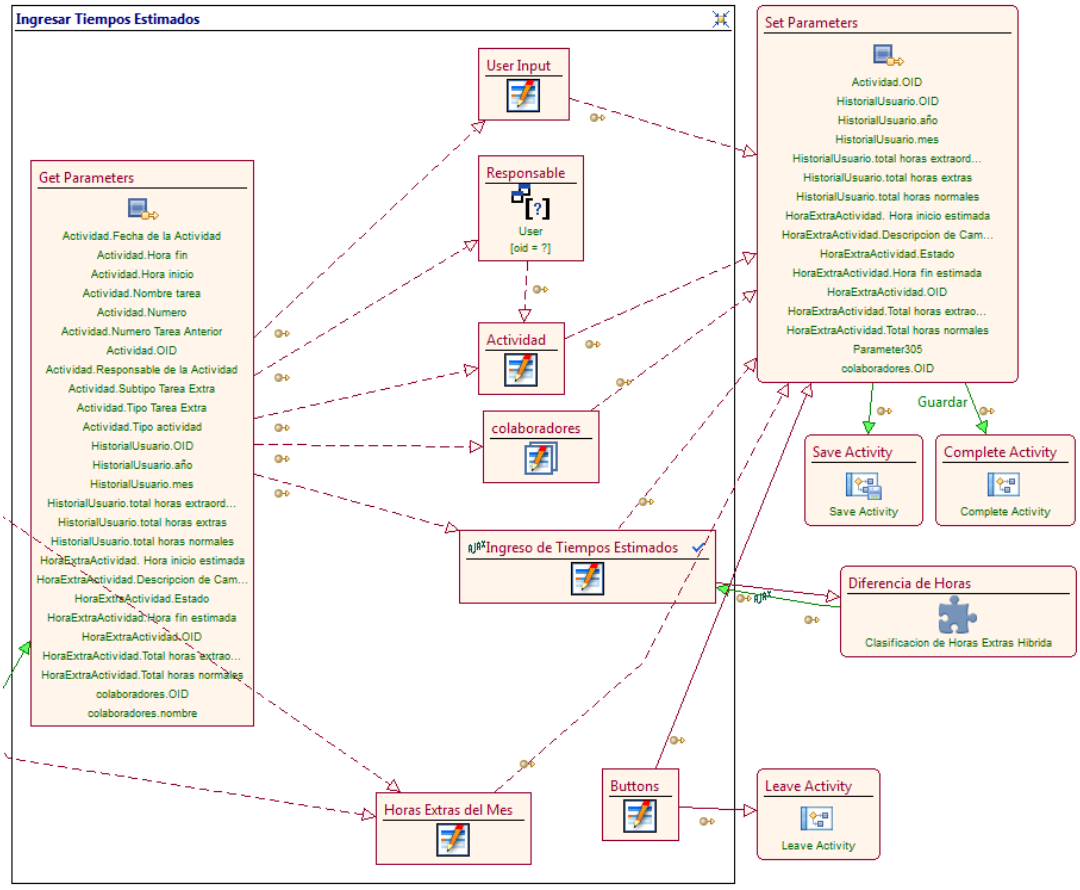

Figura 3.43 WebML de Ingresar Tiempos Estimados (parte 2)

Como podemos Observar en la Figura 3.43 ya tenemos las *Entry Units* que mostrarán al usuario los datos de la Actividad. También tenemos la *Entry Unit* que permitirá al empleado

ingresar la Hora inicio Estimada y la Hora Fin Estimada. Para el cálculo automático del total de Horas Extras a reportar por Actividad se lo realizo mediante un requerimiento Ajax y con la creación de un Módulo llamado “Módulos Extras – Clasificación de Horas Extras Hibrida” el cual se muestra en la Figura 3.44.

Figura 3.44 Módulo de Clasificación de Horas Extras Hibrida

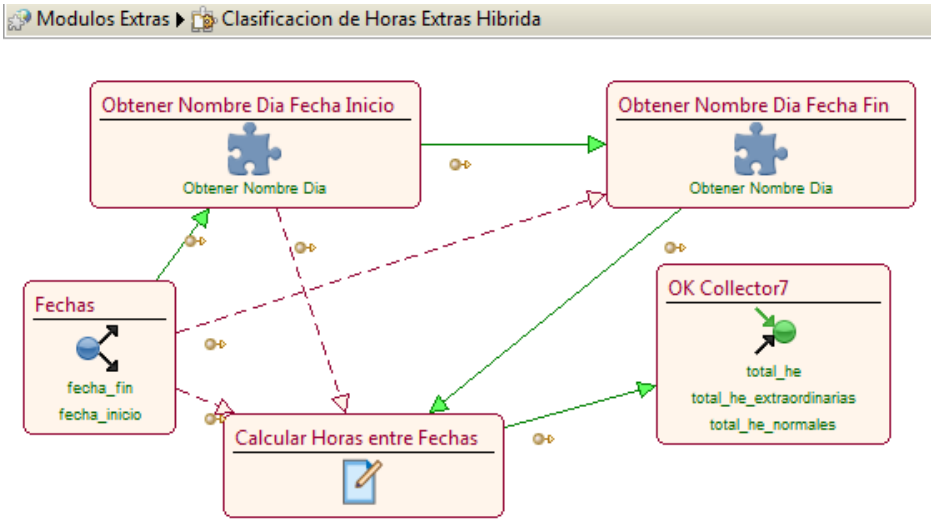

Figura 3.45 WebML del Módulo Clasificación de Horas Extras Hibrida

En la figura 3.45 observamos el diseño del módulo de Clasificación de Horas Extras Hibrida, en donde tenemos las Units que intervienen en el proceso de la Clasificación de las

Horas Extras de una Actividad, entre ellas describiremos una nueva que utilizamos que es la *Script Unit* que es en realidad la que realiza toda la clasificación de las Horas Extras de la Actividad, pero antes de ver su funcionalidad describiremos un Módulo llamado “Módulos Extras – Obtener Nombre Día” que se muestra en la Figura 3.46 y el cual es utilizado para la obtención del Nombre de un Día a partir de una fecha.

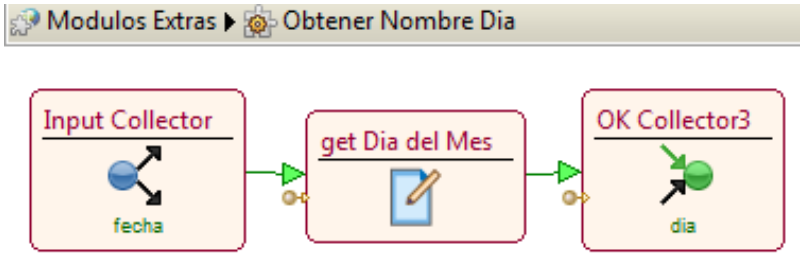

Figura 3.46 WebML del Módulo Obtener Nombre Día

Como podemos observar en este módulo también se hace uso de la *Script Unit* para obtener el nombre del día. En el Anexo B podemos encontrar el código que se implementó en la *Script Unit* el cual esta codificado en Groovy [10].

Continuando con la explicación del WebML del Módulo Clasificación de Horas Extras Hibrida, en el Anexo C tenemos el código del script que realiza la Clasificación de las Horas Extras, el cual dependiendo del día y de la hora, clasifica las Horas Extras de una Actividad como extraordinarias o normales.

Esta clasificación depende del horario laboral del empleado, y el sistema las calculará automáticamente al momento de Ingresar los Tiempos Estimados, o cuando vence el plazo del mismo. El sistema creará una Hora Extra por Actividad con los datos proporcionados y con la clasificación de las Horas Extras a reportar, para que el Jefe Departamental correspondiente realice la debida Revisión y Aprobación de la misma.

Existe un módulo llamado “Hora Extra – Calcular Acumulado Horas Extras” que se presenta en la Figura 3.47, el cual se ejecuta en varios pasos del Proceso y es el encargado de calcular el acumulado de las Horas Extras del Empleado por mes.

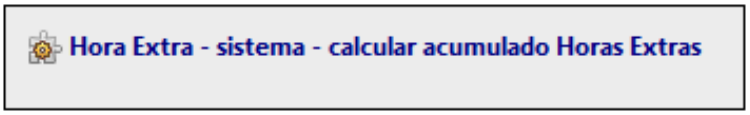

Figura 3.47 Módulo de Calcular el acumulado de las Horas Extras del Empleado por Mes

Cuando el Empleado Ingresa los Tiempos estimados o se vence el plazo del mismo se ejecuta este módulo el cual está diseñado de la siguiente manera:

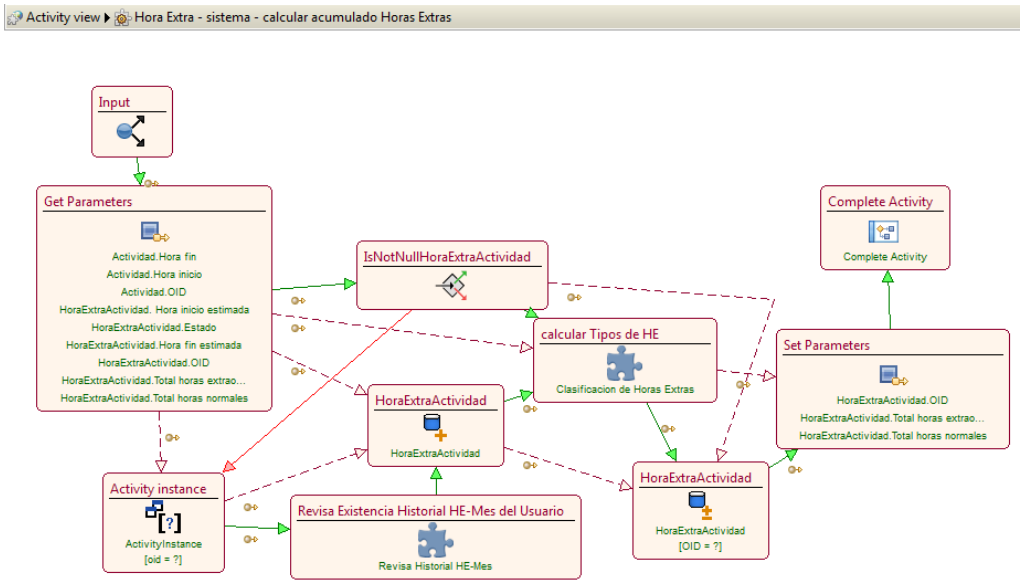

Figura 3.48 WebML del Módulo Calcular Acumulado Horas Extras

En la figura 3.48 observamos que se realiza la verificación de si existe o no la Hora Extra de la Actividad, si esta aún no existe (que sería por el caso en que se venció el plazo del ingreso de tiempos estimados), revisa la existencia del Historial de las Horas Extras del Mes del Empleado utilizando el Módulo antes visto (Figura 3.38), luego crea la Hora Extra de la Actividad para obtenida esta información calcular, clasificar y actualizar las Horas Extras que contiene la Actividad, haciendo uso del Módulo antes visto (Figura 3.45).

Por el contrario si existe la Hora Extra de la Actividad el módulo simplemente volverá a calcular, clasificar y actualizar las Horas Extras de la misma.

Siguiendo con la explicación del Diseño, ahora veremos el módulo que se presenta en la Figura 3.49, el cual se ejecuta al terminar el Ingreso de los Tiempos Estimados o cuando se vence el plazo del mismo.

Figura 3.49 Módulo de Distribución de las Horas Extras a los Jefes Departamentales

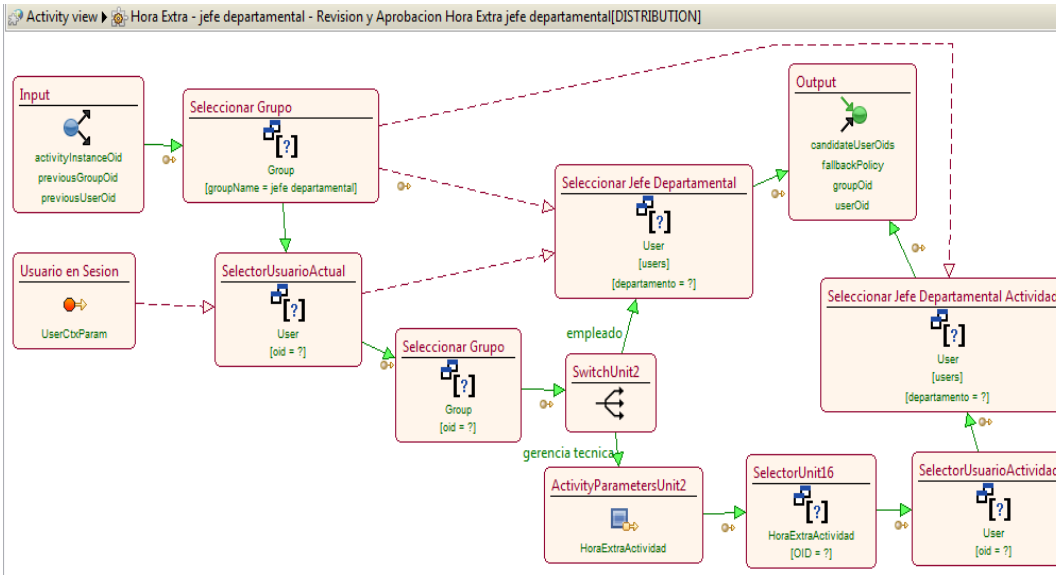

Figura 3.50 WebML del Módulo de Distribución de las Horas Extras al Jefe Departamental

El diseño del módulo “Hora Extra – Jefe Departamental – Revisión y Aprobación Hora Extra Jefe Departamental [DISTRIBUTION]” presentado en la Figura 3.50, es el que realiza la Distribución de las Horas Extras al Jefe Departamental correspondiente dependiendo del Departamento del Empleado para su debida Revisión y Autorización.

A continuación se detallara la Site View “Hora Extra – Jefe Departamental – Revisión y Aprobación Hora Extra”, la cual la podemos observar en la Figura 3.51.

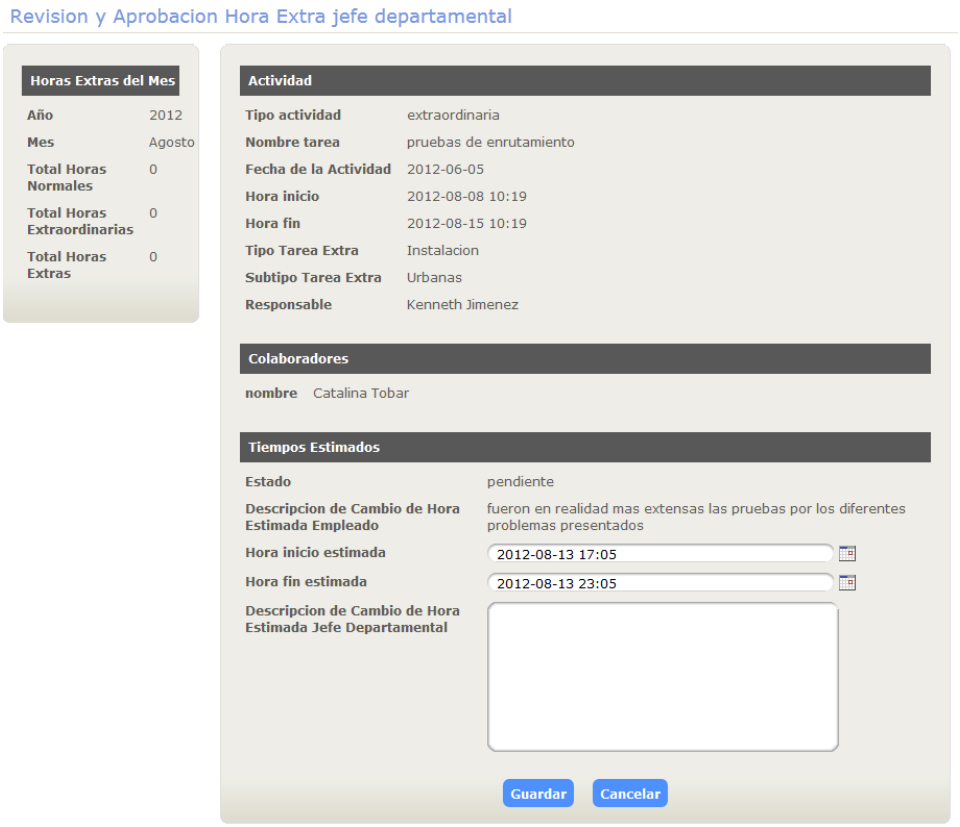

Figura 3.51 Página de Revisión y Aprobación Hora Extra Jefe Departamental

En esta Página el Jefe Departamental podrá modificar los tiempos estimados ingresados por el Empleado. En el caso de que el Empleado no haya ingresado los mismos, los tiempos estimados serán los mismos que se establecieron cuando fue ingresada la Actividad tal como se explicó anteriormente.

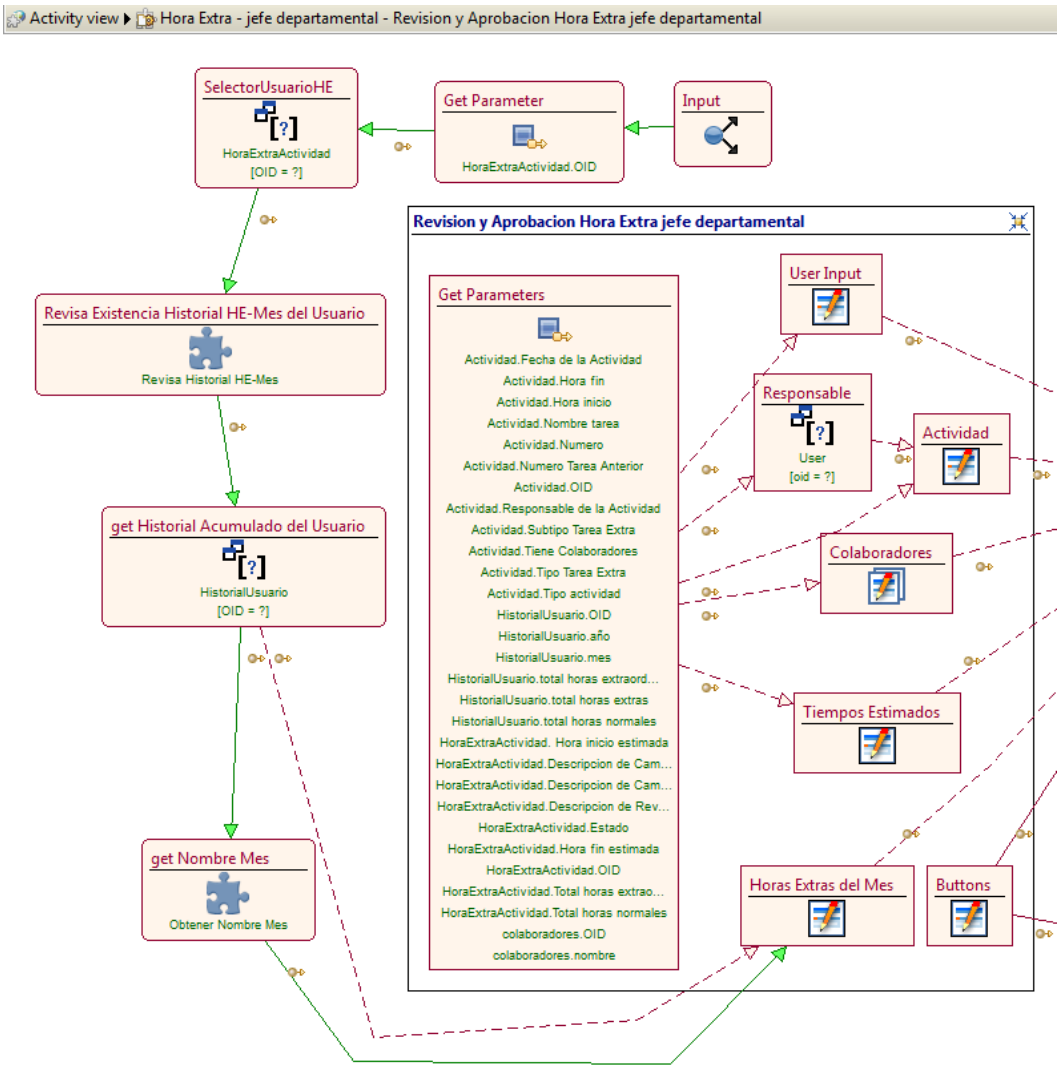

Figura 3.52 WebML del Módulo de Revisión y Aprobación Hora Extra Jefe Departamental

En la Figura 3.52 se observa el diseño de la site view “Hora Extra – Jefe Departamental – Revisión y Aprobación Hora Extra”, la cual contiene la misma verificación de existencia del Historial de las Horas Extras del mes del empleado involucrado en la Actividad a Revisar y Autorizar para poder mostrar en la *Entry Unit* correspondiente. Estas Units y Módulos que las vimos con anterioridad en las Figuras 3.37, 3.38, 3.39, 3.40, 3.41, 3.42.

Lo demás que podemos observar son *Entry Units* que son utilizadas para presentar información de la Actividad y de las Horas extras de la misma. Aquí el único campo obligatorio para el Jefe Departamental es la Descripción de su Autorización. Este también es uno de los pasos del Proceso en el cual se ejecuta el Módulo “Hora Extra – Calcular Acumulado Horas Extras” ver Figuras 3.47, 3.48, que es el encargado de calcular el acumulado de las Horas Extras del Empleado por mes.

Con respecto a la Revisión y Aprobación de las Horas Extras por parte de la Gerencia Técnica es muy similar a como se maneja tanto la Site View como el diseño, el cual veremos a continuación en la Figura 3.53.

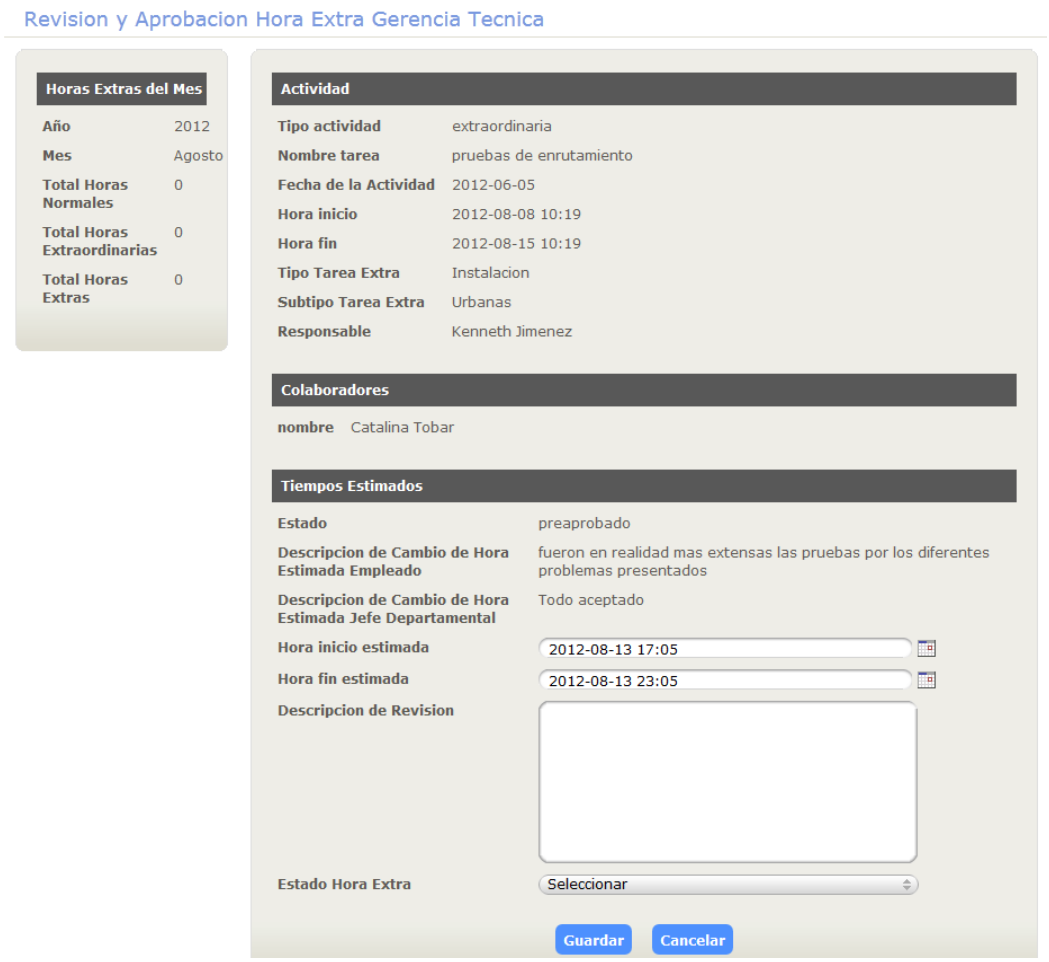

Figura 3.53 Página de Revisión y Aprobación Hora Extra Gerencia Técnica

En la página observamos que la Gerencia Técnica también puede modificar la hora inicio estimada y la hora fin estimada, y como campo obligatorio al igual que en la Revisión y Aprobación del Jefe Departamental, se encuentra la Descripción.

Lo adicional que podemos mencionar en estas Aprobaciones de Horas Extras de los empleados por parte de la Gerencia Técnica es que existe la opción de poder “Aprobar” o mandarlas

a “Revisión” las horas extras al Jefe Departamental. Este también es uno de los pasos del proceso donde se ejecuta el Módulo “Hora Extra – Calcular Acumulado Horas Extras” ver Figuras 3.46, 3.47, que es el encargado de calcular el acumulado de las Horas Extras del Empleado por mes.

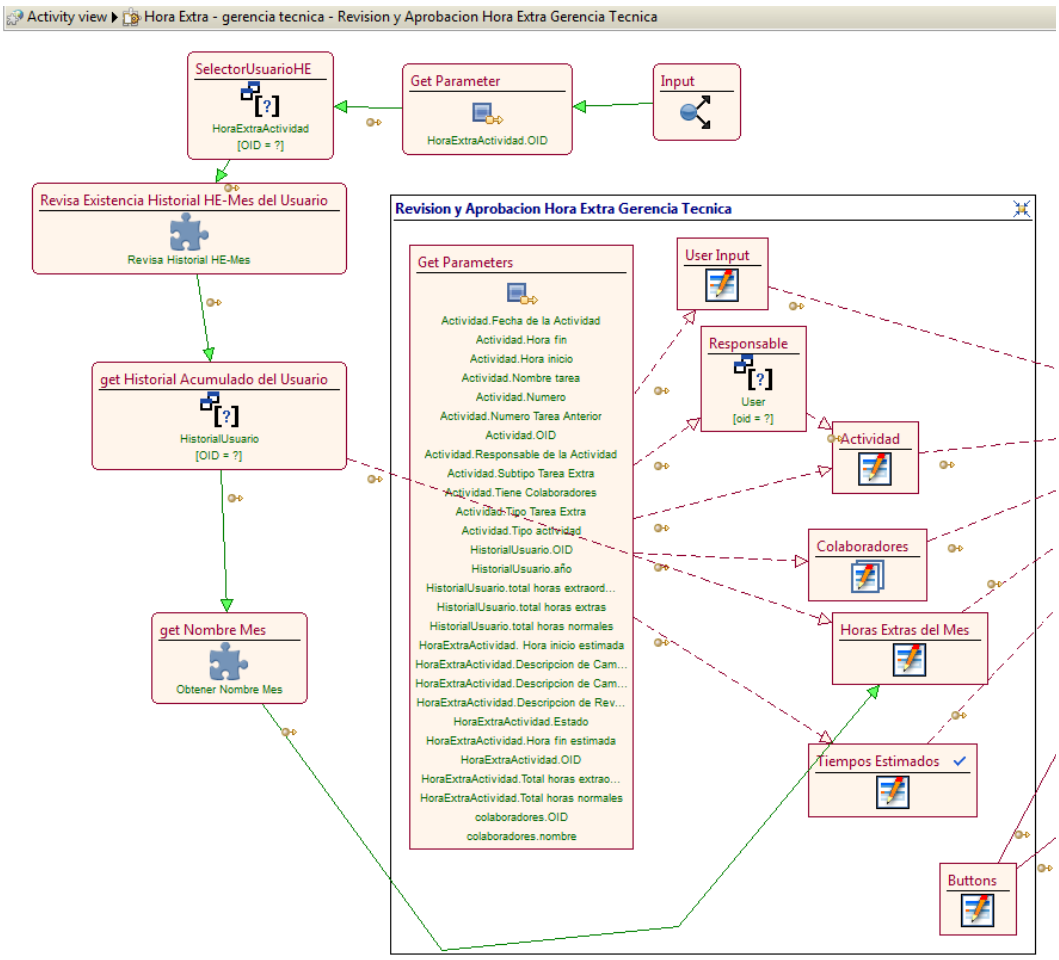

Figura 3.54 WebML de la Revisión y Aprobación de la Hora Extra de Gerencia Técnica

En la Figura 3.54 se observa que el diseño es muy similar al que se maneja en la site view “Hora Extra – Jefe Departamental – Revisión y Aprobación Hora Extra” ver Figura 3.51.

El ultimo módulo ejecutado en el Proceso de Aprobación de las Horas Extras luego que la Gerencia Técnica Aprueba es el “Hora Extra – Sistema – Listo para Nomina” presentado en la Figura 3.55.

Figura 3.55 Módulo de Listo para nomina

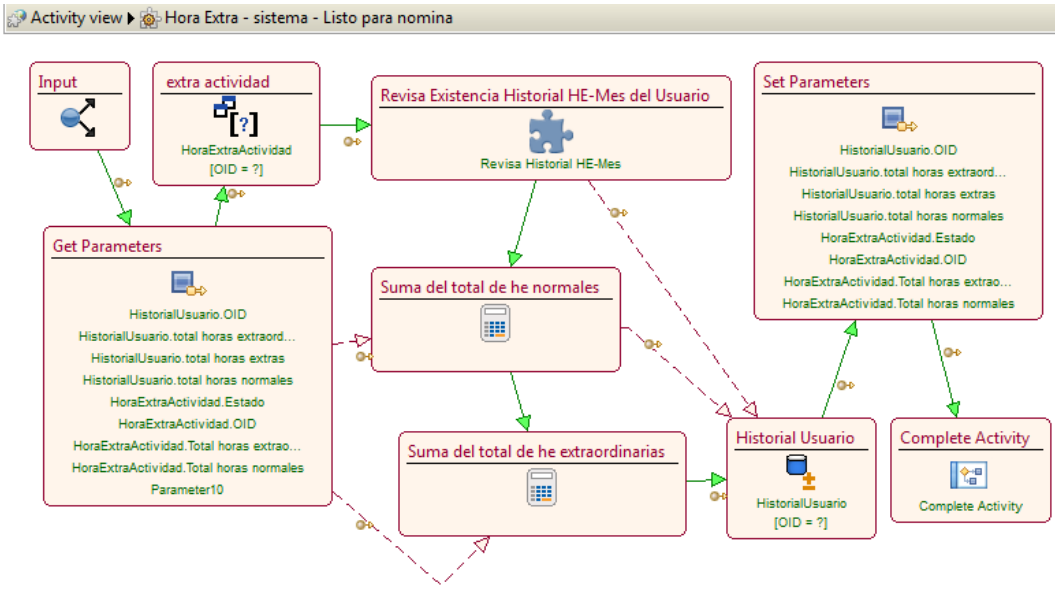

Figura 3.56 WebML del Módulo de Listo para Nomina

La figura 3.56 presenta el diseño del módulo Listo para Nómina. En este diseño se observa que se reutiliza el módulo que verifica el Historial de Horas Extras por mes del Empleado (ver Figuras 3.37, 3.38) y también la *Math Unit,* Unit que permite realizar la suma acumulada de las horas extras por tipo por mes y por año del empleado para luego hacer la actualización respectiva de la misma.

3.7 IMPLEMENTACIÓN DE LOS DATOS

La implementación de los datos del diagrama E-R, se realizó mediante un script SQL generado por la aplicación WebML. El script lo podemos encontrar en los Anexo D de la presente tesis.

3.8 INSTALACIÓN DEL HIPERTEXTO

Las páginas y unidades creadas en las site view del WebMl, se transforman en páginas web dinámicas JSP que se ejecutan sobre la arquitectura seleccionada.

Las páginas JSP son páginas web que contienen html y código java. Una JSP se compilla a un programa Java la primera vez que es invocada. Al invocarse se crea una clase java que se ejecutará como un servlet en el servidor, la cual es manejada por el motor de servlets. El motor carga la clase servlet y la ejecuta creando HTML

dinámico que se presentará en el navegador del usuario. En la Figura 3.57 podemos observar el proceso de ejecución de una página JSP.

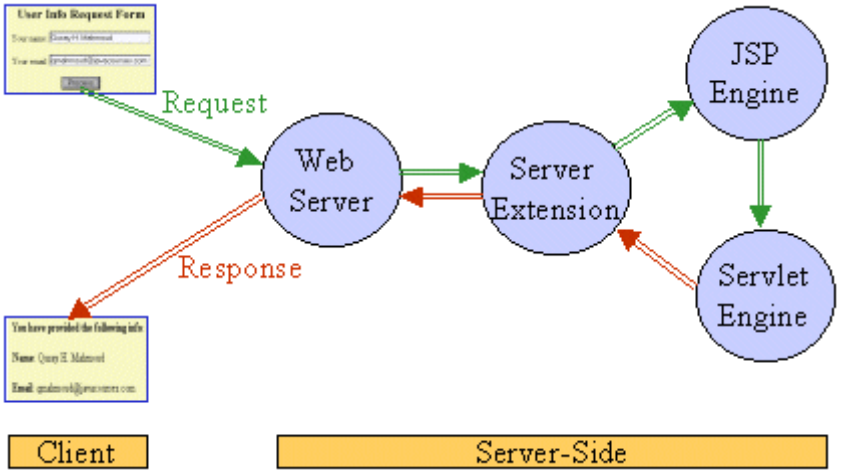

Figura 3.57 Proceso ejecución página JSP

Fuente: http://j2ee.ibsi.cl/desarrollo/java-j2ee/tecnologia-jsp-java-server-pages/

3.9 PRUEBA Y EVALUACIÓN

En esta etapa comprobamos que la aplicación logra realizar los requisitos funcionales y no funcionales satisfactoriamente. Debido a que la aplicación fue diseñada de forma modular, durante el desarrollo, se efectuaron pruebas evaluando cada módulo y corrigiéndolo según los requerimientos.

Al realizar las pruebas se hizo la presentación del primer demo del proceso de Aprobación de Horas Extras ante los usuarios, creado en base a los requisitos del sistema. Utilizando usuarios, información de

prueba y diseñando una interfaz en base a los aplicativos existentes en la empresa, se explicó paso a paso como se manejará la Aprobación de Horas extras en el sistema, indicando las opciones y datos que maneja cada usuario dependiendo de su rol: Empleado, Jefe Departamental y Gerente Técnico. Como resultado se obtuvo la inclusión de la opción para editar los tiempos de las tareas de cada empleado por parte de los Jefes Departamentales y Gerencia Técnica.

Nuevamente se realizó la presentación de un demo que contiene las observaciones y opciones solicitadas por los usuarios. Utilizando usuarios e información de prueba se explicó nuevamente como se manejará la Aprobación de Horas extras de los Empleados en el sistema, obteniendo como resultado incluir comentarios para las respectivas aprobaciones de los jefes departamentales y Gerencia Técnica.

Luego se realizó una nueva presentación de un demo donde se incluyen las observaciones y opciones solicitadas por los usuarios en la presentación anterior. Utilizando usuarios e información de prueba se explicó nuevamente como se manejará la Aprobación de Horas extras de los Empleados en el sistema, dando como resultado la aprobación del proceso, además de solicitar la modificación de la

página de inicio de cada usuario para que visualice las horas extras en lugar de ActivityInstance.

Finalmente se realizó una nueva presentación del demo del proceso de Aprobación de Horas Extras incluyendo todas las observaciones y opciones solicitadas por los usuarios anteriormente. Utilizando usuarios e información de prueba se explicó nuevamente paso a paso como se manejará la Aprobación de Horas extras de los Empleados en el sistema, indicando las opciones y datos que maneja cada usuario dependiendo de su rol: Empleado, Jefe Departamental y Gerente Técnico. Como resultado se obtuvo la Satisfacción y Aprobación total del sistema con respecto a la interfaz y el flujo del proceso de Aprobación de Horas Extras.

3.10 MANTENIMIENTO Y DESARROLLO

Una vez que la aplicación ha sido liberada, se podrán producir cambios tanto para corregir errores como para mejorarla. Estos cambios se realizarán en los modelos conceptuales de datos e hipertexto, procediendo luego a la implementación, logrando que los cambios sean efectivos. De esta manera, se mantiene el proceso de ingeniería de software usado para el desarrollo de la aplicación (MDD).

CONCLUSIONES

De este proyecto de tesis podemos concluir lo siguiente:

1. El desarrollo dirigido por modelos actualmente está tomando mucha fuerza, ya que permite que personas que no tengan el suficiente conocimiento sobre desarrollo de software, como los analistas y los expertos de negocios, puedan crear sus aplicaciones sin la ayuda de un experto informático permitiéndoles resolver sus problemas rápidamente.

2. El desarrollo del Sistema de Horas extras utilizando la herramienta MDD WebRatio fue óptimo y eficaz, ya que al sincronizar el lenguaje BPM a WebML obtenemos un modelo visual, potente e intuitivo formado por un conjunto de gráficos que representan los requisitos del sistema, los cuales se transforman automáticamente en una aplicación web completa y en funcionamiento. De esta manera podemos poner más atención a los requerimientos de los negocios, a los usuarios y a

la validación de los resultados, dejando las partes repetitivas y los detalles de implementación técnica a mecanismos automáticos.

3. Gracias a que WebRatio nos ofrece una rápida generación de prototipos del proyecto en cualquier momento del desarrollo, podemos comprobar y evaluar constantemente junto al usuario si el camino tomado es el correcto, con el fin de obtener una aplicación que cumpla con las expectativas del usuario. Esta ventaja es muy importante ya que como el proyecto se encuentra modularizado, realizar modificaciones es sencillo puesto que solo debemos cambiar las partes que lo necesiten.

4. Con WebRatio el correcto diseño e implementación de módulos mejora de forma significativa el tiempo de desarrollo de un sistema, otorgándole mayor flexibilidad, ya que cada parte del diseño (lógica de negocio, componentes de integración, partes del modelo, estilos de presentación) puede reutilizarse tanto en la aplicación web como en cualquier nuevo proyecto de manera inmediata.

5. Debido a la rapidez de desarrollo que brinda Webratio, podemos utilizar de manera óptima nuestros recursos para la creación de nuevas aplicaciones, en lugar de utilizarlos para el mantenimiento de las aplicaciones existentes.

6. Al utilizar WebRatio, en el modelado BPM no se pueden crear múltiples instancias de un Business Object, ya que si se declaran en la pool del proceso dos variables del mismo BO, al momento de realizar alguna acción sobre una de las variables, se verán afectadas las demás variables del mismo Business Object.

RECOMENDACIONES

Las recomendaciones que se pueden otorgar para este proyecto de graduación son:

1. Realizar el correcto levantamiento de requisitos teniendo en claro el proceso el cual se va automatizar y los actores que participan en el mismo, con el fin de lograr un modelado del proceso que cumpla a cabalidad con los requerimientos del negocio. De esta manera evitamos tener que volver a modelar el proceso, disminuyendo el tiempo de desarrollo.

2. Tener en claro los conceptos Fundamentales del manejo de Objetos y tipos de Datos, ayudará considerablemente a la ágil creación de los scripts en groovy que la aplicación web ejecutará para manipular la información.

3. Para desarrollar una aplicación se recomienda comprender el funcionamiento del lenguaje BPM y WebML, con el fin de realizar el

diseño de la aplicación de una manera rápida y eficaz, evitando gastar tiempo y esfuerzo en errores presentados.

4. Se recomienda durante toda la etapa de diseño y desarrollo involucrar a los usuarios que usarán la aplicación presentándoles los prototipos generados. Esto es muy importante ya que gracias a la retroalimentación brindada, mejoraremos el modelo hasta poder obtener el definitivo, con el cual lograremos generar una aplicación robusta y fiel a los requerimientos de los usuarios.

5. Durante la etapa de diseño se recomienda trabajar en el estilo de presentación del sistema requerido por los usuarios, además de utilizar usuarios reales para la autenticación y proceso. De esta manera obtendremos prototipos personalizados con los que el usuario podrá observar cómo quedará finalmente la aplicación.

BIBLIOGRAFÍA

[1] Jose Manuel Pérez, Francisco Ruiz, Mario Piattini. "*Model Driven Engineering Aplicado a Business Process Management*". [Artículo en línea]. Informe Técnico UCLM-TSI-002 Marzo 2007. Págs. 10-11

Fuente: http://www.uclm.es/dep/tsi/pdf/UCLM-TSI-002.pdf

[2] Wikipedia.org. *"Object Management Group"*

Fuente: http://es.wikipedia.org/wiki/Object_Management_Group

[3] Juan Bernardo Quintero, Raquel Anaya. "*MDA y el Papel de los Modelos en el Proceso de Desarrollo de Software*". [Artículo en línea]. Págs. 3-4

Fuente: http://revista.eia.edu.co/articulos8/Art.10.pdf

[4] WikiWebRatio. *"Getting started with WebRatio 6 BPM"*

Fuente:http://wiki.webratio.com/index.php/Getting_started_with_WebRatio_6_BPM

[5] María Valeria de Castro. "*Aproximación MDA para el Desarrollo Orientado a Servicios de Sistemas de Información Web: Del Modelo de Negocio al Modelo de Composición de Servicios Web*". [Artículo en línea]. Universidad Rey Juan Carlos, Departamento de Lenguajes y Sistemas Informáticos. Móstoles (Madrid), Marzo de 2007. Págs. 36-37

Fuente: http://es.scribd.com/doc/59013250/24/WebML-Web-Modeling-Language

[6] María José Escalona Cuaresma. "*Modelos y técnicas para la especificación y el análisis de la navegación en sistemas software*". [Artículo en línea]. Universidad de Sevilla, Departamento de Lenguajes y Sistemas Informáticos. Octubre 2004. Págs. 18-19
Fuente: http://www.lsi.us.es/docs/doctorado/tesis/tesis.pdf

[7] WebRatio.org
Fuente: http://www.webratio.com/

[8] WikiWebRatio. *"How to create a BAM Project"*
Fuente: http://wiki.webratio.com/index.php/How_to_create_a_BAM_Project

[9] WikiWebRatio. *"The derivation in the data model"*
Fuente:http://wiki.webratio.com/index.php/The_derivation_in_the_data_model

[10] WikiWebRatio. *"Getting started with the Script unit"*
Fuente:http://wiki.webratio.com/index.php/Getting_started_with_the_Script_unit

ANEXOS

ANEXO A

DISEÑO COMPLETO DEL PROCESO BPM

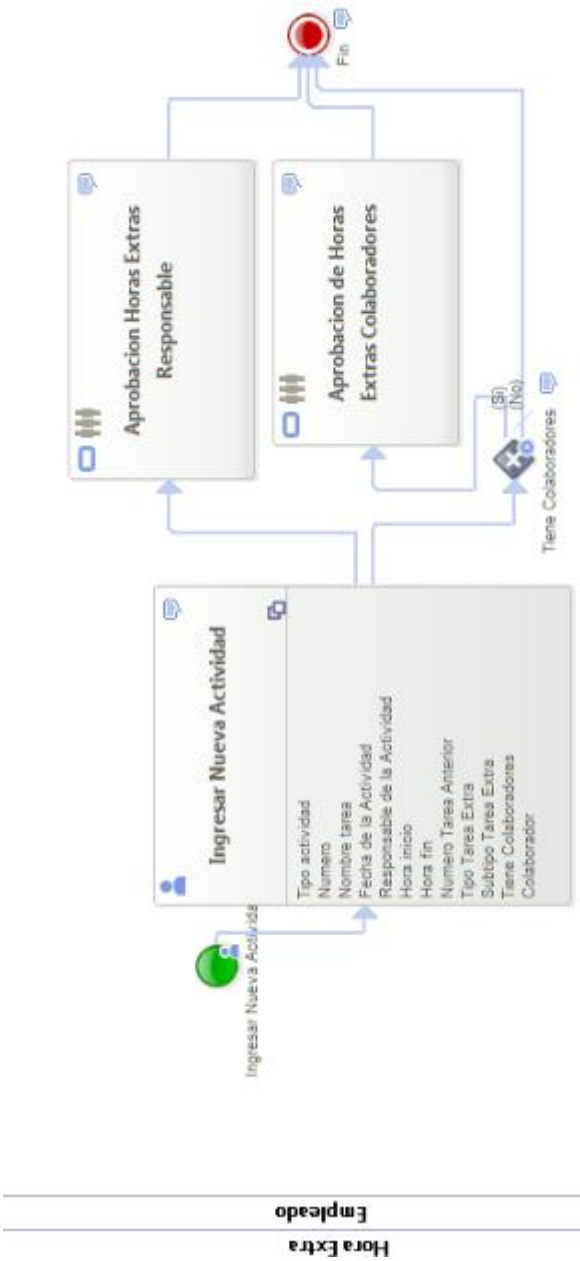

Figura A.1 Proceso Ingresar Nueva Actividad

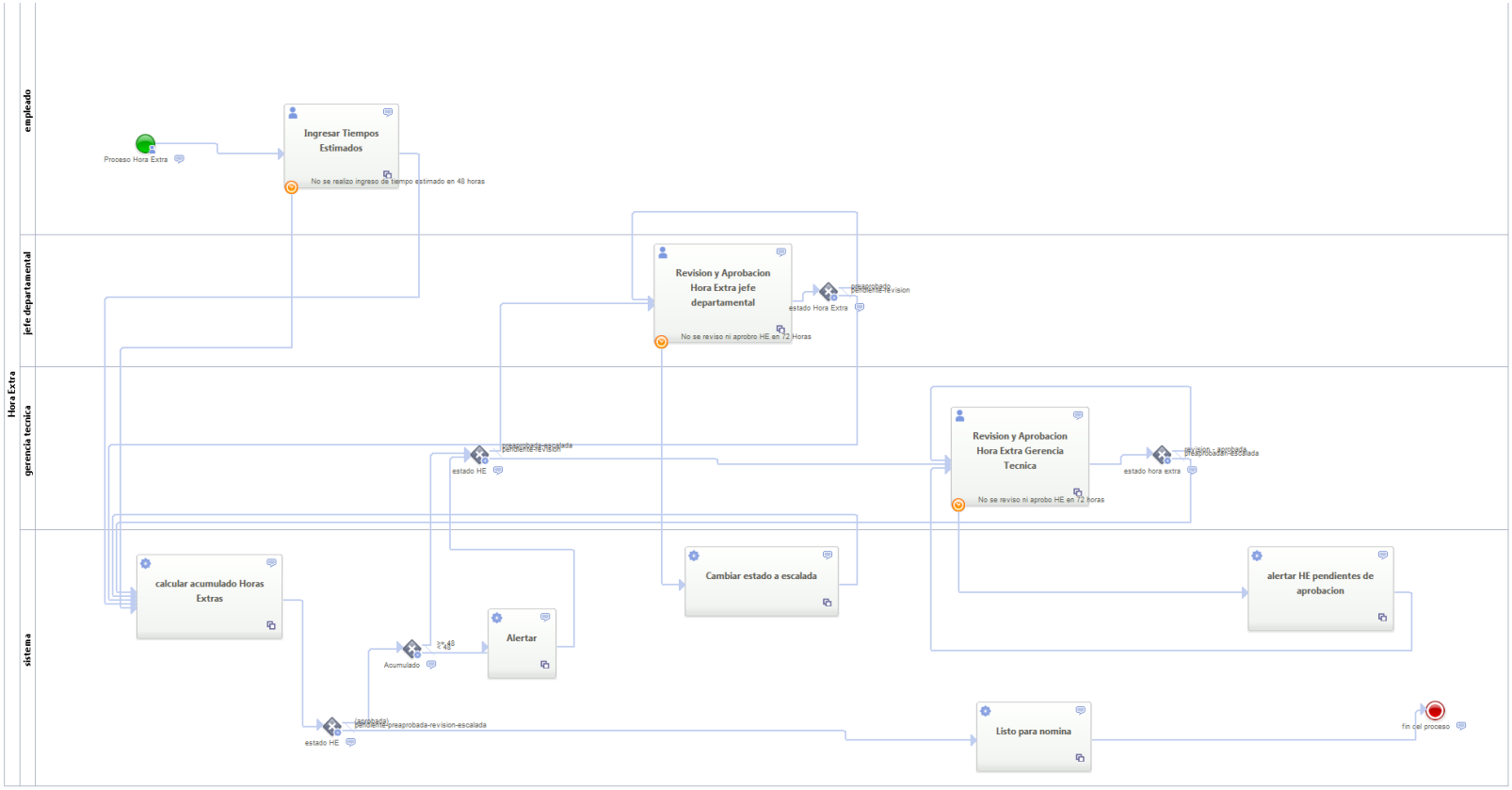

Figura A.2 Subproceso Aprobación Horas Extras

ANEXO B

CÓDIGO SCRIPT UNIT GET DÍA DEL MES

```
import java.text.SimpleDateFormat

def fecha = fecha.toString()
def dayOfMonth = new SimpleDateFormat('EEEEE').format(new
SimpleDateFormat('yyyy-MM-dd HH:mm:ss').parse(fecha))

return dayOfMonth;
```

ANEXO C

CÓDIGO SCRIPT UNIT CALCULA HORAS EXTRAS ENTRE FECHAS

```
import java.text.SimpleDateFormat
import java.math.MathContext
import java.math.RoundingMode
import java.util.*

#output def total_he = 0
#output def total_he_normales = 0
#output def total_he_extraordinarias = 0

Date d1 = new SimpleDateFormat("yyyy-MM-dd
H:mm:ss").parse(fecha_inicio)
Date d2 = new SimpleDateFormat("yyyy-MM-dd
H:mm:ss").parse(fecha_fin)

Date dcero = new SimpleDateFormat("yyyy-MM-dd").parse(fecha_inicio)
def date_cero = dcero + 1

Date date_siete = d2.clone()
date_siete.setHours(07)
date_siete.setMinutes(00)
date_siete.setSeconds(00)


def hoursBetweenIniCero = (date_cero.time - d1.time) / (60 * 60 *
1000)
def hoursBetweenCeroFin = (d2.time - date_cero.time) / (60 * 60 *
1000)

def hoursBetweenIniSiete = (date_siete.time - d1.time) / (60 * 60 *
1000)
def hoursBetweenFinSiete = (date_siete.time - d2.time) / (60 * 60 *
1000)
def hoursBetweenSieteFin = (d2.time - date_siete.time) / (60 * 60 *
1000)

def hoursBetweenIniFin = (d2.time - d1.time) / (60 * 60 * 1000)

total_he = hoursBetweenIniFin.round(new MathContext(3));
def finde_ini = 0
def finde_fin = 0
```

```
if(dia_inicio.compareTo("Saturday")==0 ||
dia_inicio.compareTo("Sunday")==0)
      finde_ini = 1

if(dia_fin.compareTo("Saturday")==0 ||
dia_fin.compareTo("Sunday")==0)
      finde_fin = 1

if( finde_ini && finde_fin  ){
      return ["total_he":total_he
      ,"total_he_normales":"0","total_he_extraordinarias":hoursBetwee
      nIniFin]
}

if(finde_ini == 1 && finde_fin==0){
      if(hoursBetweenIniCero > 0 && hoursBetweenCeroFin > 0){
            total_he_normales =  0
            total_he_extraordinarias = hoursBetweenCeroFin.round(new
            MathContext(3)) + hoursBetweenIniCero.round(new
            MathContext(3));

            return ["total_he":total_he
            ,"total_he_normales":total_he_normales,"total_he_extraord
            inarias":total_he_extraordinarias]
      }
}

if(hoursBetweenIniCero > 0 && hoursBetweenCeroFin > 0){
      total_he_normales =  hoursBetweenIniCero.round(new
      MathContext(3));
      total_he_extraordinarias = hoursBetweenCeroFin.round(new
      MathContext(3));

      return ["total_he":total_he
      ,"total_he_normales":total_he_normales,"total_he_extraordinaria
      s":total_he_extraordinarias]
}

if(hoursBetweenIniSiete > 0 && hoursBetweenSieteFin > 0){
      total_he_normales =  hoursBetweenSieteFin.round(new
      MathContext(3));
      total_he_extraordinarias = hoursBetweenIniSiete.round(new
      MathContext(3));

      return ["total_he":total_he
      ,"total_he_normales":total_he_normales,"total_he_extraordinaria
      s":total_he_extraordinarias]
```

```
}

if(hoursBetweenIniSiete > 0 && hoursBetweenFinSiete > 0){
      total_he_normales =  0
      total_he_extraordinarias = hoursBetweenIniFin.round(new
      MathContext(3));

      return ["total_he":total_he
      ,"total_he_normales":total_he_normales,"total_he_extraordinaria
      s":total_he_extraordinarias]
}

total_he_normales = hoursBetweenIniFin.round(new MathContext(3));
total_he_extraordinarias = 0

return ["total_he":total_he
,"total_he_normales":total_he_normales,"total_he_extraordinarias":to
tal_he_extraordinarias]
```

ANEXO D

SCRIPT SQL PARA IMPLEMENTACIÓN DE LOS DATOS

```
-- ActivityInstance [ActivityInstance]
create table `activityinstance` (
   `oid`  integer  not null,
   `aborted_at`  datetime,
   `active_since`  datetime,
   `cancelled_at`  datetime,
   `completed_at`  datetime,
   `context`  varchar(255),
   `eventid`  varchar(255),
   `worked_at`  datetime,
   `name`  varchar(255),
   `ready_since`  datetime,
   `rollbackable`  bit,
   `status`  varchar(255),
  primary key (`oid`)
);


-- ActivityType [ActivityType]
create table `activitytype` (
   `oid`  integer  not null,
   `code`  varchar(255),
   `default_instance_name`  varchar(255),
   `description`  longtext,
   `execution`  varchar(255),
   `id`  varchar(255),
   `name`  varchar(255),
   `sort_number`  double precision,
   `type`  varchar(255),
   `uuid`  varchar(255),
  primary key (`oid`)
);


-- Attachment [Attachment]
create table `attachment` (
   `oid`  integer  not null,
   `file`  varchar(255),
   `timestamp`  datetime,
   `title`  varchar(255),
```

```
  primary key (`oid`)
);


-- Group [Group]
create table `group` (
   `oid`  integer  not null,
   `groupname`  varchar(255),
  primary key (`oid`)
);


-- Lock [Lock]
create table `lock` (
   `lockname`  varchar(255)  not null,
   `timestamp`  datetime,
  primary key (`lockname`)
);


-- Module [Module]
create table `module` (
   `oid`  integer  not null,
   `moduleid`  varchar(255),
   `modulename`  varchar(255),
  primary key (`oid`)
);


-- Note [Note]
create table `note` (
   `oid`  integer  not null,
   `text`  longtext,
   `timestamp`  datetime,
  primary key (`oid`)
);


-- Notification [Notification]
create table `notification` (
   `oid`  integer  not null,
   `message`  varchar(255),
   `read`  bit,
  primary key (`oid`)
);
```

```
-- ParameterInstance [ParameterInstance]
create table `parameterinstance` (
   `oid`  integer  not null,
   `current`  bit,
   `timestamp`  datetime,
   `value`  varchar(255),
  primary key (`oid`)
);


-- ParameterType [ParameterType]
create table `parametertype` (
   `oid`  integer  not null,
   `description`  varchar(255),
   `name`  varchar(255),
   `type`  varchar(255),
  primary key (`oid`)
);


-- Process [Process]
create table `process` (
   `oid`  integer  not null,
   `code`  varchar(255),
   `default_instance_name`  varchar(255),
   `description`  longtext,
   `name`  varchar(255),
   `uuid`  varchar(255),
   `version`  varchar(255),
  primary key (`oid`)
);


-- ProcessInstance [ProcessInstance]
create table `processinstance` (
   `oid`  integer  not null,
   `aborted_at`  datetime,
   `active_since`  datetime,
   `cancelled_at`  datetime,
   `completed_at`  datetime,
   `name`  varchar(255),
   `status`  varchar(255),
   `subprocessindex`  integer,
  primary key (`oid`)
);
```

```
-- SequenceFlow [SequenceFlow]
create table `sequenceflow` (
   `oid`  integer  not null,
   `condition`  varchar(255),
   `default`  bit,
   `label`  varchar(255),
   `sort_number`  double precision,
  primary key (`oid`)
);


-- User [User]
create table `user` (
   `oid`  integer  not null,
   `email`  varchar(255),
   `password`  varchar(255),
   `nombres`  varchar(255),
   `apellidos`  varchar(255),
   `departamento`  varchar(255),
   `username`  varchar(255),
  primary key (`oid`)
);


-- HistorialUsuario [ent1]
create table `historialusuario` (
   `oid`  integer  not null,
   `a_o`  varchar(255),
   `mes`  integer,
   `total_horas_normales`  decimal(19,2),
   `total_horas_extraordinarias`  decimal(19,2),
  primary key (`oid`)
);


-- Colaborador [ent2]
create table `colaborador` (
   `oid`  integer  not null,
  primary key (`oid`)
);


-- actividad [ent3]
create table `actividad` (
   `oid`  integer  not null,
   `tipo_actividad`  varchar(255),
   `numero`  integer,
```

```
   `nombre_tarea`  varchar(255),
   `fecha_de_la_actividad`  date,
   `hora_inicio`  datetime,
   `hora_fin`  datetime,
   `tipo_tarea_extra`  varchar(255),
   `subtipo_tarea_extra`  varchar(255),
   `numero_tarea_anterior`  integer,
   `tiene_colaboradores`  varchar(255),
  primary key (`oid`)
);


-- HoraExtraActividad [ent4]
create table `horaextraactividad` (
   `oid`  integer  not null,
   `hora_inicio_estimada`  datetime,
   `hora_fin_estimada`  datetime,
   `estado`  varchar(255),
   `descripcion_de_cambio_de_hora`  longtext,
   `descripcion_de_cambio_de_hor_2`  longtext,
   `descripcion_de_revision`  longtext,
   `total_horas_normales`  decimal(19,2),
   `total_horas_extraordinarias`  decimal(19,2),
  primary key (`oid`)
);


-- Asignacion [ent5]
create table `asignacion` (
   `oid`  integer  not null,
  primary key (`oid`)
);


-- ActivityInstance_CandidateUser [ActivityInstance_CandidateUser]
create table `activityinstance_candidateuser` (
   `activityinstance_oid`  integer not null,
   `user_oid`  integer not null,
  primary key (`activityinstance_oid`, `user_oid`)
);
alter table `activityinstance_candidateuser`   add index
fk_activityinstance_candidateu (`activityinstance_oid`), add
constraint fk_activityinstance_candidateu foreign key
(`activityinstance_oid`) references `activityinstance` (`oid`);
alter table `activityinstance_candidateuser`   add index
fk_activityinstance_candidat_2 (`user_oid`), add constraint
```

```
fk_activityinstance_candidat_2 foreign key (`user_oid`) references
`user` (`oid`);
create index `idx_activityinstance_candidate` on
`activityinstance_candidateuser`(`activityinstance_oid`);
create index `idx_activityinstance_candida_2` on
`activityinstance_candidateuser`(`user_oid`);


-- ActivityInstance_Group [ActivityInstance_Group]
alter table `activityinstance`  add column  `group_oid`  integer;
alter table `activityinstance`   add index fk_activityinstance_group
(`group_oid`), add constraint fk_activityinstance_group foreign key
(`group_oid`) references `group` (`oid`);
create index `idx_activityinstance_group` on
`activityinstance`(`group_oid`);


-- ActivityInstance_NextActivityInstanceNotes
[ActivityInstance_NextActivityInstanceNotes]
alter table `note`  add column  `activityinstance_oid`  integer;
alter table `note`   add index fk_note_activityinstance
(`activityinstance_oid`), add constraint fk_note_activityinstance
foreign key (`activityinstance_oid`) references `activityinstance`
(`oid`);
create index `idx_note_activityinstance` on
`note`(`activityinstance_oid`);


-- ActivityInstance_Notification [ActivityInstance_Notifications]
alter table `notification`  add column  `activityinstance_oid`
integer;
alter table `notification`   add index
fk_notification_activityinstan (`activityinstance_oid`), add
constraint fk_notification_activityinstan foreign key
(`activityinstance_oid`) references `activityinstance` (`oid`);
create index `idx_notification_activityinsta` on
`notification`(`activityinstance_oid`);


-- ActivityType_ActivityInstance [ActivityType_ActivityInstance]
alter table `activityinstance`  add column  `activitytype_oid`
integer;
alter table `activityinstance`   add index
fk_activityinstance_activityty (`activitytype_oid`), add constraint
fk_activityinstance_activityty foreign key (`activitytype_oid`)
references `activitytype` (`oid`);
```

```
create index `idx_activityinstance_activityt` on
`activityinstance`(`activitytype_oid`);


-- Attachment_ProcessInstance [Attachment_ProcessInstance]
create table `attachment_processinstance` (
   `attachment_oid`  integer not null,
   `processinstance_oid`  integer not null,
  primary key (`attachment_oid`, `processinstance_oid`)
);
alter table `attachment_processinstance`   add index
fk_attachment_processinstance (`attachment_oid`), add constraint
fk_attachment_processinstance foreign key (`attachment_oid`)
references `attachment` (`oid`);
alter table `attachment_processinstance`   add index
fk_attachment_processinstanc_2 (`processinstance_oid`), add
constraint fk_attachment_processinstanc_2 foreign key
(`processinstance_oid`) references `processinstance` (`oid`);
create index `idx_attachment_processinstance` on
`attachment_processinstance`(`attachment_oid`);
create index `idx_attachment_processinstan_2` on
`attachment_processinstance`(`processinstance_oid`);


-- Attachment_User [Attachment_User]
alter table `attachment`  add column  `user_oid`  integer;
alter table `attachment`   add index fk_attachment_user
(`user_oid`), add constraint fk_attachment_user foreign key
(`user_oid`) references `user` (`oid`);
create index `idx_attachment_user` on `attachment`(`user_oid`);


-- Group_DefaultModule [Group2DefaultModule_DefaultModule2Group]
alter table `group`  add column  `module_oid`  integer;
alter table `group`   add index fk_group_module (`module_oid`), add
constraint fk_group_module foreign key (`module_oid`) references
`module` (`oid`);
create index `idx_group_module` on `group`(`module_oid`);


-- Group_Module [Group2Module_Module2Group]
create table `group_module` (
   `group_oid`  integer not null,
   `module_oid`  integer not null,
  primary key (`group_oid`, `module_oid`)
);
```

```
alter table `group_module`   add index fk_group_module_group
(`group_oid`), add constraint fk_group_module_group foreign key
(`group_oid`) references `group` (`oid`);
alter table `group_module`   add index fk_group_module_module
(`module_oid`), add constraint fk_group_module_module foreign key
(`module_oid`) references `module` (`oid`);
create index `idx_group_module_group` on
`group_module`(`group_oid`);
create index `idx_group_module_module` on
`group_module`(`module_oid`);


-- Group_ActivityType [Group_ActivityType]
create table `group_activitytype` (
   `group_oid`  integer not null,
   `activitytype_oid`  integer not null,
  primary key (`group_oid`, `activitytype_oid`)
);
alter table `group_activitytype`   add index
fk_group_activitytype_group (`group_oid`), add constraint
fk_group_activitytype_group foreign key (`group_oid`) references
`group` (`oid`);
alter table `group_activitytype`   add index
fk_group_activitytype_activity (`activitytype_oid`), add constraint
fk_group_activitytype_activity foreign key (`activitytype_oid`)
references `activitytype` (`oid`);
create index `idx_group_activitytype_group` on
`group_activitytype`(`group_oid`);
create index `idx_group_activitytype_activit` on
`group_activitytype`(`activitytype_oid`);


-- NextActivityType_PreviousSequenceFlow
[NextActivityType_PreviousSequenceFlow]
alter table `sequenceflow`  add column  `activitytype_oid`  integer;
alter table `sequenceflow`   add index fk_sequenceflow_activitytype
(`activitytype_oid`), add constraint fk_sequenceflow_activitytype
foreign key (`activitytype_oid`) references `activitytype` (`oid`);
create index `idx_sequenceflow_activitytype` on
`sequenceflow`(`activitytype_oid`);


-- Note_ActivityInstance [Note_ActivityInstance]
create table `note_activityinstance` (
   `note_oid`  integer not null,
   `activityinstance_oid`  integer not null,
  primary key (`note_oid`, `activityinstance_oid`)
```

```
);
alter table `note_activityinstance`   add index
fk_note_activityinstance_note (`note_oid`), add constraint
fk_note_activityinstance_note foreign key (`note_oid`) references
`note` (`oid`);
alter table `note_activityinstance`   add index
fk_note_activityinstance_activ (`activityinstance_oid`), add
constraint fk_note_activityinstance_activ foreign key
(`activityinstance_oid`) references `activityinstance` (`oid`);
create index `idx_note_activityinstance_note` on
`note_activityinstance`(`note_oid`);
create index `idx_note_activityinstance_acti` on
`note_activityinstance`(`activityinstance_oid`);


-- Note_ProcessInstance [Note_ProcessInstance]
create table `note_processinstance` (
   `note_oid`  integer not null,
   `processinstance_oid`  integer not null,
  primary key (`note_oid`, `processinstance_oid`)
);
alter table `note_processinstance`   add index
fk_note_processinstance_note (`note_oid`), add constraint
fk_note_processinstance_note foreign key (`note_oid`) references
`note` (`oid`);
alter table `note_processinstance`   add index
fk_note_processinstance_proces (`processinstance_oid`), add
constraint fk_note_processinstance_proces foreign key
(`processinstance_oid`) references `processinstance` (`oid`);
create index `idx_note_processinstance_note` on
`note_processinstance`(`note_oid`);
create index `idx_note_processinstance_proce` on
`note_processinstance`(`processinstance_oid`);


-- Note_User [Note_User]
alter table `note`  add column  `user_oid`  integer;
alter table `note`   add index fk_note_user (`user_oid`), add
constraint fk_note_user foreign key (`user_oid`) references `user`
(`oid`);
create index `idx_note_user` on `note`(`user_oid`);


-- ParameterInstance_ActivityInstance
[ParameterInstance_ActivityInstance]
alter table `parameterinstance`  add column  `activityinstance_oid`
integer;
```

```
alter table `parameterinstance`   add index
fk_parameterinstance_activityi (`activityinstance_oid`), add
constraint fk_parameterinstance_activityi foreign key
(`activityinstance_oid`) references `activityinstance` (`oid`);
create index `idx_parameterinstance_activity` on
`parameterinstance`(`activityinstance_oid`);


-- ParameterType_ParameterInstance [ParameterType_ParameterInstance]
alter table `parameterinstance`  add column  `parametertype_oid`
integer;
alter table `parameterinstance`   add index
fk_parameterinstance_parameter (`parametertype_oid`), add constraint
fk_parameterinstance_parameter foreign key (`parametertype_oid`)
references `parametertype` (`oid`);
create index `idx_parameterinstance_paramete` on
`parameterinstance`(`parametertype_oid`);


-- PreviousActivityInstance_NextActivityInstance
[PreviousActivityInstance_NextActivityInstance]
create table `previousactivityinstance_nexta` (
   `activityinstance_oid_2`  integer not null,
   `activityinstance_oid`  integer not null,
  primary key (`activityinstance_oid_2`, `activityinstance_oid`)
);
alter table `previousactivityinstance_nexta`   add index
fk_previousactivityinstance_ne (`activityinstance_oid_2`), add
constraint fk_previousactivityinstance_ne foreign key
(`activityinstance_oid_2`) references `activityinstance` (`oid`);
alter table `previousactivityinstance_nexta`   add index
fk_previousactivityinstance_2 (`activityinstance_oid`), add
constraint fk_previousactivityinstance_2 foreign key
(`activityinstance_oid`) references `activityinstance` (`oid`);
create index `idx_previousactivityinstance_n` on
`previousactivityinstance_nexta`(`activityinstance_oid_2`);
create index `idx_previousactivityinstance_2` on
`previousactivityinstance_nexta`(`activityinstance_oid`);


-- PreviousActivityType_NextSequenceFlow
[PreviousActivityType_NextSequenceFlow]
alter table `sequenceflow`  add column  `activitytype_oid_2`
integer;
alter table `sequenceflow`   add index
fk_sequenceflow_activitytype_2 (`activitytype_oid_2`), add
```

```
constraint fk_sequenceflow_activitytype_2 foreign key
(`activitytype_oid_2`) references `activitytype` (`oid`);
create index `idx_sequenceflow_activitytyp_2` on
`sequenceflow`(`activitytype_oid_2`);


-- ProcessInstance_ActivityInstance
[ProcessInstance_ActivityInstance]
alter table `activityinstance`  add column  `processinstance_oid`
integer;
alter table `activityinstance`   add index
fk_activityinstance_processins (`processinstance_oid`), add
constraint fk_activityinstance_processins foreign key
(`processinstance_oid`) references `processinstance` (`oid`);
create index `idx_activityinstance_processin` on
`activityinstance`(`processinstance_oid`);


-- ProcessInstance_ParameterInstance
[ProcessInstance_ParameterInstance]
alter table `parameterinstance`  add column  `processinstance_oid`
integer;
alter table `parameterinstance`   add index
fk_parameterinstance_processin (`processinstance_oid`), add
constraint fk_parameterinstance_processin foreign key
(`processinstance_oid`) references `processinstance` (`oid`);
create index `idx_parameterinstance_processi` on
`parameterinstance`(`processinstance_oid`);


-- ProcessInstance_Process [ProcessInstance_Process]
alter table `processinstance`  add column  `process_oid`  integer;
alter table `processinstance`   add index fk_processinstance_process
(`process_oid`), add constraint fk_processinstance_process foreign
key (`process_oid`) references `process` (`oid`);
create index `idx_processinstance_process` on
`processinstance`(`process_oid`);


-- Process_ActivityType [Process_ActivityType]
alter table `activitytype`  add column  `process_oid`  integer;
alter table `activitytype`   add index fk_activitytype_process
(`process_oid`), add constraint fk_activitytype_process foreign key
(`process_oid`) references `process` (`oid`);
create index `idx_activitytype_process` on
`activitytype`(`process_oid`);
```

```
-- Process_ParameterType [Process_ParameterType]
alter table `parametertype`  add column  `process_oid`  integer;
alter table `parametertype`   add index fk_parametertype_process
(`process_oid`), add constraint fk_parametertype_process foreign key
(`process_oid`) references `process` (`oid`);
create index `idx_parametertype_process` on
`parametertype`(`process_oid`);


-- SubprocessInstances_ParentProcessInstance
[SubprocessInstances_ParentProcessInstance]
alter table `processinstance`  add column  `processinstance_oid`
integer;
alter table `processinstance`   add index
fk_processinstance_processinst (`processinstance_oid`), add
constraint fk_processinstance_processinst foreign key
(`processinstance_oid`) references `processinstance` (`oid`);
create index `idx_processinstance_processins` on
`processinstance`(`processinstance_oid`);


-- User_DefaultGroup [User2DefaultGroup_DefaultGroup2User]
alter table `user`  add column  `group_oid`  integer;
alter table `user`   add index fk_user_group (`group_oid`), add
constraint fk_user_group foreign key (`group_oid`) references
`group` (`oid`);
create index `idx_user_group` on `user`(`group_oid`);


-- User_Group [User2Group_Group2User]
create table `user_group` (
   `user_oid`  integer not null,
   `group_oid`  integer not null,
  primary key (`user_oid`, `group_oid`)
);
alter table `user_group`   add index fk_user_group_user
(`user_oid`), add constraint fk_user_group_user foreign key
(`user_oid`) references `user` (`oid`);
alter table `user_group`   add index fk_user_group_group
(`group_oid`), add constraint fk_user_group_group foreign key
(`group_oid`) references `group` (`oid`);
create index `idx_user_group_user` on `user_group`(`user_oid`);
create index `idx_user_group_group` on `user_group`(`group_oid`);


-- User_ActivityInstance [User_ActivityInstance]
```

```
alter table `activityinstance`  add column  `user_oid`  integer;
alter table `activityinstance`   add index fk_activityinstance_user
(`user_oid`), add constraint fk_activityinstance_user foreign key
(`user_oid`) references `user` (`oid`);
create index `idx_activityinstance_user` on
`activityinstance`(`user_oid`);


-- User_Notifications [User_Notifications]
create table `user_notifications` (
   `user_oid`  integer not null,
   `notification_oid`  integer not null,
  primary key (`user_oid`, `notification_oid`)
);
alter table `user_notifications`   add index
fk_user_notifications_user (`user_oid`), add constraint
fk_user_notifications_user foreign key (`user_oid`) references
`user` (`oid`);
alter table `user_notifications`   add index
fk_user_notifications_notifica (`notification_oid`), add constraint
fk_user_notifications_notifica foreign key (`notification_oid`)
references `notification` (`oid`);
create index `idx_user_notifications_user` on
`user_notifications`(`user_oid`);
create index `idx_user_notifications_notific` on
`user_notifications`(`notification_oid`);


-- User_actividad [rel1]
alter table `actividad`  add column  `user_oid`  integer;
alter table `actividad`   add index fk_actividad_user (`user_oid`),
add constraint fk_actividad_user foreign key (`user_oid`) references
`user` (`oid`);
create index `idx_actividad_user` on `actividad`(`user_oid`);


-- User_Asignacion [rel10]
alter table `asignacion`  add column  `user_oid`  integer;
alter table `asignacion`   add index fk_asignacion_user
(`user_oid`), add constraint fk_asignacion_user foreign key
(`user_oid`) references `user` (`oid`);
create index `idx_asignacion_user` on `asignacion`(`user_oid`);


-- actividad_Colaborador [rel2]
alter table `colaborador`  add column  `actividad_oid`  integer;
```

```
alter table `colaborador`   add index fk_colaborador_actividad
(`actividad_oid`), add constraint fk_colaborador_actividad foreign
key (`actividad_oid`) references `actividad` (`oid`);
create index `idx_colaborador_actividad` on
`colaborador`(`actividad_oid`);


-- Colaborador_User [rel3]
alter table `colaborador`  add column  `user_oid`  integer;
alter table `colaborador`   add index fk_colaborador_user
(`user_oid`), add constraint fk_colaborador_user foreign key
(`user_oid`) references `user` (`oid`);
create index `idx_colaborador_user` on `colaborador`(`user_oid`);


-- HistorialUsuario_User [rel4]
alter table `historialusuario`  add column  `user_oid`  integer;
alter table `historialusuario`   add index fk_historialusuario_user
(`user_oid`), add constraint fk_historialusuario_user foreign key
(`user_oid`) references `user` (`oid`);
create index `idx_historialusuario_user` on
`historialusuario`(`user_oid`);


-- HoraExtraActividad_User [rel5]
alter table `horaextraactividad`  add column  `user_oid`  integer;
alter table `horaextraactividad`   add index
fk_horaextraactividad_user (`user_oid`), add constraint
fk_horaextraactividad_user foreign key (`user_oid`) references
`user` (`oid`);
create index `idx_horaextraactividad_user` on
`horaextraactividad`(`user_oid`);


-- actividad_HoraExtraActividad [rel6]
alter table `horaextraactividad`  add column  `actividad_oid`
integer;
alter table `horaextraactividad`   add index
fk_horaextraactividad_activida (`actividad_oid`), add constraint
fk_horaextraactividad_activida foreign key (`actividad_oid`)
references `actividad` (`oid`);
create index `idx_horaextraactividad_activid` on
`horaextraactividad`(`actividad_oid`);


-- actividad_Asignacion [rel9]
alter table `asignacion`  add column  `actividad_oid`  integer;
```

```
alter table `asignacion`   add index fk_asignacion_actividad
(`actividad_oid`), add constraint fk_asignacion_actividad foreign
key (`actividad_oid`) references `actividad` (`oid`);
create index `idx_asignacion_actividad` on
`asignacion`(`actividad_oid`);


-- User.nombre completo [User#att35]
create view `user_nombre_completo_view` as
select AL1.`oid` as `oid`,  concat(AL1.`nombres`, ' ',
AL1.`apellidos`) as `der_attr`
from  `user` AL1
where AL1.`oid` = AL1.`oid`;


-- HistorialUsuario.total horas extras [ent1#att6]
create view `historialusuario_total_horas_e` as
select AL1.`oid` as `oid`, AL1.`total_horas_extraordinarias` +
AL1.`total_horas_normales` as `der_attr`
from  `historialusuario` AL1 ;


-- ProcessInstance.attachment count [processInstanceAttachmentCount]
create view `processinstance_attachment_cou` as
select AL1.`oid` as `oid`, count(distinct AL2.`attachment_oid`) as
`der_attr`
from  `processinstance` AL1
               left outer join `attachment_processinstance` AL2 on
AL1.`oid`=AL2.`processinstance_oid`
group by AL1.`oid`;
```