BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

“ADMINISTRACIÓN Y CONTROL DE MANTENIMIENTO EN EL DESARROLLO DE PROYECTOS”

T E S I S

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

OSCAR MANUEL JUÁREZ VILLAFAÑA

ASESOR

DR. MARIO ANZURES GARCÍA

Puebla, Pue., Octubre 2009

ÍNDICE

INTRODUCCIÓN

Como es bien sabido, los objetivos primordiales de las compañías son la reducción de costos y la mejora continua en el control, planeación y toma de decisiones. En la actualidad el ambiente cada vez más competitivo que se vive en el mundo empresarial ha dado como resultado que las Tecnologías de Información (TI) tomen un rol protagónico y formen parte de las estrategias de negocio para mejorar cada uno de sus procesos.

Internacional de Cerámica S.A. de C.V (de ahora en adelante nos referiremos a ella como Interceramic) es una empresa ampliamente conocida, dedicada al ramo de la manufactura de loseta, pisos cerámicos y sus derivados. Como toda gran empresa, Interceramic cuenta con un departamento de Tecnologías de Información, que se encarga desde desarrollar e instalar aplicaciones hasta diseñar complejas redes de computación y bases de datos. Algunas de las tareas que llevan a cabo los profesionales de TI que laboran en esta empresa incluyen, administración de datos, redes, ingeniería de hardware, diseño de programas y bases de datos, así como la administración y dirección de sus sistemas. Al igual que la mayoría de las empresas en la actualidad, Interceramic cuenta con aplicaciones empresariales propietarias y comerciales para administrar y concentrar su información. Estas aplicaciones de software son comúnmente llamadas *ERP*'s (Planificadores de Recursos Empresariales, cuyas siglas son derivadas del Ingles). Éstas ayudan a integrar la información y los procesos de todas las áreas o departamentos de la compañía, los cuales de manera directa o indirecta participan en la generación de productos y servicios.

Uno de los mayores retos a los cuales se ha enfrentado el área de TI ha sido el adaptar los diferentes subsistemas a la lógica del negocio de la compañía. Es por ello, que se han hecho una gran cantidad de modificaciones al *ERP*, así como nuevos desarrollos y soluciones para satisfacer la demanda de los usuarios que utilizan el *ERP*. En este trabajo de tesis denominado "Administración y Control de Mantenimiento en el Desarrollo de Proyectos", se pretende implementar un sistema que administre y controle el desarrollo de nuevos proyectos que sirven para que el *ERP* se mantenga actualizado con soluciones a las nuevas demandas y necesidades de los usuarios. Este tema de tesis debe controlar y administrar *qué* (¿Qué cambio, modificación o mejora se realizó?), *quién* (¿Quién lo hizo?) y *cuándo* (¿Cuándo se llevo a cabo dicho cambio?).

Dicho sistema concentrará el proceso de captura de requerimientos (que involucra a su vez la modificación y creación de programas) así como también administrará los recursos humanos inmiscuidos en los nuevos proyectos. De esta manera, el software que se desarrollará permitirá:

- Definir un flujo de trabajo de cada uno de los nuevos procesos a desarrollar, permitiendo controlar y gestionar el equipo humano y tecnológico de la empresa evitando tiempos muertos tanto en el área de análisis como en la de programación.

- Almacenar todos los cambios que se le han hecho al el *ERP* a lo largo del tiempo.

- Documentar quienes fueron los encargados de los cambios hechos al ERP.

- Organizar y estandarizar la forma de trabajar dentro del departamento.

En el desarrollo del software se empleará la tecnología *Oracle* adquirida para el desarrollo del *ERP*, de esta forma se trabajará de manera integrada siguiendo la filosofía de los *ERP´*s.

CAPITULO 1

CONCEPTOS BASICOS

1.1. Ingeniería de Software

En la computación, toda aplicación esta programada para realizar tareas específicas. A la secuencia de instrucciones que cambian el estado del hardware de una computadora o dispositivo programable se le conoce como código. A este conjunto de instrucciones que cuando se ejecutan proporcionan la función y el comportamiento deseado se le conoce como Software.

El software se suele escribir en un lenguaje de programación de alto nivel, que es más sencillo de entender (pues es más cercano al lenguaje natural humano), pero debe convertirse a lenguaje máquina para ser ejecutado.

El software se divide en tres categorías: software de sistema, software de programación y software de aplicación.

- Software de sistema: ayuda a funcionar al hardware y a la computadora. Incluye el sistema operativo, controladores de dispositivos, herramientas de diagnóstico, servidores, sistema de ventanas, utilidades y más. Su propósito es evitar en la medida de lo posible los detalles complejos de la computación, especialmente la manipulación de la memoria y el hardware.

- Software de programación: provee herramientas de asistencia al programador. Incluye editores de texto, compiladores, intérpretes de instrucciones, enlazadores, *debuggers*, etc.

- Software de aplicación: permite a los usuarios finales hacer determinadas tareas. El software en este caso depende de la necesidad del usuario y van desde los editores de texto, editores gráficos, antivirus, mensajeros, navegadores, herramientas de diseño gráfico, hasta aplicaciones de manufactura y empresariales, etc.

Por otra parte, la ingeniería consiste en aplicar conocimientos y experiencias para que mediante diseños, modelos y técnicas se resuelvan problemas que afectan a la humanidad. Su misión es transformar una idea en realidad para beneficio de la sociedad.

Al conjugarse varias disciplinas como lo son la ingeniería, la computación, la administración y la economía se conforma la ingeniería de software. El término ingeniería de software fue introducido por primera vez en una conferencia de la OTAN (Organización del Tratado del Atlántico Norte) en 1968 realizada en Alemania sobre Ciencias de la Computación.

La ingeniería de Software se define como una disciplina tecnológica y administrativa dedicada a la producción sistemática de productos de software, que son desarrollados y modificados a tiempo y dentro de un presupuesto definido. [FARLEY 1988]

El proceso de ingeniería de software se define como "un conjunto de etapas parcialmente ordenadas con la intención de lograr un objetivo, en este caso, la obtención de un producto de software de calidad" [Jacobson 1998].El proceso de desarrollo de software "es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y

el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo" [JACOBSON 1998].

El proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del software que comprende cuatro grandes fases: concepción, elaboración, construcción y transición. La concepción define el alcance del proyecto y desarrolla un caso de negocio. La elaboración define un plan del proyecto, especifica las características y fundamenta la arquitectura. La construcción crea el producto y la transición transfiere el producto a los usuarios.

Uno de los mayores problemas en el desarrollo del software es la comunicación en todos los niveles involucrados: entre usuarios, analistas y desarrolladores. Para minimizar estos problemas se emplean las técnicas, métodos y herramientas de la Ingeniería de Software.

El proceso de creación de software debe realizarse de acuerdo con una serie de normas y estándares como son:

- **Fiable**.- Capacidad de ofrecer los mismos resultados bajo las mismas condiciones.
- **Eficiente**.- Utilización óptima de los recursos de la máquina.
- **Robusto**.- No poseer un comportamiento catastrófico ante situaciones excepcionales (Tolerante a fallos).
- **Correcto**.- Se ajusta a las especificaciones dadas por el usuario.
- **Portable**.- Capaz de integrarse en entornos distintos con el mismo esfuerzo.
- **Adaptable** (*extensibilidad*).- Modificar alguna función sin que afecte a sus actividades.
- **Inteligible**.- Diseño claro, bien estructurado y documentado.
- **No Erróneo**.- No exista diferencia entre los valores reales y los calculados.
- **Reutilizable** (*reusabilidad*).- Creación de códigos u objetos que puedan ser empleados en proyectos futuros.

En la actualidad, los sistemas software son más grandes y complejos que nunca. Ello se debe en parte a los avances en rendimiento y costo del hardware, que han reducido la necesidad de limitar el tamaño y la complejidad del sistema como el principal objetivo de diseño.

Hay otras razones para el incremento de tamaño y complejidad. El software se ha convertido en la tecnología dominante en la mayoría (si no en todos) los dominios técnicos de aplicación. Normalmente es el software el que proporciona la cohesión y el control de datos que permiten a un sistema complejo resolver problemas.

1.1.1. Fases genéricas de la Ingeniería del Software

La visión general del proceso de Ingeniería de Software con independencia del área de aplicación, tamaño o complejidad del proyecto aplicado a cualquier sistema se encontrará en al menos una de las siguientes fases genéricas:

- **Análisis de requisitos**

  Extraer los requisitos de un producto de software es la primera etapa para crearlo. Mientras que los clientes piensan que ellos saben lo que el software tiene que hacer, se requiere de habilidad y

experiencia en la ingeniería de software para reconocer requisitos incompletos, ambiguos o contradictorios. El resultado del análisis de requisitos con el cliente se plasma en un documento llamado *ERS* (Especificación de Requerimientos del Sistema) cuya estructura puede venir definida por varios estándares, tales como *CMM*. Asimismo, se define un diagrama de Entidad/Relación, en el que se plasman las principales entidades que participarán en el desarrollo del software.

- **Especificación**

Es la tarea de describir detalladamente el software a ser escrito de una forma rigurosa. En la realidad, la mayoría de las buenas especificaciones han sido escritas para entender y afinar aplicaciones que ya estaban desarrolladas.

- **Diseño y arquitectura**

Se refiere a determinar como funcionará de forma general sin entrar en detalles. Según Yourdon consiste en incorporar consideraciones de la implementación tecnológica, como el hardware, la red, etc. Se definen los casos de uso para cubrir las funciones que realizará el sistema, y se transforman las entidades definidas en el análisis de requisitos en clases de diseño, obteniendo un modelo cercano a la programación orientada a objetos.

- **Programación**

Reducir un diseño a código puede ser la parte más obvia del trabajo de ingeniería de software, pero no es necesariamente la porción más larga.

- **Pruebas**

Consiste en comprobar que el software realice correctamente las tareas indicadas en la especificación. Una técnica de prueba es probar por separado cada módulo del software, y luego probarlo de forma integral.

- **Documentación**

Realización del manual de usuario, y posiblemente un manual técnico con el propósito de mantenimiento futuro y ampliaciones al sistema.

- **Mantenimiento**

La fase de mantenimiento comienza una vez construido el sistema, es decir, cuando se empieza a utilizar. El software es sometido a una serie de reparaciones y modificaciones cada vez que se detecta un fallo o se necesita cubrir una nueva necesidad de los usuarios. En la fase de mantenimiento es donde recae el mayor porcentaje del coste de un sistema (que algunos estudios cifran en hasta un 90%).

Un buen sistema no es sólo un conjunto de programas que funcionan, sino que es necesario que sea fácil de mantener para adaptarlo a modificaciones, y para poder corregir los posibles errores de programación producidos durante el desarrollo. En este sentido, la documentación del proceso de desarrollo es fundamental, y debe ser suficientemente completa y exhaustiva. Con frecuencia esta

documentación tiende a quedar desfasada con respecto a los programas y estructuras de datos del sistema. Para minimizar en la medida de lo posible este problema es conveniente hacer uso de una herramienta *CASE* tanto en el desarrollo del sistema como durante el proceso de mantenimiento.

Los tipos de mantenimiento se dividen en:

- **Correctivo**: un programa no realiza correctamente la aplicación para la que ha sido diseñado, y, por tanto, debe ser modificado.
- **Perfectivo**: modificaciones a los programas para conseguir mayor adecuación a los requisitos, mayor eficiencia, o simplemente recoger nuevas funcionalidades no expresadas en la fase de definición del sistema.
- **Adaptativo**: Adaptar los programas para acomodarlos a los cambios de su entorno externo (modificaciones en la legislación, CPU, SO, las reglas de negocio, etc.)
- **Preventivo**: El software se deteriora con los cambios, y este tipo de mantenimiento hace cambios en los programas para que se puedan corregir, adaptar y mejorar más fácilmente (Reingeniería del software).

Los costos del mantenimiento son:

- Oportunidades de desarrollo que se pierden.
- Insatisfacción del cliente cuando no se puede atender en un tiempo aceptable una petición de reparación que parece razonable.
- Los errores ocultos que se introducen al modificar el software.
- Durante el mantenimiento se reduce la calidad global del producto.
- Perjuicio en otros proyectos de desarrollo cuando el equipo de programación tiene que dejarlos, total o parcialmente, para atender peticiones de mantenimiento.
- El proceso del mantenimiento en el ciclo de vida del software radica en la implementación del proceso, el análisis de problemas y modificaciones, la implementación de las modificaciones, la revisión y aceptación del mantenimiento.

1.1.2. Ciclo de vida del Software

Todo proyecto de ingeniería tiene como fin la obtención de un producto, proceso o servicio que es necesario generar a través de diversas actividades. Algunas de estas actividades pueden agruparse en fases porque globalmente contribuyen a obtener un producto intermedio, necesario para continuar hacia el producto final y facilitar la gestión del proyecto. Al conjunto de fases empleadas se le denomina “ciclo de vida”.

Sin embargo, la forma de agrupar las actividades, los objetivos de cada fase, los tipos de productos intermedios que se generan, etc. pueden ser muy diferentes dependiendo del tipo de producto o proceso a generar y de las tecnologías empleadas.

La complejidad de las relaciones entre las distintas actividades crece exponencialmente con el tamaño, por lo cual se aplican técnicas como la de “divide y vencerás”. De esta forma la división de los proyectos en fases sucesivas es un primer paso para la reducción de su complejidad, tratándose de escoger las partes de manera que sus relaciones entre sí sean lo más simples posibles.

La definición de un ciclo de vida facilita el control sobre los tiempos en que es necesario aplicar recursos de todo tipo (personal, equipos, suministros, etc.) al proyecto. Si el proyecto incluye subcontratación de partes a otras organizaciones, el control del trabajo subcontratado se facilita en la medida en que esas partes encajen bien en la estructura de las fases. El control de calidad también se ve facilitado si la separación entre fases se hace corresponder con puntos en los que ésta deba verificarse (mediante comprobaciones sobre los productos parciales obtenidos).

De la misma forma, la práctica acumulada en el diseño de modelos de ciclo de vida para situaciones muy diversas permite que nos beneficiemos de la experiencia adquirida utilizando el enfoque que mejor se adapte a nuestros requerimientos.

Un ciclo de vida para un proyecto se compone de fases sucesivas compuestas por tareas planificadas. Según el modelo de ciclo de vida, la sucesión de fases puede ampliarse con bucles de realimentación, de manera que lo que conceptualmente se considera una misma fase se pueda ejecutar más de una vez a lo largo de un proyecto, recibiendo en cada pasada de ejecución aportaciones de los resultados intermedios que se van produciendo (realimentación). Ver Fig. 1.1.

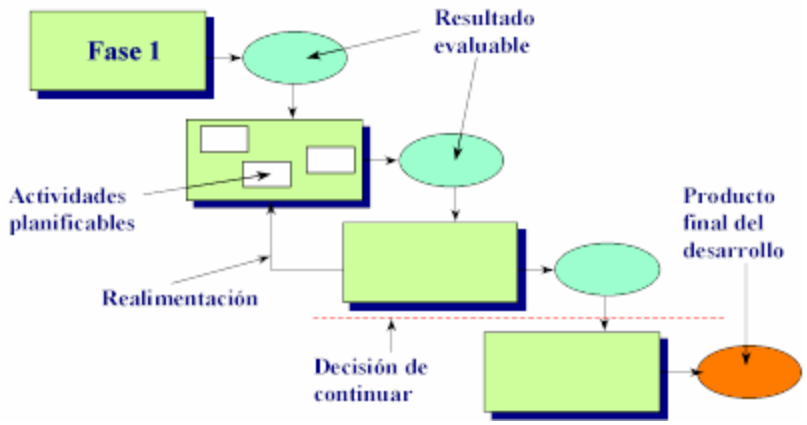

Figura 1.1. Ejemplo de ciclo de vida.

Para un adecuado control de la progresión de las fases de un proyecto se hace necesario especificar con suficiente precisión los resultados evaluables, es decir, los productos intermedios que deben resultar de las tareas incluidas en cada fase. Normalmente estos productos marcan los puntos de referencia entre las fases.

Elementos del ciclo de vida

Fases

Una fase es un conjunto de actividades relacionadas con un objetivo en el desarrollo del proyecto. Se construye agrupando tareas (actividades elementales) que pueden compartir una porción determinada del tiempo de vida de un proyecto. La agrupación temporal de tareas impone requisitos temporales correspondientes a la asignación de recursos (ya sean humanos, financieros o materiales).

Cuanto más grande y complejo sea un proyecto, mayor detalle se necesitará en la definición de las fases para que el contenido de cada una siga siendo manejable. De esta forma, cada fase de un proyecto puede considerarse un "micro-proyecto" en sí mismo, compuesto por un conjunto de micro-fases.

Cada fase viene definida por un conjunto de elementos observables externamente, como son las actividades con las que se relaciona, los datos de entrada (resultados de la fase anterior, documentos o productos requeridos para la fase, experiencias de proyectos anteriores), los datos de salida (resultados a utilizar por la fase posterior, experiencia acumulada, pruebas o resultados efectuados) y la estructura interna de la fase. Ver Fig. 1.2.

Figura 1.2. Esquema general de operación de una fase.

Entregables

Son los productos intermedios que generan las fases. Pueden ser tangible (componentes, equipos) o intangible (documentos, software). Los entregables permiten evaluar la marcha del proyecto mediante comprobaciones de su adecuación o no a los requisitos funcionales y de condiciones de realización previamente establecidos. Cada una de estas evaluaciones puede servir, además, para la toma de decisiones a lo largo del desarrollo del proyecto.

1.1.3. Tipos de modelo de ciclo de vida

Las principales diferencias entre los distintos modelos de ciclo de vida están en:

El alcance del ciclo dependiendo de hasta dónde llegue el proyecto correspondiente. Un proyecto puede comprender un simple estudio de viabilidad del desarrollo de un producto o su desarrollo completo.

Las características (contenidos) de las fases en que dividen el ciclo. Esto puede depender del propio tema al que se refiere el proyecto (no son lo mismo las tareas que deben realizarse para proyectar un avión que un puente), o de la organización (interés de reflejar en la división en fases aspectos de la división interna o externa del trabajo).

La estructura de la sucesión de las fases que puede ser lineal, con prototipos, o en espiral. Veámoslo con más detalle:

Ciclo de vida lineal

Es el más utilizado, siempre que es posible, precisamente por ser el más sencillo. Consiste en descomponer la actividad global del proyecto en fases que se suceden de manera lineal, es decir, cada una se realiza una sola vez, cada una se realiza tras la anterior y antes que la siguiente. El ciclo de vida lineal se muestra en la Fig. 1.3.

Con un ciclo lineal es fácil dividir las tareas entre equipos sucesivos, y prever los tiempos (sumando los de cada fase).Requiere que la actividad del proyecto pueda descomponerse de manera que una fase no necesite resultados de las siguientes (realimentación), aunque pueden admitirse ciertos supuestos de realimentación correctiva. Desde el punto de vista de la gestión (para decisiones de planificación), requiere también que se sepa bien de antemano lo que va a ocurrir en cada fase antes de empezarla.

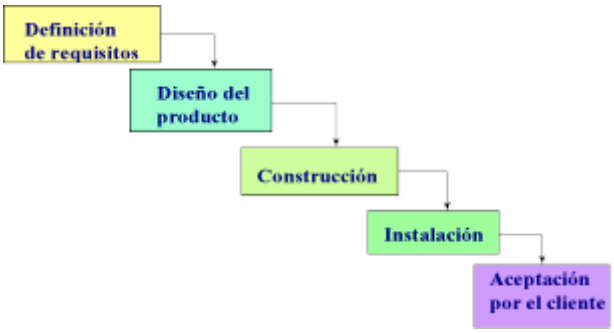

Figura 1.3. Ejemplo de ciclo lineal para un proyecto de construcción.

Ciclo de vida con prototipos

A menudo ocurre en desarrollos de productos con innovaciones importantes, o cuando se prevé la utilización de tecnologías nuevas o poco probadas, que las incertidumbres sobre los resultados realmente alcanzables, o las ignorancias sobre el comportamiento de las tecnologías, impiden iniciar un proyecto lineal con especificaciones cerradas.
Si no se conoce exactamente cómo desarrollar un determinado producto o cuáles son las especificaciones de forma precisa, suele recurrirse a definir especificaciones iniciales para hacer un prototipo, o sea, un producto parcial (no hace falta que contenga funciones que se consideren triviales o suficientemente probadas) y provisional (no se va a fabricar realmente para clientes, por lo que tiene menos restricciones de coste y/o prestaciones). Este tipo de procedimiento es muy utilizado en desarrollo avanzado.

Ciclo de vida en espiral

El ciclo de vida en espiral puede considerarse como una generalización del anterior para los casos en que no basta con una sola evaluación de un prototipo para asegurar la desaparición de incertidumbres y/o ignorancias. El propio producto a lo largo de su desarrollo puede así considerarse como una sucesión de prototipos que progresan hasta llegar a alcanzar el estado deseado. En cada ciclo (espirales) las especificaciones del producto se van resolviendo paulatinamente.

A menudo la fuente de incertidumbres es el propio cliente, que aunque sepa en términos generales lo que quiere, no es capaz de definirlo en todos sus aspectos sin ver como unos influyen en otros. En estos

casos la evaluación de los resultados por el cliente no puede esperar a la entrega final y puede ser necesaria repetidas veces.

El análisis de sistemas consiste en la descomposición de un sistema en componentes para poder estudiar y analizar como trabajan tanto de forma aislada como interactuando con el resto. Se apoya en representaciones graficas (modelos) que describen los procesos del negocio, el problema a resolver y el sistema que ha de ser desarrollado.

1.1.4. Modelo de Software

Un modelo es una simplificación de la realidad. Es el resultado de un proceso de abstracción y ayuda a comprender y razonar sobre una realidad. En términos de Ingeniería de Software, un modelo es una descripción de un aspecto del sistema, expresada en un lenguaje bien definido. Es decir los modelos visualizan como es o que queremos que sea el sistema, especifican la estructura y comportamiento del mismo, así como guían su construcción.

La clave del correcto desarrollo del software es basarse en un modelo o notación (visual) bien definido apoyándose en las herramientas necesarias para poder representar los procesos de la manera más simple posible.

La notación visual ayuda a:

- Manejar la complejidad que se presenta dentro de los procesos del negocio.
- Modelar el sistema independientemente del lenguaje de implementación.
- Promover la reutilización de componentes.

La utilidad del modelado consiste en ayudar a razonar sobre como se implementará el sistema, facilitando la comunicación entre el equipo al existir un lenguaje común. La generación del código se realizará a partir de modelos (bien documentados), los cuales trascenderán al proyecto.

En conclusión, el modelado es el análisis y diseño de aplicaciones de software antes de escribir el código. Dentro de el, se crean un conjunto de modelos o “planos del software” que permiten especificar aspectos del sistema como los requisitos, la estructura y el comportamiento.

1.1.5. Análisis Estructurado

El análisis estructurado es una actividad de construcción de modelos, utiliza una notación propia con la que se crean modelos que reflejan el flujo y el contenido de la información (datos y control), también se divide el sistema funcionalmente estableciendo lo que se debe construir.

Algunas de las características que debe cumplir el modelo construido son:

- Los productos de análisis deben ser claros y fáciles de entender (especificación de requisitos del software).
- Fragmentar los problemas de gran tamaño.
- Utilizar esquemas, diagramas y gráficos.
- Diferenciar las consideraciones lógicas y físicas (para la implantación).

Metodología del análisis estructurado

El análisis estructurado es una filosofía de ingeniería de software, de esta se desprenden diversas metodologías que presentan diferentes características. Algunas de estas metodologías son: Jackson, Page-Jones, Gane & Sarson , Merise, Jourdon/De Marco, Warnier, Chen, SSADM.

Distinguiendo las de Chen que incluye el modelado de datos. Las metodologías tienen diferentes ventajas y se deben aplicar de acuerdo al problema que se va a analizar.

Elementos del modelo de análisis

En las metodologías del modelo de análisis encontramos en común los siguientes objetivos primarios:

1. Describir las necesidades del cliente.
2. Establecer una base para la creación de un diseño de software.
3. Definir un conjunto de requisitos que se puedan validar.
4. Obtener la aprobación del cliente.

El análisis estructurado, se apoya en lo siguiente:

Diccionario de datos (DD): contiene definiciones de todos los objetos de datos consumidos y producidos por el software.

Diagrama entidad-relación (DER): representa las relaciones entre entidades de datos. Los atributos de cada entidad se pueden describir mediante la descripción de datos.

Diagrama de flujo de datos (DFD): modelo utilizado para mostrar al sistema como una red de procesos funcionales y permite describir el movimiento de los datos a través del sistema.

1.1.6. Diagramas de flujo de datos

En pocas palabras, los DFD son representaciones gráficas de los procesos de un sistema, incluidas sus entradas y salidas de información. Estos están enfocados en el flujo de los datos del sistema y son representados en forma de red. Pero, ¿cómo se representa un sistema gráficamente? Para ello se utilizan cuatro elementos principales, que deben representar en el diagrama cada componente a considerar del sistema. Estos son los flujos, los procesos, los archivos y las fuentes de datos o depósitos. A continuación se define cada componente.

Flujo. Los flujos representan la comunicación entre el resto de los componentes de un diagrama y, aunque por lo general están conectados solamente a dos de ellos, pueden divergir o converger, es decir, pueden separarse o unirse dos o más. El nombre del flujo nos da una idea de los datos que se traspasan de un elemento a otro, pero la definición correcta y única de ello se encuentra en el Diccionario de Datos. Los flujos se representan mediante una flecha unidireccional.

Proceso. Se pueden considerar como la parte principal de un diagrama. Representan las acciones u operaciones que se llevan a cabo en el sistema, de acuerdo a las entradas que reciba, para producir salidas esperadas. Éstos se dibujan con una burbuja, como se puede apreciar en la Figura 1.4.

Archivo/Almacén de Datos. Como archivo podemos tomar cualquier contenedor de datos o información relevante al sistema, que se almacene de manera temporal, ya sea en memoria, disco, cinta magnética, etc. De ellos se pueden introducir y extraer datos. La notación para los archivos, que se puede ver en la Figura, consta de un título sobre una línea recta o entre dos líneas horizontales.

Fuente de datos/ Unidad Externa. Las fuentes de datos son personas u organizaciones exteriores al sistema que se analiza, pero que son considerados como aportadores o receptores de datos del sistema. Estos son incluidos para añadir comprensibilidad al diagrama y al análisis en general. En la Figura siguiente (1.4) se aprecia que son representados mediante cajas con su nombre en el interior.

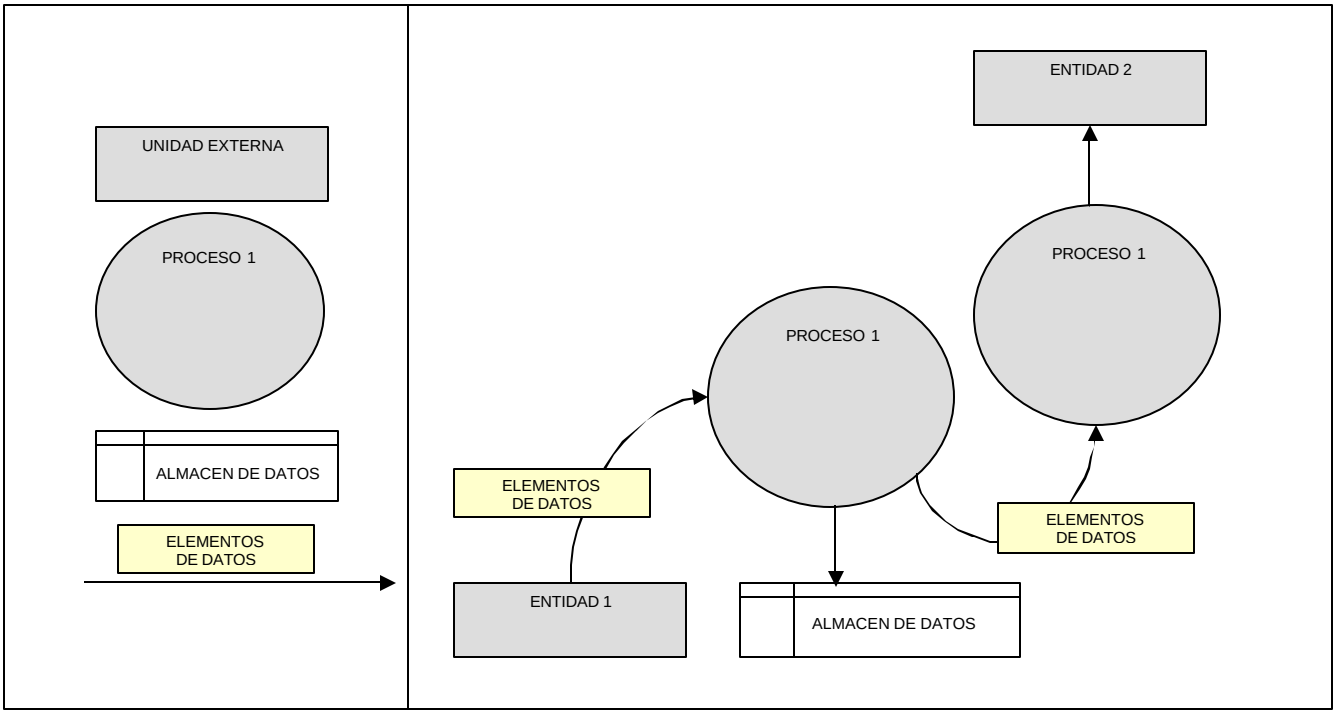

Figura 1.4. Componentes de Un DFD.

Asimismo, los DFD pueden componerse por distintos “niveles”, los cuales detallan los procesos que se llevan a cabo durante cada proceso. Estos niveles aumentan la información y nuestro conocimiento del sistema en su totalidad. Este ejemplo de un DFD se encuentra en el primer nivel, y cada una de sus procesos se puede descomponer en un DFD interno con sus propios procesos y flujos de datos.

Como se debe de conservar la coherencia en las relaciones entre los DFD de los distintos niveles, el principal requisito para que este bien desarrollado un DFD en un nivel inferior es que sus entradas de datos deben de ser completamente equivalentes a las entradas de datos que tenga el proceso que lo contiene en el nivel superior inmediato. Esta misma regla debe aplicarse con los flujos de datos de salida. Finalmente los procesos dejarán de descomponerse en más DFD cuando se dé una de las siguientes condiciones: que los procesos sean lo suficientemente sencillos como para que su funcionamiento sea obvio y lógico para cualquier persona, o bien, que la complejidad del sistema limite

el número de DFD a desarrollar. En cualquiera de los dos casos el DFD debe ser lo suficientemente completo para dar una buena explicación del funcionamiento del sistema, sin ser tan vasto y complejo que sea extremadamente difícil su comprensión.

1.2. Modelo Entidad Relación

El modelo entidad - relación (MER) fue formalizado por Peter P. Chen en el año 1976 [CHEN 1976]. El objetivo es modelar una base de datos. Dispone de una notación para modelar datos y genera un diagrama. Los conceptos del MER son similares a la composición de caracteres chinos, ideogramas y jeroglíficos. El modelo se fundamenta en la teoría de conjuntos, el álgebra moderna, la lógica y la teoría de mallas. Los símbolos más importantes del MER son:

1. Entidad: Concepto acerca del cual se registra información que debe tener la característica de ser distinguible.

2. Atributos: Representan propiedades elementales de las entidades o de las relaciones. Los atributos **pueden ser simples o compuestos.**

El MER es un modelo semántico centrado en las relaciones más que en los atributos. EL MER ve el mundo como un conjunto de entidades y relaciones entre estas entidades. La filosofía subyacente es que un atributo es sólo un simple hecho acerca de una entidad, mientras que una relación puede modelar la construcción de entidades más complejas a partir de otras entidades.

En la Figura 1.5 están los símbolos gráficos básicos para elaborar el diagrama entidad relación, se tomo la notación de Chen [CHEN 1976]. La cardinalidad asociada con las relaciones entre las entidades se expresa por una notación entre paréntesis, la cardinalidad es un índice del mapeo de una relación sobre el grupo de entidades.

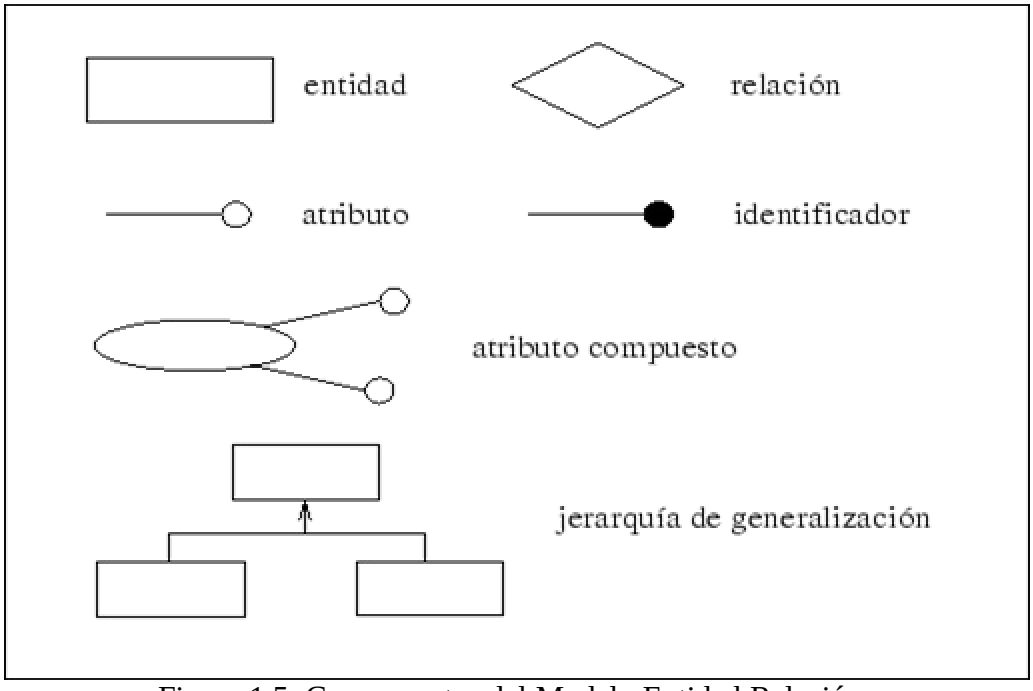

Figura 1.5. Componentes del Modelo Entidad Relación.

1.3. Bases de Datos

Conforme la tecnología Informática ha evolucionado, la necesidad de almacenar la información en forma ordenada y eficiente se ha incrementado. En la actualidad la gestión de información ha dejado de ser solo una aplicación más para convertirse en la parte fundamental de cualquier sistema de información. Un sistema de información será más valioso conforme la base de datos que lo soporte sea de mayor calidad, de tal forma que puede llegar a afirmarse que es imposible la existencia de un sistema de información sin una base de datos, pues esta cumple la función de "memoria", en todas sus acepciones posibles, del sistema.

Las bases de datos almacenan como su nombre lo indica, datos, estos datos son representaciones de sucesos y objetos, a diferente nivel, existentes en el mundo real, pues en su conjunto representan algún tipo de entidad existente. En el mundo real se tiene percepción sobre las entidades u objetos y los atributos de estos; en las bases de datos se trata de registrar esta información de una forma estructurada y bajo ciertas reglas, lo que se le conoce como el análisis y diseño de bases de datos. Sin embargo, es necesario definir en primer lugar, qué es una base de datos, independientemente de su Diseño y/o su orientación.

1.3.1. Definición de Base de Datos

Debido a las numerosas definiciones que existen en la Bibliografía, se ha elegido la siguiente dada su exhaustividad:

"Una base de datos es una colección de datos correspondientes a las diferentes perspectivas de un sistema de información (de una empresa o institución), existentes en algún soporte de tipo físico (normalmente de acceso directo), agrupados en una organización integrada y centralizada en la que figuran no solo datos en si, sino también las relaciones existentes entre ellos, de forma que se minimiza la redundancia y se maximiza la independencia de los datos de las aplicaciones que los requieren" [GUILERA 1993]

1.3.2. Sistemas Gestores de Bases de Datos

Hace algunos años, las bases de datos eran el resultado de una compleja programación y de complicados mecanismos de almacenamiento. Con la popularización de la microinformática, la aparición de aplicaciones específicas trajo consigo la disponibilidad de herramientas de gestión de datos, las cuales acabaron desembocando en los denominados Sistemas de Gestión de Bases de Datos, también conocidos como SGBD (*DBMS* en inglés, siglas de *DataBase Management System*). De esta manera, la gestión de bases de datos pudo liberarse de los grandes ordenadores centrales, pudiendo distribuirse según los intereses y demandas de los usuarios, y dotando de autonomía en la gestión de información a muchas entidades.

Un Sistema de Gestión de Bases de Datos consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a ellos. La colección de datos se denomina base de datos (BD). El objetivo primordial de un *DBMS* es proporcionar un entorno que a su vez sea conveniente y eficiente para ser utilizado al extraer o almacenar información en la BD.

Los Sistemas de Gestión de Bases de Datos están diseñados para gestionar grandes bloques de información, que implica tanto la definición de estructuras para el almacenamiento como de mecanismos para la gestión de la información.

Además los *DBMS* deben mantener la seguridad de la información almacenada pese a la caída del sistema o accesos no autorizados.

1.3.3. Objetivos de los Sistemas de Bases de Datos

Los Sistemas de Gestión de Bases de Datos surgieron debido a que existían muchas dificultades en Programas de Aplicación que manejaban grandes cantidades de información, como son:

1.- Redundancia e inconsistencia de los datos.
2.- Dificultad para tener acceso a los datos.
3.- Aislamiento de los datos.
4.- Anomalías del acceso concurrente.
5.- Problemas de seguridad.
6.- Problemas de integridad.

1.3.4. Arquitectura de las Bases de Datos

Los SGBD permitieron a todo tipo de usuarios crear y mantener sus bases de datos, dotándolos de una herramienta que era capaz de transformar el nivel lógico que éstos diseñaban en un conjunto de datos, representaciones y relaciones, traduciéndolo al nivel físico correspondiente. Para que fuese posible, y para asegurar a los usuarios cierta seguridad en el intercambio de datos entre diferentes sistemas, y en el diseño de ficheros y bases de datos, fue necesario normalizar los esquemas que guiaban la creación de las bases de datos.

Las bases de datos respetan la arquitectura de tres niveles definida, para cualquier tipo de base de datos, por el grupo *ANSI/SPARC*. En esta arquitectura la base de datos se divide en los niveles interno, conceptual y externo. Ver Fig. 1.6.

1.- Nivel interno. El nivel más bajo de abstracción describe como se almacenan realmente los datos, se describen en detalle las estructuras de datos complejas del nivel bajo.

2.- Nivel conceptual. Describe qué datos son realmente almacenados en la BD y las relaciones que existen entre ellos. Aquí se describe la BD completa en términos de un número pequeño de estructuras sencillas. El nivel conceptual de abstracción lo usan los administradores de BD, quienes deben decidir que información se va a guardar en la BD.

3.- Nivel externo. El nivel más alto de abstracción describe sólo parte de la BD completa. Muchos usuarios no se interesan por toda la información, sólo necesitan una parte de la BD. Para simplificar su interacción con el sistema se define el nivel externo. El sistema puede proporcionar muchas vistas para la misma BD.

En el nivel físico, un registro puede describirse como un bloque de posiciones de memoria consecutivas. En el nivel conceptual, cada registro se describe por medio de una definición de tipo, y se define la interrelación entre los tipos. Finalmente, en el nivel de visión, se definen varias visiones de la

BD. Por ejemplo, un cajero sólo ve información sobre las cuentas de los clientes, sin poder acceder a los salarios de los empleados.

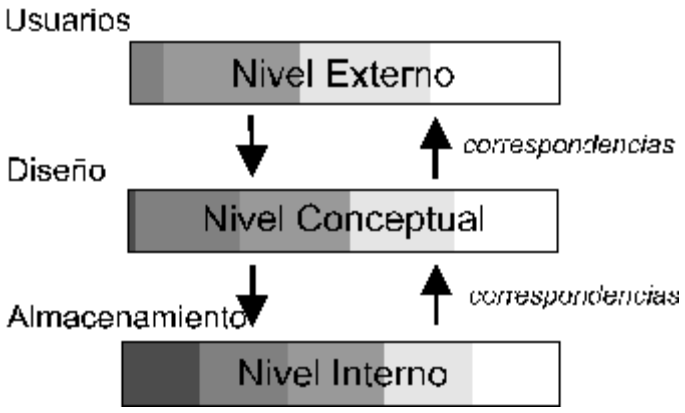

Figura 1.6. Arquitectura de 3 Niveles.

1.3.5. Modelos de datos

Es el proceso de abstracción que conduce a la creación de una base de datos. Desempeña una función prioritaria el modelo de datos. El modelo de datos, como abstracción del universo de discurso, es el enfoque utilizado para la representación de las entidades y sus características dentro de la base de datos, y puede ser dividido en tres grandes tipos [KORTH y SILBERSCHATZ 1993]:

1. Modelos lógicos basados en objetos: los dos más extendidos son el modelo entidad-relación y el orientado a objetos. El modelo entidad-relación (E-R) se basa en una percepción del mundo compuesta por objetos, llamados entidades, y relaciones entre ellos. Las entidades se diferencian unas de otras a través de atributos. El orientado a objetos también se basa en objetos, los cuales contienen valores y métodos, entendidos como órdenes que actúan sobre los valores, en niveles de anidamiento. Los objetos se agrupan en clases, relacionándose mediante el envío de mensajes. Algunos autores definen estos modelos como "modelos semánticos".

2. Modelos lógicos basados en registros: el más extendido es el relacional, mientras que los otros dos existentes, jerárquico y de red, se encuentran en retroceso. Estos modelos se usan para especificar la estructura lógica global de la base de datos, estructurada en registros de formato fijo de varios tipos. El modelo relacional representa los datos y sus relaciones mediante tablas bidimensionales, que contienen datos tomados de los dominios correspondientes. El modelo de red está formado por colecciones de registros, relacionados mediante punteros o ligas en grafos arbitrarios. el modelo jerárquico es similar al de red, pero los registros se organizan como colecciones de árboles. Algunos autores definen estos modelos como "modelos de datos clásicos".

3. Modelos físicos de datos: muy poco usados, son el modelo unificador y el de memoria de elementos. Algunos autores definen estos modelos como "modelos de datos primitivos".

De lo anterior se deduce que el punto clave en la construcción de una base de datos será el modelo de datos. Se denomina modelo:

"...al instrumento que se aplica a una parcela del mundo real (universo del discurso) para obtener una estructura de datos a la que denominamos esquema. Esta distinción entre el modelo (instrumento) y el esquema (resultado de aplicar el instrumento) es importante... Es importante también distinguir entre mundo real y universo del discurso, ya que este último es la visión que del mundo real tiene el diseñador... podemos definir un modelo de datos como un conjunto de conceptos, reglas y convenciones que nos permiten describir los datos del universo del discurso." [MIGUEL y PIATTINI, 1993]

Los objetivos del modelo de datos son dos:

1. Formalización: definir formalmente las estructuras permitidas y las restricciones a fin de representar los datos de un Sistema de Información.

2. Diseño: el modelo resultante es un elemento básico para el desarrollo de la metodología de diseño de la base de datos.

Los diferentes modelos de datos comparten, aunque con diferentes nombres y notaciones, unos elementos comunes, estos son componentes básicos de la representación de la realidad que realizan. Estos componentes se identifican gracias a la clasificación, y pueden identificarse como conceptos estáticos y conceptos dinámicos. Los conceptos estáticos corresponden a:

a. Objeto: cualquier entidad con existencia independiente sobre el que almacenan datos. Puede ser simples o compuestos.

b. Relación: asociación entre objetos.

c. Restricción estática: propiedad estática del mundo real que no puede expresarse con los anteriores, ya que sólo se da en la base de datos; suele corresponder a valores u ocurrencias, y puede ser sobre atributos, entidades y relaciones.

d. Objeto compuesto: definidos como nuevos objetos dentro de la base de datos, tomando como punto de partida otros existentes, mediante mecanismos de agregación y asociación.

e. Generalización: se trata de relaciones de subclase entre objetos, es decir, parte de las características de diferentes entidades pueden resultar comunes entre ellas.

Por su parte, los conceptos dinámicos responden a:

a. Operación: acción básica sobre objetos o relaciones (crear, modificar, eliminar...).
b. Transacción: conjunto de operaciones que deben ejecutarse en su conjunto obligatoriamente.
c. Restricción dinámica: propiedades del mundo real que restringen la evolución en el tiempo de la base de datos.

1.3.6. Funciones de los Sistemas de Gestión de Bases de Datos

Para plasmar los tres niveles en el enfoque o modelo de datos seleccionado, es necesaria una aplicación que actúe de interfaz entre el usuario, los modelos y el sistema físico. Ver Figura 1.7. Esta es la función que desempeñan los SGBD, ya reseñados, y que pueden definirse como un paquete generalizado de software, que se ejecuta en un sistema computacional anfitrión, centralizando los accesos a los datos y actuando de interfaz entre los datos físicos y el usuario. Las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad. Se han señalado como componentes de un sistema ideal de gestión de bases de datos los siguientes [FROST 1989]:

- Un lenguaje de definición de esquema conceptual.
- Un sistema de diccionario de datos.
- Un lenguaje de especificación de paquetes de entrada/salida.
- Un lenguaje de definición de esquemas de base de datos.
- Una estructura simétrica de almacenamiento de datos.
- Un módulo de transformación lógica a física.
- Un subsistema de privacidad de propósito general.
- Un subsistema de integridad de propósito general
- Un subsistema de reserva y recuperación de propósito general.
- Un generador de programas de aplicación.
- Un generador de programas de informes.
- Un lenguaje de consulta de propósito general.

El SGBD incorpora como herramienta fundamental dos lenguajes, para la definición y la manipulación de los datos.

El lenguaje de definición de datos (*DDL, Data Definition Language*) provee de los medios necesarios para definir los datos con precisión, especificando las distintas estructuras. Acorde con el modelo de arquitectura de tres niveles, habrá un lenguaje de definición de la estructura lógica global, otro para la definición de la estructura interna, y un tercero para la definición de las estructuras externas.

El lenguaje de manipulación de datos (*DML, Data Manipulation/Management Language*), que es el encargado de facilitar a los usuarios el acceso y manipulación de los datos. Pueden diferenciarse en procedimentales (aquellos que requieren qué datos se necesitan y cómo obtenerlos) y no procedimentales (que datos se necesitan, sin especificar como obtenerlos), y se encargan de la recuperación de los datos almacenados, de la inserción y supresión de datos en la base de datos, y de la modificación de los existentes.

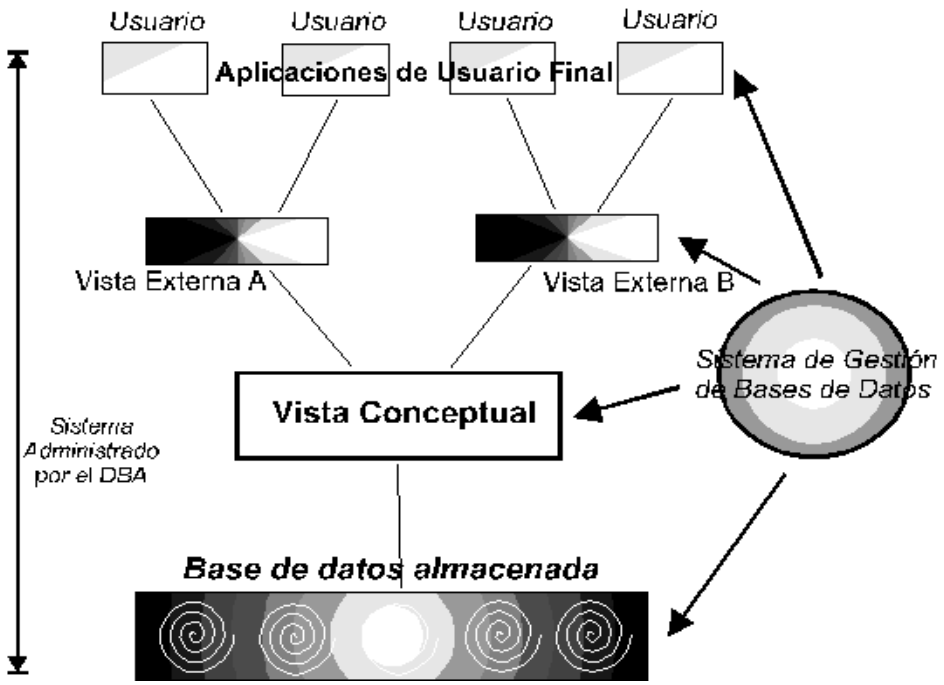

Figura 1.7. Sistema Gestor de Bases de Datos.

1.3.7. El Modelo Relacional

El modelo relacional se basa en dos ramas de las matemáticas: la teoría de conjuntos y la lógica de predicados de primer orden. El hecho de que el modelo relacional esté basado en la teoría de las matemáticas es lo que lo hace tan seguro y robusto. Al mismo tiempo, estas ramas de las matemáticas proporcionan los elementos básicos necesarios para crear una base de datos relacional con una buena estructura, y proporcionan las líneas que se utilizan para formular buenas metodologías de diseño.

La teoría matemática proporciona la base para el modelo relacional y, por lo tanto, hace que el modelo sea predecible, fiable y seguro. La teoría describe los elementos básicos que se utilizan para crear una base de datos relacional y proporciona las líneas a seguir para construirla. El organizar estos elementos para conseguir el resultado deseado es lo que se denomina diseño.

El modelo relacional, como todo modelo de datos, tiene que ver con tres aspectos de los datos:

- Estructura de datos.
- Integridad de datos.
- Manejo de datos.

Estructura de Datos Relacional

Relaciones

El modelo relacional se basa en el concepto matemático de relación, que gráficamente se representa mediante una tabla. Codd, que era un experto matemático, utilizó una terminología perteneciente a las matemáticas, en concreto de la teoría de conjuntos y de la lógica de predicados.

Una relación es una tabla con columnas y filas. Un SGBD sólo necesita que el usuario pueda percibir la base de datos como un conjunto de tablas. Esta percepción sólo se aplica a la estructura lógica de la base de datos (en el nivel externo y conceptual de la arquitectura de tres niveles ANSI-SPARC). No se aplica a la estructura física de la base de datos, que se puede implementar con distintas estructuras de almacenamiento.

Un atributo es el nombre de una columna de una relación. En el modelo relacional, las relaciones se utilizan para almacenar información sobre los objetos que se representan en la base de datos. Una relación se representa gráficamente como una tabla bidimensional en la que las filas corresponden a registros individuales y las columnas corresponden a los campos o atributos de esos registros. Los atributos pueden aparecer en la relación en cualquier orden.

Un dominio es el conjunto de valores legales de uno o varios atributos. Los dominios constituyen una poderosa característica del modelo relacional. Cada atributo de una base de datos relacional se define sobre un dominio, pudiendo haber varios atributos definidos sobre el mismo dominio.

Una tupla es una fila de una relación. Los elementos de una relación son las tuplas o filas de la tabla. Las tuplas de una relación no siguen ningún orden.

El grado de una relación es el número de atributos que contiene. Esto quiere decir que cada fila de la tabla es una tupla con n valores, donde n es el numero de atributos. El grado de una relación no cambia con frecuencia.

La cardinalidad de una relación es el número de tuplas que contiene. Ya que en las relaciones se van insertando y borrando tuplas a menudo, la cardinalidad de las mismas varía constantemente.

Una base de datos relacional es un conjunto de relaciones normalizadas.

Propiedades de las relaciones

Las relaciones tienen las siguientes características:

- Cada relación tiene un nombre y éste es distinto del nombre de todas las demás.
- Los valores de los atributos son atómicos: en cada tupla, cada atributo toma un solo valor. Se dice que las relaciones están normalizadas.
- No hay dos atributos que se llamen igual.
- El orden de los atributos no importa: los atributos no están ordenados.
- Cada tupla es distinta de las demás: no hay tuplas duplicadas.
- El orden de las tuplas no importa: las tuplas no están ordenadas.

Tipos de relaciones

En un SGBD relacional pueden existir varios tipos de relaciones, aunque no todos manejan todos los tipos.

- Relaciones base. Son relaciones reales que tienen nombre y forman parte directa de la base de datos almacenada (son autónomas).

- Vistas. También denominadas relaciones virtuales, son relaciones con nombre y derivadas: se representan mediante su definición en términos de otras relaciones con nombre, no poseen datos almacenados propios.
- Instantáneas. Son relaciones con nombre y derivadas. Pero a diferencia de las vistas, son reales, no virtuales: están representadas no sólo por su definición en términos de otras relaciones con nombre, sino también por sus propios datos almacenados. Son relaciones de sólo de lectura y se refrescan periódicamente.
- Resultados de consultas. Son las relaciones resultantes de alguna consulta especificada. Pueden o no tener nombre y no persisten en la base de datos.
- Resultados intermedios. Son las relaciones que contienen los resultados de las subconsultas. Normalmente no tienen nombre y tampoco persisten en la base de datos.
- Resultados temporales. Son relaciones con nombre, similares a las relaciones base o a las instantáneas, pero la diferencia es que se destruyen automáticamente en algún momento apropiado.

Llaves

Ya que en una relación no hay tuplas repetidas, éstas se pueden distinguir unas de otras, es decir, se pueden identificar de modo único. La forma de identificarlas es mediante los valores de sus atributos.

Una llave primaria es un atributo o un conjunto de atributos que identifican de modo único las tuplas de una relación.

Una relación puede contener varias llaves candidatas. Las llaves candidatas son aquellas que cumplen con la definición de llave primaria, pero dado que solo puede existir una llave primaria por relación, a estas se les conoce como llaves candidatas.

Una llave candidata es una llave en la que ninguno de sus subconjuntos es una llave primaria de la relación. El atributo o conjunto de atributos K de la relación R es una clave candidata para R si y sólo si satisface las siguientes propiedades:

- Unicidad: nunca hay dos tuplas en la relación R con el mismo valor de K.
- Irreductibilidad (minimalidad): ningún subconjunto de K tiene la propiedad de unicidad, es decir, no se pueden eliminar componentes de K sin destruir la unicidad.

Cuando una llave candidata está formada por más de un atributo, se dice que es una llave compuesta.

Para identificar las llaves candidatas de una relación no hay que fijarse en un estado o instancia de la base de datos. El hecho de que en un momento dado no haya duplicados para un atributo o conjunto de atributos, no garantiza que los duplicados no sean posibles. Sin embargo, la presencia de duplicados en un estado de la base de datos sí es útil para demostrar que cierta combinación de atributos no es una llave candidata. El único modo de identificar las llaves candidatas es conociendo el significado real de los atributos, ya que esto permite saber si es posible que aparezcan duplicados. Sólo usando esta información semántica se puede saber con certeza si un conjunto de atributos forman una llave candidata.

La llave primaria de una relación es aquella llave candidata que se escoge para identificar sus tuplas de modo único. Ya que una relación no tiene tuplas duplicadas, siempre hay una llave candidata y, por lo

tanto, la relación siempre tiene una llave primaria. En el peor caso, la llave primaria estará formada por todos los atributos de la relación, pero normalmente habrá un pequeño subconjunto de los atributos que haga esta función.

Las llaves candidatas que no son escogidas como llave primaria son denominadas llaves alternativas.

Una llave ajena es un atributo o un conjunto de atributos de una relación cuyos valores coinciden con los valores de la llave primaria de alguna otra relación (puede ser la misma). Las llaves ajenas representan relaciones entre datos. Se dice que un valor de llave ajena representa una referencia a la tupla que contiene el mismo valor en su llave primaria (tupla referenciada).

Esquema de una base de datos relacional

Una base de datos relacional es un conjunto de relaciones normalizadas. Para representar el esquema de una base de datos relacional se debe dar el nombre de sus relaciones, los atributos de éstas, los dominios sobre los que se definen estos atributos, las llaves primarias y las llaves ajenas.
En el esquema, los nombres de las relaciones aparecen seguidos de los nombres de los atributos encerrados entre paréntesis. Las llaves primarias son los atributos subrayados.

Integridad de los Datos

Reglas de integridad

Las reglas de integridad buscan garantizar que los datos almacenados en dicha estructura sean correctos.

Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo. A este tipo de restricciones se les denomina restricciones de dominios. Hay además dos reglas de integridad muy importantes que son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados o instancias (las reglas se deben cumplir todo el tiempo). Estas reglas son la regla de integridad de entidades y la regla de integridad referencial. Antes de definirlas, es preciso conocer el concepto de nulo.

Nulos

Cuando en una tupla un atributo es desconocido, se dice que es nulo. Un nulo no representa el valor cero ni la cadena vacía, éstos son valores que tienen significado. El nulo implica ausencia de información, bien porque al insertar la tupla se desconocía el valor del atributo, o bien porque para dicha tupla el atributo no tiene sentido. Ya que los nulos no son valores, deben tratarse de modo diferente, lo que causa problemas de implementación. De hecho, no todos los SGBD relacionales soportan los nulos.

Regla de integridad de entidades

La primera regla de integridad se aplica a las llaves primarias de las relaciones base: ninguno de los atributos que componen la llave primaria puede ser nulo.

Por definición, una llave primaria es un identificador irreducible que se utiliza para identificar de modo único las tuplas. Que es irreducible significa que ningún subconjunto de la llave primaria sirve para

identificar las tuplas de modo único. Si se permite que parte de la llave primaria sea nula, se está diciendo que no todos sus atributos son necesarios para distinguir las tuplas, con lo que se contradice la irreducibilidad.

Nótese que esta regla sólo se aplica a las relaciones base y a las llaves primarias, no a las llaves alternativas.

Regla de integridad referencial

La segunda regla de integridad se aplica a las llaves ajenas: si en una relación hay alguna llave ajena, sus valores deben coincidir con valores de la llave primaria a la que hace referencia, o bien, deben ser completamente nulos.

La regla de integridad referencial se enmarca en términos de estados de la base de datos: indica lo que es un estado ilegal, pero no dice cómo puede evitarse. La cuestión es ¿qué hacer si estando en un estado legal, llega una petición para realizar una operación que conduce a un estado ilegal? Existen dos opciones: rechazar la operación, o bien aceptar la operación y realizar operaciones adicionales compensatorias que conduzcan a un estado legal.

Por lo tanto, para cada llave ajena de la base de datos habrá que contestar a tres preguntas:

Regla de los nulos: ¿Tiene sentido que la clave ajena acepte nulos?

Regla de borrado: ¿Qué ocurre si se intenta borrar la tupla referenciada por la clave ajena?

Restringir: no se permite borrar la tupla referenciada.

Propagar: se borra la tupla referenciada y se propaga el borrado a las tuplas que la referencian mediante la clave ajena.

Anular: se borra la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).

Regla de modificación: ¿Qué ocurre si se intenta modificar el valor de la clave primaria de la tupla referenciada por la clave ajena?

Restringir: no se permite modificar el valor de la clave primaria de la tupla referenciada.

Propagar: se modifica el valor de la clave primaria de la tupla referenciada y se propaga la modificación a las tuplas que la referencian mediante la clave ajena.

Anular: se modifica la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).

Manejo de los Datos

Lenguajes relacionales

La tercera parte de un modelo de datos es la de la manipulación. Son varios los lenguajes utilizados por los SGBD relacionales para manejar las relaciones. Algunos de ellos son procedurales, lo que quiere decir que el usuario dice al sistema exactamente cómo debe manipular los datos. Otros son no procedurales, que significa que el usuario dice qué datos necesita, en lugar de decir cómo deben obtenerse.

La base de los lenguajes relacionales son el álgebra relacional que es un lenguaje procedural (de alto nivel) y el cálculo relacional que es un lenguaje no procedural.

Álgebra relacional

El álgebra relacional es un lenguaje formal con una serie de operadores que trabajan sobre una o varias relaciones para obtener otra relación resultado, sin que cambien las relaciones originales. Tanto los operandos como los resultados son relaciones, por lo que la salida de una operación puede ser la entrada de otra operación. Esto permite anidar expresiones del álgebra, del mismo modo que se pueden anidar las expresiones aritméticas. A esta propiedad se le denomina clausura: las relaciones son cerradas bajo el álgebra, del mismo modo que los números son cerrados bajo las operaciones aritméticas.

En este apartado se presentan los operadores del álgebra relacional de un modo informal. Las definiciones formales pueden encontrarse en la bibliografía.

Cood propuso originalmente ocho operadores, de estos, cinco son fundamentales: restricción, proyección, producto cartesiano, unión y diferencia, que permiten realizar la mayoría de las operaciones de obtención de datos. Los operadores no fundamentales son la concatenación (join), la intersección y la división, que se pueden expresar a partir de los cinco operadores fundamentales.

La restricción y la proyección son operaciones unarias porque operan sobre una sola relación. El resto de las operaciones son binarias porque trabajan sobre pares de relaciones.

Cálculo relacional

El álgebra relacional y el cálculo relacional son formalismos diferentes que representan distintos estilos de expresión del manejo de datos en el ámbito del modelo relacional. El álgebra relacional proporciona una serie de operaciones que se pueden usar para decir al sistema cómo construir la relación deseada a partir de las relaciones de la base de datos.

El cálculo relacional proporciona una notación para formular la definición de la relación deseada en términos de las relaciones de la base de datos.

El cálculo relacional toma su nombre del cálculo de predicados, que es una rama de la lógica. Hay dos tipos de cálculo relacional, el orientado a tuplas, propuesto por Codd, y el orientado a dominios, propuesto por otros autores.

En el cálculo de predicados (lógica de primer orden), un predicado es una función con argumentos que se puede evaluar a verdadero o falso. Cuando los argumentos se sustituyen por valores, la función lleva a una expresión denominada proposición, que puede ser verdadera o falsa. Por ejemplo, las frases

“Carlos Baeza es un miembro de la plantilla” y “Carlos Baeza gana más que Amelia Pastor” son proposiciones, ya que se puede determinar si son verdaderas o falsas. En el primer caso, la función “es un miembro de la plantilla” tiene un argumento (Carlos Baeza) y en el segundo caso, la función “gana más que” tiene dos argumentos (Carlos Baeza y Amelia Pastor).

Si F es un predicado, la siguiente expresión corresponde al conjunto de todos los valores de x para los que F es cierto:

x WHERE F(x)

Los predicados se pueden conectar mediante AND, OR y NOT para formar predicados compuestos.

Aunque el cálculo relacional es difícil de entender y de usar, tiene una propiedad muy atractiva: es un lenguaje no procedural. Esto ha hecho que se busquen técnicas no procedurales algo más sencillas, resultando en dos nuevas categorías de lenguajes relacionales: orientados a transformaciones y gráficos.

Los lenguajes orientados a transformaciones son lenguajes no procedurales que utilizan relaciones para transformar los datos de entrada en la salida deseada. Estos lenguajes tienen estructuras que son fáciles de utilizar y que permiten expresar lo que se desea en términos de lo que se conoce. Uno de estos lenguajes es *SQL (Structured Query Language).*

Los lenguajes gráficos visualizan en pantalla una fila vacía de cada una de las tablas que indica el usuario. El usuario rellena estas filas con un “ejemplo” de lo que desea y el sistema devuelve los datos que siguen tal ejemplo. Uno de estos lenguajes es *QBE (Query-by-Example)*.

1.4. ORACLE

Oracle es relacionado comúnmente como un sistema de gestión de bases de datos relacionales. Actualmente *Oracle* es una compañía que tiene una amplia gama de productos como son la base de datos, sus aplicaciones comerciales, sus herramientas de desarrollo de aplicaciones y sus herramientas para la toma de decisiones.

Dentro de Interceramic, se manejan varios de estos productos. Para el *ERP* en el cual se integrará el sistema a desarrollar, se interactuará con las siguientes tecnologías (Ver Fig. 1.8.):

- Base de Datos
- Servidor de Aplicaciones
- Herramientas de Desarrollo

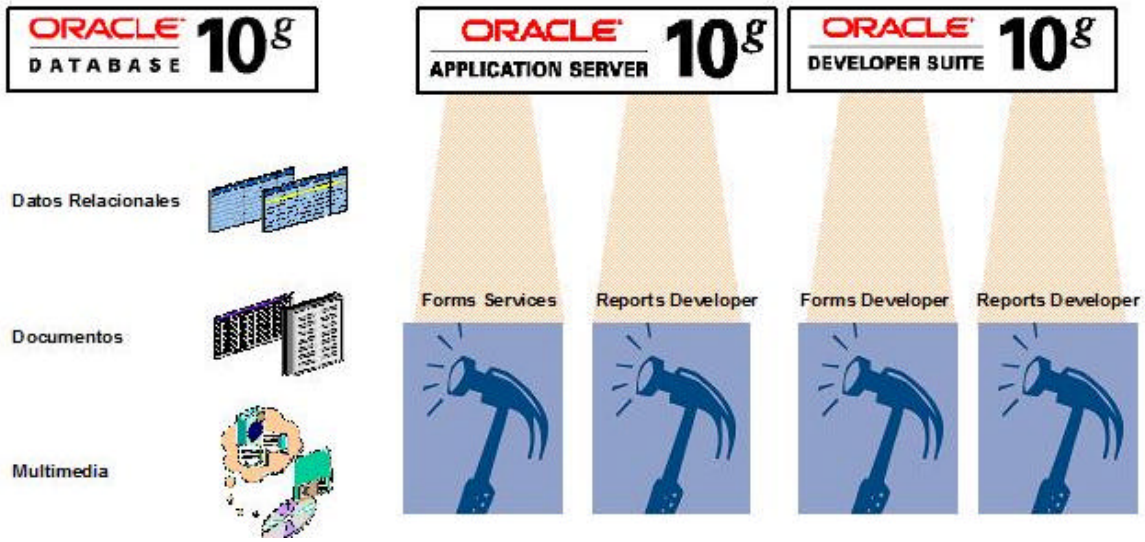

Figura 1.8. Elementos empleados para el desarrollo del Sistema.

La Base de Datos Oracle

Es un sistema de gestión de base de datos relacional. Se le considera uno de los más completos y se destaca por su:

Soporte de transacciones
Estabilidad
Escalabilidad
Soporte multiplataforma

En la base de datos se pueden alojar documentos como *Word*, *Excel*, imágenes, videos, etc. Tiene soporte para el manejo de lenguajes como *XML* y *JAVA*. El lenguaje nativo con el que se programa dentro de la base de datos es conocido como *PL/SQL*.

Oracle Application Server (OAS)

EL Servidor de Aplicaciones *Oracle* permite ejecutar aplicaciones Java, de tipo *Wireless*, portales y de inteligencia empresarial. El núcleo del OAS consiste en un servidor *HTTP* (basado en *Apache HTTP Server*) y su propia implementación de la especificación *J2EE* (*Java2 Enterprise Edition*) llamado *Oracle Containers for J2EE*. (*OC4J*).

Empleando Oracle Application Server se pueden publicar y administrar aplicaciones desarrolladas con la suite de desarrollo de Oracle.

Algunos de los servicios que contiene son los siguientes:

- *Oracle Portal*
- *Oracle Forms Services*
- *Oracle Reports Services*
- *Oracle Wireless*

- *Web Container*
- *Enterprise JavaBeans Container*

Oracle Developer Suite

Para aprovechar la infraestructura ofrecida por el *OAS* y la Base de Datos, *Oracle* ofrece una herramienta que permite a los desarrolladores construir rápida y fácilmente aplicaciones *e-business*. Toda la suite proporciona un entorno de desarrollo altamente productivo para la construcción de aplicaciones basado en ya sea en *Oracle Forms* o *Java* mediante *JDeveloper*. Para desarrollar aplicaciones de reporteo o análisis de información se puede recurrir a emplear herramientas como *Oracle Reports* u *Oracle Discoverer* que también son parte de la suite.

1.5. Planeadores de Recursos Empresariales (*ERP´s*)

La gestión de las empresas implica la adaptación constante a cambios que generan nuevos entornos y por ello se hace necesaria la ayuda tecnológica que permita que las empresas se anticipen y se adapten a las necesidades de los clientes. Con el surgimiento de los ERP's (*Enterprise Resource Planning*) en los 90's esto es posible ya que el ERP es una herramienta de software que ha ayudado a las empresas en la gestión de sus procesos. El ERP es una tecnología que se enfoca a los procesos internos de una empresa.

Entre las características que un ERP debe tener, están la optimización de los procesos empresariales, el acceso a información confiable, precisa y oportuna, la posibilidad de compartir información entre todos los componentes de la organización, la eliminación de datos y operaciones innecesarias y la reducción de tiempos y de los costos de los procesos.

El contar con una herramienta que conecte todas las áreas de información de una empresa es un valor que permitirá que se tenga éxito en el entorno actual que viven las organizaciones, pero se debe tener especial cuidado con el factor humano ya que el cambio que genera la implantación de este tipo de tecnologías en cualquier empresa es muy importante porque cambia la estructura organizacional, afecta la cultura y principalmente cambia el rol individual de los miembros.

Definición de ERP

ERP.- *Enterprise Resource Planning* o Planeador de Recursos de la Empresa, es un software que permite gestionar las áreas funcionales de la empresa, los proveedores de este tipo de software en México lo definen como "un sistema de gestión de información estructurado, diseñado para satisfacer soluciones de gestión empresarial". Entre sus principales características están la capacidad de modelar y automatizar la mayoría de los procesos básicos de una organización, desde la orden de venta, hasta la distribución del producto; la gama de funciones que cubren los *ERP* son: contabilidad, finanzas, administración de órdenes de venta, logística, producción y recursos humanos. Debemos tener en cuenta que las soluciones *ERP* se han especializado por segmentos de industria, lo que implica que la lógica para optimizar procesos está basada en modelos de operación específicos.

Beneficios de los *ERP*'s

Con la aparición de herramientas tecnológicas como los son los *ERP* que permiten la integración de la información, las empresas se ven beneficiadas en la administración de sus procesos. Se debe de tomar

en cuenta no solo el factor tecnológico sino también el tipo de organización, su giro, su misión, su visión, la cultura, las características generales de la empresa, ya que conociendo estos factores se puede determinar con mayor exactitud lo que realmente se desea cumplir con la implantación del sistema.

Es importante señalar que la correcta implantación de un *ERP* conlleva incrementos radicales de productividad así como la posibilidad de tener mejor información en la toma de decisiones.

Debilidades de los *ERP*'s

Mucho se ha dicho de lo que los *ERP* pueden hacer sin embargo existen pequeños huecos en el diseño que han salido a la luz, no para desacreditar este tipo de herramientas sino para hacer mención de las áreas que se pueden reforzar y corregir. Unas de estas debilidades que podemos mencionar son la complejidad del proceso para integrar información contenida en sistemas externos al *ERP* ó en aplicaciones distribuidas geográficamente, así también la eficiencia de operación de los sistemas se afecta considerablemente si se mantiene en las bases de datos la historia de las transacciones.

Situación Actual de los *ERP*'s

La implantación de un *ERP*, es una decisión que la empresa toma no sólo para obtener pequeños resultados sino resultados que realmente trasciendan en el crecimiento de la empresa, un *ERP* no es una moda más de la economía actual sino una necesidad tecnológica que día a día se esta presentando en muchas empresas y es por ello la necesidad de conocer los alcances y limitaciones de software de este tipo.

Algunos de los *ERP* más comunes son los siguientes: *Oracle E-Bussines, SAP, J.D. Edwards, BaaN, People Soft, Exactus, Platinum, Solomon, MicroSIP*, (estos últimos sólo cubren funciones de administración y nóminas).

CAPITULO 2

ANALISIS Y DISEÑO DEL SISTEMA

2.1. Antecedentes

Interceramic en su departamento de Tecnologías de Información ha adoptado el uso de *ERP*'s para los procesos de ventas, productivos (Manufactura) y servicios (Franquicias). En el proceso productivo adquirió una solución comercial (*ERP Oracle E-Business*), y para el manejo de la información de las Franquicias optó por crear un *ERP* basado en *Oracle* con la finalidad de integrar la información entre ambos *ERP*'s. El *ERP* para Franquicias se ha desarrollado de acuerdo a los procesos y necesidades de las mismas; concentra información de Ventas, Compras, Cuentas por Pagar, Cuentas por Cobrar, Inventarios, Nómina, Recursos Humanos, etc.

2.2. Planteamiento del problema

La implementación del sistema de Planeación de Recursos Empresariales (*ERP*) para el manejo de los procesos productivos y operativos de las franquicias, ha brindado a Interceramic resultados satisfactorios, ejemplo de ello es la administración centralizada de la información y la estandarización de los procesos que llevan a cabo las franquicias. Sin embargo, la necesidad de actualizar y mejorar la administración y operación de sus procesos hace que continuamente el *ERP* sea modificado o se le integren nuevos subsistemas.

La forma de administrar los cambios, correcciones y mejoras que se realizan al *ERP* no cumple con los lineamientos marcados por la nueva plataforma de Software. Aunado a ello, el número de personas que laboran en el departamento de Sistemas creció y la administración del personal y proyectos derivados del mantenimiento al *ERP* se volvió más compleja.

De forma irónica, Interceramic cuenta con un sistema para hacer más eficientes los procesos y alinear las prácticas de las franquicias con respecto a su forma de trabajar, sin embargo ahora carecía de un control sobre los cambios que se llevan a cabo en el sistema.

Los problemas que se derivaron al no contar con un proceso estandarizado de modificación de programas, fueron varios, el más grave era la continua sobre escritura de los mismos, suplantando con ello alguna funcionalidad previamente programada. Otro problema era la falta de conocimiento con respecto a los programas que se estaban modificando y por quien; ya que a pesar de llevar un control con una hoja de cálculo, era muy común que a los encargados se les olvidara registrar dicha información.

Ya que no se contaba con una herramienta que permitiera administrar al personal y los proyectos de mantenimiento en los que trabajan, era muy común que los tiempos de entrega de estos se alargaran, o peor aún, solicitudes generadas por los usuarios para realizar cierta modificación o mejora al sistema nunca se llevaran a cabo debido a que no se tenia una base de datos en la cual registrar estos requerimientos. También sucedía con frecuencia que más de una persona estuviera realizando un análisis en paralelo o incluso programando alguna solución a partir de una petición de usuario, conllevando con esto pérdida de tiempo, falta de comunicación y re-trabajo.

Tomando en consideración estos problemas surgió la necesidad de crear un sistema que permita concentrar, administrar y gestionar toda la información que se ve involucrada en el proceso de creación de nuevos proyectos y mantenimiento del *ERP*.

El sistema que de ahora en adelante llamaremos "Control del Sistema y Proyectos (CDS)" pretende concentrar y agilizar la captura de requerimientos y de paso eliminar los problemas provocados por la falta de administración en los cambios o adecuaciones al sistema. Otro de los objetivos es facilitar la toma de decisiones tanto en la asignación de proyectos de software derivados de los requerimientos capturados, como en el seguimiento de los mismos. También se creará un mecanismo de alertas automatizado para el seguimiento de proyectos, tareas de programación, etc.

La información relacionada con este sistema se almacenará en la base de datos con la que trabaja el *ERP* (*Oracle*). La aplicación se desarrollará siguiendo los estándares y plataforma de desarrollo actual, cumpliendo con la misión dentro del departamento de sistemas que consiste en la integración de todas sus tecnologías.

2.3. Recolección de Requerimientos del Sistema

Debido al contexto en el cual trabajará el sistema a crear, los requerimientos se dividieron en 2 clasificaciones, la primera tiene que ver con la integración del sistema y su desarrollo; la segunda trata acerca de los procesos que se deben llevar a cabo.

2.3.1. Requerimientos de Integración y Desarrollo

El principal requerimiento por parte del gerente de sistemas de Franquicias Interceramic es contar con un sistema que se integre dentro del *ERP*. Para esto es necesario conocer como esta compuesto el *ERP*, la información que se maneja en el mismo y las reglas que rigen los sistemas que contiene.

Como se ha descrito anteriormente, un *ERP*, es un sistema computarizado, donde su tamaño y complejidad dependen del ámbito y empresa donde se este empleando. Como todo sistema de cómputo, un *ERP* se compone hablando en términos de software de una interfaz de usuario y un motor de base de datos. Con lo que respecta al hardware las configuraciones de almacenamiento, instalación de la infraestructura de base de datos y de la capa de aplicación (interfaz de usuario) dependen también de las necesidades de la empresa y obviamente de la demanda a la cual sea sometido el *ERP*.

Actualmente el *ERP* esta compuesto por:

Hardware

Servidores para alojar las instancias de Base de Datos del *ERP*. (5)
Servidores para contener la capa de Aplicación, comúnmente llamada capa de presentación. (5)
Una *SAN* para la comunicación entre los servidores y los dispositivos de almacenamiento de la información.

El **Software** sobre el que trabaja el *ERP* es el siguiente:

Sistema Operativo: *Red Hat Enterprise Linux 4.0*
Base de Datos: *Oracle 9.2.0.8.0* en *RAC.*

Aplicación: *Oracle Aplication Server 10g R2*. (Componentes activados para el *ERP Forms* y *Reports Services*)

Con respecto a la distribución de la ejecución del ERP; se esta empleando el modelo de 3 capas (en cada capa se instala un componente de software). El cual consiste en:

- 1ra. Capa (*Front End*): donde se ejecutan las *IU* (interfases de usuario) del cliente. **Los Browsers.**
- 2da. Capa (*MiddleWare*): Recibe solicitudes de las *IU* a través de la red. Estos son mensajes que se envían mediante protocolos de transporte. (*HTTP, TCP*, etc). **El servidor Web.**
- 3ra. Capa (*BackEnd*): Almacena la información. **El servidor de Base de Datos.**

2.3.2. Funcionamiento del ERP

Uno de los beneficios de trabajar con Bases de Datos *Oracle* es la creación y gestión de usuarios, sus mecanismos de seguridad basados en roles y privilegios, el uso de perfiles (llamados *profiles*; cuya misión es fijar límites en aspectos relacionados con el uso de recursos, por ejemplo: tiempo de inactividad por sesión).

A los usuarios se les dota con ciertos privilegios para:

Poder acceder a la Base de Datos (*login*).
Realizar ciertas acciones (insertar, actualizar, eliminar, seleccionar, ejecutar).
Crear objetos dentro de la Base de Datos.

Para poder comprender el funcionamiento del *ERP*, es necesario contar con las siguientes definiciones:

Objeto de Base de Datos.- tablas, procedimientos, funciones, paquetes. [A4]

Usuario.- Colección de objetos y privilegios identificado con un nombre y *password*. Cuando un usuario tiene al menos creado un objeto de base de datos en su cuenta, se le conoce como **"Esquema".**

Privilegios.- Permiso para realizar una acción; este se asigna a un usuario o un rol.

Roles de Base de Datos.- Conjunto de privilegios; se otorgan a un usuario o un rol.

Roles de Sistema o Aplicación.- Conjunto de privilegios, asignables a un usuario o rol que son propios de la lógica del negocio.

El *ERP* de Interceramic cuenta con esquemas y usuarios, a éstos últimos se les denomina usuarios de aplicación. También se basa en la asignación de roles de Base de Datos y Roles de Sistema o Aplicación. Los dos tipos de roles se asignan a cada uno de los usuarios de aplicación existentes.

En la figura 2.1 se muestra como se interrelacionan las definiciones descritas anteriormente para proporcionar un sistema versátil, robusto y auditable.

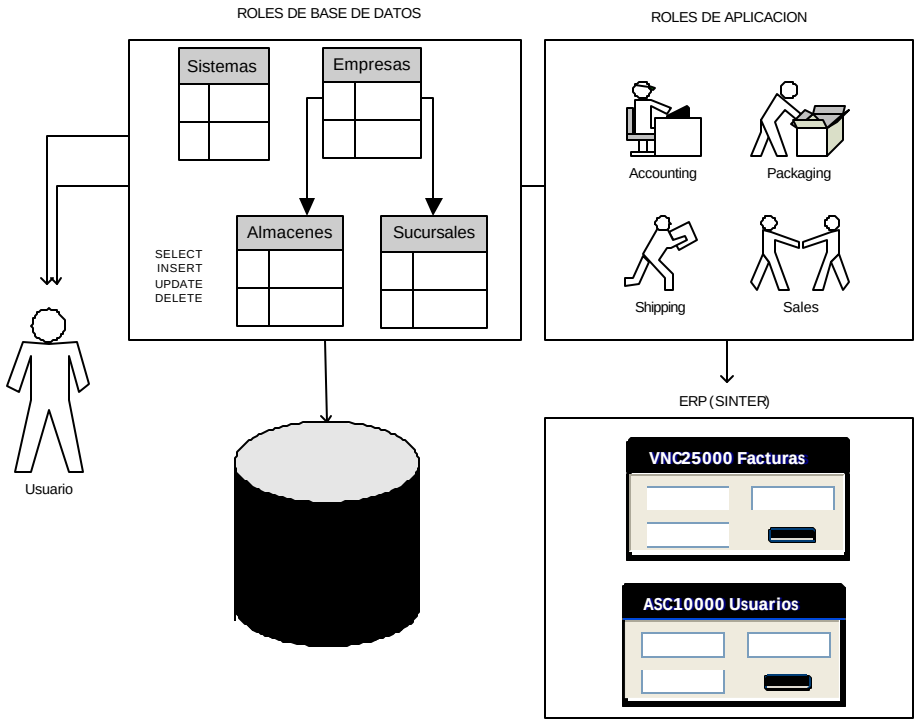

Figura 2.1. Modelo basado en roles del ERP.

Es bien sabido que los mecanismos de seguridad son la base para cualquier sistema de información y aún más cuando se trata de un *ERP*, es por ello que el nuevo sistema deberá contemplar el uso de los roles de aplicación y base de datos.

En lo que respecta a las herramientas de desarrollo con las que se programará el sistema se emplearán las mismas que utiliza el departamento de sistemas. (*Oracle Developer Suite 10g R2*, para la capa de aplicación y para la capa de Base de Datos, *Oracle 9i* en su versión 9.2.0.8.0).

Al quedar integrado dentro del *ERP*, se cumplirá con otro requerimiento, que es el de trabajar bajo el esquema de 3 capas.

2.3.3 Requerimientos de Proceso

Continuando con la recolección de requerimientos para poder implantar el sistema de "Control de Sistemas y Proyectos" fue necesario realizar una inspección al proceso real para poder identificar las diversas fases por las que pasa un requerimiento de software. Para ellos se procedió a documentar las actividades del personal, también se entrevistó a la gente involucrada en el proceso para concentrar y analizar los requerimientos.

Nombre del Cliente: Internacional de Cerámica S.A. de C.V. (Interceramic)
Nombre o Unidad Administrativa: Tecnologías de Información
Departamento de Servicio: Franquicias (SINTER NG)

El departamento de Franquicias es parte del Área de Tecnologías de Información de Interceramic y se encarga de administrar, atender y dar respuesta a las demandas y necesidades de los usuarios del ERP

de todas las franquicias de la República Mexicana y Centroamérica. Físicamente se ubica en la ciudad de Chihuahua, Chihuahua con dirección en Av. Universidad No. 1306 Col. Santo Niño.

Personal del Departamento de Franquicias

Es de suma importancia dentro del análisis de sistemas la identificación del personal que esta inmiscuido en el proceso que se pretende automatizar, dentro de esta etapa se encontraron a los siguientes actores:

Puesto: Gerente de Sistemas de Franquicias

Depende de: Director de Tecnologías de Información

Supervisa a: Coordinador de Proyectos, Coordinador de Desarrollo, Analistas de Sistemas y Desarrolladores.

Número de puestos con estas características: 1

Descripción de Actividades: Supervisa las diversas actividades del personal a su cargo, asigna proyectos y define sus prioridades.

Puesto: Coordinador de proyectos

Depende de: Gerente de Sistemas de Franquicias

Supervisa a: Analistas de Sistemas.

Número de puestos con estas características: 1

Descripción de Actividades: Supervisa las diversas actividades del personal a su cargo, verifica que los proyectos asignados a los analistas se cumplan en tiempo y forma, y asigna junto con el Gerente de Sistemas los proyectos derivados de los diversos requerimientos.

Puesto: Coordinador de Desarrollo

Depende de: Gerente de Sistemas de Franquicias

Supervisa a: Desarrolladores.

Número de puestos con estas características: 1

Descripción de Actividades: Supervisa las diversas actividades del personal a su cargo, verifica que las tareas asignadas a los desarrolladores se cumplan en tiempo y sigan los estándares de código y apariencia del ERP. Sirve como enlace entre los analistas de Sistemas y Desarrolladores cuando se presentan problemas técnicos o cuando no se tienen los datos necesarios para desarrollar las tareas.

Puesto: Analistas de Sistemas

Depende de: Gerente de Sistemas de Franquicias, Coordinador de Proyectos

Supervisa a: El desarrollador(es) asignado a su proyecto.

Número de puestos con estas características: 12

Descripción de Actividades: Encargado de atender directamente a los usuarios del sistema para solucionar sus dudas o en caso de necesitarlo auxiliarles en los procesos operativos u administrativos. Ejemplo de estos procesos son el seguimiento y manejo de la cobranza, creación y liberación de embarques, compras de material y alta en las bodegas, pago de la nómina, etc. Debido a su conocimiento en los procesos que se llevan a cabo en el ERP, son los responsables de realizar el análisis e impacto en las modificaciones o desarrollo de nuevos proyectos.

Puesto: Desarrollador

Depende de: Coordinador de Desarrollo, Analistas

Supervisa a: No aplica
Número de puestos con estas características: 8
Descripción de Actividades: Encargado de crear o en su defecto modificar los programas solicitados por el analista de sistemas. Son responsables de la codificación de los programas.

El siguiente organigrama (Fig. 2.2) muestra las jerarquías existentes dentro del Departamento de Sistemas de Franquicias. Los puestos sombreados identifican a los usuarios relacionados directamente con el proceso.

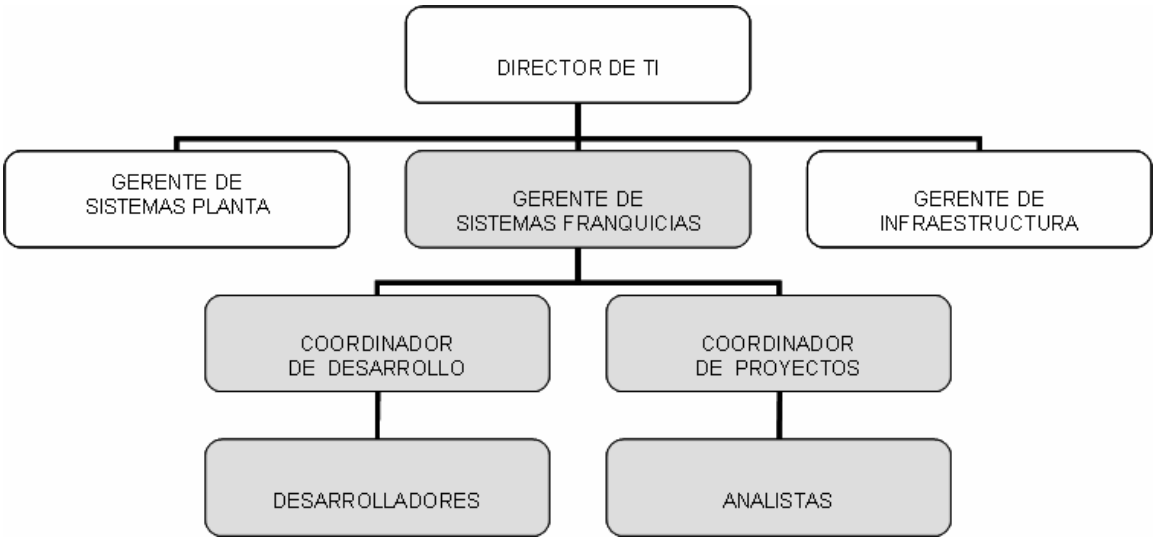

Figura 2.2. Organigrama del Departamento de Sistemas antes del análisis.

2.3.4. Descripción de los procedimientos

El procedimiento mediante el cual se trabajaba anteriormente no estaba bien definido y por lo tanto era susceptible a errores. No obstante sirvió como base para el desarrollo del sistema.

La forma de operar del sistema anterior permitía capturar los requerimientos de usuarios mediante una aplicación desarrollada en *Visual Basic* y a veces en *Excel*. La información se almacenaba en una base de datos *SQL Server* [A1]. Para utilizar la aplicación era necesario instalar en la maquina cliente el ejecutable de dicho programa. El acceso a la base de datos se realizaba mediante *ODBC* [A2], para ello se configuraba la conexión hacia el servidor donde se tenía la Base de Datos en cada computadora, en dado caso de que se cambiará el servidor, se tenía que reconfigurar esta parte para emplear el sistema.

Otro de los inconvenientes de trabajar de esta forma era que cada vez que surgía la necesidad de realizar un cambio a la aplicación, se tenía que volver a instalar el programa ejecutable en cada una de las máquinas. Además, el sistema solo servía para concentrar los requerimientos y capturar los programas que se iban modificando o creando en cada requerimiento.

La forma de operar era la siguiente:

Procedimiento: Captura de requerimientos

La información relevante en la captura de requerimientos era la siguiente:

Fecha de captura del requerimiento
Usuario que solicitó el requerimiento
Comentarios

Procedimiento: Asignación de requerimientos

Una vez capturada esta información en el sistema, se imprimía el requerimiento y se depositaba en una bandeja. La prioridad en primera instancia era dictada por el orden en el cual se depositaban. Posteriormente el Gerente de Sistemas se reunía con un comité de asignación de proyectos (el cual estaba formado por el Coordinador de Proyectos y algunos analistas) para revisar los requerimientos y asignarlos a los analistas y desarrolladores que se harían cargo del análisis y programación respectivamente.

Procedimiento: Petición de objetos

Una vez que se asignaban los requerimientos a los analistas, estos se encargaban de pedirle a otro analista (que fungía como encargado de fuentes [A3]) que le proporcionara los programas a modificar por el desarrollador. Este último capturaba en un archivo en Excel los programas que serían modificados bajo previa verificación de que no estaban registrados en algún otro requerimiento.

Procedimiento: Validación de requerimiento

Ya que se tenían los programas finalizados. El analista contactaba al usuario que había realizado el requerimiento y le pedía que validara dichos cambios en un ambiente de pruebas. Una vez que el usuario estaba de acuerdo con el funcionamiento del sistema, le enviaba un e-mail aprobando que dichos cambios se instalaran en el *ERP*.

Procedimiento: Pase a producción del requerimiento

Ya con la aprobación del usuario (e-mail impreso), el analista anexaba este correo junto con otro documento que contenía los programas modificados (Se realizaba mediante un formulario de la aplicación y posteriormente se imprimía, o se llenaba a mano).

Con este último documento llamado “pase a producción”, se procedía a recolectar las firmas de todas las personas que habían estado involucradas en el requerimiento (analistas y programadores) para que se llevara a cabo la instalación correspondiente en los ambientes de producción.

En las figuras siguientes (2.3 a 2.5) se puede observar como se llevaba a cabo el proceso de los requerimientos.

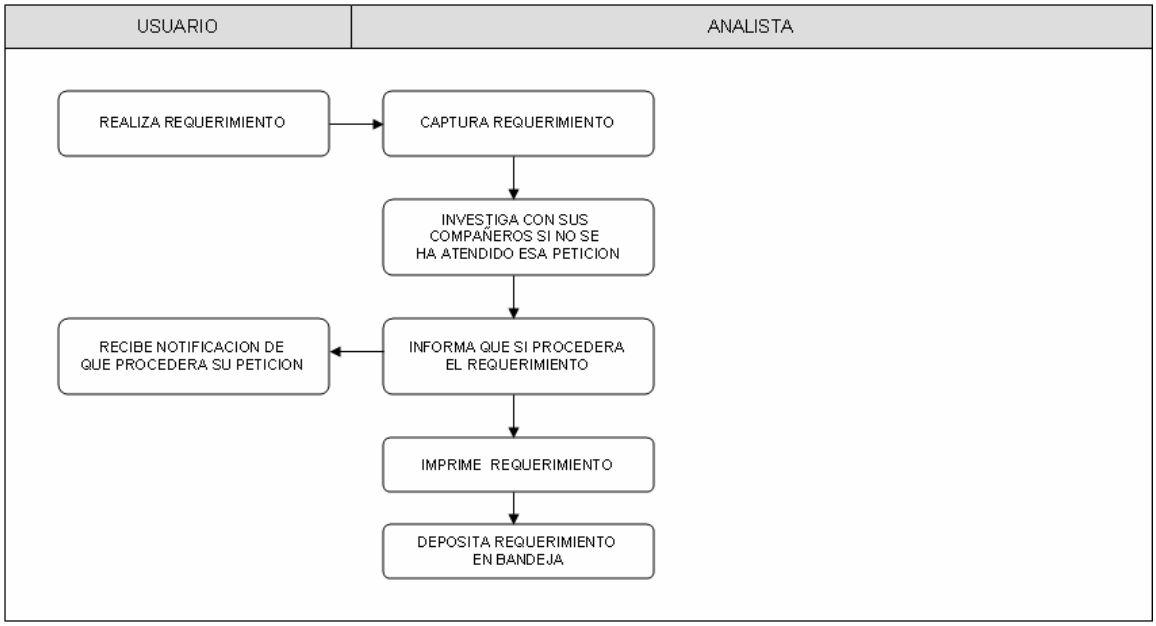

Figura 2.3. Captura de requerimiento.

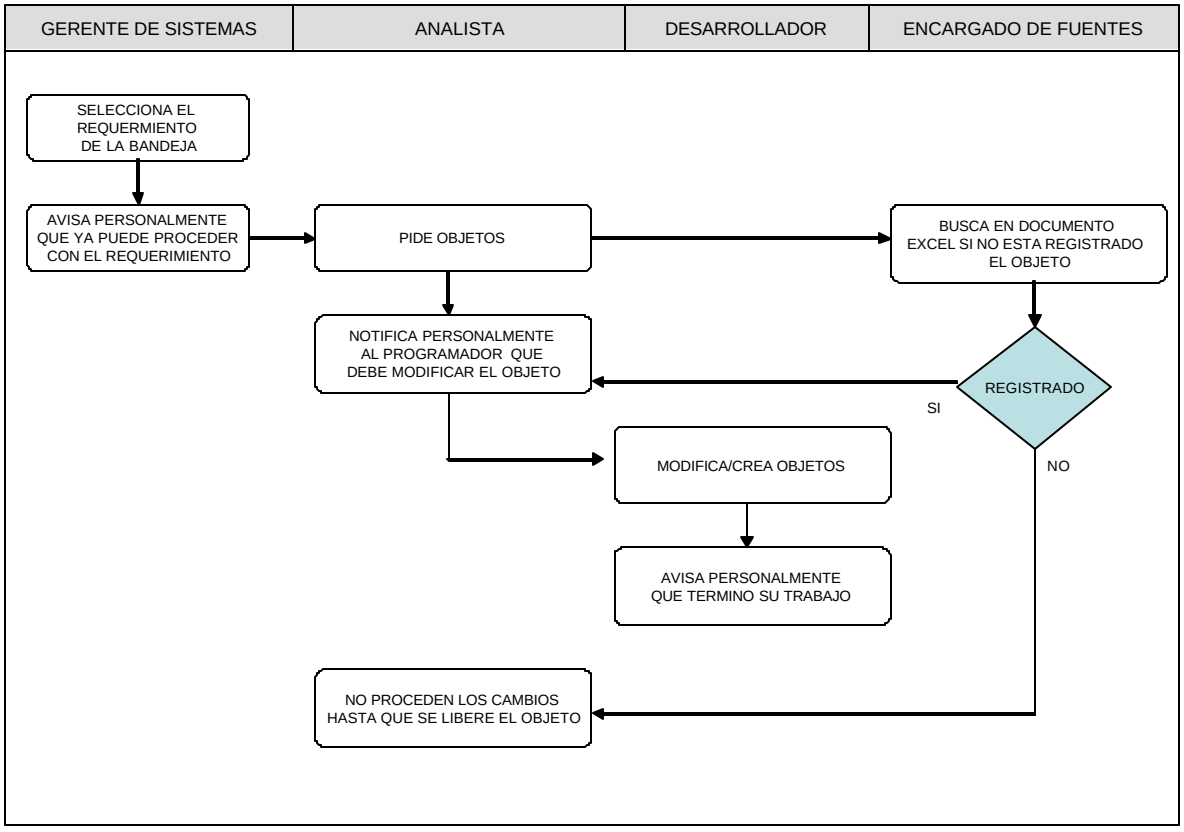

Figura 2.4. Notificación de asignación de requerimiento, petición de objeto(s) y desarrollo.

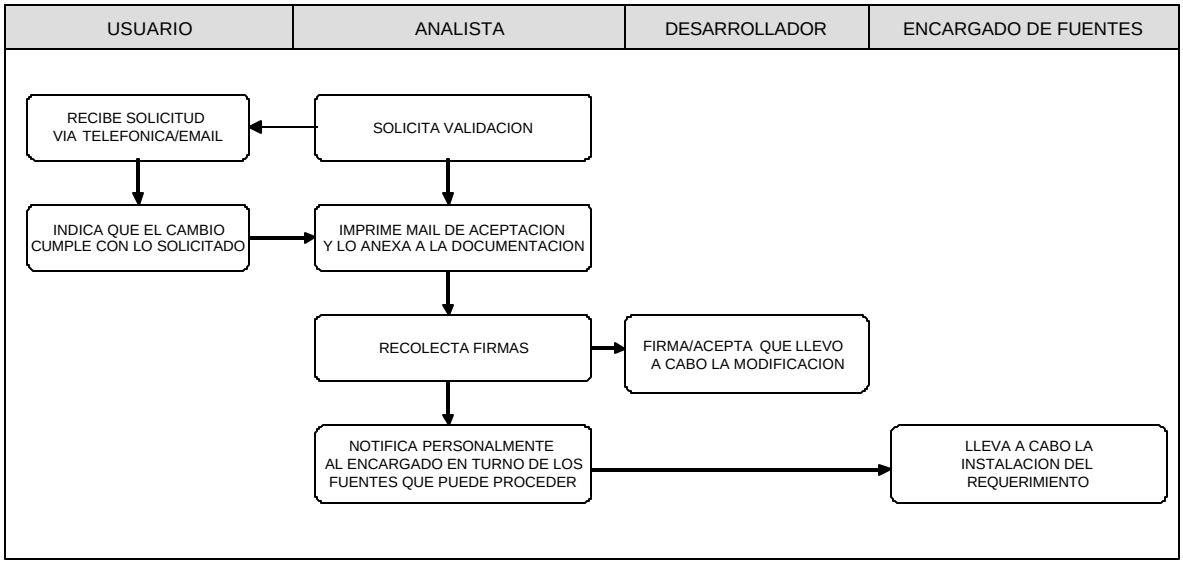

Figura 2.5. Petición de validación por parte de usuario y pase a producción.

Una vez que se comprendió el proceso, se procedió a recolectar las necesidades y puntos de mejora en el procedimiento anteriormente descrito. Para ello se entrevistó al Gerente de Sistemas y los coordinadores del Departamento de Sistemas de Franquicias.

Las necesidades que existían eran las siguientes:

1. Mantener la uniformidad de la aplicación (basándose en los estándares de programación ya definidos).
2. Que la aplicación fuera de ágil manejo para la captura de requerimientos y los procesos que surgieran a partir de ellos.
3. Automatizar las asignaciones de proyectos, tareas de programación y petición de programas.
4. Contar con un proceso formal tanto de validaciones, aprobaciones e instalaciones en ambientes de producción.
5. Apoyarse en la aplicación para administrar de mejor manera al personal que labora en el departamento.
6. Evitar en la medida de lo posible la impresión de documentos.
7. Que el sistema contara con los reportes necesarios para presentar a los auditores, ya que éstos evalúan los procedimientos realizados en la empresa con respecto a la captura de requerimientos, validación e instalación de los mismos.

2.3.5. Análisis de Factibilidad

Una vez que se ha definido la problemática y se han establecido las causas que ameritan un nuevo sistema, se llevo a cabo el análisis de factibilidad correspondiente. Éste toma en cuenta las capacidades técnicas y de infraestructura que implica el desarrollo e implantación del sistema en cuestión, así como la relación costo-beneficio y el grado de aceptación de los usuarios.

Factibilidad Técnica

Dentro de la recolección de requerimientos se obtuvo información sobre los componentes técnicos que posee la organización y se analizó la posibilidad de emplear tanto el hardware y software con el que cuenta la empresa para desarrollar el sistema propuesto.

Los servidores de base de datos y de aplicación, así como las computadoras existentes dentro del departamento, la infraestructura de red y el software con el que se cuenta para el desarrollo y mantenimiento del ERP soportan sin ningún problema un nuevo sistema, ya que el número de usuarios que lo emplearan no rebasa los 35. (Actualmente el ERP trabaja con 2500 usuarios concurrentes, el uso de cpu y carga promedio dentro de los servidores esta a mitad de su capacidad.).

Análisis de Costo - Beneficio

Costos generales

Los gastos derivados por los procesos que se llevaban a cabo estaban principalmente destinados a papelería, consumibles y organizadores para la documentación.

Costos de Personal

Al no contar con una herramienta que permitiera llevar el control adecuado de los requerimientos por parte de los usuarios ya sea para mejoras, modificaciones o incluso nuevos proyectos, el departamento de Sistemas de Franquicias empleaba mucho tiempo en las actividades administrativas y de manejo de recursos humanos.

Costo del sistema Propuesto

El sistema automatizado involucra los siguientes costos:

Costos generales

Interceramic es una empresa “Green”, es decir, se preocupa por el medio ambiente, por tal motivo, al automatizar los procesos y mantenerlos en medios electrónicos, se esta eliminando en un alto porcentaje el uso de los materiales mencionados anteriormente. Debido a que el sistema propuesto se integrará al ERP actual, se tienen otras ventajas, como son:

- Inversión nula ya que se empleara el mismo hardware para soportar el sistema.
- Para desarrollar y mantener la aplicación se ocupará el software con el que cuenta Interceramic,
- Al ser una aplicación que se emplea a través de Internet, solo bastara dar de alta los requerimientos y dar seguimiento a los mismos accediendo al ERP, sin importar el lugar donde se encuentren.

Costos de Hardware y Software

Debido a que la organización cuenta con los equipos y recursos técnicos suficientes, el costo del desarrollo e implantación es nulo.

Costos de Personal

El sistema propuesto no incluyo variaciones en el personal ni costo alguno ya que es un proyecto elaborado como trabajo de grado.

Beneficios

Reducción de costos en papelería, artículos de oficina y suministros para los equipos empleados.

Cabe destacar que al automatizar los procesos de seguimiento y control, se reducen y se aligeran las cargas laborales de personal que normalmente dedicaba buena parte de su jornada laboral a atender todo lo relacionado a estos procesos y por consecuencia este tiempo ahora lo pueden invertir en otras actividades

Relación Costo – Beneficio

Se presentan grandes ventajas con respecto a los costos, ya que como se ha mencionado anteriormente, la inversión por parte de la empresa es nula. Otra ventaja es la administración del personal tanto de sus actividades como de la información que se desencadena a partir de los requerimientos. De esta forma aumentara la competitividad del departamento y por consecuencia de la empresa. Ya que se disminuirán las actividades redundantes y se proporcionara agilidad en el desarrollo de las actividades involucradas.

Factibilidad Operativa

La necesidad de automatización de los procesos que se llevan actualmente dentro del departamento de sistemas, lleva consigo la aceptación del nuevo sistema. Adicionalmente, si son necesarias adecuaciones que permitan una mejor administración de las tareas y del personal, éstas se realizarán sin mayor problema puesto que el sistema se desarrollará bajo los mismos estándares que existen actualmente para los demás sistemas del ERP

En base a lo mencionado anteriormente se llego a la conclusión de que era posible generar una aplicación integral cuyo objetivo se resume en la administración y automatización del manejo de proyectos, generados a partir de requerimientos de software.

2.4. Componentes del Sistema

Una vez recolectados los requerimientos por parte de los usuarios que emplearan el sistema, es necesario explicar como esta conformado el *ERP* al cual se integrará el sistema a desarrollar.

2.4.1. Funcionalidad del *ERP*

El ERP de Interceramic se conforma por varios sistemas. Estos sistemas interactúan entre sí. Hay algunos que lo hacen en línea dada la naturaleza de su proceso, otros en cambio, lo hacen solo para depositar o tomar cierta información.

Por ejemplo, en un proceso de venta de material con tipo de pago "CONTADO", se ven implicados varios sistemas. A continuación se detalla uno de tantos procesos y flujos por los que pasa la información (Ver Fig. 2.6.):

1. El cliente indica al vendedor los artículos a comprar.
2. El vendedor crea un documento llamado "FACTURA" y captura los artículos en el sistema.
3. Por cada artículo capturado, se verifica su existencia en las bodegas.
4. Por cada artículo que exista en la bodega, se descuenta la existencia física.
5. Una vez que se han capturado todos los artículos, se confirma la venta.
6. Al confirmarse la venta, se genera un nuevo documento llamado "CUENTA X COBRAR".
7. En la caja de la sucursal, el cliente efectúa el pago. Para esto, el cajero selecciona la "CUENTA X COBRAR" asociada a la "FACTURA", captura el "TIPO DE PAGO" y confirma el cobro.
8. Cuando se confirma el cobro, la mercancía se libera para su entrega en la bodega.

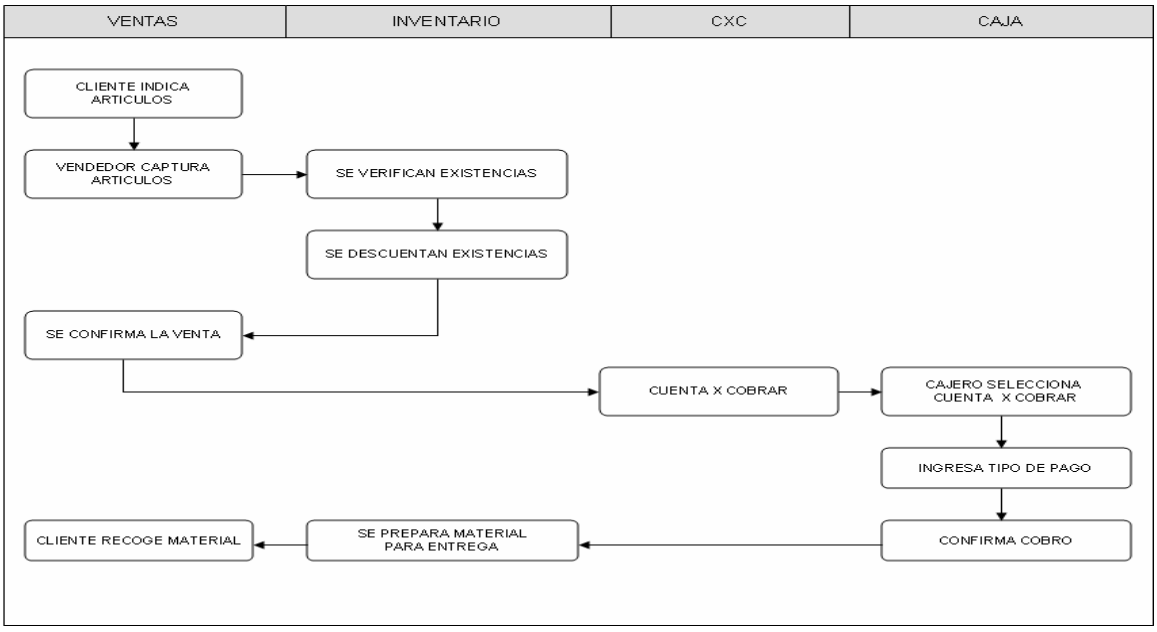

Figura 2.6. Ejemplo de un proceso de Venta.

Los sistemas (Ver Tabla 2.1) que actualmente tiene el ERP son los siguientes:

SISTEMA	NOMBRE
1	Contabilidad
2	Bancos
3	Nómina
4	Recursos Humanos
5	Activos Fijos
6	Ventas
7	Compras
8	Inventarios
9	Cuentas por Cobrar
10	Cuentas por Pagar
11	Caja
12	Control de Obras y Prospectos
13	Ubicaciones
14	Inventario Físico
15	Reparto y Entrega a Domicilio
16	**Administración del Sistema (CAS)**
17	SIFE
18	Caja de Ahorro
19	Especificaciones
20	Presupuestos
41	Gastos Médicos Mayores
42	Servicio Médico
43	Lista de Asistencia (Tiempo Extra)
44	Seguro de Autos
45	Venta a Empleados
46	Bodegas de Liquidación
47	Objetivos
48	Becas
49	Puntos
50	Seguridad y Ecología
51	Seguros
52	Caja Chica
53	Evaluación del Desempeño
54	Resguardos y Flotillas

Tabla 2.1. Sistemas existentes en el ERP.

La forma de trabajar del *ERP* es totalmente configurable y dependiendo de las necesidades de las franquicias se les habilitan los sistemas que soliciten. Por ejemplo la Franquicia de Villahermosa administra la nómina de sus empleados con otro sistema, motivo por el cual no se le configuran los sistemas 3 y 4 referentes a este manejo.

El sistema identificado con el número 16 (Administración del Sistema, también llamado CAS) se encarga de mantener y gestionar la información genérica del *ERP*; como por ejemplo, la administración de países, monedas, tipos de documentos, empresas, sucursales, bodegas, sistemas y módulos (módulos forma y módulos reporte) que lo componen, etc. Este sistema también se encarga de contener la lógica y reglas relacionadas con el acceso al *ERP*. Para poder acceder al mismo, se necesita contar con usuario creado en la Base de Datos y registrado dentro de las tablas del sistema del CAS. Pero eso no es todo,

puesto que como se comento en el capitulo anterior, para poder llevar a cabo una acción dentro del *ERP*, el usuario debe tener asignados ciertos roles (tanto de base de datos como de aplicación), los cuales dependerán de la función que lleva a cabo dentro de su Empresa.

Es importante mencionar que el sistema a desarrollar interactúa de forma directa con el sistema identificado como CAS por lo siguiente:

Integración:

- Cada usuario del sistema a desarrollar ya tiene su respectiva cuenta de usuario creada dentro del CAS.

- Para poder llevar a cabo las funcionalidades requeridas dentro del nuevo sistema, se crearan nuevos roles, los cuales se definirán dentro del CAS.

- Estos roles se asignarán a las cuentas de los usuarios que emplearán el nuevo sistema.

Funcionalidad:

- Una de las razones fundamentales para el desarrollo del nuevo sistema "CDS", es tener registradas las modificaciones o creación de objetos dentro del *ERP*.

- La información de las modificaciones de aplicación se tomaran del CAS y cuando se trate de objetos nuevos dentro del *ERP*, previamente se registraran en el CDS y posteriormente cuando se realice el pase a producción quedaran almacenados en el CAS.

- Para el caso de modificaciones de objetos de base de datos, como son, procedimientos, funciones, paquetes, especificaciones de paquete, etc. Se consultará directamente el diccionario de la base de datos del *ERP* filtrando solo aquellos usuarios (esquemas) que contienen objetos del mismo. Para poder llevar a cabo tal filtro, se empleará la tabla de sistemas que pertenece al CAS.

Ya se ha mostrado la lista de sistemas que componen el *ERP* y la funcionalidad del CAS, pero aún falta indicar en donde se almacenan los diferentes programas que componen cada sistema, los roles de base de datos, los roles de aplicación, los privilegios que existen sobre cada programa de la aplicación, etc. Para ello, a continuación se muestra un diagrama entidad relación de algunas de las tablas del esquema del CAS.

Para evitar confusiones entre los conceptos de sistema y esquema, se procederá a ejemplificar como se relacionan estos 2 conceptos. Ver Fig. 2.7.

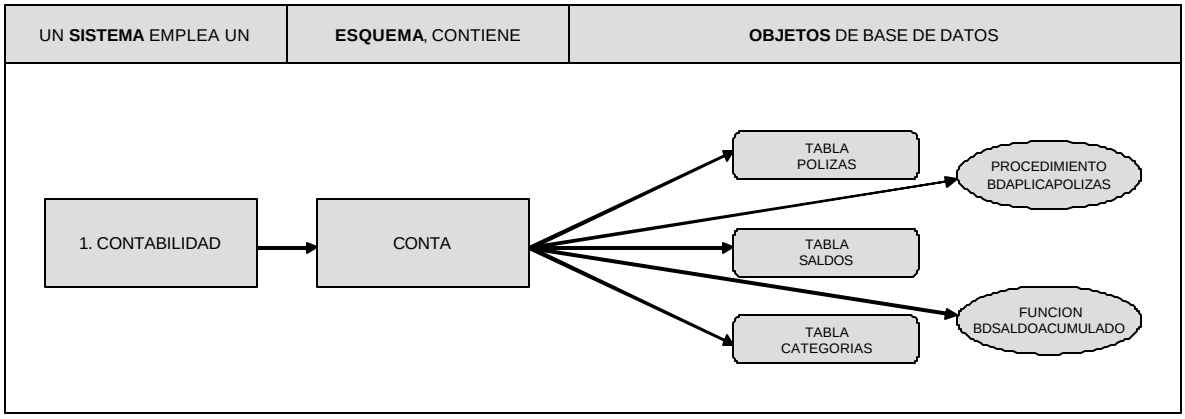

Figura 2.7. Relación entre sistema y esquema dentro del *ERP*.

Como se muestra en la figura anterior, un sistema es un identificador dentro del *ERP* que sirve para agrupar lógicamente procesos que tienen relación, este a su vez, utiliza un esquema para contener los objetos de base de datos que son necesarios para almacenar o procesar la información derivada del sistema. La información de la relación entre un sistema y un esquema se almacena en la tabla de sistemas, en los campos: sisnum, esquema. Para el caso del CDS, se creará su sistema y esquema relacionado.

Los módulos son los programas de aplicación que contiene cada sistema y permiten al usuario interactuar con el *ERP*, ya sea para almacenar, procesar o extraer información. Éstos se almacenan dentro de la tabla de módulos y junto con las tablas de roles y privilegios, se definen las reglas de acceso y acciones que se tienen para trabajar con la aplicación.

La clasificación de los módulos es la siguiente:

F- Forma
R – Reporte
M – Menú
S - Submenú
O- Menú Raíz

Acciones:

F2 – Grabar registro o cambios.
F3 – Eliminar registro.
F4 – Generar reporte.
F5 – Llamar a un módulo relacionado.
F6 – Llamar a menú principal.

2.4.2. Descripción del Sistema CDS

El CDS presentará una interfaz de usuario como cualquier otro sistema del *ERP* y estará compuesto lógicamente por los siguientes módulos (Ver Fig. 2.8.):

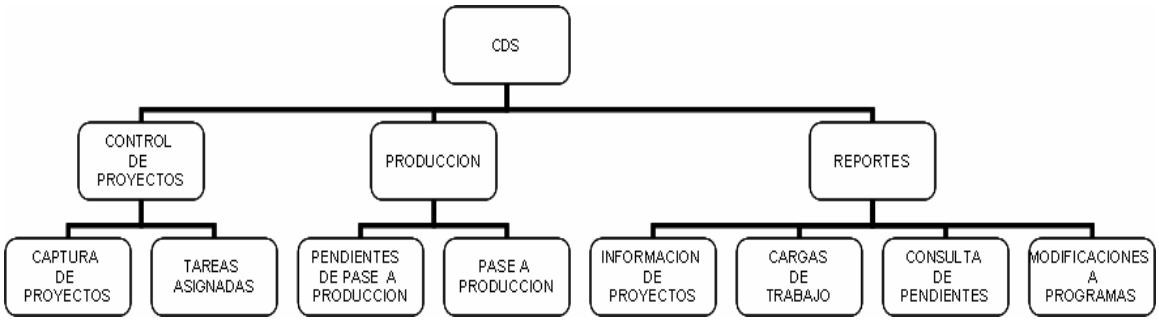

Figura 2.8. Módulos con los que contará el sistema CDS.

Organización del CDS

Las opciones que presentará el sistema así como la funcionalidad en las pantallas, dependerá de los roles que tenga asignado el usuario.

Dentro de la recopilación de requerimientos para el desarrollo del sistema, se habían identificado los siguientes puestos:

- Gerente de Sistemas de Franquicias
- Coordinador de Proyectos
- Coordinador de Desarrollo
- Analistas
- Desarrolladores

Al realizar el análisis de los usuarios que se involucran en el proceso; se creo un nuevo puesto en el departamento llamado: Administrador de Objetos o Control de Fuentes.

Puesto: Administrador de Objetos/Control de Fuentes
Depende de: Gerente de Sistemas de Franquicias
Supervisa a: No aplica
Número de puestos con estas características: 2
Descripción de Actividades: Encargado de administrar los diferentes objetos que componen el ERP; así como también de implementar los cambios y creación de objetos en los ambientes de producción.

Finalmente el organigrama del departamento quedo de la siguiente manera (Ver Fig. 2.9.):

Figura 2.9. Organigrama del Departamento de Sistemas después del análisis.

En base al análisis del nuevo organigrama se determinaron los roles que empleará el sistema. Estos son:

Control Internos
Comité de Asignación de Proyectos
Control de Fuentes
Control Desarrollo
Analista
Desarrollador

Los roles y responsabilidades dentro del sistema quedaron de la siguiente forma (Ver tabla. 2.2.):

Puesto	**Roles Asignados**	**Responsabilidad**
Gerente de Sistemas	Comité de Asignación de Proyectos, Control Internos	Asignación de proyectos, seguimiento de status de requerimientos y tareas asignadas a analistas y desarrolladores respectivamente. Autorización de pases a producción internos (requerimientos derivados por el propio sistema, ya sea por alguna mejora a un proceso o un error detectado por los mismos analistas)
Analistas	Comité de Asignación de Proyectos y/o Analista	Encargados de asignar requerimientos junto con el Gerente de Sistemas y el Coordinador de Proyectos. Los analistas dan de alta requerimientos en el sistema y trabajan con los que se le asignen. Asignan tareas a los desarrolladores que se le preasignen, validan las tareas y crean los pases a producción una vez que se terminen los proyectos.
Coordinador de Proyectos	Analista, Comité de Asignación de Proyectos	Encargado de coordinar y asignar pendientes junto con el Gerente de Sistemas.

Coordinador de Desarrollo	Control Desarrollo, Analista, Desarrollador	Encargado de coordinar las tareas que se asignan a los desarrolladores, supervisar que se cumplan en tiempo y conforme a los estándares.
Desarrolladores	Desarrollador	Encargados de crear o modificar los distintos objetos que se usarán en el *ERP*.
Administrador de Objetos	Control de Fuentes	Su tarea es aplicar los cambios o creación de objetos en los sistemas de Producción.

Tabla 2.2. Roles y Responsabilidades para el manejo del CDS.

Ya que se tienen identificados los tipos de usuarios que interactuaran con el sistema y las opciones con que contará, ahora se podrán realizar los diagramas de procesos de una forma clara pues se tiene pleno conocimiento de las actividades que realiza cada uno de ellos.

Cabe resaltar que la empresa es auditada por personal externo, y aunque no existe un rol para este tipo de usuarios en el CDS, es necesario contar con opciones dentro del sistema que permitan proporcionar la información necesaria para esta supervisión. Las auditorias se basan en verificar que los procesos surgidos a partir de requerimientos de usuarios o del propio departamento de sistemas se sujeten a ciertos lineamientos o directivas bien definidas. (Estos lineamientos estarán marcados por el sistema y los procesos se describirán posteriormente).

Objetos que se registrarán en el sistema

Mucho se ha escrito acerca del objetivo del sistema (administración de las modificaciones realizadas al *ERP*). Por lo tanto es necesario identificar y definir aquellos posibles objetos de programación (o entes lógicos) con los que se trabajan dentro del *ERP* y por consecuencia se registrarán dentro del CDS.

Tipos de Objetos:

Base de Datos: Procedimiento almacenado [A6] (*procedure, function, package, package body, trigger, view, java class, java source, object type*).
Aplicación: Módulo forma (*form*) [A4], módulo reporte (*report*) [A5].
Personalizado: *Script* [A7]
Librería (*Library*) [A8].

2.5. Diagramas de Procesos

En la sección anterior, se explicó la funcionalidad del *ERP* y como el sistema a desarrollar deberá integrarse. También se fijaron las bases con respecto a los distintos tipos de usuario que interactuaran con el sistema y se propuso su estructura general en base a los requerimientos recabados con el personal entrevistado.

Para que el desarrollo del sistema en términos de codificación se lleve a cabo de forma exitosa, es necesario contar con una correcta identificación tanto de los procesos como de los flujos de información permitidos. En este apartado se describen los procesos que se presentaran dentro del sistema a desarrollar.

El siguiente diagrama (Ver Fig. 2.10.) muestra de manera general la vida de un requerimiento de software. Desde su creación hasta su implantación en el ambiente de producción.

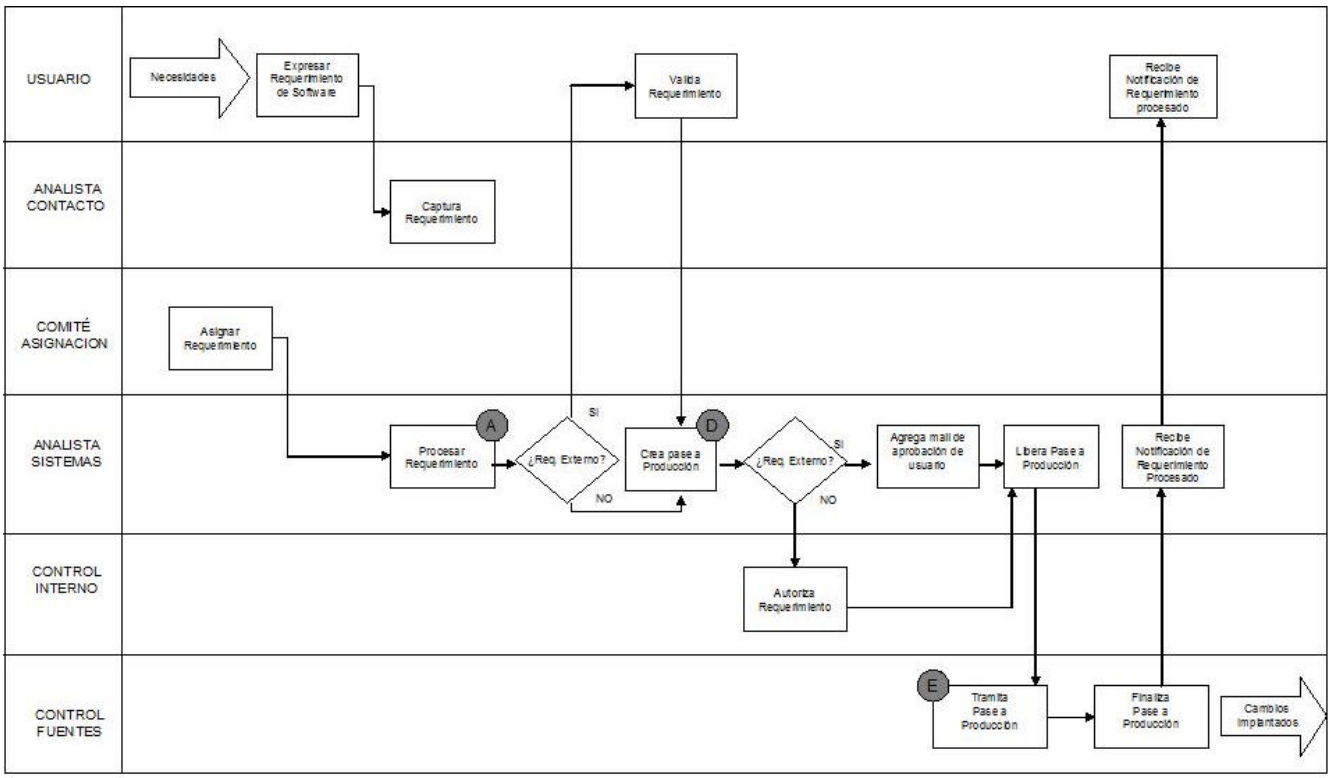

Figura 2.10. Diagrama de procesos que describe la vida de un requerimiento.

En los diagramas A. “Procesar requerimiento” y B “Desarrollar tareas” se describe el proceso de captura, asignación, desarrollo y validación de las tareas de programación asociadas a un requerimiento. Ver Fig. 2.11 y 2.12.

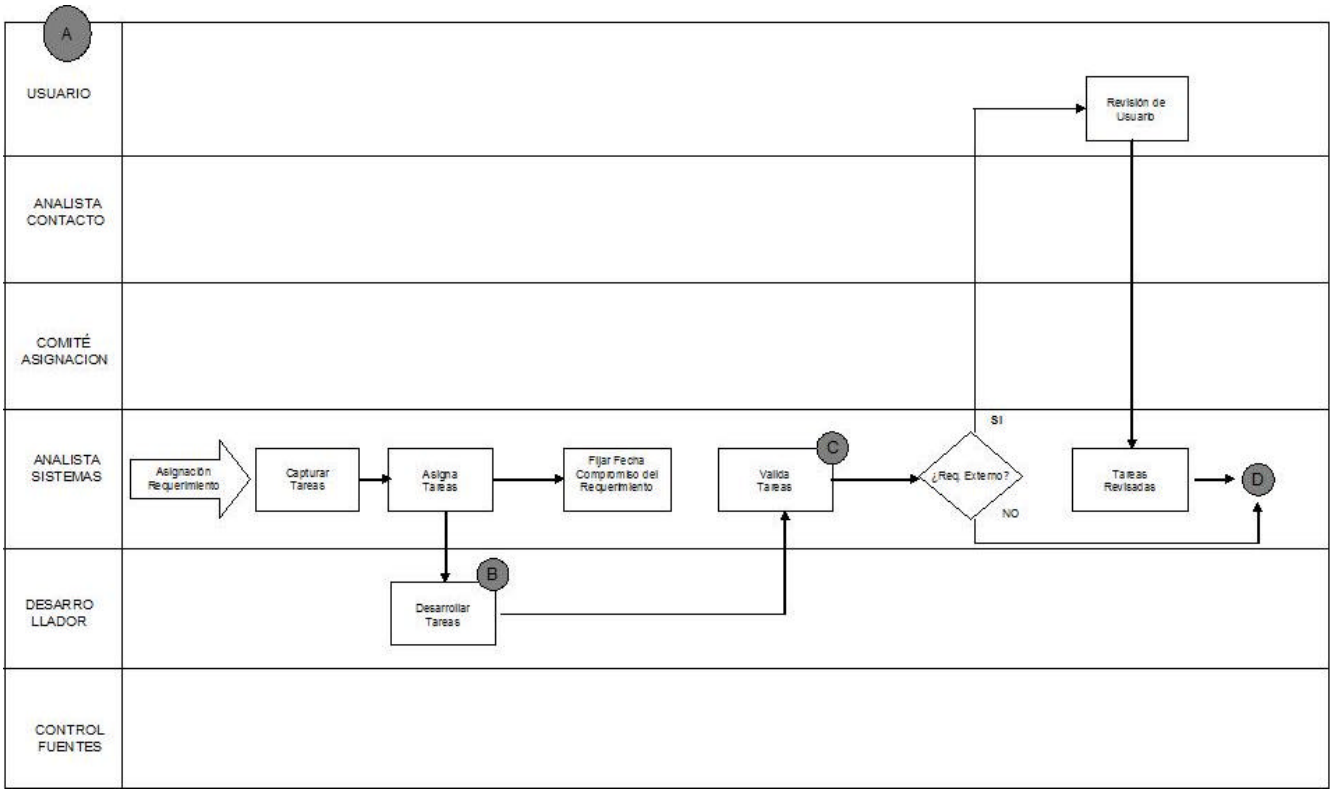

Figura 2.11. Diagrama A.

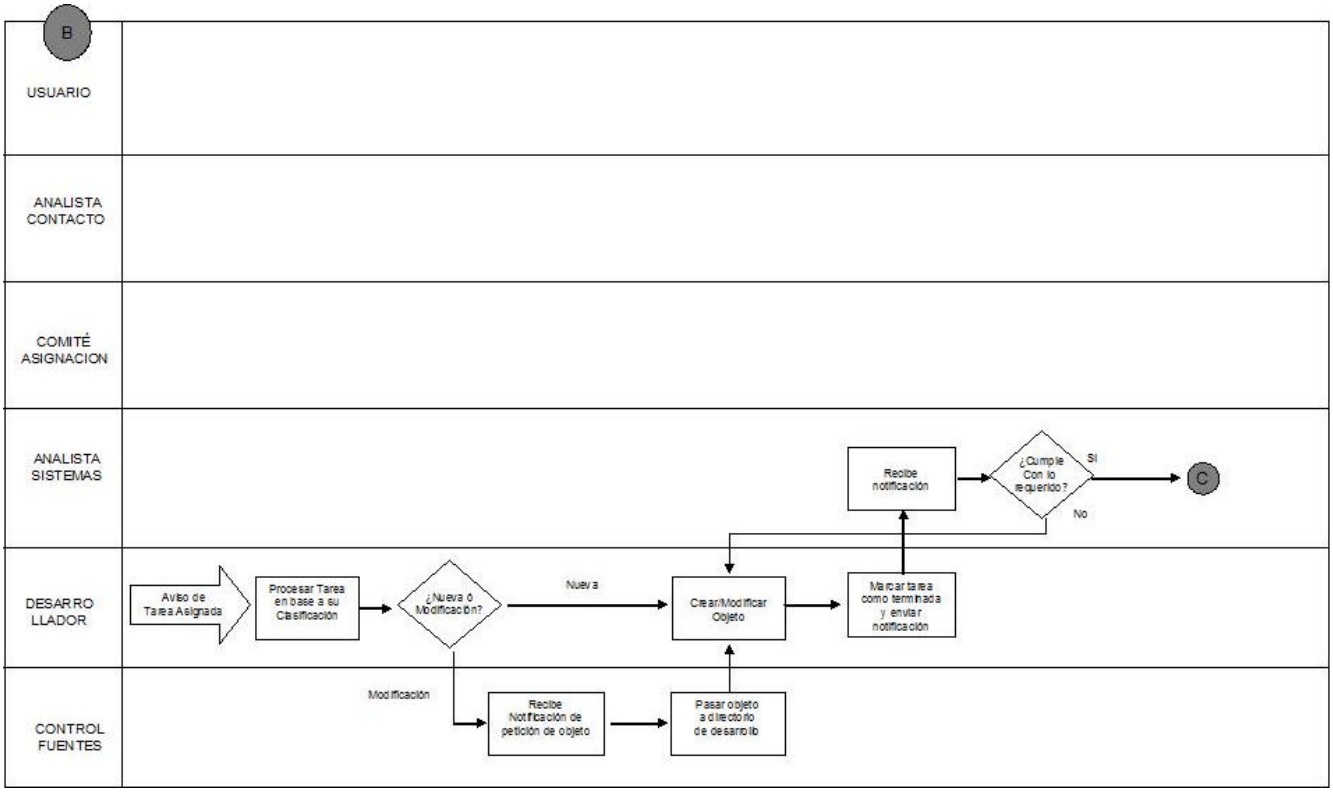

Figura 2.12. Diagrama B.

Por ultimo el Diagrama E “Tramita pase a producción”, detalla la forma en que el usuario “Control de Fuentes” realiza la implementación de un pase a producción generado por un requerimiento de software. Ver Fig. 2.13.

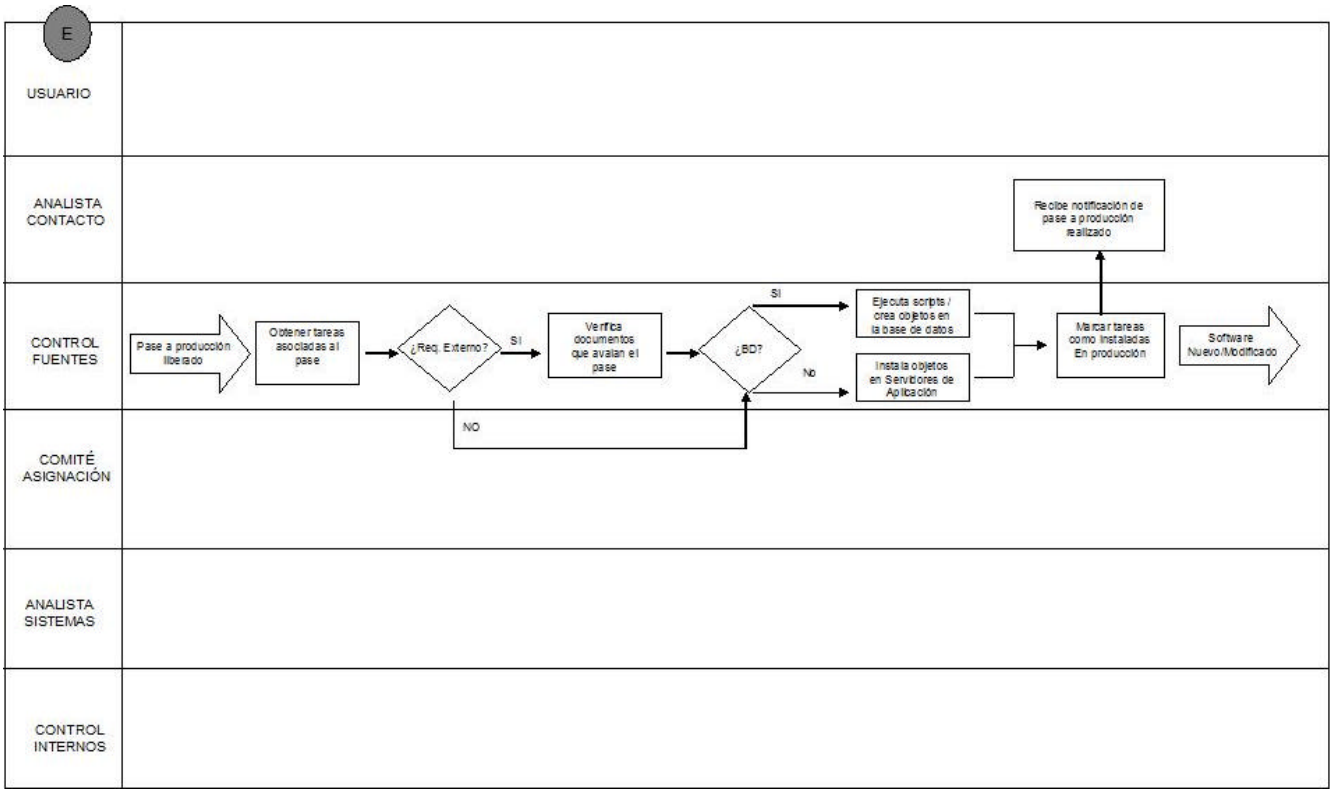

Figura 2.13. Diagrama E.

2.6. Diagramas de Flujos de Datos (DFD's)

Para mostrar los flujos de entrada y salida de información nos apoyamos en los DFD's. Como se explico en el capítulo referente a los conceptos básicos, estos diagramas nos sirven de apoyo para representar las interrelaciones que se dan entre un proceso y otro en cuanto a términos de flujos de información.

En el DFD de nivel 0 se muestran solo las entidades externas que interactuarán con el sistema; como son el ingreso de información y la espera de resultados. Ver Fig. 2.14.

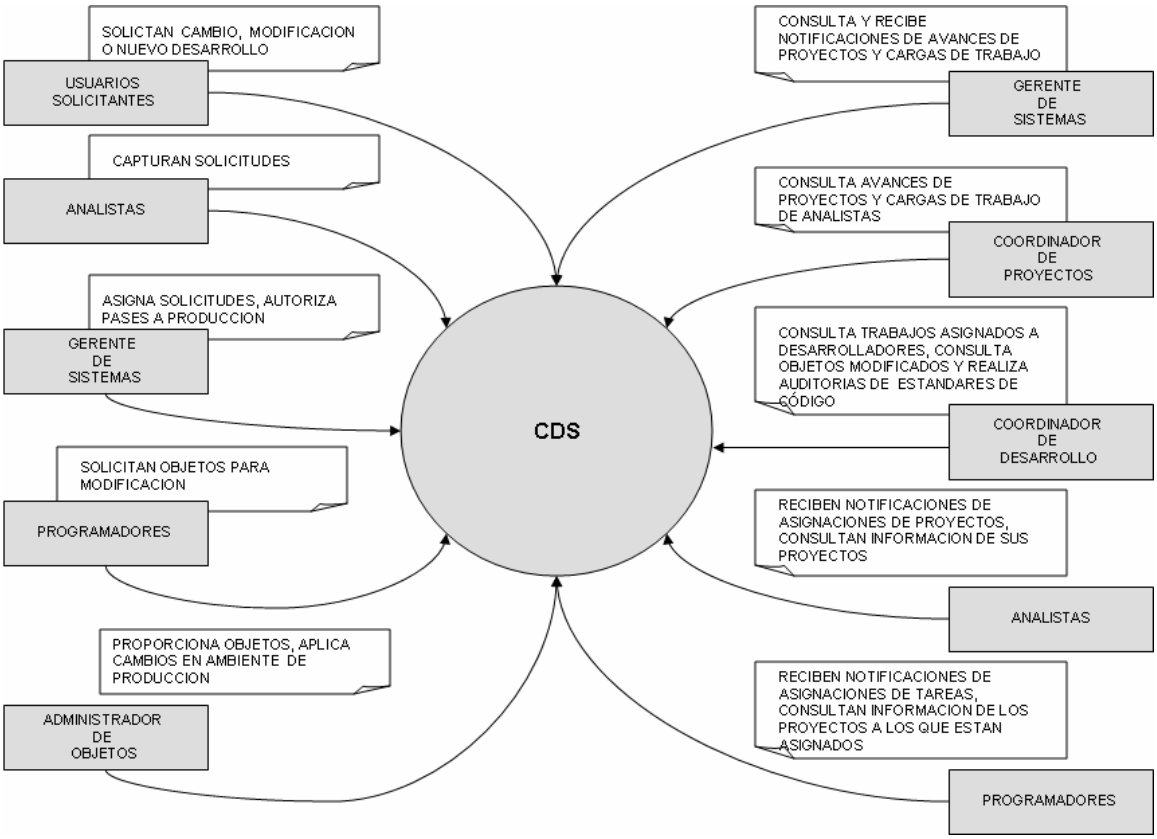

Figura 2.14. DFD de Nivel 0.

En la figura anterior, las entidades externas son todas aquellas personas involucradas en el proceso de desarrollo de requerimientos de software; como son: usuarios solicitantes, desarrolladores, analistas, coordinadores, etc. Las entradas al sistema son: la solicitud de un usuario para que se realice cierto cambio al *ERP*, la solicitud de objetos para su correspondiente modificación.

El sistema CDS no interactúa con dispositivos o sistemas externos. Se puede considerar como salida la información derivada de los procesos que se generan dentro del sistema, esta información se presenta mediante pantallas de consulta, generación de reportes (formato *PDF, XLS*) y envío de correos electrónicos.

Los flujos de datos que se dan dentro del sistema se presentan solo entre el personal del departamento. Siendo la única excepción el contacto entre el analista que captura la solicitud y el usuario solicitante que puede ser de una franquicia de cualquier parte de la República Mexicana y Centro América.

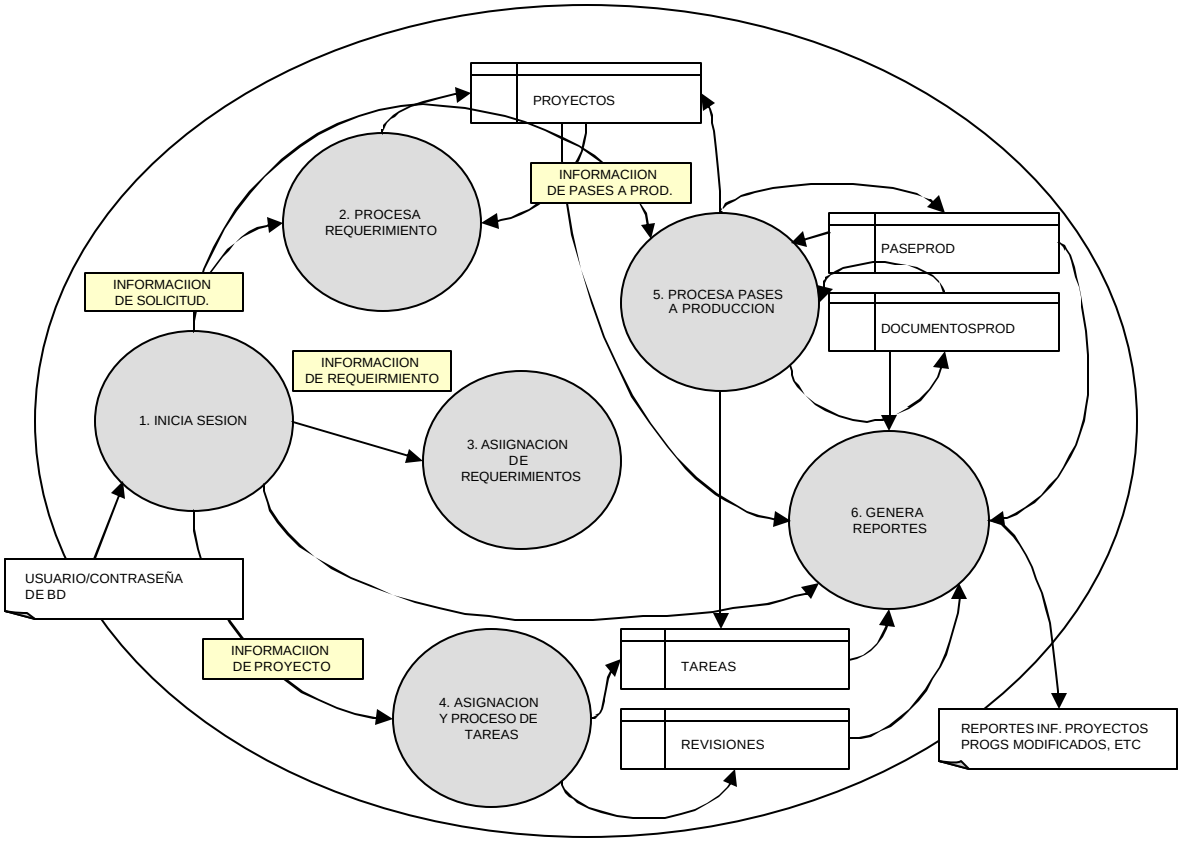

Figura 2.15. DFD de Nivel 0 con sus subprocesos.

En la figura 2.15, se muestran los 6 procesos principales que componen al sistema a desarrollar. En los diagramas posteriores se presentaran con mayor detalle cada uno de los procesos. Los procesos 4 y 5 (“Asignación y proceso de tareas”, “Procesa pases a producción” respectivamente) son procesos mas complejos, por lo cual contienen niveles de mayor profundidad.

Proceso 1. Inicia Sesión

El proceso de inicio de sesión trata acerca de la conexión al sistema por parte del usuario, para ello se necesita ingresar una clave de usuario y contraseña adecuados. Se tiene definido un número de intentos permitidos para poder acceder al sistema; en dado caso de que se hayan alcanzado el máximo número de intentos, el usuario se bloqueará y será necesario pedir que se desbloquee la cuenta. (esto se hace mediante la definición de *profiles* a los usuarios. Los *profiles* son un mecanismo de asignación de recursos para los usuarios y son parte del gestor de base de datos *Oracle*) Por el contrario si el usuario y *password* se proporcionan de manera correcta, se ingresará al sistema y se presentaran los módulos a los cuales tiene acceso.

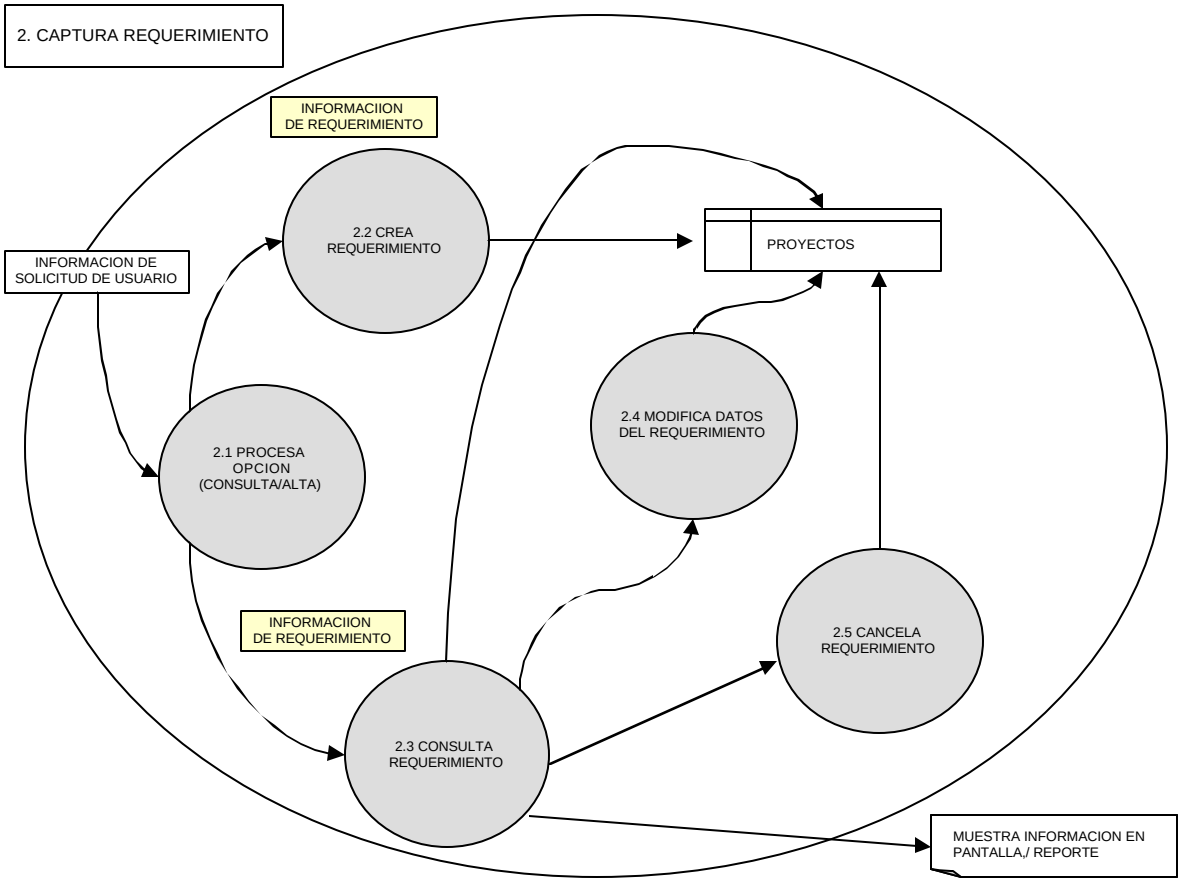

Figura 2.16. DFD 2, captura de requerimiento.

Proceso 2. Captura de requerimiento

El proceso de captura de requerimientos (Fig. 2.16) permite en primera instancia consultar o dar de alta en el almacén de datos “Proyectos” requerimientos provenientes de los usuarios. Una vez creado un requerimiento, este puede ser consultado, modificado o cancelado. La consulta de la información se puede realizar a través de pantallas o reportes.

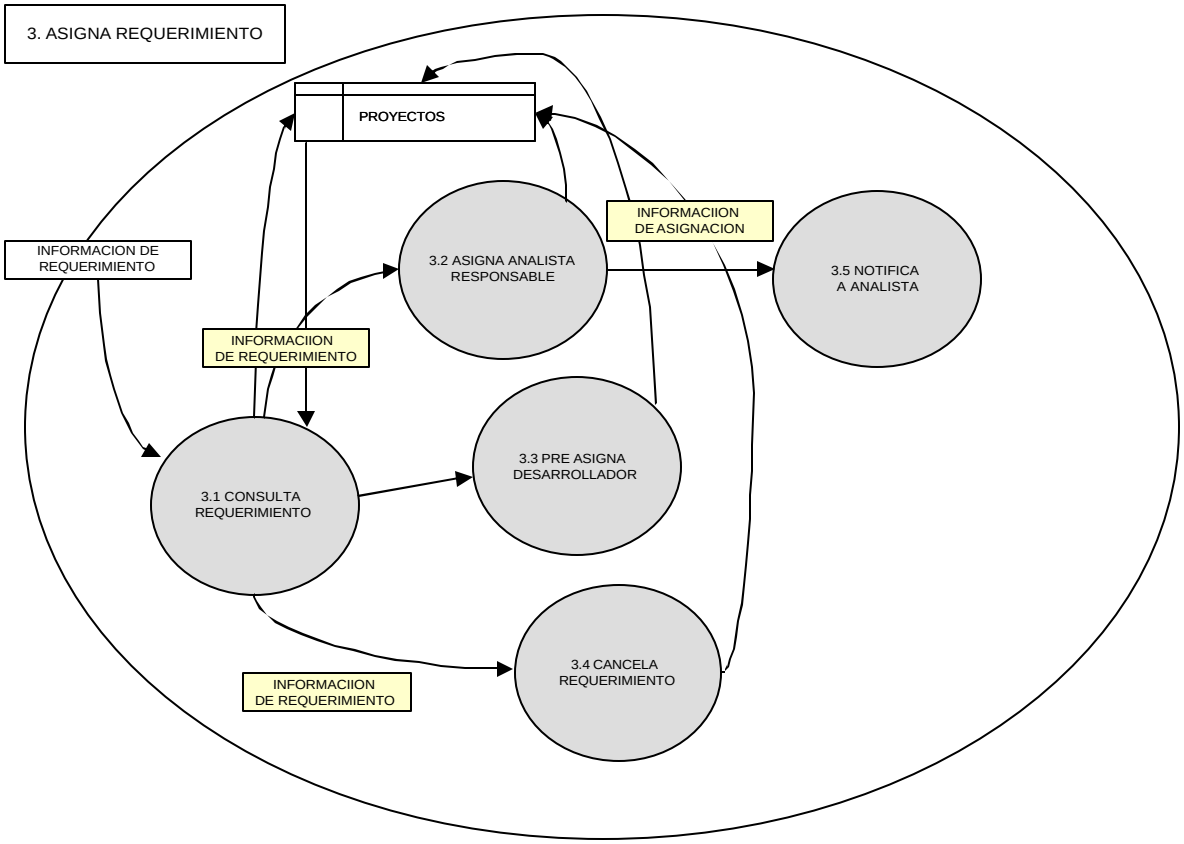

Figura 2.17. DFD 3, asigna requerimiento.

Proceso 3. Asigna requerimiento

Este proceso (Fig. 2.17), permite consultar requerimientos provenientes del almacén de datos "Proyectos" para su posterior asignación a cualquiera de los analistas que laboran en el departamento de Sistemas de Franquicias de Interceramic. Es necesario resaltar que se debe pre-asignar también a un desarrollador de aplicaciones, el cual se hará cargo de la programación necesaria de las tareas resultantes de dicho requerimiento. La información del requerimiento que se asignó al analista se envía a través de un correo electrónico para que éste último este enterado de su nueva asignación.

Proceso 4. Asigna y procesa tareas

Este proceso se divide en 2 subprocesos, el primero se refiere a la captura y asignación de tareas a desarrolladores y el segundo trata acerca del procesamiento y validación de las tareas por parte del analista o usuarios de la aplicación. Ver Fig. 2.18.

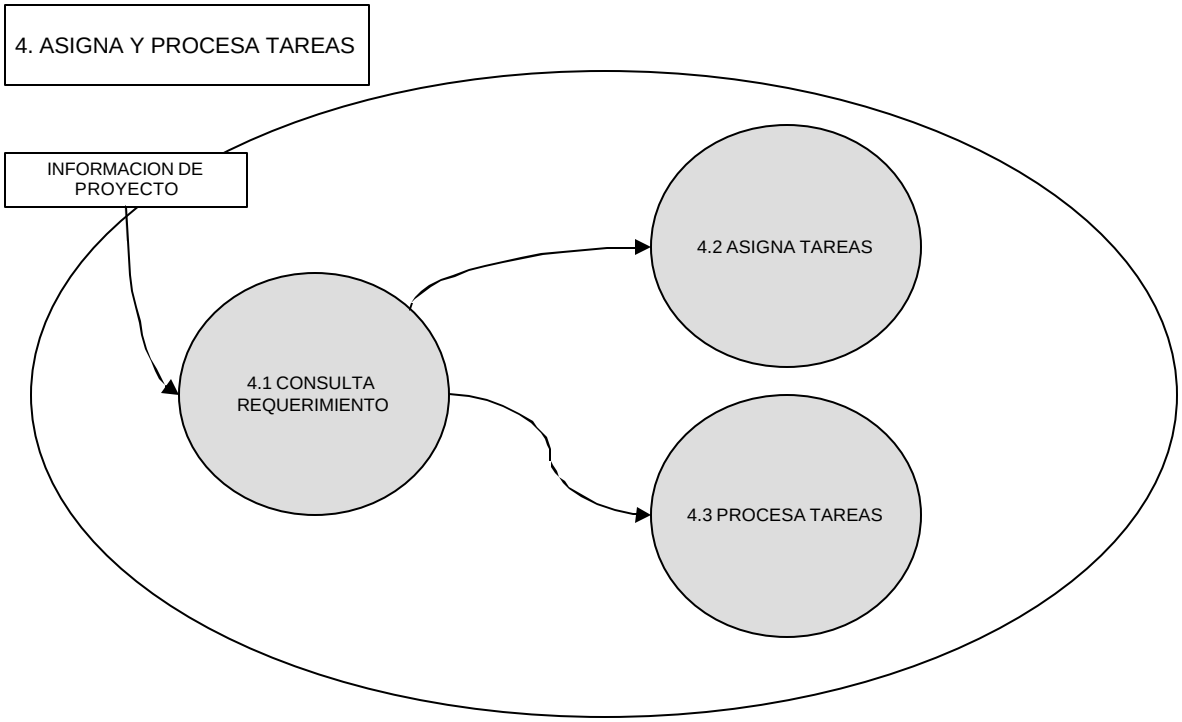

Figura 2.18. DFD 4, asigna y procesa tareas.

Proceso 4, Nivel 1 Asigna tareas

La asignación de tareas es muy parecido en concepto al proceso de asignación de requerimientos, con la particularidad de que en este caso no se asignan requerimientos si no objetos de tipo software que forman parte de la aplicación y que deben desarrollarse o modificarse según sea el caso. Dentro de este proceso se pueden crear tareas (de distintos tipos), asignar o cancelar. Al momento de asignar la(s) tarea(s) se envía un correo electrónico para avisar al desarrollador que debe comenzar con el proceso de la(s) tarea(s). Ver Fig. 2.19.

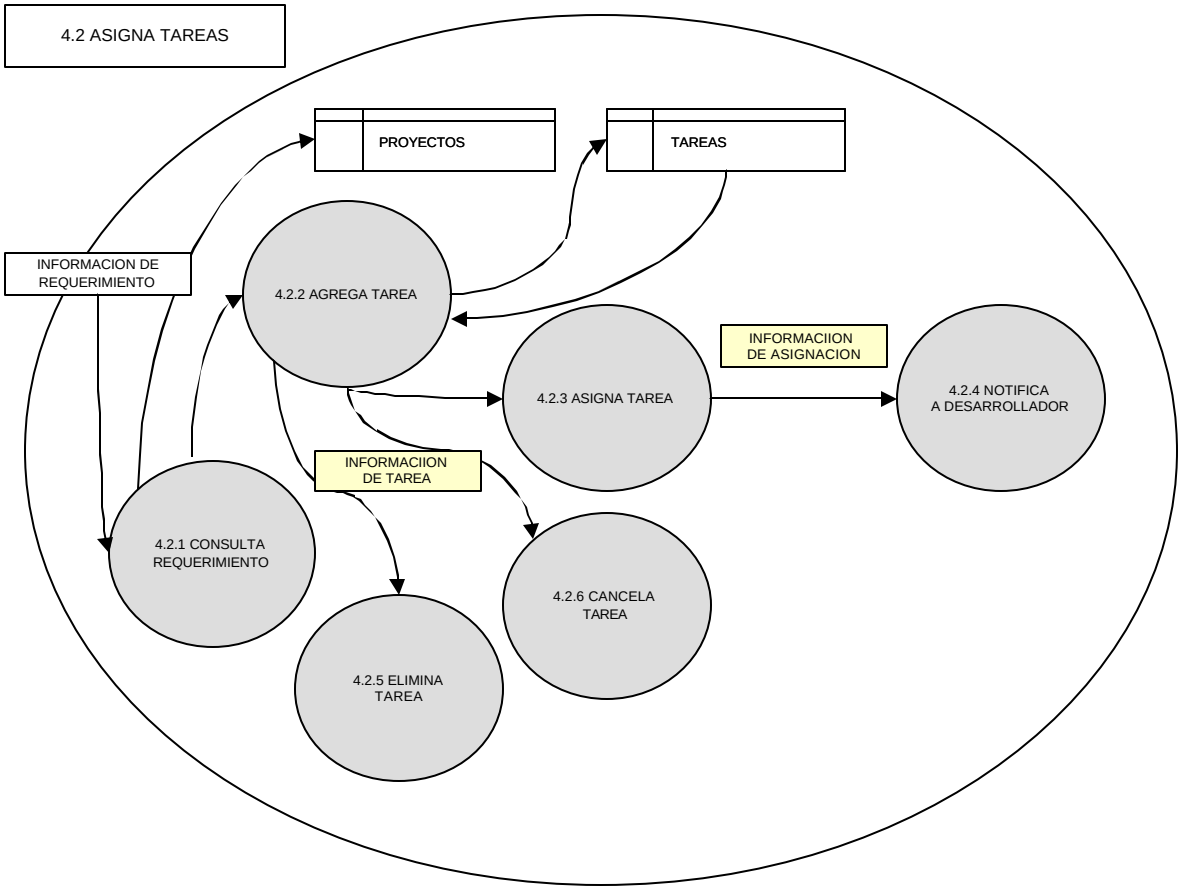

Figura 2.19. DFD 4, Nivel 1 Asigna tareas.

Proceso 4, Nivel 2 Procesa tareas

En este proceso se consultan las tareas asignadas, estas se pueden marcan en status de proceso o terminadas.

Es importante mencionar que al momento de marcar una tarea como “proceso”, en el caso de que la tarea sea una modificación, se envía un correo electrónico al usuario que tiene el rol de “Control de fuentes” para que ponga el objeto/código fuente apropiado dentro de un directorio conocido para que el desarrollador pueda comenzar a trabajar.

Una vez que las tareas se han finalizado, éstas se deben marcar como terminadas para que el sistema envié un correo electrónico al analista que asigno las tareas. Este correo tendrá la información referente a todas las tareas marcadas como terminadas, para que este proceda a realizar la validación correspondiente. En el caso de que una o más tareas tengan error o algo se haya omitido ya sea por parte del desarrollador o del analista, se debe marcar nuevamente la tarea como “proceso”; esto provoca que se repita el proceso antes mencionado hasta que se cumpla con lo requerido por parte del analista/usuario de la aplicación. Ver Fig. 2.20.

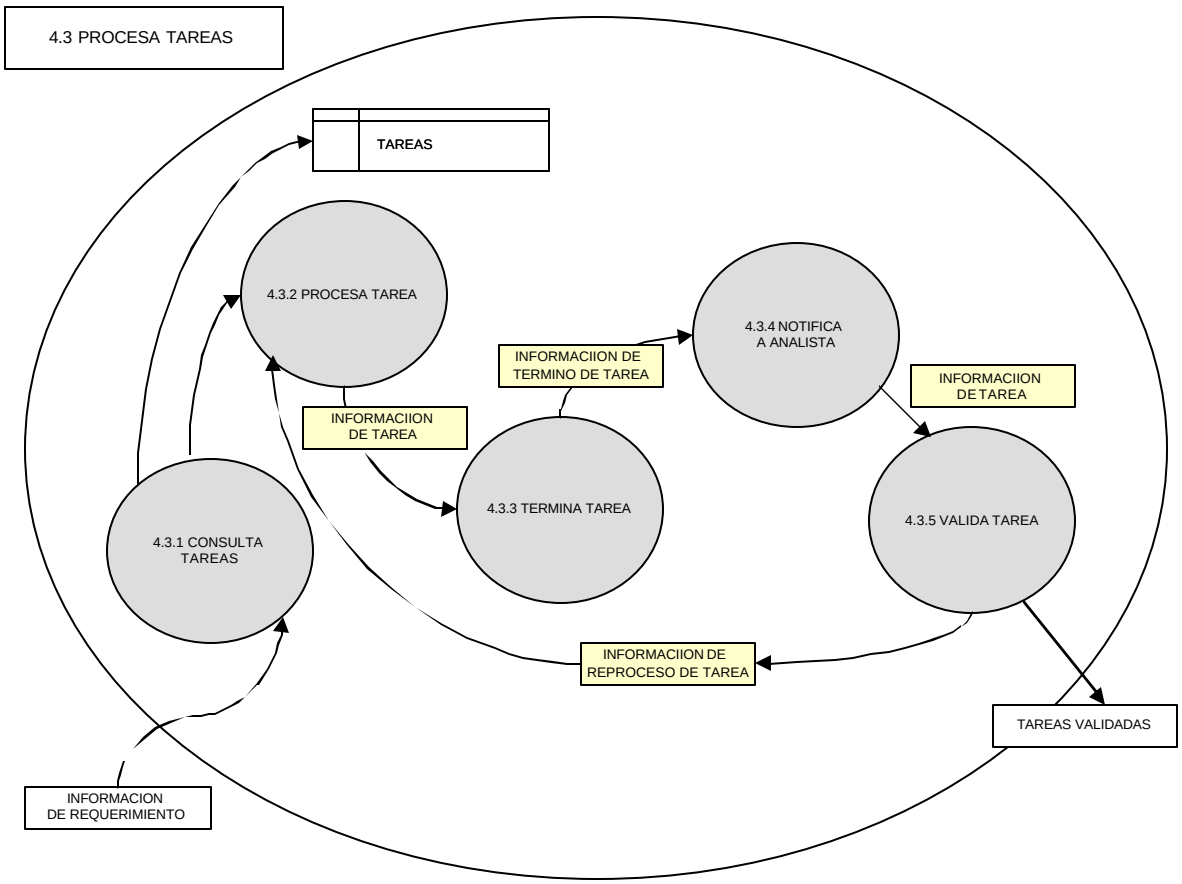

Figura 2.20. DFD 4, Nivel 2 Procesa tareas.

Proceso 4, Nivel 2 Subproceso: Validación de tareas

Dentro de la validación de tareas se tienen dos subprocesos, y el camino que se debe seguir depende de la naturaleza del requerimiento. “Externo” significa que fue una petición hecha por un usuario final. “Interno” quiere decir que el requerimiento surgió dentro del departamento de sistemas.

En el caso de que el requerimiento sea externo, se debe crear una solicitud de revisión que contenga todas las tareas modificadas o creadas para ese requerimiento. Esta solicitud se enviará por correo electrónico al usuario que solicitó dicho requerimiento y una vez que el usuario solicitante haya verificado que el funcionamiento de su petición es el adecuado, éste responderá vía correo electrónico aprobando todas las tareas que están asociadas a la revisión.

Si el requerimiento es interno, el analista es el responsable de validar que las tareas asignadas al desarrollador funcionen correctamente.

Una vez realizado lo descrito anteriormente (según sea el caso) se procederá a marcar las tareas como validadas. Ver Fig. 2.21.

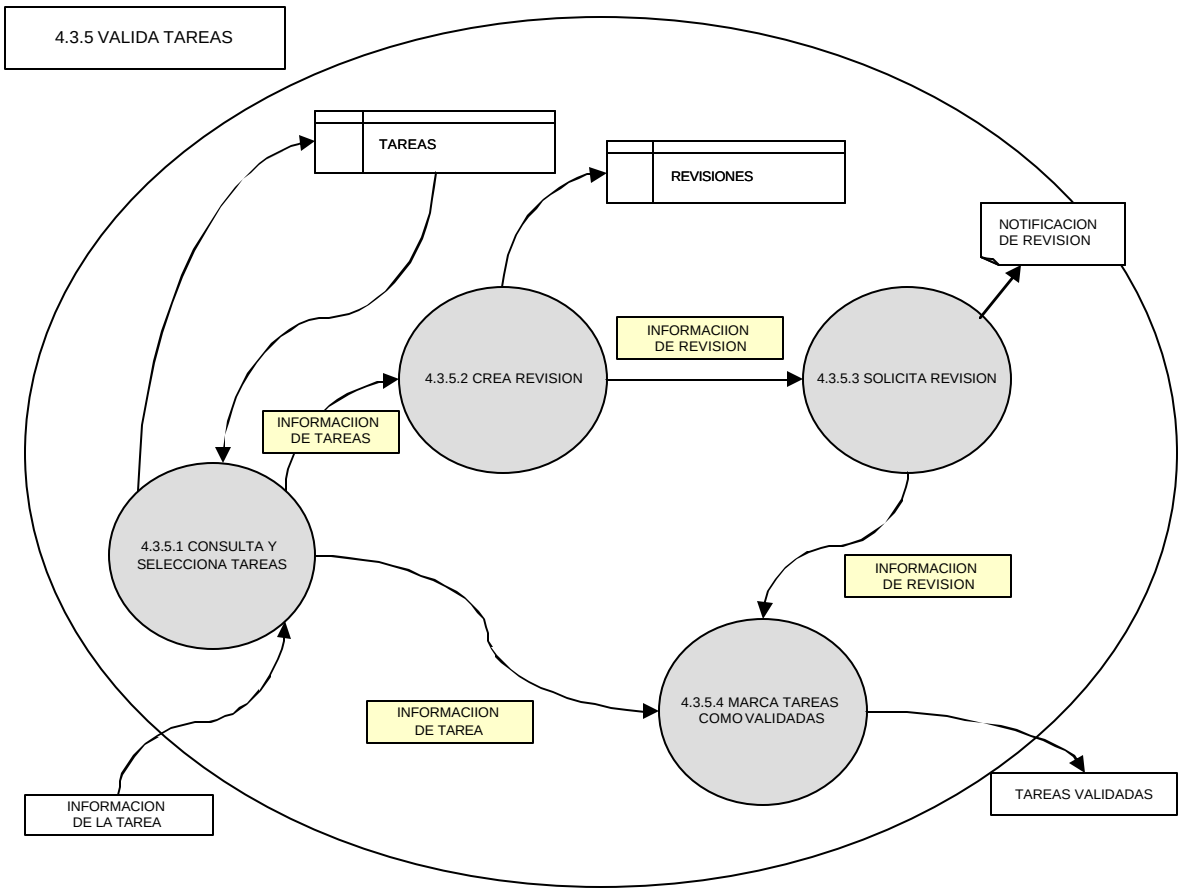

Figura 2.21. DFD 4, Nivel 1 Validación de tareas.

Proceso 5. Procesa pases a producción

Dentro del proceso de pases a producción, se pueden consultar, crear, cancelar o modificar pases a producción. El pase a producción contiene las tareas que se han validado en un requerimiento. Una vez que se han agregado las tareas pertinentes, se procede aprobar del pase. (Ver Fig. 2.22)

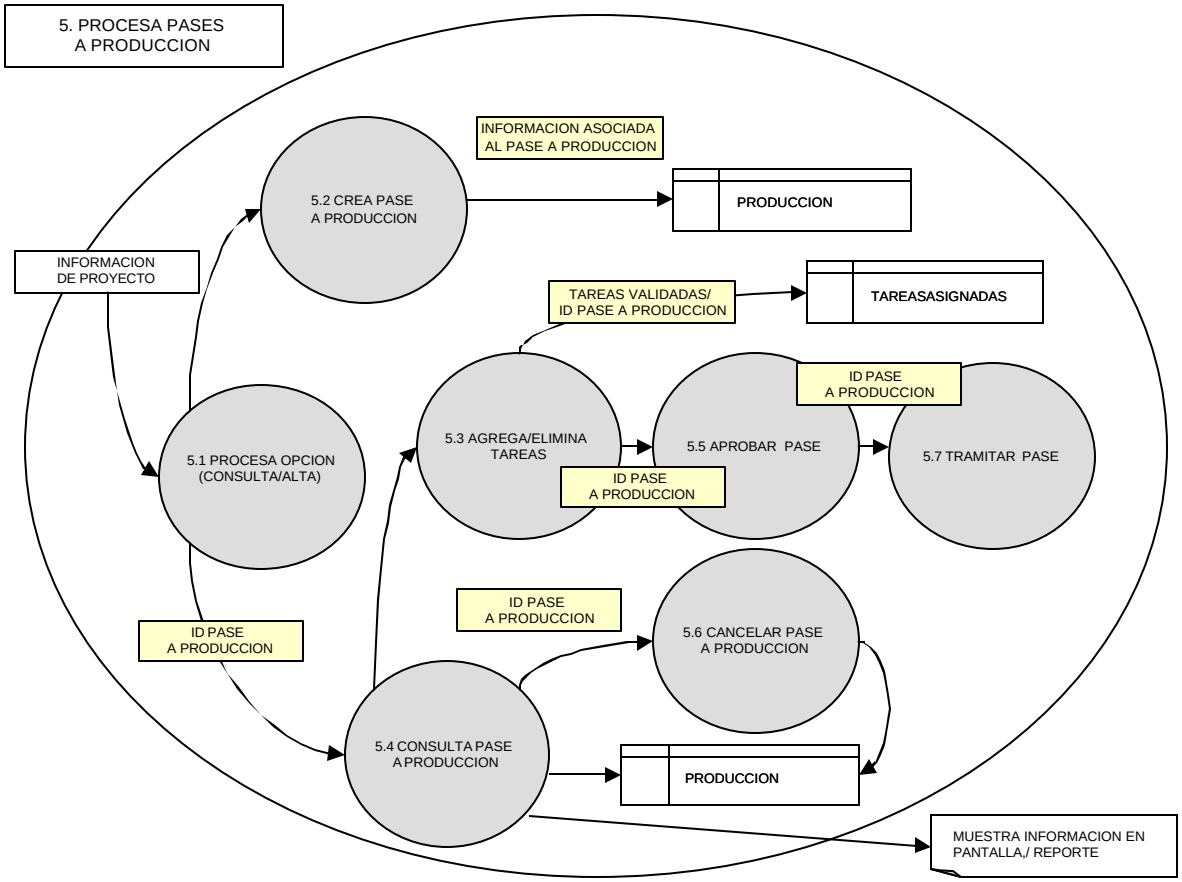

Figura 2.22. DFD 5, procesa pases a producción.

Proceso 5, Nivel 1 Aprobar pase

Si el pase a producción proviene de un requerimiento interno, se realiza una petición de aprobación del pase; en este caso se notifica mediante un correo electrónico al usuario con el rol “Control Internos” verifique que todo este en orden con este pase a producción y apruebe el mismo.

Si el pase a producción esta asociado a un requerimiento externo, el analista encargado del requerimiento, debe anexar la documentación necesaria para poder marcar el pase como aprobado (correos electrónicos de respuesta de revisión de los programas modificados/creados asociados al pase).

Ya con el pase aprobado el analista libera el pase a producción; con esto se notifica por medio de un correo electrónico al usuario con el rol “Control de fuentes” para que este realice el pase a producción correspondiente. Ver Fig. 2.23.

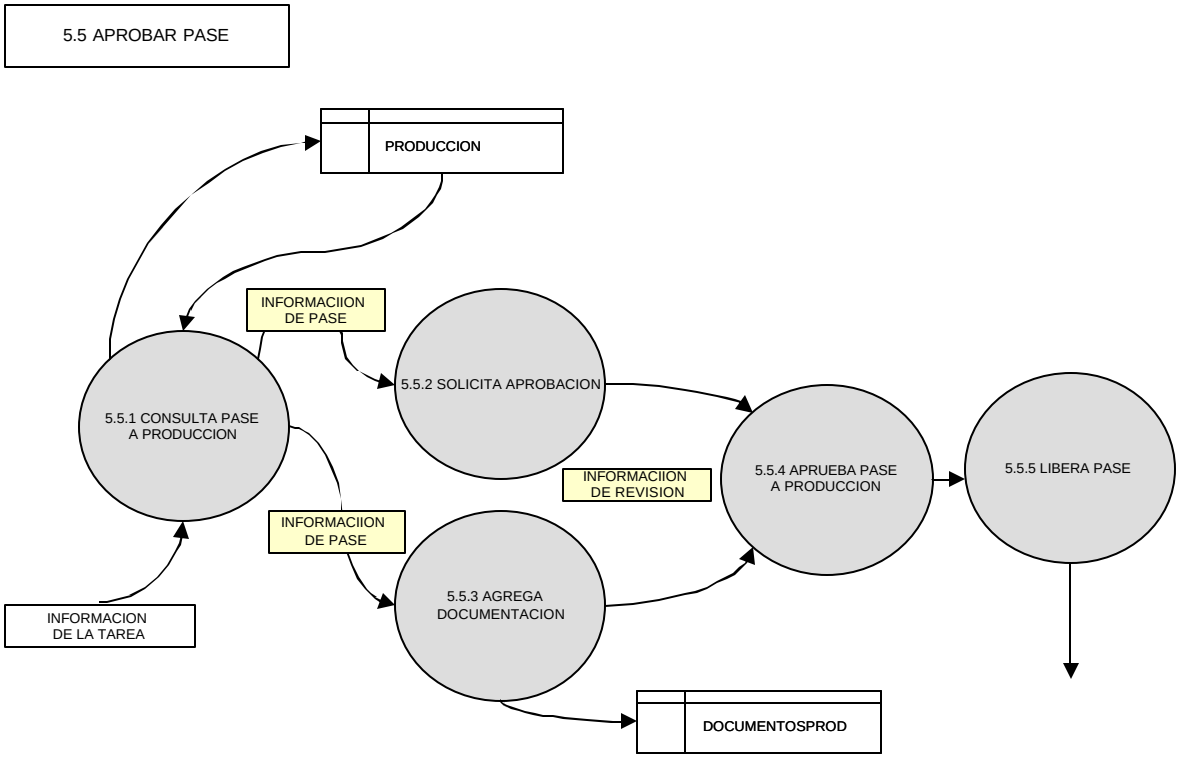

Figura 2.23. DFD 5, Nivel 1 Aprobar pase a producción.

Proceso 5, Nivel 2 Tramitar pase

Una vez que el usuario con el rol "Control de fuentes" ha recibido la notificación de que un pase a producción fue liberado, debe comenzar con el trámite del mismo. Para ello marca el pase a producción como "Tramite" y comienza a realizar las acciones necesarias dependiendo de la naturaleza del requerimiento. Si es externo, primero procede a verificar que la documentación asociada al pase avale que se lleven a cabo los cambios o creación de objetos en los ambientes de producción. Una vez realizada esta verificación o si el pase proviene de un requerimiento interno solo procede a implantar en el ambiente de producción los cambios o creación de objetos indicados por el pase a producción.

Terminando el tramite del pase, éste se marca con el status de "Producción" (en automático se envía una notificación al analista responsable del requerimiento indicándole que los cambios al sistema se llevaron a cabo de forma satisfactoria). Ver Fig. 2.24.

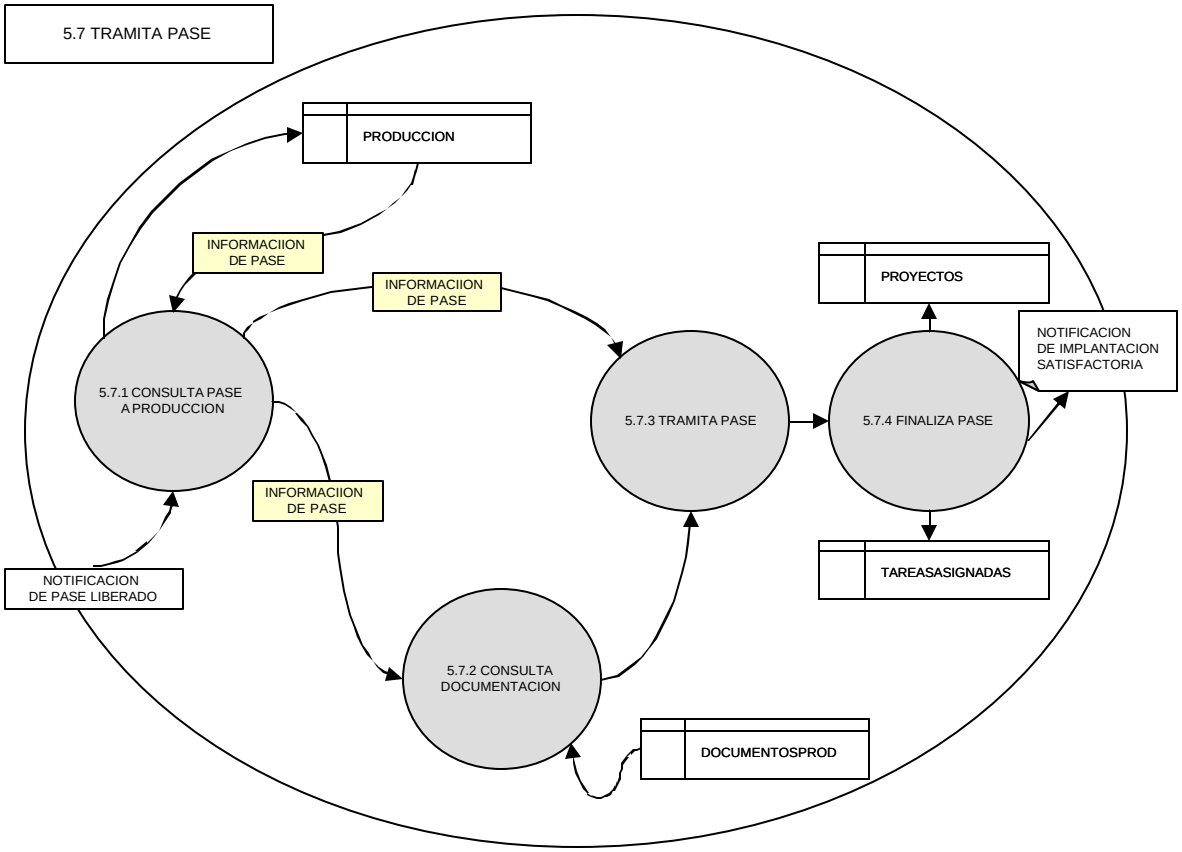

Figura 2.24. DFD 5, Nivel 1 Tramitar pase a producción.

Gracias a los diagramas de flujo de datos se ha podido mostrar de forma gráfica con qué almacenes de datos se trabajará en cada proceso. También han servido como punto de partida para describir de forma general cómo se emplea la información almacenada y cómo a través de cada proceso la información se va transformando para ir desencadenando nuevos procesos.

2.7. Modelado Conceptual (Modelo Entidad Relación)

El modelo conceptual es un conjunto de conceptos que permiten describir la realidad mediante representaciones lingüísticas y gráficas. El modelo conceptual más empleado es el modelo entidad relación.

A continuación se presenta el esquema conceptual del CDS, el cual será independiente de todas las consideraciones físicas. Ver Fig. 2.25.

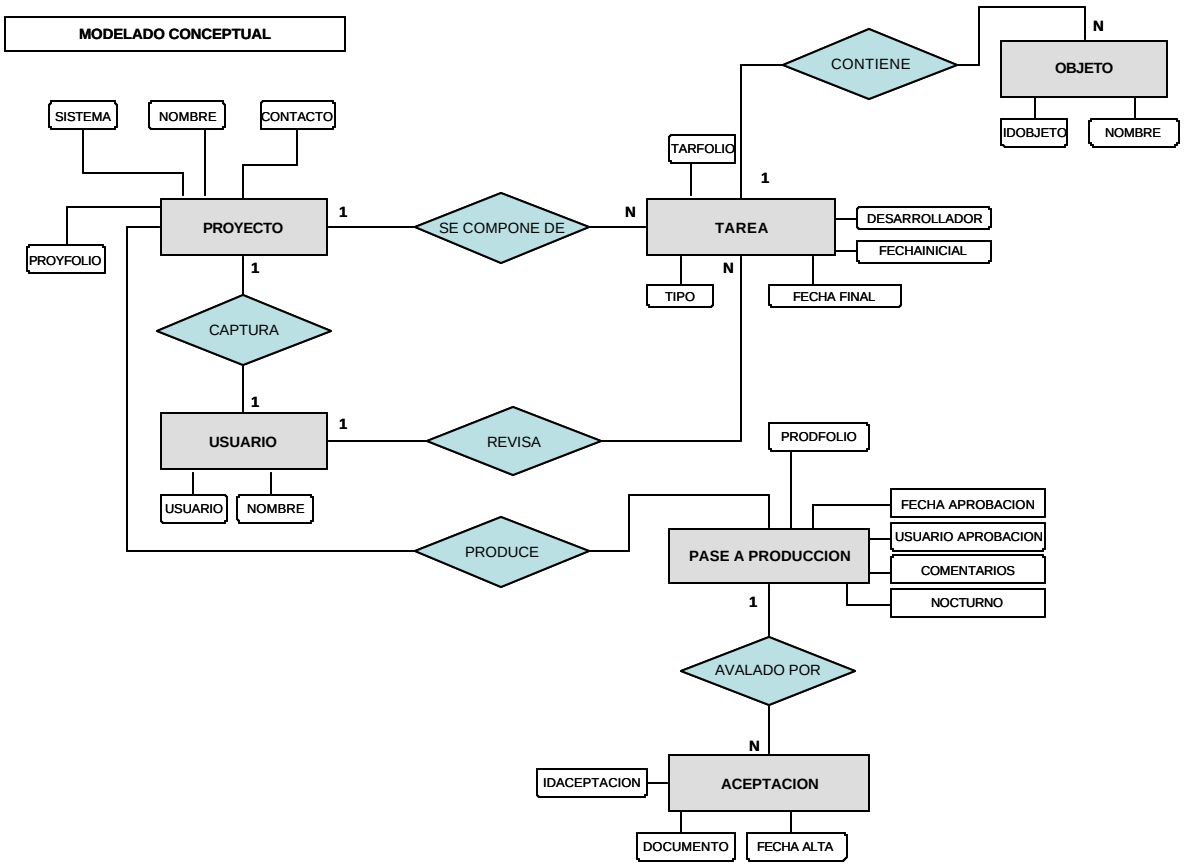

Figura 2.25. Diagrama Entidad Relación del CDS

La figura anterior muestra las entidades que se han identificado a lo largo del análisis del sistema; las cuales serán el punto de partida para definir el modelo relacional de base de datos.

2.8. Diseño de la Base de Datos

El proceso de normalización consiste en la descomposición sin pérdida de las tablas que están en una forma normal inferior, obteniéndose una forma normal superior. El objetivo de ésta tesis no consiste en mostrar paso a paso las técnicas empleadas para normalizar tablas, por lo cual solo se muestran las diferentes tablas ya normalizadas que deberán crearse en el esquema destinado para el CDS.

DSPARAMETROS

Almacena los parámetros generales de la Aplicación. Ver Tabla 2.3.

ATRIBUTO	TIPO	LLAVE	NOMBRE LLAVE	NULO	COMENTARIOS
ROLADMIN	VARCHAR2-30			SI	Rol asignado a usuarios que administran de forma total el sistema de control de proyectos
ROLCOMITE	VARCHAR2-30			SI	Rol asignado a usuarios del comite
ROLCONTROL	VARCHAR2-30			SI	Rol asignado a usuarios que pueden autorizar pase a produccion de proyectos internos
ROLCONTROLFUENTES	VARCHAR2-30			SI	Rol asignado a usuarios encargados del pase a produccion
ROLCONTROLDESARROLLO	VARCHAR2-30			SI	Rol asignado al coordinador de desarrollo
ROLDS	VARCHAR2-30			SI	Rol asignado a usuarios que desarrollan proyectos
ROLAN	VARCHAR2-30			SI	Rol asignado a usuarios que analizan proyectos
RANGOVENC1_1	NUMBER-3			NO	Lim. Inferior 1 para mails de seguimiento a proyectos (vencidos)
RANGOVENC1_2	NUMBER-3			NO	Lim. Superior 1 para mails de seguimiento a proyectos (vencidos)
RANGOVENC2_1	NUMBER-3			NO	Lim. Inferior 2 para mails de seguimiento a proyectos (vencidos)
RANGOVENC2_2	NUMBER-3			NO	Lim. Superior 2 para mails de seguimiento a proyectos (vencidos)
RANGOVENC3_1	NUMBER-3			NO	Lim. Inferior 3 para mails de seguimiento a proyectos (vencidos)
RANGOVENC3_2	NUMBER-3			NO	Lim. Superior 3 para mails de seguimiento a proyectos (vencidos)
RANGOXVENC1_1	NUMBER-3			NO	Lim. Inferior 1 para mails de seguimiento a proyectos (x vencer)
RANGOXVENC1_2	NUMBER-3			NO	Lim. Superior 1 para mails de seguimiento a proyectos (x vencer)
RANGOXVENC2_1	NUMBER-3			NO	Lim. Inferior 2 para mails de seguimiento a proyectos (x vencer)
RANGOXVENC2_2	NUMBER-3			NO	Lim. Superior 2 para mails de seguimiento a proyectos (x vencer)
RANGOXVENC3_1	NUMBER-3			NO	Lim. Inferior 3 para mails de seguimiento a proyectos (x vencer)
RANGOXVENC3_2	NUMBER-3			SI	Lim. Superior 3 para mails de seguimiento a proyectos (x vencer)

Tabla 2.3. Descripción de campos de la tabla DSPARAMETROS

DSPROYECTOS

Esta tabla permite concentrar los datos de los requerimientos realizados ya sea por usuarios finales o del propio departamento de sistemas. En caso de que proceda dicho requerimiento, se almacena la información referente al líder del proyecto, quién es el desarrollador preasignado para realizar las tareas de programación, las fechas de inicio y fin del proyecto y los diferentes status a los que se ve sometido un proyecto. Ver Tabla 2.4.

ATRIBUTO	TIPO	LLAVE	NOMBRE LLAVE	NULO	COMENTARIOS
PROYFOLIO	NUMBER-10	Llave Primaria	DSPR_PK	NO	Folio del Proyecto/Pendiente
NOMBRE	VARCHAR2-300			NO	Nombre del Proyecto/Pendiente
PROYECTO	NUMBER-1			NO	Indicativo del tipo de Requerimiento (0 - Proyecto, 1 - Pendiente 2 - BD)
TIPO	VARCHAR2-2			NO	Tipo de Proyecto/Pendiente (NUEVO/MEJORA/ERROR)
INTERNO	NUMBER-1			NO	Indica si el Proyecto/Pendiente surgió como requerimiento interno
PRIORIDAD	NUMBER-1			NO	Prioridad del Proyecto/Pendiente (ALTA/MEDIA/BAJA)
STATUSPR	VARCHAR2-2			NO	Status del proyecto (PEND/ASIG/ANAL/DESA/VALID/TERM/CANC/SUSP)
USUCLAVEANCONTACTO	VARCHAR2-30	Llave Foránea	DSPR1_USU_FK	SI	Clave del usuario que sirve como contacto con la Franquicia
EMPNUMSOL	NUMBER-3	Llave Foránea	DSPR_EMP_FK	SI	Franquicia Solicitante
USUCLAVESOL	VARCHAR2-30	Llave Foránea	DSPR2_USU_FK	SI	Clave del usuario que funge como solicitante del Proyecto/Pendiente
USUCLAVELIDER	VARCHAR2-30	Llave Foránea	DSPR3_USU_FK	SI	Clave del usuario que funge como líder del Proyecto/Pendiente
USUCLAVEDSPRE	VARCHAR2-30	Llave Foránea	DSPR4_USU_FK	SI	Desarrollador preasignado para trabajar en el proyecto
USUCLAVECANCEL	VARCHAR2-30	Llave Foránea	DSPR5_USU_FK	SI	Usuario que canceló el requerimiento
JUSTIFICACION	VARCHAR2-4000			SI	Justificación del Proyecto/Pendiente
FECHACOMPROMISO				SI	Fecha en la que se compromete a liberar el Proyecto/Pendiente
FECHAINICIAL				SI	Fecha en la que se comenzó el Proyecto/Pendiente
FECHAFINAL				SI	Fecha en la que se terminó el Proyecto/Pendiente
COMENTARIOS	VARCHAR2-4000			SI	Comentarios adicionales
PROCESO	VARCHAR2-4000			SI	Descripción breve del proceso
FECHAALTA				NO	Fecha de alta del requerimiento
FECHAASIGNACION				SI	Fecha en que se asignó el Proyecto/Pendiente al analista
ORDENDSPRE	NUMBER-10			SI	Priodidad del proyecto asignado al desarrollador
CAMBIOINTEXT	NUMBER-1			NO	Indica si cambio el proyecto de externo a interno

Tabla. 2.4. Descripción de campos de la tabla DSPROYECTOS

DSJUSTPROYECTOS

Almacena las justificaciones de aquellos requerimientos que se les ha cambiado la fecha compromiso. Ver Tabla 2.5.

ATRIBUTO	TIPO	LLAVE	NOMBRE LLAVE	NULO	COMENTARIOS
PROYFOLIO	NUMBER-10	Llave Primaria Llave Foránea	DSJPR_PK, DSJPR_DSPR_FK	NO	Folio del Proyecto/Pendiente
JUSTNUM	NUMBER-10	Llave Primaria	DSJPR_PK	NO	Justificacion del porque la fecha compromiso de la tarea no se alcanzo
USUCLAVE	VARCHAR2-30			NO	Analista que realizo el cambio de fecha de compromiso
JUSTIFICACION	VARCHAR2-4000			NO	Justificacion/Comentarios
FECHAANTERIOR				NO	Fecha compromiso anterior

Tabla 2.5. Descripción de campos de la tabla DSJUSTPROYECTOS

DSDETPROYECTOS

Esta tabla contiene tanto al analista líder del requerimiento como a los analistas que se incorporen al mismo. Ver Tabla 2.6.

ATRIBUTO	TIPO	LLAVE	NOMBRE LLAVE	NULO	COMENTARIOS
PROYFOLIO	NUMBER-10	Llave Primaria Llave Foránea	DSDPR_PK, DSDPR_DSPR_FK	NO	Folio del Proyecto/Pendiente
USUCLAVEAN	VARCHAR2-30	Llave Primaria Llave Foránea	DSDPR_PK, DSDPR1_USU_FK	NO	Analista responsable
COMENTARIOS	VARCHAR2-4000			SI	Comentarios Adicionales

Tabla 2.6. Descripción de campos de la tabla DSDETPROYECTOS

DSTAREASASIGNADAS

Almacena cada una de las tareas asignadas a los desarrolladores que participan en el requerimiento, ya sea para modificación o creación. La clasificación de las tareas, el tipo de objeto, los diferentes status por los que pasa una tarea, fecha de inicio, fin y validación se almacenan también en esta tabla. Ver Tabla 2.7.

ATRIBUTO	TIPO	LLAVE	NOMBRE LLAVE	NULO	COMENTARIOS
PROYFOLIO	NUMBER-10	Llave Primaria, Llave Foránea	DSTAAS_PK, DSTAAS_DSDPR_FK	NO	Folio del Proyecto/Pendiente
USUCLAVEAN	VARCHAR2-30	Llave Primaria, Llave Foránea	DSTAAS_PK, DSTAAS_DSDPR_FK	NO	Analista responsable
TARFOLIO	NUMBER-10	Llave Primaria	DSTAAS_PK	NO	Folio de la tarea
CLASIFTA	VARCHAR2-2			NO	Clasificacion de la tarea (NUEVA/MODIFICACION/SCRIPT)
STATUS	VARCHAR2-2			NO	Status (PEND/ASIGN/PROC/TERM/CANC/ SUSP/VAL/PORAPROB/APROB/PROD)
TIPO	VARCHAR2-2			NO	Tipo de objeto (BD/APLIC/EXT/LIB/NA)
TIPOOBJ	VARCHAR2-30			SI	Tipo de objeto de base de datos
ESQUEMA	VARCHAR2-30			SI	Esquema al que pertenece el objeto de bd
NOMBREOBJETO	VARCHAR2-30			SI	Esquema al que pertenece el objeto de bd
SISNUM	NUMBER-2			SI	Número de Sistema
SISNUMDESC	VARCHAR2-60			SI	Nombre del Sistema
MODCLAVE	VARCHAR2-30			SI	Clave del Módulo
MODCLAVEDESC	VARCHAR2-50			SI	Nombre del Módulo
SCRIPT	VARCHAR2-100			SI	Script
LIBRERIA	VARCHAR2-100			SI	Nombre de la libreria de código o de objetos
USUCLAVEDS	VARCHAR2-30	Llave Foránea	DSTAAS_USU1_FK	SI	Usuario asignado para desarrollar la tarea
FECHACOMPROMISO				SI	Fecha de compromiso
FECHAINICIAL				SI	Fecha Inicial
FECHAFINAL				SI	Fecha final
COMENTARIOS	VARCHAR2-4000			SI	Comentarios adicionales
USUCLAVEVALIDA	VARCHAR2-30	Llave Foránea	DSTAAS_USU2_FK	SI	usuario que valido la tarea
FECHAVALIDA				SI	Fecha de validación de la tarea
REVFOLIO	NUMBER-10	Llave Foránea	DSTAAS_DSREVM_FK	SI	Folio de revisión
ENREVISION	NUMBER-1			NO	Indica si la tareas se encuentra en revisión 1 - SI, 0 NO
PRODFOLIO	NUMBER-10	Llave Foránea	DSTAAS_DSPROD_FK	SI	Folio de producción
ORDEN	NUMBER-3			SI	Orden en el que se pasará a producción la tarea
COMENTARIOSDS	VARCHAR2-4000			SI	Comentarios del desarrollador
FECHAASIGNACION				SI	Fecha de asignación de la tarea
TIPOMODULO				SI	Tipo de Módulo (FORMA/REPORTE)
VISIBLEMENU	NUMBER-1			SI	Indica si el módulo es visible en el menú principal
SISNUMEXT	NUMBER-3			SI	Número de Sistema Externo
SISNUMEXTDESC	VARCHAR2-100			SI	Nombre del Sistema Externo
OBJETOEXT	VARCHAR2-100			SI	Clave del Objeto Externo
OBJETOEXTDESC	VARCHAR2-200			SI	Nombre del Objeto Externo

Tabla 2.7. Descripción de campos de la tabla DSTAREASASIGNADAS

DSJUSTTAREASASIGNADAS

Esta tabla sirve para almacenar que tareas se han retrasado con respecto a su fecha compromiso. Ver Tabla 2.8.

ATRIBUTO	TIPO	LLAVE	NOMBRE LLAVE	NULO	COMENTARIOS
PROYFOLIO	NUMBER-10	Llave Primaria, Llave Foránea	DSJTAAS_PK, DSJTAAS_DSTAAS_I	NO	Folio del Proyecto/Pendiente
USUCLAVEAN	VARCHAR2-30	Llave Primaria, Llave Foránea	DSJTAAS_PK, DSJTAAS_DSTAAS_I	NO	Analista responsable
TARFOLIO	NUMBER-10	Llave Primaria, Llave Foránea	DSJTAAS_PK, DSJTAAS_DSTAAS_I	NO	Folio de la tarea
JUSTNUM	NUMBER-10	Llave Primaria	DSJTAAS_PK	NO	Folio de la justificación
JUSTIFICACION	VARCHAR2-4000			SI	Justificacion/Comentarios

Tabla 2.8. Descripción de campos de la tabla DSJUSTTAREASASIGNADAS

DSREVISION

Esta tabla junto con la tabla **dsrevisionmails** sirve para almacenar las peticiones de revisión hacia los usuarios finales cuando un requerimiento es externo. Ver Tabla 2.9 y 2.10.

ATRIBUTO	TIPO	LLAVE	NOMBRE LLAVE	NULO	COMENTARIOS
REVFOLIO	NUMBER-10	Llave Primaria	DSREV_PK	NO	Folio de Revisión
STATUS	VARCHAR2-2			NO	Status del Proyecto
SUBJECT	VARCHAR2-1000			SI	Título del Email
MENSAJE	VARCHAR2-4000			SI	Mensaje del Email
LINK	VARCHAR2-4000			SI	Liga de pruebas (URL)
FECHAALTA	VARCHAR2-1000			SI	Fecha de alta de la Revisión

Tabla 2.9. Descripción de campos de la tabla DSREVISION

DSREVISIONMAILS

ATRIBUTO	TIPO	LLAVE	NOMBRE LLAVE	NULO	COMENTARIOS
REVFOLIO	NUMBER-10	Llave Primaria, Llave Foránea	DSREVM_DSREV_FK	NO	Folio de Revisión
EMAIL	VARCHAR2-100	Llave Primaria	DSREVM_PK	NO	Email destino
FECHASOLREVISION	DATE			NO	Fecha de solicitud de revisión

Tabla 2.10. Descripción de campos de la tabla DSREVISIONMAILS

DSPRODUCCION

En la tabla se registran los pases a producción de los requerimientos. Se cuenta con varios campos de tipo fecha y usuario, esto con el fin de ir grabando los usuarios y fechas en que se dieron ciertos eventos. Por ejemplo el usuario que genera el pase a producción es por lo general el analista líder (aunque puede ser cualquiera), el usuario que aprueba el pase es aquel con el rol de "Control internos" (gerente de sistemas), el usuario que libera el pase puede ser cualquier analista, los usuarios con el rol de "Control fuentes" son los que tramitan y finalizan el pase a producción. Ver Tabla 2.11.

ATRIBUTO	TIPO	LLAVE	NOMBRE LLAVE	NULO	COMENTARIOS
PRODFOLIO	NUMBER-10	Llave Primaria	DSPROD_PK	NO	Folio del Pase a Producción
USUCLAVEAN	VARCHAR2-30	Llave Foránea	DSPROD_USU1_FK	NO	Clave del Analista
PROYFOLIO	NUMBER-10	Llave Foránea	DSPROD_DSPROY_FK	NO	Folio del Proyecto
FECHAALTA				NO	Fecha Alta
STATUS	VARCHAR2-2			NO	Status del Pase a Producción (PEND/PORAPRO/APROB/ CANC/PROD)
FECHAPROD				SI	Fecha del Pase a Producción
USUCLAVEPROD	VARCHAR2-30	Llave Foránea	DSPROD_USU2_FK	SI	Usuario que efectuo el Pase a Producción
OBSERVACIONES	VARCHAR2-4000			SI	Observaciones para realizar el pase a Producción
COMENTARIOSREVISION	VARCHAR2-4000			SI	Comentarios
FECHAAPROB				SI	Fecha de Aprobación del Pase a Producción
USUCLAVEAPROB	VARCHAR2-30	Llave Foránea	DSPROD_USU3_FK	SI	Usuario que aprobo el Pase a Producción
FECHALIBERA				SI	Fecha de Liberación
USUCLAVELIBERA	VARCHAR2-30			SI	Usuario que libero el Pase a Producción
FECHATRAMITE				SI	Fecha en que se comenzó el tramite del Pase a Producción
USUCLAVETRAMITE	VARCHAR2-30	Llave Foránea	DSPROD_USU4_FK	SI	Usuario que inició el trámite del Pase a Producción
NOCTURNO	NUMBER-1			NO	Indica si el Pase a Producción se realiza por la Noche

Tabla 2.11. Descripción de campos de la tabla DSPRODUCCION

DSDOCUMENTOSPROD

Los programas que pertenecen a un requerimiento externo, debieron ser revisados en su momento por un usuario final. Por lo tanto cuando estos programas se asocian a un pase a producción se necesita almacenar los *emails* del usuario que avalen dicha revisión. Ver Tabla 2.12.

ATRIBUTO	TIPO	LLAVE	NOMBRE LLAVE	NULO	COMENTARIOS
PRODFOLIO	NUMBER-10	Llave Primaria, Llave Foránea	DSDOCPR_PK, DSDOCPR_DSPROD	NO	Folio del Pase a Producción
PARTIDA	NUMBER-10	Llave Primaria	DSDOCPR_PK	NO	Número de Documento
DOCUMENTO				SI	Documento
FECHAALTA				NO	Fecha de Alta
NOMBREARCHIVO	VARCHAR2-200			NO	Nombre del Documento

Tabla 2.11. Descripción de campos de la tabla DSDOCUMENTOSPROD

2.8.1. Diseño relacional de la Base de Datos

El diagrama relacional de la base de datos para el CDS permite observar de manera clara las tablas con sus atributos, llaves primarias, llaves foráneas y las relaciones que existen entre si; gráficamente queda representado en la figura 2.26:

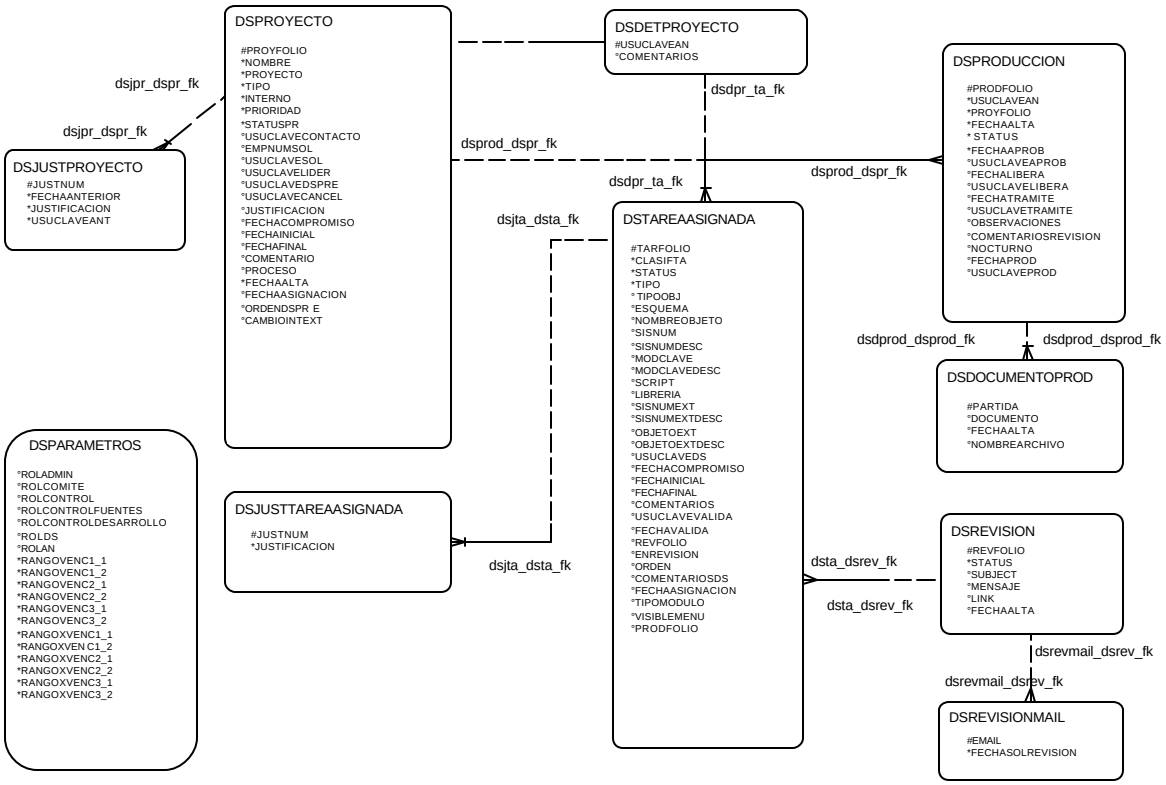

Figura 2.26. Diagrama Relacional de la Base de Datos para el sistema CDS

CAPITULO 3

DESARROLLO DEL SISTEMA

En el capitulo anterior se analizó y estableció el modelo conceptual de base de datos. Con estos fundamentos se presentarán los componentes mas importantes tanto de la Base de Datos como de las herramientas para la creación de las interfases del sistema.

Componentes de Base de Datos.

3.1. Definición de objetos de Base de Datos (tablas, atributos, definición de datos)

Debido a que el CDS formará parte del *ERP*, es necesario describir los distintos tipos de archivo que se emplean para la creación de objetos dentro de la BD. Para identificar fácilmente que tipo de objeto se creará se utilizan las siguientes nomenclaturas:

- .sch.- Archivo que contiene instrucciones para crear un nuevo esquema en la bd.
- .tab.- Archivo que contiene instrucciones para la creación de una tabla en la bd.
- .con.- Archivo que contiene instrucciones para crear *constraints* (restricciones) sobre una tabla.
- .ind.- Archivo que contiene instrucciones para la creación de índices sobre una tabla en particular.
- .grn.- Archivo que contiene instrucciones para otorgar permisos de bd.
- .syn.- Archivo que contiene las instrucciones para crear sinónimos a los objetos de base de datos.
- .vw.- Código para crear una vista de bd.
- .prc.- Código para crear un procedimiento.
- .fnc.- Código para crear una función.
- .pks.- Definición de un paquete de base de datos.
- .pkb-.- Cuerpo de un paquete de base de datos.
- .sql.- Archivo de propósito general (Para ejecutar sentencias *DML*, etc.)
- .alter.- Archivo que modifica la estructura de alguna tabla o vista de la bd.

A continuación se muestra el contenido de algunos archivos cuyo objetivo es la creación de objetos de base de datos.

Primeramente se necesita crear el esquema que será el contenedor lógico de todos los objetos de base de datos. El archivo que contiene las instrucciones necesarias para la creación del esquema se muestra a continuación:

creaEsquema.sch

```
--
-- CDS (Creación de Usuario)
--

CREATE USER CDS
  IDENTIFIED BY CDS08
  DEFAULT TABLESPACE CDSTAB
  TEMPORARY TABLESPACE SNTRTMP
  PROFILE DEFAULT
  ACCOUNT UNLOCK
/
```

```
--
-- Se asigna rol que contiene permisos para poder crear tablas, procedimientos, etc.
-- dentro de la base de datos
--

GRANT RESOURCE TO CDS
/

--
-- Se asignan algunos privilegios adicionales para manipulación de objetos,
-- así como también de interacción con otros esquemas  de la base de datos
--
GRANT SELECT ANY SEQUENCE TO CDS
/
GRANT SELECT ANY TABLE TO CDS
/
GRANT UNLIMITED TABLESPACE TO CDS
/

GRANT EXECUTE ON CAS.BDNOMBREUSUARIO TO CDS
/
GRANT EXECUTE ON CAS.BDSENDMAIL TO CDS
/
GRANT REFERENCES ON CAS.EMPRESAS TO CDS
/
GRANT REFERENCES ON CAS.USUARIOS TO CDS
/
```

Para poder consultar, modificar, eliminar o insertar registros dentro de las tablas del CDS, es necesario crear el rol de base de datos pertinente. También dentro de este tipo de roles se otorgan los permisos necesarios para la ejecución de los diversos procedimientos, funciones y paquetes que contiene el esquema.

El contenido del *script* que contiene la creación de los roles asociados al esquema CDS se muestra a continuación:

creaRolesBD.sql

```
--
-- OBJS_CDS_CT
--

CREATE ROLE OBJS_CDS_CT NOT IDENTIFIED
/
CREATE ROLE OBJS_CDS_LE NOT IDENTIFIED
/
```

Las instrucciones para crear los roles a nivel aplicación:

creaRolesSistema.sql

```
--
-- Creación de roles empleados por el sistema CDS
--

CREATE ROLE ANALISTA NOT IDENTIFIED
/
CREATE ROLE COMITE NOT IDENTIFIED
/
CREATE ROLE CONTROLDESARROLLO NOT IDENTIFIED
/
CREATE ROLE CONTROLFUENTES NOT IDENTIFIED
/
CREATE ROLE CONTROLINTERNOS NOT IDENTIFIED
/
```

```
CREATE ROLE DESARROLLADOR NOT IDENTIFIED
/
```

Dentro del esquema CDS se crearán las tablas que se describieron en el capítulo anterior. El siguiente es un ejemplo completo de la creación de la tabla ***dsproyectos***.

dsproyectos.tab

```
--
-- DSPROYECTOS  (Instrucciones para crear la tabla)
--

CREATE TABLE CDS.DSPROYECTOS
(
  PROYFOLIO            NUMBER(10),
  NOMBRE               VARCHAR2(300 BYTE)       NOT NULL,
  PROYECTO             NUMBER(1)                DEFAULT 0          NOT NULL,
  TIPO                 VARCHAR2(2 BYTE)         NOT NULL,
  INTERNO              NUMBER(1)                DEFAULT 0          NOT NULL,
  PRIORIDAD            NUMBER(1)                NOT NULL,
  STATUSPR             VARCHAR2(2 BYTE)         NOT NULL,
  USUCLAVEANCONTACTO   VARCHAR2(30 BYTE),
  USUCLAVESOL          VARCHAR2(30 BYTE),
  EMPNUMSOL            NUMBER(3),
  USUCLAVELIDER        VARCHAR2(30 BYTE),
  JUSTIFICACION        VARCHAR2(4000 BYTE),
  NDESARROLLADORES     NUMBER(2),
  FECHACOMPROMISO      DATE,
  FECHAINICIAL         DATE,
  FECHAFINAL           DATE,
  USUCLAVEREVISION     VARCHAR2(30 BYTE),
  FECHAINICIOREVISION  DATE,
  FECHAFINALREVISION   DATE,
  COMENTARIOS          VARCHAR2(4000 BYTE),
  PROCESO              VARCHAR2(4000 BYTE),
  FECHAALTA            DATE                     DEFAULT SYSDATE    NOT NULL,
  FECHAASIGNACION      DATE,
  USUCLAVECANCEL       VARCHAR2(30 BYTE),
  USUCLAVEDSPRE        VARCHAR2(30 BYTE),
  ORDENDSPRE           NUMBER(10),
  CAMBIOINTEXT         NUMBER(1)                DEFAULT 0          NOT NULL
)
TABLESPACE CDSTAB
/
```

dsproyectos.ind

```
--
-- DSPR1_USU_FK_I  (Index)
--

CREATE INDEX CDS.DSPR1_USU_FK_I ON CDS.DSPROYECTOS
(USUCLAVEANCONTACTO)
TABLESPACE CDSINDX
/


--
-- DSPR2_USU_FK_I  (Index)
--

CREATE INDEX CDS.DSPR2_USU_FK_I ON CDS.DSPROYECTOS
(USUCLAVESOL)
TABLESPACE CDSINDX
/
```

```
--
-- DSPR3_USU_FK_I  (Index)
--

CREATE INDEX CDS.DSPR3_USU_FK_I ON CDS.DSPROYECTOS
(USUCLAVELIDER)
TABLESPACE CDSINDX
/


--
-- DSPR4_USU_FK_I  (Index)
--

CREATE INDEX CDS.DSPR4_USU_FK_I ON CDS.DSPROYECTOS
(USUCLAVEREVISION)
TABLESPACE CDSINDX
/


--
-- DSPR_EMP_FK_I  (Index)
--

CREATE INDEX CDS.DSPR_EMP_FK_I ON CDS.DSPROYECTOS
(EMPNUMSOL)
TABLESPACE CDSINDX
/


--
-- DSPR_PK  (Index)
--

CREATE UNIQUE INDEX CDS.DSPR_PK ON CDS.DSPROYECTOS
(PROYFOLIO)
TABLESPACE CDSINDX
/
```

dsproyectos.con

```
--
-- Constraints para la tabla DSPROYECTOS
--

ALTER TABLE CDS.DSPROYECTOS ADD (
  CONSTRAINT DSPR_CK1
 CHECK (PROYECTO IN (0,1,2)))
/

ALTER TABLE CDS.DSPROYECTOS ADD (
  CONSTRAINT DSPR_CK2
 CHECK (INTERNO IN (0,1)))
/

ALTER TABLE CDS.DSPROYECTOS ADD (
  CONSTRAINT DSPR_CK3
 CHECK (PRIORIDAD IN (0,1,2,3)))
/

ALTER TABLE CDS.DSPROYECTOS ADD (
  CONSTRAINT DSPR_CK4
 CHECK (STATUSPR IN ('PE','AS','AN','DE','VA','TE','CA','SU')))
/

--
-- Constraint de llave primaria para la tabla DSPROYECTOS
--

ALTER TABLE CDS.DSPROYECTOS ADD (
  CONSTRAINT DSPR_PK
 PRIMARY KEY
```

```
 (PROYFOLIO)
    USING INDEX
    TABLESPACE CDSINDX)
/


--
-- Constraints de llave foránea para la tabla DSPROYECTOS
--

ALTER TABLE CDS.DSPROYECTOS ADD (
  CONSTRAINT DSPR1_USU_FK
 FOREIGN KEY (USUCLAVEANCONTACTO)
 REFERENCES CAS.USUARIOS (USUCLAVE))
/

ALTER TABLE CDS.DSPROYECTOS ADD (
  CONSTRAINT DSPR2_USU_FK
 FOREIGN KEY (USUCLAVESOL)
 REFERENCES CAS.USUARIOS (USUCLAVE))
/

ALTER TABLE CDS.DSPROYECTOS ADD (
  CONSTRAINT DSPR3_USU_FK
 FOREIGN KEY (USUCLAVELIDER)
 REFERENCES CAS.USUARIOS (USUCLAVE))
/

ALTER TABLE CDS.DSPROYECTOS ADD (
  CONSTRAINT DSPR4_USU_FK
 FOREIGN KEY (USUCLAVEREVISION)
 REFERENCES CAS.USUARIOS (USUCLAVE))
/

ALTER TABLE CDS.DSPROYECTOS ADD (
  CONSTRAINT DSPR_EMP_FK
 FOREIGN KEY (EMPNUMSOL)
 REFERENCES CAS.EMPRESAS (EMPNUM))
/
```

dsproyectos.grn

```
--
-- DSPROYECTOS  (Privilegios)
--


GRANT SELECT ON CDS.DSPROYECTOS TO BITACORAS
/

GRANT DELETE, INSERT, SELECT, UPDATE ON CDS.DSPROYECTOS TO OBJS_CDS_CT
/

GRANT SELECT ON CDS.DSPROYECTOS TO OBJS_CDS_LE
/
```

dsproyectos.syn

```
--
-- DSPROYECTOS  (Sinonimo)
--

CREATE PUBLIC SYNONYM DSPROYECTOS FOR CDS.DSPROYECTOS
/
```

Las demás tablas de Base de Datos que empleara el CDS se crean bajo los estándares y nomenclaturas empleadas en el ejemplo previo.

La generación de notificaciones de correos electrónicos que se enviaran en el sistema, se empleara un paquete de base de datos, este es el código que contiene tanto el archivo de definición como el archivo del cuerpo del paquete.

bdCtrlProy.pks

```
CREATE OR REPLACE PACKAGE CDS.bdCtrlProy AS

--
-- DEFINICION DE CURSORES
--


    --CURSOR PARA OBTENER LA INFORMACION DEL PROYECTO

    CURSOR curProyectos(pcProyFolio dsproyectos.proyfolio%TYPE) IS
      SELECT pr.proyfolio,pr.nombre,pr.proyecto,
             DECODE(pr.tipo, 'NU','NUEVO','ME','MEJORA','ER','ERROR') AS tipo,
             pr.usuclavesol, pr.empnumsol, pr.usuclavelider,
             pr.justificacion, pr.proceso, pr.comentarios, pr.prioridad
      FROM   dsproyectos pr
      WHERE  pr.proyfolio = pcProyFolio;

    --CURSOR PARA OBTENER LA MAIL DEL USUARIO

    CURSOR curMailUsuario(pcUsuClaveLider dsproyectos.usuclavelider%TYPE) IS
      SELECT email
      FROM   usuarios
      WHERE  usuclave = pcUsuClaveLider;

    --CURSOR PARA OBTENER LA INFORMACION DE LA EMPRESA

    CURSOR curEmpresas(pcEmpnum dsproyectos.empnumsol%TYPE) IS
      SELECT nombrecorto
      FROM   empresas
      WHERE  empnum = pcEmpnum;


--
-- DEFINICION DE PROCEDIMIENTOS Y FUNCIONES
--


   /*-------------------------------------------------------------------------
   Nombre:      enviarMailLider
   Autor:       Oscar Manuel Júarez Villafaña
   Fecha:       01/07/2008
   Descripción: Cuando el status cambia de pendiente a análisis
                enviar mail al analista definido como líder del proyecto o pendiente
   Parámetros   pProyFolio: Folio del proyecto/pendiente
   -------------------------------------------------------------------------*/

  PROCEDURE enviarMailLider(pProyFolio dsproyectos.proyfolio%TYPE);

   /*-------------------------------------------------------------------------
   Nombre:      enviarMailLider
   Autor:       Oscar Manuel Júarez Villafaña
   Fecha:       01/07/2008
   Descripción: Cuando se detecta un cambio de líder, se avisa al líder anterior y al
                nuevo líder
   Parámetros   pProyFolio:  Folio del proyecto/pendiente
                pLiderAnt:   Clave de usuario que estaba dado de alta como lider del
                             proyecto/pendiente
                pLiderAct:   Clave de usuario el va a quedar como lider del
                             proyecto/pendiente
   -------------------------------------------------------------------------*/

  PROCEDURE enviarMailLider(pProyFolio dsproyectos.proyfolio%TYPE,
                            pLiderAnt  dsproyectos.usuclavelider%TYPE,
                            pLiderAct  dsproyectos.usuclavelider%TYPE);
```

```
 /*-------------------------------------------------------------------------
 Nombre:      enviarMailContacto
 Autor:       Oscar Manuel Juárez Villafaña
 Fecha:       01/07/2008
 Descripción: Cuando se cancela un proyecto/pendiente se envía mail al usuario de
              contacto para informar la cancelación del mismo
 Parámetros   pProyFolio:  Folio del proyecto/pendiente
 -------------------------------------------------------------------------*/

PROCEDURE enviarMailContacto(pProyFolio dsproyectos.proyfolio%TYPE);

 /*-------------------------------------------------------------------------
 Nombre:      enviarMailPetProg
 Autor:       Oscar Manuel Juárez Villafaña
 Fecha:       01/07/2008
 Descripción: Cuando se pide descongelar programas al encargado de control de fuentes
 Parámetros   pProyFolio:  Folio del proyecto/pendiente
              pAnalista:   Analista que dio de alta las tareas
             pTarea:       Numero de tarea con la cual se obtiene el
                           módulo, objeto, etc. para descongelar
 -------------------------------------------------------------------------*/

PROCEDURE enviarMailPetProg(pProyFolio dstareasasignadas.proyfolio%TYPE,
                            pAnalista  dstareasasignadas.usuclavean%TYPE,
                            pTarea     dstareasasignadas.tarfolio%TYPE);

 /*-------------------------------------------------------------------------
 Nombre:      enviarMailPetRev
 Autor:       Oscar Manuel Juárez Villafaña
 Fecha:       01/07/2008
 Descripción: Cuando se solicita la revisión de los programas
 Parámetros   pRev:       Folio de revisión que se asigno la tarea
              pDest:      Mail del usuario que va a realizar la revisión del programa
 -------------------------------------------------------------------------*/

PROCEDURE enviarMailPetRev(pRev dsrevision.revfolio%TYPE,
                           pDest usuarios.email%TYPE);


 /*-------------------------------------------------------------------------
 Nombre:      enviarMailSolAprobacion
 Autor:       Oscar Manuel Juárez Villafaña
 Fecha:       01/07/2008
 Descripción: Cuando se solicita la aprobación del usuario de control internos
 Parámetros   pProdfolio: Folio del pase a producción que debe aprobar el usuario
                          de control internos
 -------------------------------------------------------------------------*/

PROCEDURE enviarMailSolAprobacion(pProdfolio dsproduccion.prodfolio%TYPE);


 /*-------------------------------------------------------------------------
 Nombre:      enviarMailAprobacion
 Autor:       Oscar Manuel Juárez Villafaña
 Fecha:       01/07/2008
 Descripción: Cuando el usuario de control internos aprueba el pase a producción
 Parámetros   pProdfolio:  Folio del pase a producción que aprobó el usuario de control
                           internos
              pUsuario:    Clave de usuario que dio de alta el pase a producción
 -------------------------------------------------------------------------*/

PROCEDURE enviarMailAprobacion(pProdfolio dsproduccion.prodfolio%TYPE,
                               pUsuario usuarios.usuclave%TYPE);


 /*-------------------------------------------------------------------------
 Nombre:      enviarMailIncorporacion
 Autor:       Oscar Manuel Juárez Villafaña
 Fecha:       04/08/2008
 Descripción: Cuando se incorpora un nuevo analista al proyecto se le notifica al
              analista lider y al analista nuevo
 Parámetros   pProyFolio:      Folio del proyecto/pendiente
```

```
                pUsuclaveNuevo:  Clave de usuario del analista que se incorporo al
                                 proyecto/pendiente
    ----------------------------------------------------------------------------*/

   PROCEDURE enviarMailIncorporacion(pProyFolio dsproyectos.proyfolio%TYPE,
                                     pUsuclaveNuevo dsproyectos.usuclavelider%TYPE);

    /*----------------------------------------------------------------------------
    Nombre:      enviarMailJustificacion
    Autor:       Oscar Manuel Juárez Villafaña
    Fecha:       27/08/2007
    Descripción: Cuando se cambia la fecha compromiso del proyecto se notifica al control
                 internos

    Parámetros   pProyfolio:  Folio del proyecto al que se le cambio la fecha
                 pJustNum:    Folio de justificación del cambio de la fecha del proyecto

    Comentarios:
    Modificaciones: (Fecha:
                     Autor:
                     Descripción:)
    ----------------------------------------------------------------------------*/

   PROCEDURE enviarMailJustificacion(pProyfolio dsproyectos.proyfolio%TYPE,
                                     pJustNum dsjustproyectos.justnum%TYPE);


    /*----------------------------------------------------------------------------
    Nombre:      enviarMailLiberacion
    Autor:       Oscar Manuel Juárez Villafaña
    Fecha:       06/11/2007
    Descripción: Se envia la información relevante del pase a producción para
                 que el usuario que tiene asignado el rol de controlfuentes
                 comienze a pasarlo a producción
    Parámetros   pProyfolio:  Folio del proyecto al que se le cambio la fecha
                 pUsuario:    Clave del usuario que libero el pase a producción
    ----------------------------------------------------------------------------*/

   PROCEDURE enviarMailLiberacion(pProdFolio dsproduccion.prodfolio%TYPE);


    /*----------------------------------------------------------------------------
    Nombre:      enviarMailProduccion
    Autor:       Oscar Manuel Juárez Villafaña
    Fecha:       06/11/2007
    Descripción: Se envía notificación del pase a producción que ya se subió
                 a producción
    Parámetros   pProdFolio:  Folio del pase a producción
    ----------------------------------------------------------------------------*/

   PROCEDURE enviarMailProduccion(pProdFolio dsproduccion.prodfolio%TYPE);

END;
/
```

```
CREATE OR REPLACE PACKAGE BODY CDS.bdCtrlProy AS

   /*-------------------------------------------------------------------------
   Nombre:      enviarMailLider
   Autor:       Oscar Manuel Júarez Villafaña
   Fecha:       01/07/2008
   Descripción: Cuando el status cambia de pendiente a análisis
                enviar mail al analista definido como líder del proyecto o pendiente
   Parámetros   pProyFolio: Folio del proyecto/pendiente
   -------------------------------------------------------------------------*/


  PROCEDURE enviarMailLider(pProyFolio dsproyectos.proyfolio%TYPE) IS

    vEMail         usuarios.email%TYPE;
    vNombreCorto   empresas.nombrecorto%TYPE;

    vSubject       VARCHAR2(1000);
    vMensaje       VARCHAR2(4000);
    vProyecto      VARCHAR2(20);
    vPrioridad     VARCHAR2(10);

    rProyectos   curProyectos%ROWTYPE;

  BEGIN

    OPEN  curProyectos(pProyFolio);
    FETCH curProyectos INTO rProyectos;
    CLOSE curProyectos;

    OPEN  curEmpresas(rProyectos.empnumsol);
    FETCH curEmpresas INTO vNombreCorto;
    CLOSE curEmpresas;

    --
    --  EVALUCION DE PIPITIYA O PROYECTO
    --

    IF rProyectos.proyecto = 1 THEN
      vProyecto:= 'PROYECTO';
    ELSIF rProyectos.proyecto = 0 THEN
      vProyecto:= 'PIPITIYA';
    ELSE
      vProyecto:= 'BASE DE DATOS';
    END IF;

    --
    --  EVALUCION DE PRIORIDAD DE PENDIENTE O PROYECTO
    --

    IF (rProyectos.prioridad = 1) THEN
      vPrioridad:= 'ALTA';
    ELSIF (rProyectos.prioridad = 2) THEN
      vPrioridad:= 'MEDIA';
    ELSE
      vPrioridad:= 'BAJA';
    END IF;

    vSubject := ' Asignación de Proyecto No.  ' || TO_CHAR(rProyectos.proyfolio);

    vMensaje :=

    'NOMBRE  -> '                          || rProyectos.nombre    || CHR(10) ||
    'TIPO    -> '                          || rProyectos.tipo      || CHR(10) ||
    '(PROYECTO/PIPITIYA/BASE DE DATOS)  -> ' || vProyecto          || CHR(10) ||
    'PRIORIDAD    -> '                     || vPrioridad           || CHR(10) ||
    'EMPRESA SOLICITANTE  -> '             || rProyectos.empnumsol
                                           || '-' || vnombrecorto  || CHR(10) ||
    'USUARIO SOLICITANTE  -> '             || rproyectos.usuclavesol  || '-' ||
                                              bdNombreUsuario(rProyectos.usuclavesol)
                                           || CHR(10) || CHR(10) ||
```

```
    'JUSTIFICACION   -> '                    || rProyectos.justificacion
                                             || CHR(10) || CHR(10)||
    'PROCESO     -> '                        || rProyectos.proceso
                                             || CHR(10) || CHR(10)||
    'COMENTARIOS COMITE  -> ' || rProyectos.comentarios || CHR(10) || CHR(10);

    OPEN  curMailUsuario(rProyectos.usuclavelider);
    FETCH curMailUsuario INTO vEmail;
    CLOSE curMailUsuario;

    IF vEmail IS NOT NULL THEN
      bdSendMail ('cds@interceramic.com', vEMail, vSubject, vMensaje);
    END IF;

  END;

.
.
.

    /*-------------------------------------------------------------------------
    Nombre:      enviarMailProduccion
    Autor:       Oscar Manuel Juárez Villafaña
    Fecha:       06/11/2007
    Descripción: Se envía notificación del pase a producción que ya se subió
                 a producción
    Parámetros   pProdFolio:  Folio del pase a producción
    -------------------------------------------------------------------------*/

  PROCEDURE enviarMailProduccion(pProdFolio dsproduccion.prodfolio%TYPE) IS

    vEMailAnalista             usuarios.email%TYPE;
    vEMailControlFuentes       usuarios.email%TYPE;

    vSubject                   VARCHAR2(1000);
    vMensaje                   VARCHAR2(4000);
    vAtte                      VARCHAR2(100);
    vProyecto                  VARCHAR2(20);
    vPrioridad                 VARCHAR2(20);
    vNombrecorto               VARCHAR2(100);
    vNocturno                  VARCHAR2(100);

    --CURSOR PARA OBTENER LA INFORMACION DEL PROYECTO

    CURSOR curProyectos(pcFolio dsproyectos.proyfolio%TYPE) IS
      SELECT *
      FROM   dsproyectos
      WHERE  proyfolio = pcFolio;

    --CURSOR PARA OBTENER LA INFORMACION DEL PASE A PRODUCCION

    CURSOR curProduccion(pcPase dsproduccion.prodfolio%TYPE) IS
      SELECT *
      FROM   dsproduccion
      WHERE  prodfolio = pcPase;

    --CURSOR PARA OBTENER LOS USUARIOS CON PRIVILEGIOS

    CURSOR curUsuPrivilegiados(pcProdFolio NUMBER,
                               pcUsuClaveLider VARCHAR2) IS
      SELECT usuclave usuario
      FROM   (SELECT pcUsuClaveLider usuclave
              FROM   dual
              UNION
              SELECT r.usuclavelibera usuario
              FROM   dsproyectos p, dsproduccion r
              WHERE  p.proyfolio = r.proyfolio
              AND    r.prodfolio = pcProdFolio);

    rProyectos  curProyectos%ROWTYPE;
    rPase       curProduccion%ROWTYPE;

  BEGIN
```

```
    OPEN  curProduccion(pProdFolio);
    FETCH curProduccion INTO rPase;
    CLOSE curProduccion;

    OPEN  curProyectos(rPase.proyfolio);
    FETCH curProyectos INTO rProyectos;
    CLOSE curProyectos;

    -- DATOS DEL ENCARGADO DE CONTROL DE FUENTES QUE SUBIO A PRODUCCION EL PASE

    OPEN  curMailUsuario(rPase.usuclaveprod);
    FETCH curMailUsuario INTO vEMailControlFuentes;
    CLOSE curMailUsuario;

    vAtte:= bdNombreUsuario(rPase.usuclaveprod);

    --
    -- EVALUACION DE PIPITIYA O PROYECTO
    --

    IF rProyectos.proyecto = 0 THEN
      vProyecto := 'PIPITIYA';
    ELSIF rProyectos.proyecto = 1  THEN
      vProyecto := 'PROYECTO';
    ELSE
      vProyecto := 'BASE DE DATOS';
    END IF;

    --
    -- GENERACION DEL MAIL
    --

    vSubject :=
    'Pase a Producción No. ' || pProdFolio || ' Realizado';

    vMensaje :=
    'EL PASE A PRODUCCION NO.: '                || pProdfolio        || CHR(10) ||
    'CORRESPONDIENTE AL PROYECTO. '             || rPase.proyfolio   || ' - '   ||
                                                   rProyectos.nombre ||
    ' SE REALIZO CORRECTAMENTE.'                || CHR(10) ||
    'TIPO -> '                                  || rProyectos.tipo   || CHR(10) ||
    '(PROYECTO/PIPITIYA/BASE DE DATOS) -> '     || vProyecto         || CHR(10) ||
    'REALIZADO POR -> '                         || rPase.usuclaveprod   || '-' ||
                                                   bdNombreUsuario(rPase.usuclaveprod)
                                                || CHR(10) ||
    to_char(rPase.fechaprod, 'DD/MM/RRRR HH24:MI:SS')
                                               || CHR(10) || CHR(10) || CHR(10) ||
    ' Atte:  ' || vAtte || CHR(10) || CHR(10);

    FOR regPriv IN curUsuPrivilegiados(pProdFolio, rProyectos.usuclavelider) LOOP

      OPEN  curMailUsuario(regPriv.usuario);
      FETCH curMailUsuario INTO vEMailAnalista;
      CLOSE curMailUsuario;

      IF vEMailAnalista IS NOT NULL THEN
        bdSendMail (vEMailControlFuentes, vEMailAnalista, vSubject, vMensaje);
      END IF;

    END LOOP;

  END;

END;
/
```

Componentes para la creación de Interfases de Usuario.

Como se menciono en el capítulo 1, para la creación de las interfases de usuario, se empleara la *Developer Suite 10g.* A continuación se describe de manera puntual el objetivo de los módulos forma y reporte; así como una breve descripción de sus componentes.

3.2. Ejemplo de un módulo forma

Un módulo forma es un programa que almacena un conjunto de objetos en un solo archivo. La pantalla actúa como interfaz de usuario la cual permite visualizar o manipular información de la base de datos. La ventaja de los programas hechos con esta herramienta radica en que esta pensada para interactuar con bases de datos *Oracle*. El elemento principal de un módulo forma es el *"bloque"*, el cual puede estar asociado a un objeto de base de datos tal como una tabla o un procedimiento almacenado.

Cada pantalla contiene uno o más bloques, y estos a su vez otros elementos llamados *"ítems"*. Los datos pertenecientes a cierta tabla se pueden consultar, insertar, borrar o actualizar ya sea mediante estos *"ítems"* (corresponden uno a uno con los campos de una tabla) o a través de instrucciones *PL/SQL*. Generalmente los *"ítems"* aparecen en una pantalla para que el usuario pueda interactuar con ellos. Visualmente los *"ítems"* son controles de interfaz gráfica tales como elementos de texto, casillas de verificación, listas de opciones, botones, etc.

Tanto los *"ítems"* como los *"bloques"* pueden responder a eventos generados sobre ellos (*triggers*). El código donde se programa la funcionalidad que debe realizar cierto componente dentro de un módulo forma puede ir dentro del cuerpo del *"trigger"* o se puede emplear llamados a procedimientos, funciones o paquetes que están definidos dentro de la sección de unidades de programa.

En las figuras 3.1, 3.2 y 3.4 se muestran diversos aspectos de un módulo forma.

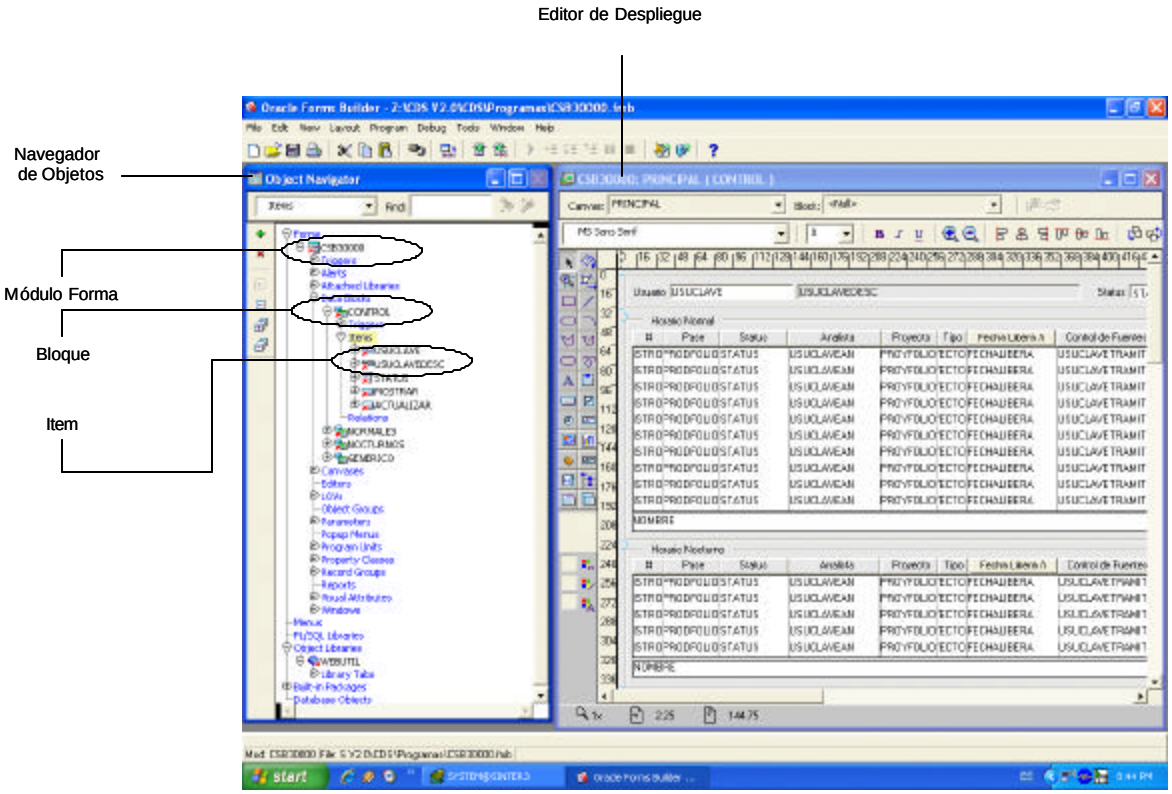

Figura 3.1. Ejemplo de un módulo forma. Modelo de despliegue.

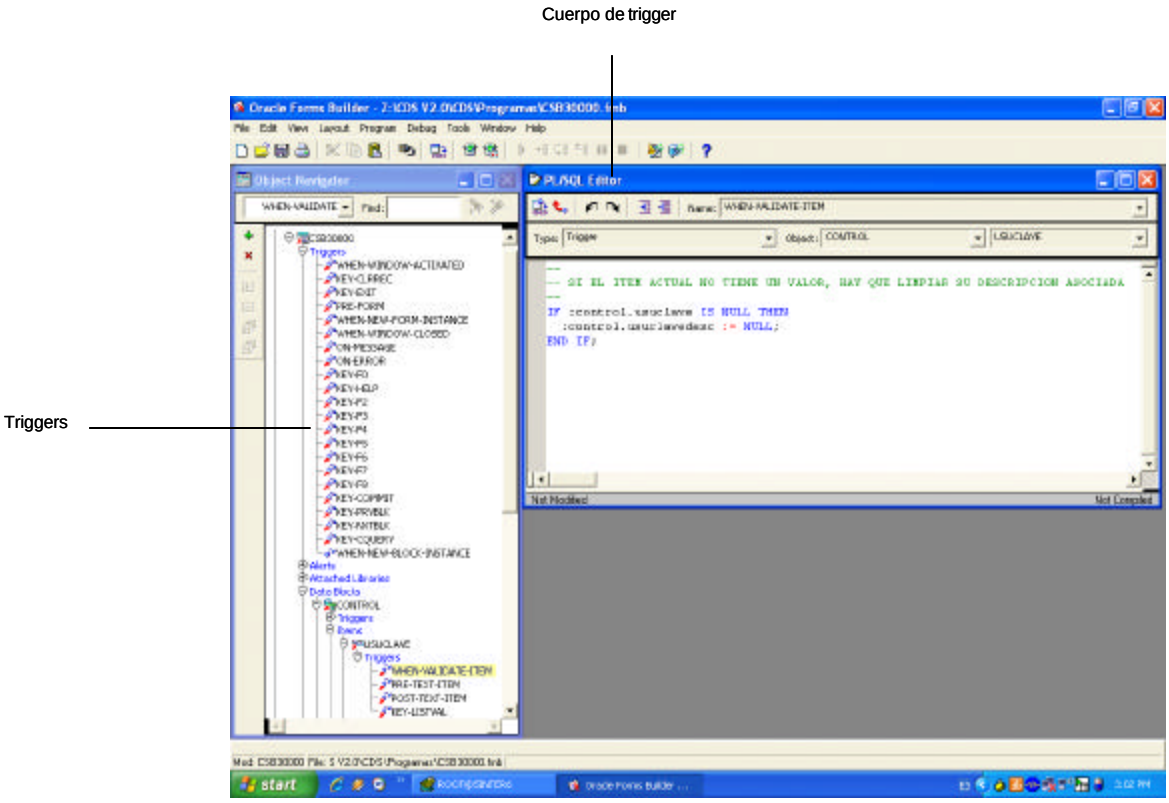

Figura 3.2. Ejemplo de un módulo forma. Editando un disparador (*trigger)*.

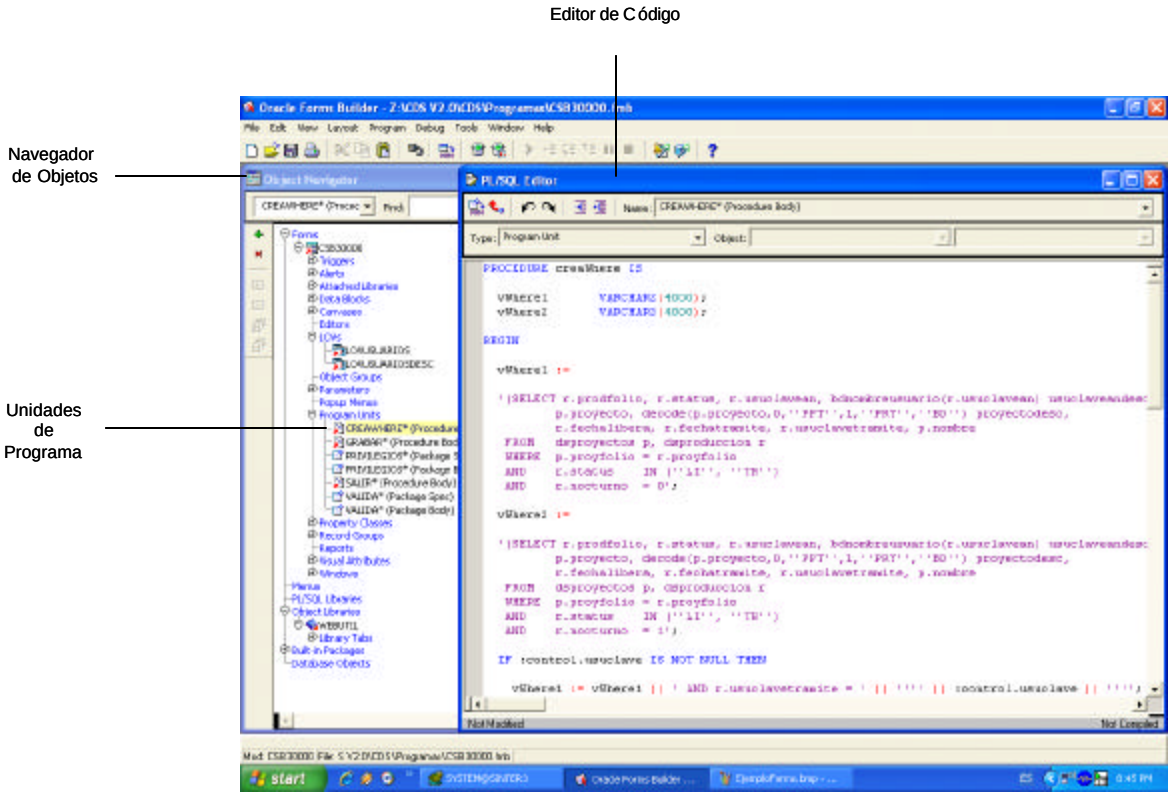

Figura 3.3. Ejemplo de un módulo forma. Creando una unidad de programa.

3.3. Ejemplo de un módulo reporte

Mediante *Oracle Reports Builder* se pueden desarrollar programas para análisis de información y reporteo. Un módulo reporte es aquel que se desarrolla mediante esta herramienta. Tanto *Oracle Reports* como *Oracle Forms* contienen un modelo lógico y un módulo de despliegue de información. Dentro de *Reports*, al modelo lógico se le conoce como modelo de datos, y es aquí donde se define la lógica para acceder a la base de datos. La forma de hacerlo es mediante consultas o procedimientos PL/SQL.

Dentro del modelo de datos de un modulo reporte, se tienen los siguientes componentes:

Query.- Es una consulta hacia la base de datos.
Group (Grupo de Datos).- Es un agrupador lógico de datos, el cual esta ligado de manera directa a los grupos de repetición en el modelo de despliegue.
Column (Columna).- Asociado a un valor resultante de una columna regresada en la definición de una consulta. Las columnas se asignan como fuentes de información dentro del modelo de despliegue de un *field*.

También existen otro tipo de columnas, las cuales sirven para hacer más poderosa esta herramienta ya que mediante ellas se pueden realizar nuevamente accesos a la base de datos para consultar más información, realizar cálculos o simplemente para mantener información.

En el modelo de despliegue, se tienen varios tipos de elementos, los cuales permiten crear reportes desde muy planos hasta muy elaborados como por ejemplo el formato de una factura de venta, todo depende del objetivo del reporte.

Los componentes principales del modelo de despliegue son (Ver Fig. 3.4 y 3.5):

Frame.- Contenedor de objetos tales como *repeating frames*, *fields*, *labels*, etc.
Repeating Frame.- Se asocian a los agrupadores lógicos de datos para desplegar información resultante de ejecutar una consulta en la base de datos.
Field.- Componente donde se asocia el valor de una columna.
Label.- Etiqueta.
Image.- Objeto que permite mostrar elementos de tipo .jpg, .gif, .bmp dentro del reporte.

Cabe resaltar que esta herramienta permite producir salidas hacia diferentes tipos de destino y en diversos formatos. Los destinos pueden ser: pantalla, impresora, email, ftp entre otros. Dentro de los formatos que puede manejar se tienen: archivos de texto delimitados por tabuladores, archivos tipo .rtf, .pdf, .xls, .html, y .jsp.

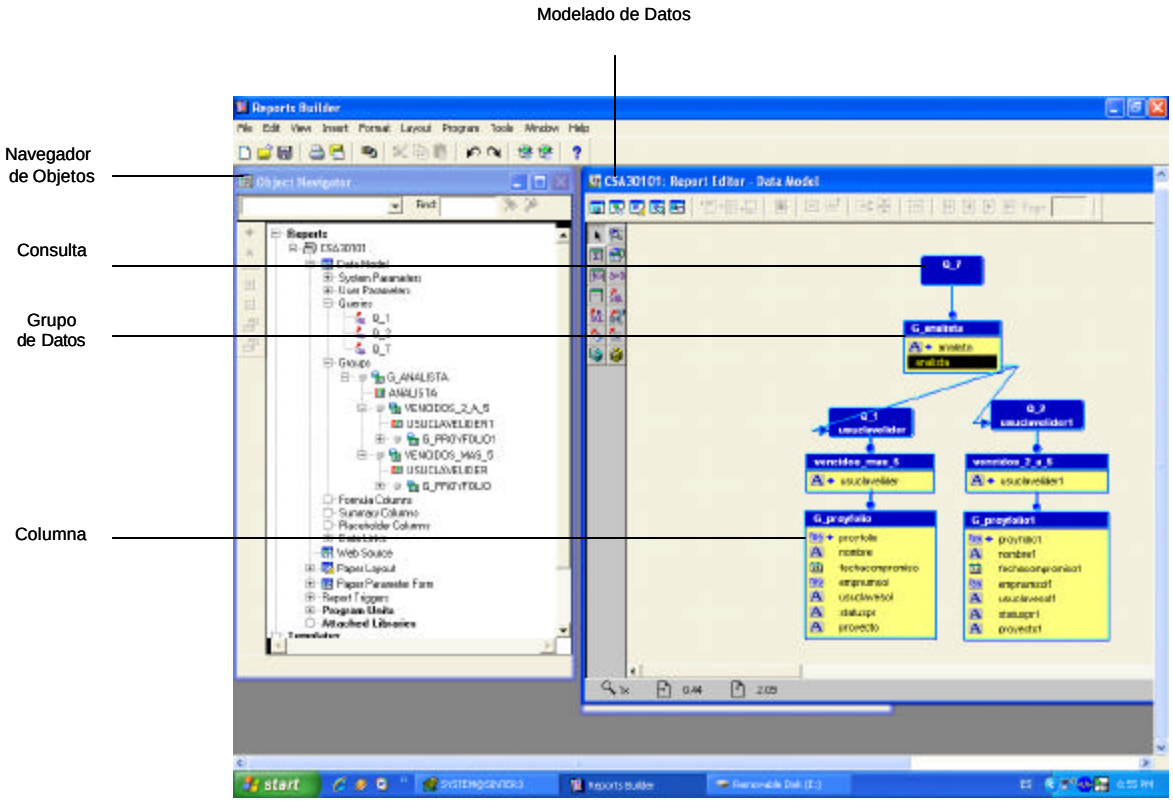

Figura 3.4. Ejemplo de un módulo reporte. Modelo de datos.

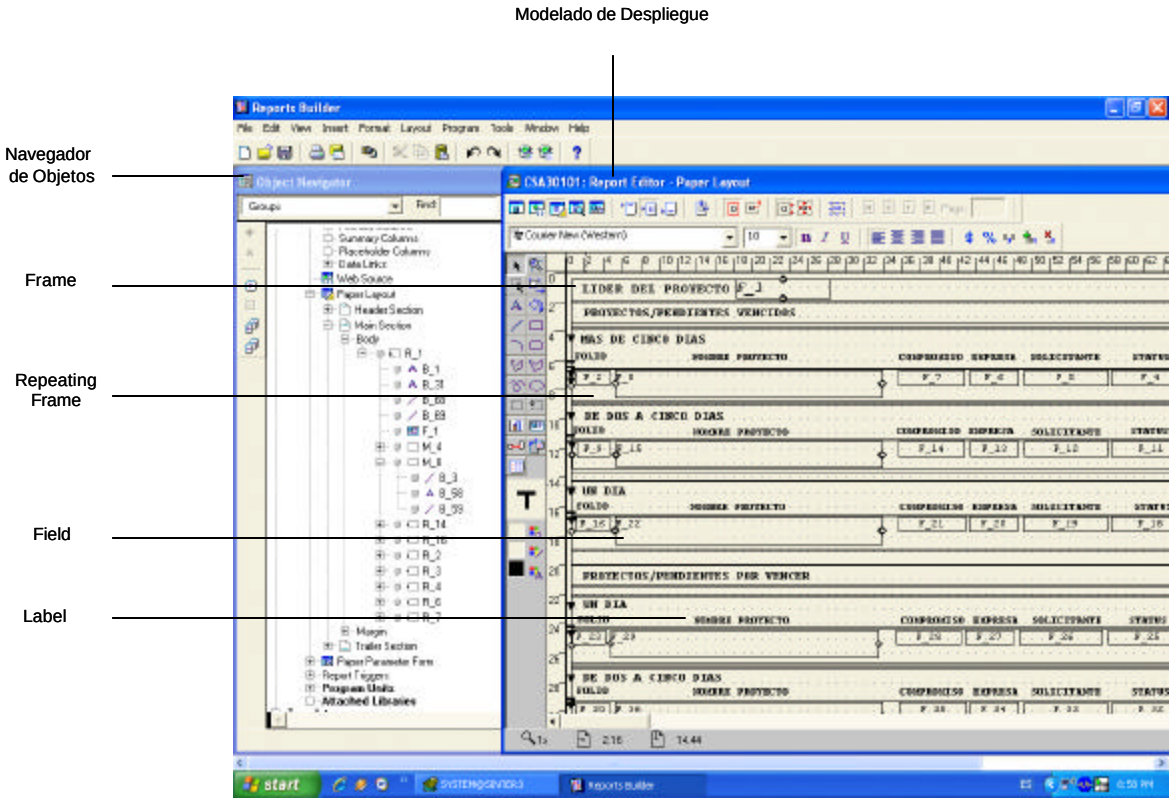

Figura 3.5. Ejemplo de un módulo reporte. Modelo de despliegue.

CAPITULO 4

CASO DE ESTUDIO

Dentro del desarrollo de un sistema existe una fase de pruebas en la cual se validan las operaciones básicas y las funciones que contiene. En este capítulo se presenta un ejemplo funcional del sistema el cual pretende mostrar de inicio a fin el proceso por el cual pasa un requerimiento de usuario.

4.1. Presentación del Sistema

Como se ha mencionado anteriormente, el objetivo principal del sistema es proporcionar módulos que permitan administrar los proyectos surgidos por modificaciones, corrección de errores o mejoras del ERP. También es necesario contar con módulos que permitan llevar un seguimiento de la creación y modificación de objetos o programas del *ERP* derivados de los proyectos.

El sistema se compone de diversos módulos, los cuales permiten:

- La captura, asignación y seguimiento de proyectos.
- La asignación de tareas.
- El pase a producción o instalación de los programas modificados y/o creados durante el proyecto.
- Extracción de información para poder supervisar los avances de los proyectos, las cargas de trabajo de los analistas y desarrolladores.
- Extracción de información para proporcionar los datos necesarios tanto al personal de sistemas como a auditores externos.

En la Fig. 4.1. Se muestra el menú principal del sistema, mediante el cual se pueden acceder a los módulos descritos previamente.

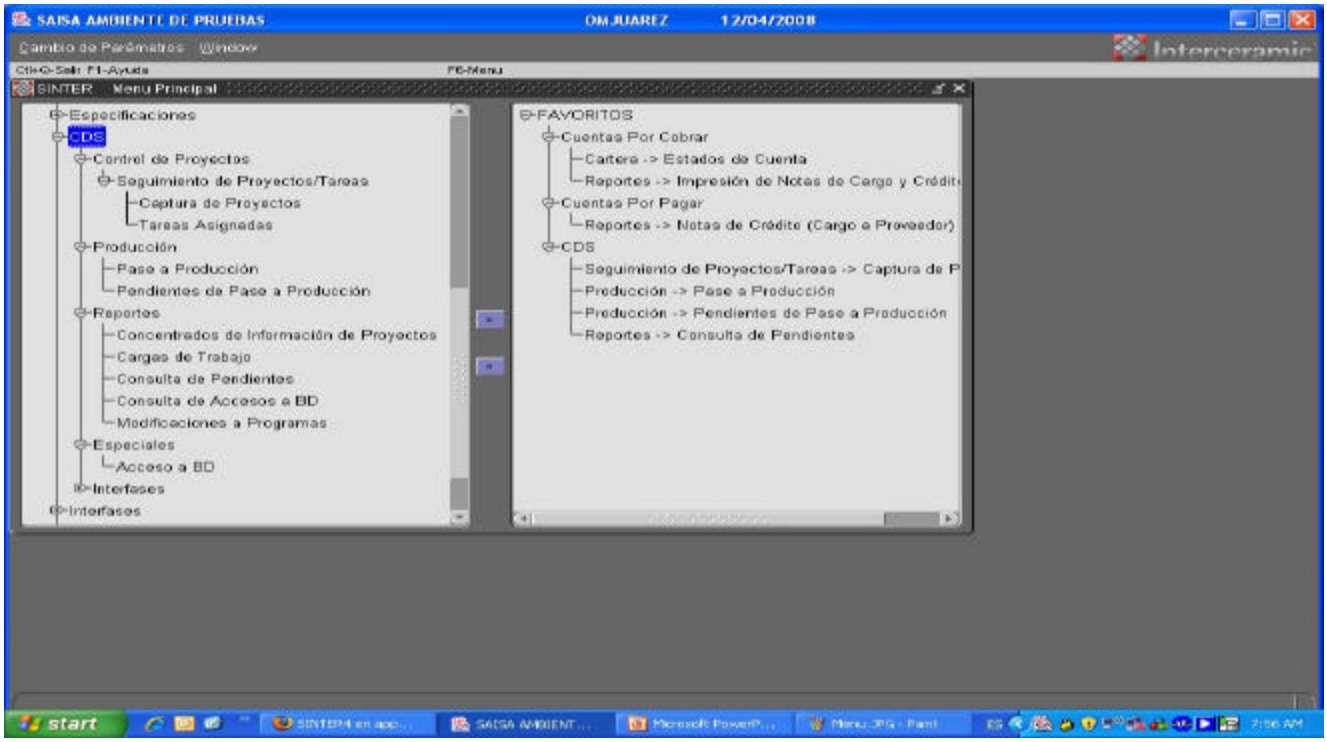

Figura 4.1. Menú Principal del Sistema.

4.2. Funcionamiento del Sistema.

En las siguientes imágenes se pretende mostrar el funcionamiento de las distintas pantallas a través de un ejemplo práctico.

Captura de Proyectos

En este módulo se dan de alta los requerimientos ya sean provenientes de usuarios (requerimientos externos) o por el propio personal de sistemas (requerimientos internos). En esta captura se asigna una prioridad (alta/media/baja) y que tipo de proyecto es (mejora/error/modificación). Posteriormente el gerente de sistemas junto con el coordinador de proyectos y/o comité asigna recursos (analistas/desarrolladores) a los requerimientos capturados (con status PENDIENTE). Ver Fig. 4.2.

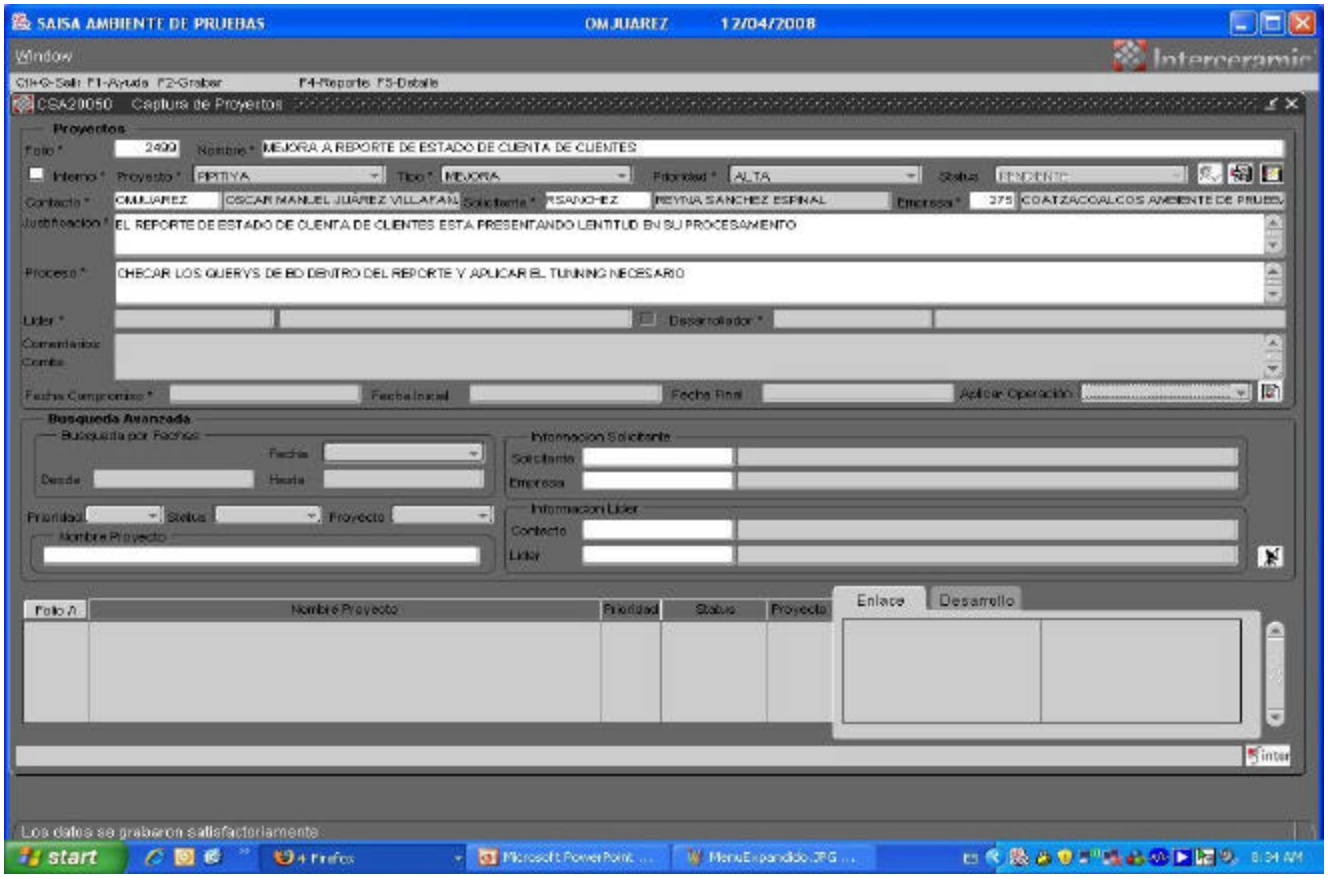

Figura 4.2. Pantalla de Captura de Proyectos

Los status por los que pasa el proyecto son los siguientes:

- PE – PENDIENTE. Status inicial de un requerimiento.
- AS – ASIGNADO. En este status se asigna al analista encargado del proyecto y a un recurso de programación inicial (desarrollador).
- AN – ANALISIS. En esta etapa, el analista verifica que objetos se modificarán o crearán y procede a registrarlos en el sistema.
- DE – DESARROLLO. Este status indica que el desarrollador ya esta trabajando en la modificación o creación de los objetos que se le asignaron.
- VA – VALIDACION. Indica cuando el proyecto esta siendo evaluado por el usuario que solicitó el cambio, mejora, etc.
- TE – TERMINADO. Un proyecto se encuentra en este status cuando se instaló de forma total en el sistema.
- CA – CANCELADO. Este status indica que el requerimiento no procedió.

Al momento de asignar el requerimiento se envía una notificación por correo electrónico al analista que se designo como líder, tal como se muestra en las siguientes imágenes. Ver Fig. 4.3 y 4.4.

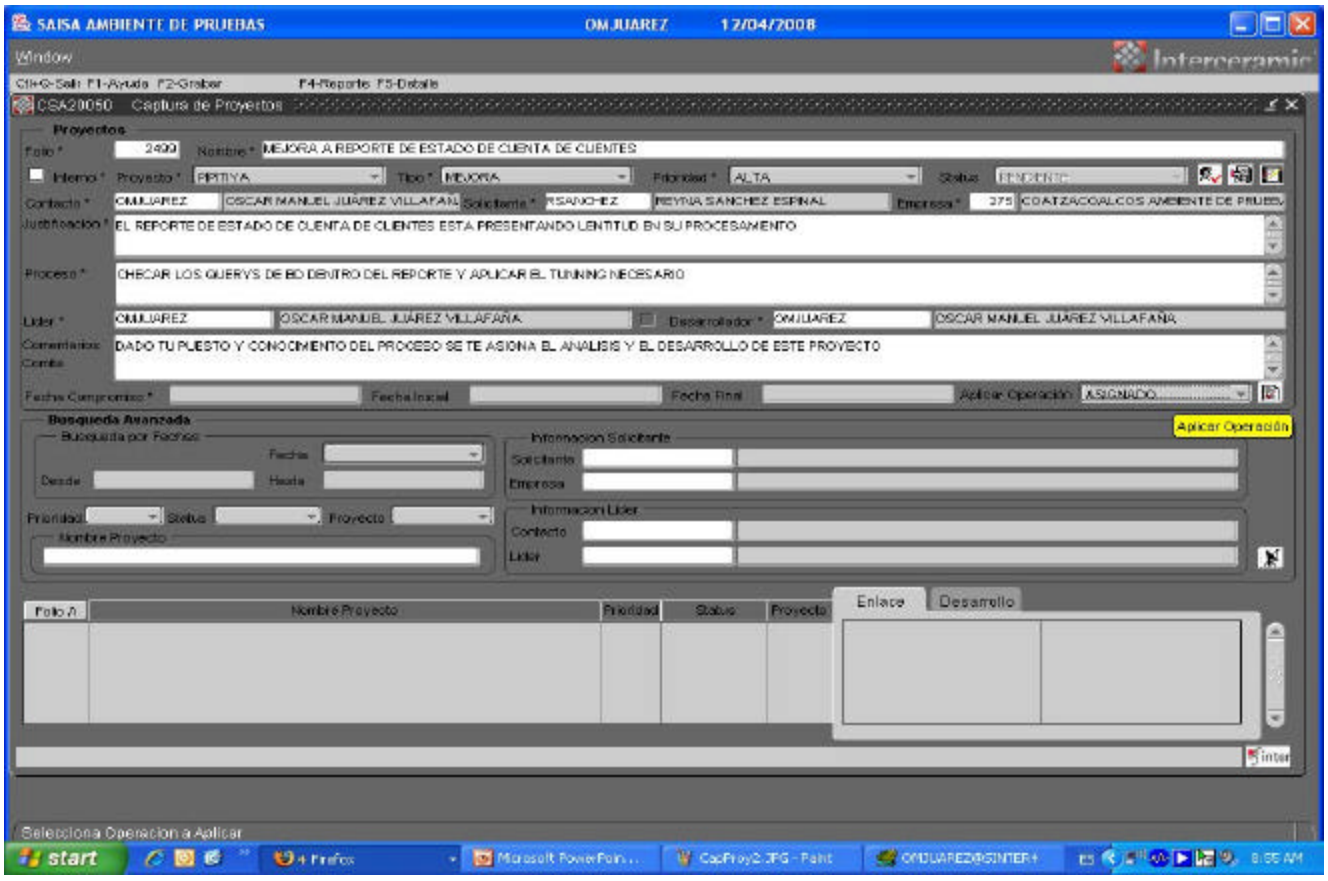

Figura 4.3. Asignación del Proyecto.

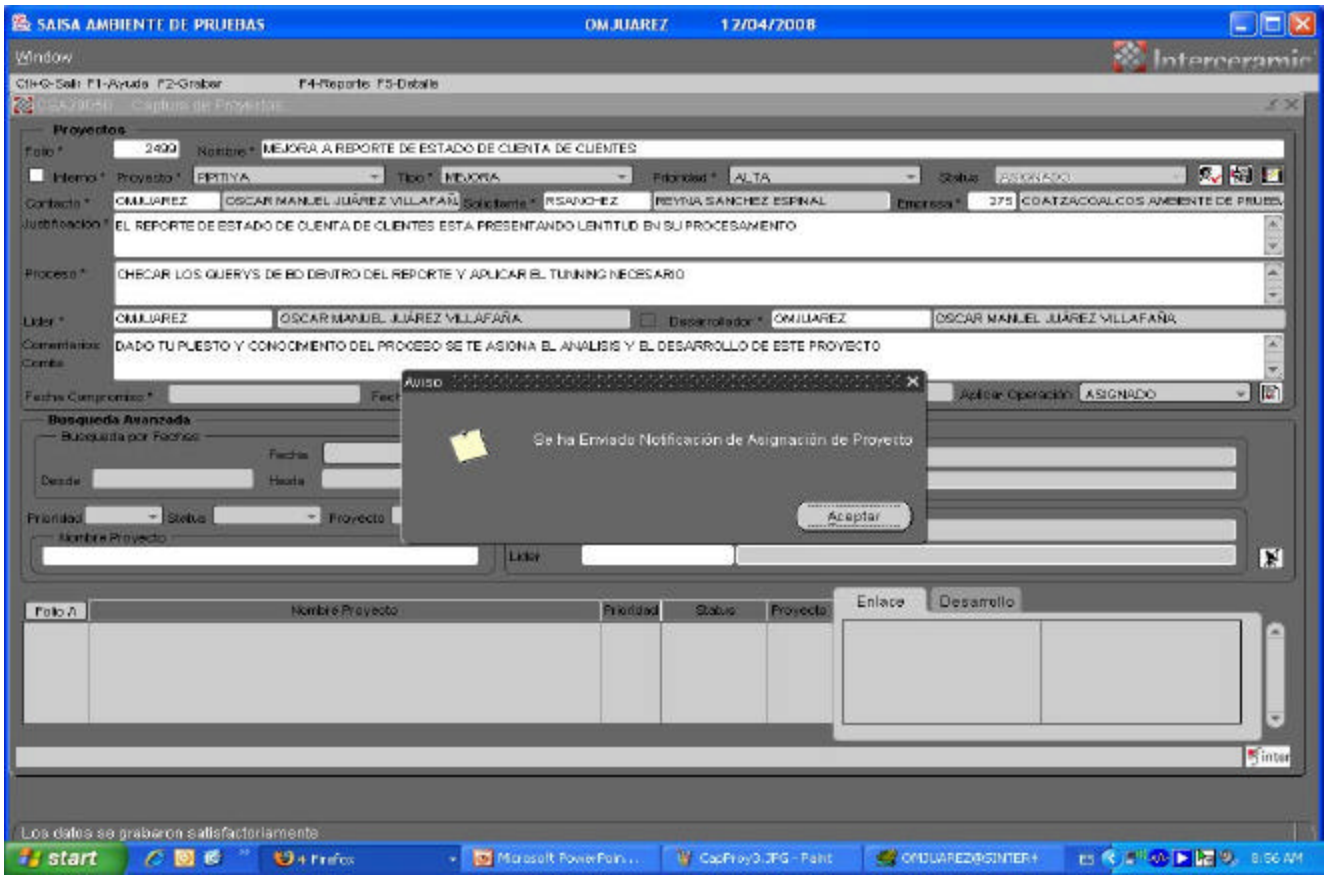

Figura 4.4. Notificación de la Asignación del Proyecto.

Tareas Asignadas.

En esta pantalla se capturan los objetos que se modificaran o crearán dentro del proyecto. Las tareas se clasifican en Modificación, Nueva y sin Clasificación. Estas clasificaciones aplican sobre objetos de base de datos, objetos de aplicación o *scripts.* Ver. Fig. 4.5.

Como ejemplo se muestra la captura de la modificación de un objeto de aplicación. (Un reporte de estado de cuenta de clientes que necesita ser optimizado).

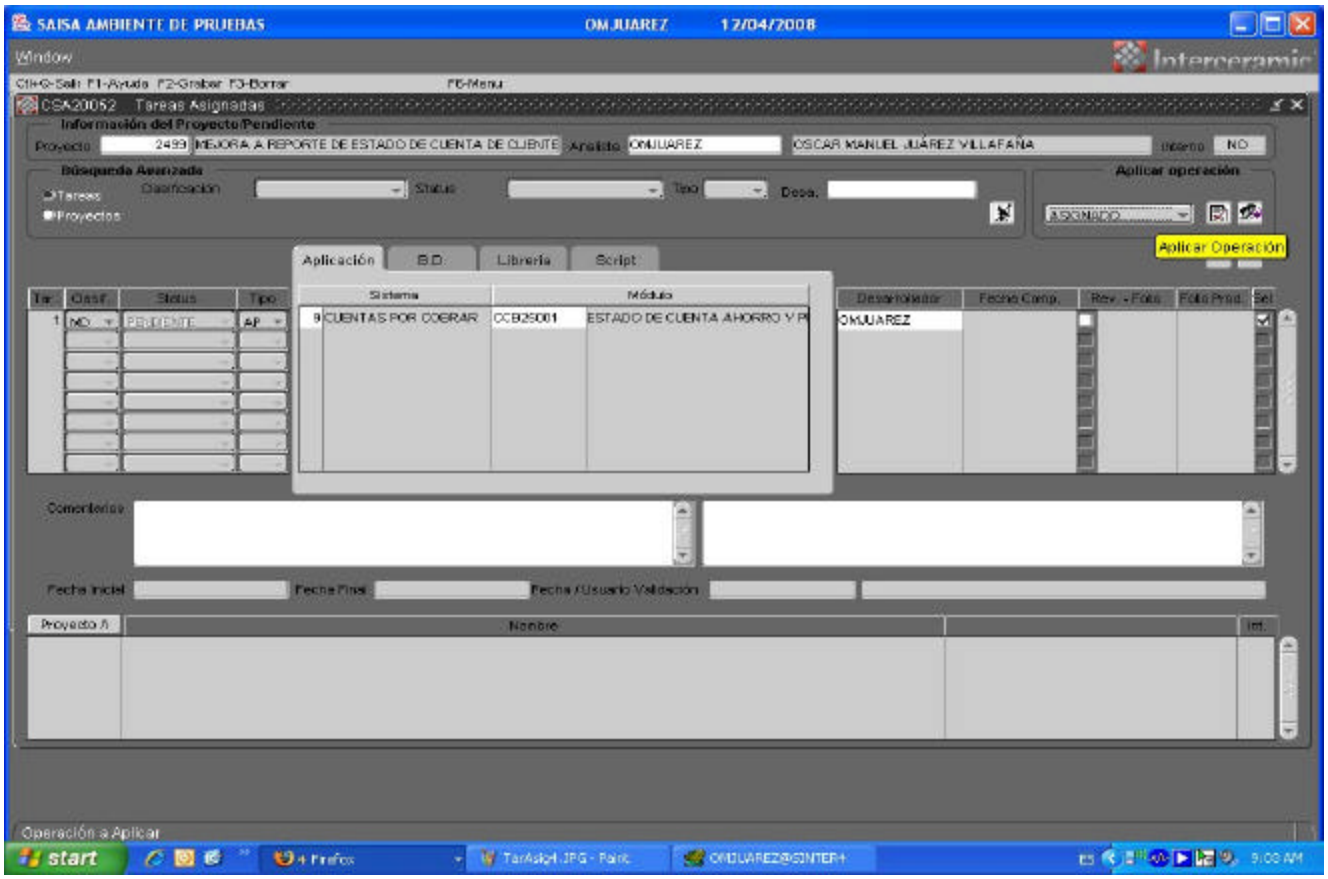

Figura 4.5. Captura de Tareas asignadas al Proyecto.

Una vez capturada la tarea, se le asigna a un programador y se captura la fecha de término (fecha compromiso) del proyecto. Cuando el analista asigna la tarea al programador se le envía una notificación vía email. Ver Fig. 4.6.

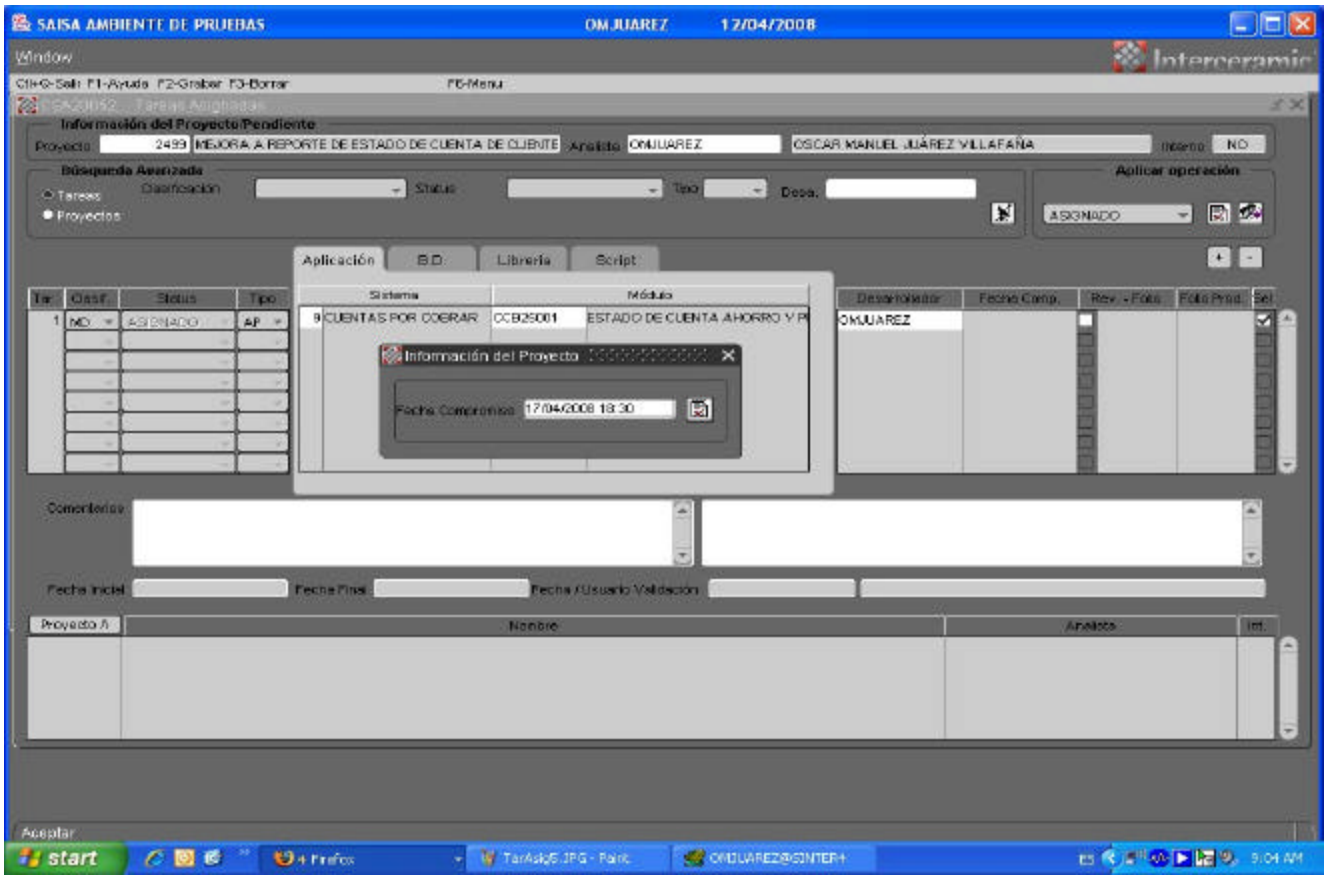

Figura 4.6. Asignación de Tareas a Desarrolladores y Fecha Compromiso del Proyecto.

Una vez que el programador recibe la notificación, pone en proceso la(s) tarea(s) para indicar que comenzará con el desarrollo. En caso de ser una modificación, el usuario con el rol de control de objetos recibe un email con la petición de los mismos para que le sean proporcionados al desarrollador. Es importante mencionar que al marcar la tarea en proceso y posteriormente terminada, se graban las fechas en que se dieron estas acciones. Esto es con la finalidad de almacenar información para poder medir la productividad de los programadores. Ver Fig. 4.7.

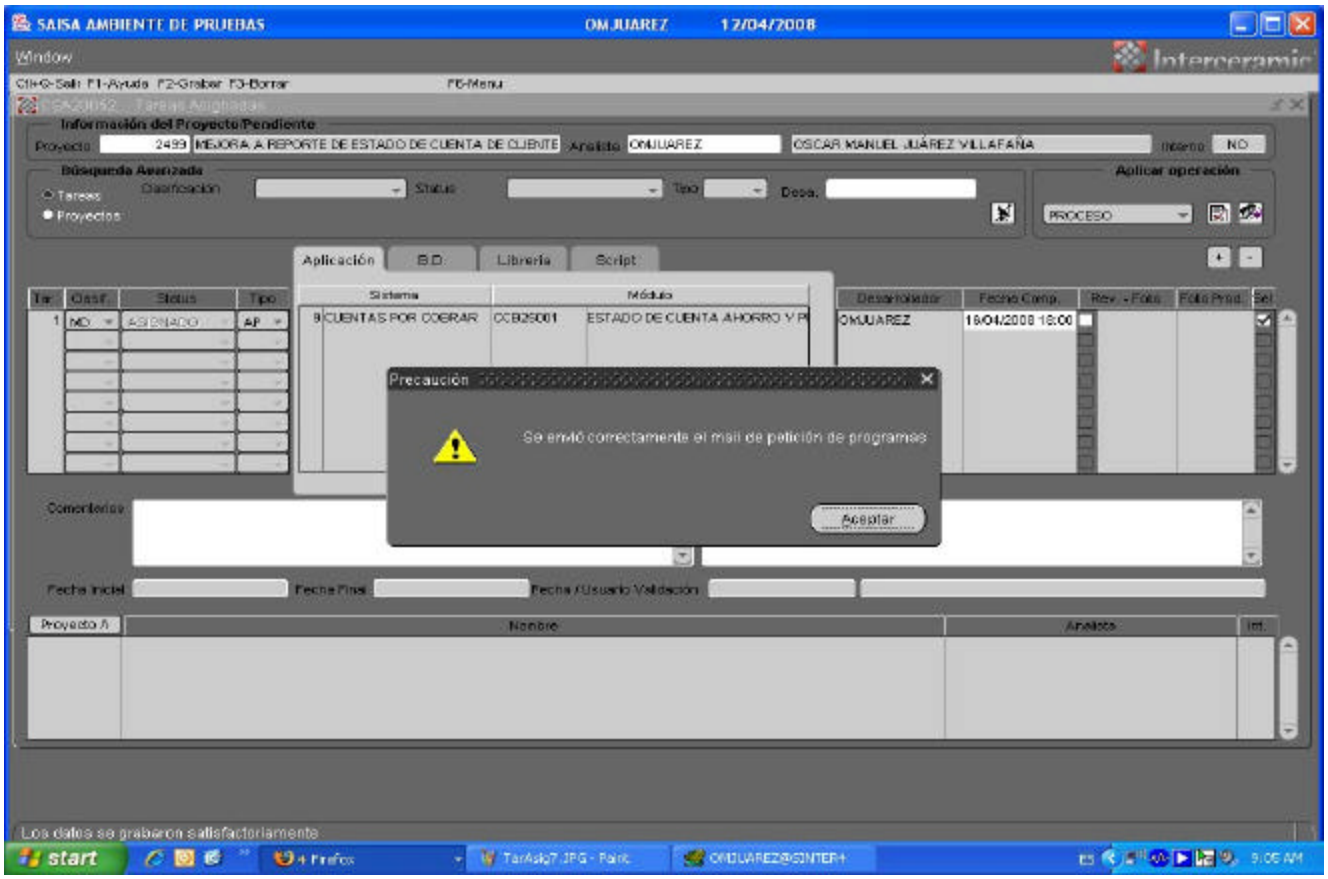

Figura 4.7. Notificación de Petición de Tareas a Control de Fuentes.

Cuando las tareas se han finalizado, se procede a realizar la petición formal de revisión al usuario que solicitó el requerimiento. Esto se realiza mediante el envío de un correo electrónico. El correo electrónico contiene los objetos que se crearon o modificaron durante el desarrollo del proyecto o pendiente. Una vez que el usuario recibe el mail se encargará de verificar que el sistema trabaje conforme a lo solicitado. Ver Fig. 4.8.

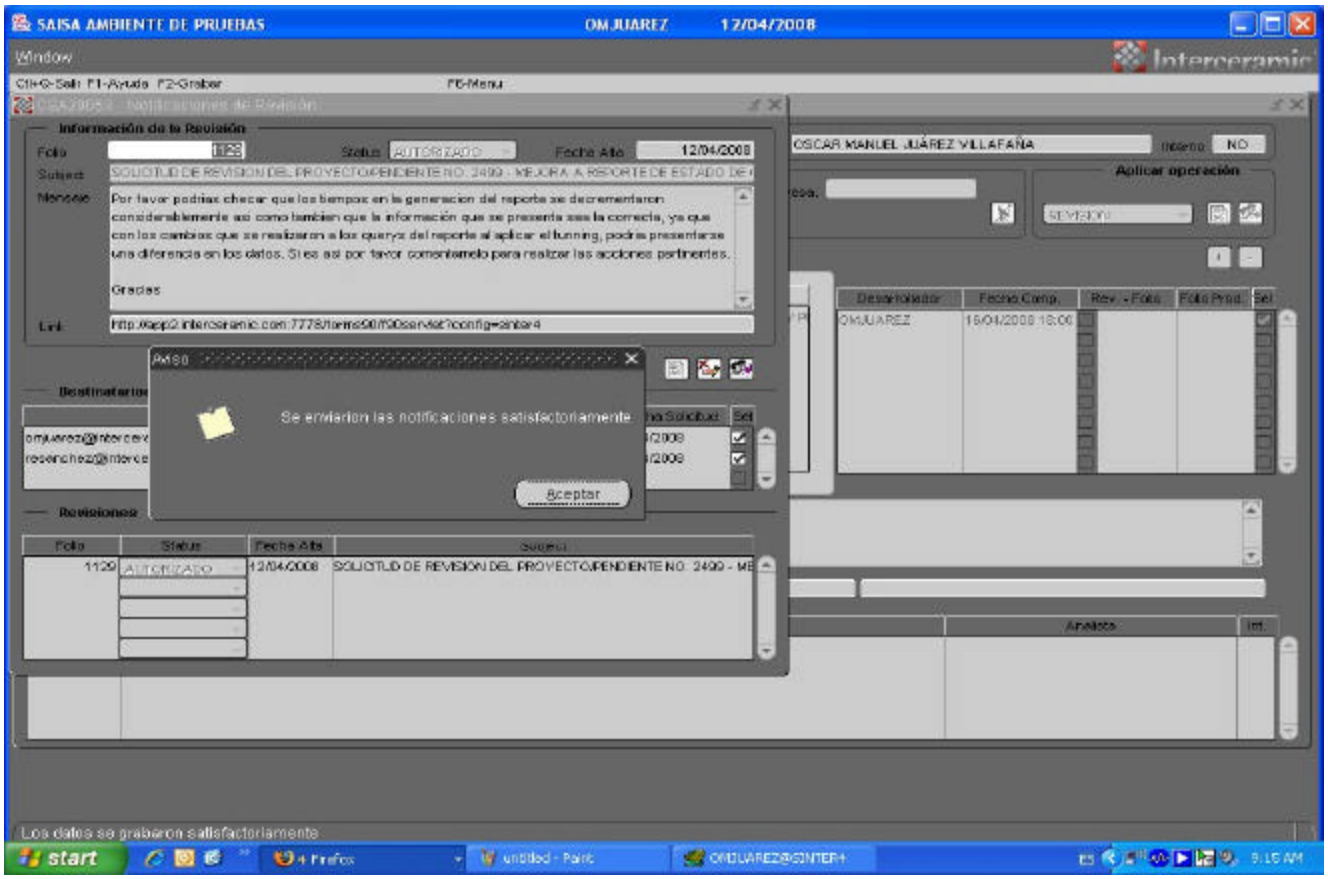

Figura 4.8. Notificación de Revisión de Tareas a Usuario.

Una vez que el usuario revisó el funcionamiento del sistema y esta conforme con lo que solicito, responde el email al analista líder del proyecto para que éste a marque las tareas como validadas. Este es el último paso con lo que respecta a las tareas que se incluyen en un proyecto. Ver Fig. 4.9.

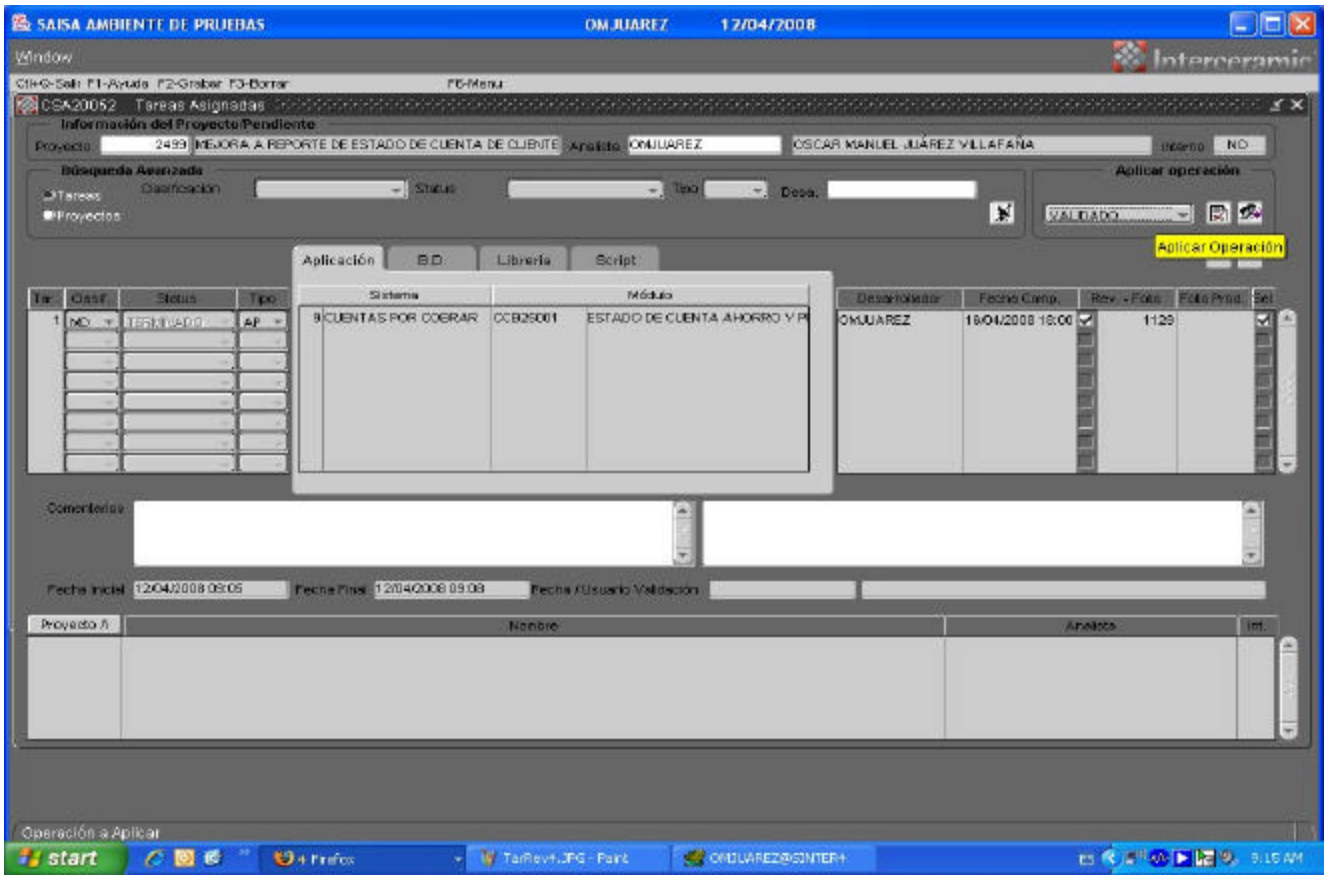

Figura 4.9. Tareas Validadas.

Pase a Producción.

Ya que las tareas están revisadas por el usuario y marcadas como validadas por el analista lo siguiente a realizar es la petición de instalación de los programas modificados/creados en el ambiente de producción. Para esto se crea un folio de Pase a Producción seleccionando el proyecto y las tareas que incluirá dicho pase. Ver. Fig. 4.10.

Los status de un Pase a Producción son los siguientes:

- PE - PENDIENTE. Status inicial del pase.
- PA – POR APROBAR. Este status se emplea cuando un requerimiento es de tipo interno.
- AP - APROBADO. Indica que el pase ha sido aprobado por el usuario que tenga el rol de Control Internos (para el tipo de requerimiento interno) o por el analista una ves que anexo la documentación necesaria que avala su aprobación (para el tipo de requerimiento externo).
- LI - LIBERADO. Se emplea este status cuando el analista le indica al usuario con rol de Control de Fuentes que puede iniciar con el proceso de instalación de este pase a producción.
- TR - TRAMITE. Este status se usa cuando el usuario encargado de instalar los pases a producción inicia con el proceso de instalación de las tareas asociadas al proyecto.
- PD - PRODUCCION. Es cuando el proyecto se instaló correctamente en el sistema.
- CA - CANCELADO. Este status indica que el Pase a Producción no procedió.

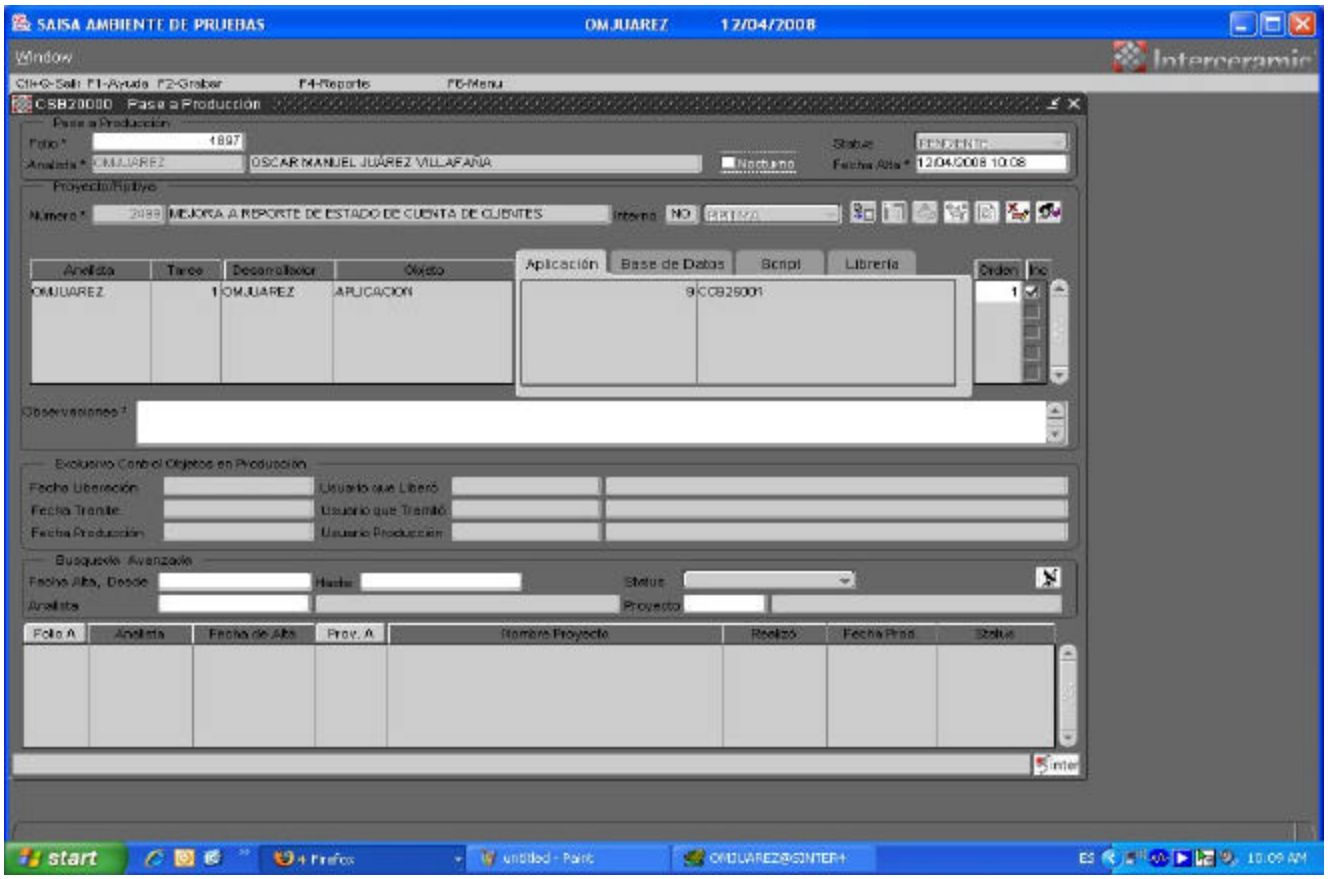

Figura 4.10. Captura de Pases a Producción.

Dependiendo del tipo de proyecto (interno/externo) el seguimiento del pase a producción varía. En el caso de un proyecto externo se tiene que anexar el correo electrónico del usuario (donde acepta que esta conforme con los cambios, mejoras o correcciones según sea el caso). En un proyecto interno en la aprobación para poder instalar un pase proviene del Gerente de Sistemas. Ver Fig. 4.11.

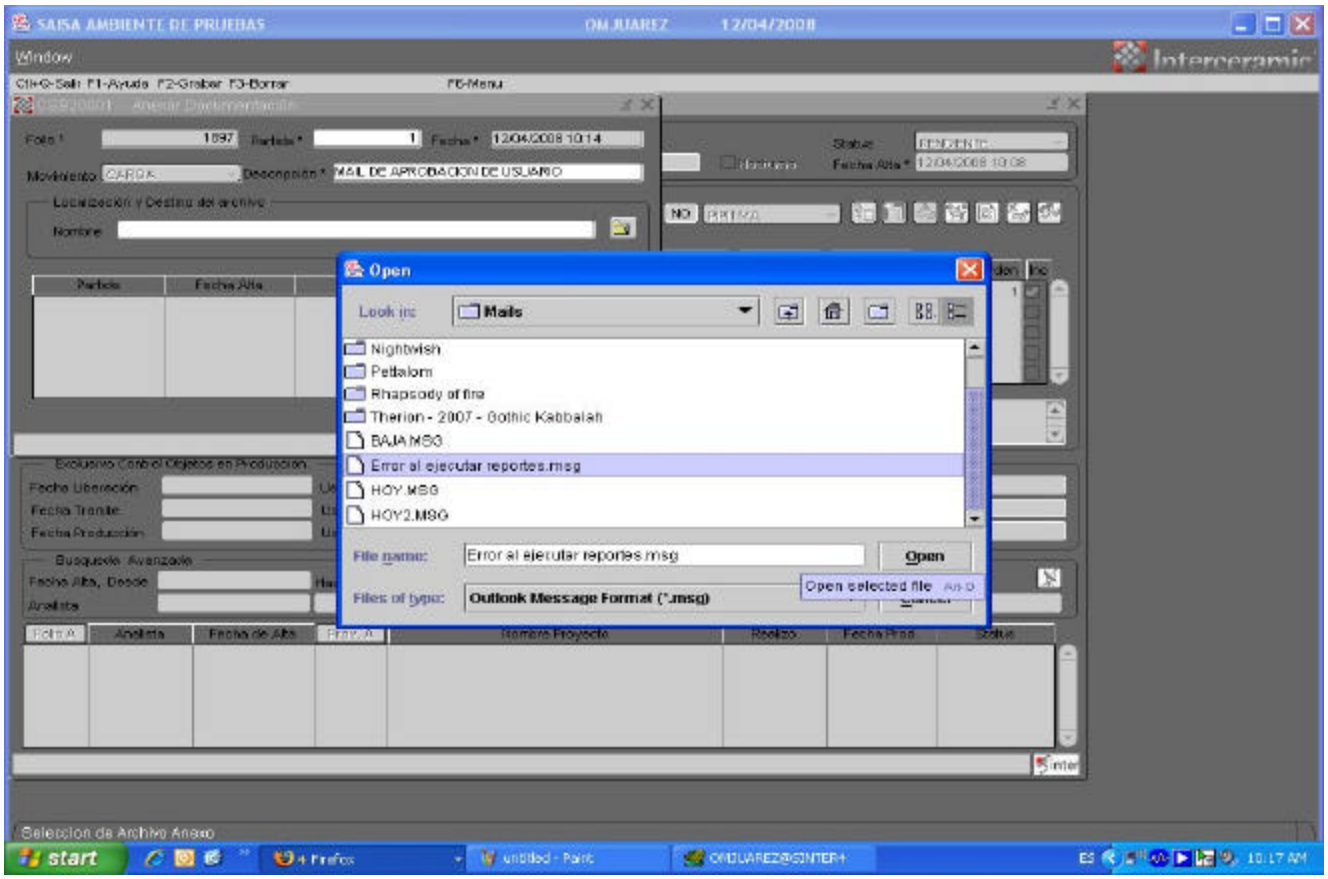

Figura 4.11. Anexar Documentación al Pase a Producción.

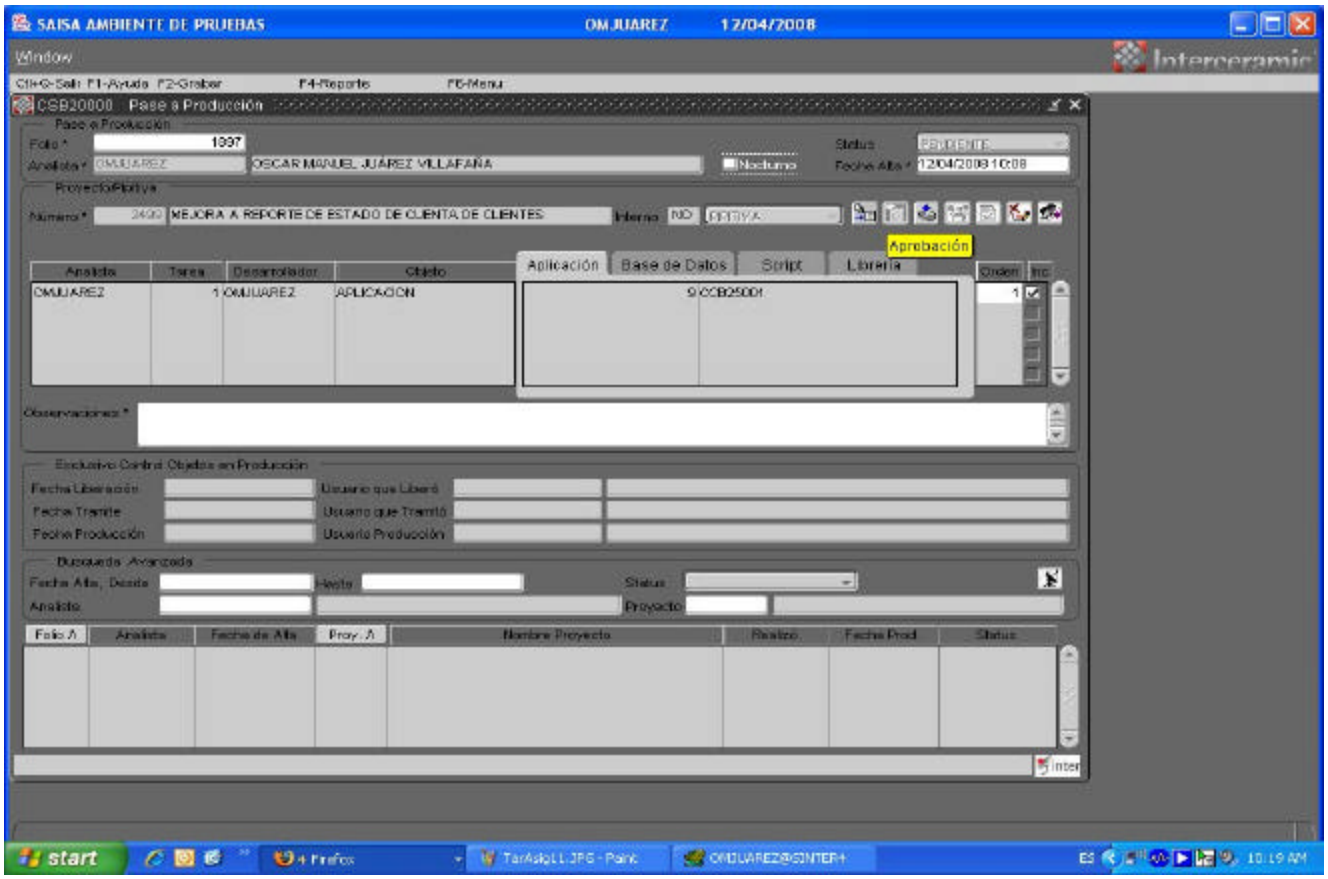

Figura 4.12. Aprobación del Pase a Producción.

Después de que el pase a producción ha sido aprobado por el Gerente de sistemas (proyecto interno) ó que se ha almacenado el mail de aceptación por parte del usuario (proyecto externo). Se notifica al usuario de Control de Fuentes que el pase a producción esta listo para ser procesado. Ver Fig. 4.12 y 4.13.

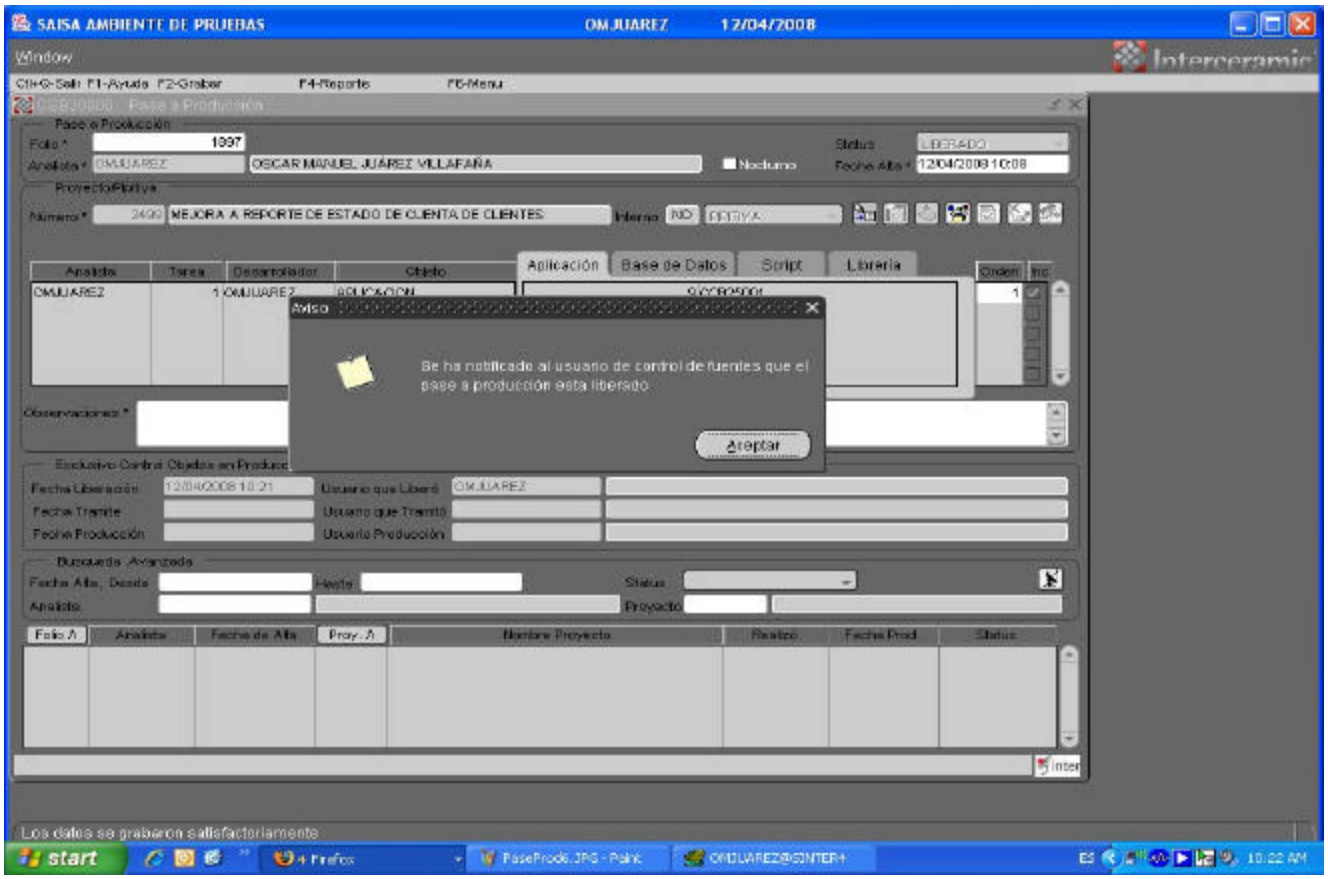

Figura 4.13. Notificación de Liberación de Pase a Producción.

Una ves que el usuario encargado de tramitar los pases a producción recibe la petición de instalación, entra a la Opción “Pendientes de Pase a Producción” le da trámite. Los pases a producción pueden ser nocturnos o normales, si son nocturnos solo se podrá realizar el pase por la noche debido a que se modificara la estructura de la base de datos de tal manera que muchos objetos que directa o indirectamente están ligados a los cambios hechos por el proyecto, necesitan ser recompilados. Los pases normales se pueden realizar inmediatamente después de ser liberados por los analistas. Ver Fig. 4.14.

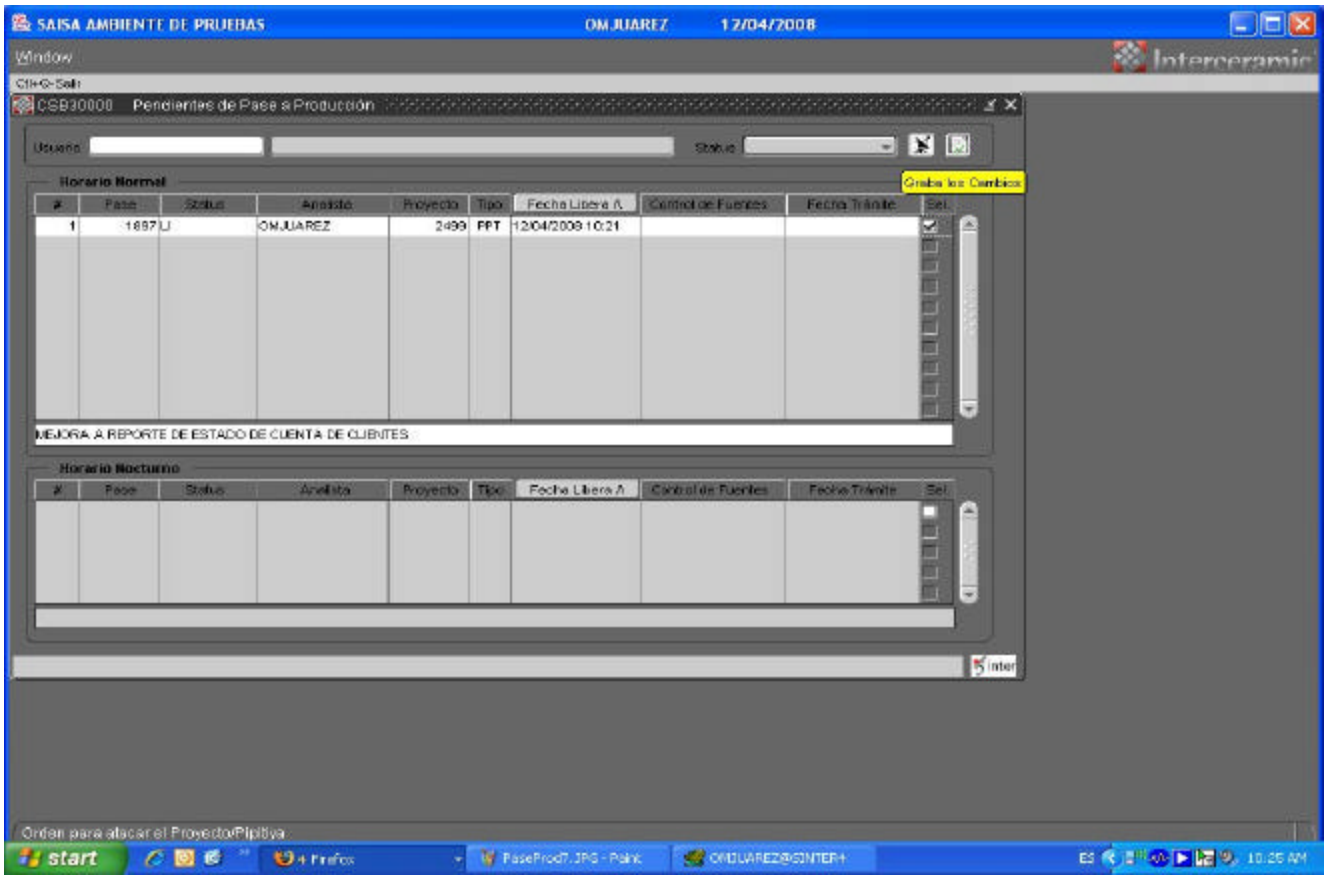

Figura 4.14. Pendientes de Pase a Producción para realizar su trámite.

El usuario encargado de los pases verifica que exista el email de aprobación por parte del usuario (en caso de proyectos externos), y procede a realizar los cambios en el ambiente de producción. Una vez instaladas las tareas se marca el pase con status PRODUCCION y se envía un email al Analista que capturó el pase a producción en el sistema para indicarle que el pase se tramitó correctamente. Ver Fig. 4.15.

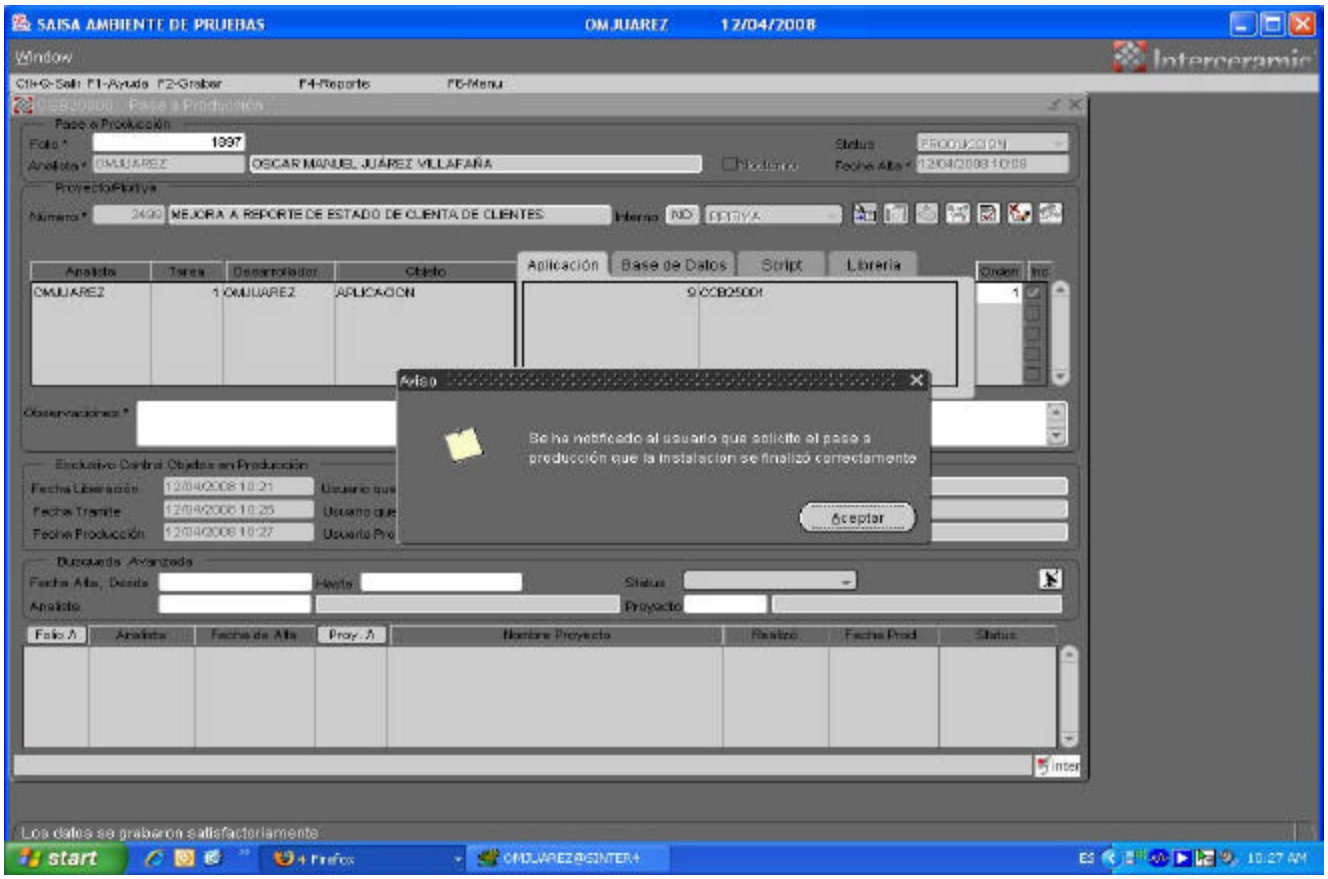

Figura 4.15. Notificación de Instalación de Pase a Producción.

CONCLUSIONES

Uno de los principales objetivos de este tema de tesis fue el poner en práctica los conceptos propuestos dentro del análisis estructurado y del modelado entidad-relación para poder llevar a cabo de forma exitosa el análisis, desarrollo e implementación del sistema CDS.

Gracias a la adecuada recolección de requisitos y a la cooperación del personal de Sistemas de Franquicias de Interceramic, se pudo implementar el sistema CDS dentro del ERP.

La funcionalidad del sistema ha satisfecho las expectativas de los usuarios, la facilidad en la captura y administración de la información aunado a su sistema de alertas para dar seguimiento a los proyectos y tareas, han dado como resultado que el gerente de sistemas y su personal trabajen de forma mas eficiente, desencadenando con ello que sus esfuerzos y tiempo se enfoquen en el desarrollo de nuevos proyectos apoyándose en esta herramienta para automatizar un proceso que anteriormente era engorroso y sobre todo no formal.

El sistema CDS ha tenido un gran éxito dentro del departamento de Sistemas de Franquicias y actualmente se encuentra en un proceso de mejora y adecuación para hacerlo más genérico e independiente. Con esto se pretende alinear y reglamentar las metodologías que se emplean dentro de los diferentes departamentos de sistemas de Interceramic.

De manera personal, el conocimiento adquirido tanto en el análisis como en el desarrollo de la aplicación apoyado en la tecnología Oracle me brindó un crecimiento profesional satisfactorio.

APÉNDICE

[1] Open Database Connectivity (ODBC) es un estándar de acceso a Bases de Datos desarrollado por Microsoft Corporation, el objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos (DBMS por sus siglas en inglés) almacene los datos, ODBC logra esto al insertar una capa intermedia llamada manejador de Bases de Datos, entre la aplicación y el DBMS, el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el DBMS entienda. Para que esto funcione tanto la aplicación como el DBMS deben ser compatibles con ODBC, esto es que la aplicación debe ser capaz de producir comandos ODBC y el DBMS debe ser capaz de responder a ellos. Desde la versión 2.0 el estándar soporta SQL.

Para conectarse a la Base de Datos se crea una DSN (Data Source Name o Nombre de Fuente de Datos) dentro del ODBC que define los parámetros, ruta y características de la conexión según los datos que solicite el fabricante.

[2] Microsoft SQL Server es un sistema de gestión de bases de datos relacionales basado en el lenguaje Transact-SQL, y específicamente en Sybase IQ, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, PostgreSQL o MySQL.

[3] En el contexto de desarrollo de software un fuente u objeto, se refiere a un determinado código de programación que puede ser identificado como un archivo (en el caso de este trabajo, un módulo forma, un módulo reporte, una librería de código, una librería de objetos) ó un script (archivo de texto que contiene instrucciones para la creación de tablas, procedimientos, funciones, paquetes, etc., los cuales residirán en la Base de Datos).

[4] Un módulo forma, es una aplicación visual desarrollada en la herramienta Oracle Forms.

[5] Un módulo reporte es un programa desarrollado en Oracle Reports que permite al usuario obtener información de la base de datos vía diferentes presentaciones. Por ejemplo: Página HTML, PDF, Excel.

[6] Objetos de Base de Datos, puede ser cualquiera de los siguientes:

Un ***procedimiento almacenado*** (stored procedure) es un programa (o procedimiento) el cual es almacenado físicamente en una base de datos. Generalmente son escritos en un lenguaje de bases de datos propietario (en este caso PL/SQL para Oracle). La ventaja de un procedimiento almacenado es que al ser ejecutado, en respuesta a una petición de usuario, es ejecutado directamente en el motor de bases de datos, el cual usualmente corre en un servidor separado. Como tal, posee acceso directo a los datos que necesita manipular y solo requiere enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes. Los procedimientos almacenados en Oracle se clasifican en: Procedimientos y Funciones.

Los ***paquetes de base de datos*** tienen el objetivo de agrupar procedimientos y funciones de forma lógica. De esta manera, se consigue agrupar en un único objeto, toda la casuística asociada a un determinado tipo de tarea. Por ejemplo, si tenemos un conjunto de procedimientos y funciones para realizar cálculos matemáticos complejos, los podemos poner en un paquete.

La ventaja de los paquetes es que la primera vez que se invoca, es cargado en memoria, por lo que las siguientes veces ya no será necesario hacerlo de nuevo. Además, el paquete mantiene una copia en memoria para todos los usuarios. Otras ventajas, es que se pueden encapsular procedimientos y funciones que no forman parte de la interfaz de usuario, o se puede ocultar ciertos objetos y solo hacer públicos los que se necesiten.

Los paquetes constan de definición y cuerpo.

Un ***trigger*** o un disparador en una Base de Datos es un evento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

Una ***vista*** de Base de Datos es un resultado de una consulta SQL de una o varias tablas; también se le puede considerar una tabla virtual. Las vistas tienen la misma estructura que una tabla: filas y columnas. La única diferencia es que sólo se almacena de ellas la definición, no los datos. Los datos que se recuperan mediante una consulta a una vista se presentarán igual que los de una tabla. De hecho, si no se sabe que se está trabajando con una vista, nada hace suponer que es así. Al igual que sucede con una tabla, se pueden insertar, actualizar, borrar y seleccionar datos en una vista. Aunque siempre es posible seleccionar datos de una vista, en algunas condiciones existen restricciones para realizar el resto de las operaciones sobre vistas.

[7] Un ***script*** es un archivo de texto que puede contener instrucciones para la creación de tablas, secuencias, o para modificar información de la Base de Datos.

[8] Una librería, es un agrupador de lógica. Existen 2 tipos:

Librería de código: es un conjunto de procedimientos, y funciones que tienen relación.
Librería de Objetos: es un conjunto de elementos visuales y de código fijo que permiten la reutilización de componentes y código respectivamente.

[9] Ambiente de producción, es donde el sistema esta funcionando para el usuario final.
[10] Ambiente de desarrollo, es donde los programadores, analistas e incluso usuarios finales realizan las pruebas del sistema.

BIBLIOGRAFÍA

[CHEN 1976] Peter Chen "The Entity-Relationship Model: Toward a Unified of Data",
ACM Trans on Database Systems, Vol. 1 No. 1. 1976.

[FARLEY 1988] Richard Farley, "Ingeniería de Software".
Ed. Mc Graw Hill, 1988.

[FROST 1989] R. Frost "Bases de Datos y Sistemas Expertos"
Ed. Diaz Santos, 1989.

[GUILERA 1993] GUILERA, L. "Introducción a la informática" Barcelona: EDUNSA, 1993.

[JACOBSON 1998] Jacobson I. "Applying UML in The Unified Process"
Presentación. Rational Software, 1998.

[KORTH y SILBERSCHATZ 1993] KORTH, H. y SILBERSCHATZ
"Fundamentos de bases de datos"
Madrid: McGraw.Hill, 1993.

[MIGUEL y PIATTINI 1993] MIGUEL, A. DE, y PIATTINI, M.
"Concepción y diseño de bases de datos"
Madrid: RA-MA, 1993.

[ORACLE UNIVERSITY 2002] "Oracle 9i Forms Developer: New Features"
2002
"Oracle 9i: Advanced PL/SQL"
2002

[ORACLE UNIVERSITY 2004] "Oracle Reports Developer 10g Build Reports"