

**Universidad de las Ciencias Informáticas
Facultad 1**



**“Vistas de arquitectura de sistema y datos de las
soluciones Activos Fijos y Facturación del proyecto ERP-Cuba”**

**Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas**

Autor: Enrique Yuniór Almarales Pupo.

Tutores: Ing. Erich Mario Gómez Pérez.

Ing. Pedro Manuel Nogales Cobas.

Ciudad de La Habana, junio de 2010

“Año 52 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al proyecto ERP-Cuba de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de ____ del año 2010.

Firma del Autor

Enrique Yuniór Almarales Pupo.

Firma del Tutor

Ing. Pedro Manuel Nogales Cobas.

DATOS DE CONTACTO

Ing. Erich Mario Gómez Pérez:

Correo electrónico: emgomez@uci.cu

Graduado de Ingeniero en Ciencias Informáticas en la UCI. Actualmente instructor recién graduado. Con un año de experiencia como arquitecto principal de datos del proyecto ERP-Cuba y actualmente figura como jefe de proyecto del proyecto Mantenimiento.

Ing. Pedro Manuel Nogales Cobas:

Correo electrónico: pmnogales@uci.cu

Graduado de Ingeniero en Ciencias Informáticas en la UCI en el año 2008. Actualmente pertenece a la subdirección de tecnología del Centro de Soluciones de Gestión.



AGRADECIMIENTOS

A la generosa Revolución Cubana, a Fidel y a los profesores que han influido en mi formación.

A mis amigos, que me han dado tantos buenos momentos en la universidad.

A todos los que de una u otra forma han contribuido a la realización de este trabajo.

DEDICATORIA

*A mis padres, por su apoyo y comprensión durante todos estos años de estudio y desear
formarme como un hombre de bien.*

A mi hermana, por estar siempre dispuesta a ayudarme y guiarme.

A Adita, por ser esa personita especial que me ha brindado su ayuda incondicional.

A Reynaldito, solo él sabe la valiosa ayuda, gracias por el tiempo dedicado hermano.

A mis Tíos Nelson, Mima y la Niña por acogerme como su hijo en este período.

A mis tutores Erich y Pedro Manuel por su ayuda siempre que necesité.

A Fidel y la Revolución por permitirme formarme en esta prestigiosa escuela.

RESUMEN

En la actualidad es muy frecuente encontrarse con algunos productos informáticos con una escasa documentación del software realizado. En el proyecto ERP-Cuba, específicamente en los subsistemas Activos Fijos y Facturación, no se cuenta con una descripción arquitectónica que presente el nivel que se desea, por lo que surge la necesidad de realizar las vistas de arquitectura de sistema y datos de dichos subsistemas.

El presente trabajo tiene como objetivo construir las vistas de arquitectura de sistema y datos de los subsistemas Activos Fijos y Facturación. Se realiza un estudio del estado del arte sobre el tema enmarcado en el campo de acción, así como de las herramientas a utilizar, y se aborda sobre el expediente de arquitectura del proyecto, además se realizan las vistas de sistema y datos correspondientes a ambos subsistemas. En la vista arquitectónica de sistema se generan varios artefactos, siendo el de mayor importancia el Documento de Línea base, el cual hace referencia a otros como son: Análisis de complejidad y criticidad de los componentes, Especificación de componentes y la Matriz de integración. Por otro lado, en la vista arquitectónica de datos se generan los artefactos: Modelo de datos, Diccionario de datos, Matriz de trazabilidad y Pruebas de concepto. La realización de este trabajo permitirá contar con una mejor vista de arquitectura de sistema y datos de dichos subsistemas.

PALABRAS CLAVE

Subsistemas, arquitectónicas, Activos Fijos, Facturación.

ÍNDICE

INTRODUCCIÓN..... 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 4

 1.1 Arquitectura de software4

 1.2 Estilos arquitectónicos.....5

 1.3 Clasificaciones de vistas.....10

 1.3.1 Conceptos de vistas arquitectónicas13

 1.3.2 Arquitectura de sistema y datos14

 1.3.3 Beneficios de la Arquitectura de software ¡Error! Marcador no definido.

 1.4 Soluciones arquitectónicas de diversos ERP.....15

 1.5 Arquitectura del proyecto ERP-Cuba16

 1.6 Expediente tecnológico del proyecto ERP-Cuba18

 1.7 Artefactos generados en el proyecto ERP-Cuba.....21

 1.8 Herramientas a utilizar23

 1.9 Conclusiones parciales.....24

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA 25

 2.1 Mapa de procesos y modelo conceptual25

 2.2 Descripción de la estructura de empaquetamiento de los componentes.....31

 2.3 Vista arquitectónica de los subsistemas Activos Fijos y Facturación.....32

 2.3.1 Complejidad y criticidad32

 2.3.2 Priorización de los componentes34

 2.3.3 Matriz de integración35

 2.3.4 Documento Línea Base37

 2.4 Conclusiones parciales39

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS 40

 3.1 Modelo de datos.....40

 3.2 Diccionario de datos42

 3.3 Matriz de trazabilidad de datos.....55

3.4 Pruebas de concepto	57
3.5 Conclusiones parciales.....	61
CONCLUSIONES GENERALES	62
RECOMENDACIONES	63
REFERENCIAS BIBLIOGRÁFICAS	64
BIBLIOGRAFÍA.....	65
GLOSARIO DE TÉRMINOS	66
ANEXOS.....	67

ÍNDICE DE FIGURAS

Fig. 1-Diagrama simple del enfoque 4+1(Kruchten, 1995)-----	12
Fig. 2-Expediente de la Arquitectura de sistema -----	19
Fig. 3-Expediente de la Arquitectura de datos -----	19
Fig. 4-Modelo de datos -----	20
Fig. 5-Mapa de Procesos de Negocio de AFT-----	27
Fig. 6-Mapa de Procesos de Negocio de Facturación -----	29
Fig. 7-Modelo Conceptual de AFT-----	30
Fig. 8-Modelo Conceptual de Facturación -----	31
Fig. 9-Estructura de un componente -----	32
Fig. 10- Representación del artefacto Análisis de Complejidad y Criticidad del subsistema Facturación -----	33
Fig. 11- Representación de la Matriz de integración AFT -----	36
Fig. 12- Representación de la matriz de integración del subsistema Facturación -----	37
Fig. 13-Mapa de componentes AFT -----	38
Fig. 14-Mapa de Componente de Facturación-----	39
Fig. 15-Representación del modelo de datos del subsistema Activos Fijos-----	40

Fig. 16- Representación del modelo de datos del subsistema Facturación.....	41
Fig. 17- Ejemplo de concurrencia	42
Fig. 19- Matriz de integración del subsistema	57

ÍNDICE DE TABLAS

Tabla 1-Relación artefacto-descripción-roles. (Leyet, 2010).....	22
Tabla 2--Priorización de los componentes	34
Tabla 3-Diccionario de datos del subsistema Activos Fijos	42
Tabla 4--Diccionario de datos del subsistema Activos Fijos	50
Tabla 5- Matriz de integración del subsistema Activos Fijos.....	56

INTRODUCCIÓN

El desarrollo de sistemas automatizados de gestión empresarial ha tomado un gran auge a nivel mundial, sobre todo a partir de la generalización de las nuevas tecnologías de la información y las comunicaciones.

La Industria de Software se ha convertido en una de las más poderosas y crecientes del momento. Cada día son más las empresas que se suman motivadas por la competencia y el desarrollo de productos de alta calidad. No obstante, se han visto en la necesidad de centralizar y controlar su información para garantizar que la misma sea íntegra y confiable para asegurar el cumplimiento de objetivos estratégicos y una mejor toma de decisiones.

Una de las alternativas para garantizar lo antes mencionado son los Sistemas de Planificación de Recursos Empresariales (sus siglas en Inglés ERP); Los sistemas ERP son sistemas integrales, adaptables y modulares que se caracterizan por estar compuestos por diferentes partes integradas en una única aplicación.

Cuba, a pesar de sus limitaciones económicas, no queda exenta de este progreso tecnológico, En la actualidad constituye un factor estratégico el desarrollo de un sistema ERP, el cual parte de la idea de contar con una solución a nivel nacional y basada en los principios de independencia tecnológica. En coordinación con el Ministerio de Finanzas y Precios, la Universidad de las Ciencias Informáticas (UCI), se encuentra desarrollando en conjunto con especialistas de diversos organismos, estudiantes y trabajadores, lo que hoy se denomina proyecto ERP-Cuba.

Dentro del proyecto ERP-Cuba se desarrollan varios subsistemas, dos de ellos son Activos Fijos y Facturación. En estos subsistemas la documentación de la arquitectura de sistema y datos es escasa y poco entendible, se desconoce la complejidad y criticidad de los componentes, así como la especificación de los servicios que brinda y la integración entre estos mediante servicios. En el Modelo de datos no se puede determinar con claridad el significado de una tabla o de un campo, esto imposibilita acciones como:

- La comprensión del diseño arquitectónico por los involucrados y el equipo de desarrollo nuevo.
- El soporte y desarrollo de nuevas versiones del sistema.
- La reutilización de la solución implementada en otros componentes.

El presente trabajo de diploma propone darle solución a los problemas expresados en la situación descrita con anterioridad, mediante el cual se delimitó el siguiente **problema a resolver**: El estado actual de la solución arquitectónica de los subsistemas de Activos Fijos y Facturación afecta la capacidad de entendimiento de los involucrados y por consiguiente el soporte y desarrollo de nuevas versiones del sistema.

Se define como **objeto de estudio**: La Arquitectura de sistema y datos, para el cual se delimita el siguiente **campo de acción**: Arquitectura de sistema y datos de la solución Activos Fijos y Facturación.

Para dar respuesta al problema planteado se trazó el siguiente **objetivo general**: Realizar las vistas de Arquitectura de sistema y datos de las soluciones Activos Fijos y Facturación, para facilitar su entendimiento por los involucrados y apoyar el soporte y desarrollo de nuevas versiones del sistema.

Objetivos específicos:

1. Realizar marco teórico de la investigación.
2. Realizar la vista de arquitectura de sistema de las soluciones Activos Fijos y Facturación.
3. Realizar la vista de arquitectura de datos de las soluciones Activos Fijos y Facturación.

Este trabajo defiende la siguiente idea: Si se realizan las vistas de Arquitectura de sistema y datos de las soluciones Activos Fijos y Facturación, facilitará su entendimiento por los involucrados y apoyará el soporte y desarrollo de nuevas versiones del sistema.

Para cumplir dichos objetivos se trazaron las siguientes **Tareas Investigativas**:

1. Realizar un estudio del estado del arte sobre arquitectura de software, y algunas soluciones financieras para poder definir la situación actual.
2. Estudiar los estándares de documentación utilizados por el proyecto.
3. Estudiar los artefactos arquitectónicos de la vista de arquitectura.
4. Analizar con profundidad la arquitectura de los subsistemas para lograr entender y poder documentar, así como estudiar los procesos de negocio y requisitos.
5. Describir la vista de Arquitectura de sistema y datos.
6. Realización del Documento complejidad y criticidad.
7. Realización del Documento especificación de componentes.
8. Realización de la Matriz de integración.
9. Realización del Documento Línea Base.

10. Descripción del Modelo de datos.
11. Realización del Diccionario de datos.
12. Realización de la Matriz de trazabilidad.
13. Realización de las Pruebas de concepto.

Este trabajo de diploma se estructura en tres capítulos:

- En el Capítulo 1: **Fundamentación Teórica**, se muestran varios conceptos de arquitectura de software, específicamente de Arquitectura de sistema y datos y otras definiciones estrechamente relacionadas con la misma. Además, se aborda sobre soluciones arquitectónicas de diversos ERP existentes y el expediente del proyecto ERP-Cuba.
- En el Capítulo 2: **Vista de arquitectura de sistema**, se realiza un previo análisis de los artefactos de entrada del negocio y se muestran los artefactos generados en la realización de la vista de Arquitectura de sistema de los subsistemas Activos Fijos y Facturación.
- En el Capítulo 3: **Vista de arquitectura de datos**, se muestran los artefactos generados en la realización de la vista de arquitectura de datos, así como los resultados de las Pruebas de concepto de diseño realizadas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se tratan los conceptos relacionados con la arquitectura de software, arquitectura de sistema y datos, se aborda sobre las diferentes soluciones arquitectónicas de diversos ERP, además del expediente de arquitectura del proyecto ERP-Cuba.

1.1 Arquitectura de software

En la actualidad se pueden encontrar gran cantidad de conceptos sobre arquitectura de software, es válido decir que no todos los arquitectos tienen la misma concepción sobre este tema por lo que cada uno asume su propia interpretación, en este epígrafe se mencionan algunos de estos conceptos:

Una definición reconocida es la de Clements: La arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión del detalle inherente a la mayor parte de las abstracciones. (Reynoso, 2004)

A pesar de la abundancia de definiciones sobre el campo de la arquitectura de software, existe un acuerdo en general de que ella se refiere a la estructura a grandes rasgos del sistema, estructura consistente en componentes y relaciones entre ellos. Estas cuestiones estructurales se vinculan con el diseño, pues la arquitectura de software es ante todo una forma de diseño de software que se manifiesta tempranamente en el proceso de creación de un sistema; pero este diseño ocurre a un nivel más abstracto que el de los algoritmos y las estructuras de datos. En el que muchos consideran un ensayo seminal de la disciplina, Mary Shaw y David Garlan sugieren que dichas cuestiones estructurales incluyan organización a grandes rasgos y estructura global de control; protocolos para la comunicación, la sincronización y el acceso a datos; la asignación de funcionalidad a elementos del diseño; la distribución física; la composición de los elementos de diseño; escalabilidad y rendimiento; y selección entre alternativas de diseño. (Reynoso, 2004)

En una definición tal vez demasiado amplia, David Garlan establece que la arquitectura de software constituye un puente entre el requerimiento y el código, ocupando el lugar que en los gráficos antiguos se reservaba para el diseño. La definición "oficial" de arquitectura de software se ha acordado que sea la que brinda el documento de IEEE Std 1471-2000, adoptada también por Microsoft, que expresa lo siguiente: La arquitectura de software es la organización fundamental de un sistema encarnada en sus

componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. (Reynoso, 2004)

Teniendo en cuenta las definiciones antes plasmadas, podemos afirmar que la arquitectura de software más óptima es aquella que esté orientada a un mejor diseño, así como la evolución del sistema en desarrollo.

1.2 Estilos arquitectónicos

Dewayne Perry y Alexander Wolf¹ establecen el razonamiento de estilos de arquitectura como uno de los aspectos fundamentales de la arquitectura de software. Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de estilos cataloga las formas básicas viables de estructuras de software, mientras que las formas complejas se articulan mediante la composición de los estilos fundamentales. Cada estilo arquitectónico conjuga los componentes que realizan una función requerida por el sistema; los conectores que permiten la comunicación y cooperación entre dichos componentes; las restricciones que definen como estos pueden integrarse en el sistema; y modelos semánticos que permiten al diseñador entender las propiedades globales de un sistema para analizar las propiedades conocidas de sus partes constituyentes. Un software puede tener presente varios estilos arquitectónicos. Algunos de los más usados son los mostrados a continuación:

Arquitecturas basadas en componentes

Actualmente en el desarrollo de software hay una gran necesidad de hacer uso de la reutilización de partes o módulos de software existentes, que podrían ser utilizados para la generación de nuevas extensiones de las aplicaciones o las aplicaciones completas.

Características:

La arquitectura basada en componentes permite la reutilización de los mismos. Para ello los componentes deben satisfacer como mínimo el siguiente conjunto de características:

¹ La arquitectura de software comenzó su expansión explosiva con los manifiestos de Dewayne Perry, de AT&T Bell Laboratorios de New Jersey y Alexander Wolf, de la Universidad de Colorado. Puede decirse que ambos fundaron la disciplina. Autores del libro "Foundations for the study of software architecture".

- **Identificable:** Un componente debe tener una identificación clara y consistente que facilite su catalogación y búsqueda en repositorios de componentes.
- **Accesible sólo a través de su interfaz:** El componente debe exponer al público únicamente el conjunto de operaciones que lo caracteriza (interfaz) y ocultar sus detalles de implementación. Esta característica permite que un componente sea reemplazado por otro que implemente la misma interfaz.
- **Servicios invariantes:** Las operaciones que ofrece un componente, a través de su interfaz, no deben variar. La implementación de estos servicios puede ser modificada, pero no deben afectar la interfaz.
- **Documentado:** Un componente debe tener una documentación adecuada que facilite su búsqueda en repositorios de componentes, evaluación, adaptación a nuevos entornos, integración con otros componentes y acceso a información de soporte.

Ventajas:

- **Reutilización del software:** Lleva a alcanzar un mayor nivel de reutilización de software.
- **Simplifica las pruebas:** Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
- **Simplifica el mantenimiento del sistema:** Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- **Mayor calidad:** Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

Desventajas:

- Si no existen los componentes, hay que desarrollarlos y se puede perder mucho tiempo.
- Las actualizaciones de los componentes adquiridos no están en manos de los desarrolladores del sistema.

Arquitectura orientada a servicios web

Una arquitectura orientada a servicios web es un conjunto de servicios interconectados cuyo objetivo es automatizar uno o varios procesos del negocio. Los servicios web son aplicaciones que utilizan estándares para el transporte, codificación y protocolo de intercambio de información. Los servicios web permiten la intercomunicación entre sistemas de cualquier plataforma y se utilizan en una gran variedad de escenarios de integración, tanto dentro de las organizaciones como con los socios del negocio. Dentro de los principales beneficios que trae consigo el uso de los servicios web, se pueden mencionar que:

- **Promueven la interoperabilidad:** La interacción entre un proveedor y un solicitante de servicio está diseñada para que sea completamente independiente de la plataforma y el lenguaje. Esta interacción requiere un documento WSDL² para definir la interfaz y describir el servicio, junto con un protocolo de red, generalmente HTTP. (Lebrún, 2007)
- **Permiten la integración justo a tiempo:** El proceso de descubrimiento se ejecuta dinámicamente, a medida que los solicitantes de servicio utilizan a los agentes para encontrar proveedores de servicio. Una vez el solicitante y el proveedor de servicio se han ubicado, se utiliza el documento WSDL del proveedor para enlazar al solicitante con el servicio. Esto significa que los solicitantes, los proveedores y los agentes actúan en conjunto para crear sistemas que son auto-configurables, adaptativos y robustos. (Lebrún, 2007)
- **Reducen la complejidad por medio del encapsulamiento:** Los solicitantes y los proveedores del servicio se preocupan por las interfaces necesarias para interactuar. Como resultado, un solicitante de servicio no sabe cómo fue implementado el servicio por parte del proveedor, y éste a su vez, no sabe cómo utiliza el cliente el servicio. Estos detalles se encapsulan en los solicitantes y proveedores. El encapsulamiento es crucial para reducir la complejidad. (Lebrún, 2007)
- Disminuyen el tiempo de desarrollo de las aplicaciones.
- Minimizan el costo de implementación.
- Aumentan la productividad debido al incremento de fluidez en las relaciones entre proveedores, socios, empleados y clientes.
- Proporcionan una mejor gestión, mantenimiento y actualización de la información.
- Evitan problemas con los cortafuegos.

² Son las siglas de *Web Services Description Language*, un formato XML que se utiliza para describir servicios Web.

Arquitectura Modelo Vista Controlador (MVC³)

El estilo conocido como Modelo Vista Controlador separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres clases distintas: el modelo, que administra el comportamiento y los datos del dominio de aplicación; la vista, que maneja la visualización de la información representada por el modelo; y el controlador, que responde a eventos, generalmente acciones del usuario, e invoca cambios en la vista y en el modelo, según resulte necesario.

Tanto la vista como el controlador dependen del modelo, mientras que este no depende de las otras clases. Dicha separación permite realizar y probar el modelo, independientemente de la representación visual. Todo esto también propicia que la interfaz de usuario pueda mostrar, simultáneamente, múltiples vistas de los mismos datos; al igual que la adaptación al cambio, puesto que los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas del negocio, agregar nuevas opciones de presentación que no afecten al modelo generalmente. La separación entre la vista y el controlador puede ser secundaria en aplicaciones de clientes ricos, pero en aplicaciones web, la separación entre la vista (el browser) y el controlador (los componentes del lado del servidor que manejan los requerimientos de HTTP) está mucho más claramente definida.

A pesar de ofrecer un soporte de múltiples vistas y de ser adaptable al cambio, esta arquitectura también presenta sus desventajas. La misma introduce nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad de la solución, y también está el costo de actualizaciones frecuentes, debido a que desacoplar el modelo de la vista no indica que los desarrolladores del modelo puedan ignorar la naturaleza de las vistas.

Arquitectura en capas

La arquitectura en capas constituye uno de los estilos que aparecen con mayor frecuencia. Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Este estilo instrumenta una vieja idea de organización estratigráfica que se remonta a las concepciones formuladas por Edsger Dijkstra en la década de 1960, de que los grandes sistemas podrían organizarse por niveles de complejidad asemejándose a las capas de una cebolla.

³ Sigla que denomina el estilo o patrón Modelo Vista Controlador; denominación en inglés Model View Controller.

Para algunos autores más estrictos para con el estilo, las capas internas están ocultas a todas las demás. En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas que encapsulan tanto recursos como servicios, facilitando el aumentando de los niveles de abstracción y proporcionando la comprensión global del sistema. Entre las principales ventajas que se señalan del estilo (Reynoso, 2004) destacan:

- **Alto nivel de abstracción:** Estilo que soporta un diseño basado en niveles de complejidad, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- **Optimización y refinamiento:** En principio, el estilo admite la sustitución directa de una capa intermedia sin afectar el funcionamiento del sistema, siempre y cuando se respeten las dependencias entre niveles. Los cambios en una capa dan un menor impacto global al sistema.

Entre las desventajas sin embargo se manifiesta que algunos problemas no admiten un mapeo a una estructura jerárquica, aun cuando resulte lógico consideraciones asociadas al rendimiento, que las capas superiores requieran acceder directamente a servicios de las inferiores.

También se argumenta que si bien facilita el control y encapsulamiento de aplicaciones complejas, resulta excesivamente pesado para aplicaciones simples. Respecto a las desventajas de dicho estilo el autor considera que no resulta relevante puesto que el sistema que se concibe es una clásica aplicación empresarial en la cual la arquitectura multicapa ha resultado más que probada con excelentes resultados en tanto las concesiones de rendimiento son superadas por lo ganado en organización y abstracción. (Reynoso, 2004)

Arquitectura cliente/servidor

Con la proliferación de ordenadores personales de bajo coste en el mercado, los recursos de sistemas de información existentes en cualquier organización se pueden distribuir entre ordenadores de diferentes tipos: ordenadores personales de gama baja, media y alta, estaciones de trabajo, miniordenadores o incluso grandes ordenadores.

El concepto de cliente/servidor proporciona una forma eficiente de utilizar todos estos recursos de máquina de tal forma que la seguridad y fiabilidad que proporcionan los entornos *mainframe* se

traspasan a la red de área local. A esto hay que añadir la ventaja de la potencia y simplicidad de los ordenadores personales.

La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes.

Al hacer un análisis de los estilos arquitectónicos antes mencionadas, queda demostrado que cada uno de ellos tiene sus beneficios y debilidades, seleccionar uno u otro depende de las necesidades y servicios del sistema en cuestión.

Luego del estudio antes realizado, para el diseño de la arquitectura del sistema ERP-Cuba, se adoptaron los estilos de arquitecturas basadas en componentes y el MVC. El uso de los mismos fue una propuesta de la subdirección de arquitectura del Centro de Soluciones de Gestión, se considera que la decisión fue acertada, teniendo en cuenta el modelo de desarrollo adoptado para el proyecto y los beneficios que aportan estos estilos. Ambos pertenecen al grupo de Estilos de Llamada y Retorno, en el cual se enfatiza la modificabilidad y la escalabilidad. Además son los más generalizados en sistemas en gran escala. Particularmente el estilo MVC fue re-implementado, por el equipo central de arquitectura, de forma tal que las clases controladoras de los diferentes componentes pueden obtener los datos mediante instanciaciones a objetos de clases del modelo del negocio, o tomarlos directamente del modelo del dominio.

1.3 Clasificaciones de vistas

La arquitectura está representada por un número de vistas arquitectónicas. Estas vistas capturan las mayores decisiones de diseño estructural. En esencia, las vistas arquitectónicas son abstracciones o simplificaciones del diseño entero, en las cuales las características importantes se hacen más visibles dejando los detalles a un lado.

La vista arquitectónica, aporta el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones. (Bass, 1998)

❖ David Parnas (´74) las estructura en tres grupos:

- **Estructura de módulos:** Es parte de o comparte el mismo secreto que la asignación de trabajo.
- **Estructura de uso:** Depende de la corrección de programas.
- **Estructura de procesos:** Brinda trabajo computacional a procesos.
- ❖ Perry y Wolf ('94)
 - Reconocen la necesidad de vistas que enfatizan ciertos aspectos arquitectónicos útiles para distintas audiencias o para diferentes propósitos.
- ❖ Siemens Corporate Research ('95)
 - Vista conceptual: Principales elementos de diseño y su interrelación.
 - Vista de módulos: Estructura funcional y de capas.
 - Vista de ejecución: Estructura dinámica.
 - Vista de código: Organización de código fuente, binarios y bibliotecas en el ambiente de desarrollo.
- ❖ SEI (§02): Racionalización de Vistas
 - Categorizan las vistas en "*ViewTypes*".
 - Viewtypes: definen los tipos de elementos y los tipos de relaciones usados para una descripción desde una perspectiva particular.
 - ViewType Modular: ¿Cómo está el sistema estructurado como conjunto de unidades de implementación?
 - ViewType Componente y Conector: Como un conjunto de elementos que tienen comportamiento e interacción en tiempo de ejecución.
 - ViewType Asignación: ¿Cómo se relaciona el software con elementos que no son software?
- ❖ Propuestas de Arquitectura 4+1 Vistas de Philippe Kruchten (RUP):
 - Vista lógica: Describe el modelo de objetos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

- Vista de procesos: Muestra la concurrencia y sincronía de los procesos.
- Vista física: Muestra la ubicación del software en el hardware.
- Vista de desarrollo: Describe la organización del entorno de desarrollo.

Un enfoque en la presentación de un sistema en UML es conocida como 4+1 vistas. Esta forma de documentar estos modelos divide lo que sabe de él en cinco áreas:

- ❖ Vista de Casos de Uso: Contiene requisitos desarrollados en las restantes vistas.
- ❖ Vista Lógica: Muestra la estructura estática del sistema.
- ❖ Vista Física: Muestra el despliegue de la aplicación en la red de computadoras.
- ❖ Vista de Procesos: Muestra los hilos y procesos de ejecución así como la comunicación entre estos.
- ❖ Vista de Desarrollo: Muestra la estructura en modelos del código del sistema. (Vistas, 2007)

Estas vistas se presentan tradicionalmente en una figura de cuatro cajas con un óvalo central que representa al modelo de casos de uso. La siguiente figura (ver Fig.1) corresponde a esta imagen de la que se habla:



Fig. 1-Diagrama simple del enfoque 4+1(Kruchten, 1995)

De acuerdo con el Sr. Kruchten, las distintas vistas del enfoque responden a las necesidades de las distintas partes interesadas: clientes, programadores, administradores, etc. Para cada uno de estos, se presenta una visión resumida del sistema con la información que requieren para satisfacer sus necesidades.

Es así que la vista de desarrollo le indica al programador como iniciar y organizar su código; la vista física ayuda a los administradores de sistemas a decidir la infraestructura que se ha de dedicar al sistema; la vista de procesos es útil para realizar análisis de integridad y tomar decisiones de integración con otros sistemas; finalmente, siempre de acuerdo con el Sr. Kruchten, la vista lógica le sirve a los usuarios y clientes a visualizar la funcionalidad que el sistema les provee.

Este enfoque es uno de los más extendidos en la literatura, sin embargo, su aplicación es de alcance limitado en los tiempos modernos, donde las aplicaciones tradicionales han dejado su lugar a sistemas basados en web. Es entonces un enfoque digno de estudio aunque es probable que en el proyecto se sigan otras aproximaciones para la organización y presentación de los modelos. (Kruchten, 1995)

1.3.1 Conceptos de vistas arquitectónicas

Luego de haber analizado las anteriores definiciones de los diferentes autores y organizaciones acerca de las vistas se centrará la investigación a las concernientes al presente trabajo de diploma Vista de sistema y Vista de datos.

Vista de sistema

La arquitectura de sistema es una de las más complejas dentro de la arquitectura de software. Es la encargada de definir las principales interacciones, los conectores, y las configuraciones que serán asumidos por los componentes computacionales en función de los elementos del negocio que los mismos abstraen. La vista de sistema constituye una proyección simétrica de alto nivel, de los procesos de negocio o arquitectura de negocio que se trabaja, expresada en elementos, conectores, restricciones y configuraciones. (Mena, 2009). En el próximo epígrafe será abordado el concepto de arquitectura de sistema de forma más profunda.

Vista de datos

La vista de datos describe y representa la arquitectura de datos la cual es la responsable de diseñar las soluciones de datos en cada uno de los subsistemas, diseña las soluciones de integración en los mismos, construye la vista de datos general y la vista de datos particular para cada subsistema. Dicha vista tiene como objetivos fundamentales definir las políticas de integración y los objetivos de trabajo en función del Centro de datos, definir la metodología de trabajo de la arquitectura de datos, definir metodología de implementación de réplica, definir, construir y actualizar permanentemente el

expediente de la vista de datos, definir método de actualización y control de versiones de la base de datos y definir indicadores y métricas para el manejo de los datos. (Mena, 2009).

A continuación se muestran las definiciones de arquitectura de sistema y datos:

1.3.2 Arquitectura de sistema y datos

Arquitectura de sistema

La arquitectura de sistema al igual que la arquitectura de software no posee un único concepto, existen diferentes definiciones de esta. A continuación se brindan algunos elementos importantes para definir una arquitectura de sistema.

La arquitectura de sistema no es más que el diseño conceptual que define la estructura y/o el comportamiento de un sistema. Define la organización principal de un sistema basada en sus componentes y sus relaciones.

Se considera además que la arquitectura de sistema se puede subdividir en tres etapas:

- Elección del estilo arquitectónico.
- Selección de los patrones de diseño.
- Diseño de componentes.

Arquitectura de datos

La arquitectura de datos identifica y define las mejores clases de datos que apoyan las funciones del negocio, definidas en el modelo de negocios. Es una de las primeras arquitecturas a ser definidas porque la calidad de los datos es el producto básico.

La arquitectura de datos consta de entidades de datos, cada una de las cuales tiene atributos y relaciones con otras entidades de datos. Establece las directrices comunes para las operaciones de datos que permitan predecir y controlar el flujo de datos en el sistema. Describe los grupos de datos y sus elementos así como asigna los artefactos de datos. Se divide en tres niveles generales: interno, conceptual y externo.

Nivel Interno: es el más cercano al almacenamiento físico, es decir, le concierne la manera de como los datos se almacenan en realidad. La arquitectura de datos proporciona criterios para los tratamientos de los datos que hacen posible el diseño de flujo de datos y además permite controlar el flujo de datos en el sistema.

Nivel Externo: es el más cercano a los usuarios, es decir, le concierne la forma de como cada usuario ve los datos.

Nivel Conceptual: es un nivel de mediación entre los otros dos.

La arquitectura de datos es un conjunto de capas de los modelos que proporciona una base sólida para las iniciativas estratégicas, tales como:

- Una estrategia de datos, destacando los objetivos de la empresa y los objetivos de mejorar la recopilación y uso de datos.
- Mejoras en los procesos de negocios.
- Las decisiones sobre el futuro de los sistemas nuevos y modificados.
- Integración de almacenamiento de datos, y las iniciativas de presentación de informes.

El estudio de estos tipos de arquitectura resulta de gran importancia para la presente investigación, pues nos permite conocer a fondo las características elementales para el desarrollo de este trabajo. A continuación se muestran algunos beneficios de la Arquitectura de software:

1.4 Soluciones arquitectónicas de diversos ERP

Un ERP puede verse como una arquitectura de software que facilita el flujo de información entre las funciones de producción, logística, finanzas y recursos humanos de una empresa. ERP es la columna vertebral tecnológica del Negocio electrónico, la aplicación al área transaccional. La tecnología de los sistemas ERP se basa en la arquitectura cliente / servidor, en la que un computador central (servidor), tiene capacidad para atender a varios usuarios simultáneamente (clientes).

A continuación se aborda sobre algunas soluciones arquitectónicas de ERP existentes en el mundo desde el punto de vista arquitectónico:

Openbravo ERP: Se desarrolla utilizando estándares abiertos. La arquitectura empleada por el sistema es MVC (Modelo Vista Controlador). Incorpora muchas ingeniosas características que le hacen destacarse entre la multitud y le convierten en un perfecto software para empresas. (Openbravo, 2007)

Compiere: es una solución 100% Java sobre base de datos Oracle, con servidor de aplicaciones JBOSS. Utiliza como base de datos PostgreSQL u Oracle y está desarrollado utilizando la herramienta OpenACS basada a su vez en el servidor web AOLserver. Se ejecuta bajo la licencia pública Compiere

(CPL), que permite el paso a privativo de dicho software transcurridos dos años desde su fecha de lanzamiento. Mantiene una arquitectura modelo-vista-controlador, basada en web y en la interfaz de usuario. Considera cada cliente como una entidad que tiene la posibilidad de ejecutar transacciones, las cuales estarán asociadas a un documento (facturas de venta, recibo de materiales, documentos de entregas, pagos a proveedores o recibos de clientes). Este sistema se organiza en procesos de negocio y no en módulos.

OpenXpertya: Es el ERP de software libre en español y latinoamericano con la mayor red de soporte profesional, el mayor número de usuarios del sistema, e implantaciones finales reales y está situado en el grupo de cabeza de los de mayor madurez. Está desarrollado en tres capas e íntegramente en Java, con lo que funciona sobre cualquier sistema operativo y plataforma (Windows, Solaris, FreeBSD, Linux, UNIX, AIX, MacOS, etc.) sin dependencias de ningún tipo. (OpenXpertya, 2005)

Luego de este estudio realizado sobre la solución arquitectónica de algunos sistemas ERP existentes en el mundo, teniendo en cuenta el tipo de arquitectura que utilizan y su solución de arquitectura, se puede concluir que entre las más aplicadas se encuentra la cliente/servidor y la arquitectura en capas o de tres niveles.

1.5 Arquitectura del proyecto ERP-Cuba

El proyecto ERP-Cuba tiene una arquitectura basada en componentes lo cual favorece la reutilización de todos sus elementos incluyendo los que definen las distintas relaciones entre ellos, al ser este tipo de arquitectura completamente modular. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según las necesidades, sin que se afecte otra parte del sistema. Dado que un componente puede ser construido y luego mejorado continuamente, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

Dentro del proyecto existen varios subsistemas y cada subsistema cuenta con una serie de componentes que pueden ser mejorados de forma individual además reutilizados por otro subsistema que lo necesite. Dentro de cada uno de estos componentes el patrón de diseño que se utiliza es el Modelo-Vista-Controlador.

Modelo-Vista-Controlador: Se utiliza cuando es necesario modularizar la interfaz de usuario, las reglas de negocios y el control de eventos.

El **MVC** divide una aplicación interactiva en tres componentes en donde, de manera general, el modelo contiene la capa de datos y la lógica de negocio.

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). Mantiene el conocimiento del sistema. No depende de ninguna vista y de ningún controlador.

Vista: Maneja la visualización de la información.

Controlador: Analiza los mensajes de eventos que recibe el sistema y modifica u obtiene datos del modelo en respuesta a las peticiones del usuario. Evita poner código indebido en la capa impropia, por ejemplo instrucciones de base de datos (modelo) en interfaz de usuario (vista).

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual. En aplicaciones web la separación entre la vista y el controlador está claramente muy bien definida.

Arquitectura de sistema

En el ámbito del proyecto ERP, la arquitectura de sistema es la vista encargada de proponer las partes del software: componentes, conectores, las restricciones y las configuraciones de estas partes, se subdivide en tres vistas fundamentales.

Por las características que presenta el dominio de los negocios a incidir con el desarrollo de la aplicación, y las tendencias y experiencias del desarrollo de otros sistemas ERP, se decide adoptar para el desarrollo horizontal del sistema el estilo arquitectónico orientado a componentes.

Por consiguiente todas las funcionalidades levantadas y modeladas en las fases de negocio y requerimientos quedan expresada o contenida en al menos un componente, y las distintas interacciones entre estos componentes originan funcionalmente la existencia de subsistemas.

Arquitectura de datos

Encargada de todas las definiciones a nivel de datos, de la integración de los distintos modelos, de los

patrones, estándares y definiciones a este nivel.

Manteniendo el enfoque del desarrollo orientado a componentes definido en la descripción de la vista de sistema, y la misma estructura de empaquetamiento se define un esquema para cada subsistema, reduciendo a la menor cantidad posible las integraciones a nivel de datos entre entidades de distintos subsistemas, dejando estas integraciones para que se realicen a nivel de sistema.

Arquitectura de integración

Encargada de los procesos de integración interna (entre componentes de un mismo proyecto) y externa (entre proyectos distintos), establece las definiciones, estándares, protocolos de comunicación y reglas de intercambio de información.

En el desarrollo del proyecto se pueden identificar tres niveles de integración, un primer nivel que se lleva a cabo entre componentes de un mismo subsistema, un segundo nivel el que se origina entre componentes de subsistemas distintos (integración entre subsistemas), y un tercer nivel y más alto en nivel de abstracción el que se efectúa entre sistemas externos y el sistema ERP-Cuba en desarrollo.

1.6 Expediente tecnológico del proyecto ERP-Cuba

Todo control de versiones del flujo de trabajo del proceso de construcción de la arquitectura debe mantenerse organizado, estructurado y ubicado en un sitio de acceso para el equipo de desarrollo, facilitando el uso del mismo y que sirva de guía para el proceso de desarrollo del software en general. Para esto el proyecto cuenta con un expediente tecnológico. El cual está conformado por las vistas definidas en el proyecto, las cuales se listan a continuación:

- ❖ Arquitectura de datos
- ❖ Arquitectura de información
- ❖ Arquitectura de infraestructura
- ❖ Arquitectura de presentación
- ❖ Arquitectura de seguridad
- ❖ Arquitectura de sistema
- ❖ Arquitectura de tecnología

Expediente Arquitectura de sistema

El Expediente de la Arquitectura de sistema recoge la documentación formal e imprescindible con la que se debe contar para la creación de la arquitectura de sistema, ver Fig. 2



Fig. 2-Expediente de la Arquitectura de sistema

Expediente de Arquitectura de datos



Fig. 3-Expediente de la Arquitectura de datos

El espacio **Documentos de soporte** contiene los documentos creados por el equipo de datos para apoyar sus soluciones. Esta puede contener la Matriz de integración que se describe en la vista de datos o cualquier otro tipo de artefacto generado en este sentido. La carpeta llamada **Integración de datos** contiene el documento de integración de datos el cual define cómo se realiza la integración de los diferentes subsistemas y módulos a nivel de datos. Este documento rige el proceso de integración indicando qué componentes entran y salen de cada subsistema, especificando cómo queda modelada esta integración en la base de datos y modelando por cada subsistema cómo se integra a los demás a ese nivel. Además, en este documento se realiza una Matriz de trazabilidad de datos por cada subsistema que resulta de gran ayuda para la integración. Especifica detalladamente los datos de entrada y de salida de los subsistemas, por cada subsistema se declara con cual otro se integra especificando la tabla origen, la tabla destino, la relación entre las tablas y la variable que será objeto de esta integración. A este documento también se le pueden agregar otros aspectos que se consideren

importantes para la integración por parte del equipo de Arquitectura de datos, además de los mencionados. Luego, el **Modelo de datos** contiene los modelos de datos de cada uno de los subsistemas y módulos del sistema en general. Por cada módulo se tienen 3 subcarpetas, como se muestra en la Figura 5, una que contiene un Diccionario de datos, que es la descripción de todos los objetos de la base de datos; otra para el modelo de la base de datos que recoge el archivo con la base de datos modelada y por último una subcarpeta en la que se guarda una imagen de este mismo modelo para facilitar su visualización sin tener que ejecutar la herramienta con que se modeló, para ver el Modelo de datos ver Fig.4.



Fig. 4-Modelo de datos

En espacio **Pruebas de concepto** están contenidas las pruebas realizadas directamente a la base de datos como son la prueba de rendimiento, que implica un trabajo alterno con la prueba de tamaño de la base de datos. La prueba de concurrencia a la base de datos es el acceso a ella de varios usuarios al mismo tiempo sin tener fallas en las peticiones, además otro aspecto a tener en cuenta es el del respaldo o seguridad de la base de datos, entre otros que deben ser documentados como constancia de los chequeos al sistema.

El documento de **Soluciones de la Arquitectura de datos** se encuentra en la carpeta **Soluciones de la Arquitectura de datos**, en este se reflejan todas las soluciones implementadas por el equipo de arquitectura de datos, dando primeramente una panorámica de la situación que originó la implementación de la misma. Luego detalla la solución implementada y finalmente se refleja la prueba de concepto realizada a esta solución y su resultado.

La **Vista de datos** contiene en su interior el documento de Vista de Arquitectura de datos, este documento constituye la guía para el desarrollo de la base de datos desde el punto de vista arquitectónico, con el propósito de estandarizar el desarrollo de todas las líneas, generalizando patrones, herramientas, nomenclaturas y otros aspectos que mejoran el rendimiento de la base de datos. Aquí se definen las políticas de trabajo en cuanto al desarrollo de la base de datos, es decir, políticas para la integración entre líneas, para la creación de índices, el control de cambios, para el uso de UPDATE y DELETE en CASCADA, para el uso de los tipos de datos teniendo en cuenta el uso de réplicas entre base de datos y además para el uso de reglas de nomenclatura y estándares definidos para los objetos de la base de datos. De igual forma se indica cómo se garantiza la seguridad en la base de datos, especificando qué gestor y qué servidores de base de datos utilizar, cómo realizar el respaldo de la misma, qué usuarios tiene la base de datos y qué permisos tiene cada uno. Para asegurar un código más legible y reutilizable, de manera que se pueda aumentar su mantenibilidad a lo largo del tiempo, en este documento también se establecen las pautas para normar el comentariado en la base de datos. Igualmente se estandariza la nomenclatura que se utiliza en la creación de tablas, esquemas, base de datos, campos, llaves primarias, llaves foráneas, secuencias, funciones, triggers, tipos de datos, vistas, dominios, etc. La Vista de datos declara además qué herramientas se usan en el diseño y la implementación de la base de datos. Describe cómo son representados los diferentes tipos de datos (Numeric, Date, Varchar, etc.), cómo se realizará la generación de llaves primarias, la concurrencia y la normalización de árboles. En este orden también se define la política de indexado por la que se regirá la base de datos y se regula como se asegura el rendimiento de la misma indicando aspectos que pueden mejorar dicho rendimiento, como son cambios en el diseño de las tablas o de las consultas y la reducción del tamaño. La normalización y la desnormalización de la base de datos también se definen aquí a fin de guiar a los implementadores de la base de datos en este sentido. Otro aspecto que debe quedar plasmado en este documento es cómo se realiza la distribución a nivel de datos que resulta de gran ayuda al equipo de implantación para realizar el despliegue del sistema.

1.7 Artefactos generados en el proyecto ERP-Cuba

En el centro se utiliza un estándar de arquitectura llamado modelo de producción para el flujo de arquitectura el cual tiene como objetivo definir una guía de actividades para el flujo de arquitectura en el desarrollo de los productos del Centro, en aras de proveer un mecanismo de comunicación unificado entre las estructuras del proyecto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el proyecto se generan varios artefactos que son recogidos en el expediente de arquitectura de sistema, de ellos solo los que conciernen a este trabajo son los que se muestran en la siguiente tabla, la cual guarda una descripción de cada uno en cuanto a artefacto, descripción y roles, ver Tabla1.

Tabla 1-Relación artefacto-descripción-roles. (Leyet, 2010)

Artefacto	Descripción	Roles
Documento Línea Base de Subsistema	Delimita las responsabilidades arquitectónicas de un subsistema. Permite establecer la comunicación y dependencias entre los componentes y módulos que lo integran, así como plasmar la criticidad y complejidad de los componentes con vistas a su reutilización y prioridad en el desarrollo.	Arquitecto de componentes
Complejidad y criticidad	Contiene todos los componentes definidos en los subsistemas y por cada componente los requisitos funcionales que implementan, la complejidad de dichos requisitos y una serie de datos desde el punto de vista de integración.	Arquitecto de sistema
Especificación de componentes	Se listan todos los componentes definidos en los subsistemas y de cada uno de ellos se especifica el nombre, una descripción además de sus características fundamentales y los servicios que brinda.	Arquitecto de sistema
Matriz de integración	Contiene todos los componentes definidos en los subsistemas de forma matricial y en las intercepciones se especifican los servicios que consume dicho componente.	Arquitecto de sistema

Modelo de datos	Contiene todas las tablas, atributos y relaciones.	Arquitecto de datos
Diccionario de datos	Contiene una descripción detallada de las tablas, atributos y relaciones del subsistema.	Arquitecto de datos
Matriz de trazabilidad de datos	Establece las relaciones existentes entre las tablas de los subsistemas.	Arquitecto de datos
Pruebas de concepto	Describe los diferentes escenarios de prueba, junto con los resultados arrojados del subsistema en que se realizan dichas pruebas.	Arquitecto de datos

Para la realización de algunos de estos artefactos se utilizó la herramienta de modelado, Visual Paradigm en su versión 6.3, en el siguiente epígrafe se detallan características de la misma.

1.8 Herramientas a utilizar

Visual Paradigm 6.3

Visual Paradigm para UML es una herramienta ampliamente utilizada en el mundo del software que permite a los profesionales modelar sus diseños. Posibilita la integración de aplicaciones empresariales a las bases de datos. Es capaz de generar código e ingeniería inversa para lenguajes como el PHP. Permite manejar grandes estructuras de manera eficiente, solo requiere una configuración de escritorio común. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegues (*Visual Paradigm, 2006*).

PostgreSQL 8.2

PostgreSQL es una fuente poderosa, abierta al sistema de la base de datos correlativo. Posee más de 15 años de desarrollo activo y una arquitectura probada, ha ganado una reputación fuerte para la

fiabilidad, integridad de los datos y exactitud. Corre en todos los sistemas operativos incluidos Linux y Windows. También apoya el almacenamiento de grandes objetos binarios, incluso imágenes, sonidos y videos. Tiene interfaces nativas de la programación para los lenguajes C++, Java.net, Perl, Python, Ruby, entre otros y una documentación excepcional. Entre sus principales características pueden encontrarse:

Disponible totalmente sin costo alguno.

Disponible para los Sistemas Operativos UNIX y Windows.

Soporte total del Modelo Relacional de Bases de datos.

Extensiones propias a SQL para realizar consultas sobre la base de datos.

Dependencias entre objetos, integridad referencial.

Soporta valores no atómicos como dominio de un campo. (Gómez,2008)

1.9 Conclusiones parciales

En este capítulo se realizó un estudio de los temas relacionados con la arquitectura de software, el cual permitió sentar las bases teóricas para la correcta realización del trabajo que se pretende lleva a cabo. Se tratan temas de arquitectura de software, Vistas de sistema y datos, se muestran algunos sistemas de planificación de recursos empresariales existentes en cuanto a la arquitectura utilizada, así como algunos elementos de la arquitectura y expediente tecnológico del proyecto ERP-Cuba.

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

En este capítulo se hace un análisis de los mapas de procesos del negocio y modelos conceptuales de los subsistemas Activos Fijos y Facturación, los cuales son los artefactos de entrada del negocio, es válido decir que estos artefactos constituyen elementos esenciales que se tienen en cuenta para la realización de los artefactos generados en la vista de arquitectura de sistema, los cuales serán realizados contando con una breve descripción de cada uno de ellos.

2.1 Mapas de procesos y modelo conceptual

Los artefactos del negocio constituyen la entrada para la realización de los artefactos de la vista de arquitectura de sistema y es de gran importancia su previo análisis.

El mapa de procesos permite conocer cómo se entrelazan los distintos procesos por los que pasa una determinada institución, para lograr un objetivo específico. En este artefacto se muestran las entradas que necesita cada proceso para desarrollarse, así como las salidas generadas por los mismos. Con un mapa de procesos se puede explicar de manera global el funcionamiento de los procesos existentes, pues este se realiza en función de las relaciones entre ellos mediante los elementos considerados como entradas y salidas.

Para lograr entender los artefactos del negocio, es necesario tener las descripciones de los procesos de ambos subsistemas. En el subsistema Activos Fijos se definieron los siguientes procesos:

Altas de activos fijos tangibles: El objetivo del proceso de negocio Altas de Activos Fijos Tangibles es registrar las adquisiciones de activos nuevos, de uso, construidos con medios propios y contratados con terceros, así como las donaciones.

Bajas de activos fijos tangibles: El objetivo del proceso de negocio Baja de Activos Fijos Tangibles es registrar las bajas que causan los AFT por realizarse una venta, un envío a reparar, un traslado a ocioso, un traslado interno, un alquiler o un ajuste de inventario.

Traslados internos de activos fijos tangibles: El objetivo del proceso de negocio Traslados Internos de Activos Fijos Tangibles es registrar el cambio (administrativamente dispuesto) de ubicación física o de destino económico de los Activos Fijos Tangibles entre establecimientos, dependencias o de un área de responsabilidad a otra, dentro de una misma entidad.

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

Traslados a ocioso y ocioso a activo de activos fijos tangibles: El objetivo del proceso de negocio Traslados a Ocioso y Ocioso a Activo de Activos Fijos Tangibles es controlar el estado técnico en que se encuentran los AFT.

Activos fijos tangibles enviados a reparar: El objetivo del proceso de negocio Activos Fijos Tangibles Enviados a Reparar es registrar el movimiento que se le hace a un activo fijo cuando se envía a reparar, dentro o fuera de la entidad.

Arrendamientos de activos fijos tangibles: El objetivo del proceso de negocio Arrendamientos de Activos Fijos Tangibles es registrar el movimiento que se le hace a un activo fijo cuando realiza un alquiler a alguna entidad de un AFT.

Inventarios totales y parciales de activos fijos tangibles: El objetivo del proceso de negocio Inventarios Totales y Parciales de Activos Fijos Tangibles es llevar un control de todos los Activos Fijos Tangibles que se encuentran registrados contablemente, en cada área de responsabilidad.

Ajuste de inventario de activos fijos tangibles: El objetivo del proceso de negocio Ajuste de Inventario de Activos Fijos Tangibles es llevar un control de los movimientos que se realizan originados por las siguientes causas:

1. Por Alta: Sobrante detectado al efectuarse inventarios o comprobaciones.
2. Por Baja: Faltante detectado al efectuarse inventarios, comprobaciones o robos.
3. Pérdidas: Representa el monto de los Activos Fijos Tangibles que son dados de baja definitivamente por diversas causas, tales como: pérdidas por siniestros y desastres naturales

Movimientos de los activos componentes de un módulo o unidad básica o funcional de activos fijos tangibles: El objetivo del proceso de negocio Movimientos de los Activos Componentes de un Módulo o Unidad Básica o Funcional de Activos Fijos Tangibles es definir el tratamiento a seguir en los casos de movimientos de Activos Fijos Tangibles integrantes de un Módulo.

Apertura y cierre de activos fijos tangibles: El proceso de Apertura tiene el objetivo, fijar un inventario inicial de Activos Fijos Tangibles.

A continuación se muestra el mapa de proceso del subsistema Activos Fijos Tangibles:

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

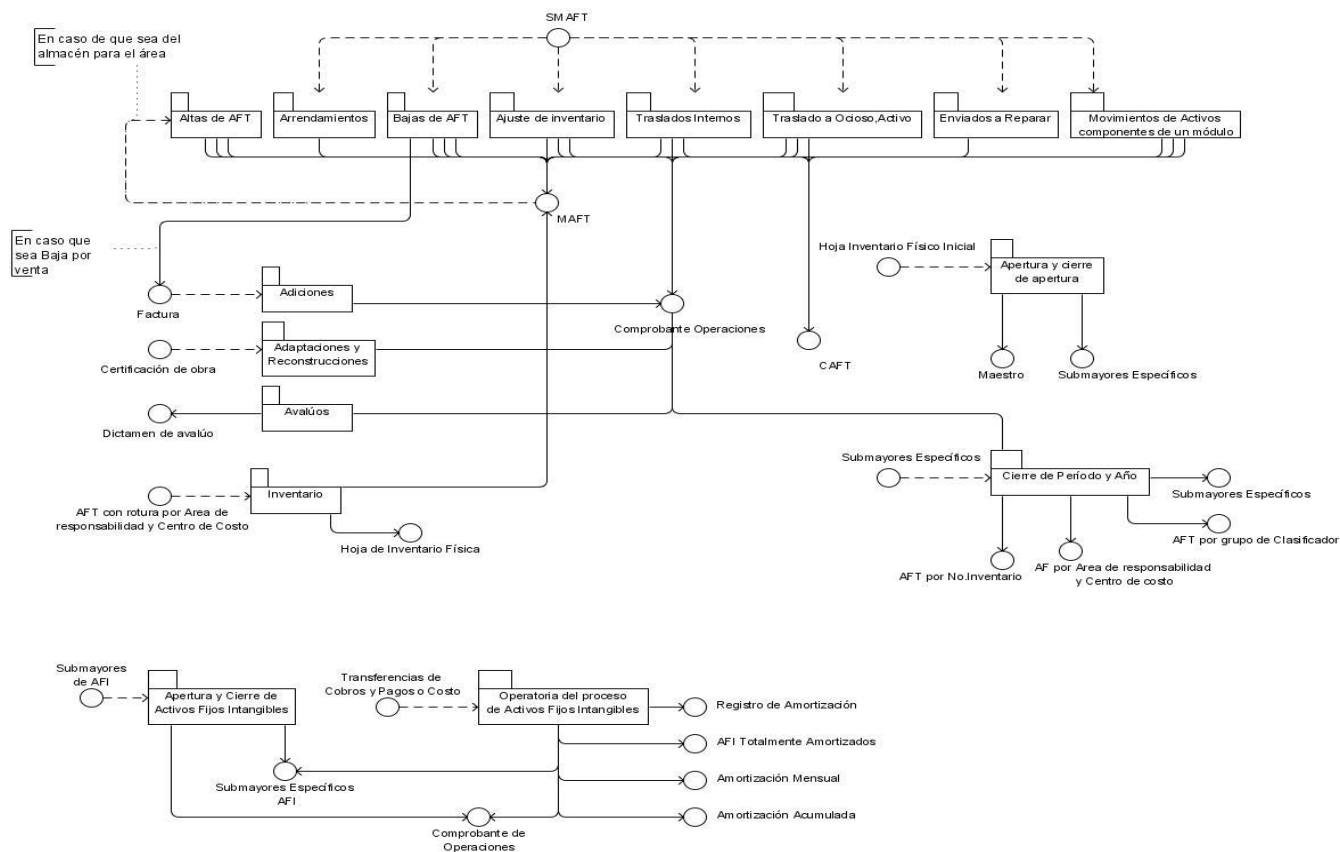


Fig. 5-Mapa de Procesos de Negocio de AFT

Seguidamente se muestran las descripciones de los procesos pertenecientes al subsistema Facturación:

Descripción de procesos de negocio: el proceso de facturación es el encargado de formalizar las ventas de productos, prestaciones de servicios, así como las entregas de productos en consignación o en depósito, para esto se necesitarán las autorizaciones de entregas y como salida del proceso sería el documento de entrega. Cada uno de los procesos identificados a continuación está dado por el tipo de facturación que se realiza.

Facturación Venta de Mercancías: En este proceso se crean los documentos que formalizan la venta de mercancía en cualquier entidad, en dependencia del documento de autorización de entrega que sea

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

recibida, para obtener el documento de entrega que se obtiene en facturación, que puede ser únicamente una factura.

Entradas:

Orden de Entrega o Despacho

Plan de Distribución

Conduces

Salidas:

Facturas de inventario

Comprobante de operaciones

Facturación Prestación de Servicios: En este proceso de acuerdo a la Orden de Servicio que es una de las autorizaciones de entrega, se generará la factura que amparará el servicio prestado y así formalizar el pago de dicho servicio.

Entradas:

Orden de Servicio

Salidas:

Facturas de Servicio

Comprobante de operaciones

Facturación Venta de AF: Se realiza la facturación de la autorización de entrega correspondiente, es decir facturar un modelo de movimiento de activo fijo por venta con el objetivo de formalizar el pago de dicho activo fijo.

Entradas:

Modelo de movimiento de AF por venta

Salidas:

Facturas de activo fijo

Comprobante de operaciones

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

Prefacturación venta de mercancías o prestación de servicio: Se realizan las prefecturas de los pedidos que realizó un cliente.

Entradas:

Solicitud de compra de mercancías o prestación de servicios.

Salidas:

Prefacturas

A continuación se muestra el mapa de procesos del subsistema facturación:

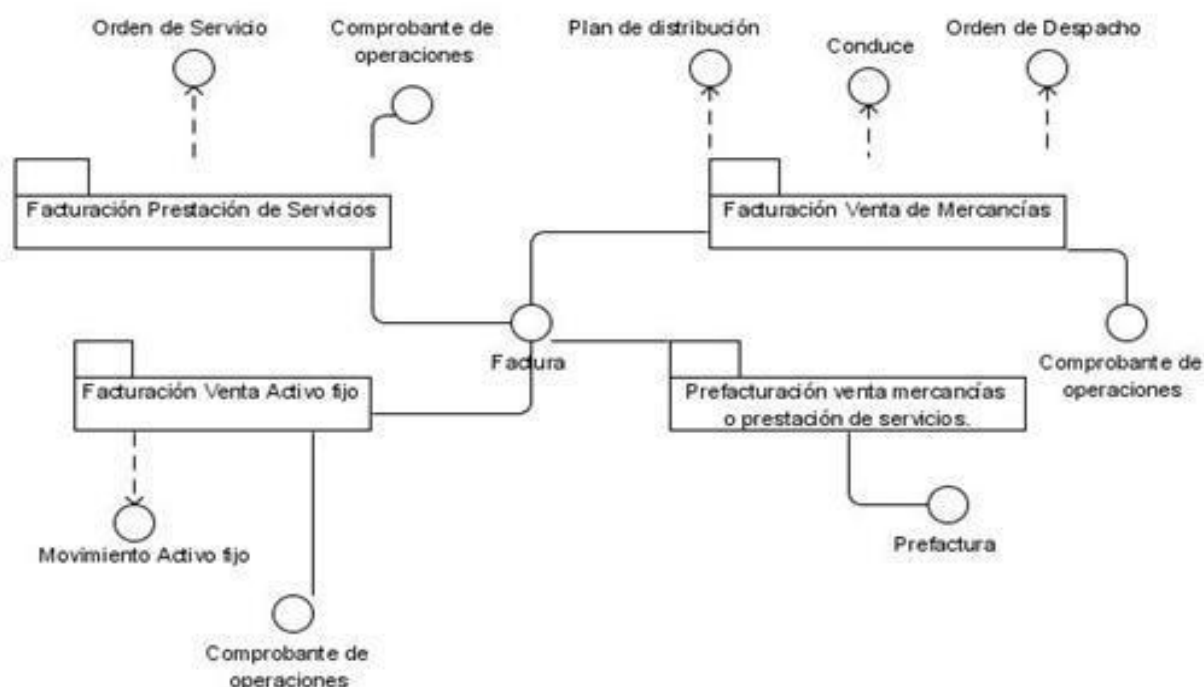


Fig. 6-Mapa de Procesos de Negocio de Facturación

Se hace necesario identificar primeramente los conceptos fundamentales del negocio para realizar un correcto diseño de la base de datos, los conceptos que deben ser almacenados en la base de datos así como los atributos que le van a dar respuesta a los requisitos. Para alcanzar el éxito en este sentido es necesario realizar un mapa de conceptos que posteriormente serán representados en el modelo de datos físico de la base de datos, para ver el modelo conceptual de AFT, ver Fig.7.

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

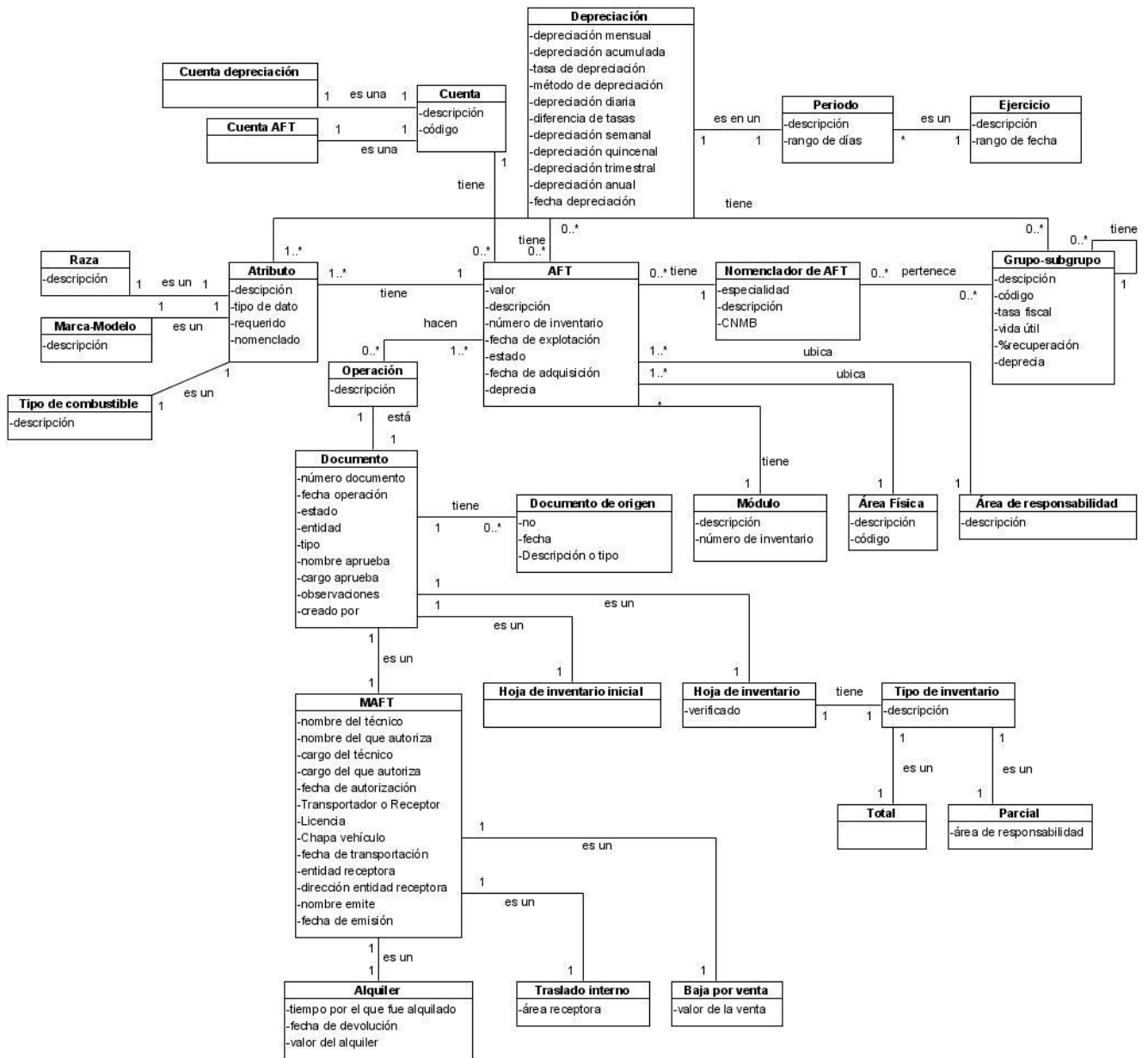


Fig. 7-Modelo Conceptual de AFT

A continuación se muestra un ejemplo del modelo conceptual del subsistema Facturación.

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

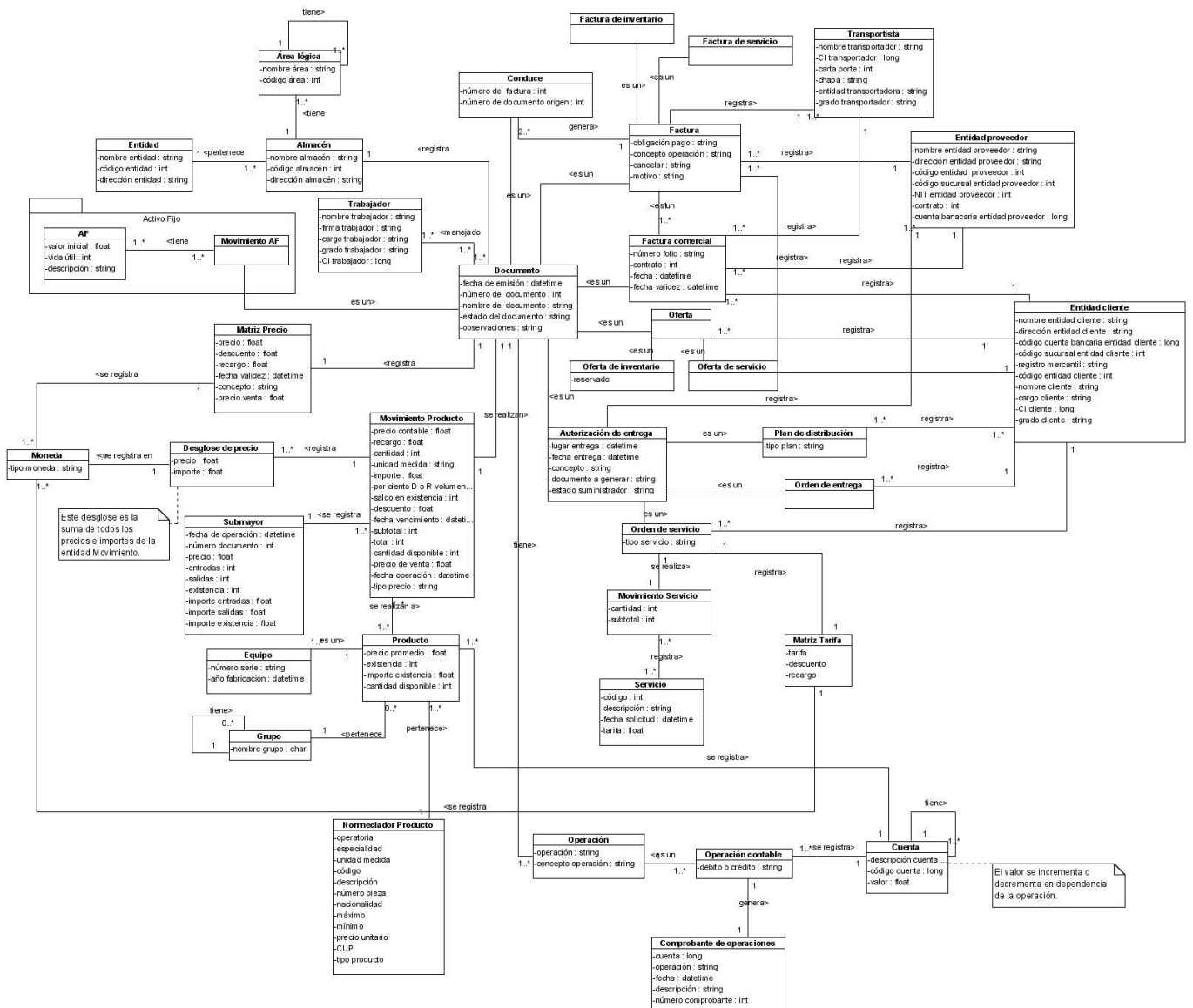


Fig. 8-Modelo Conceptual de Facturación

2.2 Descripción de la estructura de empaquetamiento de los componentes

Cada componente está empaquetado independiente del resto de los componentes del sistema, acorde a las decisiones de arquitectura de sistema tomadas por el proyecto ERP-Cuba. Cada componente tiene implementado un modelo, un controlador y una vista (*models*, *controllers* y *views* en inglés, en ese mismo orden), compuesto por un conjunto de clases y paquetes, encargados de gestionar el

acceso a datos, los eventos del sistema, atendiendo así las solicitudes que provienen de la vista y del modelo, y de definir las interfaces de usuario para modelar visualmente la funcionalidad del componente en cuestión. Dentro del paquete Modelo, se localizan los paquetes Negocios (*business* en inglés), que comprende las clases del negocio, y dominio (*domain* en inglés), que presenta las clases entidad, en las cuales se gestiona la persistencia de los datos en la base de datos. También contiene una interfaz para establecer la comunicación con los componentes con los que se relaciona, además de brindarles una serie de funcionalidades que se agrupan como servicios dentro de un mismo paquete denominado Validaciones (*validators* en inglés), que engloba las reglas del negocio.

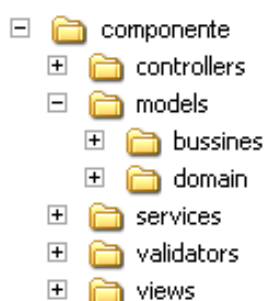


Fig. 9-Estructura de un componente

2.3 Vista arquitectónica de los subsistemas Activos Fijos y Facturación

2.3.1 Complejidad y criticidad

El artefacto análisis de complejidad y criticidad contiene todos los componentes definidos en el subsistema, y por cada componente los requisitos funcionales que implementa, la complejidad de dichos requisitos, la cantidad de servicios que brinda y la categoría o tipo de componente. También muestra la dependencia entre componentes, a partir de la cual, conjuntamente con la prioridad de los requisitos se calcula la complejidad y criticidad de los componentes. Este análisis se realiza para conocer la complejidad de cada componente y su relevancia desde el punto de vista de integración. Estos aspectos son de gran importancia a tener en cuenta a la hora de establecer una prioridad de implementación o para aceptar o no un cambio determinado.

El valor de complejidad de un componente se calcula mediante la cantidad de requisitos que contiene este, más la complejidad de cada requisito por cuatro, más la cantidad de servicios que este ofrece,

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

duplicada. Mientras, la criticidad del componente depende de la cantidad de componentes que utilicen sus servicios, sumado a los valores de complejidad que presentan. Las fórmulas serían las siguientes:

Complejidad = cantidad de requisitos + (complejidad de los requisitos*4) + (cantidad de servicios*2)

Criticidad = dependencia + \sum complejidad de los componentes dependientes

Tiene como objetivo evaluar y valorar cuán importante pudiera ser cada componente desde el punto de vista de integración. Por tanto la dependencia entre componentes y la prioridad de los requisitos permitirá definir una prioridad de implementación, así como determinar la complejidad y criticidad de cada componente.

Hay elementos que se deben tener en cuenta, por ejemplo el tipo de componente, en los subsistemas en cuestión sólo existen dos tipos, ellos son:

Núcleo: Representa los componentes con la responsabilidad de abstraer las principales funcionalidades del negocio de la organización, agrupan la mayor parte de los conceptos del modelo conceptual.

Configuración: Representa los componentes con la responsabilidad de abstraer las configuraciones y parametrizaciones estáticas o dinámicas del sistema.

A continuación se muestra el análisis de complejidad y criticidad para el componente Configuración del subsistema Facturación, ver Fig. 10.

Componente	Cantidad	No.	Requisitos	Código	Complejidad	Servicios	Componentes	Tipo de Comp	Comple	Criticidad d
Configuración	16	1	Adicionar precio de venta	Log_Conf_Adic_pre	2	6	1	Configuración	212	632
		2	venta	Log_Conf_Mod_pre	1					
		3	Eliminar precio de venta	Log_Conf_Elim_pre	2					
		4	Listar precio de venta	Log_Conf_List_prec	2					
		5	Buscar producto	Log_Conf_Busc_Pro	2					
		6	Mostrar logotipo de entidad	Log_Conf_Mostrar	2					
		7	calidad en la factura	Log_Conf_Imprimir	2					
		8	Adicionar tarifa	Log_Conf_Adic_Tarif	2					
		9	Modificar tarifa	Log_Conf_Modif_Tar	3					
		10	Eliminar tarifa	Log_Conf_Elim_Tarif	7					
		11	Listar tarifa	Log_Conf_List_Tarif	6					
		12	Buscar servicio	Log_Conf_Buscar_s	2					
		13	Adicionar cuenta por desg	Log_Conf_Adic_Cue	4					
		14	Modificar cuenta por desg	Log_Conf_Mod_Cue	2					
		15	Eliminar cuenta por desg	Log_Conf_Elim_Cue	3					
		16	precio	Log_Conf_List_desg	4					

Fig. 10- Representación del artefacto Análisis de Complejidad y Criticidad del subsistema Facturación

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

Luego del análisis de criticidad y complejidad de los componentes del subsistema Facturación se obtuvo como resultado que el componente menos complejo es Nomencladores con un valor de ciento ochenta y siete. Luego Configuración con doscientos doce y como componente más complejo, Facturación con un valor de 444. De igual forma, los componentes menos críticos resultaron ser Configuración y nomencladores con valor de tres, y el de mayor valor de criticidad fue el componente facturación con valor de cuatro.

Seguidamente se muestra el artefacto Análisis de complejidad y criticidad del componente AFT perteneciente al subsistema Activos Fijos:

Componente	Cantidad	No.	Requisitos	Código	Complejidad	Servicios	Componentes	Tipo de Compl	Criticidad	
aft	13	1	Ubicar AFT	Log_aft_ubicarAFT	2	11	3	Configuración	219	544
		2	Modificar ubicación física de AFT	Log_aft_modUbicació	1					
		3	Adicionar marca-modelo	Log_aft_AdicMarca	6					
		4	Modificar marca-modelo	Log_aft_ModMarca	5					
		5	Eliminar marca-modelo	Log_aft_ElimMarca	1					
		6	Buscar marca-modelo	Log_aft_BuscarMarca	4					
		7	Adicionar operaciones no contables	Log_aft_AdicionarOpe	7					
		8	Modificar operación no contable	Log_aft_ModOper	2					
		9	Eliminar operación no contable	Log_aft_ElimOperacN	2					
		10	Listar operaciones no contables	Log_aft_ListarOperNo	2					
		11	Buscar operación no contable	Log_aft_BuscarOpNo	3					
		12	Importar activo fijo tangible	Log_aft_ImpotAFT	1					
		13	Realizar apertura	Log_aft_RealizApert	5					
		14	Realizar cierre de apertura	Log_aft_RealizCierreA	5					

2.3.2 Priorización de los componentes

Para decidir qué componentes desarrollar primero se tuvo en cuenta la cantidad de dependencias que generan cada uno de ellos y la criticidad de los componentes dependientes, donde el resultado de esta suma proporciona la criticidad del componente (calculada en el artefacto anterior) y a su vez define su prioridad. Por ello se consultó el artefacto Criticidad Y Complejidad donde se analizó por cada componente su criticidad, la cual va a decidir la prioridad de desarrollo de los componentes. A continuación se muestra la los componentes priorizados del subsistema Facturación. Ver Tabla 2

Tabla 2-Priorización de los componentes

Componente	Criticidad	Prioridad
Facturación	403	1

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

Nomencladores	213	2
Configuración	188	3

Seguidamente se mostrará la priorización de los componentes del subsistema Activos Fijos:

Tabla 3-Priorización de los componentes del subsistema Activos Fijos

Componente	Criticidad	Prioridad
documentos	675	1
modificaciones	675	1
inventario	577	2
aft	449	3
nomencladores	286	4
depreciación	286	4
configuración	260	5
módulos	219	6
movimiento	0	7
cierre	0	7

2.3.3 Matriz de integración

Contiene de forma matricial los componentes definidos para los subsistemas de la solución propuesta, con el objetivo de registrar las relaciones entre los componentes mediante los servicios consumidos por cada componente dentro del subsistema, a continuación se muestra la Matriz de integración del subsistema Activos Fijos, ver Fig.11.

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

	Componentes	Brindan						
		nomencladore	AFT	movimientos	documentos	configuracion	comun	depreciacion
Consumien	AFT	ObtenerGrupo, devolverGrupo subgrupos, Dev	existeActivoFT	buscarMovimientosPorDocumentoAft				
	comun				GetDocumento			
	conciliacion		obtenerCodigoAFTDadold	cantidadMovimientosPorDocumento, buscarMovimientosPorDocumentoAft, crearMovimientoAft, adicionarMovimientoAft, eliminarMovimientoAft, obtenerAreasResponsabilidad	BuscarDocsInventarioAFT			
	configuracion			buscarMovimientosPorDocumentoAft	GetDocsAFT			
	depreciacion		obtenerActivosAreaResponsabilidad, obtenerAftDadoldAFT, devolverIdgrupoDa			DevolverConfiguracionAFT, DevolverNomDepreciacion		
	documentos	ObtenerGrupo, ObtenerAtributos, ObtenerOperacionDadold	modificarActivoFT	cantidadMovimientosPorDocumento	AdicionarDocAFT, CargarDatosAdicionarAFT, CargarDatosModificarAFT, ModificarDocAFT, EliminarDocAFT, ContabilizarDocAFT, CancelarDocAFT, ConfirmarDocAFT, CancelarEstadoDocAFT, CargarDatosAdicionarAFT, CargarDatosModificarAFT, cantidadMovimientosPorDo, ModificarDocAFT_Modelcumento, ObtenerOperacionDadold, AdicionarDocAFT_Model, CargarDatosEncabezadoModificarDocAFT_Model			
	entrada		eliminarActivosFT, o	eliminarMovimientoA	AdicionarDocAFT_Model, Adic			
	inventario		obtenerCodigoAFTD	cantidadMovimientos	DatosEncabezadoInventario, b			
	modificaciones	ObtenerGrupo, ObtenerAtributos, buscarValorAtributoActivo, modificarValorAtributoActivo,	devolverIdgrupoDadoldActivo, obtenerAftDadoldAFT, modificarActivoFTDadold, devolverTodosActivos	cantidadMovimientosPorDocumento, buscarMovimientosPorDocumentoAft, adicionarMovimientoAft, eliminarMovimientoAft	ConfirmarDocAFT			
	modulos		eliminarActivosFT, devolverActivosDeModulo, obtenerModulosAft, devolverTodosActivos, obtenerAftDadoldAFT, obtenerInstanciaDatModulo, modificarModuloAFT, obtenerAftDadoldA					
	noentrada		obtenerAftDadoldAFT, modificarActivoFT, modificarActivoFT, ConfirmarDocAFT, obtenerDatosNomxId	eliminarMovimientoAft, existeMovimientoAft, crearMovimientoAft, buscarMovimientosPorDocumentoAft, adicionarMovimientoAft	AdicionarDocAFT_Model, ModificarDocAFT_Model, EliminarDocAFT, ContabilizarDocAFT, CancelarDocAFT			
	nomencladores		DevolverAFTenArea, SetearAFT, existeGrupoEnActivos			InsertarOperacion, OperacionByIdoperacion, EliminarOperacion	DevolverColeccionElementos, Devolve	
	recuperaciones	devolverGrupo subgrupos	obtenerTodosAFTParaSubmayor, obtenerCodigoAFTDadold					
submayor	devolverGrupo subgrupos, obtenerTodosAFT	obtenerTodosAFTParaSubmayor, obtenerCodigoAFTDadold			Buscartipodoc		operacionDe prec, calcTasa Deprec	

Fig. 11- Representación de la Matriz de integración AFT

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

En la siguiente figura se muestra la Matriz de integración del subsistema Facturación:

		Brindan			
Componentes		configuracion	nomencladores	documentos	servicio
Consumen	configuracion		devolverServicios,GetCantServicios		
	documentos				
	facturacion	preciosdeproducto,preciosdeproducto,serviciopreciove	buscarconcepto,BuscarServicioDadold	Operaciones,GetDocumento,BuscDocGenerado	Obtenermovserv,Eliminarmovserv,Existencia
	nomencladores	conceptoasignado			

Fig. 12- Representación de la matriz de integración del subsistema Facturación

Para ver el artefacto Matriz de Integración completo ver Anexo 7.

2.3.4 Documento Línea Base

La realización del documento línea base pretende especificar las dependencias existentes entre los componentes. Tiene como objetivo fijar las responsabilidades arquitectónicas de los subsistemas, así como plasmar la criticidad y complejidad de los componentes con vistas a su reutilización y prioridad en el desarrollo.

El diagrama de componentes del subsistema Activos Fijos cuenta con un total de 18 componentes como se muestra en la figura, (ver Fig. 13). Como puede observarse el modelo contiene las relaciones entre ellos y representan los servicios que brindan y consumen de otros componentes mediante las interfaces que tienen previamente definidas.

Por solo poner un ejemplo para su mejor comprensión, el componente AFT consume servicios de Documentos general y a su vez el primero le brinda servicios a los componentes GestionInventario, Módulo, Gestión de Modificaciones, Gestión de entradas y No Entrada.

CAPÍTULO 2: VISTA DE ARQUITECTURA DE SISTEMA

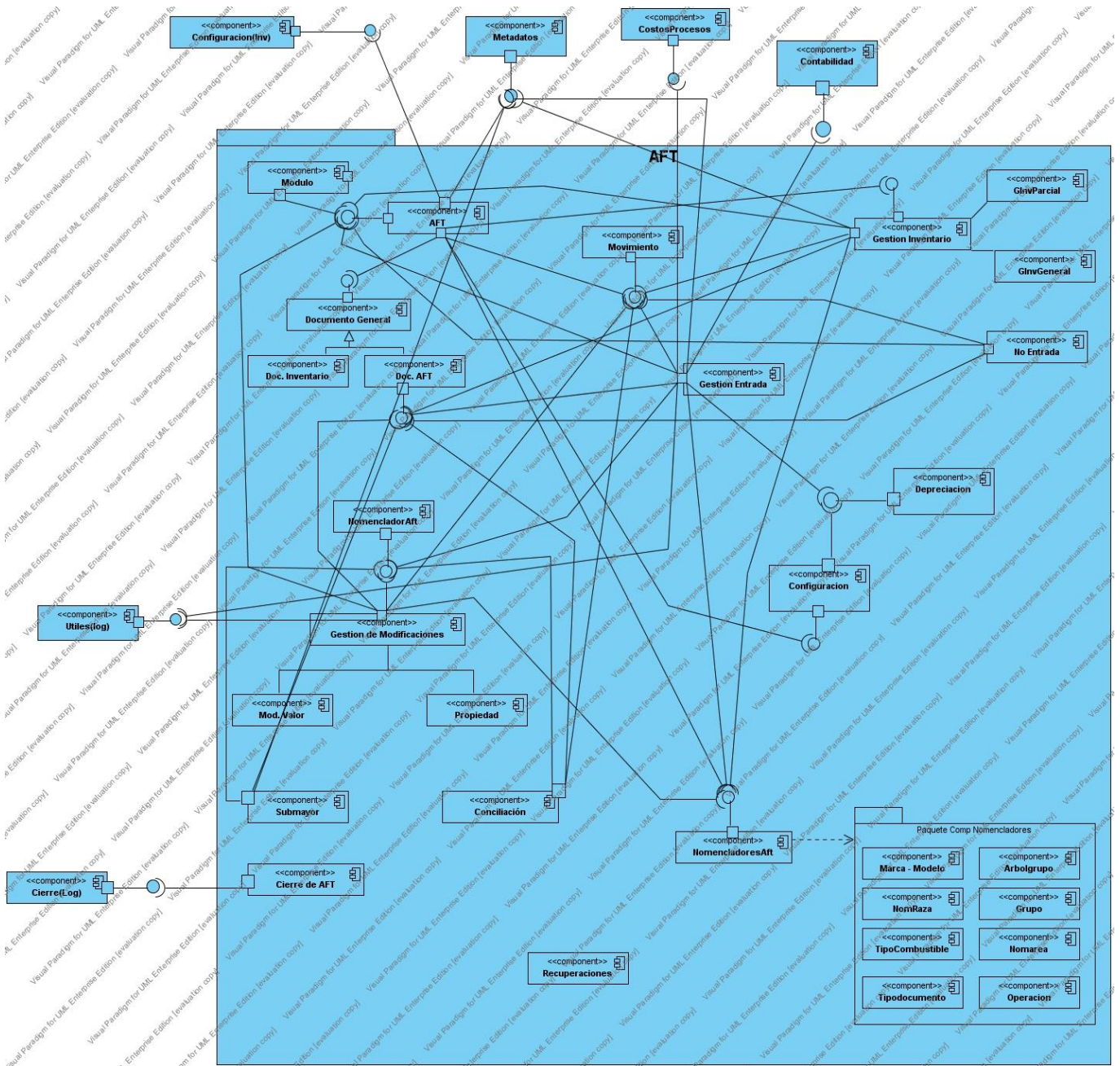


Fig. 143-Mapa de componentes AFT

A continuación se muestra el diagrama de componentes del subsistema Facturación, ver Fig. 14.

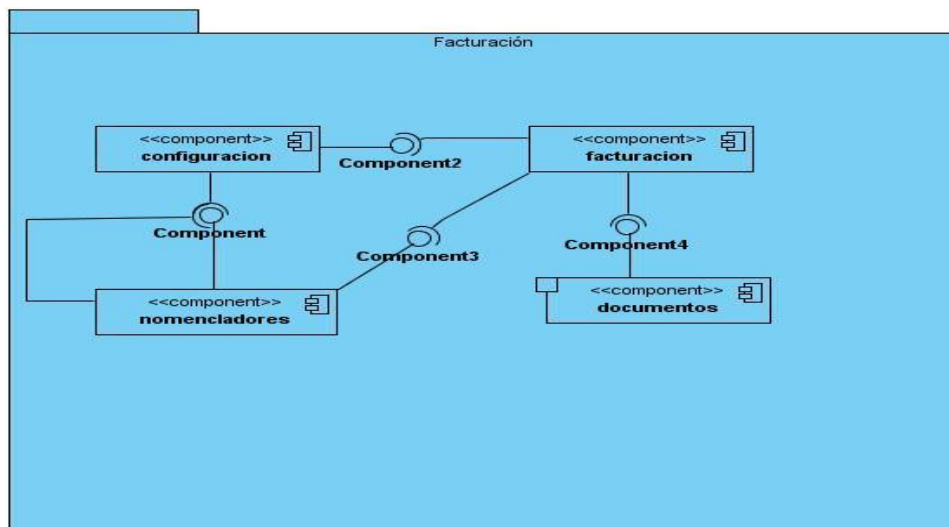


Fig. 154-Mapa de Componente de Facturación

En el subsistema Facturación, el componente facturación le brinda servicios a los componentes configuración, nomencladores y documentos, este último a su vez le brinda servicios a configuración y a sí mismo, y el componente documentos le proporciona servicios a facturación.

2.4 Conclusiones parciales

Luego de la realización de este capítulo, queda elaborada la vista de sistema, se logra establecer y detallar los lazos de colaboración entre los distintos componentes, dentro de cada uno de los subsistemas analizados. Se obtiene la formalización de la descripción arquitectónica mediante los artefactos propuestos por la arquitectura de sistema, los cuales son: Especificación de componentes, Criticidad y complejidad de los componentes, la Matriz de integración y el documento Línea Base. Además se determina el nivel de complejidad y de criticidad que presentan estos componentes desde el punto de vista de la integración; elementos utilizados por el equipo de desarrollo del proyecto, para puntualizar la prioridad que estos presentan durante el desarrollo de los subsistemas a los cuales pertenecen.

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

La vista de arquitectura de datos es la tercera y última vista con que cuenta la vista de sistema y queda conformada a partir de los elementos resueltos en el capítulo anterior. En este capítulo se muestran los artefactos generados en la vista de arquitectura de datos de los subsistemas Activos Fijos y Facturación, como son el Diccionario de datos y la Matriz de trazabilidad de datos y las Pruebas de concepto.

3.1 Modelo de datos

El Modelo de datos del subsistema Activos Fijos está compuesto por treinta y cuatro tablas, once son nomencladores, veintidós contienen los datos que persisten y una tabla que guarda el historial de los datos. En la fig.15 se describen algunas de las tablas más importantes. Para ver el Modelo de datos de forma íntegra debe dirigirse al Diccionario de datos, en el expediente de datos correspondiente a este módulo ubicado en el repositorio del proyecto ERP-Cuba.

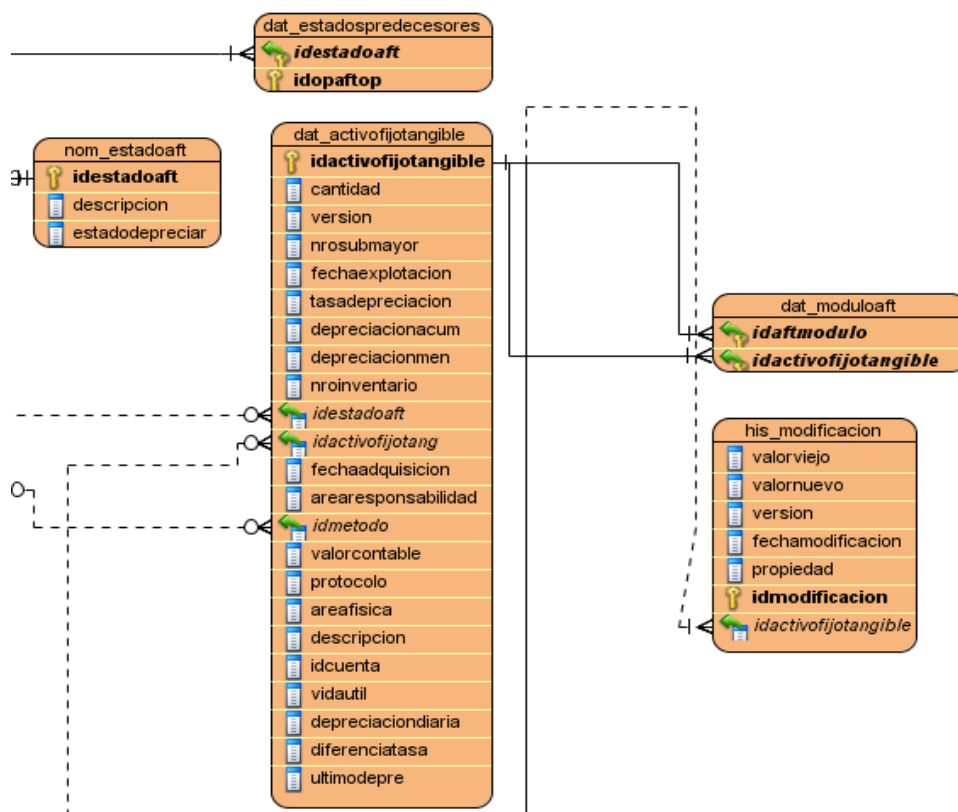


Fig. 165-Representación del modelo de datos del subsistema Activos Fijos

Para ver el Modelo de datos del subsistema Activos Fijos completo, ver Anexo1.

Por otro lado, el Modelo de datos del subsistema Facturación está compuesto por diecisiete tablas, catorce guardan los datos que persisten en la base de datos, cuatro son nomencladores y dos se encargan de las configuraciones. En la fig. 16 se muestran algunas tablas de este subsistema. Para ver el Modelo de datos del subsistema de forma íntegra debe dirigirse al Diccionario de datos, en el expediente de datos correspondiente a este módulo ubicado en el repositorio del proyecto ERP-Cuba.

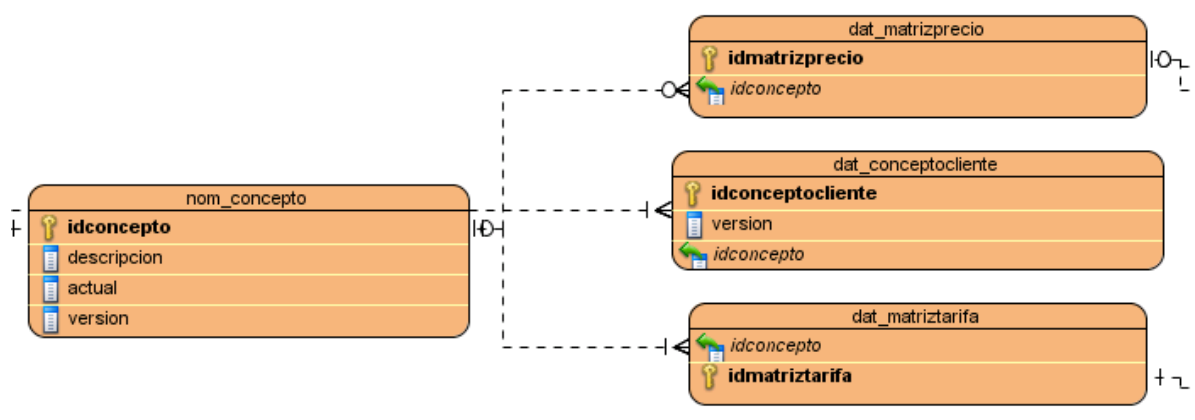


Fig. 176- Representación del modelo de datos del subsistema Facturación

Para ver el modelo de datos del subsistema Facturación completo, ver Anexo2

3.1.1 Algunas especificaciones del Modelo de datos

Concurrencia

La concurrencia no es más que cuando se accede a modificar algo por una o más personas al mismo tiempo, por tanto, el control de esta es otra de las soluciones presentes en el modelo. Dicho control evita que una misma tupla pueda ser accedida por uno o varios clientes al unísono con el propósito de realizar una acción determinada, en especial a la hora de hacer una modificación (Ver figura 17), esto a la vez evita la pérdida de datos o información, mantiene la integridad de los datos y no permite que un usuario sobrescriba la información de otro, siendo esta última la más actualizada.

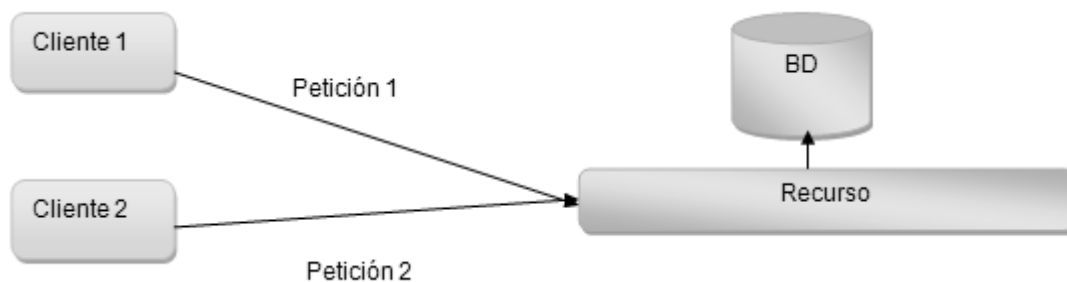






Fig. 187- Ejemplo de concurrencia

Para evitar estas situaciones se le agrega a la tabla un campo versión con un límite X, el cual informa las veces que la tupla es modificada, algunas de las tablas que cuentan con esto son: nom_gruposubgrupo en Activos Fijos y dat_precioentidad en Facturacion. Dicha actualización es controlada mediante un trigger que verifica que la nueva versión que pone el cliente es igual a la antigua versión, si esto ocurre, entonces en este campo se incrementará en uno y permitirá modificar la tupla accedida.












3.2 Diccionario de datos

El Diccionario de datos es una plantilla que permite conocer los datos con sus respectivas validaciones, restricciones, relaciones, dependencia y descripciones. Permitiendo conocer el flujo de datos en una entidad específica, es un artefacto que permite contar con una descripción detallada de todas las tablas con que cuenta el Modelo de datos, relaciones entre ellas y funciones. A continuación se muestra el Modelo de datos del subsistema Activos Fijos un ejemplo del Diccionario de datos del mismo:















Tabla 4-Diccionario de datos del subsistema Activos Fijos

Name	Documentation
 Subsistema Activo Fijo Tangible	
 dat_devolucionaft	Guarda los datos sobre la devolución del documento
 nom_areafisica	Nomenclador que almacena datos del área física del activo
 nom_tipocombustible	Nomenclador que guarda el tipo de combustible

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

 dat_activofijotangible	Tabla que almacena los datos de los AFTs.
 nom_estadoaft	Nomenclador que describe los datos del estado del AFT
 dat_areatrasladoaft	Tabla que guarda el lugar de traslado del AFT
 dat_moduloaft	Tabla que guarda el id del modulo de un AFT
 his_modificacion	Recoge datos sobre el proceso de modificación
 dat_modelo	Se guardan los datos del documento
 nom_metododepreamort	Nomenclador donde se almacenan los datos de los métodos de pre-amortización
 dat_movimientoinventarioaft	Tabla que guarda los datos de los movimientos que se hagan de inventario
 dat_movimientoaft	El objetivo de esta tabla es tener un registro histórico de los documentos, los activos que entraron en un documento y las propiedades tanto generales como específicas del activo en ese momento
 dat_transportacionaft	Tabla que guarda los datos de la persona que transporta y la entidad
 nom_activofijotangible	Nomenclador que guarda los datos del AFT
 dat_movimientodualidadaft	Tabla que guarda los datos de los movimientos de dualidad monetaria del activo
 dat_tipomodeloaft	Almacena los datos del tipo de modelo
 dat_alquileraft	Almacena los datos sobre el alquiler
 dat_movimientoactivo	Tabla que registra el movimiento de un activo en un movimiento dado
 dat_activoentidad	Tabla que guarda el movimiento de un activo en una entidad dada
 nom_atributo	Nomenclador que guarda los datos del atributo

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

 nom_grupo subgroup	Nomenclador grupos de subgrupos
 dat_entidad grupo	Tabla que guarda los datos de la entidad de un grupo de subgrupo cogiendo el id (idestructuracomun) de Estructura y Composición.
 dat_grupo atributos	Tabla en que se almacenan los datos de atributos de los subgrupos
 nom_tipo dato aft	Nomenclador que almacena los datos del tipo de activo fijo tangible
 dat_atributos	Tabla que guarda los datos de los atributos
 dat_entidad atributos	Tabla que guarda datos sobre la entidad cogiendo el id de Estructura y Composición
 dat_datos grupo subgroup	Tabla que almacena los datos de los grupos de subgrupos
 nom_marca modelo	Nomenclador que guarda los datos sobre la marca modelo del AFT
 nom_raza	Nomenclador que guarda los datos de raza
 nom_secuencia activo fijo tangible	Nomenclador que guarda los datos sobre las secuencias de un activo fijo tangible
 conf_cierre depreciacion	Almacena los datos sobre el cierre de depreciación
 dat_referencia doc	Tabla que almacena la referencia del documento
 dat_formato aft	Almacena el formato de la entidad
 dat_ref internado doc	Tabla que almacena la referencia interna del documento

dat_activofijotangible

Name	Value
Documentation	Tabla que almacena los datos de un AFT

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

Schema	mod_activofijotangible
Primary Key Constraint Name	dat_activofijotangible_pkey

Columnas

Name	Date type	PK/FK	Nullable	Documentation
idactivofijotangible	Numeric (19)	PK	No	Id de la tabla de dat_AFT
cantidad	Numeric (19)		Yes	Este atributo si es mayor que 1 es un módulo.
versión	Numeric (6)		No	Esto es para la concurrencia
nrosubmayor	Numeric (5)		Yes	Número consecutivo del activo
fechaexplotación	Date (0)		Yes	Fecha de alta
tasadepreciación	Numeric (8)		Yes	Tasa anual de depreciación que se aplicó
depreciaciónacum	Numeric (8)		Yes	Sumatoria de las depreciaciones en un rango de fechas
depreciaciónmen	Numeric (8)		Yes	Importe de lo que va a depreciar mes a mes
nroinventario	Varchar (255)		Yes	Puede entrar letras y números
idestadoaft	Numeric	FK	Yes	Llave foránea de la tabla dat_activofijotangible que es llave


CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS


	(19)			primaria en la tabla nom_estadoaft
idactivofijotang	Numeric (19)	FK	Yes	Llave foránea de la tabla que es llave primaria en la tabla nom_activofijotangible
fechaadquisición	Date (0)		Yes	Fecha en la que fue adquirido el AFT, puede no coincidir con la fecha de alta
árearesponsabilidad	Numeric (19)		Yes	Área que tiene un jefe, no es un área física
idmétodo	Numeric (19)	FK	Yes	Llave foránea de la tabla dat_activofijotangible que es llave primaria en la tabla nom_metododepremort
valorcontable	Numeric (19)		Yes	Sumatoria de los valores contables de las monedas alternativas
protocolo	Varchar (19)		Yes	Es para guardar los datos dinámicos del AFT
areafísica	Numeric (19)		Yes	Identificador del local físico donde se encuentra el AFT
descripción	Varchar (255)		Yes	Descripción del AFT
idcuenta	Numeric (19)		Yes	Cuenta de gasto del AFT
vidaútil	Numeric (19)		Yes	Vida útil probable del activo
depreciacióndiaria	Numeric (8)		Yes	Importe de lo que va a depreciar diariamente
diferenciatasa	Numeric		Yes	Diferencia entre la tasa fiscal y la

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS


	(8)			calculada al insertar el AFT
últimodepre	Numeric (19)		Yes	Último período que depreció el activo


Relaciones


dat_activofijotangible_fk1 : Relationship	
From	 nom_estadoaft
Documentation	Relación de cero o uno, a cero o muchos entre las tablas nom_estadoaft y dat_activofijotangible que guarda el identificador del estado del AFT(idestadoaft), que será llave primaria de la tabla nom_estadoaft y llave foránea de la tabla dat_activofijotangible
Identifying	False
To Multiplicity	0..*
From Multiplicity	0..1


dat_activofijotangible_fk : Relationship	
From	 nom_activofijotangible
Documentation	Relación de uno, a muchos entre las tablas nom_activofijotangible y dat_activofijotangible que guarda el identificador del AFT(idactivofijotang), que será llave primaria en la tabla nom_activofijotangible y llave foránea en dat_activofijotangible
Identifying	False
To Multiplicity	0..*
From Multiplicity	0..1


CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

dat_activofijotangible_fk3 : Relationship	
From	 nom_metododepreamort
Documentation	Relación de cero o uno, a cero o muchos entre las tablas nom_métododepreamort y dat_activofijotangible que guarda el identificador del método de preamortización(idmétodo) y será llave primaria en la tabla nom_métodopreamort y llave foránea en dat_activofijotangible
Identifying	False
To Multiplicity	0..*
From Multiplicity	0..1

dat_moduloaft_fk : Relationship	
To	 dat_moduloaft
Documentation	Relación de uno a muchos entre las tablas dat_activofijotangible y dat_moduloaft que tiene como llave foránea en la tabla dat_moduloaft a identificador idactivofijotangible
Identifying	True
To Multiplicity	1..*
From Multiplicity	1

dat_moduloaft_fk1 : Relationship	
To	 dat_moduloaft
Documentation	Relación de uno a muchos entre la tablas dat_activofijotangible y datmoduloaft que guarda el identificador de el AFT(idactivofijo tangible),que será llave primaria de la tabla dat_activo fijo tangible y llave foránea de dat_moduloaft
Identifying	True
To Multiplicity	1..*
From Multiplicity	1

fk_idactivofijotangible : Relationship	
To	 dat_movimientoinventarioaft
Documentation	Relación de uno a muchos entre las tablas dat_activofijotangible y dat_movimientoinventarioaft que guarda el identificador del AFT (idactivofijotangible), que será llave primaria de la tabla dat_activofijotangible y llave foránea en dat_movimientoinventarioaft
Identifying	True
To Multiplicity	1..*
From Multiplicity	1

his_modificacion_fk1 : Relationship	
To	 his_modificacion
Documentation	Relación de uno a muchos entre las tablas dat_activofijotangible y his_modificación que guarda el identificador de una AFT (idactivofijo tangible), que será llave primaria de la tabla dat_activofijotangible y llave foránea de his_modificación
Identifying	False
To Multiplicity	1..*
From Multiplicity	1

Triggers

Nombre	Documentación
t_actualizar_cambio_condicion_de_pago	Crea un disparador antes de insertar en la tabla dat_condiciónpago y después ejecuta la función ft_actualización_cambio_condición_de_pago ()

Procedimientos Almacenados



CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

ft_actualizacion_arbol_nom_raza()	
Documentación	Actualiza el árbol del nomenclador de raza, pregunta si el padre es nulo le asigna el nuevo idraza al padre, sino retorna el nuevo idraza
Esquema	mod_activofijotangible
Create Statement	<pre>CREATE FUNCTION ft_actualización_árbol_nom_raza() RETURNS AS ' BEGIN if (new.idpadre is null) then new.idpadre = new.idraza; end if; RETURN new; END; ' LANGUAGE plpgsql</pre>
Procedure Name	ft_actualización_árbol_nom_raza
Comentario	Description
Procedure Type	Returns Result
Datos Modelo	2


Para ver el Diccionario de datos del subsistema completo ver Anexo 1.

A continuación se muestra un ejemplo del Diccionario de datos del subsistema Facturación:






Tabla 5--Diccionario de datos del subsistema Activos Fijos

Nombre	Documentación
 Subsistema Facturación	
 dat_matrizprecio	Tabla de datos que almacena el id del producto, el id del concepto por entidad y el id de la entidad a la que corresponde el producto. Tabla que permite la relación de todos ellos y el almacenamiento de los precios de los productos

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

	por concepto.
 dat_matrizpreciodualidadmonetaria	Tabla que contiene la estructura requerida para almacenar los datos de un producto asociado a una entidad por concepto donde su valor esta dado en más de una moneda.
 dat_conceptocliente	Nomenclador que almacena los conceptos por clientes.
 nom_concepto	Nomenclador que contiene los conceptos por los que se agrupan los precios de venta de los productos y el precio de los servicios.
 dat_matriztarifa	Donde se ubican los precios de venta de los servicios.
 dat_matriztarfdualmon	Es la que lleva la dualidad monetaria para los servicios.
 dat_precio	Tabla que almacena todas las operaciones que se realizan en las entidades, dado que registra las cuentas que controla la entidad (precio, recargo, descuento e impuesto de circulación, margen comercial, subsidio).
 dat_movimientoventa	Es la que guarda los datos de los movimientos de venta que se realizan
 dat_tarifa	Tabla que guarda las tarifas de precios de los servicios por entidad.
 dat_relacionprecio	Tabla que almacena datos sobre la categoría de precio y la entidad.
 nom_categoriaprecio	Nomenclador que contiene los precios con sus descripciones por entidad.
 dat_precioentidad	Tabla que almacena todas las operaciones que se realizan en las entidades, dado que registra las cuentas que controla la entidad (precio,

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

	recargo, descuento e impuesto de circulación, margen comercial, subsidio).
 nom_secuenciafacturacion	Nomenclador que guarda datos sobre la secuencia de facturación.
 nom_certificacionlogotipo	Nomenclador que contiene datos e imágenes asociados a la certificación y logotipos de las facturas.
 dat_impuestocirculacion	Tabla que recoge los datos del impuesto de circulación
 cfg_formatoentidad	Tabla donde se relacionan el formato que viene de estructura y composición y la entidad
 cfg_configuracionfactura	Se almacenan los datos previos para realizar una factura según configuración.

dat_matrizpreciodualidadmonetaria

Nombre	Valor
Documentation	Tabla que contiene la estructura requerida para almacenar los datos de un producto asociado a una entidad por concepto donde su valor esta dado en más de una moneda.
Schema	mod_facturación
Primary Key Constraint Name	dat_matrizpreciodualidadmonetaria_pkey


Columnas


Nombre	Tipo dato	PK/FK	Nulo	Documentation
idmatrizpreciodualidadmonetaria	numeric (19)	PK	No	Identificador que se genera automáticamente con la secuencia sec_datmatrizpreciodualidadmonetari

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

a_seq				
idmatrizprecio	numeric(19)	FK	Si	Llave foránea que viene de dat_matrizprecio dado que el precio de la misma pudiera estar en otro tipo de moneda.
precioventa	numeric(25)		Si	Precio de venta de los productos.

Relationships

dat_matrizpreciodualidadmonetaria_fk : Relationship	
From	 dat_matrizprecio
Documentation	Relación de cero a muchos entre las tablas dat_matrizprecio y at_matrizdualidadmonetaria que guarda el identificador de la matriz de precio (idmatrizprecio), que será llave primaria de la tabla dat_matrizprecio y llave foránea de dat_matrizpreciodualidadmonetaria
Identifying	false
To Multiplicity	0..*
From Multiplicity	0..1

dat_precio_fk1 : Relationship	
To	 dat_precio
Documentation	Relación de uno a muchos entre las tablas dat_matrizpreciodualidadmonetaria y dat_precio que guarda el identificador(idmatrizpreciodualidadmonetaria), que será llave primaria de la tabla dat_matrizpreciodualidadmonetaria y foránea de dat_precio
Identifying	False

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

To Multiplicity	1..*
From Multiplicity	1

Constraints

dat_matrizpreciodualidadmonetaria_idx		
Columns	Idmatrizprecio	
	Documentation	Llave foránea del la tabla dat_matrizpreciodualidadmonetaria
	Type	numeric
	Nullable	true
	Unique	false
	Primary Key	false
	Length	19
	Scale	0
	Generated	0

Triggers

Nombre	Documentación
t_actualizar_cambio_condición_de_pago	Crea un disparador antes de insertar en la tabla dat_condición_pago y después ejecuta la función ft_actualizacion_cambio_condición_de_pago()

Procedimientos Almacenados

ft_chequear()	
Documentación	Chequea si la vieja versión es igual a la nueva, si es así, la incrementa en 1 sino

	lanza una excepción y devuelve el nuevo elemento.
Esquema	mod_facturación
Create Statement	<pre> CREATE FUNCTION ft_chequear() RETURNS AS ' BEGIN IF(OLD.version = NEW.version)THEN NEW.version = NEW.version + 1; ELSE RAISE EXCEPTION 'CONCURRENCE RESTRICTION'; END IF; RETURN NEW; END; ' LANGUAGE plpgsql </pre>
Procedure Name	ft_chequear
Comentario	Description
Procedure Type	Returns Result
Datos Modelo	2

Para ver el Diccionario de datos del subsistema completo ver Anexo 2.

3.3 Matriz de trazabilidad de datos

La Matriz de trazabilidad es uno de los artefactos que generan los arquitectos de datos, se desarrolla por la presencia en las clases, de datos que son de subsistemas diferentes. Por esta razón se hace necesario contar con algún tipo de documento que sirva de guía al arquitecto de datos a la hora del desarrollo o mantenimiento de la base de datos. Estos datos se utilizan en más de un componente y en más de una clase dentro de cada componente. En esto radica su importancia y aplicabilidad, ya que sirve para tener el control de que subsistema proviene cada dato y donde se utiliza. Para realizar la matriz se analizaron cada una de las clases persistentes con la que cuentan los subsistemas Activos Fijos y Facturación. De esta forma, se conforma la Matriz de trazabilidad que se muestra a continuación del subsistema Activos Fijos:

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

Tabla 6- Matriz de integración del subsistema Activos Fijos

Esquemas Entrada →	Activo Fijo					
Esquemas Salida ↓	tabla origen	Variable	relación	tabla destino	valor por defecto	estado de integraci
	nom_filaestruc	idfila		dat_entidadatributos	no null	✘
	nom_filaestruc	idfila		dat_entidadgrupo	no null	✔
Configuración(Datos Maestros)						!
	dat_formato	idformato	uno a muchos	dat_gruposubgrupos	no null	✔
						!
						!
Nomencladores	nom_activofijotangible	idactfijotang	uno a muchos	dat_activoentidad	no null	✔
	nom_gruposubgrupo	idgruposubgrupos	uno a muchos	dat_activoentidad	no null	✔
	nom_activofijotangible	idactfijotang	uno a muchos	dat_activofijotangible	no null	✔
	nom_estadoaft	idestadoaft	uno a muchos	dat_activofijotangible	no null	✔
	nom_metododepreamort	idmetodo	uno a muchos	dat_activofijotangible	no null	✔
	nom_tipodatoaft	idtipodatoaft	uno a muchos	dat_atributos	no null	✔
	dat_formato	idformatoaft	uno a muchos	dat_cfgaft	no null	✔
	nom_estadoaft	idestadoaft	uno a muchos	dat_estadospredecesores	no null	✔
	nom_activofijotangible	idactfijotang	uno a muchos	dat_movimientoaft	no null	✔
	nom_atributo	idatributos	uno a muchos	dat_valoratributoactivo	no null	✔
	nom_gruposubgrupo	idgruposubgrupo	uno a muchos	dat_datosgruposubgrupo	no null	✔
	nom_areafisica	idareafisica			no null	✘

Fig. 18- Matriz de integración del subsistema Activos Fijos

Para ver el artefacto completo Matriz de trazabilidad de Activos Fijos, ver Anexo 3

A continuación se muestra el artefacto Matriz de trazabilidad del subsistema Facturación:

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

Esquemas Entrada →	Facturación					
Esquemas Salida ↓	tabla origen	Variable	relación	tabla destino	valor por defecto	estado de integración
Estructura y Composición	nom_filaestruc	id_fila	uno a muchos	cfg_configuracionfactura	no null	✓
	nom_filaestruc	id_fila	uno a muchos	cfg_formatoentidad	no null	✓
	nom_filaestruc	id_fila	uno a muchos	dat_matrizprecio	no null	✓
Configuración(Datos Maestros)	dat_operacion	id_operacion	uno a muchos	cfg_configuracionfactura	no null	✓
	dat_nommoneda_entidad	idnommonedaentidad	uno a muchos	dat_matrizpreciodualidadmonetaria	no null	✓
	dat_nommoneda_entidad	id_operacion	uno a muchos	cfg_formatoentidad	no null	✓
	dat_operacion	idnommonedaentidad	uno a muchos	dat_matriztarifadualmon	no null	✓
Nomencladores	nom_concepto	idconcepto	uno a muchos	dat_conceptocliente	no null	✓
	nom_concepto	idconcepto	uno a muchos	dat_matrizprecio	no null	✓
	nom_concepto	idconcepto	uno a muchos	dat_matriztarifa	no null	✓
	nom_categoriaprecio	idcategoriaprecio	uno a muchos	dat_precioentidad	no null	✓
	nom_categoriaprecio	idcategoriaprecio	uno a muchos	dat_relacionprecio	no null	✓
	nom_concepto	idconcepto	uno a muchos	dat_movimientoventa	no null	✓
	nom_clientesproveedores	idclientesproveedores	uno a muchos	dat_conceptocliente	no null	✓

Fig. 199- Matriz de integración del subsistema

Para ver el artefacto completo Matriz de trazabilidad del subsistema Facturación, ver Anexo 4

3.4 Pruebas de concepto

Las Pruebas de concepto constituyen una etapa absolutamente crítica en el lanzamiento de nuevos productos. En ellas se pueden utilizar enfoques cualitativos o cuantitativos. Estos últimos sin embargo, resultan especialmente relevantes para la investigación de mercados que pretenden hacer predicciones. En este artefacto, se analizan todos los requisitos del sistema y las funciones o consultas que dan respuesta a las pruebas realizadas. Además, por cada función o consulta, se analizaron las tablas relacionadas, y se midieron parámetros como: el tiempo en que se ejecutaban, las filas que devolvía y el costo de la función.

Para el llenado de los datos se utilizó la herramienta EMS Data Generator 2005 for PostgreSQL, la cual constituye una herramienta poderosa en la generación de datos de pruebas para base de datos construidas en PostgreSQL. Permite seleccionar las tablas para la generación de los datos, definir para cada campo el rango de valores admisibles, además permite llenar la base de datos de forma eficiente y rápida teniendo en cuenta la integridad referencial de los datos.

Se generaron para la realización de las Pruebas de concepto, un volumen de 5000 tuplas para cada una de las tablas de datos, estas pruebas se hicieron a funciones que responden de manera general a

requisitos de reportes. Se realizó la prueba de conceptos a la función `f_listadodocumentos ()`, perteneciente al subsistema Activos Fijos:

1. Requisito: Listar documento.

Función: `mod_activofijotangible.f_listadodocumentos ()`.

Descripción: Función que retorna todos los datos de los documentos.

Tablas relacionadas:

`mod_documento.dat_documentogeneral`

`mod_documento.dat_documentoaft`

`mod_estructuracomp.dat_estructuraop`

`mod_estructuracomp.dat_estructuraop`

`mod_nomencladores.nom_centrocostopatrimonial`

`mod_datosmaestros.dat_operacion`

`mod_datosmaestros.nom_operaciones`

`mod_datosmaestros.dat_documentoprim`

`mod_datosmaestros.nom_docprim`

Resultados:

Total cost =0.260.00

Plan rows=1000

Plan width=3658

Tiempo total de ejecución de la consulta: 0.790 ms

A continuación se muestra los datos arrojados durante las Pruebas de concepto realizada a la función `f_modificacionaft ()` del subsistema Activos Fijos:

2. Requisito: Modificar AFT en el documento.

Función: `mod_activofijotangible.f_modificacionaft (iddocumento "numeric")`

CAPÍTULO 3: VISTA DE ARQUITECTURA DE DATOS

Descripción: Esta función modifica los datos de un Activo Fijo Tangible, dado el iddocumento del activo que se quiere modificar.

Tablas relacionadas:

mod_documento.dat_documentogeneral

mod_documento.dat_documentoaft

mod_estructuracomp.dat_estructuraop

mod_estructuracomp.dat_estructuraop

mod_estructuracomp.dat_estructuraop estp

mod_activofijotangible.dat_movimientoaft

mod_activofijotangible.dat_movimientoactivo

mod_activofijotangible.his_modificacionaft

mod_nomencladores.nom_centrocostopatrimonial

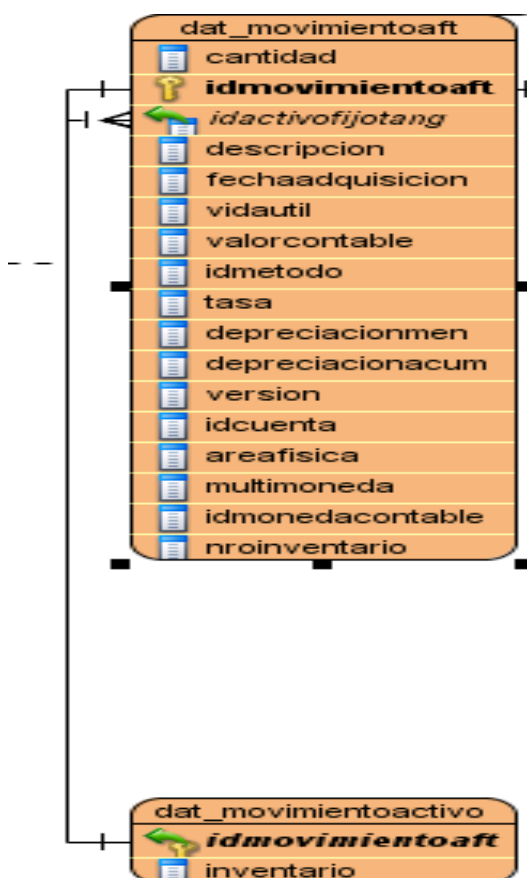


Fig. 20- Tablas relacionadas

Resultados:

Total cost =0.260

Plan rows=1000

Plan width=4710

Tiempo total de ejecución de la consulta: 0.414 ms.

Se puede afirmar que la base de datos para esta función cumple con un tiempo de respuesta bastante bajo, pues retorna todas las filas de la función en un tiempo total de ejecución de sólo 0.04 ms. Para ver el artefacto completo, consultar el mismo en el expediente de datos de cada subsistema.

3.5 Conclusiones parciales

En este capítulo se mostraron los artefactos generados en el Modelo de datos, así como una breve descripción de cada uno de ellos. Se muestra una descripción detallada del Modelo de datos de cada uno de los subsistemas en cuestión. Además, se explicaron situaciones que se presentan en el Modelo de datos de estos subsistemas como concurrencia. Además, se le realizaron las Pruebas de concepto a los requisitos más críticos de cada subsistema obteniendo resultados cuantitativos y cualitativos de los mismos, de tal forma que al final se cuente con una documentación para el mejor entendimiento del equipo de trabajo.

CONCLUSIONES GENERALES

En la realización de este trabajo:

- Se hizo un profundo estudio teórico, que favoreció en el entendimiento de los temas tratados sobre arquitectura de software, lo cual permitió comprender mejor los patrones de diseño y estilos utilizados en los subsistemas Activos Fijos y Facturación. Se estudiaron algunas soluciones arquitectónicas desde el punto de vista de arquitectura evidenciando la necesidad de construir un sistema CEDRUX debido a las ventajas de este y las particularidades del país.
- Se realizó la vista de arquitectura de sistema y datos de las soluciones Activos Fijos y Facturación, la vista de arquitectura de sistema facilitó ver cómo están estructuradas las soluciones antes mencionadas en cuanto a relaciones de dependencias e integración, analizándose la complejidad y criticidad de cada componente, lo cual constituye un punto de partida para la implementación de versiones futuras.
- En la vista de arquitectura de dato se realizó una descripción detallada del Modelo de datos de los subsistemas Activos Fijos y Facturación incluyendo las relaciones existentes dentro del subsistema y la integración con otros componentes fuera del subsistema mediante la matriz de trazabilidad, permitiendo validar el diseño del Modelo de datos.

RECOMENDACIONES

A partir de los resultados obtenidos con la realización del presente trabajo de diploma, se proponen las siguientes recomendaciones:

- Continuar mejorando y actualizando el trabajo realizado para que sirva de referencia o guía de apoyo en el repositorio y lo pueda utilizar todo personal nuevo que se integre al proyecto.
- Realizar las Pruebas de concepto correspondientes al resto de los requisitos funcionales de las soluciones Activos Fijos y Facturación.

REFERENCIAS BIBLIOGRÁFICAS

1. **ARQ_Datos, Equipo de. 2008.** *Documento de Integración de Datos*. UCI. Ciudad habana : s.n., 2008.
2. **Bass, L., Clements, P., Kazman, R. 1998.** download-book.net. *download-book.net*. [En línea] 1998. <http://download-book.net/Bass-L,-Clements-P,-and-Kazman-R.1998-doc.html>.
3. **Carrascoso, Yoan A. y Chaviano, Enrique. 2008.** *Propuesta de Arquitectura Orientada a Servicios para el Módulo de Inventario del ERP Cubano*. C. Habana : s.n., 2008.
4. **Cliente/Servidor, Arquitectura.** csi.map.es. *csi.map.es*. [En línea] <http://www.csi.map.es/csi/silice/Global71.html>.
5. **Georgas, John C., Dashofy, Eric M. y Taylor, Richard N. 2010.** acm. *acm*. [En línea] 2010. [Citado el: 19 de Febrero de 2010.] <http://www.acm.org/crossroads/espanol/xrds12-4/argcentric.html#PerryWolf1992>
6. **2007.** openbravo. *openbravo*. [En línea] 2007. [Citado el: 18 de Febrero de 2010.] <http://www.openbravo.com>.
7. **Kruchten, Philippe. 1995.** synergix.wordpress.com. *synergix.wordpress.com*. [En línea] 1995. <http://synergix.wordpress.com/2008/07/31/las-4-mas-1-vistas>.
8. **MatrixChile. 2007.** matrixchile.com. *matrixchile.com*. [En línea] 2007. [Citado el:] www.matrixchile.com/home/pdf/compiere.ppt.
9. **Mena, Grette L. y Escalona, Arianna. 2009.** *Expediente para la documentación técnica de la Arquitectura de Software de Sistemas de Gestión*. C. Habana : s.n., 2009.
10. **Leyet, Ing. Osmar. 2010.** *Descripción de la Arquitectura de Software*. C. Habana : s.n., 2010.
11. **Reynoso, Carlos Billy. 2004.** willydev.net. *willydev.net*. [En línea] Versión 1.0, Marzo de 2004. <http://www.willydev.net/descargas/prev/IntroArq.pdf>.
12. **Oduardo Gómez, L. B. (2008).** Análisis y diseño del Componente del Sistema de Ingresos dentro del ERP Posta. La Habana: Universidad de las Ciencias Informáticas: s. n., 2008
13. **openbravo. openbravo. 2007.** [En línea] 2007. [Citado el: 18 de Febrero de 2010.] <http://www.openbravo.com>.
14. **2005.** openxpertya. *openxpertya*. [En línea] 2005. [Citado el: 19 de Febrero de 2010.] http://www.openxpertya.org/index.php?option=com_content&task=view&id=3&Itemid=4
15. **Vistas, Introduccion a. 2007.** dc.uba.ar. *dc.uba.ar*. [En línea] 2007. <http://www.dc.uba.ar/materias/arg-soft/2007/cuat1/descargas/>.
16. **Web. 2009.** scribd.com. *scribd.com*. [En línea] 2009. <http://www.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes>.

BIBLIOGRAFÍA

1. **ARQ_Datos, Equipo de. 2008.** *Documento de Integración de Datos*. UCI. Ciudad habana : s.n., 2008.
2. **Bass. L., Clements, P., Kazman, R. 1998.** download-book.net. *download-book.net*. [En línea] 1998. <http://download-book.net/Bass-L.-Clements-P.-and-Kazman-R.1998-doc.html>.
3. **CEIGE. MODELO DE PRODUCCIÓN PARA EL FLUJO DE ARQUITECTURA DE SISTEMA DEL PROYECTO ERP-CUBA. 2009.**
4. **Cliente/Servidor, Arquitectura.** csi.map.es. *csi.map.es*. [En línea] <http://www.csi.map.es/csi/silice/Global71.html>.
5. **Cristiá, Maximiliano. Introducción a la Arquitectura de software. 2007.**
6. **Company, Visual Paradigm international. 2007.** Build Quality Applications Faster, Better and Cheaper. [En línea] 2007. [Citado el: 30 de enero de 2009.] <http://www.visual-paradigm.com/>.
7. **González, Ing. Lariza. 2009.** *Modelo de producción para el flujo de arquitectura*. C. Habana : s.n., 2009.
8. **Kruchten, Philippe. 1995.** synergix.wordpress.com. *synergix.wordpress.com*. [En línea] 1995. <http://synergix.wordpress.com/2008/07/31/las-4-mas-1-vistas>.
9. **MatrixChile. 2007.** matrixchile.com. *matrixchile.com*. [En línea] 2007. [Citado el:] www.matrixchile.com/home/pdf/compiere.ppt.
10. **Mena, Grette L. y Escalona, Arianna. 2009.** *Expediente para la documentación técnica de la Arquitectura de software de Sistemas de Gestión*. C. Habana : s.n., 2009.
11. **Leyet, Ing. Osmar. 2010.** *Descripción de la Arquitectura de software*. C. Habana : s.n., 2010.
12. **Reynoso, Carlos Billy. 2004.** willydev.net. *willydev.net*. [En línea] Versión 1.0, Marzo de 2004. <http://www.willydev.net/descargas/prev/IntroArq.pdf>.
13. **Vistas, Introduccion a. 2007.** dc.uba.ar. *dc.uba.ar*. [En línea] 2007. <http://www.dc.uba.ar/materias/arg-soft/2007/cuat1/descargas/>.
14. **Visual Paradigm. 2006.** [En línea] 2006. [Citado el: 25 de febrero de 2009.] <http://www.visual-paradigm.com/product/vpuml/>.
15. **Web. 2009.** scribd.com. *scribd.com*. [En línea] 2009. <http://www.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes>.
16. **Kicillof, Carlos Reynoso – Nicolás.** Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. [En línea] Marzo de 2004. <http://carlosreynoso.com.ar/archivos/arquitectura/Estilos.PDF>. versión 1.0.

GLOSARIO DE TÉRMINOS

Componente: Menor nivel de abstracción dentro de un sistema. Agrupación de funcionalidades que responden a una necesidad de negocio y que contribuye a favorecer el mantenimiento, la adaptabilidad y la flexibilidad.

Facturación: Operaciones a través de las que la empresa eléctrica calcula y especifica las cantidades adeudadas por el cliente por los consumos de energía realizados en un tiempo determinado, y por otros conceptos derivados de las características de su suministro.

Mantenimiento: Combinación de acciones técnicas y administrativas desarrolladas para sustentar un producto de software

Reutilización: Acción de volver a utilizar los bienes o productos.

SOA: Arquitecturas Orientadas a Servicios.

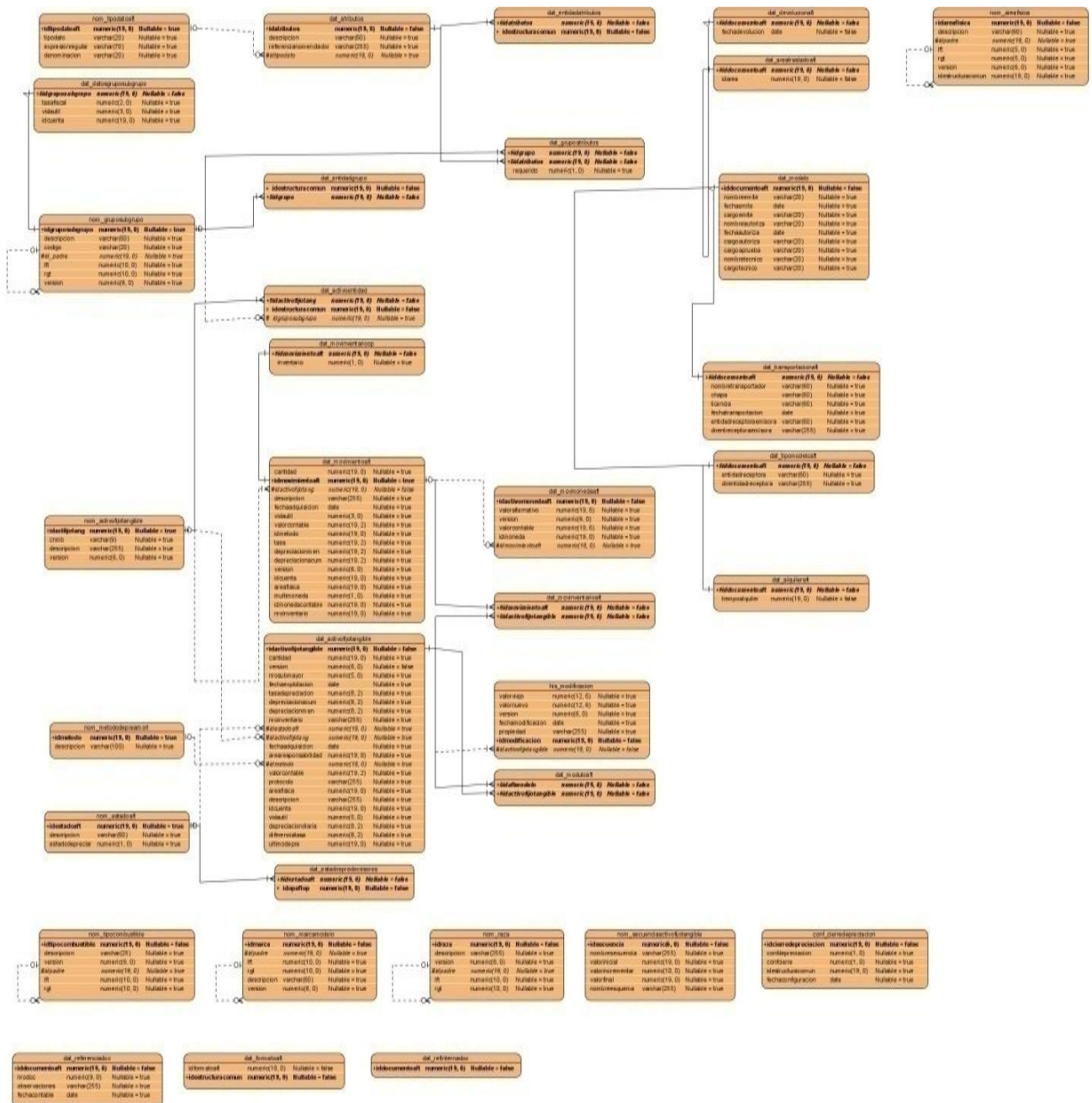
Soporte: Acciones que permiten mantener disponibles los recursos de hardware o software que necesita un producto de software.

Software: Conjunto de instrucciones que al ser ejecutadas proporcionan las características, funciones y el grado de desempeño deseados.

Trigger: Disparador en una Base de datos, es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

ANEXOS

Anexo1: Modelo de datos del subsistema Activos Fijos



Anexo2: Modelo de datos del subsistema Facturación

