Facultad Regional Mártires de Artemisa

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

MINISTERIO DE COMERCIO EXTERIOR
Y LA INVERSIÓN EXTRANJERA

Título: Diseño e implementación de la base de datos para el módulo de la dirección de importaciones (DI) del Ministerio de Comercio Exterior y la Inversión Extranjera (MINCEX).

Autora:

Yisel Melendez Puentes

Tutora:

Ing. Annies Cedeño López

Co-Tutor:

Msc. Rodolfo del C. Piedra Cabrera

Artemisa, Julio de 2012

"Año 54 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas y a la Facultad Regional Mártires de Artemisa para que hagan el uso que estimen pertinente con este trabajo.

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

_____ _____

Yisel Melendez Puentes

Autora

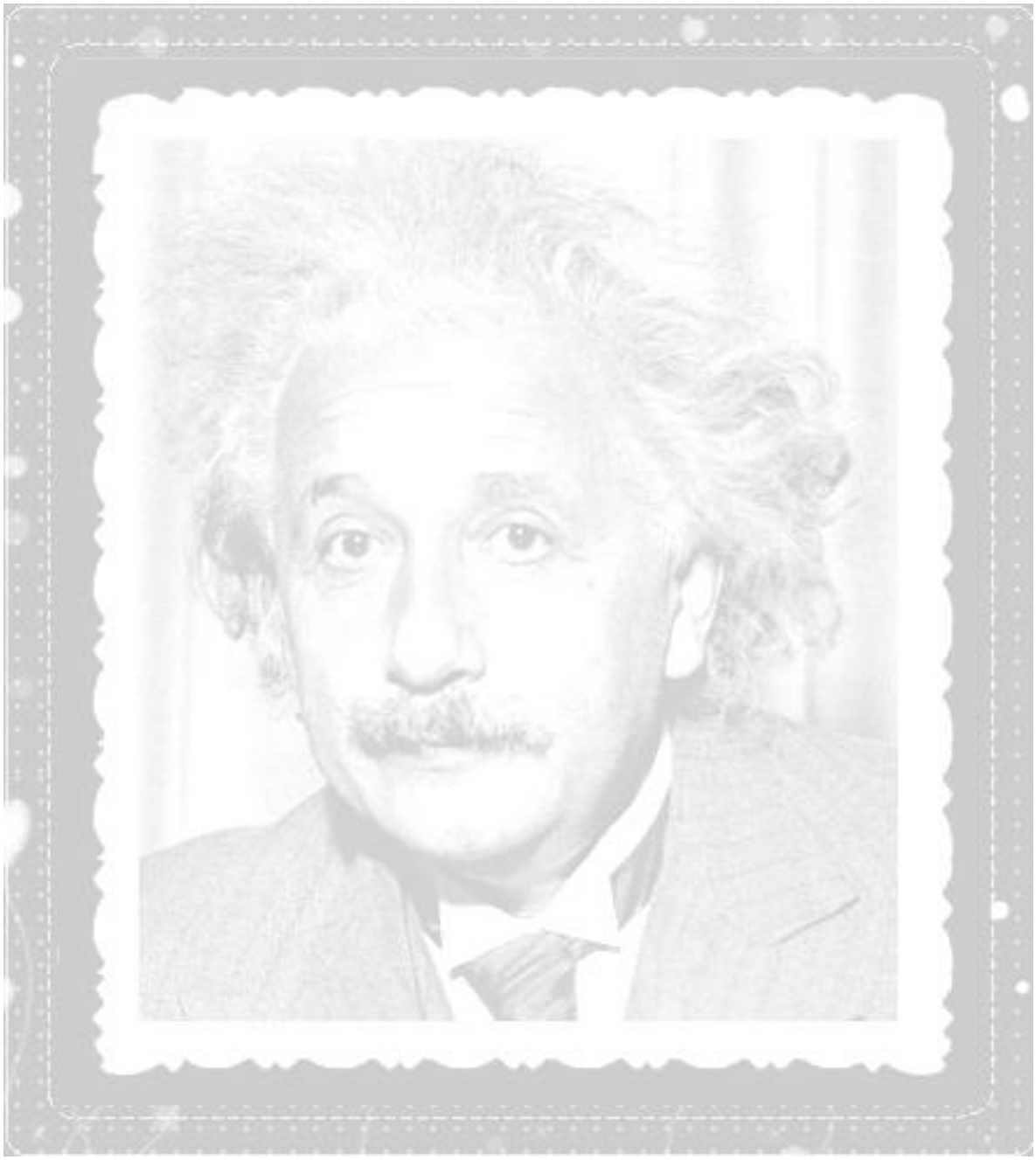
Ing. Annies Cedeño López

Tutora

_____

Msc. Rodolfo del C. Piedra Cabrera

Co-Tutor



Todos somos muy ignorantes. Lo que ocurre es que no todos ignoramos las mismas cosas.

--Albert Einstein--

AGRADECIMIENTOS

La realización de este trabajo fue gracias al apoyo y paciencia de muchas personas. Por lo mismo es posible que se me olvide nombrar algunas de ellas en estas líneas por lo que les pido disculpas.

Agradezco primeramente a la Universidad de las Ciencias Informáticas por darme la oportunidad de realizar uno de mis mayores sueños y es poder graduarme de Ingeniera de las Ciencias Informáticas.

A mi mamá que siempre estuvo ahí para mí en las buenas y en las malas, que nunca dejó que me rindiera por muy duras que fueran las pruebas o los seminarios, siempre estuvo apoyándome y dándome aliento para que saliera bien.

A mis profesores porque sin ellos no hubiera aprendido a estudiar y a ser un profesional, gracias por compartir sus experiencias con nosotros a lo largo de los años.

A mis amistades que gracias a ellas todos pudimos salir adelante ayudándonos unos a otros para que todo salga bien, entre ellas a Oniel, muchas gracias por la ayuda que me diste, a la gente de mi grupo, a todos aquellos que alguna vez me preguntaron cómo me iba en la tesis, a los que me han ayudado de una forma u otra a realizar este sueño....

Muchas Gracias.

DEDICATORIA

A mi mamá por hacer de mi lo que soy. No hay una sola línea de este trabajo donde no esté presente. Nunca me alcanzará el tiempo para hacer por ella lo que ha hecho por mí. Siempre ha luchado porque yo tenga una carrera y ha confiado ciegamente en todo lo que he hecho y hacer que se sienta orgullosa ha sido el camino que he tomado desde que tuve conciencia de ello. Es la mejor madre del mundo y sabe que nunca le fallaré. Vive muchos años más mamá que aún te necesito.

A mis abuelos que aunque viven lejos de mi no pasa un momento en el que no me acuerde de ellos, ellos me dan la fuerza que necesito para seguir adelante con todo lo que me propongo, me han demostrado su apoyo siempre y eso muy importante para mi.

A mi novio Yordan que ha estado presente en estos últimos años ayudándome, apoyándome, enseñándome a ser mejor cada día dándome su amor y dedicación en todo momento.

A mis amigo(a)s en especial a aquellos con los que compartí 4 años de la carrera, los cuales han sido los mejores de mi vida, no importa la distancia que nos separe siempre seremos el grupo 6, también a aquellos que no se graduaron con nosotros les dedico este trabajo ya que al entrar a la Universidad todos teníamos el mismo sueño, graduarnos.

Yisel Melendez Puentes

RESUMEN

El ministerio de comercio exterior y la inversión extranjera de Cuba cuenta con un incipiente nivel de informatización en los procesos que desarrolla cada dirección que lo conforma. Una de ellas es la dirección de importaciones (DI). En dicha dirección existen problemas para el correcto acceso, persistencia e integridad de la información con que se trabaja, por lo que el presente trabajo de diploma tiene como objetivo desarrollar una base de datos y la capa de acceso a datos que garanticen el acceso, persistencia e integridad de la información para el módulo de la DI del MINCEX.

Para el desarrollo de la base de datos y de la capa de acceso a datos se utilizó postgresql como sistema gestor de la base de datos. Se empleó la tecnología hibernate-spring para el acceso a datos, con el mapeo de las tablas de la BD utilizando el patrón de diseño DAO; e hibernate como motor de persistencia para el procesamiento de datos en la capa de acceso a estos. SXP como metodología de desarrollo de software para proyectos con pequeños equipos de trabajo. Se realizó el diagrama entidad-relación correspondiente a las clases persistentes generando el script físico de las tablas.

Una vez terminada la etapa de implementación se tuvo en cuenta la integridad de la información, mediante los métodos utilizados para la validación teórica y funcional de la solución propuesta, verificándose si la información se puede acceder correctamente, si es persistida e íntegra.

Palabras Claves

Base de datos, capa de acceso a datos, sistema gestor de base de datos (SGBD), hibernate.

Índice

Índice

INTRODUCCIÓN

Desde el principio de la civilización se ha recorrido la importancia de organizar y almacenar grandes cantidades de información para después recuperarla en forma eficaz, por lo que surgen primeramente los sistemas de archivos, de la necesidad de reemplazar el manejo de los archivos manuales para obtener acceso a los datos con mayor rapidez.

Estos sistemas también presentaron algunos inconvenientes que se atribuyen a que la definición de los datos se encuentra codificada dentro de los programas de aplicación, y no siendo almacenada de forma independiente. Debido a los inconvenientes que presentaban los sistemas de archivos surgieron las bases de datos y los sistemas gestores de bases de datos (SGBD). Sobre todo, desde la aparición de las primeras computadoras, el concepto de bases de datos ha estado siempre ligado a la informática.

Cada día son más las empresas conscientes del hecho, de que gran parte de su éxito y estabilidad depende de la disponibilidad y seguridad de todas sus informaciones y datos, tanto en la rapidez del acceso como su fiabilidad. Este almacenamiento en las empresas se ha convertido en un proceso estratégico de cara a sus competidores.

El hecho de poder compartir la información existente entre todos los departamentos de una misma organización pudiendo limitar accesos determinados en función de la responsabilidad dentro de la misma, favorece una mayor comunicación que lleva consigo un incremento del rendimiento laboral, pudiendo realizar el mismo trabajo de una forma más sencilla, rápida y eficaz, traduciéndose en un ahorro sensible de costos y agilidad en las acciones. Esto mejora ostensiblemente la relación con el cliente ya que se pueden ofrecer respuestas a sus necesidades más inmediatas.

De forma unísona con el desarrollo y optimización de las tecnologías de la información fue surgiendo la necesidad de persistir y gestionar los datos que se capturaban o generaban mediante software o aplicaciones que se hacían más

potentes debido a dichos avances.

Cuba no ha estado ajena a estos intereses ya que desde los inicios de la revolución, su dirección, encabezada por su líder se propuso un proceso de transformaciones educacionales y sociales que ha tenido como colofón el programa de la batalla de ideas, a partir del cual se emprendieron y se llevan a cabo nuevos programas destinados a elevar el nivel cultural de la población y su calidad de vida.

En estas circunstancias surge como una idea del comandante en jefe Fidel Castro Ruz la creación de la Universidad de Ciencias Informáticas (UCI), la cual trabaja para convertirse en un centro de excelencia, de dicha institución se derivaron tres facultades regionales siendo una de ella la Facultad Regional Mártires de Artemisa donde existe una fuerte infraestructura tecnológica, informática y telemática para la formación de profesionales altamente calificados y la producción de aplicaciones informáticas y donde se desarrolla una aplicación que mejorará el rendimiento de la administración de datos en la dirección de importaciones del MINCEX.

Con el incremento del uso de las tecnologías de la información y las comunicaciones (TIC) la automatización de los procesos que se realiza es cada vez mayor permitiendo ejecutar las actividades en menor tiempo y aportando mayor productividad, gracias a esto es posible un incremento en la calidad del trabajo realizado.

No está exento de estos problemas el MINCEX en Cuba, el cual se crea el 2 de marzo del 2009, mediante el decreto ley No. 264. Este ministerio tiene como objetivo proponer y preparar la política integral del estado y del gobierno en cuanto a la actividad de comercio exterior, la creación de empresas mixtas, la colaboración económica con otros países, organizaciones y asociaciones extranjeras y las inversiones que se negocien.

A pesar del impulso en las últimas décadas de las TIC, tanto en el mundo como en Cuba se ha hecho necesaria la creación de bases de datos para mantener segura y organizada toda la información que se genera cada día. El MINCEX siendo un ministerio que cuenta con un incipiente nivel de informatización en los procesos que

desarrolla cada dirección que lo conforma no se encuentra ajeno a esta necesidad.

La presente institución cuenta con 23 direcciones, siendo la dirección de importaciones (DI) una de ellas, la misma fue creada en el año 1995. Esta dirección existió en el antiguo MINCEX, permitiéndole a las entidades realizar actividades de importación, garantizando los recursos y servicios necesarios demandados por la economía y el mercado nacional, en condiciones que resulten económicamente ventajosas, con la inmediatez requerida y la calidad exigida.

A esta dirección le corresponde el control de la ejecución del plan de importaciones tanto de las empresas del sistema del MINCEX, como de la rama del comercio exterior, teniendo participación también en los aspectos metodológicos y de lineamientos de dicho plan. Son tareas importantes a controlar, las referidas a concentración de compras, comité de productos, comité de contratación, cartera de proveedores, atención a programas priorizados de la economía, desarrollo de negociaciones importantes; así como la elaboración de directivas, metodologías y dictámenes sobre cualquier aspecto relacionado con las importaciones que se soliciten.

En la DI existen problemas para acceder a las informaciones con que se trabaja diariamente ya que se encuentra almacenada en lugares distintos como repositorios, discos duros de las computadoras (PC), donde, si no se encuentra la persona autorizada a entrar en la PC no se puede utilizar, se puede encontrar además en formato impreso lo que dificulta la búsqueda, ya que pueden existir datos duplicados y estos son de vital importancia para las importaciones de productos.

La integridad de la información se ve comprometida a la hora de realizar el perfil de una empresa que opte por ser proveedor de productos del país, ya que si se duplicaran o extraviaran estos datos sería difícil controlar los proveedores con que se realizan las importaciones.

En dicha dirección los documentos de antigüedad como los reportes inmediatos y los informes mensuales entregado por las empresas importadoras, son guardados

en viejos archivos, lo que provoca que se extravíen por causas ajenas al personal que trabaja en la DI como ciclones, inundaciones o incendios.

Todo esto ocasiona pérdida de la información de los productos y los clientes con los que se realizan contratos, llevando a la falta de organización, poco acceso a la información y problemas para la gestión de los documentos presentados por las empresas importadoras.

De lo descrito anteriormente se plantea el siguiente **problema de la investigación**: ¿Cómo garantizar el acceso, persistencia e integridad de los datos en la DI del MINCEX?

Objeto de estudio: Almacenamiento y procesamiento de la información.

Campo de acción: Almacenamiento y procesamiento de la información en base de datos relacionales.

El **objetivo general de la investigación es:** Desarrollar la base de datos y la capa de acceso a datos que garanticen el acceso, persistencia e integridad de la información para el módulo de la DI del MINCEX.

Preguntas científicas

- ¿Cuáles son los fundamentos teóricos del proceso de información en bases de datos relacionales?
- ¿Cuáles son los medios a utilizar para lograr un desarrollo óptimo de la solución?
- ¿Cómo modelar la base de datos y la capa de acceso a datos del módulo de la DI del MINCEX?
- ¿Cómo implementar y validar la capa de acceso a datos para asegurar la persistencia, la integridad y el acceso a los datos en la DI del MINCEX?

Para dar cumplimiento al objetivo propuesto se acometen las siguientes **tareas de la investigación:**

1. Definición del marco teórico y estudio del estado del arte de la investigación.
2. Selección de la metodología, las herramientas y tecnologías a utilizar para el

desarrollo de la solución.

3. Diseño de la base de datos para el módulo de la DI del MINCEX.

4. Implementación y validación a través de pruebas funcionales a la capa de acceso a datos.

Variables de la investigación.

Variables independientes.

- Base de datos.
- Capa de acceso a datos.

Estas variables son independientes porque son las que se manipulan para conocer su relación con las variables dependientes, además la ausencia de estas es la causa de que exista desorden en los documentos con que se trabaja en la DI del MINCEX.

Variables dependientes.

- Persistencia de los datos
- Integridad de los datos

Las variables mencionadas anteriormente son dependientes ya que estas representan las características de los problemas encontrados en la DI.

Los métodos empleados en la investigación son:

Métodos teóricos:

- **Método análisis histórico – lógico:** Este método se utiliza para realizar un análisis histórico sobre los sistemas de gestión de base de datos en sistemas de información en el mundo, así como las herramientas y tecnologías más factibles a usar en la presente investigación.
- **Método analítico-sintético:** Es necesario la utilización de dicho método porque a través de este se hace un análisis de las principales herramientas a utilizar en la investigación, así como de los sistemas de gestión de bases de datos en el mundo.
- **Método modelación:** El presente método es uno de los más significativos en el

desarrollo de un software informático ya que haciendo uso del mismo se crea el modelo de datos de la base de datos de la DI del MINCEX.

Métodos empíricos:

- **Método observación:** Registro visual de la investigación que ofrece la posibilidad de poder identificar los problemas existentes en la DI del MINCEX.
- **Método entrevista:** El empleo de este método sirvió para conocer la opinión de los principales asesores y especialistas sobre los aspectos relativos al uso y acceso a la información y su necesidad en la DI del MINCEX. (Ver anexos).

Definición de la población y muestra.

Población:

- Los 10 especialistas que trabajan en la DI del MINCEX, incluyendo el director y el subdirector.

Muestra:

- La muestra que se tomó para realizar la encuesta fueron 4 especialistas, siendo uno de ellos el director que representan un 40% de la población.

Aporte práctico:

Consiste en la creación de una base de datos y de la capa de acceso a estos para la dirección de importaciones del MINCEX.

Estructura de la tesis.

Introducción: En la misma se presenta y argumenta la necesidad social del tema a tratar así como el diseño teórico-metodológico de la investigación.

Capítulo 1: Fundamentación teórica: En este capítulo se realiza una fundamentación teórica acerca del desarrollo y evolución de los sistemas gestores de bases de datos y sus principales conceptos; teniendo en cuenta las tecnologías y herramientas más usadas y las posibles a utilizar, así como la justificación de las mismas.

Capítulo 2: Diseño y arquitectura de la capa de acceso a datos: En este se describen los patrones de diseño a utilizar y se definen los artefactos

correspondientes al rol diseñador de base de datos, dígase artefactos, modelo entidad relación, modelo relacional. Además, se verán las reglas de transformación del modelo entidad relación al modelo relacional.

Capítulo 3: Implementación y validación de la capa de acceso a datos: En este se lleva a cabo la **i**mplementación de los artefactos del diseño de la base de datos e integración con la aplicación en la capa de persistencia y se valida mediante el empleo de validaciones funcionales.

Por último se hacen las conclusiones, las recomendaciones y se da a conocer las fuentes bibliográficas, la bibliografía utilizada y los anexos.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA.

1.1. Introducción.

Con el desarrollo de este capítulo se realiza un estudio detallado de los principales conceptos y tecnologías utilizadas en el desarrollo del módulo de la DI del MINCEX, donde se abordan temas relacionados con las bases de datos (BD), el uso de sistemas de gestión de bases de datos (SGBD), las metodologías empleadas para el análisis, diseño e implementación de bases de datos, así como aspectos relacionados con el uso de las nuevas tecnologías de la información y las comunicaciones, la persistencia y acceso de la información.

1.2. Tipos de bases de datos.

Tipos de bases de datos: [35]

a) Según la variabilidad de los datos

- Base de datos estática
- Base de datos dinámicas

b) Según el contenido

- Bases de datos bibliográficas
- Bases de datos de texto completo

c) Según el modelo de administración de datos

- Bases de datos jerárquicas
- Base de datos de red
- Base de datos relacional
- Bases de datos orientadas a objetos

Bases de datos estática.

Éstas son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

Bases de datos dinámicas.

Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un video club.

Bases de datos bibliográficas.

Solo contienen un surrogante (representante) de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque si no, se estaría en presencia de una base de datos a texto completo. Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio.

Bases de datos de texto completo.

Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científica. [32]

Bases de datos jerárquicas.

Esta base de datos tiene como objetivo establecer una jerarquía de fichas, de manera que cada ficha puede contener a su vez listas de otras fichas, y así sucesivamente.

Una base de datos jerárquica está compuesta por una secuencia de bases de datos físicas, de manera que cada base de datos física se compone de todas las ocurrencias[1] de un tipo de registro o ficha determinada.

Una ocurrencia de registro es una jerarquía de ocurrencias de segmento.

Cada ocurrencia de segmento está formada por un conjunto de ocurrencias o

[1] La ocurrencia son los datos que hay almacenados en el esquema en un determinado momento y que varían. [36]

instancias de los campos que componen el segmento. (Sergio, 2007)

Bases de datos en red.

Una base de datos en red consiste en un conjunto de registros conectados entre si mediante punteros. Los registros son en muchos aspectos parecidos a las entidades del modelo entidad-relación (E-R). Cada registro es un conjunto de campos (atributos), cada uno de los cuales sólo contiene un valor de datos. Los punteros son asociaciones entre exactamente dos registros. Por tanto, los punteros pueden considerarse una forma restringida (binaria) de relación en el sentido del modelo E-R. [33]

Bases de datos orientadas a objetos.

Desde la aparición de la programación orientada a objetos (POO u OOP) se empezó a pensar en bases de datos adaptadas a estos lenguajes. En estos lenguajes los datos y los procedimientos se almacenan juntos. Esta es la idea de las bases de datos orientadas a objetos.

A través de esta idea se intenta que estas bases de datos consigan arreglar las limitaciones de las relacionales. Por ejemplo el problema de la herencia, tipos definidos por el usuario, disparadores almacenables en la base de datos y soporte multimedia.

Se supone que son las bases de datos de tercera generación (la primera fue las bases de datos en red y la segunda las relacionales), lo que significa que el futuro parece estar a favor de estas bases de datos. Pero siguen sin reemplazar a las relacionales. (Jorge Sánchez, 2004).

Bases de datos relacionales.

Edgar Frank Codd definió las bases del modelo relacional a finales de los 60. En las bases de Codd se definían los objetivos de este modelo:

- Independencia física. La forma de almacenar los datos, no debe influir en su manipulación lógica.
- Independencia lógica. Las aplicaciones que utilizan la base de datos no deben ser modificadas por que se modifiquen elementos de la base de

datos.

- Flexibilidad. La base de datos ofrece fácilmente distintas vistas en función de los usuarios y aplicaciones.
- Uniformidad. Las estructuras lógicas siempre tienen una única forma conceptual (las tablas).
- Sencillez.

Entidades.

Representa una “cosa” u "objeto" del mundo real con existencia independiente, es decir, se diferencia unívocamente de otro objeto o cosa, incluso siendo del mismo tipo, o una misma entidad.

Una entidad puede ser un objeto con existencia física como: una persona, un animal, una casa (entidad concreta); o un objeto con existencia conceptual como: un puesto de trabajo, una asignatura de clases, un nombre (entidad abstracta).

Terminología relacional.

Tupla. Cada fila de la tabla o entidad (cada ejemplar que la tabla representa).

Atributo. Cada columna de la tabla o entidad.

Grado. Número de atributos de la tabla o entidad.

Cardinalidad. Número de tuplas de una tabla o entidad.

Dominio. Conjunto válido de valores representables por un atributo.

Dominios.

Los dominios suponen una gran mejora en este modelo ya que permiten especificar los posibles valores válidos para un atributo. Cada dominio incorpora su nombre y una definición del mismo. Ejemplos de dominio:

- Dirección: 50 caracteres
- Nacionalidad: Español, Francés, Italiano,...

Los dominios pueden ser también compuestos a partir de otros (año, mes y día = fecha). (Jorge Sánchez, 2004).

Atributos.

Los atributos son las características que definen o identifican a una entidad. Los

atributos son las propiedades que describen a cada entidad en un conjunto de entidades.

Para cada atributo, existe un dominio del mismo, éste hace referencia al tipo de datos que será almacenado o a restricciones en los valores que el atributo puede tomar (cadenas de caracteres, números, solo dos letras, solo números mayores que cero, solo números enteros...).

Cuando algún atributo correspondiente a una entidad no tiene un valor determinado recibe el valor nulo, bien sea porque no se conoce, porque no existe o porque no se sabe nada al respecto del mismo.

Relación.

Describe cierta dependencia entre entidades o permite la asociación de las mismas

Ejemplo:

"Dadas dos entidades "habitación 502" y "Mark Henry Jonshon Mcfly Bogard", es posible relacionar que la habitación 502 se encuentra ocupada por el huésped de nombre Mark."

Una relación tiene sentido al expresar las entidades que relaciona. En el ejemplo anterior, un huésped (entidad), se aloja (relación) en una habitación (entidad).

Restricciones.

Son reglas que deben mantener los datos almacenados en la base de datos.

Correspondencia de cardinalidades.

Dado un conjunto de relaciones en el que participan dos o más conjuntos de entidades, la correspondencia de cardinalidad indica el número de entidades con las que puede estar relacionada una entidad dada.

Dado un conjunto de relaciones binarias y los conjuntos de entidades A y B, la correspondencia de cardinalidades puede ser:

- **Uno a Uno:** Una entidad de A se relaciona únicamente con una entidad en B y viceversa.
- **Uno a varios:** Una entidad en A se relaciona con cero o muchas entidades en B. Pero una entidad en B se relaciona con una única entidad en A.
- **Varios a Uno:** Una entidad en A se relaciona exclusivamente con una

entidad en B. Pero una entidad en B se puede relacionar con 0 o muchas entidades en A.

- **Varios a Varios:** Una entidad en A se puede relacionar con 0 o muchas entidades en B y viceversa.

Claves.

Es un subconjunto del conjunto de atributos comunes en una colección de entidades, que permite identificar unívocamente cada una de las entidades pertenecientes a dicha colección. Asimismo, permiten distinguir entre sí las relaciones de un conjunto de relaciones.

- Clave primaria: Es una clave candidata, elegida por el diseñador de la base de datos, para identificar unívocamente las entidades en un conjunto de entidades.
- Clave ajena (foreign key o clave foránea): es aquella columna que existiendo como dependiente en una tabla, es a su vez clave primaria en otra tabla.

Tipo de base de datos seleccionado.

En forma de resumen se puede decir que el tipo de base de datos utilizado para la confección de la base de datos del módulo de la DI del MINCEX fue la base de datos relacional ya que utiliza un conjunto de tablas que están vinculadas entre sí. Puede reducir mucho la cantidad de datos que debe ingresar cada vez que se agrega un registro. Para un número grande de registros, una base de datos relacional puede buscar más rápido entre estos.

1.3. Definiciones según autores reconocidos.

Datos.

Según el criterio del Ing. Daniel Ambrosio:

- Los datos se caracterizan por no contener ninguna información. Un dato puede significar un número, una letra, un signo ortográfico o cualquier símbolo que represente una cantidad, una medida, una palabra o una descripción.

La importancia de los datos está en su capacidad de asociarse dentro de un

contexto para convertirse en información. Por si mismos los datos no tienen capacidad de comunicar un significado y por tanto no pueden afectar el comportamiento de quien los recibe. Para ser útiles, los datos deben convertirse en información para ofrecer un significado, conocimiento, ideas o conclusiones.

Según la definición brindada por la Lic. Luisa María González:

- Los datos son información individual que no tiene importancia en sí misma.

Profesor: Bachiller: Wilmer García

- Los datos son símbolos que describen condiciones, hechos, situaciones o valores.

Información.

La información no es un dato o conjunto cualquiera de ellos. Es más bien una colección de hechos significativos y pertinentes, para el organismo u organización que los percibe.

La definición de información según el Lic. Daniel Ambrosio es la siguiente:

- Información es un conjunto de datos significativos y pertinentes que describan sucesos o entidades.

Según el Lic. Victor Díaz Morales:

- La información es aquel conjunto de datos significativos para un usuario interesado y preparado para entenderlos. Los sistemas informáticos procesan datos cuya dimensión semántica es ajena. El usuario y el analista diseñador necesitan ver sentido (la información) en los datos almacenados y procesados.

Procesamiento de datos.

Según el criterio del Ing. Henry Stallman:

- Es la técnica que consiste en la recolección de los datos primarios de entrada, que son evaluados y ordenados, para obtener información útil, que luego serán analizados por el usuario final, para que pueda tomar las decisiones o realizar las acciones que estime conveniente.

Bases de datos.

Sobre las bases de datos existen diferentes criterios:

Según el criterio de Mato G. Lic. Rosa M:

- Una base de datos es un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una base de datos puede considerarse una colección de datos variables en el tiempo [4].

Según la definición dada por Date, Christopher J:

- Una base de datos se puede considerar como una especie de armario electrónico para archivar; es decir, es un depósito o contenedor de una colección de archivos de datos computarizados [5].

Según la opinión de Marqués A. María M:

- Un conjunto de datos almacenados entre los que existen relaciones lógicas y ha sido diseñada para satisfacer los requerimientos de información de una empresa u organización. En una base de datos, además de los datos, también se almacena su descripción [6].

Definición encontrada en la conferencia

Introducción_a_los_Sistemas_de_Bases_de_Datos. Pág: 2

- Una base de datos puede considerarse un conjunto de datos relacionados entre sí, entendiéndose por datos los hechos conocidos, que pueden registrarse y que tienen significado implícito**.** [7]

A modo general una base de datos es un conjunto de datos relacionados entre sí que comparten características similares, organizados y estructurados, los cuales son coleccionados y explotados por los sistemas de información de una empresa o negocio en particular.

Sistemas gestores de bases de datos (SGBD).

Por otro lado, existen varias definiciones sobre los SGBD, algunas se muestran a continuación:

Definición encontrada en la conferencia:

Introducción_a_los_Sistemas_de_Bases_de_Datos. Pág: 2

- Un sistema gestor o manejador de bases de datos es un conjunto de programas que permite a los usuarios crear y mantener una base de datos, por lo tanto, el SGBD es un software de propósito general que facilita el proceso de definir, construir y manipular la base de datos para diversas aplicaciones. Estos pueden ser de propósito general o específico. [7]

Según el criterio de Mato G. Lic. Rosa M:

- El software que permite la utilización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez es lo que se conoce como un sistema gestor de base de datos [4].

Según los criterios de los autores Korth, Henry F, Silberschatz y Abraham:

- Un sistema de gestión de bases de datos consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. El objetivo primordial de un SGBD es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información de la base de datos [8].

Según los criterios de los autores Marqués Andrés y María Mercedes:

- El sistema de gestión de la base de datos es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, y proporciona acceso controlado a la misma. [9].

Un SGBD no es más que un tipo de aplicación informática o software que actúa como interfaz entre el usuario y la base de datos. Funciona como un intermediario para que el usuario pueda comunicarse con la base de datos en términos abstractos y de una forma práctica y eficiente, abstrayéndolo del modo físico de almacenamiento de la información en la computadora, y del método de acceso empleado, permitiéndole definir, crear y mantener la base de datos y proporcionándole el acceso controlado a la misma.

1.4. Herramientas, tecnologías y metodologías a utilizar.

A continuación se realizará una breve definición acerca de las tecnologías, herramientas y metodologías a emplear en el diseño e implementación de la base de datos del módulo de la DI del MINCEX.

1.4.1. Sistemas gestores de bases de datos más utilizados.

Numerosas empresas se han volcado al desarrollo de SGBD como oracle, informix, postgresql, sybase, microsoft y existen tanto soluciones comerciales de pago, como soluciones de acceso libre.[19]

Existen diferentes tipos de gestores de bases de datos (GBD), cada uno es utilizado para el mismo propósito, gestionar bases de datos, pero no todos se pueden adquirir de la misma manera ya que existen GBD para software libre, para software propietario o para software comerciales.

A continuación algunos SGBD libres:

- Postgresql
- MySQL
- Firebird
- SQLite
- DB2 Express-C
- Apache Derby

SGBD gratuitos

- Microsoft SQL Server Compact Edition
- Sybase ASE Express Edition para Linux

SGBD comerciales

- dBase
- Fox Pro
- IBM DB2 Universal Database (DB2 UDB)
- IBM Informix
- Microsoft Access

- Microsoft SQL Server
- Oracle
- Paradox
- Sybase ASE
- Sybase ASA
- Sybase IQ

Los SGBD suelen incluir herramientas de administración que permiten ajustar el rendimiento en función de las necesidades particulares. Muchas empresas cuentan son sus propios administradores de bases de datos, pero también hay muchas otras que no, y lo más probable es que el diseñador web tenga que administrar también las bases de datos. [19]

Oracle.

Oracle es básicamente una herramienta cliente/servidor para la gestión de bases de datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hace que sólo se vea en empresas muy grandes y multinacionales. En el desarrollo de páginas web pasa lo mismo: como es un sistema muy caro no está tan extendido como otras bases de datos, como, access, mysql y sql server.

El SGBD oracle, ha incorporado en su sistema el modelo objeto-relacional, pero al mismo tiempo garantiza la compatibilidad con el tradicional modelo relacional de datos. Es uno de los más conocidos y ha alcanzado un buen nivel de madurez y de profesionalidad. Se destaca por su soporte de transacciones, estabilidad y escalabilidad. Los tipos objeto de oracle son tipos de datos definidos por el usuario que permiten modelar entidades complejas del mundo real en una estructura que trata cada entidad como una unidad atómica simple en la base de datos. [20]

Ventajas.

- Permite reutilizar objetos para el desarrollo de base de datos de una forma más rápida y con mayor eficiencia.
- Los programadores de aplicaciones pueden acceder directamente a tipos de

objetos oracle, sin necesidad de ninguna capa adicional entre la base de datos y la capa cliente.

- Las aplicaciones que utilizan objetos de oracle son fáciles de entender y mantener porque soportan las características del paradigma orientado a objetos.
- Tiene buen rendimiento y hace buen uso de los recursos. Posee un rico diccionario de datos.
- Brinda soporte a la mayoría de los lenguajes de programación.
- Es un sistema multiplataforma, disponible en windows, linux y unix.
- Permite tener copias de la base de datos productiva en lugares lejanos a la ubicación principal.
- Las copias de la base de datos productiva pueden estar en modo de lectura solamente. [20]

Desventajas.

Es un producto de elevado precio por lo que por lo general se utiliza en empresas muy grandes y multinacionales. Los costos de soporte técnico y mantenimiento son elevados. Vulnerabilidades en la seguridad de la plataforma, se hace necesario aplicar parches de seguridad. [20]

MySQL.

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL2 de la GNU3. Su diseño le permite soportar una gran carga de forma muy eficiente. Mysql fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor sql, así como también de la marca.[21]

Aunque MySQL es software libre, MySQL ab distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que

[2] La **licencia GPL** (General Public Licence) significa que el software debe permanecer perpetuamente libre, es decir que cualquiera puede utilizarlo, modificarlo y redistribuirlo, con la condición de que los cambios que haga también los ponga a disposición de quien lo requiera. [38]

[3] **GNU** es el conjunto de programas desarrollados por la Free Software Foundation (Fundación por el Software Libre) que son de uso libre. [39]

se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL. [21]

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. [21]

MySQL carece de algunas características, las cuales se presentan a continuación:

1. Subconsultas: Tal vez ésta sea una de las características que más se echan en falta, aunque gran parte de las veces que se necesitan, es posible reescribirlas de manera que no sean necesarias.
2. "Select into table": Esta característica propia de Oracle, todavía no está implementada.
3. Triggers y procedures: Se tiene pensado incluir el uso de procedures almacenados en la base de datos, pero no el de triggers, ya que los triggers reducen de forma significativa el rendimiento de la base de datos, incluso en aquellas consultas que no los activan.
4. Transacciones: A partir de las últimas versiones ya hay soporte para transacciones, aunque no por defecto (se ha de activar un modo especial).
5. Integridad referencial: Aunque sí que admite la declaración de claves ajenas en la creación de tablas, internamente no las trata de forma diferente al resto de campos.

Los desarrolladores comentan en la documentación que todas estas carencias no les resultaba un problema, ya que era lo que ellos necesitaban. De hecho, MySQL fue diseñada con estas características, debido a que lo que buscaban era un gestor de bases de datos con una gran rapidez de respuesta. Pero ha sido con la distribución de MySQL por internet, cuando más y más gentes les están pidiendo estas funcionalidades, por lo que serán incluidas en futuras versiones del gestor. [21]

Postgresql.

Postgresql aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arrays. [22]

Control de Concurrencia Multi-Versión (MVCC).

MVCC, es la tecnología que postgresql usa para evitar bloqueos innecesarios. Mediante el uso de MVCC, postgresql evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, postgresql mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. Postgresql es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles. [22]

Cliente/Servidor.

Postgresql usa una arquitectura proceso-por-usuario cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a postgresql. [22]

Postgresql es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Es más completo que MySQL ya que permite métodos almacenados, restricciones de integridad y vistas, aunque en las últimas versiones de MySQL se han hecho grandes avances en ese sentido. [16]

Las tablas con la información que contienen no son los únicos objetos que componen una BD y su existencia no puede estar aislada de otros elementos que aportan facilidades para el trabajo con las mismas.

El lenguaje pl/pgsql es uno de los lenguajes soportado por el SGBD postgresql y dentro de sus potencialidades se incluye el manejo de vistas, como vía para visualizar la información independientemente de la manera en que físicamente está almacenada para los diferentes tipos de usuario. Como la mayoría de los lenguajes

de programación que se conoce, es posible agrupar bloques de sentencias con una funcionalidad específica dentro de objetos denominados funciones, que tienen sus características y para el desarrollo de las cuales se utilizan estructuras y herramientas incluidas en el lenguaje.

SGBD utilizado:

Resumiendo, el SGBD usado para la confección de la base de datos relacional para el módulo de la DI del MINCEX fue postgresql porque está considerado como la base de datos de código abierto más avanzada del mundo. Proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como Oracle. Es más completo que MySQL ya que permite métodos almacenados, restricciones de integridad y vistas.

1.4.2. Herramientas para modelado de datos.

Las herramientas de modelado de datos son muy útiles a la hora de realizar un diseño de datos nuevo o para acercarse a la comprensión y estudio de un modelo existente (en aquellas con características de ingeniería inversa). La capacidad de realizar un diseño de modelo visualizándolo de forma gráfica, a través de los diagramas entidad-relación, o de obtener un diagrama y documentación de una base de datos existente aumenta enormemente la eficiencia del trabajo reduciendo drásticamente el tiempo necesario con respecto a una realización manual de dichas tareas de diseño, análisis y estudio.

Estudio de herramientas de modelado de datos.

Power designer.

La mejor y más completa herramienta de modelado que se conoce es power designer. No es objeto de la presente investigación describir la enorme cantidad de características que tiene, ni lo perfectamente resuelto que están en ella las tareas de modelado e ingeniería directa e inversa, pero es casi perfecta, contemplando todo el ciclo de modelado (lógico y físico, con multitud de bases de datos) de datos, además de otras tareas de modelado como el de objetos (con generación de código) y esquemas xml.

Power designer tiene entonces un (grave) inconveniente: sólo está disponible para Windows, y en sybase/sap no se han preocupado (al contrario que otras compañías, como por ejemplo spotify) de que pueda ejecutarse con wine perfectamente.

Existen herramientas que pueden servir para modelar una BD, pero son muchas y no daría tiempo probarlas. Así que, se plantearon una serie de características y condicionantes necesarios para el proyecto, con lo que el número de herramientas candidatas se redujo considerablemente.

La lista de características y condiciones ha sido la siguiente:

1. Debe estar disponible para Linux.
2. Debe tener características de modelado de datos (no basta con una herramienta de diagramas o que sea sólo para UML).
3. Debe tener características de ingeniería directa e inversa.
4. Debe soportar (para la ingeniería directa e inversa) como mínimo, postgresql.
5. Debe ser gratuita y libre.

A continuación, se detallan las herramientas analizadas:

Open system architect 4.0.0 (OSA).

De todas las herramientas analizadas es la que más se parece a power designer, y es la que más características y posibilidades incorpora. Es la única de las analizadas que integra el modelado lógico. Tiene la mejor organización de proyectos y menús y es rápida. Sin embargo también es la que más ha decepcionado en el resultado final. El acceso a las bases de datos es vía ODBC[4] (es la única de las analizadas que no está hecha en java), lo cual ha resultado un poco engorroso de configurar, aunque al final ese no ha sido el inconveniente, sino la falta de acabados finales que hacían perder la lucidez de las funcionalidades:

- Diagramas

[4] Open Data Base Connectivity (ODBC) es un estándar de acceso a bases de datos con el objetivo de hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos (DBMS) almacene los datos.

- Faltan opciones de presentación gráficas, de reordenado y de diseño de las tablas y sus relaciones.

- Ingeniería inversa:

  - La importación de tablas no es cómoda y no se sabe muy bien de qué esquemas se está importando. No se pueden agregar tablas referenciadas relacionadas (hijas) o que la relacionen (padres) automáticamente.

- Documentación:

  - No se encontró las características de documentación mínimas que se buscaba.

Sql power architect 0.9.13.

Aunque es un producto comercial, la versión community es código abierto y gratuita. La versión community gratuita es suficiente para un trabajo de modelado básico. Soporta ingeniería directa e inversa de forma cómoda (se pueden agregar fácilmente las tablas hijas) con bastantes opciones gráficas de presentación. Es rápido, amigable y con él se puede generar una documentación mínima.

El sql power architect es una herramienta de modelado de datos que fue creada por los diseñadores de almacenamiento de datos y tiene muchas características únicas dirigidas específicamente para el arquitecto de almacenamiento de datos. Permite a los usuarios de la herramienta ingeniería inversa de bases de datos existentes, realizar perfiles de datos en bases de datos de origen y generar automáticamente los metadatos de extract, transform and load (ETL, extracción, transformación y carga). [12]

A continuación algunas características del power architect:

- Permite acceder a las bases de datos a través de JDBC[5].
- Permite la conexión a múltiples bases de datos al mismo tiempo.

[5] **Java Database Connectivity (JDBC),**es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.[13]

- Compara modelos de datos y estructuras de bases de datos e identifica las discrepancias.
- Drag-and-drop[6] de las tablas origen y las columnas en el área de trabajo.
- Ingeniería directa/inversa para postgresql, Oracle, ms SQL server y muchas más.
- Todos los proyectos se guardan en formato XML.
- OLAP[7] modelos de esquema: cubos, medidas, dimensiones, jerarquías y niveles.

Squirrel SQL 3.2.1.

Esta herramienta es en realidad un cliente SQL multi-bases-de-datos con capacidades de modelado e ingeniería directa e inversa.

Squirrel no destaca por sus capacidades gráficas de diagramas entidad-relación. La organización de las tablas en una importación vía ingeniería inversa deja bastante que desear aunque salvando el detalle de tener que organizar y posicionar las tablas a mano, se tienen suficientes opciones para añadir tablas referenciadas o elegir la presentación de las tablas. Sin embargo, como se ha comentado, Squirrel es un cliente multipropósito y multibase bastante completo, con muchos plugins y potentes editores de datos, SQL, DML y esquemas. Este es probablemente el más completo multi-propósito de todos los analizados.

Druid III (3.11).

Druid es fundamentalmente una herramienta pensada para crear bases de datos de forma gráfica y documentarlas. Su capacidad de documentación de bases de datos supera a todas las del análisis: es capaz de generar documentación tipo javadoc de una base de datos con los diagramas entidad-relación y sus tablas de forma detallada.

---

6 **Arrastrar y soltar (drag and drop)** es una expresión informática que se refiere a la acción de mover con el ratón objetos de una ventana a otra o entre partes de una misma ventana.[14]

7 **OLAP** es el acrónimo en inglés de procesamiento analítico en línea **(On-Line AnalyticalProcessing)**. Es una solución utilizada en el campo de la llamada Inteligencia empresarial (o Business Intelligence) cuyo objetivo es agilizar la consulta de grandes cantidades de datos.[15]

Permite documentar una base de datos existente vía ingeniería inversa, realizando diagramas temáticos y realizando documentación adicional, de forma muy sencilla, añadiendo tablas relacionadas, y presentando la información que se desee. En todo caso, a pesar de que le faltan opciones gráficas y de tener una interfaz difícil, para documentar una base de datos vía ingeniería inversa está entre las mejores.

Después de un estudio realizado sobre las herramientas de modelado de datos, se concluye que:

- Todos tienen carencias en la presentación de los diagramas entidad-relación: faltan opciones de "layout[8]" y los diagramas son poco vistosos.
- Todos son multiplataforma (salvo open systems architect que está hecho en java).
- Todas las herramientas son válidas. Y se deberían probar todas. Ahora bien, cada una tiene un punto fuerte:
  - Documentación: Druid
  - Algo para empezar, sencillo: Sql power architect.
  - Cliente completo, todo en uno: Squirrel
  - OSA no ha gustado, a pesar de ser la única que incorpora modelado lógico de forma explícita.

Herramienta de modelado de datos seleccionada:

De todas las herramientas que aquí se mencionan el equipo de desarrollo se inclinó por power architect, éste, aunque es un producto comercial, la versión community es código abierto y gratuita. La versión community gratuita es suficiente para un trabajo de modelado básico. Soporta ingeniería directa e inversa de forma cómoda (se pueden agregar fácilmente las tablas hijas) con bastantes opciones gráficas de presentación. Es rápido, amigable y con él se puede generar una documentación mínima, por todo esto la herramienta cumple con las necesidades planteadas.

[8] **Layout**: suele utilizarse para nombrar al esquema de distribución de los elementos dentro un diseño [37]

1.4.3. Metodologías de desarrollo de software.

Dentro de la ingeniería del software (IS) se propone el uso de metodologías que definan y especifiquen el proceso de desarrollo de un producto, con el propósito de obtener resultados que garanticen y cumplan los requerimientos impuestos tanto por calidad como por los clientes, además de que se desarrollen en el tiempo estimado y los costos presupuestados. Por otra parte las metodologías de desarrollo de software, se pueden resumir como un conjunto de procedimientos, técnicas y herramientas que ayudan a los desarrolladores a realizar un nuevo producto. [10]

Metodología XP.

XP es la metodología candidata para guiar el proceso ingenieril, puesto que le precede su alto grado de aceptación por la comunidad internacional de desarrollo ágil, además que facilita una documentación más discreta y mayor dinamismo para el desarrollo.

Básicamente la idea de la programación extrema consiste en trabajar estrechamente con el cliente, haciéndole mini-versiones con mucha frecuencia. En cada mini-versión se debe hacer el mínimo de código y lo más simple posible para que funcione correctamente. El diseño se hace sobre la marcha, haciendo un mini-diseño para la primera mini-versión y luego modificándolo en las siguientes mini-versiones. Además, no hay que hacer una documentación para el diseño, no hay mejor documentación que el mismo código. El código, por tanto, también se modifica continuamente de mini-versión en mini-versión, añadiéndole funcionalidad y extrayendo sus partes comunes. [41]

Metodología SCRUM.

SCRUM es una metodología ideal para toda la gestión de proyectos y sirve de soporte para acelerar el dinamismo que se identificó en XP.

Más que una metodología de desarrollo software, es una forma de auto-gestión de los equipos de programadores. Un grupo de programadores deciden cómo hacer sus tareas y cuánto van a tardar en ello. Scrum ayuda a que trabajen todos juntos,

en la misma dirección, con un objetivo claro. [40]

Metodología SXP.

SXP es un híbrido cubano que toma lo mejor de XP y SCRUM, este ofrece una estrategia tecnológica a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo. Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto. Basada completamente en los valores y principios de las metodologías ágiles expuestos en el manifiesto ágil. Como método de estimación se utiliza la opinión de expertos y constan con métricas o indicadores para lograr una eficiente calidad. [11]

SXP ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro, y permite además seguir de forma clara el avance del equipo de desarrollo por parte del cliente, de forma que los jefes pueden ver día a día cómo progresa el trabajo. [11]

Metodología seleccionada:

La metodología que el equipo de desarrollo seleccionó fue la SXP ya que está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad.

1.4.4. Motores de persistencia.

Para entender que es un motor de persistencia primeramente se plantea el problema de hacer que dos componentes con formatos de datos muy diferentes puedan comunicarse y trabajar conjuntamente. Se trata de hacer que dos personas que hablan idiomas diferentes se comprendan.

La solución es la misma que se daría en la vida real. Se debe encontrar un traductor que sepa traducir de cada idioma al otro. De esta forma, las dos personas

se entenderán sin necesidad de que uno hable el idioma del otro. En el mundo de la programación este traductor no es más que un componente de software (concretamente, una capa de programación), al que se le dan los nombres de “capa de persistencia”, “capa de datos”, “correspondencia O/R” (“OR mapping”) o “motor de persistencia”, en otras palabras, un motor de persistencia es un componente software encargado de traducir entre objetos (de un programa orientado a objetos) y registros (de la base de datos relacional). Es decir es el encargado de que el programa y la base de datos se “entiendan”.

Ventajas:

- Se puede programar con orientación a objetos, aprovechando las ventajas de flexibilidad, mantenimiento y reusabilidad.
- Se puede usar una base de datos relacional, aprovechando de su madurez y su estandarización así como de las herramientas relacionales que hay para ella.

Se calcula que un motor de persistencia puede reducir el código de una aplicación en un 40%, haciéndola menos costosa de desarrollar. Además, el código que se obtiene programando de esta manera es más limpio y sencillo y, por lo tanto, más fácil de mantener y más robusto. Por añadidura, el motor de persistencia no sólo simplifica la programación, sino que permite hacer ciertas optimizaciones de rendimiento que serían difíciles de programar por nosotros mismos. [28]

Opciones para motores de persistencia.

A continuación, se explican algunos de los motores de persistencia más importantes para la plataforma java y para la plataforma .net, con el fin de que se pueda comenzar una investigación que lleve a escoger el que más se ajuste a las necesidades del proyecto.

En cuanto a la plataforma java, los servidores de aplicaciones basados en la especificación EJB (“enterprise javabeans”), incorporan un motor de persistencia a través del mecanismo conocido como “entitybeans”.

Entre los de código abierto se destaca: hibernate, castor, torque, ojb y cayenne, entre los comerciales, se destacan toplink, cocobase y fastobjects. En los últimos

años se ha creado una especificación llamada jdo, para estandarizar la forma de programar en java con esos motores de persistencia. Ejemplos de motores de persistencia que cumplen el estándar jdo son kodo, jdo genie, lido, exadel jdo, intellibo, jrelay jdo (todos ellos comerciales), tjdo y xorm (de código abierto). [28]

Hibernate 3.0.2

Hibernate por su parte es una herramienta de mapeo objeto-relacional (ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans9 de las entidades que permiten establecer estas relaciones. [18]

Características de Hibernate:

Como todas las herramientas de su tipo, hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información hibernate le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la programación orientada a objetos (POO). Hibernate convertirá los datos entre los tipos utilizados por java y los definidos por SQL. Hibernate genera las sentencias SQL[10] y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

Hibernate ofrece también un lenguaje de consulta de datos llamado HQL (hibernate query language), al mismo tiempo que una API[11] para construir las consultas

---

9 Los **JavaBeans** son un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java.[17]

10 "SQL. **lenguaje de consulta estructurado** o **SQL** (por sus siglas en inglés structured query language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas.)

11 **Interfaz de programación de aplicaciones** o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser

programáticamente (conocida como ″criteria″).

Hibernate proporciona grandes beneficios como:

- soporte a múltiples motores de base de datos.
- bajo acoplamiento entre negocio y persistencia, ya que su diseño está orientado a objetos así como el soporte a consultas y operaciones (HQL).
- desarrollo robusto, ya que el framework ha madurado tras años de uso en decenas de miles de proyectos.
- optimizado, ya que el SQL generado contiene optimizaciones específicas para cada motor de base de datos mediante componentes especializados llamados dialectos.
- rápido y completo, ya que con la funcionalidad estándar de hibernate se podrá cubrir el 80 - 90% de la persistencia de nuestra aplicación.

Todo esto permite centrar los esfuerzos en desarrollar la funcionalidad de la aplicación.

Motor de persistencia seleccionado:

El equipo de desarrollo del proyecto decide seleccionar a Hibernate como motor de persistencia porque está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.

1.4.5. Persistencia en bases de datos.

Se entiende por persistencia como la acción de preservar la información de un objeto de forma permanente (guardar), pero a su vez también se refiere a poder recuperar la información del mismo (leer) para que pueda ser nuevamente utilizada. En el caso de persistencia de objetos la información que persiste en la mayoría de los casos son los valores que contienen los atributos en ese momento, no necesariamente la funcionalidad que proveen sus métodos. [29]

utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas (también denominadas comúnmente "librerías").

La persistencia no es ni una capacidad ni una propiedad de la programación orientada a objetos (POO) no tiene nada que ver con el paradigma en sí, solo es el mecanismo que se usa para persistir información de un determinado tipo (como puede ser serializar, guardar los datos en una tabla, en un archivo plano).[29]

De forma sencilla puede entenderse que los datos tienen una duración efímera, desde el momento en que estos cambian de valor se considera que no hay persistencia de los mismos. Sin embargo en informática hay varios ámbitos donde se aplica y se entiende la persistencia.

La persistencia en base de datos relacionales se suele implementar mediante el desarrollo de funcionalidad específica utilizando la tecnología JDBC o mediante frameworks[12] que automatizan el proceso a partir de mapeos (conocidos como Object Relational Mapping, ORM) como es el caso de Hibernate. [34] (Javier Antoniucci, 2007).

1.4.6. Spring framework.

Spring Framework es una plataforma que proporciona una infraestructura que actúa de soporte para desarrollar aplicaciones Java. Spring maneja toda la infraestructura y así se puede centrar en la aplicación. Diciéndolo mas coloquialmente, Spring es el "pegamento" que une todos los componentes de la aplicación, maneja su ciclo de vida y la interacción entre ellos.

La primera versión de Spring se lanzó en junio de 2003, aunque el gran lanzamiento se hizo en Marzo de 2004, con la versión 1.0. Meses más tarde, en concreto el 21 de Junio de 2004, Rod Johnson, creador de Spring, publicó el libro: "J2EE development without EJB". [42]

Módulos.

Spring es bastante grande, por ello el proyecto esta dividido en módulos. No siempre se utiliza en un proyecto todo lo que tiene spring. Por poner un ejemplo, se

[12] Un **framework** es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado.

podría utilizar Struts13 para la parte web, en vez de Spring Modelo-Vista-Controlador (MVC). Si se utiliza un framework de persistencia, como Hibernate o iBatis, se tendría que incluir la librería spring-orm en la variable de entorno. (Ver Fig.). [42]

En lugar de proporcionar su propio módulo ORM (Object-Relational Mapping), para los usuarios que no se sientan confiados en utilizar simplemente JDBC, Spring propone un módulo que soporta los frameworks ORM más populares del mercado, entre ellos:

- Hibernate (2.1 y 3.0) – es una herramienta de mapeo O/R OpenSource muy popular que utiliza su propio lenguaje de query (consulta) llamada HQL.
- iBATIS SQL Maps (1.3 y 2.0) – es una solución sencilla pero poderosa para hacer externas las declaraciones de SQL en archivos XML.
- Apache OJB (ObJect Relational Bridge, 1.0) – es una plataforma de mapeo O/R con múltiples APIs para clientes.
- Otros como JDO (Java Data Objects, 1.0 y 2.0) y Oracle TopLink.

Todo esto se puede utilizar en conjunto con las transacciones estándar del framework. Spring e Hibernate es una combinación muy popular. [42]

1.4.7. JUnit framework.

JUnit es un framework para automatizar las pruebas unitarias de aplicaciones Java. Se utiliza en la fase de desarrollo, su utilización por parte de los desarrolladores permite la creación de software de mayor calidad.

JUnit es un conjunto de clases que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera.

Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado, si la clase cumple la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba. En caso de que el valor esperado

[13] **Struts** es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC.

sea diferente al retornado por el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

JUnit es también un medio para controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple los requisitos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

Este framework se encuentra en la versión 4.5 con algunas mejoras:

- Configuración pre y post prueba.
- Verificación de valores esperados.
- Especificación de tiempo de espera por cada test.

JUnit es una herramienta código abierto (open source) para la realización de test unitarios en la fase de desarrollo para el entorno Java.

Es la herramienta de test más utilizada en entornos Java, lo suficientemente madura para su implantación en entornos comerciales y con un coste cero.

Por todo ello, los principales IDEs de desarrollo (Eclipse, NetBeans, RAD) incorporan plugins para su utilización.

Además es complementaria con otras herramientas de pruebas como Cobertura, el cual facilita la creación de informes HTML y XML de cobertura de las pruebas realizadas mediante JUnit (% de código probado, % métodos correctos, errores y líneas de código probadas) en un proyecto. [31]

1.4.8. NetBeans IDE.

NetBeans IDE es una aplicación de código abierto diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java.

NetBeans IDE dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis y por si fuera poco sus funcionalidades son ampliables mediante la

instalación de packs.

En NetBeans IDE se encontrará la solución más completa para programar en Java. [43]

NetBeans IDE es fácil de usar y se ejecuta en muchas plataformas, incluyendo Windows, Linux, Mac OS X y Solaris.

Mejoras de la versión 7.0.1.

- La certificación completa en la versión final del JDK 7.
- Paquete de actualización de GlassFish 3.1.1.
- HTML5 soporte de edición.
- Compatible con Maven 3.
- Tecnologías soportadas: Java EE 6, Java EE 5 y J2EE 1.4, Spring 3.0, 2.5, Hibernate 3.2.5, Apache Maven 3.0.3 o posterior. [44]

1.5. Conclusiones parciales.

En este capítulo se hizo un estudio de las tecnologías a utilizar en el desarrollo de la propuesta de solución, así como algunos conceptos y tendencias que se deben tener en cuenta. Se fundamentó la selección de las herramientas escogidas para la solución. Dichas herramientas son: postgresql como SGBD, SQL power architect para el modelado de datos y para la manipulación de modelos de datos llevados a la POO, hibernate como motor de persistencia para la comunicación entre la aplicación y la base de datos y por último se utiliza la metodología ágil de desarrollo de software SXP.

CAPÍTULO II: DISEÑO Y ARQUITECTURA DE LA CAPA DE ACCESO A DATOS

2.1. Introducción.

En este capítulo se abordan las principales características del módulo, se describe el objeto de estudio en el cual se llevan a cabo los procesos involucrados en el campo de acción y el desarrollo de la propuesta de solución, se definen los patrones de diseño a utilizar y el modelo entidad relación, modelo relacional, los requisitos funcionales de la base de datos, así como la descripción de las tablas. Además, se verán las reglas de transformación del modelo entidad relación al modelo relacional y el modelo de implementación para desarrollar la capa de acceso a datos del módulo de la DI del MINCEX.

2.2. Procesos que serán objeto de automatización.

Con el presente trabajo se pretende desarrollar un módulo capaz de garantizar el acceso, la persistencia y la integridad de la información de todos los procesos que se realizan en la dirección de Importaciones del MINCEX. Entre estos procesos se encuentran:

- **Comité de productos:** El comité de productos son contratos que se realizan con empresas importadoras. Cada empresa envía mensualmente una tabla donde se ofrece el cliente nacional, el producto, entre otras informaciones. Se envía también por parte de cada empresa el cierre anual.
- **Control de proveedores**: Se lleva el control de la cartera de proveedores del sistema MINCEX. Esta cartera de proveedores está formada por el conjunto de firmas que producen o comercializan los diferentes productos que compra o comercializa la entidad, hayan tenido o no vínculo con la misma. Es aprobada por el consejo de dirección de la empresa, al igual que cualquier nueva inclusión, con el visto bueno del director que atiende la actividad del comercio exterior en el organismo.

- **Comité de contratación**: El comité de contratación es el encargado de evaluar y aprobar, previa a su formalización por las entidades, las operaciones comerciales de importación de mercancías, cuyos importes sean superiores al equivalente a doscientos cincuenta mil pesos cubanos convertibles.
- **Control del plan de importaciones**: Llevar el control del plan de importaciones de los organismos autorizados a realizar comercio exterior.
- **Control de planes priorizados:** Llevar el control y resumen de los planes priorizados en las empresas.

2.3. Propuesta de solución.

Después de haber realizado un estudio y análisis de las dificultades existentes en la dirección de importaciones en sus niveles correspondientes con el manejo y la administración de toda la información que se necesita controlar, buscar y obtener; y teniendo en cuenta de que no existe un sistema anterior que de cumplimiento total a lo requerido, y que cubra en su totalidad todas las necesidades, se propone desarrollar una base de datos y su capa de acceso a datos que den cumplimiento a los problemas encontrados en la DI y que garantice la persistencia de los datos así como la accesibilidad e integridad de los mismos.

La capa de acceso a datos contribuirá a mejorar la gestión de procesos como el control de los proveedores que comercializan los diferentes productos que compra la entidad, el control de los planes de importaciones de los organismos autorizados a realizar comercio exterior, también gestionará la información del comité de contratación el cual evalúa y aprueba las operaciones comerciales de importación de mercancías así como la organización de los contratos que se realizan con empresas importadoras.

Se espera que la base de datos y la capa de acceso a datos de la DI del MINCEX centralicen la información en esta dirección y facilite su búsqueda y almacenamiento.

2.4. Especificación de los requisitos de la base de datos.

2.4.1. Requisitos funcionales.

Los requisitos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir, más específicamente son acciones que el sistema debe ser capaz de realizar.

A continuación se listan las principales funcionalidades que deben tener la base de datos y la capa de acceso a datos de la dirección de importaciones del MINCEX:

Prioridad muy alta

RF1: Gestionar control del plan de importaciones. La BD debe permitir gestionar los datos del plan de importaciones.

RF1.1: Insertar control del plan de importaciones: la BD debe permitir crear un nuevo plan de importaciones.

RF1.2: Modificar control del plan de importaciones: la BD debe permitir modificar los datos del plan de importaciones.

RF1.3 Eliminar control del plan de importaciones: la BD debe permitir eliminar un plan de importaciones.

RF2: Gestionar control de planes priorizados: la BD debe permitir gestionar los datos de los planes priorizados.

RF2.1: Insertar control de planes priorizados: la BD debe permitir crear un nuevo plan priorizado.

RF2.2: Modificar control de planes priorizados: la BD debe permitir modificar los datos del plan priorizado.

RF2.3 Eliminar control de planes priorizados: la BD debe permitir eliminar un plan priorizado.

RF4: Gestionar control de proveedores : la BD debe permitir gestionar los datos de los proveedores del sistema.

RF4.1: Insertar control de proveedores: la BD debe permitir crear un nuevo proveedor.

RF4.2: Modificar control de proveedores: la BD debe permitir modificar los datos del proveedor.

RF4.3 Eliminar control de proveedores: la BD debe permitir eliminar un proveedor.

RF5: Gestionar comité de contratación: la BD debe permitir gestionar los datos del comité de contratación.

RF5.1: Insertar comité de contratación: la BD debe permitir crear un nuevo comité de contratación.

RF5.2: Modificar comité de contratación: la BD debe permitir modificar los datos del comité de contratación.

RF5.3 Eliminar comité de contratación: la BD debe permitir eliminar un comité de contratación.

Prioridad alta:

RF6: Buscar control del plan de importaciones: la BD debe permitir buscar un plan de importaciones existente.

RF7: Buscar control de planes priorizados: la BD debe permitir buscar un plan priorizado existente.

RF8: Buscar control de proveedores: la BD debe permitir buscar un proveedor existente.

RF9: Buscar comité de contratación: la BD debe permitir buscar un comité existente.

Prioridad media:

RF10: Mostrar el control del plan de importaciones: la BD debe permitir mostrar los planes de importaciones existentes.

RF11: Mostrar el control de planes priorizados: la BD debe permitir mostrar los planes priorizados existentes.

RF12: Mostrar el control de proveedores: la BD debe permitir mostrar los proveedores existentes.

RF13: Mostrar comité de contratación: la BD debe permitir mostrar los comité de

contrataciones existentes.

2.4.2. Requisitos no funcionales.

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

RNF de usabilidad.

- El sistema garantizará el acceso, integridad y persistencia de la información almacenada en la base de datos.

RNF de rendimiento.

- Se deberá reducir el tiempo de respuesta a las peticiones por parte de los usuarios a la BD.
- El sistema no debe tener más de 1 hora de no disponibilidad en el mes.

RNF de soporte.

- Se le debe dar mantenimiento periódicamente a los servidores de bases de datos controlando la integridad de la información. Además se harán salvas de los datos cada cierto tiempo.
- Para el servidor de base de datos: se requiere que esté instalado un gestor de base de datos que soporte grandes volúmenes de datos y velocidad de procesamiento.

RNF de portabilidad.

- Mantener la integridad de la información, es decir que no se pierda durante su almacenamiento o transporte.

RNF políticos-culturales.

- Las herramientas propuestas para la base de datos deberán responder a los intereses de la constitución de la república de cuba, asimismo no existirán prioridades en el servicio según el nivel social, cultural o étnico. No se permitirán la divulgación de los datos. Todos los procesos responderán a las resoluciones establecidas por el ministerio de salud pública cumpliendo las normas instituidas

en el código penal.

RNF legales.

- La mayoría de las herramientas de desarrollo son libres y del resto, las licencias están avaladas.

RNF de confiabilidad.

- La base de datos debe permitir a los usuarios conectarse desde cualquier computadora que esté conectada a la red dentro del ministerio.
- La información se debe transmitir a través de canales seguros. Se debe chequear la integridad de los datos.

RNF de hardware.

- Procesador: Pentium iv 1.8 GHz, disco duro: 80 gb, pc con 512 Mb (o superior) de memoria RAM para el desarrollo de la aplicación.

RNF de Software.

- Postgresql 8.4+ permite el uso de bases de datos relacionales.
- La base de datos podrá ser utilizada en sistemas operativos Microsoft Windows millennium (o superior) o Linux por las características que presenta el SGBD utilizado.

RNF Restricciones en el diseño y la implementación.

- Se utilizará power architect para el diseño y modelado de la base de datos.
- La base de datos será implementada utilizando como sistema gestor de base de datos postgresql 8.4+
- La aplicación será implementada con el framework JwebSocket y la tecnología spring-hibernate.

RNF de integridad.

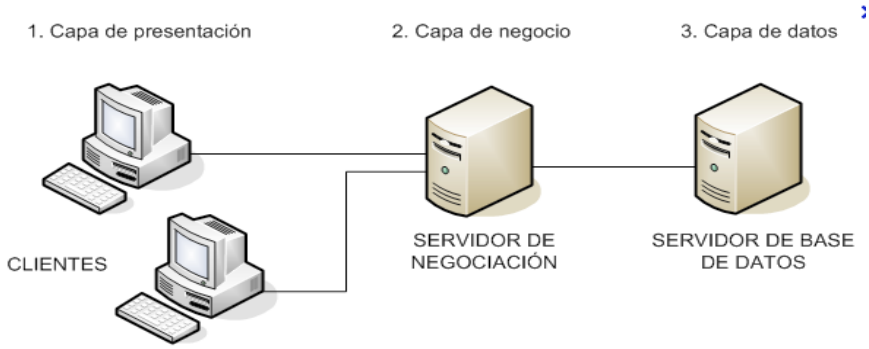
- Se deben adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, hay que proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

2.5. Arquitectura de tres niveles.

También conocida como arquitectura de tres capas, define cómo organizar el modelo de diseño en capas, que pueden estar físicamente distribuidas, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores.

Con esta arquitectura la aplicación se divide en tres capas lógicas distintas, cada una de ellas con un grupo de interfaces perfectamente definido. La primera capa se denomina capa de presentación y normalmente consiste en una interfaz gráfica de usuario de algún tipo.

La capa intermedia, o capa de negocio, consiste en la aplicación o lógica de negocio, y la tercera capa, la capa de datos, contiene los datos necesarios para la aplicación. La capa intermedia (lógica de aplicación) es básicamente el código al que recurre la capa de presentación para recuperar los datos deseados. La capa de presentación recibe entonces los datos y los formatea para su presentación.



Arquitectura de tres niveles.

Para la solución propuesta se enfatizará el estudio en la capa de datos donde se encuentra la base de datos física y la forma de acceder a los datos.

2.5.1. Capa de acceso a datos (CAD).

La capa de acceso a datos describe cómo se implementan los elementos del modelo de diseño. Al analizarla se puede observar una vista general y detallada de

cada uno de los paquetes, en aras de lograr una mayor claridad y comprensión del modelo.

El modelo de la capa de acceso a datos está formado por 3 paquetes, ver la figura 1 en los Anexos.

El paquete modelo contiene las entidades persistentes que representan las clases persistentes que constituyen las entidades del negocio con los atributos que se guardan en la base de datos además contiene las clases generadas del mapeo de la base de datos con la herramienta hibernate donde por cada tabla se genera una clase entidad que representa una tupla de la tabla; a continuación está el paquete data access object (DAO), el cual permite coger las entidades persistentes del negocio y guardarlas en la base de datos, así como recuperarlas de la misma, además de proveer algunas funciones específicas para obtener la información necesaria de la base de datos.

2.5.1. Patrón de diseño utilizado. Data access object (DAO, objeto de acceso a datos).

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un DAO es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una base de datos o un archivo. El término se aplica frecuentemente al patrón de diseño object.

El problema que viene a resolver el patrón de diseño DAO es el de contar con diversas fuentes de datos (base de datos, archivos, servicios externos). De tal forma que se encapsula la forma de acceder a la fuente de datos. Este patrón surge históricamente de la necesidad de gestionar una diversidad de fuentes de datos, aunque su uso se extiende al problema de encapsular no sólo la fuente de datos, sino además ocultar la forma de acceder a los datos. Se trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza

el acceso a los datos o de cuál es la fuente de almacenamiento.

El patrón de diseño DAO implementa el mecanismo de acceso requerido para trabajar con la fuente de datos. Los componentes de negocio que se basan en DAO utilizan la interfaz más simple expuesta por este para sus clientes. DAO oculta completamente los detalles del origen de datos de la aplicación hacia sus clientes. Debido a que la interfaz expuesta por el DAO a los clientes no cambia cuando los datos subyacentes cambian su implementación de código, este modelo permite al DAO adaptarse a los sistemas de almacenamiento sin que ello afecte a sus clientes o componentes de negocio. Esencialmente, DAO actúa como un adaptador entre el componente y la fuente de datos.

La figura 1.1 en los anexos muestra el diagrama de clases que representa las relaciones para el patrón DAO.

Business object (objeto de negocio).

Representa al cliente los datos. Es el objeto que requiere el acceso a la fuente de datos para obtener y almacenar datos.

Data access object (objeto de acceso a datos).

Es el objeto principal de este patrón. El data access object abstrae la implementación del acceso a los datos subyacentes para el business object para permitir el acceso transparente a la fuente de datos. El business object también delega la carga de datos y operaciones de almacenamiento al data access object.

Data source (fuente de datos).

Esto representa una implementación de fuente de datos. Una fuente de datos podría ser una base de datos como un SGBD, repositorio XML, sistema de archivos planos, y así sucesivamente. Una fuente de datos también puede ser una oficina de tarjetas de crédito), o algún tipo de repositorio (LDAP).

Transfer object (objeto de transferencia).

Esto representa un objeto de transferencia que se utiliza como soporte de datos. El data access object puede utilizar un objeto de transferencia para devolver los datos al cliente. El data access object también pueden recibir los datos del cliente en un

objeto de transferencia para actualizar los datos del origen de datos.

La ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio (aquel que contiene detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula.

2.6. Clases persistentes.

El diagrama de clases persistentes es usado para representar las clases que maneja la aplicación y contienen los datos que se guardan de la BD, como se observa en la figura 2. (Ver anexos)

2.7. Modelo relacional.

El modelo relacional representa la segunda generación de los SGBD. En él, todos los datos están estructurados a nivel lógico como tablas formadas por filas y columnas, aunque a nivel físico pueden tener una estructura completamente distinta. Un punto fuerte del modelo relacional es la sencillez de su estructura lógica.

En este modelo todos los datos son almacenados en relaciones, y como cada relación es un conjunto de datos, el orden en el que éstos se almacenen no tiene relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar por un usuario no experto. La información puede ser recuperada o almacenada por medio de consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

Para la manipulación de la información se utiliza un lenguaje relacional, actualmente se cuenta con dos lenguajes formales el álgebra relacional y el cálculo relacional. El álgebra relacional permite describir la forma de realizar una consulta, en cambio, el cálculo relacional sólo indica lo que se desea devolver.

2.8. Diagrama entidad-relación (DER).

Un diagrama o un modelo entidad-relación es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de

información así como sus interrelaciones y propiedades. Ver figura 3 en los anexos.

El modelo entidad-relación.

1. Se elabora el diagrama (o diagramas) entidad-relación.
2. Se completa el modelo con listas de atributos y una descripción de otras restricciones que no se pueden reflejar en el diagrama.

El modelado de datos no acaba con el uso de esta técnica. Son necesarias otras técnicas para lograr un modelo directamente implementable en una base de datos.

Brevemente:

- Transformación de relaciones múltiples en binarias.
- Normalización de una base de datos de relaciones (algunas relaciones pueden transformarse en atributos y viceversa).
- Conversión en tablas.

El modelo de datos entidad-relación está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre esos objetos.

2.8.1. Descripción de las entidades del DER.

Nombre: 1		
Descripción: 2		
Atributo	**Tipo**	**Descripción**
3	**4**	**5**

Leyenda:

(1) Nombre de la tabla.

(2) Descripción general de qué datos almacena la tabla.

(3) Nombre de cada uno de los atributos.

(4) Tipo de dato de cada uno de los atributos.

(5) Breve explicación qué es ese atributo.

Nombre: dimport_Committee		
Descripción: Entidad que recoge los datos de la tabla dimport_Committee identificados por un código		
Atributo	**Tipo**	**Descripción**

id_committee	Entero (Identificador)	Código numérico
act_number	Entero	Valor numérico
importing_entity	Cadena caracteres	String corto
advice	Cadena caracteres	String corto
id_product	Entero (llave foránea tabla dimport_Product)	Valor numérico
numb_contract	Entero	Valor numérico
id_profile_data	Entero (llave foránea tabla dimport_Supplier_Control)	Valor numérico

Nombre: dimport_Hired_Committee		
Descripción: Entidad que recoge los datos de la tabla dimport_Hired_Committee identificados por un código.		
Atributo	**Tipo**	**Descripción**
id_committee	Entero (Identificador)	Código numérico
date	Cadena caracteres	String corto
name_client	Cadena caracteres	String corto
id_coin	Entero (llave foránea tabla dimport_Coins)	Valor numérico
committee_Participant	Cadena caracteres	String corto
company_Participant	Cadena caracteres	String corto
hiring_Proposal	Cadena caracteres	String del tamaño necesario
value	Double	Valor double
delivery	Cadena caracteres	String corto
payment_Way	Cadena caracteres	String del tamaño necesario
observations	Cadena caracteres	String del tamaño necesario

Nombre: dimport_Operations_Control		
Descripción: Entidad que recoge los datos de la tabla dimport_Operations_Control		
Atributo	**Tipo**	**Descripción**
id_group	Entero (llave foránea tabla dimport_Committee_Group)	Valor numérico
id_committee	Entero (llave foránea tabla dimport_Committee_Group)	Valor numérico
received_Date	Cadena caracteres	String corto
analyzed_Date	Cadena caracteres	String corto
observations	Cadena caracteres	String del tamaño necesario

Para ver el resto de las tablas ver los anexos.

2.9. Reglas de transformación del DER al modelo relacional.

La metodología para la transformación consta de los siguientes pasos:

1. Transformación de cada conjunto de entidades del modelo conceptual en un esquema relacional.
2. Transformación de cada interrelación.

Paso 1: Transformación de entidades.

Este paso es muy sencillo, se transforma cada entidad en un esquema, los atributos y llave primaria de la entidad pasan a ser los atributos y llave primaria del esquema.

Paso 2: Transformación de interrelaciones.

La forma de transformar las interrelaciones binarias está muy influenciada por la cardinalidad mínima de los dos conjuntos de entidades asociados.

- Interrelaciones uno-uno: En el caso de que los dos conjuntos de entidades tengan participación obligatoria (cardinalidad mínima 1) la llave de una de ellas pasa a formar parte del esquema que genera la otra, se recomienda que se incluya en el esquema que sea más natural.
- Interrelaciones uno-muchos: Sea R una interrelación entre los conjuntos de entidades E1 y E2 con cardinalidad uno y muchos respectivamente, la interrelación R se transformará usualmente incluyendo la llave primaria de E1 (lado uno) en el esquema correspondiente a E2 (lado muchos) como un atributo o atributos simples.
- Interrelaciones muchos-muchos. La solución no depende de la cardinalidad mínima de la interrelación; en estos caso se crea un nuevo esquema tomando como llave primaria una combinación de los atributos que constituyen las llaves primarias de los conjuntos de entidades asociados por la interrelación e incluye, además, sus atributos descriptivos si los tiene.
- Interrelaciones recursivas: Las interrelaciones recursivas son transformadas de la misma manera que las interrelaciones uno-uno, uno-muchos o muchos-muchos

dependiendo de su cardinalidad máxima.

- Interrelaciones de grado mayor que dos (n-arias): Las interrelaciones n-arias siguen las mismas reglas que las interrelaciones binarias muchos-muchos, para estas se crea un nuevo esquema que contiene todos los identificadores de las n entidades relacionadas, y atributos adicionales en el caso que existan.

2.10. Normalización de las entidades.

La normalización de la BD es el proceso mediante el cual se dividen las relaciones insatisfactorias distribuyendo sus atributos en relaciones mucho más pequeñas que presentan características deseables. El proceso de normalización es aplicable a los modelos entidad relación y a los modelos relacionales. Este proceso permite, una vez terminado, que en las BDs relacionales no existan datos repetidos en las tablas, que se proteja la integridad de los datos y además contribuyen a evitar los problemas de actualización de los datos en las tablas.

Básicamente, las reglas de normalización están encaminadas a eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas y cada una se realizan en orden.

2.10.1. 1era Forma Normal (1FN).

La 1FN constituye el primer nivel en el proceso de normalización y representa el más elemental de todos. Durante el proceso de normalización, se puede comprobar que satisface la (1FN), si se demuestra que no hay presencia de atributos multivaluados o atributos compuestos, o la combinación de estos. Cada uno de los atributos de la relación debe tener un valor atómico. Cada una de las tablas deben contener una clave primaria y esta no contiene atributos nulos.

Para llevar a 1FN se crea una relación con todos los atributos y se señala la llave primaria.

La base de datos para el módulo de la DI del MINCEX cumple con la 1FN, ya que los datos de las relaciones involucradas son atómicos, o sea, no son ni multivaluados, ni compuestos y cada una de las tablas obtenidas cuenta con una

clave primaria.

2.10.2. 2da Forma Normal (2FN).

Antes de pasar a la 2FN, es necesario abordar algunas definiciones previas:

Definición: Dependencia Funcional (DF):

Dada una relación R, se dice que el atributo Y de R es funcionalmente dependiente del atributo X de R, si y sólo si, cada valor X en R tiene asociado a él, precisamente, un valor de Y en R en cualquier momento del tiempo.

X ----> Y.

Definición: Dependencia funcional completa

El atributo Y es funcionalmente dependiente y completamente del atributo X, si es funcionalmente dependiente de X y no es funcionalmente dependiente de algún subconjunto de X.

Se representa: X ==> Y

Para que una relación este en 2FN debe cumplir con: primeramente debe encontrarse en 1FN y además todos los atributos que no forman parte de la llave primaria, dependen de manera total de los atributos que forman la llave primaria. Es decir se eliminan las dependencias parciales y se crean nuevas relaciones con los atributos que dependen de parte de la llave con su nuevo determinante.

Para llevar a 2FN:

- En la relación original se mantienen todos los atributos que dependen completamente de la llave.
- Crear una relación para los atributos que dependan de parte de la llave.

La base de datos para el módulo de la DI del MINCEX cumple con la 2FN, pues ya está en 1FN y cada atributo no primo de la relación es completamente dependiente de la clave primaria. La 2FN se aplicó a las relaciones que tenían claves primarias compuestas por dos o más atributos.

2.10.3. 3era Forma Normal (3FN).

Una relación está en 3FN si está en 2FN y si, y sólo si, los atributos no llaves son independientes de cualquier otro atributo no llave primaria. Esto es lo mismo que

decir que se deben eliminar las dependencias transitivas de atributos no llaves respecto a la llave primaria, estando ya la relación en 2FN.

Definición: Dependencia transitiva:

Sean A, B y C conjuntos de atributos de una relación R. Si B es dependiente funcionalmente de A y C lo es de B, entonces C depende transitivamente de A.

Para llevar a 3FN:

- En la relación original se mantienen los atributos no llaves que no dependen transitivamente de la llave primaria.
- Crear una relación para los atributos no llaves que dependen transitivamente de la llave primaria

Una vez llevada cada una de las relaciones a 2FN se pasó a eliminar las dependencias transitivas para obtener la 3FN.

Para ello, se eliminaron los atributos de cada relación que dependían transitivamente y se pusieron en nuevas relaciones con una copia del atributo o los atributos no clave de los que dependen.

Después de realizado este proceso la base de datos para el módulo de la DI del MINCEX ha quedado normalizada hasta la 3FN por lo que se considera que se desarrolló un adecuado diseño. La base de datos está en 1FN puesto que se puede asegurar que cada tupla contiene exactamente un valor para cada atributo de las tablas. También se cumple que los esquemas de relación están en 2FN, porque se encuentran en 1FN, y además todos los atributos que no son claves en las tablas, dependen totalmente de la clave primaria. Y finalmente se puede plantear que se encuentra en 3FN, porque está en 2FN, y además no existen dependencias transitivas entre llaves candidatas y atributos no primos.

2.11. Conclusiones parciales.

En el presente capítulo se describieron los procesos a automatizar, tales como llevar el control de los planes priorizados, de la cartera de los proveedores, de los planes de importación, de los comités de contratación así como el de los comités de

productos. Se propuso para la situación problemática planteada la solución de desarrollar una base de datos que de cumplimiento a los problemas encontrados en la DI y que garantice la persistencia de los datos así como su organización. Se especificaron los requisitos a implementar en la capa de acceso a datos así como la fundamentación de su arquitectura. Se fundamentó el patrón de diseño DAO como patrón utilizado en la arquitectura de 3 capas. Se realizó el diagrama de clases persistentes y el diagrama entidad-relación para el diseño de la base de datos y se argumentó la descripción de cada entidad representada en el diagrama. Se estudiaron las reglas de transformación del modelo entidad-relación al modelo relacional y por último se normalizaron todas las entidades existentes hasta la 3FN.

CAPÍTULO III: IMPLEMENTACIÓN Y VALIDACIÓN DE LA CAPA DE ACCESO A DATOS.

3.1. Introducción.

En el presente capítulo se describe la implementación del diseño de la base de datos propuesto como solución en el capítulo anterior, se realizan las validaciones correspondientes a la base de datos y a la capa de acceso a datos, a través del framework JUnit y otras pruebas realizadas.

3.2. Implementación de la capa de acceso a datos.

3.2.1. Configuración de hibernate.

Anteriormente se argumentó que hibernate es un motor de persistencia que sirve para la comunicación entre la aplicación y la base de datos, a continuación se explica cómo se configuró para darle un comienzo a la implementación de la CAD.

Pasos a tener en cuenta para su configuración:

1. Primeramente se realiza una conectividad con la base de datos de la siguiente manera:
   - Abrir el Netbeans 7.0.1 y crear un proyecto de aplicación nuevo, después ir a Services, en la pestaña que se llama database, dar clic derecho y crear una nueva conexión, rápidamente aparecerá una nueva ventana como se muestra en la figura 4 (Ver anexos) donde se seleccionará como dispositivo a postgresql ya que es el SGBD utilizado para gestionar la base de datos.
   - Seguidamente presionar “siguiente”, en la siguiente ventana de la figura 4.1, que se encuentra en los anexos, especificar la dirección, el puerto, el nombre, el usuario y la contraseña para entrar a la base de datos y por último probar que la conexión sea satisfactoria.
   - Una vez especificados los parámetros dar clic en “siguiente”. En la próxima ventana que aparece definir el esquema donde se encuentra físicamente la base de datos y por último dar clic en “finalizar”.
2. A continuación se agregan las librerías correspondientes a la conexión con

hibernate-spring: ir a la pestaña de proyecto, dar clic derecho encima del paquete llamado libraries, dar clic donde dice add library, buscar las librerías de hibernate y spring, así como las de postgresql. (ver figura 4.2)

3. En este paso se crea el archivo de configuración de hibernate, el cual se realiza de la siguiente forma: ir a la pestaña de proyecto, dar clic derecho encima del paquete llamado sourcepackages, ir a nuevo, y después a otros, seguidamente aparece una ventana como la que se presenta en la figura 4.3, dar clic en "siguiente".

4. En la próxima ventana se especifica la conexión que se había realizado anteriormente (ver figura 4.3.1)

5. Una vez creado el archivo de configuración, quedaría de la siguiente forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
<property name="hibernate.dialect">org.hibernate.dialect.postgresqlDialect</property>
<property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
<property name="hibernate.connection.url">jdbc:postgresql://10.208.3.27:5432/si_mincex
</property>
<property name="hibernate.connection.username">postgres</property>
<property name="hibernate.connection.password">postgres</property>
<property name="hibernate.show_sql">true</property>
<property name="hibernate.current_session_context_class">thread</property>
</session-factory>
</hibernate-configuration>
```

6. Ahora se crea el archivo de ingeniería inversa de hibernate, el cual se realiza de la siguiente forma: ir a la pestaña de proyecto, dar clic derecho encima del paquete llamado sourcepackages, ir a nuevo, y después a otros, seguidamente aparece una ventana como la que se presenta en la figura 4.4, seleccionarlo y dar clic en "siguiente".

7. En la siguiente imagen se muestran las tablas que existen en la base de datos y que posteriormente serán mapeadas como clases persistentes. (ver figura 4.4.1).

8. Una vez creado el archivo de ingeniería inversa, quedaría de la siguiente forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-reverse-engineering-3.0.dtd">
```

```
<hibernate-reverse-engineering>
<schema-selection match-catalog="si_mincex" match-schema="dir_importaciones"/>
</hibernate-reverse-engineering>
```

9. Luego de haber creado estos dos archivos se dispone a realizar el mapeo de las tablas existentes en la base de datos.

10. Para realizar este mapeo, se crea un nuevo paquete llamado model y dentro de este dos más llamados entity y dao, en el paquete llamado entity se guardarán las clases mapeadas y en el otro las clases dao que se vayan a utilizar.

Inicio del mapeo de clases

11. En la figura 4.5 y 4.5.1 de los anexos aparecen unas ventanas en las que se genera el código desde las tablas de la base de datos.

A continuación se presenta el código generado de una de las clases:

```
packagemodel.entity;
// Generated 19-mar-2012 21:03:16 by Hibernate Tools 3.2.1.GA
importjava.util.HashSet;
importjava.util.Set;
importjavax.persistence.CascadeType;
importjavax.persistence.Column;
importjavax.persistence.Entity;
importjavax.persistence.FetchType;
importjavax.persistence.GeneratedValue;
importjavax.persistence.Id;
importjavax.persistence.OneToMany;
importjavax.persistence.SequenceGenerator;
importjavax.persistence.Table;

/**
 * DimportPrioritizedPlan generated by hbm2java
 */
@Entity
@Table(name="dimport_prioritized_plan"
,schema="dir_importaciones"
)
public class DimportPrioritizedPlan extends AbstractEntity implements java.io.Serializable {
privateintidPrioritizedPlan;
private String madeBy;
private Set<DimportImportations>dimportImportationses = new HashSet<DimportImportations>(0);

publicDimportPrioritizedPlan() {}
publicDimportPrioritizedPlan(String madeBy ) {
        this.madeBy = madeBy;}
publicDimportPrioritizedPlan(intidPrioritizedPlan, String madeBy) {
            this.idPrioritizedPlan=idPrioritizedPlan;
            this.madeBy = madeBy; }
   @Id
```

```
@Column(name="id_prioritized_plan", unique=true, nullable=false)
 @SequenceGenerator (name = "id_seq", sequenceName = "dir_importaciones.dimport_prioritized_plan_id_prioritized_plan_seq")
@GeneratedValue(generator = "id_seq")
publicintgetIdPrioritizedPlan() {
returnthis.idPrioritizedPlan;   }

public void setIdPrioritizedPlan(intidPrioritizedPlan) {
this.idPrioritizedPlan = idPrioritizedPlan;}

@Column(name="made_by", nullable=false)
public String getMadeBy() {
returnthis.madeBy;   }

public void setMadeBy(String madeBy) {
this.madeBy = madeBy;
   }
@OneToMany(cascade=CascadeType.ALL, fetch=FetchType.LAZY, mappedBy="dimportPrioritizedPlan")
public Set<DimportImportations>getDimportImportationses() {
returnthis.dimportImportationses;
public void setDimportImportationses(Set<DimportImportations>dimportImportationses) {this.dimportImportationses = dimportImportationses;   }
}
```

Cada clase mapeada hereda de abstractentity, dicha clase tiene como objetivo incrementar automáticamente la secuencia que tienen los id de cada tabla en la base de datos.

Para un mejor entendimiento de la estructura que tiene el proyecto ver la figura 5.

En el paquete model.dao se encuentra por defecto una clase llamada genericdao esta clase tiene como objetivo brindar funcionalidades que operan directamente con la base de datos, algunas de ellas son:

saveorUpdate: Esta funcionalidad actualiza o inserta un objeto en la base de datos, recibiendo por parámetros una entidad.

save: Esta funcionalidad inserta un objeto en la base de datos, recibiendo por parámetros una entidad.

update: Esta funcionalidad actualiza un objeto en la base de datos, recibiendo por parámetros una entidad.

find: Esta funcionalidad devuelve un listado con todos los objetos encontrados de una entidad, recibiendo por parámetros la clase correspondiente a la entidad.

delete: Esta funcionalidad elimina un objeto en la base de datos, recibiendo por parámetros una entidad.

findByParamAndValue: Esta funcionalidad devuelve un listado con todos los objetos encontrados, recibiendo por parámetros la clase correspondiente a la entidad, el campo o atributo por el que se va a buscar y el posible valor que tenga.

findByParamsAndValues: Esta funcionalidad devuelve un listado con todos los objetos encontrados, recibiendo por parámetros la clase correspondiente a la entidad, varios campos o atributos por los que se van a buscar y los posibles valores que tengan.

findByHQL: La presente funcionalidad ejecuta cualquier consulta que se implemente, dicha consulta debe pasarse por parámetros y devuelve un listado con los objetos encontrados en la base de datos.

3.3. Validación de la capa de acceso a datos.

3.3.1. Validación teórica del diseño.

Al realizar el diseño de una base de datos, se tienen que tener en cuenta también los aspectos que garanticen la persistencia, integridad y acceso a la información. Los sistemas gestores de bases de datos deben cumplir estos objetivos para facilitar la manipulación y confidencialidad de los datos.

Integridad de los datos.

La implementación de la integridad relacional en una BD consiste en establecer las reglas de consistencia correspondientes a los datos requeridos de las tablas, el chequeo de los valores válidos y únicos, así como la integridad de las llaves primarias y foráneas.

Tipos de restricciones de integridad en BD relacionales.

- **Datos requeridos:** Establece que una columna tenga un valor no nulo. Se define efectuando la declaración de una columna es NOT NULL cuando la tabla que contiene las columnas se crea por primera vez, como parte de la sentencia CREATE TABLE.
- **Chequeo de validez:** Cuando se crea una tabla cada columna tiene un tipo de datos y el SGBD asegura que solamente los datos del tipo especificado sean ingresados en la tabla.

- **Integridad de entidades:** Un caso de este tipo de restricción es el de las llaves primarias para cada tabla. En el caso de la base de datos que se está analizando cada entidad tiene definida su llave primaria, que no puede ser nula ni se puede repetir. En la imagen 6 de los anexos se puede apreciar el error lanzado por el SGBD cuando se intenta introducir una llave primaria que ya existe.
- **Integridad referencial:** La regla de integridad se aplica a las claves foráneas: si en una relación hay alguna clave foránea, sus valores deben coincidir con los valores de la clave primaria a la que hace referencia. Cuando se define una columna como clave foránea, las filas de la tabla pueden contener en esa columna o bien el valor nulo (ningún valor), o bien un valor que existe en la otra tabla. Eso es lo que se denomina integridad referencial y consiste en que los datos que referencian otros (claves foráneas) deben ser correctos. La integridad referencial hace que el SGBD se asegure de que no haya en las claves foráneas valores que no estén en la tabla principal.

Se pueden producir errores en los datos cuando:

- Se inserta una nueva fila en la tabla secundaria y el valor de la clave foránea no existe en la tabla principal.
- Se modifica el valor de la clave principal de un registro que tiene 'hijos'.
- Se modifica el valor de la clave foránea, el nuevo valor debe existir en la tabla principal.
- Se desea borrar una fila de la tabla principal y ese registro tiene 'hijos'.

La imagen 6.1 que se muestra en los anexos se puede ver un error lanzado por el SGBD cuando se intenta introducir una nueva fila en la tabla secundaria y el valor de la clave foránea no existe en la tabla principal.

Asociada a la integridad referencial están los conceptos de actualizar los registros en cascada y eliminar registros en cascada.

- Actualizar registros en cascada**:** Esta opción le indica al SGBD que cuando se cambie un valor del campo clave de la tabla principal, automáticamente cambiará el valor de la clave foránea de los registros relacionados en la tabla secundaria.

- Eliminar registros en cascadas**:** Esta opción le indica al SGBD que cuando se elimina un registro de la tabla principal automáticamente se borran también los registros relacionados en la tabla secundaria.

Para mantener esta integridad referencial en la base de datos se utilizaron: las llaves foráneas o externas, las cuales obligan a que los valores introducidos en las columnas marcadas por esta restricción correspondan a valores en las tablas referenciadas, así como permite realizar acciones en caso de actualización o eliminación de los valores a los que se hacen referencia; para esto, se definió que al realizar la actualización o eliminación de datos, las mismas se realizarían en cascada. Puntualizar que a la hora que el administrador de la aplicación vaya a eliminar debe tener un previo conocimiento de las dependencias que existen entre las tablas, debido a que eliminar una tabla intermedia podría producir errores en la base de datos.

Redundancia de la información.

La redundancia de datos es aquella información duplicada o almacenada varias veces en la misma base de datos. Esto dificulta la tarea de modificación de datos y es el motivo más frecuente de inconsistencia de los mismos. Además requiere un mayor espacio de almacenamiento, que influye en un mayor coste y tiempo de acceso a los datos. La redundancia siempre debe evitarse, aunque en proyectos grandes es imposible evitarla al cien por ciento, lo que a veces es deseable por cuestiones de rendimiento.

Con un buen diseño de base de datos se logrará evitar la aparición de información repetida o redundante. Lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

3.3.2. Validación funcional.

Para comprobar que la base de datos para la DI del MINCEX cumple con los requisitos funcionales definidos, es necesario realizarle pruebas antes de dar por terminado el proceso de diseño. El propósito de estas pruebas es saber a la hora

de integrarla con la aplicación si es capaz de devolver correctamente los registros determinados por el usuario y de conocer si a partir de dicha aplicación se puede realizar operaciones tales como insertar, eliminar, buscar y actualizar sobre las tuplas de la base de datos.

Pruebas unitarias a la CAD.

Las pruebas unitarias permiten garantizar el correcto funcionamiento de pequeñas piezas de software, lo cual guía al programador hacia un proceso de integración ágil y eficiente.

La clase por probar es un DAO que contiene métodos para calcular totales y porcentajes utilizados en el negocio. Dicha clase implementa la prueba unitaria para org.junit.DaoPriorizedPlan.

```
import java.util.List;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;
/**** @author yiselm */
public class DaoPriorizedPlanTest {
 GenericDao dao;
   public DaoPriorizedPlanTest() {--------    }
@BeforeClass
   public static void setUpClass() throws Exception {  }
   @AfterClass
   public static void tearDownClass() throws Exception {   }
      @Before
   public void setUp() {    }
      @After
   public void tearDown() {    }
   /**
    * Test of Closed method, of class DaoPriorizedPlan.
    */
   @Test
   public void testClosed() {
      System.out.println("Closed");
      String date = "25/03/2012";
      DaoPriorizedPlan instance = new DaoPriorizedPlan(dao.getSessionFactory());
      String expResult = instance.Closed(date);
      String result = instance.Closed(date);
      assertEquals(expResult, result);
      // TODO review the generated test code and remove the default call to fail. }
   /**
    * Test of TotalSubmittedPlan method, of class DaoPriorizedPlan.
    */
```

```
@Test
public void testTotalSubmittedPlan() {
    System.out.println("TotalSubmittedPlan");
    DaoPriorizedPlan instance = new DaoPriorizedPlan(dao.getSessionFactory());
    double expResult = instance.TotalSubmittedPlan();
    double result = instance.TotalSubmittedPlan();
    assertEquals(expResult, result, 0.0);
    // TODO review the generated test code and remove the default call to fail.  }
```

A continuación se presenta el resultado obtenido al realizar esta prueba a todos los daos con que cuenta el proyecto.

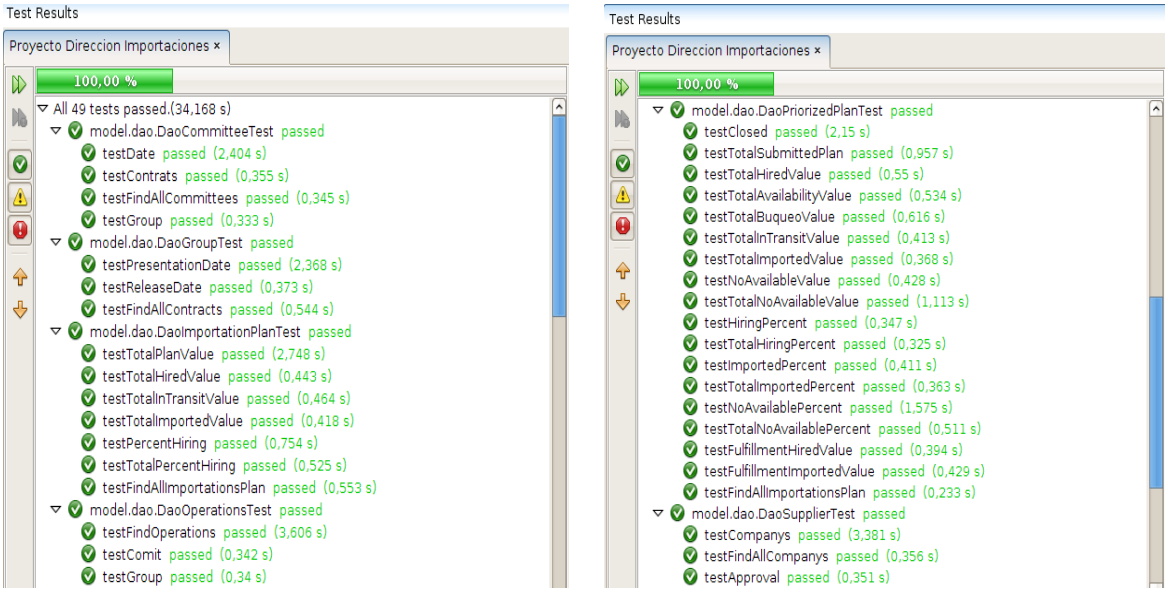


Fig. 9 Resultado de la prueba unitaria a los daos del proyecto Dirección Importaciones.

Como se muestra en las imágenes anteriores las pruebas realizadas a la capa de acceso a datos fueron satisfactorias en un 100%.

Pruebas para demostrar el acceso y la persistencia de la información.

El código que se muestra a continuación es para probar el acceso de la capa de acceso a datos a la BD del proyecto.

A continuación el código de cómo **Insertar** una fila desde la CAD:

```
DimportImportationPlanimportPlan=new DimportImportationPlan("Jose", "Enero", 1985, "Agricultura", 23654.36, 2365.1, "CUC", 2365.23, 2365.14);
dao.save(importPlan); //se inserta un nuevo objeto
```

Código de cómo **Eliminar** una fila desde la CAD:

```
String[] params = {"closed", "year"};
```

```
Object[] objects = {"Febrero", 1985};
List<DimportImportationPlan>listImportPlan=dao.findByParamsAndValues(DimportImportationPlan.class, params , objects);
dao.delete(listImportPlan.get(0)); //se busca por varios parámetros para eliminar
```

Código de cómo **Actualizar** una fila desde la CAD:

```
List<DimportImportationPlan>listImportPlan=dao.findByParamsAndValues(DimportImportationPlan.class, params , objects);
DimportImportationPlanupdateImportPlan=new DimportImportationPlan (listImportPlan.get(0).getIdImportationPlan(), "Maria", "Diciembre", 1990, "Ganaderia", 34354.36, 54265.1, "CUC", 154665.23, 26735.14);
dao.update(updateImportPlan);//se actualiza cualquier campo del objeto, buscado por varios parámetros.
```

Código de cómo **Mostrar** una fila desde la CAD:

```
List<DimportImportationPlan>listarPlanesImport=dao.find(DimportImportationPlan.class);
for (int i = 0; i <listarPlanesImport.size(); i++) {
DimportImportationPlanplanImport=listarPlanesImport.get(i);
System.out.println(planImport.getMadeBy()+" "+planImport.getOrganizations()+" "+planImport.getClosed()+" "+planImport.getHired()+" "+planImport.getImported()+" "+planImport.getYear());}
```

Resultados de las pruebas

En la imagen 7 se plasma el resultado mostrado al ejecutar la funcionalidad de Mostrar desde la aplicación así como en la imagen 7_1 una tabla de la base de datos. (ver imágenes en los anexos).

3.4. Resultado de la entrevista.

La entrevista no estructurada es más abierta que la estructurada, prevé el tema pero no lleva un cuestionario rígido y puede variar de una persona a otra, es más flexible. Se aplica a especialistas en el tema, es una forma de obtener criterios de expertos.

De acuerdo con la información que se quiere obtener los tipos de preguntas a realizar se pueden clasificar de la forma siguiente:

- Cerradas: Se limita su respuesta a varias posibilidades previstas, donde la respuesta esta estructurada por comparaciones.
- Abiertas: Son preguntas para ser respondidas libremente, no permiten obtener con exactitud la información deseada, sólo se logra conocer la opinión del entrevistado.
- Semicerradas: Limita la respuesta pero deja espacio libre para emitir

opiniones sobre el tema.

- Directas: Cuando el objetivo de la pregunta coincide con el objeto de interés del investigador.
- De control: Se usan para valorar la consistencia de las respuestas dadas a determinadas preguntas.

Se le realizó la entrevista a 4 especialista de la DI del MINCEX, entre ellas el director. A los entrevistados se le hicieron un total de 11 preguntas de ellas 3 cerradas, 3 abiertas, 2 semicerradas, 2 directas y 1 de control, usándose las mismas preguntas para todos.

3.5. Aporte social y económico.

Uno de los aportes más importantes y aplicables que ha traído la Informática a las actividades diarias de las organizaciones ha sido el concepto de base de datos, este término, ha pasado de ser técnico a ser de uso diario en cualquier ámbito de trabajo en la vida diaria: la escuela, la casa, la oficina.

Para la dirección de importaciones del MINCEX la creación de una base de datos así como su capa de acceso a la información resulta una manera de trabajo más cómoda y sencilla.

No solo va a permitir tener la información almacenada de forma más segura si no que va a estar disponible todo el tiempo, lo que significa que se puede acceder a estos de manera más fácil que si se hiciera manualmente.

En cuanto al aporte económico, con la confección de una base de datos y su capa de acceso a datos en la Universidad de las Ciencias Informáticas se ahorra pagarle estos servicios a una empresa productora de software lo que va a ser muy bueno para el MINCEX, además con la creación del sistema es preciso comprar un buen servidor que soporte toda la información a almacenar, por lo que ese dinero se puede emplear en esa compra.

3.6. Conclusiones parciales.

Con la implementación de la base de datos, primeramente se argumentaron los

pasos para configurar hibernate y a partir de este realizar el mapeo de las clases, donde a cada tabla de la base de datos le corresponde una entidad del negocio. Se le realizaron pruebas teóricas y funcionales a la base de datos y a la capa de acceso a datos para validar su funcionamiento, esto se puso en práctica a través de pruebas al código mediante el framework JUnit y la demostración de la funcionalidad de los métodos implementados en los DAOs para garantizar el acceso y la persistencia de la información. Por último se realizó un breve estudio del aporte social y económico brindado al MINCEX la confección de una base de datos y la implementación de su capa de acceso a datos.

CONCLUSIONES GENERALES.

- Se realizó un estudio concreto de las tendencias y tecnologías actuales, permitiendo seleccionar las herramientas adecuadas para el desarrollo de la solución propuesta.
- Fueron modelados los diagramas relacionales (teniendo en cuenta la representación de la estructura lógica y física) de la base de datos según los módulos propuestos, resaltándose la descripción de cada entidad y sus relaciones.
- Se llevó a cabo la creación de la base datos generando el Script de la estructura física, a través de construir una base de datos usando postgresql como SGBD, para implementar la BD.
- Se diseñó el diagrama de clases persistentes, con sus respectivas descripciones y haciendo uso del mismo se especificó la arquitectura de la capa de acceso a datos.
- Se empleó la tecnología hibernate-spring para el acceso a datos, con el mapeo de las tablas de la BD utilizando el patrón de diseño DAO.
- Se tuvo en cuenta la integridad de la información, las validaciones de los campos de cada tabla, además de implementarse la capa de acceso a datos, verificándose si la información es correctamente almacenada y todo con vista a las funcionalidades principales de la base de datos.
- Se logró realizar pruebas, con la perspectiva de probar el funcionamiento de la BD desde una aplicación web.
- Se integró en su uso la plataforma de java, con ésta el framework JwebSocket y la tecnología de hibernate-spring para desarrollar su conectividad con la BD.

REFERENCIAS BIBLIOGRÁFICAS.

[1] Base de Datos. Enciclopedia Libre. 2011 [Consultado el: 22 de Noviembre de 2011]; Disponible en: http://es.wikipedia.org/wiki/Base_de_datos.

[2] Página Historia de la Informática. Blog sobre Historia de la Informática. Historia de las bases de datos. 2011. [Consultado el: 28 de Noviembre 2011]; Disponible en http://histinf.blogs.upv.es/2011/01/04/historia-de-las-bases-de-datos/

[3] Ver conferencia Introducción_a_los_Sistemas_de_Bases_de_Datos. Pág: 2, 2do párrafo [Consultado el: 30 de Noviembre 2011].

[4] 1. Mato G. Lic. Rosa M. Diseño de Bases de Datos. 1999.

[5] Date, Christopher J. Introducción a los sistemas de bases de datos. 7ma ed, México: Editorial Pearson Educación S.A, 2001, ISBN 968-444-419-2.

[6] Marqués A. María M. Diseño de bases de datos, 2001. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node68.html.

[7] Ver conferencia Introducción_a_los_Sistemas_de_Bases_de_Datos. Pág: 2

[8] Korth, Henry F, Silberschatz, Abraham. Database System Concepts, McGraw-Hill, New York, NY, 1991, ISBN 0071008047.

[9] Marqués Andrés, María Mercedes. Apuntes de Ficheros y Bases de Datos. 2001. Disponible en: http://www3.uji.es/~mmarques/f47/apun/apun.html.

[10] Abad CalderínIng.Yenin. “Procedimiento para el control de tareas investigativas en la producción de software en la UCI”. Pág.14

[11] Peñalver G, Meneses A, Garcías S. “SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE”. 1Er Congreso Iberoamericano de Ingeniería de Proyectos. Mayo 2010. Universidad de las Ciencias Informáticas. Ciudad de la Habana. Cuba

[12] “SQL PowerArchitect herramienta de modelado de datos.” [Online]. Disponible: http://www.tuinformaticafacil.com/herramientas-desarrollo/sql-power-

architect-herramienta-de-modelado-de-datos.[Accedido: 05-Nov-2011].

[13] "Java Database Connectivity - Wikipedia, la enciclopedia libre." [Online]. Disponible en: http://es.wikipedia.org/wiki/Java_Database_Connectivity. [Accedido: 05-Nov-2011].

[14] "Arrastrar y soltar - Wikipedia, la enciclopedia libre." [Online]. Disponible en: http://es.wikipedia.org/wiki/Arrastrar_y_soltar. [Accedido: 05-Nov-2011].

[15] "OLAP - Wikipedia, la enciclopedia libre." [Online]. Disponible en: http://es.wikipedia.org/wiki/OLAP. [Accedido: 05-Nov-2011].

[16] "postgresql - Guía Ubuntu." [Online]. Disponible en: http://www.guia-ubuntu.org/index.php?title=postgresql. [Accedido: 05-Nov-2011].

[17] "JavaBean - Wikipedia, la enciclopedia libre." [Online]. Disponible en: http://es.wikipedia.org/wiki/JavaBean. [Accedido: 05-Nov-2011].

[18] "Hibernate - Wikipedia, la enciclopedia libre." [Online]. Disponible en: http://es.wikipedia.org/wiki/Hibernate. [Accedido: 05-Nov-2011].

[19] "Bases de datos." [Online]. Disponible: http://www.hipertexto.info/documentos/b_datos.htm.[Accessed:10-Dec-2011].

[20] "Oracle - EcuRed." [Online]. Available: http://www.ecured.cu/index.php/Oracle. [Accessed: 14-Dec-2011].

[21] "MySQL." [Online]. Available: http://danielpecos.com/docs/mysql_postgres/x57.html. [Accessed: 14-Dec-2011].

[22] "Características de postgresql | Manuales gratis." [Online]. Available: http://www.manualesdeayuda.com/manuales/bases-dedatos/postgresql/caracteristicas-de-postgresql-01844.html.[Accessed: 14-Dec-2011].

[23] "Patrones de diseño de bases de datos - EcuRed." [Online]. Available:

http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_de_bases_de_datos. [Accessed: 12-Jan-2012].

[24] GONZÁLEZ, Ó. G. M. Y. F. R. *ARQUITECTURAS DE SISTEMAS DE BASES DE DATOS*,1999/2000. [2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node33.html

[25] GONZÁLEZ, Ó. G. M. Y. F. R. *ARQUITECTURAS DE SISTEMAS DE BASES DE DATOS*,1999/2000. [2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node33.html

[26] "Persistencia Básica en Java." [Online]. Disponible: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=PersistenciaJava. [Accessed: 26-Jan-2012].

[27] Ha Muerto el Diseño? [cited 16 February 2012]. Available: http://www.programacionextrema.org/articulos/designdead.es.html.

[28] Motores de Persistencia. Programación en Castellano. [citado 23 febrero 2012]. Available from world wide web: http://www.programacion.com/articulo/motores_de_persistencia_231#joa_persistencia_motores

[29] 2010 mayo 15 «□ Gabbyt@z. [citado 23 febrero 2012]. Available from world wide web: http://gabbytaz.wordpress.com/2010/05/15/.

[30] EMS Data Generator for PostgreSQL Español Disponible en: http://es.softpicks.net/software/Desarrollo/Bases-de-datos-redes/EMS-Data-Generator-for-PostgreSQL_es-175351.htm.

[31] HÉCTOR FUENTE PÉREZ. JUnit: Manual Básico□: Héctor Fuente Pérez Disponible en: http://fuenteperez.es/blog/junit-manual-basico.

[32] Clasificación de base de datos, base de datos estáticas, base de datos dinámicas, según la variabilidad de datos, elaboración de base de datos en Lima Perú Disponible en: http://www.netronycs.com/clasificacion_de_base_datos.html.

[33] MODELO DE RED Disponible en:

http://www.angelfire.com/my/jimena/bdat1/guia8.htm.

[34] ING. JAVIER ANTONIUCCI. Persistencia Básica en Java Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=PersistenciaJava.

[35] Gestión de Bases de Datos Disponible en: http://www.slideshare.net/MauroMingo/gestion-de-bases-de-datos.

[36] MODELO DE DATOS Disponible en: http://aurea.es/wp-content/uploads/modelodedatos.pdf

[37] Definición Disponible en: http://definicion.de/layout/

[38] Ágora: Glosario Informático Disponible en: http://pefc5.ugr.es/moodle/mod/glossary/view.php?id=5&mode=letter&hook=G&sortkey=&sortorder=.

[39] Ágora: Glosario Informático Disponible en: http://pefc5.ugr.es/moodle/mod/glossary/view.php?id=5&mode=letter&hook=G&sortkey=&sortorder=.

[40] SCRUM Disponible en: http://www.chuidiang.com/ood/metodologia/scrum.php.

[41] Programación extrema Disponible en: http://www.chuidiang.com/ood/metodologia/extrema.php.

[42] Spring Framework: Introducción. [citado 9 junio 2012]. Available from World Wide Web: http://www.genbetadev.com/java-j2ee/spring-framework-introduccion.

[43] NetBeans IDE. [citado 9 junio 2012]. Available from World Wide Web: http://netbeans-ide.softonic.com/.

[44] NetBeans IDE 7.0.1. [citado 9 junio 2012]. Available from World Wide Web: http://www.argentinawarez.com/programas-gratis/1515108-netbeans-ide-7-0-1-a.html.

BIBLIOGRAFÍA.

- Peñalver G, Meneses A, García S. “SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE”. 1Er Congreso Iberoamericano de Ingeniería de Proyectos. Mayo 2010. Universidad de las Ciencias Informáticas. Ciudad de la Habana. Cuba
- “postgresql - Guía Ubuntu.” [Online]. Disponible en: http://www.guia-ubuntu.org/index.php?title=postgresql. [Accedido: 05-Nov-2011].
- “Características de postgresql | Manuales gratis.” [Online]. Available: http://www.manualesdeayuda.com/manuales/bases-dedatos/postgresql/caracteristicas-de-postgresql-01844.html.[Accessed: 14-Dec-2011].
- Mato G. Lic. Rosa M. Diseño de Bases de Datos. 1999.
- Date, Christopher J. Introducción a los sistemas de bases de datos. 7ma ed, México: Editorial Pearson Educación S.A, 2001, ISBN 968-444-419-2.
- Marqués A. María M. Diseño de bases de datos, 2001. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node68.html.

ANEXOS.

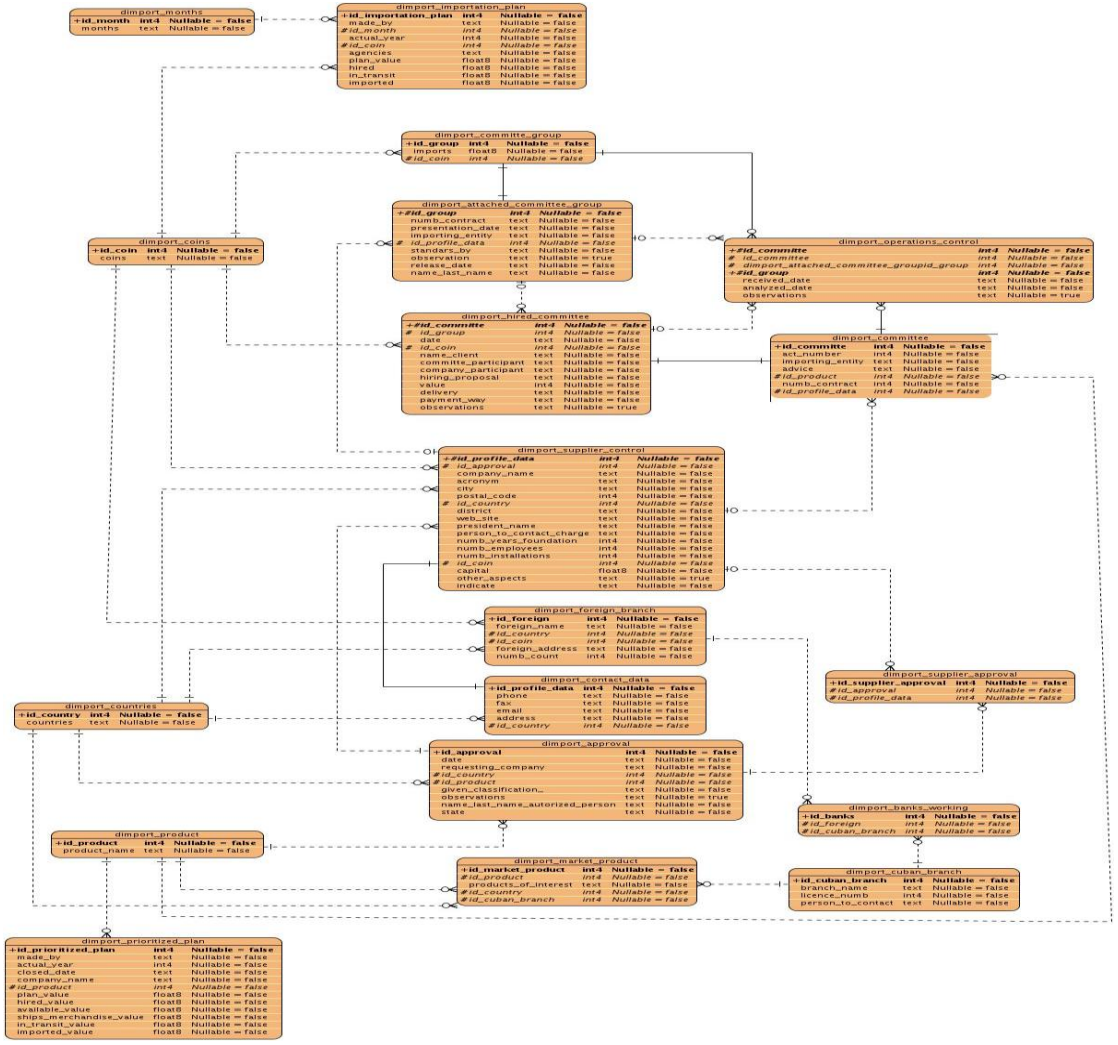


Fig.2. Diagrama de clases persistentes.

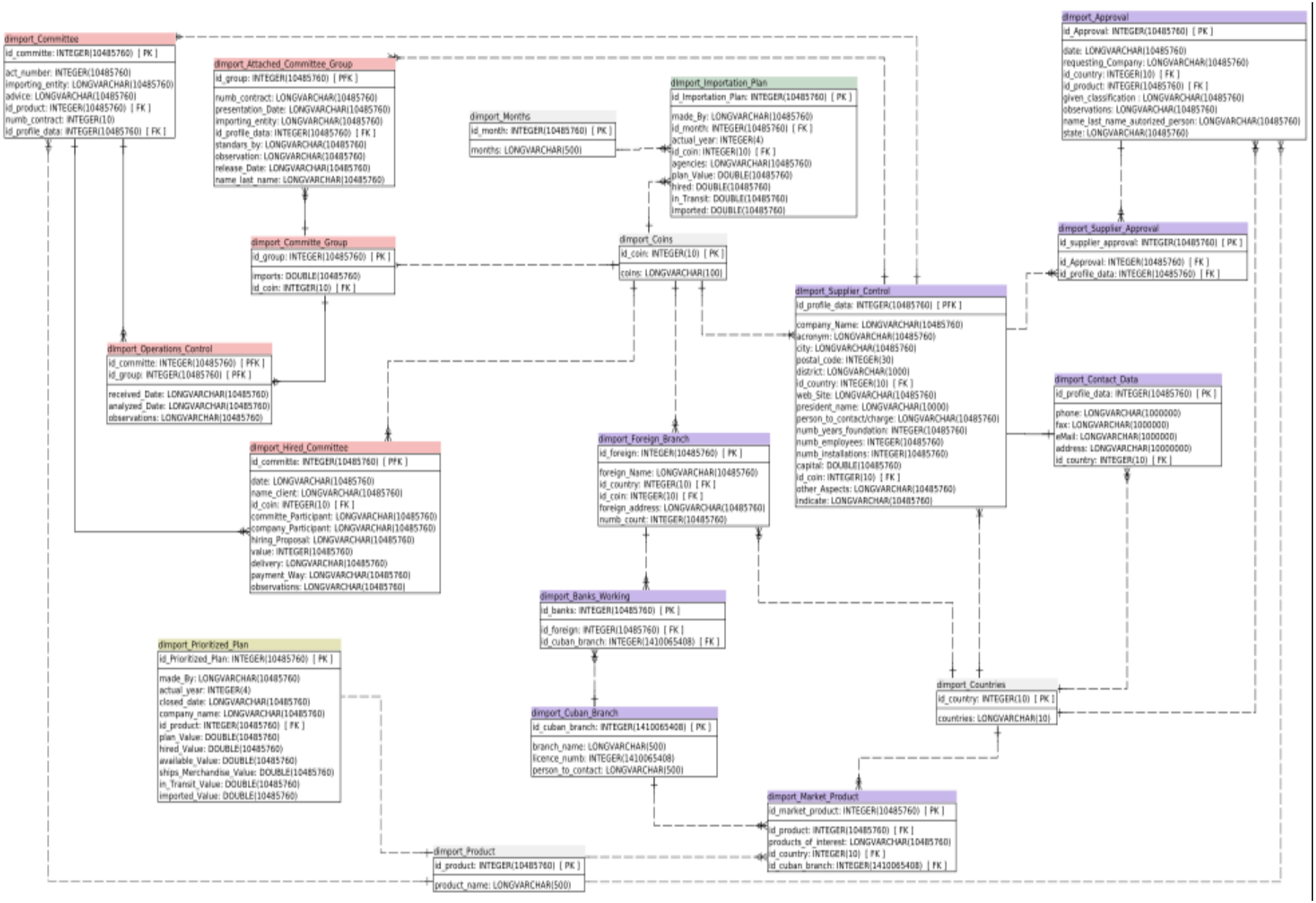


Fig. 3. Diagrama entidad-relación.