Universidad de las Ciencias Informáticas

Facultad 3

Herramienta para la detección y corrección de errores de diseño en un esquema de base de datos en Oracle 11g.

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

Autor: Disnel Arzuaga Guevara

Tutor: Ing. Liannis Soria Barreda

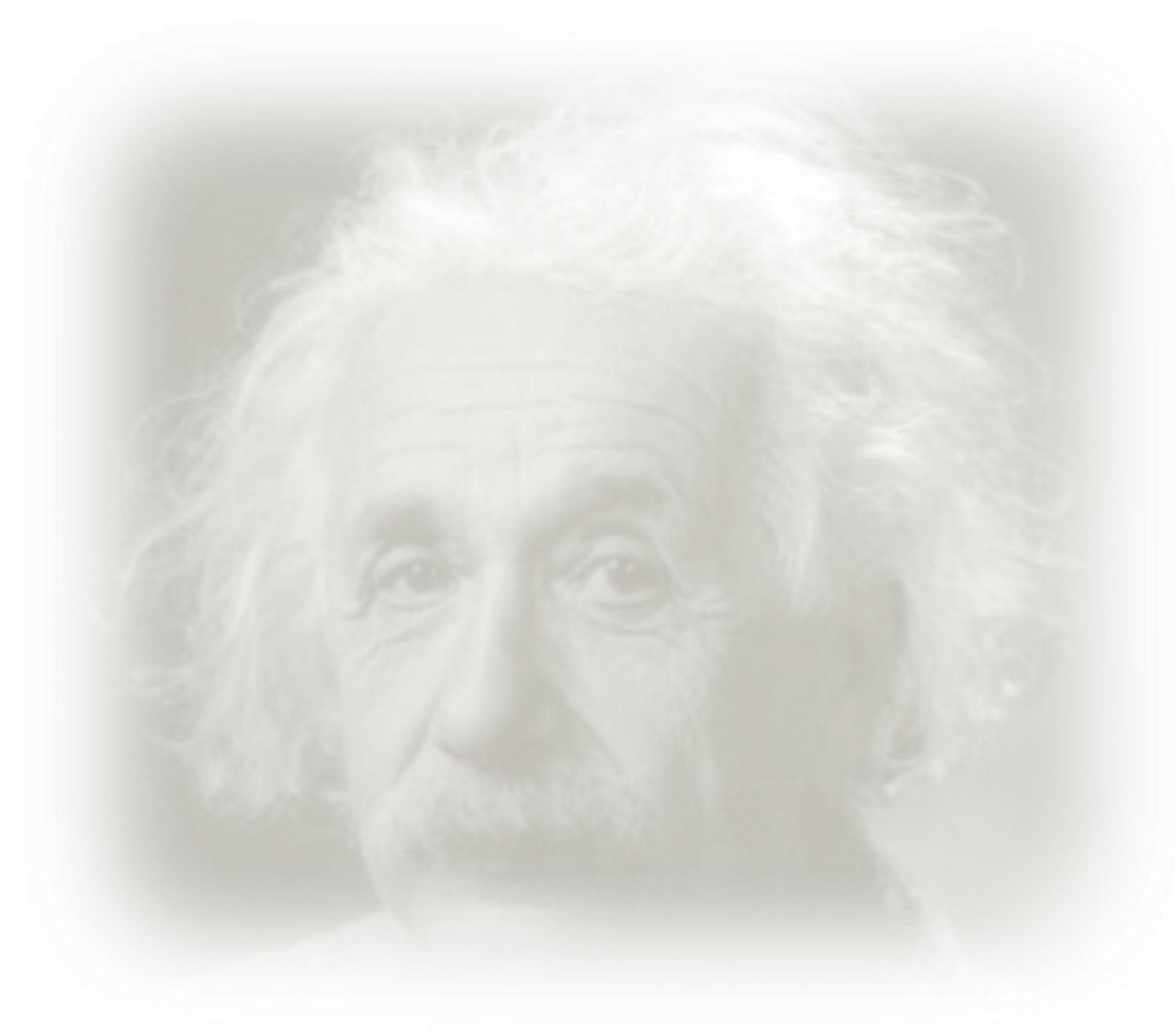
Co-tutores:

Ing. Fernando Nápoles Gámez

Ing. Adrián Naranjo García

La Habana, Junio de 2013

Año 55 de la Revolución



"Si lo puedo pensar e imaginar, es que lo puedo hacer."

Albert Einstein

DECLARACIÓN DE AUDITORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas, para que lo usen según estimen pertinente.

Para que así conste, firmamos la presente a los ___ días del mes de _____ del año _____.

Disnel Arzuaga Guevara

_____

Firma del Autor

Ing. Liannis Soria Barreda

_____

Firma del Tutor

Ing. Fernando Nápoles Gámez

_____

Firma del Tutor

Ing. Adrián Naranjo García

_____

Firma del Tutor

DATOS DE CONTACTO

Autor:

Disnel Arzuaga Guevara

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: darzuaga@estudiantes.uci.cu

Tutores:

Ing. Liannis Soria Barreda

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: lsoria@.uci.cu

Ing. Fernando Nápoles Gámez

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: fnapoles@.uci.cu

Ing. Adrián Naranjo García

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: angarcia@.uci.cu

Agradezco a:

En primer lugar a mi madre porque me ha dedicado su vida y por darme su apoyo en todo momento, por su sacrificio en todos mis años de estudio desde la primaria hasta aquí, por su entrega y por el amor que me ha brindado siempre. A mi padre por la confianza que ha tenido en mí, por su apoyo incondicional y sobre todo por su amor.

A mi tutor Fernando por todo su apoyo y por tener paciencia conmigo, gracias por todo, sin tú ayuda no hubiese sido posible todo esto.

A toda mi familia porque es lo mejor que tengo, porque siempre me han apoyado en todo momento. A mi novia porque desde que la conocí me ha ayudado mucho en todo. A todas las personas que he conocido en estos cinco años y que siempre han estado ahí en todo momento, a Tony por ser un gran amigo y por su amistad. A mis amigos de siempre en Niquero, a todos gracias por su apoyo.

Agradezco a todo el que de una forma u otra me ha ayudado a salir adelante.

Resumen.

En el Departamento de Aduana-CCHH (Capital Humano) específicamente en la Línea de producción de software se han venido presentando algunas dificultades con la estandarización de la base de datos, en Oracle, que es el gestor con que se trabaja en dicho departamento, se presentan inconvenientes para detectar y corregir de forma manual los errores de diseño presentes en los diferentes objetos de un esquema. En la línea existe el "estándar de codificación de base de datos", en el que se reflejan las especificaciones del proyecto productivo, que ayudan a obtener un mejor resultado en la tarea de diseño de los esquemas de base de datos. Este documento es ignorado en muchas ocasiones por desconocimiento, falta de práctica o negligencia por parte del personal que ejecuta el diseño, por lo que muchos de los modelos de datos, se diseñan e implantan con deficiencias. Estos errores provocan un atraso en el desarrollo de las diferentes aplicaciones implementadas en la línea, debido a que muchos de los mismos son detectados en la fase de implementación y el proceso de rectificación en el modelo físico de la base de datos se torna complejo y lento, demorándose como promedio hasta 1 día identificarlos y 2 días aproximadamente para lograr su corrección en dependencia de la cantidad de objetos y la complejidad de los errores. En este trabajo se diseña e implementa una herramienta que automatiza el proceso de detección y corrección de los errores de diseño en la base de datos de la línea, siguiendo el estándar definido para ese fin, brindándole al usuario la posibilidad de agilizar este engorroso proceso. Se identificaron los errores que afectaban la base de datos clasificándolos en críticos, menos críticos y leves y se logró su corrección. Se evalúan los esquemas según el estado en que estos se encuentran, brindando posibilidad de exportar estos reportes a formato Excel.

Palabras claves: Estándar de Codificación de Base de Datos, errores de diseño, herramienta, Oracle.

ÍNDICE

ÍNDICE

Introducción

La Aduana General de la República (AGR) es una entidad que forma parte del control estatal realizando un conjunto de operaciones que garantizan la seguridad y protección de la sociedad y de la economía nacional. Esta organización tiene como objetivo fundamental supervisar y controlar los medios de transporte, las mercancías y los viajeros que entran y salen del país, proceso en que se manipula grandes volúmenes de información que debe estar interconectada entre las diferentes entidades aeroportuarias del país. A raíz de estas necesidades los especialistas del Centro de Automatización para la Dirección de la Información (CADI), en conjunto con la Universidad de las Ciencias Informáticas (UCI), se plantearon la creación del Sistema de Gestión Integral de Aduanas (GINA), proyecto desarrollado por la Línea Aduana del Departamento Aduana-CCHH de la Facultad 3 de la UCI. En el Sistema GINA se desarrollan un conjunto de aplicaciones atendiendo a los intereses de integración de la AGR basándose en la tecnología cliente- servidor.

La adopción de estándares de diseño de bases de datos facilita las tareas a lo largo del ciclo de vida del desarrollo de un software para asegurar la calidad del mismo. El uso de estos estándares tiene innumerables ventajas que permiten asegurar la legibilidad del modelo de datos en etapas de análisis y diseño. Existen varios estándares internacionales de diseño de base de datos, especialmente para el modelo relacional y en cada proyecto de software se pueden aplicar según las necesidades del mismo.

En el Departamento de Aduana-CCHH específicamente en la Línea Aduana existe el “Estándar de Codificación de Base de Datos”, en el que se reflejan las propias especificaciones y requerimientos del proyecto productivo, que ayudan a obtener un mejor resultado en la tarea de diseño de los esquemas de base de datos, el cumplimiento de este estándar tiene como propósito lograr un diseño con calidad suficiente de una base de datos pero en muchos casos no es cumplido correctamente por los diseñadores por lo que en ocasiones se implantan modelos de datos con deficiencias. Esta línea trabaja con el SGBD Oracle fundamentalmente

por su confiabilidad y la protección segura de los datos., además de ser un requerimiento específico del cliente, la ARG.

Los errores de diseño provocan un atraso considerable en el desarrollo de las aplicaciones, debido a que muchos de ellos son detectados ya en la fase de implementación de la solución y el proceso de rectificación del modelo físico en la base de datos se torna complejo y lento debido a que tiene dos fases: detección y corrección.

El estudio de la problemática antes mencionada ha desencadenado el siguiente **problema a resolver**: ¿Cómo disminuir el tiempo de detección y corrección de errores de diseño, de un esquema de base de datos en Oracle 11g, según el estándar de codificación de base de datos de la Línea Aduana?

Para la investigación se ha definido como **objeto de estudio:** Procesos de detección y corrección de errores de diseño en base de datos.

Persiguiendo como **objetivo general:** Desarrollar una herramienta que permita la detección y corrección de errores de diseño de un esquema de base de datos en Oracle 11g según el estándar de codificación de base de datos de la Línea Aduana, contribuyendo a reducir el tiempo de desarrollo del Sistema de Gestión Integral de Aduanas.

Campo de acción: Procesos de detección y corrección de errores de diseño en los esquemas de base de datos en Oracle 11g según el estándar de codificación de base de datos de la Línea Aduana.

Idea a defender: Si se desarrolla una herramienta de detección y corrección de errores de diseño de un esquema de base datos en Oracle 11g según el estándar de codificación de base datos de la Línea Aduana entonces se reducirá el tiempo de desarrollo del Sistema de Gestión Integral de Aduanas.

Para alcanzar el objetivo propuesto se debe dar cumplimiento a los siguientes **objetivos específicos:**

- Definir el estado actual del proceso de detección y corrección de errores de diseño en los esquemas de base de datos.
- Describir los requisitos que deben estar presentes en la herramienta que se propone desarrollar.
- Diseñar e implementar la solución modelada de manera que cumpla con las necesidades del cliente.
- Validar la solución.

Tareas a desarrollar:

- Elaboración de la fundamentación teórica y la revisión bibliográfica.
- Elicitación de los requisitos.
- Validación los requisitos elicitados.
- Diseño de la base de datos.
- Descripción el modelo de datos.
- Diseño e implementación de las interfaces de usuario.
- Confección del diagrama de clases del diseño.
- Confección de los diagramas de secuencia.
- Implementación de los requisitos elícitados.
- Diseño de los casos de prueba a aplicar.
- Aplicación de las pruebas al sistema.
- Documentación de los resultados de las pruebas realizadas.

Proponiendo como posibles resultados:

- Herramienta para la detección y corrección de errores de diseño en un esquema de base de datos en Oracle 11g.

El trabajo estará estructurado en tres capítulos tal y como se muestra a continuación:

Capítulo I: Se enunciarán los principales estándares aplicables a las bases de datos. Posteriormente se enumeran los principales errores en el diseño de la base de datos de la Línea Aduana. Luego se realizará un estudio de las soluciones existentes, llegando a una solución para el desarrollo de la herramienta según el modelo de desarrollo del Centro de Informatización de la Gestión de Entidades (CEIGE).

Capítulo II: Se realiza la propuesta de la herramienta a desarrollar y se generan los artefactos que propone el modelo de desarrollo del CEIGE.

Capítulo III: Se realizará una descripción de los diagramas de despliegue y de componentes que guían el proceso de implementación del sistema. Se realiza una descripción de las pruebas realizadas.

Capítulo 1: Fundamentación teórica.

1. Introducción.

En el siguiente capítulo se enunciarán los principales estándares aplicables al diseño de bases de datos. Posteriormente se hace referencia al estándar de codificación de base datos de la Línea Aduana y se enumeran los principales errores en el diseño de la base de datos. Luego se realizará un estudio de las soluciones existentes, llegando a una solución para el desarrollo de la herramienta.

1.1 *Algunas reglas generales para el diseño de base de datos relacionales.*

1. Todas las tablas deben poseer una clave primaria.
2. Se deben definir las claves primarias con nombres descriptivos. Por ejemplo empleado_id o id_empleado, no sólo poner identificador (id).
3. Se deben poner nombres a las columnas de manera que, la base de datos pueda ser leída por cualquier persona. Por ejemplo utilizar CUENTA_BANCO en lugar de CTBC.
4. Se debe utilizar una sola columna como clave primaria, las claves primarias de más de una columna son adecuadas para las relaciones de muchos a muchos.
5. Se deben utilizar tablas de referencias en lugar de almacenar valores de gran longitud.
6. Se deben utilizar claves de tipo numérico siempre que sea posible, salvo que por la naturaleza del negocio no lo permita.
7. No se deben incluir columnas cuyos valores estén entrelazados, salvo que una de las columnas sea la clave primaria de la tabla.
8. Se deben poner los nombres de las columnas y tablas relativamente cortos ya que cada tipo de base de datos soporta un número distinto de caracteres. Por ejemplo en Oracle soporta solo hasta 30 caracteres y en Microsoft Access hasta 64.
9. Se debe normalizar tanto como sea posible tratando siempre de llegar al menos a la tercera forma normal.
10. Se debe crear una nueva tabla en las relaciones de N: M, donde la nueva tabla debe poseer los identificadores de las tablas que le dan origen como clave primaria.

11. Deben existir siempre claves foráneas o ajenas, en el caso de tablas con información relacionada, con el objetivo de garantizar la integridad referencial.
12. En las tablas que son herencias la clave del padre debe ser clave primaria y foránea en el hijo.
13. Planear con antelación la transferencia de datos a una base de datos distinta. Por ejemplo, puede que nos interese mover algunos datos de Microsoft Access a Oracle.
14. En una tabla, colocar primero la clave primaria seguida de las claves foráneas.
15. Utilizar el guión bajo (_) para separar palabras.
16. Almacenar solo la información necesaria. Debemos de ser realistas acerca de nuestras necesidades y decidir qué información es realmente necesaria (1).

1.2 Proceso a seguir para el diseño de base de datos.

En el proceso de diseño se desarrollan los siguientes pasos:

- **Determinar la finalidad de la base de datos.**

Es conveniente plasmar en papel el propósito de la base de datos: cómo piensa utilizarla y quién va a utilizarla. Esto permitirá centrarse en los objetivos para lo cual fue creada.

- **Buscar y organizar la información necesaria.**

Se debe reunir toda la información que se desea guardar.

- **Dividir la información en tablas.**

Luego de que se conoce que información se va a guardar se procede a la división de la misma en entidades o temas que luego se convertirán en tablas de la base de datos.

- **Convertir los elementos de información en columnas.**

Luego de que se decide qué información se desea almacenar en cada tabla. Cada elemento se convertirá en un campo y se mostrará como una columna en la tabla.

- **Especificar claves principales.**

Cada tabla debe incluir una columna o conjunto de columnas que identifiquen inequívocamente cada fila almacenada en la tabla. Ésta suele ser un número de identificación exclusivo. En la terminología de bases de datos, esta información recibe el nombre de clave principal de la tabla.

- **Definir relaciones entre las tablas.**

Se examina cada tabla y se decide cómo se van a relacionar los datos de una tabla con las demás tablas. Se pueden agregar campos a las tablas o crear nuevas tablas para clarificar las relaciones según sea necesario.

- **Ajustar el diseño.**

Se realiza un análisis del diseño para detectar posibles errores. Se agregan algunos registros con datos de ejemplo. Se comprueba si se pueden obtener los resultados previstos de las tablas. De no ser positivo el resultado se procede a la realización de ajustes en el diseño.

- **Aplicar las reglas de normalización.**

El siguiente paso del diseño, es la aplicación de reglas de normalización de datos (denominadas a veces simplemente reglas de normalización). Estas reglas sirven para comprobar si las tablas están estructuradas correctamente. El proceso de aplicar las reglas al diseño de la base de datos se denomina normalizar la base de datos o, simplemente, normalización.

1.3 Normalización de base de datos.

La normalización es la expresión formal del modo de realizar un buen diseño. Provee los medios necesarios para describir la estructura lógica de los datos en un sistema de información. El proceso de normalización de una base de datos consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo E-R (entidad-relación) al modelo relacional (2).

El objetivo de normalizar una base de datos relacional es:

- Evitar la redundancia de los datos.
- Evitar problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos (2).

En el modelo relacional es frecuente llamar tabla a una relación, aunque para que una tabla bidimensional sea considerada como una relación tiene que cumplir con algunas restricciones:

- Cada columna debe tener su nombre único.
- No pueden existir dos filas iguales. No se permiten los duplicados.

- Todos los datos en una columna deben ser de el mismo tipo.

La normalización posee 5 formas normales las tres primeras son suficientes para cubrir las necesidades de la mayoría de las bases de datos. El creador de estas 3 primeras formas normales (o reglas) fue Edgar F. Codd (2).

Primera Forma Normal (1FN).

Formalmente: Una relación está en (1FN) si cumple la propiedad de que sus dominios no tienen elementos que, a su vez, sean conjuntos.

- Toda relación normalizada, o sea, con valores atómicos de los atributos.
- La relación no incluye ningún grupo repetitivo (2).

Segunda Forma Normal (2FN).

Una relación R se dice que está en 2FN si está en 1FN y si, y sólo si, los atributos no llaves (ni primarias, ni candidatas) de R, si los hubiese, son funcional y completamente dependientes de la llave primaria de R (2).

Tercera Forma Normal (3FN).

Una relación R 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 (2).

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.

Forma normal de Boyce-Codd (FNBC).

Una tabla está en FNBC sí y sólo sí las únicas dependencias funcionales elementales son aquellas en las que la clave primaria determinan un atributo (2).

Cuarta Forma Normal (4FN).

Está en forma normal de Boyce-Codd y se eliminan las dependencias multi-evaluadas y se generan todas las relaciones externas con otras tablas u otras bases de datos (2).

Quinta Forma Normal (5FN).

Está en cuarta forma normal y toda dependencia-join viene implicada por claves candidatas (2).

1.4 Estándar de Codificación de Base de Datos de la Línea Aduana.

En el Departamento Aduana CC-HH específicamente en la Línea Aduana existe el "Estándar de Codificación de Base de Datos", en el que se reflejan las especificaciones del proyecto productivo, en el cual se plantean normas a seguir para la elaboración estándar de los esquemas dentro de las que se encuentran las tablas, restricciones, secuencias, sinónimos, codificación en SQL[1] , índices, entre otros.

Este estándar tiene como objetivo identificar de forma sencilla las funcionalidades que brinda cada una de las tablas y campos de la base de datos, posibilitando elevar la mantenibilidad del código y sirviendo como punto de referencia para los programadores, ayudándolos a mejorar el proceso de codificación.

Para la presente investigación se tomarán del estándar los elementos relacionados al diseño de base de datos, entiéndase referente a la nomenclatura y el diseño de base datos propios, sin atender los elementos relacionados a la codificación de funciones y procedimientos que también son planteados en el estándar con el objetivo de identificar los posibles errores que se puedan presentar (3).

Para el diseño de las tablas el estándar plantea que se debe tener en cuenta que:

- Los nombres de las tablas se escribirán siempre en mayúsculas y en singular. Esto es debido a que una tabla representa un objeto del sistema, mapeado a la base de datos.

- El nombre de las tablas comenzará con el identificador del esquema al que pertenecen, seguido de un guión bajo y el nombre de la tabla.

- En caso de que el nombre de una tabla conste de más de una palabra, los espacios en blanco se sustituirán por guiones bajos. Además, las palabras seguirán estando todas en singular.

- Solo se utilizarán caracteres alfabéticos cuando por la naturaleza del nombre se necesiten dígitos numéricos. Se prohíbe el uso de caracteres de puntuación o símbolos.

[1] **SQL:** Es un lenguaje de consulta estructurado (Structured Query Language), creado para el acceso a bases de datos relacionales.

- En el caso de la utilización de abreviaturas se deberán utilizar siempre las mismas y dejar reflejadas en un documento cuales son y su significado.

Tabla 1: Abreviaturas.

Abreviatura	**Significado**
DOC	Documento, se utiliza cuando está delante de otra palabra.
F_INICIO	Fecha de inicio, referente a las tablas de control.
F_FIN	Fecha de fin, referente a las tablas de control.
IX	Prefijo para los indices.
PK	Prefijo para las claves primarias.
FK	Prefijo para las claves foráneas.
SEQ	Sufijo para las secuencias.

- En el caso de las especializaciones las tablas se nombrarán con la nueva denominación, el nombre de la especialización no tiene que estar atado a la tabla más general que la precede.

- Las tablas de relación (objetos asociativos, representan relaciones de N: M) deben nombrarse utilizando los nombres de las tablas intervinientes, siguiendo un orden lógico de frase, se utilizarán sólo caracteres en mayúsculas, y se separarán los nombres de las tablas intervinientes en la relación utilizando guión bajo (“_”).

- A las tablas se le escribirá para que son usadas en la pestaña Notes donde se especificará de la siguiente manera:

"La tabla NOMBRE DE LA TABLA se utiliza para conocer (…), esto se conoce a partir del campo (…), además de conocer (…), mediante el campo (…)."

Para el caso de los campos el estándar plantea que se debe tener en cuenta que:

- Los campos clave deben ubicarse al inicio de la definición de la tabla (deben ser los primeros en aparecer). Toda tabla debe poseer al menos un campo clave.
- El nombre del campo clave debe estar compuesto por "id" + nombre de la tabla en singular (para claves no foráneas).
- En el caso de que se trate de claves que son foráneas y primarias a su vez, el nombre de la clave será el de la tabla de donde se genera la clave, el nombre de la tabla donde está la clave foránea o una combinación de ambas tablas.
- Los campos se escribirán en minúsculas y en singular. Solo se escribirán en plural aquellos que por su significado así lo requieran.
- Las palabras de los campos se separarán por guiones bajos, en caso de que sean compuestos.
- Los campos de tipo fecha pueden comenzar con "f_" seguido del nombre del campo. Además se pueden utilizar las nomenclaturas "fecha" para los nombres compuestos.
- Se puede usar la nomenclatura en inglés created_at y deleted_at para los campos que indiquen cuando un registro es creado o eliminado. Estos casos se utilizarán en casos muy puntuales y cuando se apruebe por los diseñadores de base de datos.
- En ninguno de los casos estará reflejado en el nombre del campo, el tipo de dato que se utiliza en el mismo, excepcionalmente en el caso que sea necesario indicar en el negocio dicho tipo de dato, por ejemplo las fechas.
- No se pueden poner caracteres extraños en los nombres de los campos cómo ñ y tilde, estos deben ser sustituidos por símbolos semejantes.

Tabla 2: Caracteres extraños.

Caracter extraño	Posibles Símbolos
ñ	nn
Á	a
É	e
Í	i
Ó	o
Ú	u
Ä	A
@	_

Para el esquema de Tablas de Control[2] (TC) el estándar plantea que:

- Las tablas deben tener invariantemente los siguientes campos:

[2] **Tablas de control:** Son las tablas en las que se almacenan los nomencladores del negocio, es decir, los valores predeterminados del sistema.

Tabla 3: Atributos Obligatorios de las Tablas de Control.

Pos	Nombre	Tipo Dato	Null
01	id_nombre_tabla	number(38, 0)	PK
02	código	varchar(50)	No
03	f_inicio	date	No
04	f_fin	date	No
05	descripcion	varchar(250)	No
06	created_at	date	No
07	deleted_at	date	No

- Los campos deben aparecer en el orden en que se muestra anteriormente. Además de estos pueden tener los demás campos que sean necesarios.
- Las tablas de control pueden tener más de un campo de código de negocio, estos se manejarán con la nomenclatura código_1, código_2, y así sucesivamente.
- Las fechas serán de tipo DATE a no ser que sea necesario otro formato.
- La explicación de para que se utiliza el campo se debe poner en la pestaña Notes. En caso de que se vayan a utilizar ciertos valores predeterminados para el campo se especificará:

“valor1” --> “Significado”

“valor2” --> “Significado”

“valor3” --> “Significado”

- Los nombres de los id auto numéricos siempre deben tener el nombre de la tabla a la que pertenecen sin incluir el nombre del esquema. Incluso en los casos que exista especialización.

Para las relaciones entre tablas de control el estándar plantea que:

- Todas las tablas de control de relaciones deben tener un identificador auto incrementable como clave primaria.
- Siempre y cuando en una tabla de control exista una clave foránea perteneciente a otra tabla de control, se propaga también el código de negocio junto con el identificador de la tabla origen.
- Las tablas que se generan a partir de la relación, muchos a muchos (N: M), entre dos tablas de control tendrán como prefijo "TCR_" seguido de los nombres de las tablas origen.

En el caso de las restricciones el estándar plantea que:

- La estructura de las claves primarias estará conformada de la siguiente manera.

[pk] [Nombre de la tabla] [pk] Todas las claves primarias comienzan con el prefijo pk **[Nombre de la tabla]** Nombre de la tabla a la que pertenece la clave primaria.

- La estructura de nombres de las claves externas estará conformada de la siguiente manera. **[fk] [Nombre de las tablas] [fk]**
- Todas las claves externas comienzan con el prefijo fk. **[Nombre de las tablas]** Nombres de las tablas implicadas separadas por "_". Sin tener en cuenta el prefijo correspondiente al nombre del esquema al que pertenece la tabla o las tablas.

En el caso de los **índices** el estándar plantea que:

- La estructura de nombres de los índices estará conformada de la siguiente manera:

[ix] [Nombre de la tabla] [Columnas]

[ix] Todos los índices comienzan con el prefijo ix.

[Nombre de la tabla] Nombre de la tabla a la que hace referencia el índice.

- Para el caso de los índices que se crean para las claves primarias no se pondrá el nombre del campo clave al final del mismo.

En el caso de las secuencias el estándar plantea que:

- El nombre de las secuencias que se crean para los id auto incrementables debe estar formado por el nombre de la tabla seguido del sufijo "_SEQ".
- Todas las tablas de control llevan secuencias, excepto aquellas que su clave primaria sea el resultado de una herencia, las que poseen más de un campo como clave primaria y que sean resultado de una relación de muchos a muchos.

Para el caso de los sinónimos:

- Los sinónimos siempre tendrán el mismo nombre de la tabla a la que están asociados, pero sin el nombre del esquema al que pertenece la misma.
- Los sinónimos deben ser públicos.

1.5 Diseño e implantación de la base de datos en la Línea Aduana.

Para el diseño e implantación de una base de datos el primer paso a desarrollar es la captura de requisitos del analista según las especificidades del cliente. Luego de que se realice el análisis pertinente, se procede al diseño de la solución donde el diseñador del sistema realiza un diagrama de clases del diseño del cual se deriva el modelo de datos. Posteriormente este modelo es enviado al equipo de base de datos donde es revisado y aprobado para su posterior implantación.

Este proceso generalmente no se lleva a cabo con la calidad requerida, en la mayoría de los casos el tiempo para la implantación de dichos esquemas es reducido y son ignorados algunos de los pasos que se mencionaron anteriormente y en otros casos no se tiene pleno conocimiento del estándar establecido en la línea sobre la base de datos.

Generalmente los errores son detectados por los programadores del sistema lo que provoca un atraso ya que cuenta con dos fases: detección y corrección. Se tomó como muestra el esquema Solicitudes para una revisión manual donde fueron encontrados 520 errores de ellos 205 menos críticos, 223 leves y 92 de tipo críticos. Se realizó un levantamiento de los errores hallados en un documento para su posterior corrección, el período de identificación demoró alrededor de 8 horas y el de corrección 16 horas, afectando un día el desarrollo del proyecto.

1.5.1 Ventajas y desventajas del uso de estándares de bases de datos.

El empleo de estándares trae considerables aportes a las bases de datos:

- Asegura la legibilidad del modelo de datos en etapas de análisis y diseño, inclusive para personas que no están relacionadas con el ambiente informático.
- Facilita la portabilidad entre motores de bases de datos, plataformas y aplicaciones.
- Facilita la tarea de los programadores en el desarrollo de los sistemas.
- Contribuye al buen funcionamiento de la base de datos.

La principal desventaja que existe en cuanto al uso de estándares es que:

- No existe un estándar único para el diseño de base de datos. Los estándares son implantados según las necesidades que se presenten (4).

1.6 Principales errores que afectan el diseño de la base de datos en la Línea Aduana.

Entre los principales errores identificados en la Línea Aduana se encuentran:

1. El nombre de las tablas excede los 26 caracteres imposibilitando la creación de secuencias.
2. El nombre de las tablas no se encuentra en singular.
3. El nombre de las tablas no se encuentra en mayúscula.
4. Las tablas no poseen el prefijo en su nombre indicando al esquema al que pertenecen.
5. Los nombres de las tablas cuando son compuestos poseen espacio entre las palabras que lo componen o estas se encuentran fusionadas.
6. Los nombres de las tablas poseen caracteres extraños.
7. Los nombres de las tablas que son especializaciones poseen el mismo nombre de la tabla origen.
8. Los nombres de las tablas que surgen de relaciones de N: M no poseen los nombres de las tablas que les dieron origen.
9. Las tablas no poseen clave primaria.
10. Las tablas no poseen la descripción para lo cual son creadas.
11. Los atributos que son claves primarias carecen del identificador “id_” o poseen el prefijo del esquema.
12. Los campos claves no aparecen en la primera posición de la tabla.
13. El nombre de los campos no se encuentra en singular.
14. Los campos cuando son compuestos poseen espacio entre las palabras que lo componen o estas se encuentran fusionadas.
15. Los campos se encuentran en mayúscula.

16. Los campos poseen caracteres extraños.
17. Los campos created_at y deleted_at se encuentran mal escritos.
18. Existen campos cuyo nombre representa un tipo de dato.
19. El nombre de los sinónimos de las TC no poseen el mismo nombre de las tablas a las cuáles pertenecen.
20. Las tablas TC creadas a partir de relaciones de N: M carecen del prefijo “TCR”.
21. Las tablas TC no poseen el campo código.
22. Las tablas TC no poseen el campo created_at.
23. Las tablas TC no poseen el campo deleted_at.
24. Las tablas TC no poseen el campo id_nombre_tabla.
25. Las tablas TC no poseen el campo f_inicio.
26. Las tablas TC no poseen el campo f_fin.
27. Las tablas TC(R) no poseen un id auto-incrementable como clave primaria.
28. Los campos de las tablas TC no poseen la descripción de su objetivo.
29. Los identificadores de las TC(R) no poseen el mismo nombre de la tabla.
30. Cuando dos TC se relacionan, y una de sus tablas obtiene la clave foránea no posee el código de la tabla padre.
31. Las tablas que poseen las claves foráneas y primarias a la vez poseen mal la combinación del nombre.
32. Todas las tablas que en un esquema tienen el permiso (grant) de lectura, actualización o eliminación para otro usuario, debe tener sinónimo público.
33. Las tablas que por la naturaleza del negocio requieren secuencia no la poseen.
34. Las restricciones que son claves primarias no poseen la estructura adecuada.
35. Las restricciones que son claves foráneas no poseen la estructura adecuada.
36. Los índices no poseen la estructura adecuada.
37. Las secuencias no poseen el sufijo “_SEQ”.

38. Los sinónimos deben ser públicos.
39. Las tablas que poseen sinónimos, el nombre del sinónimo no es el mismo nombre que el de la tabla a la cual pertenece.
40. Existen secuencias que no están asociadas a ninguna tabla.
41. En las tablas de negocio, las relaciones de N-M deben tener como llave primaria y foránea las llaves de las tablas que la originaron.

1.6.1 Clasificación de los errores según su impacto.

Los errores que son encontrados en el diseño de un esquema de base de datos, pueden ser agrupados en tres tipos: críticos, menos críticos y leves. Los errores que son agrupados en críticos son aquellos que afectan de forma directa en el acceso a los datos de la base de datos, son errores que interfieren en el funcionamiento de la base de datos. Por ejemplo: si las TC no poseen sinónimos públicos no pueden ser accedidas desde otros esquemas impidiendo el acceso a los datos ahí almacenados, datos que son valores predeterminados del sistema.

Los errores menos críticos son aquellos que afectan la legibilidad del modelo de datos. Por ejemplo: si el nombre de las tablas no se encuentra en mayúscula el gestor no lo reconoce. Por otro lado si la clave primaria de las tablas no posee el prefijo “id_” no se sabría a simple vista que ese atributo es la clave primaria de la misma.

Los errores leves son aquellos que infringen el proceso de diseño, pero no afectan el rendimiento de la base de datos. Por ejemplo: las tablas que no posean la descripción para lo cual fueron creadas, esto solo afecta a la persona que se encuentre leyendo el diseño ya que no sabría para qué sirve la tabla, sin embargo la tabla puede ser consultada y modificada sin problema alguno. Por otro lado que la clave primaria no se encuentre en la primera posición de la tabla, es un error pero no afecta el acceso al campo ni a la tabla.

Agrupación de los tipos de errores según su impacto para cada tipo de objeto.

Para las Tablas los errores son los siguientes:

Errores Críticos:

- El nombre de las tablas excede los 26 caracteres.
- Los nombres de las tablas poseen caracteres extraños.

- Las tablas deben poseer al menos una clave primaria.
- Los atributos que son claves primarias carecen del identificador “id_” y poseen el prefijo del esquema.
- Los nombres de las tablas que son especializaciones poseen el mismo nombre de la tabla origen y la clave primaria de la misma puede ser la del padre, una propia de ella o una combinación de las dos tablas.
- Las tablas que poseen las claves foráneas y primarias a la vez poseen mal la combinación del nombre.
- Todas las tablas que en un esquema tienen el permiso (grant) de lectura, actualización o eliminación para otro usuario, debe tener sinónimo público.
- El nombre de los sinónimos de las TC no poseen el mismo nombre de las tablas a las cuales pertenecen.
- Las tablas que por la naturaleza del negocio requieren secuencia no la poseen.
- Las restricciones que son claves primarias no poseen la estructura adecuada
- Las restricciones que son claves foráneas no poseen la estructura adecuada.
- Los índices no poseen la estructura adecuada.
- Las tablas TC(R) no poseen un id auto-incrementable como clave primaria.
- Los campos poseen caracteres extraños.
- Los campos created_at y deleted_at se encuentran mal escritos.
- Las claves foráneas tienen que pertenecer a una clave única para garantizar la unicidad de los registros.

Errores Menos Críticos:

- El nombre de las tablas no se encuentra en mayúscula.
- Las tablas no poseen el prefijo en su nombre indicando al esquema al que pertenecen.
- Los nombres de las tablas cuando son compuestos poseen espacio entre las palabras que lo componen o estas se encuentran fusionadas.

- Los nombres de las tablas que surgen de relaciones de N: M no poseen los nombres de las tablas que les dieron origen.
- Las tablas TC creadas a partir de relaciones de N: M carecen del identificador “TCR".
- Los identificadores de las TC(R) no poseen el mismo nombre de la tabla.
- Cuando dos TC se relacionan, y una de sus tablas obtiene la clave foránea no posee el código de la tabla padre.
- Los campos cuando son compuestos poseen espacio entre las palabras que lo componen o estas se encuentran fusionadas.
- Existen campos cuyo nombre representa un tipo de datos.
- Las tablas que poseen sinónimos, el nombre del sinónimo no es el mismo nombre que el de la tabla a la cual pertenece.
- Los campos created_at y deleted_at no se encuentran correspondidos con los registros que son creados o actualizados.
- Los campos de las TC no aparecen en el orden requerido (Ej. fecha_fin no puede aparecer primero que la fecha_inicio).
- Las tablas que son especializaciones poseen como llave foránea otra que no es la de su padre.
- Las tablas que son especializaciones poseen como llave primaria otra que no es la de su padre.
- La llave primaria de las tablas que son especializaciones se encuentra mal escrita.

Errores Leves:

- El nombre de las tablas no se encuentra en singular.
- Las tablas no poseen la descripción para lo cual son creadas.
- Los campos de las tablas TC no poseen la descripción de su objetivo.
- Los campos claves no aparecen en la primera posición de la tabla.

- El nombre de los campos no se encuentra en singular.
- Los campos se encuentran en mayúsculas.

Para las Restricciones los errores son los siguientes:
Errores Menos Críticos:

- Las restricciones que son claves primarias no poseen la estructura adecuada.
- Las restricciones que son claves foráneas no poseen la estructura adecuada.

Para las Secuencias los errores son los siguientes:
Errores Críticos:

- Las secuencias no poseen el sufijo "_SEQ".
- Existen secuencias que no están asociadas a ninguna tabla.

Para las Sinónimos los errores son los siguientes:
Errores Críticos:

- Los sinónimos deben ser públicos.

1.7 Herramientas CASE[3] que comprueban estándares de bases de datos.

Las herramientas CASE son una clase de software que automatiza muchas de las actividades que intervienen en el proceso de desarrollo. Ayudan en el proceso del diseño del proyecto, en el cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. Las herramientas automatizadas pueden facilitar la coordinación de las actividades de desarrollo de software mediante el uso de almacenes de datos o repositorios (5).

A continuación se realiza un análisis de algunas de las herramientas CASE que se utilizan para el diseño de base de datos y que de alguna forma aplican estándares.

[3] **CASE:** por sus siglas en inglés, Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador

1.7.1 ER/Studio.

Está equipado para crear y manejar diseños de bases de datos funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación, fácil creación de reportes, reingeniería inversa de base de datos, documentación basada en HTML[4] y posee un repositorio para el modelado.

Cuando en el ER/Studio se genera el modelo físico, automáticamente se aplican algunas reglas generales del diseño de base de datos entre las que se encuentran: en las herencias el identificador del padre pasa al hijo, en las relaciones de N:M se crea una nueva tabla con el identificador de los dos padres, restringe que el nombre de las tablas que no debe poseer más de 30 caracteres.

Además permite establecer una plantilla a los modelos realizados, la misma es especificada por el usuario y es aplicable a un esquema ya diseñado o a uno que este por diseñar.

Esta herramienta aunque posee características muy ventajosas para el diseño de la base de datos no satisface los requisitos especificados en el Estándar de Codificación de Base de Datos de la Línea Aduana por lo que no es una solución viable para evaluar el estado de los esquemas de base de datos de la línea (6).

1.7.2 DBDesigner.

Sistema totalmente visual de diseño de bases de datos, que combina características y funciones profesionales con un diseño simple, claro y fácil de usar, a fin de ofrecer un método efectivo para gestionar bases de datos.

Permite construir una base de datos a través de una interfaz intuitiva, donde se tiene una representación visual de las tablas y relaciones que figuran en el proyecto. Dispone de detallados manuales de uso. El diseñador puede ver rápidamente los campos de una tabla. Puede importar a partir de bases de datos existentes y guardar el proyecto en su formato

[4] **HTML:** (por sus siglas en ingles Hyper Text Markup Language) es el lenguaje que permite escribir texto de forma estructurada, y que está compuesto por etiquetas, que marcan el inicio y el fin de cada elemento del documento.

original (XML[5]) para mantener toda la información. Debido a su arquitectura, DBDesigner es fácilmente extensible para trabajar con varios servidores de base de datos. Por defecto viene con 2 conectores: uno para PostgreSQL y el otro para MySQL.

De cierta manera DBDesigner aplica una regla la cual consiste en tomar el primer atributo que se incluye en la tabla como clave primaria de la misma, lo que posibilita una búsqueda eficiente de los datos guardados ya que siempre existe esta clave en todas las tablas (7).

Esta herramienta tampoco posibilita la evaluación de un esquema ya que a pesar de que aplica la regla explicada anteriormente, no cumple con los requerimientos que se especifican en el "Estándar de Codificación de Base de Datos" de la Línea Aduana.

1.7.3 EasyCASE.

Herramienta que permite automatizar las fases de análisis y diseño dentro del desarrollo de una aplicación, para poder crear las aplicaciones eficazmente, desde procesamiento de transacciones a la aplicación de bases de datos de cliente/servidor, así como sistemas de tiempo real.

Posee disímiles características entre las que se encuentran la generación de esquemas de base de datos e ingeniería reversa, captura los detalles de diseño de un sistema y comunica las ideas gráficamente, para que sean fáciles de ver y entender, permite la creación y mantención de diagramas de flujo de datos, diagramas de entidad-relación, mapas de estructura y permite la corrección avanzada de dichos diagramas lo que posibilita revisiones generales de los mismos. Soporta una gama amplia de metodologías estructuradas, permitiendo escoger los métodos más apropiados para realizar las tareas (8).

Aunque la herramienta posee características que la hacen, muy útil en las fases de análisis y diseño, y aunque brinda la posibilidad de corrección y mantención de diagramas no es posible utilizar en el departamento ya que no cumple con muchas de las especificaciones que este requiere.

---

[5] **XML:** (siglas en inglés de eXtensible Markup Language) consiste en un conjunto de reglas para representar información en una forma fácilmente procesable por un ordenador.

1.7.4 Oracle Designer.

Herramienta de software para analizar los requerimientos de negocios y para diseñar y generar sistemas cliente/servidor que satisfagan tales requerimientos.

Ofrece una interfaz intuitiva basada en el explorador, que es capaz de administrar las bases de datos, crear tablas, vistas y otros objetos de bases de datos, importar, exportar y visualizar datos de tablas, ejecutar scripts de SQL y generar informes. Soporta transacciones, es estable, escalable y multiplataforma. Para su desarrollo utiliza PL/SQL, el cual es un lenguaje de quinta generación, bastante potente para tratar y gestionar la base de datos. Gestiona el repositorio y sus usuarios, brinda asistencia en instalaciones, actualizaciones y comprobaciones del repositorio. Hace copias de seguridad y restaura objetos del repositorio (9).

Aunque la herramienta posee características que la hacen, muy útil en el momento del diseño, no posee un módulo que sea capaz de evaluar un esquema.

1.7.5 System Architect.

Provee soporte para técnicas variadas para el desarrollo de sistemas de información. Permite generar automáticamente plantillas de código en varios lenguajes de programación y también esquemas de implementación para gestores de bases de datos relacionales.

Posee un repositorio único que integra todas las herramientas, y metodologías usadas. Conecta directamente al diccionario de datos, los elementos asociados, comentarios, reglas de validaciones y normalización. Posee control automático de diagramas y datos, normalizaciones y balanceamiento entre diagramas "Padre e Hijo", además de balanceamiento horizontal, que trabaja integrado con el diccionario de datos, asegurando la compatibilidad entre el Modelo de Datos y el Modelo Funcional. Traduce modelos de entidades a partir de la enciclopedia.

Posee esquemas de seguridad e integridad a través de contraseñas que posibilitan el acceso al sistema en diversos niveles. Posee un módulo específico para Ingeniería Reversa desde las Bases de Datos SQL. Logra leer bases de datos y construir el modelo lógico o físico (diagrama), alimentando su diccionario de datos con las especificaciones de las tablas y de sus elementos de datos, incluyendo las relaciones entre tablas y su cardinalidad (10).

Esta herramienta no evalúa un esquema a pesar de que posee características que la hacen, muy útil en el diseño de una base de datos.

1.7.6 Herramienta para la identificación de errores de diseño de un esquema de base de datos en Oracle 11g.

La herramienta fue desarrollada en la Facultad 3 de la Universidad de las Ciencias Informáticas y analiza los principales errores que se pueden cometer en el proceso de diseño de un esquema de base de datos según el "estándar de codificación de base de datos" de la Línea Aduana logrando una disminución del tiempo de detección de los errores. Esta herramienta aunque permite realizar el proceso de identificación a través de ella no se pueden corregir los errores por lo que se necesita realizar una nueva versión de este sistema que permita la detección y corrección de los posibles errores que se pueden cometer en el proceso de diseño de un esquema, el mismo debe estar guiado por el modelo del CEIGE.

1.8 Tecnología a utilizar.

1.8.1 Java como lenguaje de programación.

Java es un lenguaje de programación de alto nivel desarrollado por Sun Microsystems a principios de la década del 90. La mayor parte del código Java se encuentra bajo Licencia Pública General (GPL) de GNU[6] excepto las bibliotecas de clases de Sun.

Entre las características principales de este lenguaje se encuentran las siguientes:

- Orientado a objetos: Soporta las características esenciales del paradigma de la programación orientada a objetos: encapsulamiento[7], herencia[8] y polimorfismo[9].

[6] **GNU:** Proyecto informático creado para la construcción de sistemas operativos libres.

[7] **Encapsulamiento:** Mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el objeto se implementa.

[8] **Herencia:** Mecanismo basado en clases, por medio del cual una clase se deriva de otra de manera que extiende su funcionalidad.

[9] **Polimorfismo:** se refiere a la posibilidad de enviar un mensaje a un grupo de objetos cuya naturaleza puede ser heterogénea.

- Multiplataforma: El mismo código Java que funciona en un sistema operativo, funciona en cualquier otro que tenga instalada la máquina virtual de Java.

La versatilidad y eficiencia de la tecnología Java, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para el desarrollo de aplicaciones distribuidas.

Se decide utilizar Java como lenguaje de programación para el desarrollo de la aplicación ya que el mismo posee flexibilidad, portabilidad y es sencillo de aprender. Por otra parte es un lenguaje multiplataforma. Brinda soporte de conexión a múltiples gestores de base de datos, se integra particularmente con Oracle ya que son desarrolladas por la misma compañía (11).

1.8.2 Entorno de Desarrollo Integrado.

Un Entorno de Desarrollo Integrado, llamado también IDE[10], es un programa informático compuesto por un conjunto de herramientas de programación que le facilitan al usuario editar, compilar, depurar códigos además de construir de manera rápida interfaces gráficas de usuarios. Puede ser usado para uno o varios lenguajes de programación (12).

1.8.2.1 NetBeans 7.1 como entorno de desarrollo.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Es una aplicación de código abierto diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java, 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 soporta la instalación de módulos para ampliar las funcionalidades del IDE (13).

---

[10] **IDE:** Por sus siglas en inglés de Integrated Development Environment.

Se seleccionó NetBeans como IDE pues es el que mejor soporte brinda a las tecnologías escogidas para el desarrollo de la solución, es software libre y multiplataforma. Además de poseer disímiles características entre las que se encuentra el completamiento de código, el soporte para varios lenguajes, soporte para varios plugins que lo hace un IDE firme y confiable.

1.8.3 PostgreSQL 8.0 como gestor de base de datos.

PostgreSQL 8.0 es un Sistema de Gestión de Base de Datos Relacional (SGBDR) orientada a objetos, libre, gratuito y de código abierto (Open Source). Mediante un sistema denominado MVCC[11] PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos (14).

Se escoge PostgreSQL como gestor de base de datos ya que posee varias características que lo hacen ser un gestor potente en cuanto a seguridad de los datos, es multiplataforma, soporta una capacidad de almacenamiento en el orden de los TB (terabytes), posee gran escalabilidad y por tanto puede soportar una mayor cantidad de peticiones concurrentes y tiene la capacidad de comprobar la integridad referencial. Posee una licencia GNU, es software libre y se ajusta muy bien a la independencia tecnológica que la universidad está aplicando.

1.8.4 PgAdmin III como herramienta para administrar PostgreSQL.

En PgAdmin III se puede trabajar con los objetos de la base de datos que serán utilizados para el proceso de detección de errores, entre estos objetos se encuentran columnas, restricciones, dominios, funciones, índices, esquemas, secuencias, tablas entre otros. Una característica interesante de PgAdmin III es que, cada vez que se realiza alguna modificación en un objeto, escribe la(s) sentencia(s) SQL correspondiente(s) (15).

Se seleccionó PgAdmin III pues es el que mejor soporte brinda a las tecnologías escogidas para la administración de la base de datos con que se va a trabajar en el desarrollo de la solución, es software libre y multiplataforma. También incorpora funcionalidades para realizar consultas, examinar su ejecución y trabajar con los datos.

[11] Control de concurrencia para múltiples versiones.

1.8.5 DB Designer 4.0 para diseñar la base de datos.

DBDesigner 4.0 es un sistema visual para el diseño de base de datos que integra diseño, modelado, creación y mantenimiento en un único entorno sin fisuras. Combina características profesionales y una interfaz de usuario clara y sencilla de ofrecer la forma más eficiente para gestionar sus bases de datos. DBDesigner, desarrollado por FabForce, es una aplicación para el diseño visual de bases de datos. Permite desarrollar una base de datos teniendo en cuenta el diseño y las funcionalidades independientemente del servidor/Sistema Gestor de Bases de Datos que se utilizará. Es un proyecto Open Source disponible para Microsoft Windows NT/XP/Vista/7 y Linux KDE/Gnome. Es distribuido con licencia GPL. Es capaz de trabajar con MySQL y Oracle (16).

Se seleccionó DBDesigner 4 pues además ser software libre y multiplataforma posee una interfaz clara y fácil de usar, a fin de ofrecer un método efectivo para gestionar bases de datos.

1.9 Modelo de desarrollo de software.

El proceso de informatización de la sociedad cubana ha propiciado el aumento del uso de herramientas informáticas en los principales sectores del país. En tal sentido, la Universidad de las Ciencias Informáticas (UCI) desempeña un rol protagónico desde su surgimiento mediante la producción de soluciones y servicios informáticos. Adscrito a la Facultad 3 de la propia universidad se encuentra el Centro de Informatización de la Gestión de Entidades (CEIGE). Dicho centro productivo dirige sus resultados hacia la esfera de la gestión de entidades. La producción se concentra en el desarrollo de proyectos, por lo que se hace necesario contar con un modelo estandarizado, que establezca las distintas fases por las que se debe transitar y el conjunto de artefactos a generar en cada una de ellas. Teniendo como precedente dicha necesidad y en colaboración con las distintas líneas de desarrollo donde se ejecutan cada uno de estos proyectos se confeccionó el **Modelo de desarrollo de software para el CEIGE,** el cual detalla el ciclo de vida de sus proyectos (17).

1.9.1 Ciclo de vida de proyectos del CEIGE

Para el ciclo de vida de los proyectos del CEIGE y posterior especificación se tienen en cuenta las fases y actividades por áreas de procesos que plantea el nivel dos de CMMI[12] establecido en la UCI. En la figura se pueden apreciar el total de fases por las que pueden transitar los proyectos del centro, pudiendo realizar iteraciones a partir de la fase de Modelado del negocio. El conjunto de fases propuestas abarca el total de acciones que se realizan en las distintas líneas de desarrollo para la elaboración del servicio o producto final, sin embargo, se debe adaptar a las características particulares del proyecto que puede que no realice determinada fase, así como la elaboración de determinados artefactos del total aquí definido. Para la elaboración de este sistema se llegará hasta la fase de pruebas internas (17).

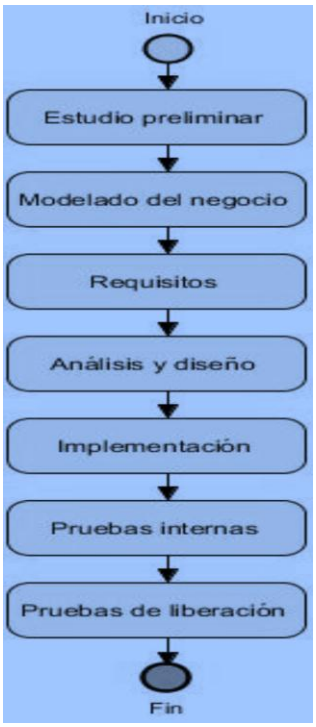


Figura 1. Fases del ciclo de vida de los proyectos del CEIGE.

[12] **CMMI (**por sus siglas en inglés Capability Maturity Model Integration**):** es un modelo de calidad del software que clasifica las empresas en niveles de madurez. Estos niveles sirven para conocer la madurez de los procesos que se realizan para producir software.

1.10 Conclusiones

En este capítulo se identificaron los errores que afectan a la base de datos de la Línea Aduana según el estándar de codificación establecido en esta. Se analizaron las principales herramientas CASE que apoyan la realización de modelos de datos y que cumplen de alguna manera con estándares de diseño. Además se describieron las herramientas seleccionadas para la solución del problema planteado.

Capítulo 2: Análisis y diseño de la propuesta de solución.

2. Introducción.

En este capítulo se realiza un levantamiento de los requisitos del sistema a implementar, además se muestra una propuesta de solución del sistema. Se diseñan los artefactos generados por el modelo de desarrollo que ayudan a construir dicha propuesta.

2.1 Descripcion de la propuesta de la solución.

Se desea una herramienta que ayude a identificar y corregir los errores en el diseño de un esquema; el esquema será seleccionado por el usuario, en este caso el administrador de base de datos del departamento. El sistema utilizará una arquitectura cliente-servidor, el mismo se conectará al servidor Oracle donde se encuentra la base de datos a ser examinada, extrae toda la información a revisar para luego almacenarla en la base de datos del sistema y que pueda ser consultada, evaluada y corregida por el usuario en cualquier momento.

Entre las principales funcionalidades de la herramienta se encuentran la revisión total o parcial de cada uno de los objetos del esquema, la evaluación de los esquemas, los reportes de errores que se generan una vez realizada cada revisión, la corrección de los errores encontrados en el esquema así como la exportación de los mismos, adicionar, modificar y eliminar conexiones a las bases de datos a ser analizadas e insertar, listar y eliminar tipos de errores.

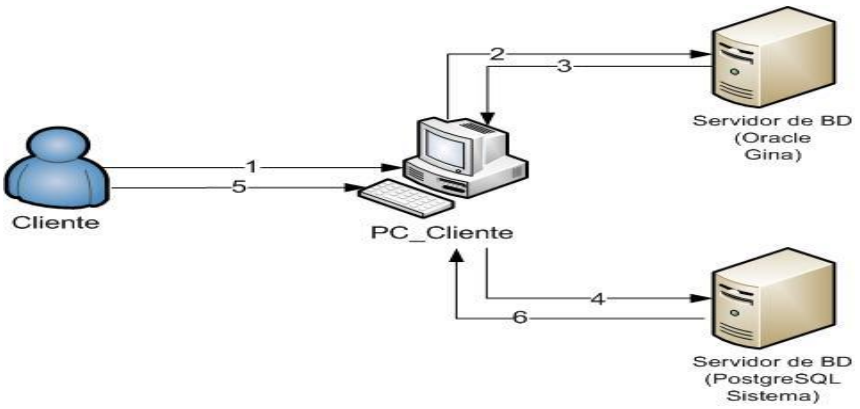


Figura 2. Propuesta de solución.

2.2 Modelo Conceptual.

Un Modelo Conceptual explica cuales son y cómo se relacionan los conceptos relevantes en la descripcion del problema, se representa utilizando un conjunto de diagramas de clases en los que no se define ninguna operación y proporciona una perspectiva conceptual a través de:

- Objetos o clases conceptuales.
- Asociaciones entre las clases conceptuales.
- Atributos de las clases conceptuales (18).

A continuación se muestra el modelo conceptual del sistema:

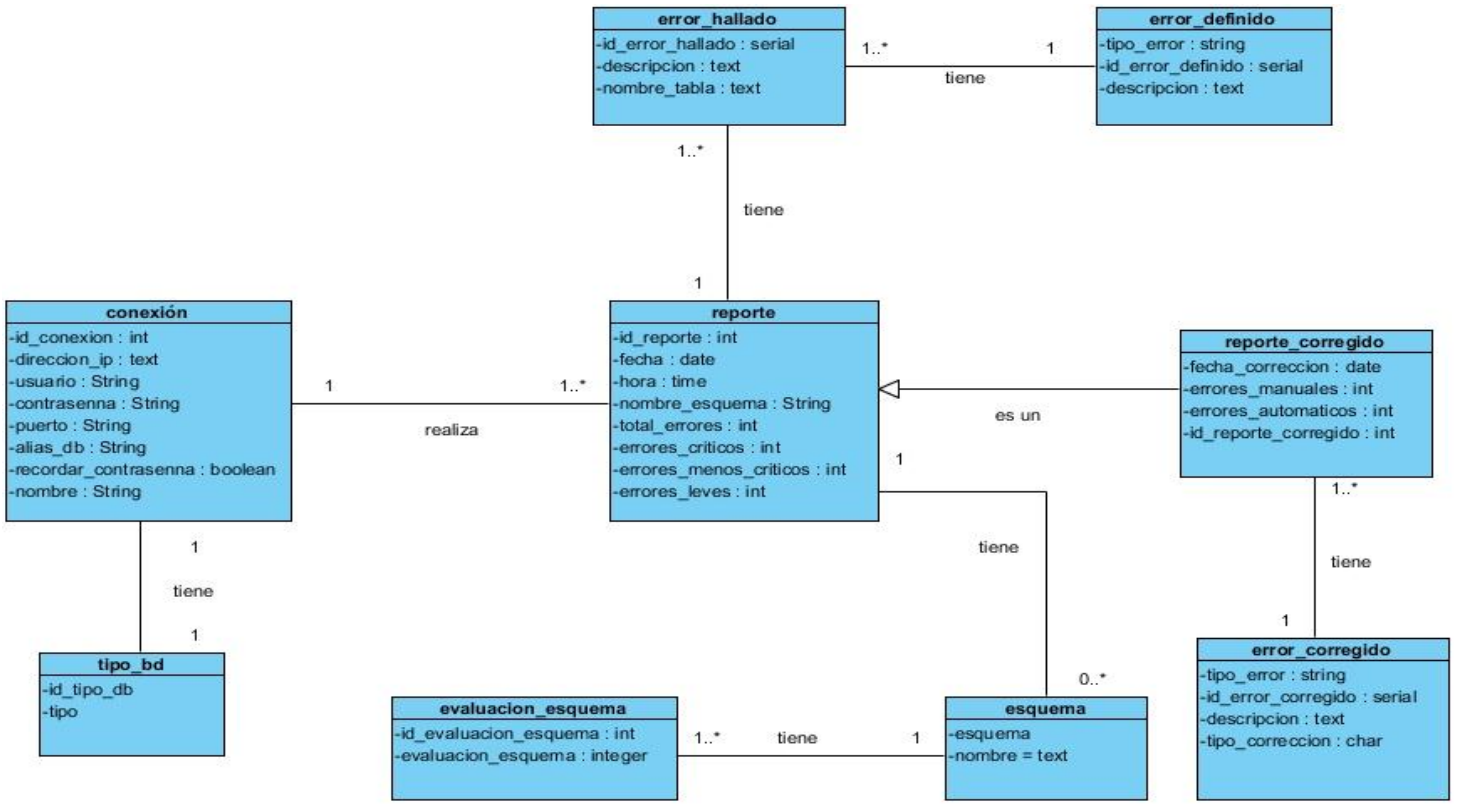


Figura 3. Modelo Conceptual.

2.2.1 *Descripción de los conceptos del modelo conceptual.*

- conexión: acceso a los datos de la base de datos, contiene los campos necesarios para conectarse a la base de datos a analizar.
- reporte: informe que transmite información al usuario, será mostrado luego del proceso de revisión de un esquema.
- esquema: colección de objetos de la base de datos, los objetos del esquema son estructuras lógicas que hacen referencia directa a los datos de la base de datos (tablas, índices, restricciones, secuencias, sinónimos, entre otros).
- error_hallado: error identificado en el proceso de revisión de un esquema.
- error_definido: error definido en crítico, menos crítico o leve, atendiendo a su impacto en el funcionamiento de la base de datos.
- reporte_corregido: informe que transmite al usuario la información luego de realizarse el proceso de corrección de un reporte.
- error_corregido: muestra la corrección que fue realizada al error_definido.
- tipo_bd: contiene las distintas conexiones a la bases de datos, entre las que se encuentran Oracle y PostgreSQL.

2.3 Técnicas de Captura de Requisitos.

La captura de requisitos es la actividad mediante la cual se extraen las necesidades del sistema, es el comienzo de cada ciclo de desarrollo. Las técnicas de captura de requisitos son las que posibilitan que la extracción sea efectiva y que la aceptación del sistema dependa de cuán bien el sistema satisfaga las necesidades del cliente. En la captura de requisitos de esta herramienta se aplicó la siguiente técnica:

- **Tormenta de ideas:** Es un modelo que se utiliza para generar ideas en grupo obteniendo la mayor cantidad de requerimientos del sistema que se pueda. Se realizaron 2 reuniones con la participación del tutor y personal del proyecto donde se mostraron ideas de forma libre las cuales permitieron lograr el proceso de captura de requisitos (19).

2.4 Requisitos.

La elicitación de requisitos es el proceso de descubrir los requerimientos para un sistema a través de la comunicación con los clientes, usuarios del sistema y otras personas que tengan algún tipo de interés y conocimiento sobre el producto a desarrollar. Esta captura de requerimientos a nivel de negocio y de producto garantiza que el proyecto alcance los objetivos dentro de limitaciones tales como la duración, el ámbito, los recursos o la calidad, se ayuda a los usuarios a entender qué es lo que quieren, qué es lo que necesitan, cuáles son las restricciones y alternativas así como ventajas y desventajas de cada una. Un requisito no es más que: “Una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema (20) .”

Los requisitos son clasificados en funcionales, no funcionales y del dominio según el profesor Ian Sommerville. Los requisitos funcionales y no funcionales son los más importantes para el desarrollo de un software con calidad, ya que son las necesidades y capacidades que debe poseer el sistema a desarrollar. Los requisitos funcionales de un sistema describen lo que este debe hacer. “Son declaraciones de los servicios que debe brindar, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. (20)”

Los requisitos no funcionales: “Son restricciones de los servicios o funciones ofrecidos por el sistema.” No se refieren a funcionalidades específicas, sino a las propiedades emergentes de este. Los requisitos del dominio: se derivan del dominio de aplicación del sistema más que de las necesidades específicas de los usuarios. “Reflejan las características y restricciones de ese dominio. (20)”

2.4.1 Requisitos Funcionales (RF).

RF1 Gestionar conexión a la base datos: Permite gestionar la conexión a la base datos a la cual se conectará el usuario para ser analizada, presenta funcionalidades para crear, modificar, eliminar y listar las conexiones.

RF2 Revisar esquema completo: Permite revisar los principales objetos de un esquema en la base de datos. Posibilita la revisión de los posibles errores que puedan presentar las restricciones, tablas, índices, secuencias y sinónimos de un esquema determinado por el usuario.

RF3 Evaluar esquema: Le muestra al usuario el estado en que se encuentra un esquema seleccionado por el usuario.

RF4 Corregir esquema: Permite al usuario corregir los errores existentes en un esquema seleccionado por el usuario. Posibilita realizar una corrección automática y otra manual a aquellos errores que así lo requieran.

RF5 Listar errores: Le muestra al usuario un listado de errores indicándole de forma detallada que es lo que debe ser revisado para ser corregido.

RF6 Gestionar reportes: Permite gestionar los reportes que serán mostrados al usuario, posibilita generar, eliminar y mostrar un reporte al usuario.

RF7 Identificar esquemas de la base de datos a analizar: El usuario solicita la información del esquema que desee evaluar.

RF8 Gestionar tipo de error: Permite gestionar un tipo de error, presenta funcionalidades para insertar, modificar, eliminar y exportar tipo de error.

RF9 Exportar reportes: Permite al usuario exportar un reporte con todos sus detalles.

RF10 Listar Objetos del esquema: Permite listar todos los objetos de un esquema seleccionado por el usuario, lista tablas, índices, restricciones, sinónimos y secuencias.

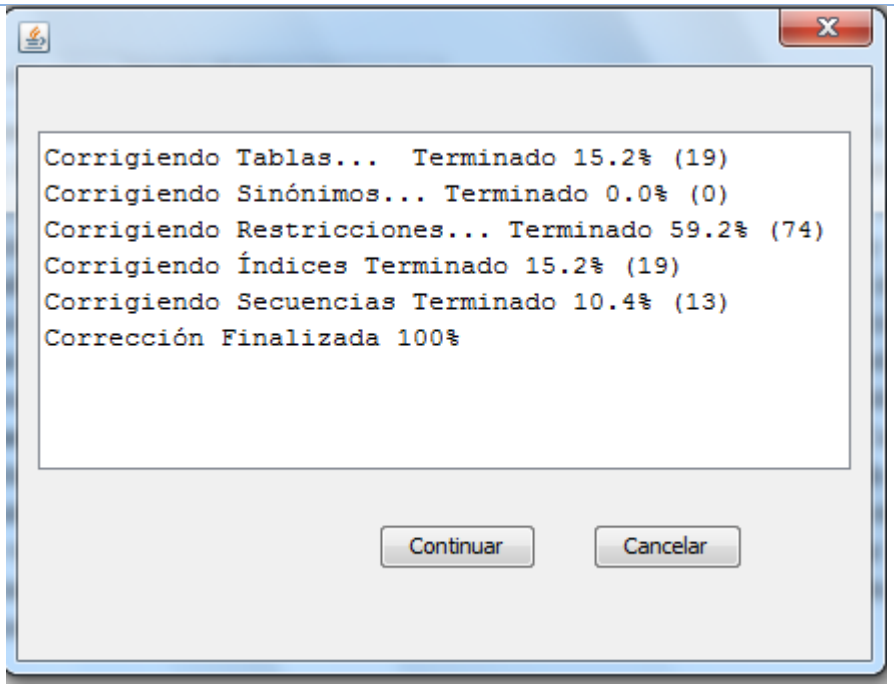
RF11 Mostrar opciones de corrección al usuario: Muestra al usuario las opciones existentes para realizar la corrección.

RF12 Mostrar breve introducción del sistema: Muestra al usuario una breve introducción de las principales funciones del sistema.

2.4.1.1 Especificación del requisito Corregir esquema.

Precondiciones	Debe existir una conexión a la base de datos a revisar y estar seleccionado un esquema.
Flujo de eventos	
Flujo básico Procesar Información de la Remisión de Entrada	
1	Se selecciona la opción corregir esquema en la interfaz principal.
2	El sistema mostrará un listado con los reportes realizados al esquema seleccionado.
3	El usuario selecciona el reporte que desea corregir.
4	El sistema muestra la interfaz “Corregir todos los objetos”.
5	El usuario selecciona la opción continuar.
6	El sistema muestra el resultado de la corrección.
7	El usuario puede exportar el resultado o terminar el proceso.
Post-condiciones	
1	Se corrige el esquema.
Flujos alternativos	
Post-condiciones	
1	Para concluir el proceso de corrección deben ser ejecutadas ambas formas de corrección, tanto manual como automática.
Validaciones	
1	Valida que el reporte a corregir debe tener fecha posterior a la última corrección

del esquema realizada, sino proponer al usuario realizar otra revisión del esquema.



Interfaz de usuario. Corregir Esquema

2.4.2 Requisitos no Funcionales (RNF).

- **RNF1 Portabilidad:** El sistema debe ser multiplataforma, haciendo énfasis en Linux y Windows.
- **RNF2 Seguridad:** Garantizar que las transacciones de datos se ejecuten de forma segura, ya sean en la red o por soporte físico.
- **RNF3 Software:** Los usuarios deben tener instalado como sistema operativo Windows o Linux, para PC o MacOS para Macintosh. La PC cliente debe tener instalado el entorno de ejecución de java, Java Runtime Enviroment (JRE) 1.6 o superior.
- **RNF4 Hardware:** La PC donde se encontrará la aplicación deberá poseer los siguientes requerimientos: Procesador Pentium IV o superior, 333 MHz mínimo pero recomendado 1.8 GHz, 2 GB de disco duro disponible o más. Mínimo de memoria RAM 128 Mb, adaptador de Red y conectividad. La

PC donde se encontrará el servidor de base de datos deberá poseer las siguientes características un CPU IntelCONEXIÓN Core2 Duo a 64 bits con 2.2 GHz, con memoria RAM de 2 GB y capacidad de disco a partir de 40 GB.

2.5 Técnicas de Validación de Requisitos.

En la validación de los requisitos de esta herramienta fueron aplicadas las siguientes técnicas:

Revisión de requisitos:

Se realizaron varias reuniones planificadas y dirigidas por el analista del proyecto donde el cliente verificó que los requisitos definidos son realmente las funcionalidades que solicitó y que los mismos tengan una descripción correcta, precisa, consistente y verificable para lograr un entendimiento de los mismos al pasar a etapas de diseño e implementación. La aplicación de esta técnica permitió reducir el tiempo de pruebas (22).

Prototipos:

A partir de las especificaciones de requisitos fueron conformados prototipos de interfaz de usuario. Con esto se validó que los requerimientos estaban en concordancia con las necesidades plasmadas por el cliente. El empleo de esta técnica ofreció como resultados que el cliente tuviera una idea de la estructura de la interfaz de usuario y se favoreciera la comunicación con el mismo, ya que tenía una visión inicial del sistema y su funcionamiento (23).

2.6 Patrones de Diseño.

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software, brindan una solución ya probada y documentada a situaciones que están sujetas a contextos similares. Otra característica es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (24).

2.6.1 Patrones GRASP aplicados a la solución.

GRASP[13] es un acrónimo que significa General Responsibility Assignment Software Patterns. Los patrones **GRASP** describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones (25).

Para el diseño de la propuesta de solución fueron aplicados los siguientes patrones GRASP:

Experto: tiene como principio fundamental asignarle una responsabilidad a una clase que contenga la información necesaria para cumplir la responsabilidad, o sea la clase debe ser la experta en la información. Ej. La clase **getConection (String driver, String url, String user, String pass)** es experta, esta clase contiene la información suficiente lograr la conexión con la base datos montada sobre el SGBD Oracle 11g.

Creador: este patrón se encarga de asignarle a una clase la responsabilidad de crear una instancia de otra determinada clase. **(Ej. Public class reporte_corregido extends reporte)**, en esta clase se crean objetos de la clase **reporte** para obtener sus atributos, demostrando que es una clase creador.

Controlador: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. (Ej. La clase **principal)**, la misma gestiona las entradas de los usuarios a través de las vistas y controla el flujo de eventos.

Alta Cohesión: Se encarga de asignar una responsabilidad de modo que la cohesión siga siendo alta. Una clase tiene responsabilidades moderadas en un área funcional y colabora con otras para llevar a cabo las tareas. Ejemplificado a través de la clase **variablesGlobales**, la misma posee la información propia de ella y colabora con la clase conectionOracle para darle solución a la funcionalidad **prefijoNombreTablaEsquema (String esquema).**

2.7 Patrón Modelo-Vista-Controlador (MVC).

Divide una aplicación interactiva en 3 componentes (26).

---

[13] GRASP: General Responsibility Assignment Software Patterns. (patrones generales de software para asignar responsabilidades)

Modelo: Está compuesto por las entidades que representan las entidades de la base de datos y sus funcionalidades correspondientes, las clases **conectionOracle, conectionPostgres**, **listaErrores.java**, **listaReportes.java, tipoError.java.**

Vista: Despliegan la información del modelo a los usuarios. Compuesto (Ej. Las interfaces: **editarConexión**, **listadoReporte**, **listadoErrores, detallesReporte**).

Controlador: La clase **principal.java** es la controladora, encargada de gestionar las entradas del usuario y ejecutar las funcionalidades correspondientes.

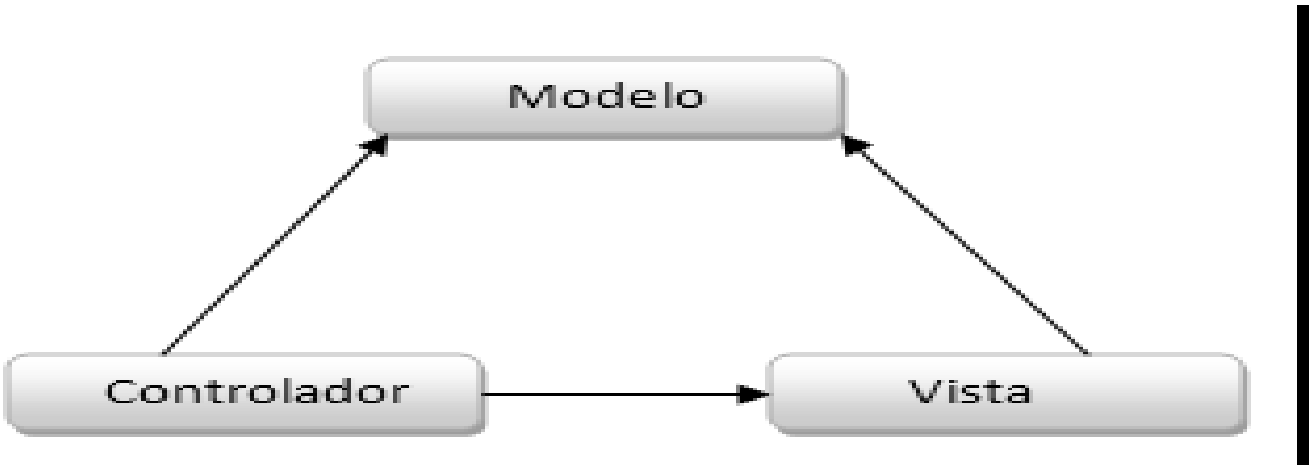


Figura 4. Modelo Vista Controlador (las flechas representan una asociación directa).

2.8 Diagrama de clases del diseño.

El diagrama de clases es utilizado para visualizar las relaciones entre las clases que componen el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación, ya que una clase es una descripción de conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica; mostrando un conjunto de elementos que son estáticos, como las clases y tipos junto con sus contenidos y relaciones. Un diagrama de clases está compuesto por dos elementos fundamentales la Clase: en la que se encuentran los atributos, métodos y la visibilidad de los mismos y las Relaciones: que pueden ser de varios tipos Herencia, Composición, Agregación, Asociación y Uso (27).

Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro (27).

A continuación se muestra el diagrama de clases del sistema que se desea implementar:

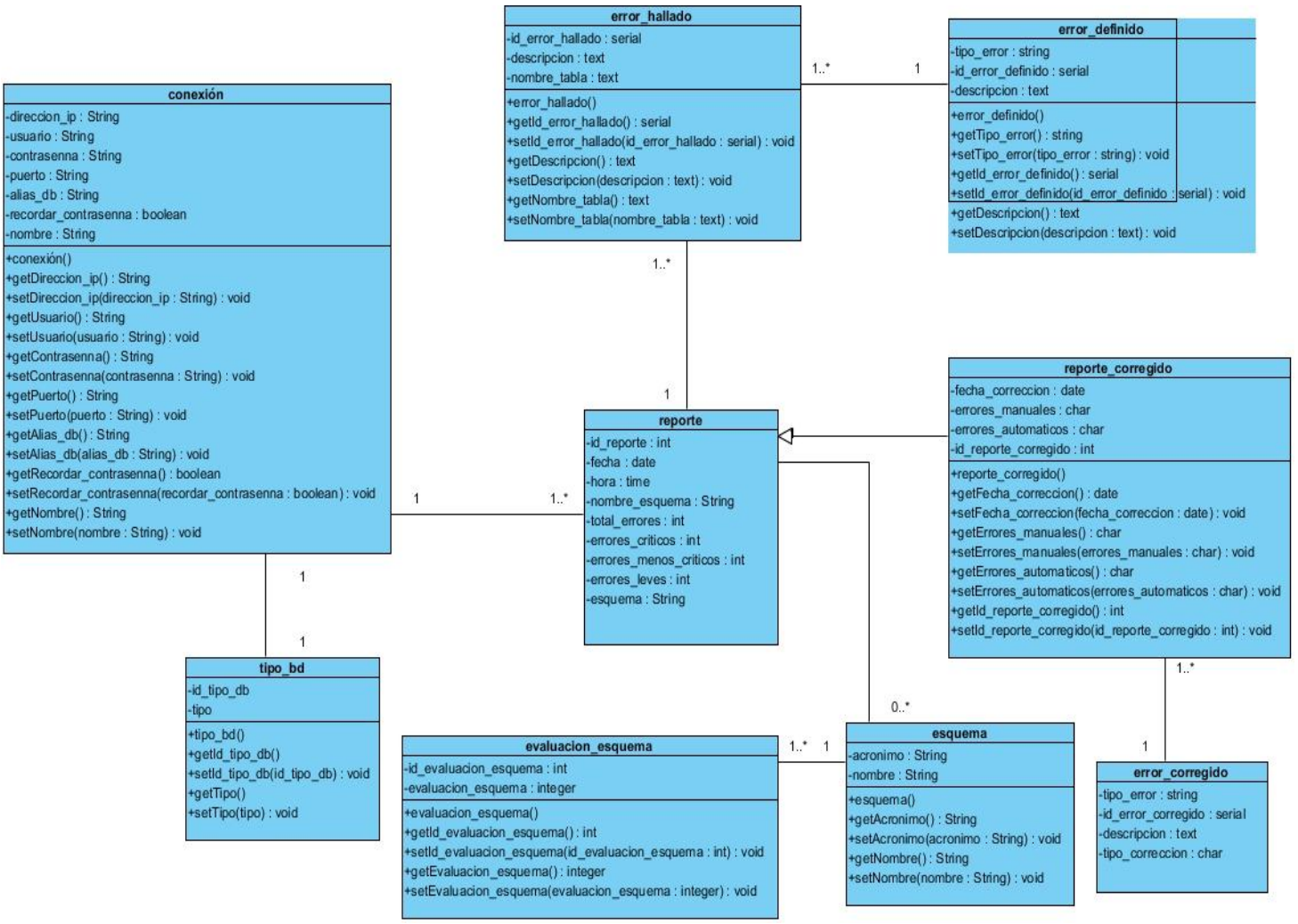


Figura 5**:** Modelo de clases del diseño.

Entre las clases del diagrama anteriormente mostrado se encuentran error_hallado, reporte y conexión.
En la clase error_hallado se encuentran los atributos que se muestran en la **tabla 4** con sus respectivos métodos de acceso. En dicha clase es donde se almacenan todos los errores que se encontraron en el proceso de revisión para luego ser mostrados en el reporte.

Tabla 4: Especificación de la entidad error_hallado del diagrama de clases.

Nombre: error_hallado	
Tipo de clase: Entidad	
Atributo	Tipo de dato
id_error_hallado	serial
descripcion_error	text
nombre_tabla	text
Error_hallado()	
getId_error_hallado() : serial	
setId_error_hallado(id_error_hallado : serial) : void	
getDescripcion() : text	
setDescripcion(descripcion : text) : void	
getNombre_tabla() : text	
setNombre_tabla(nombre_tabla : text) : void	

En la clase reporte se encuentran los atributos que se muestran en la **tabla 5** con sus respectivos métodos de acceso. En dicha tabla es donde se almacenan todos los reportes para luego mostrárselos al usuario.

Tabla 5: Especificación de la entidad reporte del diagrama de clases.

Nombre: reporte	
Tipo de clase: Entidad	
Atributo	**Tipo de dato**
id_reporte	Int
fecha	Date
hora	Time
nombre_esquema	Text
total_errores	Int
errores_criticos	Int

errores_menos_criticos	Int
errores_leves	Int
Reporte()	
getId_reporte() : int	
setId_reporte(id_reporte : int) : void	
getFecha() : Date	
setFecha(fecha : Date) : void	
getHora() : Time	
setHora(hora : Time) : void	
getNombre_esquema() : text	
setNombre_esquema(nombre_esquema : text) : void	
getTotal_errores() : int	
setTotal_errores(total_errores : int) : void	
getErrores_criticos() : int	
setErrores_criticos(errores_criticos : int) : void	
getErrores_menos_criticos() : int	
setErrores_menos_criticos(errores_menos_criticos : int) : void	
getErrores_leves() : int	
setErrores_leves(errores_leves : int) : void	

En la clase conexión se encuentran los atributos que se muestran en la **tabla 6** con sus respectivos métodos de acceso. En dicha tabla es donde se almacenan las conexiones a las diferentes bases de datos que se desean revisar.

Tabla 6: Especificación de la entidad conexión del diagrama de clases.

Nombre: Conexión	
Tipo de clase: Entidad	
Atributo	**Tipo de dato**
id_conexion	serial
conexion_ip	text
usuario	text
contrasenna	Text
puerto	int
alias_bd	text

recordar_contrasenna	bool
nombre	text
Conexión()	
getId_Conexion() : serial	
setId_Conexion(id_conexion : serial) : void	
getDireccion_ip() : text	
setDireccion_ip(direccion_ip : text) : void	
getUsuario() : text	
setUsuario(usuario : text) : void	
getContrasenna() : text	
setContrasenna(contrasenna : text) : void	
getPuerto() : int	
setPuerto(puerto : int) : void	
setTipo_bd(tipo_bd : text) : void	
getAlias_bd() : text	
setAlias_bd(alias_bd : text) : void	
getRecordar_contrasenna() : bool	
setRecordar_contrasenna(recordar_contrasenna : bool) : void	
getNombre() : text	
setNombre(nombre : text) : void	

2.9 Modelo de datos.

Un modelo de datos es la definición lógica, independiente y abstracta de los objetos, operadores y demás que en conjunto contribuyen a la máquina abstracta con la que interactúan los usuarios (28).

A continuación se muestra el modelo de datos del sistema:

Figura 6. Modelo de datos del Sistema.

2.9.1 Descripción de las entidades del modelo entidad relación.

- **conexión:** Esta entidad es la encargada de gestionar la conexión a la base de datos que va a ser evaluada.
- **reporte:** En esta entidad se almacenan los reportes generados para luego ser mostrados o eliminado.
- **error_hallado:** Se almacenan todos los errores que se encuentren durante el proceso de revisión de los diferentes esquemas.
- **error_definido:** Se encuentran almacenados todos los errores definidos por el Departamento Aduana CC-HH.
- **evalucion_esquema:** Es donde se almacenan todas las evaluaciones de los esquemas para luego procesarlas y dar un estado general de toda la base de datos o del propio esquema.
- **error_corregido:** En esta entidad se almacenan todos los errores que son corregidos.
- **tipo_db:** En esta entidad se almacenan los tipos de bases de datos, entre las que se encuentran Oracle, MySQLy PostgreSQL.
- **esquema:** En esta entidad se almacena el nombre de todos los esquemas que componen una base de datos.
- **reporte_corregido:** En esta entidad se almacenan todos los reportes que son corregidos.

2.10 Diagrama de secuencias.

Los diagramas de secuencia muestran gráficamente los eventos que fluyen desde los actores hacia el sistema, en la fig.6 se muestra el diagrama del requisito “Corregir esquema” donde el usuario solicita la opción corregir esquema en la clase “**principal.java**” y el sistema carga los reportes existentes del esquema seleccionado y se los muestra al usuario, el cual selecciona el reporte, la clase “**correccionCompletaOracle.java**” realiza la evaluación y luego se le muestra al usuario un mensaje con el resultado obtenido y la opción de continuar para ver los detalles de la corrección, luego el usuario termina el proceso (29).

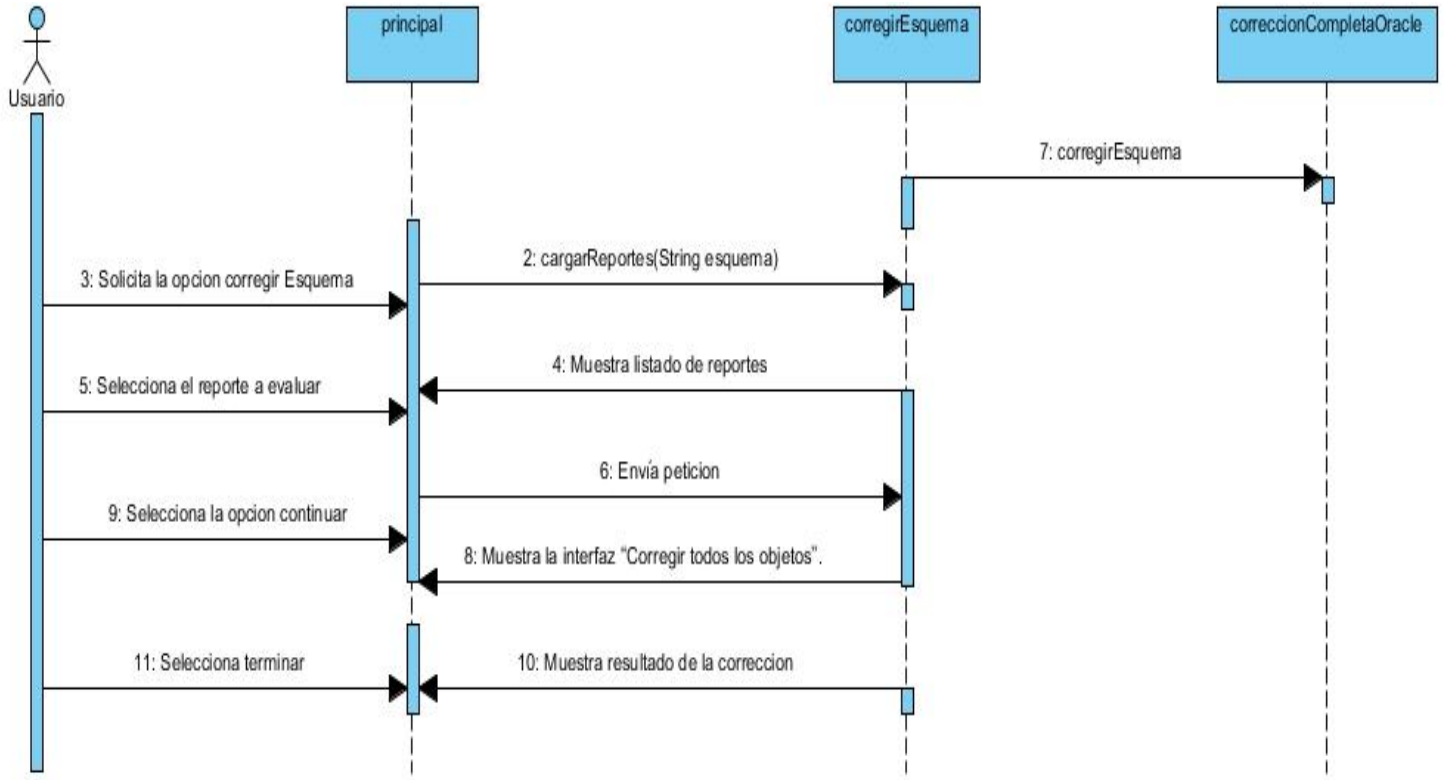


Figura 7. Diagrama de secuencias del requisito "Corrección Automática"

2.11 Métrica Tamaño Operacional de Clases (TOC).

Para realizar la validación a las clases del diseño se seleccionó como métrica Tamaño operacional de clases (TOC).

La aplicación de la métrica TOC define los siguientes atributos de calidad:

- Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

- Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
  El TOC está dado por el número de métodos asignados a una clase (30).

Tabla 7.Tamaño operacional de clase (TOC).

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

En la siguiente tabla se muestran los valores de los umbrales para los atributos de calidad necesarios para evaluar las métricas.

Tabla 8. Rango de valores para los criterios de evaluación de la métrica Tamaño Operacional de Clase (TOC).

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio

	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

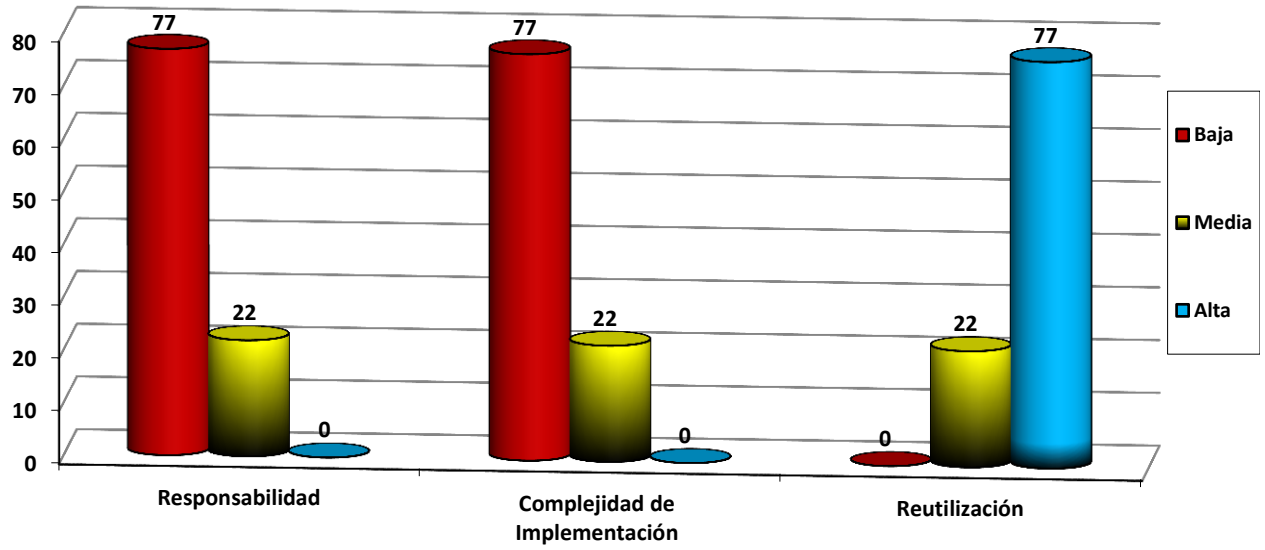


Figura 8. Resultados de la evaluación del diseño con TOC.

Haciendo un análisis de los resultados obtenidos en la evaluación de la métrica TOC, se demuestra que el diseño presenta una calidad aceptable pues el 77% de las clases poseen evaluaciones positivas en los atributos de calidad Responsabilidad, Complejidad de Implementación y Reutilización facilitando el proceso de implementación.

2.12 Métrica Relaciones entre Clases (RC).

Las RC están dadas por el número de relaciones de uso de una clase con otras (31).

Tabla 9: Tabla Relaciones entre clases (RC).

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 10: Rango de valores para los criterios de evaluación de la métrica Relaciones entre clases (RC).

Atributos de calidad	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	>Promedio y <= 2*Promedio
	Alta	>2*Promedio
Reutilización	Alta	<=Promedio
	Media	>Promedio y <= 2*Promedio
	Baja	>2*Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	>Promedio y <= 2*Promedio
	Alta	>2*Promedio

A continuación se muestran los resultados de la evaluación de la métrica RC:

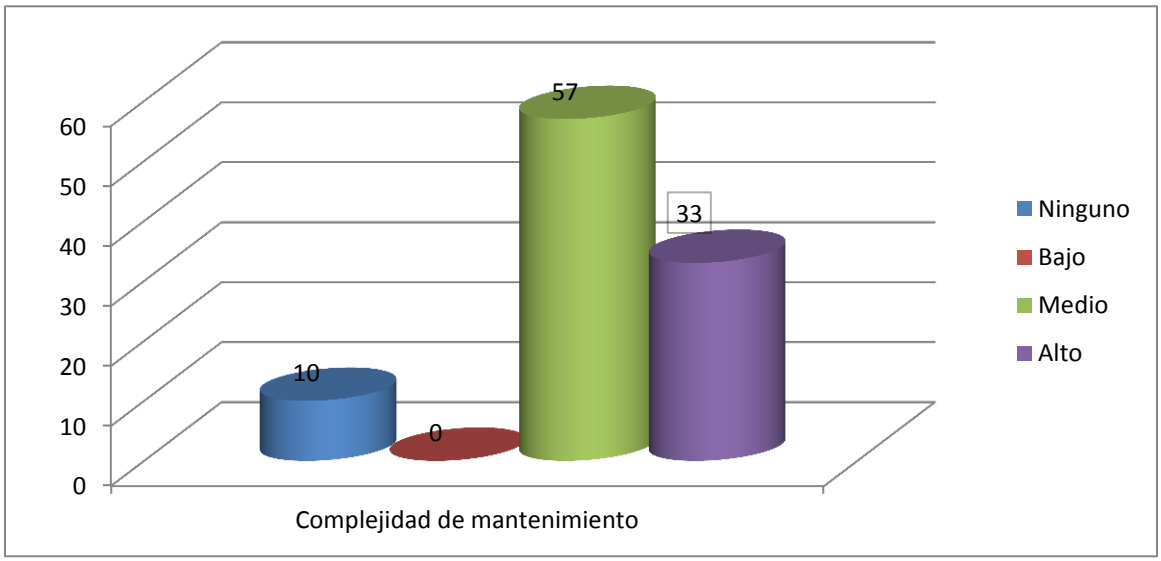


Figura 9. Gráfica que representa el % de clases por categorías del atributo Complejidad de Implementación obtenidos en la aplicación de la métrica RC.

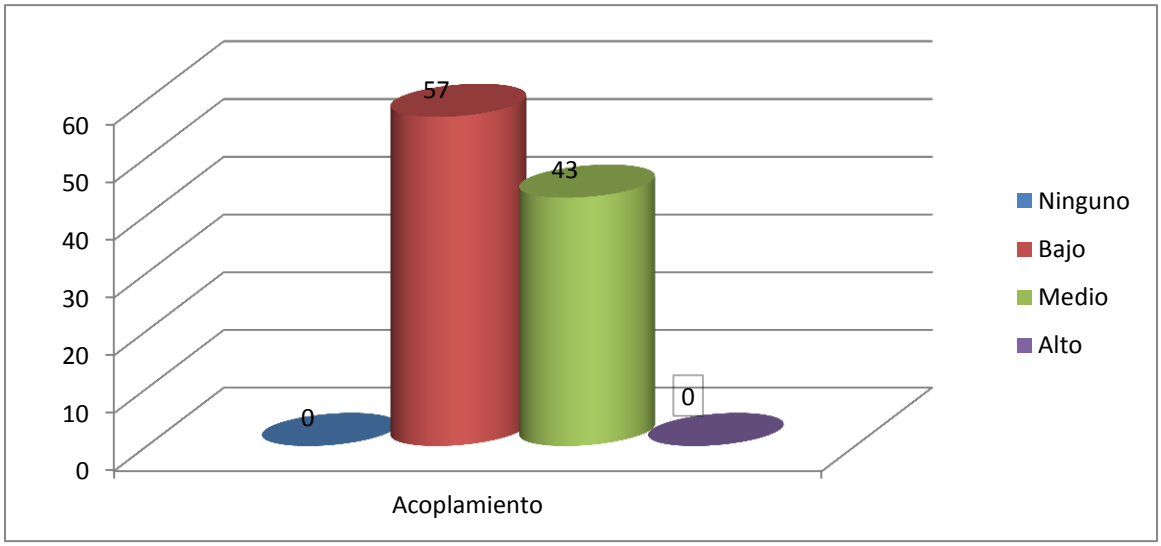


Figura 10. Gráfica que representa el % de clases por categorías del atributo Acoplamiento obtenidos en la aplicación de la métrica RC.

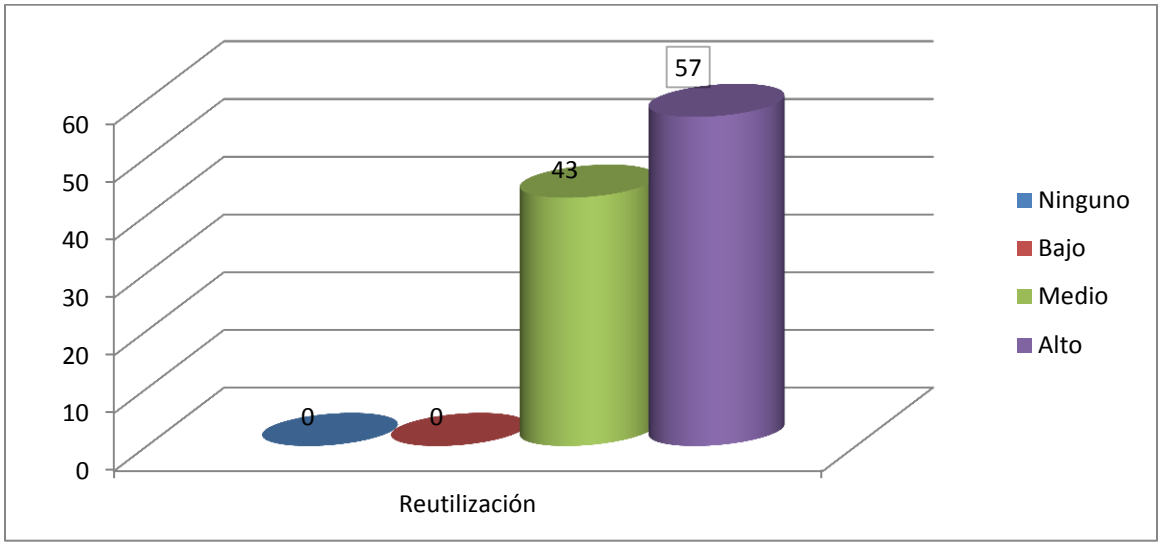


Figura11 Gráfica que representa el % de clases por categorías del atributo Reutilización obtenidos en la aplicación de la métrica RC.

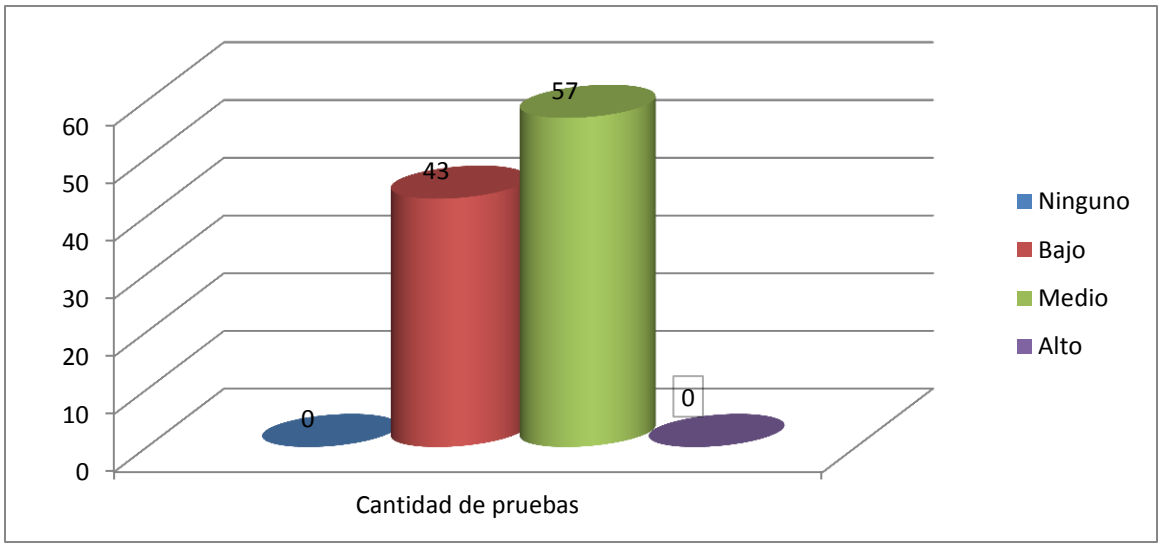


Figura 12 Gráfica que representa el % de clases por categorías del atributo Cantidad de Pruebas obtenidos en la aplicación de la métrica RC.

Haciendo un análisis de los resultados obtenidos en la evaluación de la métrica RC, se puede concluir que el diseño del sistema tiene una calidad aceptable teniendo en cuenta que el 57% de las clases tienen 3 o menos dependencias de otras clases, además el 57% de las clases posee índices aceptables en cuanto a Acoplamiento. Así mismo los atributos de Complejidad de Mantenimiento, Cantidad de pruebas y Reutilización se comportan satisfactoriamente.

Los resultados obtenidos en la aplicación de las métricas TOC y RC demuestran la realización de un buen diseño.

2.13 Guía para la evaluación de un esquema de base de datos.

Para la evaluación de un esquema se tuvo en cuenta el peso de los errores críticos. Dicho peso fue determinado teniendo en cuenta el impacto que posee cada uno, sobre el funcionamiento de la base de datos. Donde se le asigna un número entre 1 y 50 al error, el cual representa el porciento de funcionamiento en que se ve afectado el esquema si el error existe.

A continuación se muestran los posibles errores con sus pesos correspondientes.

Errores Críticos en las tablas**:**

- El nombre de las tablas excede los 26 caracteres. (35%)
- Los nombres de las tablas poseen caracteres extraños. (5%)
- Las tablas deben poseer al menos una clave primaria. (15%)
- Los atributos que son claves primarias carecen del identificador “id_” y poseen el prefijo del esquema. (5%)
- Los nombres de las tablas que son especializaciones poseen el mismo nombre de la tabla origen y la clave primaria de la misma puede ser la del padre, una propia de ella o una combinación de las dos tablas. (10%)
- Las tablas que poseen las claves foráneas y primarias a la vez poseen mal la combinación del nombre. (3%)
- Todas las tablas que en un esquema tienen el permiso (grant) de lectura, actualización o eliminación para otro usuario, debe tener sinónimo público. (20%)
- El nombre de los sinónimos de las TC no poseen el mismo nombre de las tablas a las cuales pertenecen. (25%)
- Las tablas TC no poseen todos los atributos obligatorios que por la naturaleza del negocio son necesarios. (50%)
- Las tablas TC(R) no poseen un id auto-incrementable como clave primaria. (10%)
- Los campos poseen caracteres extraños. (5%)
- Los campos created_at y deleted_at se encuentran mal escritos. (2%)
- Las tablas que por la naturaleza del negocio requieren secuencia no la poseen. (30%)
- Las claves foráneas tienen que pertenecer a una clave única para garantizar la unicidad de los registros. (15%).

Errores Críticos en las secuencias:

- Las secuencias no poseen el sufijo "_SEQ". (50%)
- Existen secuencias que no están asociadas a ninguna tabla. (2%)

Errores Críticos en los sinónimos:

- Los sinónimos deben ser públicos. (20 %)

El peso total posible según los errores críticos de los diferentes objetos del esquema se calcula de la siguiente manera:

Ppt = tottab * Spet

Ppsec = totsec * Spesec

Ppsin = totsin * Spesin

- Donde **Ppt** es el peso posible de los errores en las tablas, **tottab** es el total de tablas del esquema y **Spet** suma de los pesos de los errores críticos en tablas.
- **Ppsec** es el peso posible de los errores en las secuencias, **totsec** es el total de secuencias del esquema y **Spesec** suma de los pesos de los errores críticos en secuencias.
- **Ppsin** es el peso posible de los errores en los sinónimos y **totsin** es el total de los sinónimos del esquema y **Spesin** suma de los pesos de los errores críticos en sinónimos.

El peso total posible del esquema es la suma de los pesos posibles de cada objeto.

Ppe = Ppt + Ppsec + Ppsin

- Siendo **Ppe** el peso posible del esquema.

Para hallar el estado en que se encuentra el esquema se calcula de la forma siguiente:

$\boldsymbol{Porcpeh} = [\sum_{k=1}^{n}(tote_k * P_K) + \sum_{k=1}^{n}(tote_k * P_K) + \sum_{k=1}^{n}(tote_k * P_K)] * 100/$ Ppe

Donde **Porcpeh** es el porciento del peso de errores hallados, **tote** es el total de errores de tipo k, siendo k la cantidad de tipos de errores hallados.

Estableciendo como rango para determinar el estado de un esquema.

- Porcpeh >= 50: **No funcional**.
- 1 <= Porcpeh <= 49: **Parcialmente funcional**.
- 0 Porcpeh: **Funcional.**

2.13.1 Ejemplo.

Sobre un esquema que posee 50 objetos distribuidos de la siguiente manera:

Ect = 14, Spet = 230, Ecsec = 2, Spesec = 52, Ecsin = 1, Spesin = 20

	Tablas/errores críticos hallados	**Secuencias/errores críticos hallados**	**Sinónimos/errores críticos hallados**
Total	**20/0**	**20/40**	**10/10**

Ppt = tottab * Spet = 20 * 230 = 4600

Ppsec = totsec * Spesec = 20 * 52 = 1040

Ppsin = totsin * Spesin = 10 * 20 = 200

Ppe = Ppt + Ppsec + Ppsin = 5840

$\text{Porcpeh} = [\sum_{k=1}^{n}(tote_k * P_K) + \sum_{k=1}^{n}(tote_k * P_K) + \sum_{k=1}^{n}(tote_k * P_K)] * 100 /$ Ppe

Porcpeh = [(0 * 0) + (10 * 20) + (20 * 50 + 20*2)] * 100 / 5840 = [(200 + 1040)*100]/5840 = 21,23 %

Siendo el estado del esquema a evaluar **Parcialmente funcional**.

2.13 Conclusiones.

En este capítulo se describió de manera detallada la propuesta de solución del sistema a implementar. Se describieron los principales artefactos según el modelo de desarrollo logrando obtener un modelo de diseño legible para la implementación del sistema.

Capítulo 3: Implementación y pruebas.

3. Introducción.

En este capítulo se realizará una descripción de los principales artefactos que se generan durante la implementación de la herramienta. Se mostrarán además el diagrama de componentes y de despliegue para una mayor comprensión del sistema. Además se realizará una breve descripción de las pruebas realizadas.

3.1 Diagrama de Componentes del sistema.

Un diagrama de componentes permite visualizar la estructura del sistema y el comportamiento de los servicios que estos componentes proporcionan y utilizan a través de las interfaces.(32).
A continuación se muestra el Diagrama de componentes del sistema:

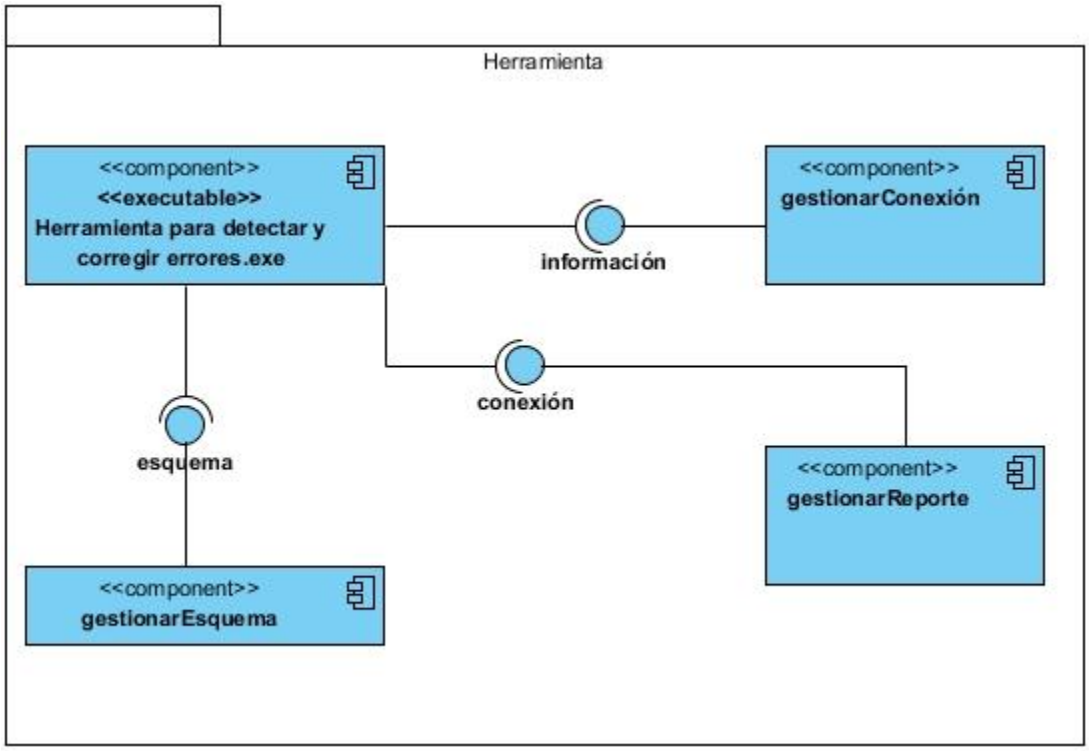


Figura 13.Diagrama de componentes del sistema.

3.2 Diagrama de despliegue del sistema.

El diagrama de despliegue muestra la configuración física sobre la que será desplegado el software. Este presenta los nodos computacionales que intervienen en el funcionamiento del sistema, las conexiones y los protocolos de comunicación que serán utilizados (32).

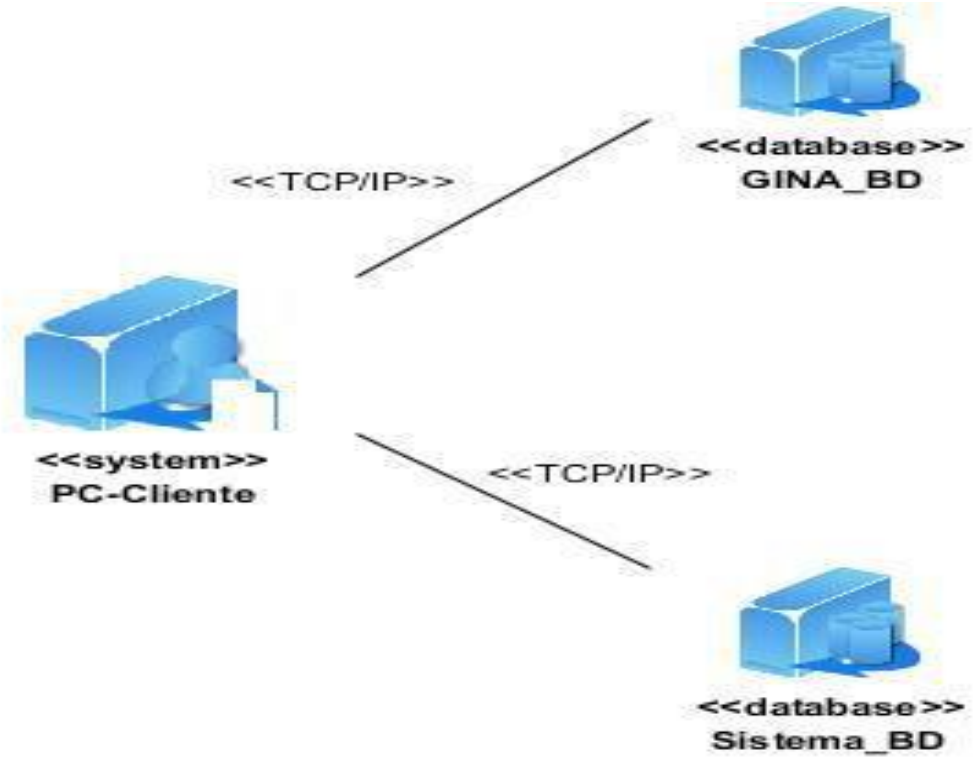


Figura 14.Diagrama de despliegue del sistema.

3.2.1 Descripción de los nodos.

PC-Cliente: Es la herramienta como tal, el sistema que va a realizar las peticiones hechas por el usuario desde la PC-Cliente.

GINA: Es la base de datos a la que se va a conectar la herramienta mediante el protocolo TCP/IP[14] para realizar las operaciones necesarias. En este caso se llama GINA que no es más que la base de datos del sistema de Gestión Integral de Aduanas. Esta base de datos tiene como SGBD a Oracle 11g.

Sistema_db: Es la base de datos de la aplicación. Montada sobre PostgreSQL 8.0.

3.3 Comparación del tiempo de detección y corrección de forma manual y automática.

Se demostró que la herramienta desarrollada permite disminuir el tiempo de detección y corrección de errores cometidos en el diseño de un esquema de base de datos según el Estándar de Codificación de Base de Datos de la Línea Aduana, se tomó como muestra el esquema solicitudes al que se le realizaron las pruebas arrojando los siguientes resultados:

Tabla 11. Tiempo de identificación y corrección de errores.

TE	Cantidad	TDM	TDA	TCM	TCA
Críticos	92	2 horas	1 seg.	4 horas.	2 seg.
Menos críticos	205	3 horas	2 seg.	6 horas.	4 seg.
Leves	223	3 horas	3 seg.	6 horas.	4 seg.
Total	520	8 horas	6 seg.	16 horas.	8 seg.

TE: Tipo de errores.
TDM: Tiempo de detección manual.
TDA: Tiempo de detección automática.
TCM: Tiempo de corrección manual.
TCA: Tiempo de corrección automática.

[14] TCP/IP: Protocolo mediante el cual se realiza la comunicación entre la aplicación y los servidores de Bases de Datos.

3.4 Pruebas.

Las pruebas realizadas a un software determinan el estado de la calidad del producto. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. El proceso de pruebas, sus objetivos y los métodos y técnicas usados se describen en el plan de prueba (33).

La prueba de software es un elemento crítico para la garantía del correcto funcionamiento del software, por lo que entre sus objetivos se encuentran los siguientes:

1. Detectar defectos en el software.
2. Verificar la integración adecuada de los componentes.
3. Verificar que todos los requisitos se han implementado correctamente.
4. Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
5. Diseñar casos de prueba que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo (33).

3.4.1 Pruebas de aceptación.

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. Para lograr esta determinación es utilizada la llamada 'prueba de aceptación'. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique.

Para eliminar la influencia de conflictos de intereses y para que sea lo más objetiva posible, la prueba de aceptación nunca debería ser responsabilidad de los ingenieros de software que han desarrollado el producto. Para la preparación, la ejecución y la evaluación de la prueba de aceptación ni siquiera hacen falta conocimientos informáticos.

El cliente es el mayor responsable de verificar cada una de las pruebas y de priorizar la corrección de las pruebas que fallan (34).

3.4.1.1 Prueba asociada al requisito crear conexión a la base datos.

La siguiente prueba evaluará el proceso de creación de una conexión en el sistema, cuando el usuario omite el llenado de algunos campos obligatorios, provee una dirección de IP que no cumple con el formato adecuado o no introduce un puerto correcto. Se hicieron 2 iteraciones, encontrando tantas no conformidades en la primera y 0 en la última, teniendo éxito en las pruebas.

Tabla 12: Prueba: Crear conexión con campos incorrectos.

Número de caso de prueba : 1	**Requisito Crear Conexión**
Nombre de la prueba: Crear conexión con campos incorrectos.	
Descripción de la prueba: El usuario accede al formulario de creación de una nueva conexión en el cual llena los datos requeridos para dicho proceso, estos datos son incorrectos o existen campos obligatorios vacíos.	
Condiciones de ejecución: El servidor de base de datos de la aplicación en ejecución se encuentre en funcionamiento.	
Entradas: - Nombre. (Alias) - Servidor. - Nombre de la base de datos. - Usuario. - Contraseña. - Puerto. - Almacenar contraseña. (Marcar)	
Resultado esperado: El sistema alerta al usuario de los errores cometidos en el momento de completar el formulario de creación de una nueva conexión, se muestra un mensaje de error.	
Evaluación: Satisfactoria.	

3.4.1.2 Prueba asociada al requisito modificar conexión a la base datos.

En esta prueba se evaluará la modificación de una conexión ya existente donde se evaluará si la aplicación es capaz de cargar los datos de la conexión seleccionada y si se pueden modificar los atributos de la misma.

Tabla 13: Prueba: Modificar conexión con campos incorrectos.

Número de caso de prueba : 2	**Requisito Modificar Conexión**
Nombre de la prueba: Modificar conexión a la base de datos con datos correctos.	
Descripción de la prueba: El usuario selecciona la conexión a ser modificada, luego accede al formulario de modificar conexión en el cual reemplaza los datos existentes por los datos nuevos, al acceder al formulario se cargan los valores de la conexión seleccionada. Los datos intercambiados son correctos.	
Condiciones de ejecución: El servidor de base de datos de la aplicación en ejecución, se encuentre en funcionamiento y que al menos exista almacenada una conexión. El usuario debe seleccionar la conexión a modificar.	
Entradas: - Nombre. (Alias) - Servidor. - Nombre de la base de datos. - Usuario. - Contraseña. - Puerto. - Almacenar contraseña. (Marcar)	
Resultado esperado: El sistema muestra un cartel indicándole al usuario que la modificación fue correcta.	
Evaluación: Satisfactoria.	

3.4.1.3 Prueba asociada al requisito eliminar conexión a la base datos.

La siguiente prueba evaluará el proceso de eliminación de una conexión a la base datos.

Tabla 14: Prueba: Eliminar conexión con campos incorrectos.

Número de caso de prueba : 3	**Requisito Eliminar Conexión**
Nombre de la prueba: Eliminar conexión de la base de datos.	
Descripción de la prueba: El usuario selecciona la conexión que desea eliminar, accede en el	

menú a la funcionalidad eliminar conexión y el sistema debe mostrar un cartel en el que se muestre si fue satisfactoria o no la operación realizada.
Condiciones de ejecución: El servidor de base de datos de la aplicación en ejecución, se encuentre en funcionamiento y que al menos exista almacenada una conexión de base de datos. El cliente debe seleccionar la conexión que desea eliminar.
Entradas: /
Resultado esperado: El sistema muestra una alerta indicándole al usuario que la conexión ha sido eliminada de forma exitosa.
Evaluación: Satisfactoria.

3.4.1.4 Prueba asociada al requisito revisar errores en las restricciones.

La siguiente prueba evaluará el proceso de revisión de las restricciones de un esquema determinado por el usuario.

Tabla 15: Prueba: Revisar restricciones.

Número de caso de prueba : 4	**Requisito Revisar Restricciones**
Nombre de la prueba: Revisar restricciones de un esquema sin seleccionar el mismo.	
Descripción de la prueba: Luego de establecer la conexión a la base de datos que desea revisar el usuario, accede a través del menú a la funcionalidad revisar restricción, pero no especifica el esquema al cual le quiere revisar las restricciones.	
Condiciones de ejecución: Tiene que estar al menos una conexión activa. El usuario debe seleccionar el esquema que desea revisar.	
Entradas: /	
Resultado esperado: El sistema muestra una alerta indicándole al usuario que debe seleccionar un esquema para ser revisado.	
Evaluación: Satisfactoria.	

3.4.1.5 Prueba asociada al requisito corregir esquema.

La siguiente prueba evaluará el proceso de corrección de un esquema determinado por el usuario.

Tabla 16: Prueba: Corregir esquema.

Número de caso de prueba : 5	**Requisito Corregir esquema**
Nombre de la prueba: Corregir esquema sin seleccionar el mismo.	
Descripción de la prueba: Luego de establecer la conexión a la base de datos que desea revisar el usuario, accede a través del menú a la funcionalidad corregir esquema, pero no especifica el esquema al cual le quiere realizar la corrección.	
Condiciones de ejecución: Tiene que estar al menos una conexión activa.	
Entradas: /	
Resultado esperado: El sistema muestra una alerta indicándole al usuario que debe seleccionar un esquema para ser revisado.	
Evaluación: Satisfactoria.	

3.4.2 Pruebas unitarias.

La prueba de unidad se realiza sobre cada módulo del software de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional de un programa independiente, está correctamente codificado. En estas pruebas cada módulo será probado por separado y lo hará, generalmente, la persona que lo creo. En general, un módulo se entiende como un componente de software que cumple las siguientes características:

- Debe ser un bloque básico de construcción de programas.
- Debe implementar una función independiente simple.
- Podrá ser probado al cien por cien por separado (35).

Junit es un “framework” de pruebas gratuito en el que se define un conjunto amplio de clases que automatizan la ejecución de pruebas unitarias para software orientado a objetos, en particular de programas Java (36).

A continuación se muestra un caso de pruebas unitarias a la clase conectionOracle realizadas con el framework Junit.

Caso de prueba 1: Nombre clase: ConectionOracle.

Tipo de prueba: Prueba de unidad.

Tabla: 17: Pruebas con Junit a los métodos de la clase conectionOracle

Métodos a probar	Descripción
establecerConexion	Establece la conexión a la base de datos.
contarTablasEsquema	Cuenta el total de tablas que posee un esquema determinado por el usuario.
contarIndicesEsquema	Cuenta el total de índices que posee un esquema determinado por el usuario.

Casos de prueba de unidad

Descripción de la prueba: Se realizaron pruebas de unidad a las principales funcionalidades de la clase conectionOracle.

Funcionalidad	Método utilizado	Recibe	Resultado esperado del método	Resultado esperado de la prueba	Resultado real de la prueba
establecerConexion	expected	Datos incorrectos de una conexion.	Lanza SQLException.	OK	OK
establecerConexion	timeout	Datos correctos de una	Establece conexión en menos de 50seg	OK	OK

		conexion.			
contarTablasEsquema	assertNotNull	Objeto null	Devuelve una lista vacía.	OK	OK
contarIndiceEsquema	assertEquals	Esquema „TC"	78	OK	OK

Ejecución de la prueba con Junit:

3.5 Conclusiones.

Se describieron los diagramas de despliegue y componentes que guiarán el proceso de implementación y de despliegue de la aplicación Se realizaron pruebas de aceptación a los principales requisitos funcionales del sistema. Luego de este proceso ya el sistema está listo para ser desplegado.

Conclusiones generales.

Luego de analizar las principales herramientas CASE que de alguna manera comprueban estándares de diseño de base de datos, se evidenció la necesidad de implementar un sistema que cumpla con las características específicas del estándar de codificación de base datos de la Línea Aduana en cuanto al diseño de esquemas con la mejor calidad posible. Para ello se realizó un proceso de investigación que propició la obtención del conocimiento necesario para cumplimentar el análisis, diseño y desarrollo de la herramienta, arribando así a las siguientes conclusiones:

- El análisis del "Estándar de Codificación de Base de Datos" establecido en la Línea Aduana, develó los principales errores que se pueden cometer en el proceso de diseño de un esquema de base de datos, posibilitando su clasificación según su impacto en el funcionamiento de la base de datos, determinando así el estado en que se encuentra un esquema.
- Se obtuvo una herramienta que permite disminuir el tiempo de detección y corrección de los posibles errores cometidos en el diseño de un esquema de base datos en Oracle 11g.
- El sistema obtenido cumple con los requisitos planteados.

Una vez implementado el sistema se puede decir que se han cumplido los objetivos propuestos en el presente trabajo de manera satisfactoria.

Recomendaciones:

Luego de haber cumplido con los objetivos planteados en la investigación se considera que para detectar y corregir los errores de diseño de una base de datos en el menor tiempo posible, se necesita incorporarle a la herramienta nuevas reglas de diseño de una base de datos y este trabajo es solo una parte del mismo. Teniendo en cuenta lo antes planteado se pueden hacer las siguientes recomendaciones:

- Agregar a la herramienta la verificación de las 3 primeras Formas Normales.
- Incorporarle a la herramienta un análisis más profundo de reglas de diseño de un esquema de base datos que permita una mejor detección de los errores u una corrección más óptima.

Bibliografía

1. **Rebecca.** *Diseño de base de datos relacionales.* España : Evolution, Artes Grificas, 2000. 84-481-2770-6.

2. Normalizacion de base de datos. [En línea] 28 de Agosto de 2012. www.slideshare.net/sesa78/normalizacion-de-base-de-datos-14102278.

3. **Naranjo, Adrian, Cobo, Jose A. y Nápoles, Fernando.** *Estándar de Codificación de Base de Datos.* Cuba : s.n., 2011.

4. Miguel Matas Blog. [En línea] 06 de Mayo de 2008. http://www.miguelmatas.es/blog/blog/2008/05/06/como-definir-un-estandar-de-codificacion-yo-trabajo/.

5. Herramientas Case. [En línea] 4 de Abril de 2012. [Citado el: 5 de Febrero de 2013.] http://herramietascase.blogspot.com/2012/04/caracteristicas-de-herramientas-case.html.

6. ER-Studio - R2 Data Technology SAC. [En línea] Enero de 2010. www.r2datatechnology.com/datatech/ERStudio.aspx.

7. FabForce.net. [En línea] http://www.fabforce.net/dbdesigner4/.

8. Web asiganturas de la Universidad de La Habna. [En línea] 2012. http://www.fec.uh.cu/websasignaturas/GI/Clases/Sistemas%20de%20Informacion/easy%20case.pdf..

9. Designer oracle . [En línea] Abril de 2010. www.slideshare.net/legalindo/designer-oracle.

10. Introducción a Herramientas CASE y System Architect. [En línea] 2010. www.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf.

11. Java. [En línea] [Citado el: 8 de Marzo de 2013.] www.java.com/es.

12. Entornos De Desarrollo Integrados. [En línea] 2012. www.slideshare.net/GhaBiithahh/entornos-de-desarrollo-integrados.

13. NetBeans IDE 7.1.2 Release Information. [En línea] 2012. netbeans.org/community.

14. PostgreSQL. [En línea] 1996-2005. www.postgresql.org/docs/8.0/.

15. pgAdmin. [En línea] www.pgadmin.org.

16. DBDesigner. [En línea] Abril de 15 de 2004. dbdesigner.softonic.com/.

17. **ENTIDADES, CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE.** *MODELO DE DESARROLLO DE SOFTWARE.* 2012.

18. **Ramos, Flor de María Olivares.** Diseño conceptual y lógico de datos. [En línea] 2013. http://es.scribd.com/doc/33287143/Disen%CC%83o-conceptual-y-logico-de-base-de-datos.

19. Tormenta de Ideas. [En línea] [Citado el: 11 de Marzo de 2013.] members.tripod.com/~hdo_zorrilla/creatividad/tormenta_de_ideas.htm.

20. **Catalunya, Universidad Politécnica de.** *Guía para la elicitación de requisitos.* Catalunya : s.n., 2008.

21. Validación de requisitos. [En línea] 2011. www.slideshare.net/jcgmoreno/tema-1-ingeniera-de-requisitos.

22. Revisiones de requitos. [En línea] 2009. www.revisiones-tecnicas.cl/tags/requisitos.

23. DESARROLLO DE PROTOTIPOS. [En línea] 2005. www.slideshare.net/myjuankiz1/desarrollo-de-prototipos-5662958.

24. **Laman, Craig.** *UML y Patrones. .* Ciudad Mexico : s.n., 1999. 970-17-0261-1.

25. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. 970-17-0261-1.

26. Patrón MVC en Java con Netbeans. [En línea] 26 de Diciembre de 2011. jc-mouse.blogspot.com/2011/12/patron-mvc-en-java-con-netbeans.html.

27. DIAGRAMAS DE CLASES. [En línea] virtual.usalesiana.edu.bo/web/practica/archiv/clases_2.doc.

28. *Diseño de base de datos relacionales.* España : Compuesto en Evolution, Artes Grificas, 2000. 84-481-2770-6.

29. *Diagrama de Secuencia.* [En línea] www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html.

30. Métricas para Sistemas Orientados a Objetos. *Métricas para Sistemas Orientados a Objetos.* Mexico : s.n., 2007.

31. Metricas de diseño. [En línea] Mexico. http://msdn.microsoft.com/es-es/library/dd409390.aspx

32. Diagramas UML: Componentes y despliegue. [En línea] 2001. http://msdn.microsoft.com/es-es/library/dd409390.aspx

33. Pruebas. [En línea] 2005. pruebasdesoftware.com/laspruebasdesoftware.htm.

34. Pruebas de Software. [En línea] pruebasdesoftware.com/pruebadeaceptacion.htm.

35. Calidad de Software. [En línea] 2005. www.calidadysoftware.com/testing/pruebas_unitarias1.php.

36. Junit. [En línea] junit.org/.