Universidad de las Ciencias Informáticas

Facultad 3

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

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

Autor: Laura Elena Torres Azniella

Tutores:

Ing. Fernando Nápoles Gámez

Ing. Adrian Naranjo Garcia

La Habana, junio del 2012

Declaración de autoría.

Declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas, para que lo use según estimen pertinente.

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

_____

Laura Elena Torres Azniella

Firma del Autor

_____ _____

Ing. Fernando Nápoles Gámez Ing. Adrian Naranjo Garcia

Firma del Tutor Firma del Tutor

Dedicatoria.

A mi abuela Alicia y a mi papá, a los cuales les debo todo lo que soy.

A mi mamá, que aunque no este entre nosotros yo se que esta muy orgullosa de la persona que me he convertido.

Laura

Agradecimientos.

Agradezco:

A mi abuela Aya (Alicia) y a mi papá por convertirme en la persona que soy, por confiar en mí siempre y dejarme tomar mis propias decisiones…

A mi hermano Karle por apoyarme siempre…

A mi novio Jorge Luis por soportarme, apoyarme y en general por compartir conmigo cuatro años maravillosos…

A toda la familia de Jorge por el apoyo que me han dado en todo este tiempo…

A mis tutores por la paciencia y dedicación que me han tenido…

A Gretel por su amistad incondicional…

A Lisbet y a su familia por su apoyo y preocupación durante toda mi carrera…

A mi madrina Mari y a su familia por su preocupación hacia mi…

A todos los muchachos del grupo 3502 por los buenos momentos que he pasado con ellos…

A todos los que de una manera u otra han hecho posible este trabajo…

Laura

Resumen.

El Departamento de Soluciones para la Aduana ha venido presentando algunas dificultades con la estandarización de la base de datos, y en Oracle, que es el gestor con que se trabaja en dicho departamento, se presentan inconvenientes para corregir de forma manual los errores de diseño presentes en los diferentes objetos de un esquema. En el departamento 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 el departamento, 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 y medio en identificarlos. En este trabajo se diseña e implementa una herramienta que automatiza el proceso de identificación de los errores de diseño en la base de datos del departamento, 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. Se evalúan los esquemas según el estado en que estos se encuentran, brindando la posibilidad de exportar estos reportes a formato Excel.

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

Índice.

Introducción.

La Aduana General de la República (AGR) es una entidad destinada a la supervisión de los medios de transporte, mercancías y viajeros que entran y salen del país. Esto trae consigo la manipulación de grandes volúmenes de información, la cual debe estar disponible entre las diferentes entidades aeroportuarias del país. A raíz de esto surge el Sistema Único de Aduana (SUA) encargado de gestionar todos los sistemas ya instaurados, garantizando así una comunicación segura y ágil entre ellos.

A petición de la AGR, representada por el Centro de Automatización para la Información y la Dirección (CADI) y con la colaboración de la Universidad de las Ciencias Informáticas (UCI), surge el sistema de Gestión Integral Aduanera (GINA), proyecto desarrollado por el Departamento de Soluciones para la Aduana de la facultad 3 de la UCI.

Como parte indispensable de cualquier sistema, la persistencia de sus datos juega un papel fundamental. Para esto se utilizan Sistemas de Gestión de Bases de Datos[1] (SGBD) o bases de datos como se les conoce comúnmente.

El Departamento de Soluciones para la Aduana, como parte de las soluciones dirigidas a satisfacer las necesidades de la AGR, diseña e implementa un conjunto de aplicaciones basándose en la tecnología cliente-servidor.

Este departamento ha venido presentando dificultades con la estandarización de la base de datos, y en Oracle, que es el SGBD con que se trabaja en dicho departamento, se presentan inconvenientes para corregir de forma manual los errores de diseño que se presentan en los diferentes objetos de un esquema. Existen varios estándares internacionales de diseño de base de datos, especialmente para el modelo relacional y en cada proyecto de software se aplican según las necesidades del mismo. El Departamento de Soluciones para la Aduana no es la excepción, en el mismo 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 el proceso de diseño de los esquemas de base de datos. Este estándar permite

[1] **Sistema gestor de base de datos:** por sus siglas en inglés DataBase Management System (DBMS).

realizar diseños de base de datos con la calidad necesaria para el buen funcionamiento del mismo, pero es ignorado en muchos casos, ya sea por desconocimiento, falta de práctica o negligencia por parte del personal que ejecuta el diseño, por lo que en ocasiones se diseñan e implantan modelos de datos con deficiencias.

Los errores de diseño provocan el atraso en el desarrollo de las aplicaciones, debido a que muchos de ellos son detectados 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, pues el primer paso es detectar las deficiencias existentes para luego eliminarlas.

El estudio de la problemática antes mencionada ha desencadenado el siguiente **problema a resolver**:
¿Cómo agilizar el tiempo de detección de errores de diseño, de un esquema de base de datos en Oracle 11g, según el "Estándar de Codificación de Base de Datos" del Departamento de Soluciones para la Aduana?

Para la investigación se ha definido como **objeto de estudio:** los estándares aplicables al diseño de base de datos.
Campo de acción: estándares aplicables al diseño de base de datos en Oracle 11g para el sistema de Gestión Integral Aduanera.

Persiguiendo como **objetivo general:** Desarrollar una herramienta que permita la detección de errores de diseño de un esquema de base de datos en Oracle 11g, garantizando la disminución del tiempo, según el "Estándar de Codificación de Base de Datos" del Departamento de Soluciones para la Aduana.

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

- Realizar el marco teórico de la investigación para identificar la existencia de sistemas homólogos.
- Analizar la información obtenida para definir el estado actual del proceso de detección de errores en los esquemas de base de datos.
- Proponer la solución para determinar los errores de diseño en un esquema de datos en Oracle

11g.

- Implementar la solución para determinar los errores de diseño en un esquema de datos en Oracle 11g.
- Validar la solución.

Proponiendo como **posibles resultados:**

- Descripción de los estándares aplicables al diseño de base de datos en Oracle 11g, para el Departamento de Soluciones para la Aduana.
- Clasificación de los tipos errores según su impacto.
- Clasificación de un esquema de datos según su estado de diseño.
- Propuesta de solución de la herramienta a desarrollar.
- Herramienta para la detección 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 del Departamento de Soluciones para la Aduana. Luego se realizará un estudio de las soluciones existentes, llegando a una solución para el desarrollo de la herramienta. Finalmente se seleccionará la metodología para guiar el desarrollo del sistema.

Capítulo II: Se realiza la propuesta de la herramienta a desarrollar y se generan los artefactos que propone la metodología de desarrollo escogida.

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 al sistema.

Capítulo 1: Fundamentación teórica.

1. Introducción.

Estándar, según el manual de Cobit 4 es: “Una práctica de negocio o producto tecnológico que es una práctica aceptada, avalada por la empresa o por el equipo gerencial de Tecnologías de la Información. Los estándares se pueden implantar para dar soporte a una política o a un proceso, o como respuesta a una necesidad operativa. Así como las políticas, los estándares deben incluir una descripción de la forma en que se detectará el incumplimiento.” (1)

Según el Diccionario de la Real Academia de la Lengua Española, un estándar es: “modelo, norma, patrón o referencia.” (2)

Se puede decir entonces que, un estándar para el diseño de base de datos es, un modelo, norma o práctica aceptada para que cualquier persona sea capaz de comprender mejor lo que esta modelado. Dichos estándares son elaborados en cada entidad según las necesidades de diseño y desarrollo que sea de interés para la propia entidad.

1.1 Reglas para el diseño de base de datos.

1.1.1 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.

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

El proceso de diseño consta de los pasos siguientes:

- **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 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. (3)

1.1.3 Fases del diseño de Base de Datos.

El proceso de diseño de una base de datos cuenta con tres fases fundamentales: Diseño Conceptual, Diseño Lógico y Diseño Físico.

1.1.3.1 Diseño Conceptual.

El objetivo del diseño conceptual, también conocido como modelo conceptual, y que constituye la primera fase de diseño, es obtener una buena representación de los recursos de información, con independencia de usuarios, aplicaciones en particular y sin considerar aspectos como eficiencia de la computadora. Esta primera fase consta de dos momentos: análisis de requisitos, donde se centra el trabajo en definir qué es

lo que se va a representar, y la conceptualización, donde se piensa en cómo se va a proceder para representar lo antes definido.

El diseño conceptual es completamente independiente de los aspectos de implementación, como puede ser el Sistema Gestor de Base de Datos (SGBD).

1.1.3.2 Diseño Lógico.

El diseño lógico es el proceso de construir un esquema de la información que utiliza la empresa, basándose en un modelo conceptual de base de datos específico, independiente del SGBD concreto que se vaya a utilizar (salvo en el modelo) y de cualquier otra consideración física. En esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el SGBD que se vaya a utilizar, como es el modelo relacional. El modelo relacional es el único modelo que ha permitido abordar la fase de diseño lógico aplicando una teoría formal. El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la BD.

1.1.3.3 Diseño Físico.

El objetivo del diseño físico es conseguir una instrumentación lo más eficiente posible del esquema lógico. Para esto se analizan aspectos como las características del sistema operativo, el sistema gestor de base de datos, la herramienta para realizar el diseño, aspectos relacionados con el rendimiento y los requisitos de procesos así como las características del hardware, en fin, cualquier factor cercano con la computadora, para con ello lograr optimizar el consumo de recursos, minimizar el espacio de almacenamiento, proporcionar la seguridad máxima, disminuir los tiempos de respuesta y evitar las reorganizaciones. (4)

1.1.4 Normalización de base de datos.

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.

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.

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

- Cada columna debe tener su nombre único.
- No puede haber dos filas iguales. No se permiten los duplicados.
- Todos los datos en una columna deben ser del 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.

- **Primera Forma Normal (1FN)**

Sea α un conjunto de atributo perteneciente (Є) a la relación R, en donde R está en la Primera Forma Normal si todos los atributos α[n] son atómicos.

- **Segunda Forma Normal (2FN)**

Esta en 2FN si esta en 1FN y si sus atributos no principales dependen de forma completa de la clave principal. Dependencia completa de la clave primaria.

- **Tercera Forma Normal (3FN)**

Está en segunda forma normal y todo atributo no primo, es implicado por la clave primaria en una secuencia no transitiva. Se eliminan las dependencias transitivas.

- **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.

- **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.

- **Quinta Forma Normal (5FN)**

Está en cuarta forma normal y toda dependencia-join viene implicada por claves candidatas. (5)

1.1.5 Estándar de Codificación de Base de Datos del Departamento de Soluciones para la Aduana.

En el Departamento de Soluciones para la 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[2] , procedimientos almacenados 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 batos, 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, con el objetivo de identificar los posibles errores que se puedan presentar.

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 bajo. Además, las palabras seguirán estando todas en singular.

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

- Únicamente se utilizarán caracteres alfabéticos, salvo que por la naturaleza del nombre se necesiten dígitos numéricos. Se prohíbe el uso de caracteres de puntuación o símbolos.
- 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.

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 índices.
PK	Prefijo para las claves primarias.
FK	Prefijo para las claves foráneas.
SEQ	Sufijo para las secuencias

Tabla 1: Abreviaturas.

- 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 a 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 actualizado. 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 como ñ y tilde, estos deben ser sustituidos por símbolos semejantes.

Caracter extraño	Posibles Símbolos
Ñ	nn
Á	a
É	e
Í	i
Ó	o

Ú	U
Ä	A
@	_

Tabla 2: Caracteres extraños.

Para el esquema de **Tablas de Control**[3] (TC) el estándar plantea que:

- Las tablas deben tener invariantemente los siguientes campos:

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

Tabla 3: Atributos Obligatorios de las Tablas de Control.

- 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** donde se especificará de la siguiente manera. En caso de que se vayan a utilizar ciertos valores predeterminados para el campo se especificará:

  “valor1” --> “Significado”

  “valor2” --> “Significado”

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

"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.
- Las claves foráneas tienen que pertenecer a una clave única para garantizar la unicidad de los registros.

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.

[Columnas] Nombre de las columnas separadas por "_".

- 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. (6)

1.2 Diseño e implantación de la base de datos en el Departamento de Soluciones para la Aduana.

El diseño e implantación de una base de datos comienza con 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 el departamento sobre la base de datos.

Generalmente los errores son detectados por los programadores del sistema lo que provoca un atraso ya

que tienen que esperar por que el error sea corregido. Estos errores son detectados y corregidos de forma manual por los integrantes del equipo de base de datos del departamento invirtiendo buena parte del tiempo en detectar los errores, como producto de una búsqueda objeto a objeto, de esta manera se logran revisar alrededor de 100 objetos diarios, dependiendo del tipo de esquema que sea, la cantidad de objetos que posea y la cantidad de errores encontrados en el mismo.

1.3 Ventajas 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, inclusive para personas que no están relacionadas con el ambiente informático, en etapas de análisis y diseño.
- 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.

1.4 Desventaja del uso de estándares bases 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 por cada empresa según las necesidades que estas presenten.

1.5 Principales errores que afectan el diseño de la base de datos en el Departamento de Soluciones para la Aduana.

Entre los principales errores identificados en el Departamento de Soluciones para la 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:N 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 cuales pertenecen.
20. Las tablas TC creadas a partir de relaciones de N:M carecen del prefijo “TC**R**”.
21. Las tablas TC no poseen todos los atributos obligatorios que por la naturaleza del negocio son necesarios.
22. Las tablas TC(R) no poseen un id auto-incrementable como clave primaria.
23. Los campos de las tablas TC no poseen la descripción de su objetivo.
24. Los identificadores de las TC(R) no poseen el mismo nombre de la tabla.
25. Cuando dos TC se relacionan, y una de sus tablas obtiene la clave foránea no posee el código de la tabla padre.
26. Las tablas que poseen las claves foráneas y primarias a la vez poseen mal la combinación del nombre.
27. Todas las tablas que en un esquema tienen el permiso (grant) de lectura, actualización o eliminación para otro usuario, debe tener sinónimo público.

28. Las tablas que por la naturaleza del negocio requieren secuencia no la poseen.
29. Las restricciones que son claves primarias no poseen la estructura adecuada.
30. Las restricciones que son claves foráneas no poseen la estructura adecuada.
31. Los índices no poseen la estructura adecuada.
32. Las secuencias no poseen el sufijo "_SEQ".
33. Los sinónimos deben ser públicos.
34. Las tablas que poseen sinónimos, el nombre del sinónimo no es el mismo nombre que el de la tabla a la cual pertenece.
35. Existen secuencias que no están asociadas a ninguna tabla.
36. Las claves foráneas tienen que pertenecer a una clave única para garantizar la unicidad de los registros.

1.6 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 afectan 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. Por otra parte si los nombres de las tablas exceden los 26 caracteres imposibilitan la creación de secuencias y sinónimos pues el gestor que se utiliza posee dicha métrica en la creación de los mismos, es decir, valida que las secuencias y sinónimos no posean un nombre demasiado largo ya que se puede perder así la lógica del mismo tornándose ambiguo.

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 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 que 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 TC no poseen todos los atributos obligatorios que por la naturaleza del negocio son necesarios.
- 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 tablas que por la naturaleza del negocio requieren secuencia no la poseen.

- 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:N 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 "TC**R**".
- 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.

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 los **Índices** los errores son los siguientes:

Errores Menos Críticos:

- Los índices 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[4] 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. (7)

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.

1.7.1 ER/Studio.

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

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[5] y posee un repositorio para el modelado. (8)

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, así como las restricciones de integridad referencial de las claves foráneas que describen el comportamiento de agregaciones o especializaciones y 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 del Departamento de Soluciones para la Aduana por lo que no es una solución viable para evaluar el estado de los esquemas de base de datos del departamento.

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

[5] **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.

existentes y guardar el proyecto en su formato original (XML[6]) 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. (9)

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.

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" del Departamento de Soluciones para la 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 el 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.

Determina los tipos de esquemas según la metodología seleccionada y notifica los errores a medida que el modelo vaya construyéndose. Ayuda en la seguridad de los datos mediante el diccionario de los datos que bloquea por niveles al registro, al archivo y al proyecto, y niveles de control de acceso. (10)

[6] **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.

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 utilizarla en el departamento ya que no cumple con muchas de las especificaciones que en este se requieren.

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. (11)

Aunque la herramienta posee características que la hacen muy útil en el momento del diseño, no posee un modulo que sea capaz de identificar y 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. (12)

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

1.8 Principales sistemas de gestión de base de datos.

Un Sistema de Gestión de Base de Datos es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. (13)

1.8.1 PostgreSQL.

PostgreSQL es un Sistema de Gestión de Base de Datos Relacional (SGBDR) orientada a objetos, libre (gratuito) y de código abierto. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es controlado por una sola empresa sino que está dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. (14)

1.8.2 MySQL.

MySQL es un Sistema de Gestión de Base de Datos Relacional y multiusuario. Brinda la posibilidad de escoger bajo que licencia se puede utilizar, libre o propietaria[7]. Está desarrollado en su mayor parte en ANSI C. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. (15)

[7] Para aquellas empresas que quieran incorporarlo en productos privativos, estas deben comprar a MySQL una licencia específica que les permita este uso.

1.8.3 Oracle.

Oracle es un Sistema de Gestión de Base de Datos objeto-relacional, desarrollado por la Corporación de Oracle (Oracle Corporation). Se considera como uno de los sistemas de bases de datos más completos, destacando el soporte de transacciones, la estabilidad, escalabilidad y su soporte multiplataforma. Posee licencia privativa aunque cuanta con Oracle Database Express Edition en su versión 10g que es gratuita y compatible con las demás ediciones de Oracle Database 10gR2 y Oracle Database 11g. (16)

1.9 Tecnología a utilizar.

1.9.1 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 GPL de GNU[8] excepto las bibliotecas de clases de Sun.

Su sintaxis es muy parecida a la de C y C++ ya que su desarrollo fue inspirado en los mismos, siempre buscando eliminar errores que suelen inducirse en lo que respecta a la manipulación de memoria y punteros.

Los programas en Java generalmente son compilados a un lenguaje intermedio llamado ByteCode[9], que luego son interpretados por una máquina virtual (JVM[10]). Esta última sirve como una plataforma de abstracción entre el hardware y el lenguaje permitiendo que se pueda ejecutar el programa sobre cualquier arquitectura. Señalar que la compilación a código máquina también es posible, brindando mayor eficiencia en la ejecución del programa pero limita su característica multiplataforma.

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

- **Simple:** Elimina la complejidad de los lenguajes como C y C++ dando paso al contexto de los lenguajes modernos orientados a objetos.
- **Orientado a objetos:** Soporta las características esenciales del paradigma de la programación orientada a objetos: encapsulamiento[11], herencia[12] y polimorfismo[13].

[8] **GNU:** Proyecto informático creado para la construcción de sistemas operativos libres.
[9] **ByteCode:** Pseudocódigo prácticamente al nivel de código máquina.
[10] **JVM:** Maquina virtual de java, por sus siglas en Java Virtual Machine.
[11] **Encapsulamiento:** Mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el objeto se implementa.

- **Robusto:** Elimina el uso de apuntadores para referenciar áreas de memoria, además, libera al desarrollador de la necesidad de liberar la memoria que la aplicación ya no usa. También requiere la declaración explícita tanto de los tipos de datos como de los métodos.
- **Multiplataforma:** El mismo código Java que funciona en un sistema operativo, funciona en cualquier otro que tenga instalada la máquina virtual de Java.
- **Multitareas:** Permite la ejecución concurrente de varios procesos ligeros o hilos de ejecución.

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. (17)

1.9.2 ¿Por qué utilizar Java?

Se decide utilizar Java como lenguaje de programación para el desarrollo de la aplicación ya que el mismo posee flexibilidad, eficacia, portabilidad, es robusto y sencillo de aprender. Por otra parte es un lenguaje multiplataforma, se encuentra bajo la licencia GNU/GPL, ofrece programación basada en hilos de ejecución en un mismo programa, permitiendo realizar varias acciones al mismo tiempo. Brinda soporte de conexión a múltiples gestores de base de datos, se integra particularmente con Oracle ya que son desarrolladas por la misma compañía. Posee un gran número de frameworks[14] para facilitar el desarrollo de todo tipo de aplicaciones. En el estudio realizado sobre las herramientas de administración de base de datos se obtuvo como resultado que un 64,77% son aplicaciones de escritorio robustas, por ello la necesidad de escoger un lenguaje de programación con las características de Java. Se puede decir además que dentro de los requerimientos no funcionales se encuentra la necesidad de que sea una aplicación con tecnología cliente servidor, lo que puede ser garantizado igualmente por una aplicación de escritorio, siempre y cuando el software esté en una computadora cliente y esta realice solicitudes a un servidor, que a su vez sean respondidas a través de la red, tampoco se encuentra la necesidad de alta accesibilidad al sistema, ya que al sistema solo accederá el administrador de base de datos del proyecto.

[12] **Herencia:** Mecanismo basado en clases, por medio del cual una clase se deriva de otra de manera que extiende su funcionalidad.
[13] **Polimorfismo:** se refiere a la posibilidad de enviar un mensaje a un grupo de objetos cuya naturaleza puede ser heterogénea.
[14] **Framework:** Es un conjunto de clases que cooperan y forman un diseño reutilizable para un tipo específico de software que brinda una guía arquitectónica que parte el diseño en clases abstractas y define sus responsabilidades y colaboraciones. Es una infraestructura software que crea un entorno común para integrar aplicaciones e información compartida dentro de un dominio dado.

De esta forma se cumple con uno de los principios elementales de la seguridad el “Mínimo acceso”, el cual garantiza se le asignen o garanticen la condiciones elementales de accesibilidad y permisos a las aplicaciones.

1.9.3 Entorno de Desarrollo Integrado.

Un Entorno de Desarrollo Integrado, llamado también IDE[15], 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. (18)

1.9.3.1 NetBeans 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. (19)

1.9.3.2 ¿Por qué utilizar NetBeans?

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 disimiles características entre las que se encuentra en completamiento de código, el soporte para varios lenguajes, soporte para disímiles plugins que lo hace un IDE firme y confiable.

1.9.4 PostgreSQL como gestor de base de datos.

PostgreSQL es un Sistema de Gestión de Base de Datos Relacional (SGBDR) orientada a objetos, libre, gratuito y de código abierto (Open Source). Como muchos otros proyectos de código abierto, el desarrollo

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

de PostgreSQL no es controlado por una sola empresa sino que está dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Mediante un sistema denominado MVCC[16] PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo modificación. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos. (20)

1.9.5 ¿Por qué utilizar PostgreSQL?

Se escoge PostgreSQL como gestor de base de datos ya que posee varias características que lo hacen ser un gestor potente en cuanto a rendimiento, presenta una estabilidad muy alta, gran 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 esta aplicando.

1.9.6 PgAdmin III como herramienta para administrar PostgreSQL.

En pgAdmin3 se puede trabajar con casi todos los objetos de la base de datos, examinar sus propiedades y realizar tareas administrativas. Entre estos objetos encontramos columnas, restricciones, dominios, funciones, índices, esquemas, secuencias, tablas entre otros.

Una característica interesante de pgAdmin3 es que, cada vez que se realiza alguna modificación en un objeto, escribe la(s) sentencia(s) SQL correspondiente(s), lo que hace que, además de una herramienta muy útil, sea a la vez didáctica. También incorpora funcionalidades para realizar consultas, examinar su ejecución y trabajar con los datos. Es multiplataforma y su licencia es libre. (21)

1.9.6.1 ¿Por qué utilizar PgAdmin III?

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

[16] **MVCC:** Control de concurrencia para múltiples versiones.

libre y multiplataforma.

1.9.7 DB Designer para diseñar la base de datos.

DBDesigner 4 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. DBDesigner se compara con productos de diseño de Base de Datos como Rational Rose y Erwin. 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, Oracle, MSSQL y cualquier ODBC, por lo que se puede utilizar con casi todas las bases de datos existentes. (22)

1.9.7.1 ¿Por qué utilizar DBDesigner?

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 tus bases de datos.

1.10 Metodología de desarrollo de software.

El desarrollo de software no es trabajo fácil y lograr que un sistema informático sea eficiente y que cumpla con los requerimientos planteados, es una tarea realmente intensa. Por ello surgen a principios de los 70s las metodologías para el desarrollo del software. Estas no son más que: un conjunto de procesos y técnicas que permiten estructurar, planificar y controlar todo proceso de desarrollo. Tienen como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. No existe una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto, exigiéndose así que el proceso sea configurable.

Las metodologías de desarrollo están divididas en dos grupos de acuerdo con sus características y los objetivos que persiguen: ágiles y robustas.

Las metodologías robustas o tradicionales son aquellas que hacen un mayor énfasis en la planificación y control del proyecto, en la especificación precisa de requisitos y modelado. Estas metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Estas metodologías se centran en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar. El Proceso Unificado de Modelado (RUP[17]), Microsoft Solution Framework (MSF) y Modelo en espiral Ganar-Ganar (Win-Win Spiral Model) son ejemplos de metodologías tradicionales.

Las metodologías ágiles son aquellas que permiten incorporar cambios con rapidez en el desarrollo de software, cuyo objetivo es esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se caracterizan por hacer énfasis en la comunicación cara a cara, es decir, se basan en una fuerte y constante interacción, donde clientes y desarrolladores trabajan constantemente juntos, estableciéndose así una estrecha comunicación. Estas metodologías están orientadas al resultado del producto y no a la documentación; exigen que el proceso sea adaptable, permitiendo realizar cambios de último momento. Son una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. Entre estas metodologías se encuentran Programación Extrema (XP[18]), Melé[19], Proceso Unificado Ágil (AUP[20]) y Método de desarrollo de sistemas dinámicos (DSDM[21]). (23)

1.10.1 Metodología a utilizar.

La metodología SXP es un híbrido compuesto por las mejores prácticas de las metodologías ágiles Melé y

[17] **RUP**: por sus siglas en inglés Rational Unified Procces.
[18] **XP**: por sus siglas en inglés Xtreme Programming.
[19] **Melé**: en inglés Scrum.
[20] **AUP**: por sus siglas en inglés Agile Unified Process.
[21] **DSDM**: por sus siglas en inglés Dynamic Systems Development Method.

Programación Extrema. Entre los objetivos de la creación de dicha metodología se encuentra que fue creada para ser aplicada a proyectos cuyo desarrollo se basa en tecnologías libres.

SXP ofrece una estrategia tecnológica a partir de la introducción de procedimientos ágiles, permitiendo actualizar los procesos de software para el mejoramiento de la actividad productiva, fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo y ayudando al líder del proyecto a tener un mejor control del mismo.

Consta de 4 fases principales y varias actividades entre las que se encuentran:

- **Planificación-Definición:** En esta primera fase es donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
  - ✓ Lista de Reserva del Producto (LRP).
  - ✓ Historias de usuario.
  - ✓ Diagrama de clases del diseño.
- **Desarrollo:** Es donde se realiza la implementación del sistema hasta que esté listo para ser entregado.
  - ✓ Estándar de codificación.
  - ✓ Plan de release.
  - ✓ Plan de pruebas.
- **Entrega:** Puesta en marcha.
  - ✓ Entrega de toda la documentación generada.
- **Mantenimiento:** Donde se realiza el soporte para el cliente.
  - ✓ Gestión de cambios.

SXP propone los siguientes roles para el trabajo en equipo:

- Líder del proyecto.
- Gerente (Management).
- Especialista.
- Cliente.
- Consultor.

- Equipo del proyecto. (24)

1.10.2 ¿Porque utilizar SXP?

Se selecciona SXP como metodología de desarrollo a utilizar pues está especialmente diseñada para ser aplicada en equipos de trabajo pequeños, con requisitos imprecisos o cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Ayuda a que trabajen todos en la misma dirección, con un objetivo claro, permitiendo además seguir de forma clara el avance de las tareas a realizar, permitiendo a los jefes apreciar el progreso del trabajo día a día. Se puede afirmar que las dimensiones del sistema a desarrollar aseguran una culminación exitosa con la implementación dicha metodología.

1.11 Conclusiones.

En este capítulo se realizó un análisis de las principales reglas para el diseño de bases de datos relacionales. Se identificaron los tipos de errores que afectan a la base de datos del departamento según el estándar establecido en este y se clasificaron según su impacto. Se analizaron las principales herramientas CASE que apoyan la realización de modelos de datos y que cumplen de alguna manera con estándares de diseño. Esta investigación arrojo como conclusión que:

- A nivel internacional existen disímiles herramientas que aplican estándares de diseño cuando se modela un esquema de base de datos pero ninguna de estas pueden ser aplicadas en el Departamento de Soluciones para la Aduana ya que no cumplen con las características necesarias para validar el estándar que se aplica en este departamento, siendo muchas de ellas software privativo, y en el caso de las que son software libre su asimilación y modificación no sería viable, dada la gran cantidad de cambios necesarios.
- Se pudo constatar que todas las herramientas CASE que fueron objeto de investigación tenían en común la característica de ser herramientas de escritorio.

Por lo expuesto anteriormente se propone implementar una herramienta que se ajuste a las particularidades propias del departamento, que sea de escritorio y guiada por la metodología de desarrollo SXP.

Capítulo 2: Diseño de la propuesta de solución.

2. Introducción.

En este capítulo se hace un levantamiento de los requisitos funcionales del sistema a implementar, además de la descripción de los artefactos del diseño que se generan durante el flujo de trabajo. Se muestran también los prototipos de interfaz de usuario correspondientes a los diferentes funcionalidades del sistema.

2.1 Propuesta de la solución.

Se desea una herramienta que ayude a identificar 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 o evaluada 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 exportación de los errores, adicionar, modificar y eliminar conexiones a las bases de datos a ser analizadas e insertar, listar y eliminar tipos de errores.

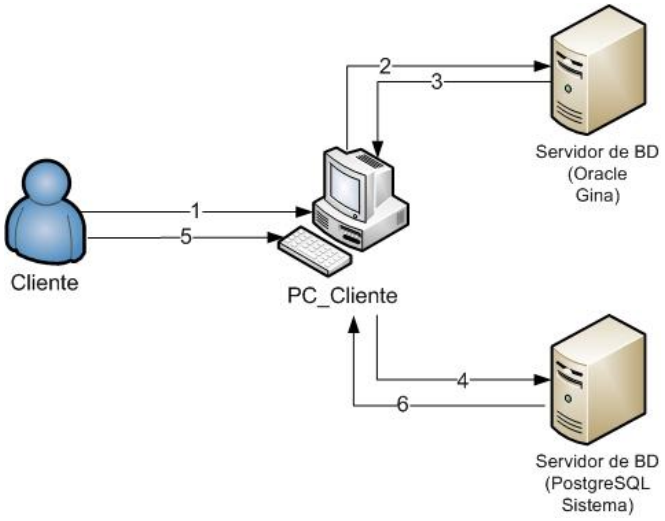


Figura 1: Propuesta de solución.

1. El cliente se crea una conexión a la base de datos que desea revisar.
2. El sistema se conecta a la base de datos descrita por el cliente.
3. Se extrae toda la información a ser revisada.
4. Se almacena en la base de datos del sistema la información revisada.
5. El usuario solicita la información para evaluar o solo consultarla.
6. Se extrae la información que el usuario solicita de la base de datos del sistema.

2.2 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 **p**eso **p**osible de los errores en las **t**ablas, **tottab** es el **tot**al de **tab**las del esquema y **Spet s**uma de los **p**esos de los **e**rrores críticos en **t**ablas.
- **Ppsec** es el **p**eso **p**osible de los errores en las **sec**uencias, **totsec** es el **tot**al de **sec**uencias del esquema y **Spesec s**uma de los **p**esos de los errores críticos en **sec**uencias.
- **Ppsin** es el **p**eso **p**osible de los errores en los **sin**ónimos y **totsin** es el **tot**al de los **sin**ónimos del esquema y **Spesin s**uma de los **p**esos de los **e**rrores críticos en **sin**ónimos.

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:

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 **porc**iento del **p**eso de **e**rrores **h**allados, **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.**

2.2.1 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

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**.

2.3 Requisitos.

Con el transcurrir de los años se ha podido constatar que los requisitos juegan un papel fundamental en el

desarrollo de software, ya que marcan el punto de partida para actividades como la planeación, básicamente en lo que se refiere a las estimaciones de tiempos y costos. Además la especificación de los requerimientos es la base que permite verificar si se alcanzan o no los objetivos establecidos para el correcto funcionamiento del software a desarrollar, ya que son un reflejo detallado de las necesidades de los clientes o usuarios del sistema.

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."

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 el sistema debe hacer. "Son declaraciones de los servicios que debe brindar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares."

Los requisitos no funcionales: "Son restricciones de los servicios o funciones ofrecidos por el sistema." No se refieren a funcionalidades específicas del sistema, 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." (25)

Con la captura de requisitos se genera la Lista de Reservas del Producto, ésta se encuentra compuesta por una lista priorizada y organizada de los requisitos funcionales y no funcionales que debe poseer el producto. Esta tiene como objetivo cubrir las cualidades requeridas en el software y determinar el orden en que se cumplimentarán durante las iteraciones o sprint[22]; además, incluye las características que el

[22] **Sprint:** Ciclo iterativo o período de tiempo en el que se implementa o mejora una funcionalidad del sistema. Durante un sprint el producto puede ser diseñado, implementado y probado para producir nuevos incrementos.

producto debe tener. (26)

A continuación se muestra Lista de Reservas del Producto.

Asignado a	IDR[23]	Descripción	Estimación[24]	Estimado por
Requisitos Funcionales				
Prioridad Muy Alta				
Laura Elena	01	Crear conexión a la base de datos.	1 día	Laura Elena
Laura Elena	02	Modificar conexión a la base de datos.	1 día	Laura Elena
Laura Elena	03	Listar conexiones a la base de datos.	1 día	Laura Elena
Laura Elena	04	Eliminar conexión de la base de datos.	1 día	Laura Elena
Laura Elena	05	Revisar errores en las tablas.	2 semanas	Laura Elena
Laura Elena	06	Revisar errores en los sinónimos.	4 días	Laura Elena
Laura Elena	07	Revisar errores en las restricciones.	1 semana	Laura Elena

[23] Identificador del requisito.
[24] Estimación de cada uno de los requerimientos para su implementación por semanas.

Laura Elena	08	Revisar errores en las secuencias.	4 días	Laura Elena
Laura Elena	09	Revisar errores en los índices.	4 días	Laura Elena
Prioridad Alta				
Laura Elena	10	Revisar esquema completo.	1 semana	Laura Elena
Laura Elena	11	Evaluar esquema.	2 semanas	Laura Elena
Laura Elena	12	Listar errores hallados.	4 días	Laura Elena
Laura Elena	13	Mostrar listado de reportes.	2 días	Laura Elena
Laura Elena	14	Eliminar reporte.	1día	Laura Elena
Laura Elena	15	Generar reporte de errores hallados.	2 semanas	Laura Elena
Prioridad Media				
Laura Elena	16	Identificar esquemas de la base de datos a analizar.	1 día	Laura Elena
Laura Elena	17	Insertar tipo de error.	3 días	Laura Elena
Laura Elena	18	Modificar tipo de error.	3 días	Laura Elena
Laura Elena	19	Eliminar tipo de error.	2 días	Laura Elena

Prioridad Baja				
Laura Elena	20	Exportar listado de tipo de errores.	3 días	Laura Elena
Laura Elena	21	Exportar listado de reportes.	3 días	Laura Elena
Laura Elena	22	Exportar errores hallados.	3 días	Laura Elena
Laura Elena	23	Exportar reporte.	3 días	Laura Elena
Requisitos No Funcionales				
Portabilidad				
	1	El sistema debe ser multiplataforma, haciendo énfasis en Linux y Windows.		
Rendimiento				
	2	Debe poseer un elevado nivel de eficiencia, teniendo en cuenta que las transacciones de datos pueden ser elevadas, así como el volumen de la Base de Datos. Por tanto se requiere el desarrollo de mecanismos y técnicas de programación óptimas para el lenguaje de programación seleccionado.		
Seguridad				

	3	Garantizar que las transacciones de datos se ejecuten por vías seguras, ya sea en la red o por soporte físico.		
Software				
	4	Los usuarios deben tener instalado como sistema operativo Windows o Linux, para PC o MacOS para Macintosh.		
	5	La PC cliente debe tener instalado el entorno de ejecución de java, Java Runtime Enviroment (JRE) 1.6 o superior.		
	6	La Base de Datos debe poseer una instalación de PostgreSQL 8.4 o superior.		
	7	La Base de Datos debe poseer una instalación de Oracle 9g o superior.		
Hardware				
	8	**Cliente:** 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, 40 GB de disco duro o más. Mínimo de memoria RAM 128 Mb, adaptador de Red y conectividad.		
	9	**Servidor de Base de Datos:** La PC donde se encontrará el servidor de base de datos deberá poseer las siguientes características un CPU Intel(R) Core2 Duo a 64 bits con 2.2 GHz, con memoria RAM de 4 GB y capacidad de disco a partir de 40 GB.		

Tabla 4: Requisitos del sistema.

2.4 Historias de Usuario.

Las historias de usuario (HU) son la técnica utilizada en XP para especificar los requisitos del software, lo que equivaldría a los casos de uso en el proceso unificado. Las mismas son escritas por los clientes como las tareas que el sistema debe hacer y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas. Son escritas en lenguaje natural, no excediendo su tamaño de unas pocas líneas de texto.

Las historias de usuario guían la construcción de las pruebas de aceptación, elemento clave en XP (deben generarse una o más pruebas para verificar que la historia ha sido correctamente implementada) y son utilizadas para estimar tiempos de desarrollo. En este sentido, sólo proveen detalles suficientes para hacer una estimación razonable del tiempo que llevará implementarlas. En el momento de implementar una historia de usuario, se debe detallar a través de la comunicación con el cliente. Estas son la base para las pruebas funcionales. (26)

<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Numero: 01</td><td colspan="2">Nombre Historia de Usuario: Crear conexión a la base de datos.</td></tr>
<tr><td colspan="3">Modificación de Historia de Usuario Número: Ninguna</td></tr>
<tr><td colspan="2">Usuario: Laura Elena Torres</td><td>Iteración Asignada: 1</td></tr>
<tr><td colspan="2">Prioridad en Negocio: Muy Alta</td><td>Puntos Estimados: 1 día</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: Media</td><td>Puntos Reales: 1 día</td></tr>
<tr><td colspan="3">Descripción: La historia de usuario permite la creación de una conexión a la base de datos a la cual se conectara el usuario para ser analizada. Para establecer la conexión se necesitan los siguientes datos: nombre, servidor, base de datos a la que se va a conectar, usuario, contraseña y puerto. Posibilita almacenar el usuario y la contraseña.</td></tr>
<tr><td colspan="3">Observaciones:
1. Debe existir al menos una conexión a una base de datos.
2. Debe estar seleccionado el esquema que se desea revisar.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz:</td></tr>
</table>

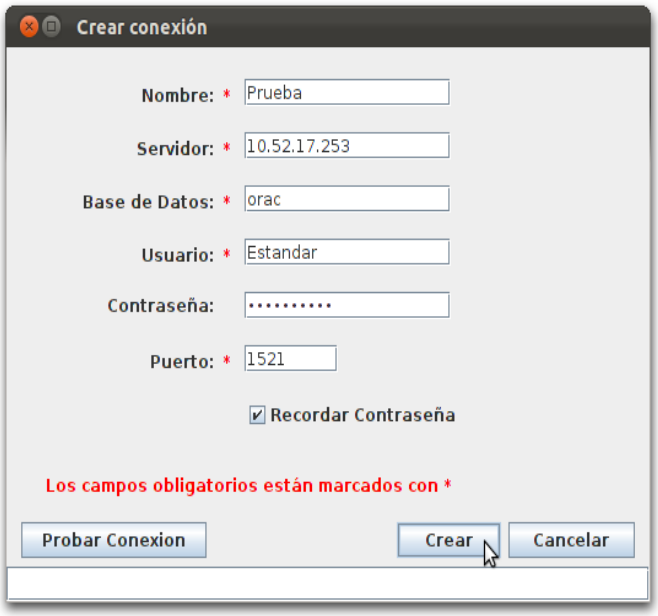
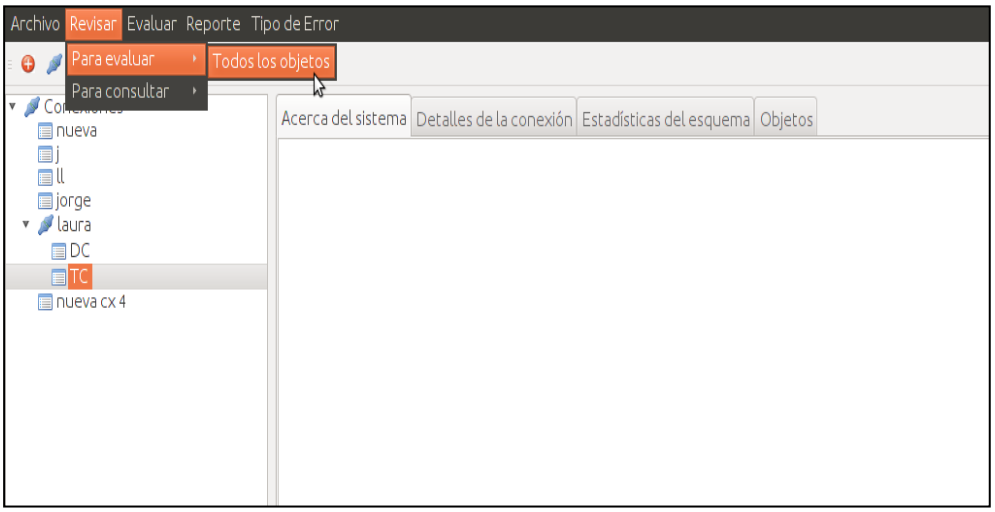
Interfaz de conexión a la		**Usuario 1.** Crear base de datos.

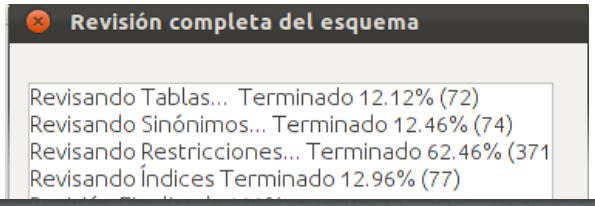
Tabla 5: HU Crear conexión a la base de datos.

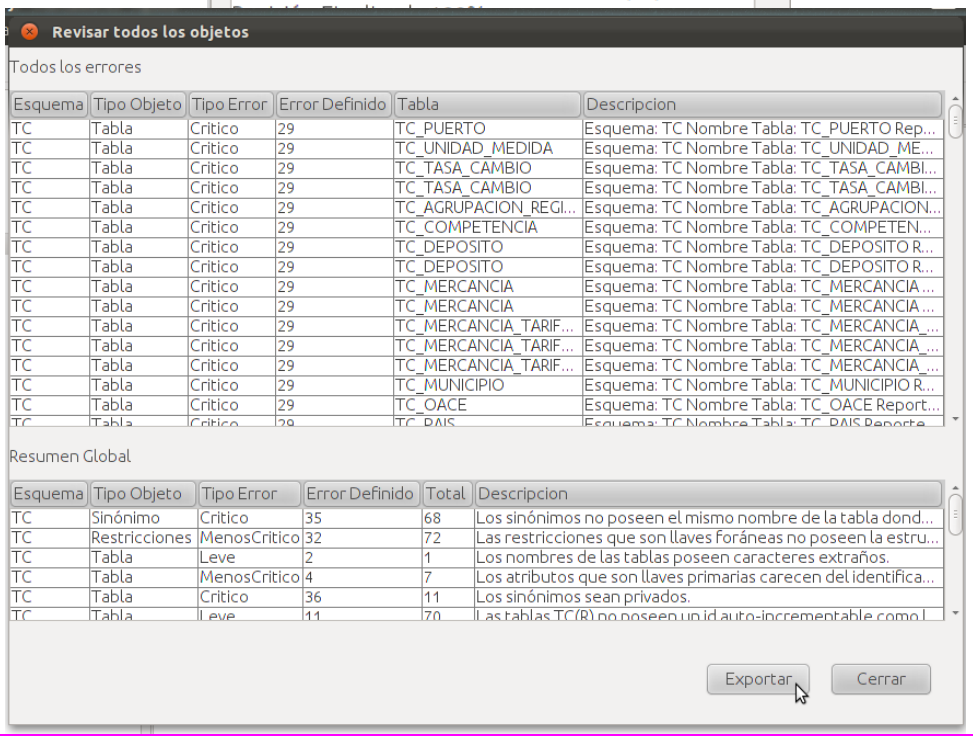
Historia de Usuario	
Numero: 10	**Nombre Historia de Usuario:** Revisar esquema completo.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Laura Elena Torres	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1semana
Descripción: Permite revisar los principales objetos de un esquema de base de datos. El esquema a revisar es seleccionado por el usuario. Luego de este proceso se muestra el resultado de la revisión con los errores encontrados en el esquema brindándole la posibilidad al usuario de exportar dicho resultado. En el resultado se muestra el esquema al que pertenece la revisión y los errores de forma detallada, lo que se	

encontró y lo que se esperaba, con un mensaje indicándole al revisor cómo debe corregir dicho error. Antes de mostrar el resultado este es almacenado en la base de datos para su posterior evaluación.
Observaciones: 3. Debe existir al menos una conexión a una base de datos. 4. Debe estar seleccionado el esquema que se desea revisar.

Prototipo de Interfaz:





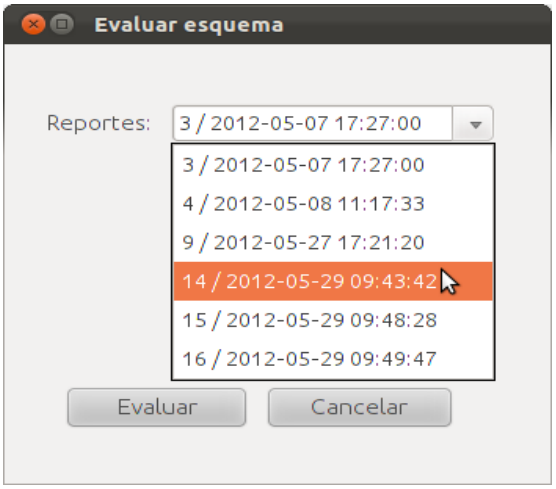


Interfaz de Usuario 2. Revisar esquema completo.

Tabla 6: HU Revisar esquema completo.

Historia de Usuario	
Numero: 11	**Nombre Historia de Usuario:** Evaluar esquema.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Laura Elena Torres	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2 semanas
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2 semanas
Descripción: Le muestra al usuario que tan mal se encuentra un esquema. Ubica al esquema en un rango según la cantidad total de errores y de estos la cantidad de errores críticos. El esquema se puede encontrar en tres estados diferentes funcional, no funcional y parcialmente funcional. Luego de la evaluación, se muestra un reporte con la cantidad total de errores que se encontraron en la revisión del esquema, la cantidad de errores críticos, menos críticos y leves, el rango en que se encuentra el esquema, la fecha y hora en que se realizo la evaluación y el nombre del esquema evaluado. Brinda	

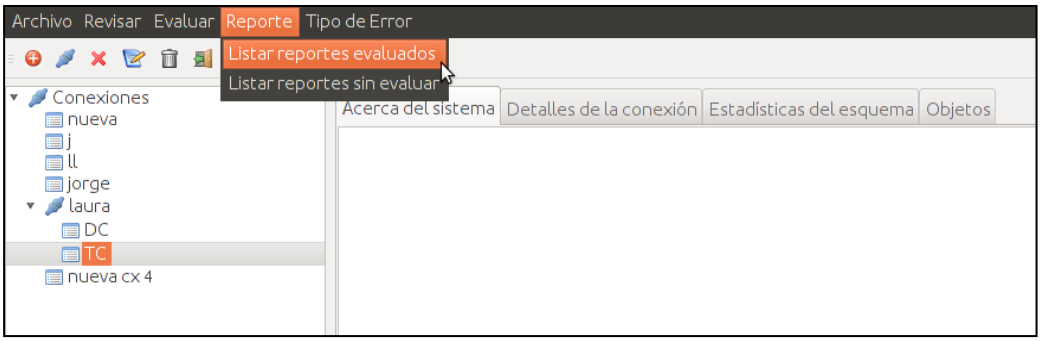
la posibilidad de exportar el reporte. Antes de mostrar el reporte este es almacenado en la base de datos para su posterior uso.
Observaciones: 1. Debe existir al menos una conexión a una base de datos. 2. Debe estar seleccionado el esquema a evaluar. 3. Debe existir al menos una revisión completa del esquema almacenada en la base de datos.
Prototipo de Interfaz:

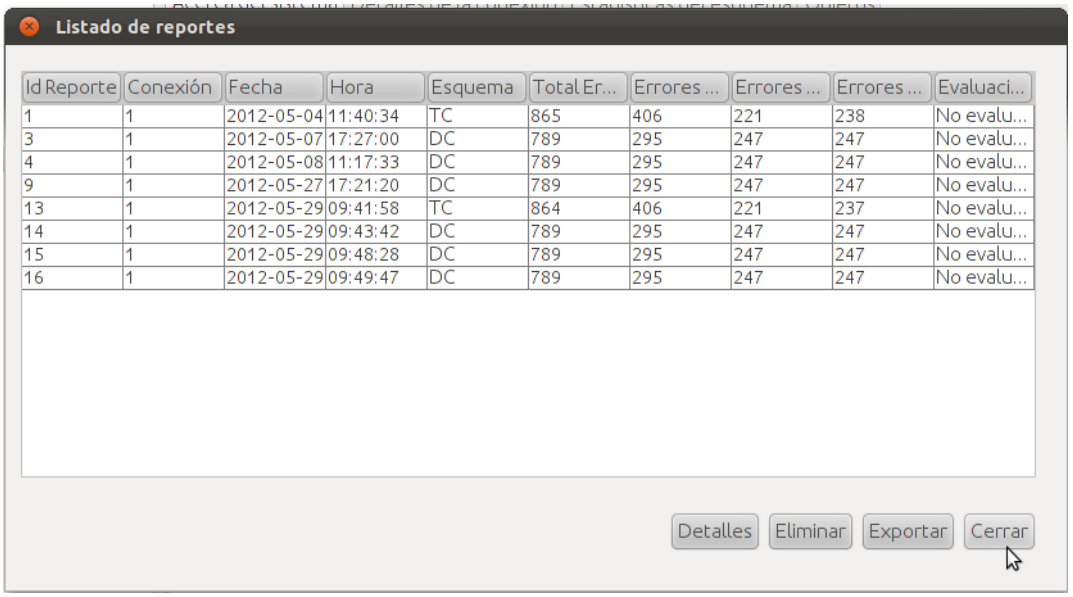


Interfaz de Usuario 3. Evaluar esquema.

Tabla 7: HU Evaluar esquema.

Historia de Usuario	
Numero: 13	**Nombre Historia de Usuario:** Mostrar listado de reportes.
Modificación de Historia de Usuario Número: Ninguna	

Usuario: Laura Elena Torres	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2 días
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2 días
Descripción: Le muestra al usuario la cantidad de reportes realizados. Muestra los detalles de un reporte cuando el usuario lo desee.	
Observaciones: 4. Debe existir al menos una conexión a una base de datos. 5. Debe estar seleccionada al menos una base de datos. 6. Deben estar guardados los reportes realizados a los diferentes esquemas de la base de datos que se encuentra seleccionada. 7. Debe seleccionar un reporte para poder consultarlo.	
Prototipo de Interfaz: 	



Interfaz de Usuario 4. **Mostrar listado de reportes.**

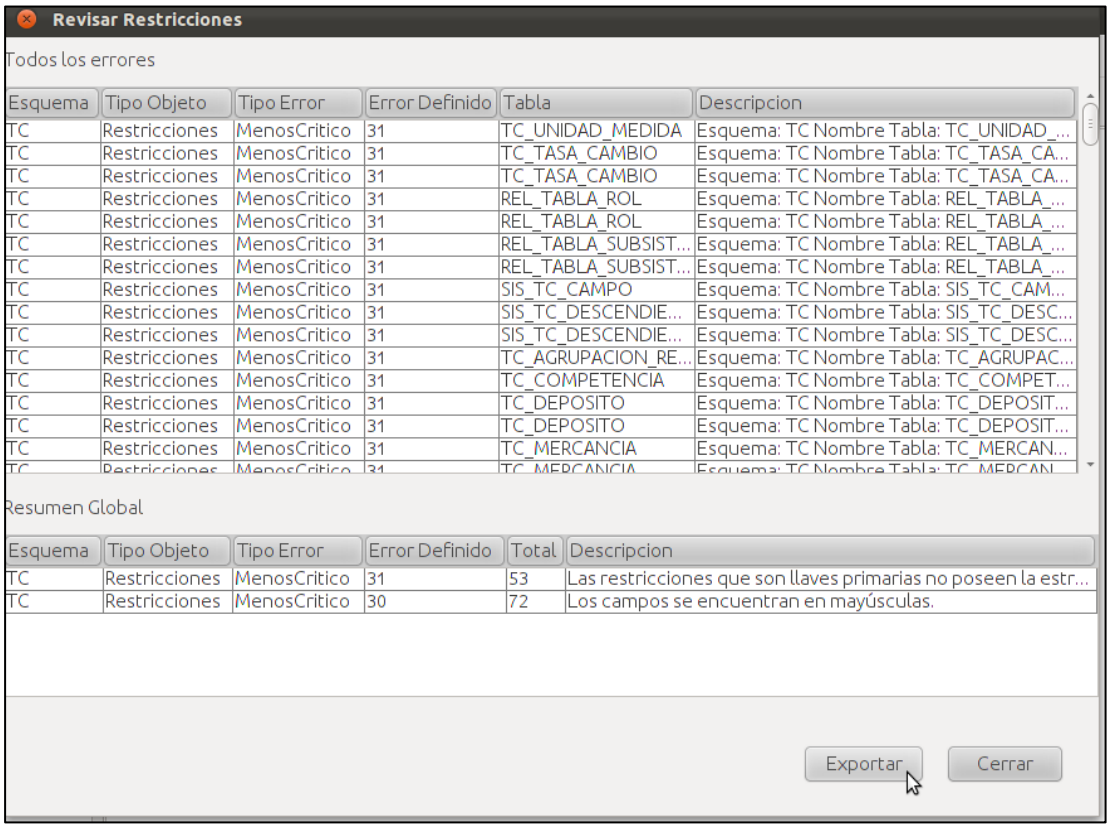
Tabla 8: HU Mostrar listado de reportes.

Historia de Usuario	
Numero: 12	**Nombre Historia de Usuario:** Listar errores hallados.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Laura Elena Torres	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 4 días
Riesgo en Desarrollo: Medio	**Puntos Reales:** 4 días
Descripción: Le muestra al usuario un reporte con los errores hallados indicándole de forma detallada que es lo que debe ser revisado para ser corregido.	

Observaciones:

1. Debe existir al menos una conexión a la base de datos.
2. Debe al menos existir una revisión realizada a un esquema.

Prototipo de Interfaz:



Interfaz de Usuario 5. Generar reporte de errores hallados.

Tabla 9: HU Generar reporte de errores hallados.

2.5 Diagrama de clases del diseño.

El diagrama de clases es utilizado para visualizar las relaciones entre las clases que involucran 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 esta 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.

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.

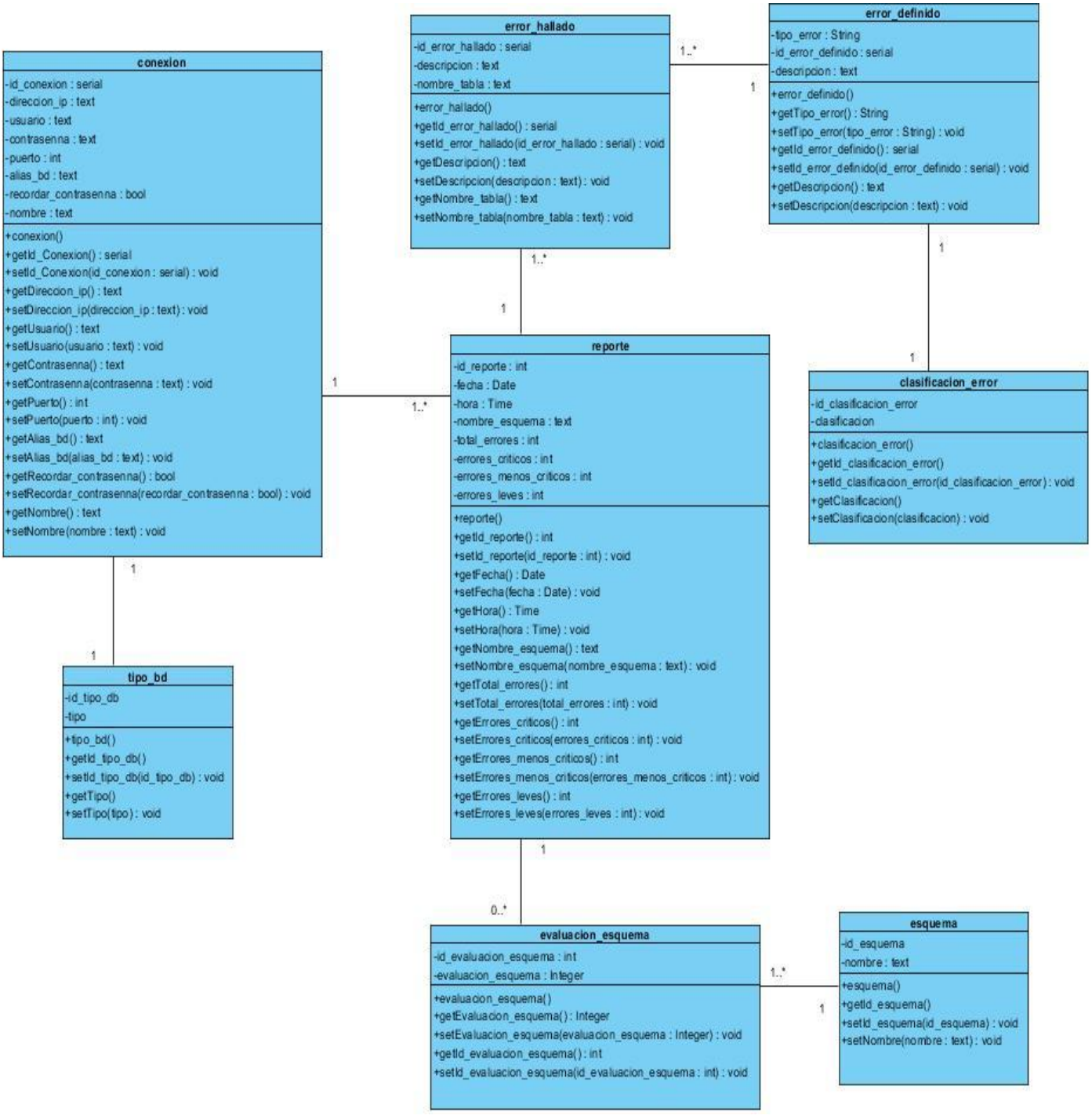


Figura 2: Modelo de clases del diseño.

Entre las principales clases del diagrama anteriormente mostrado se encuentran error_hallado, reporte y conexión.

En la tabla error_hallado se encuentran los atributos que se a continuación se muestran con sus respectivos métodos de acceso. En dicha tabla es donde se almacenan todos los errores que se encontraron en el proceso de revisión para luego ser mostrados en el reporte.

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	

Tabla 10: Especificación de la entidad error_hallado del diagrama de clases.

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

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

Tabla 11: Especificación de la entidad reporte del diagrama de clases.

En la conexión se encuentran los atributos que a continuación se muestran 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.

Nombre: Conexión	
Tipo de clase: Entidad	
Atributo	**Tipo de dato**
id_conexion	serial
direccion_ip	text
usuario	text
contrasenna	Text
puerto	int
tipo_bd	text
alias_bd	text
recordar_contrasenna	bool
nombre	text
Conexion()	
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	
getTipo_bd() : text	
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

Tabla 12: Especificación de la entidad conexión del diagrama de clases.

2.6 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 maquina abstracta con la que interactúan los usuarios. (28)

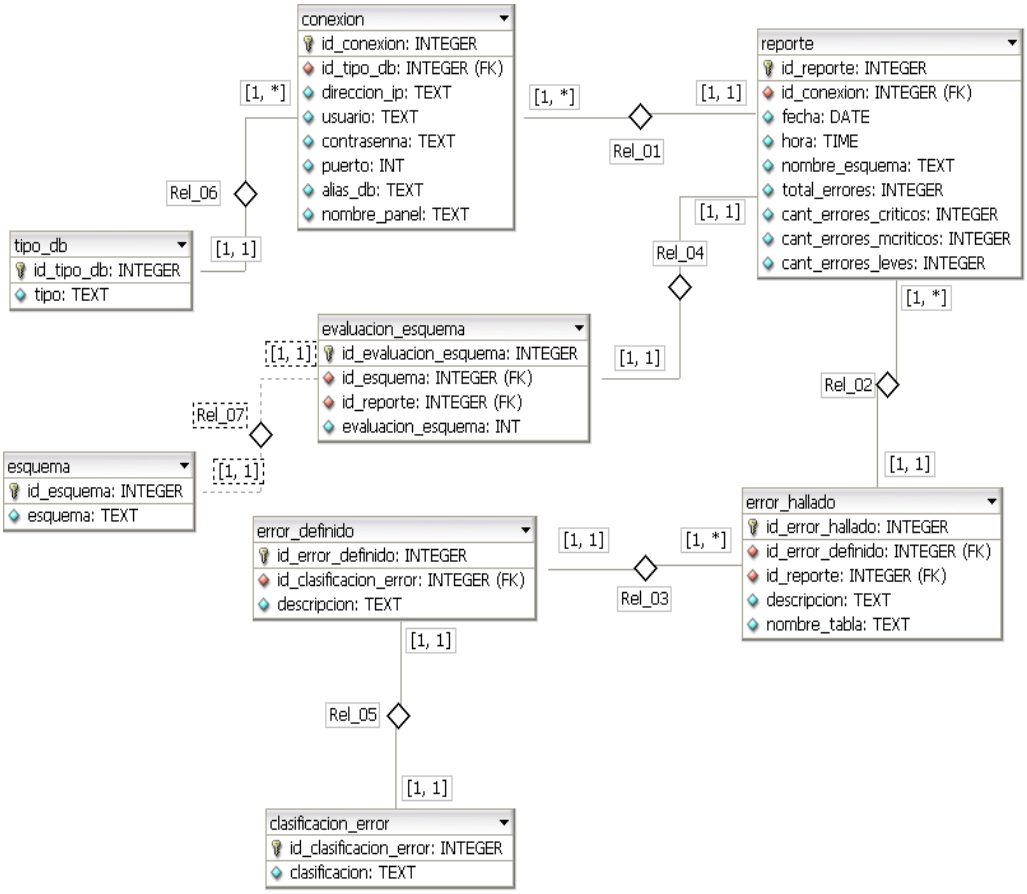


Figura 3: Modelo de datos.

2.6.1 Descripción de las entidades del modelo entidad relación.

- **Entidad conexion:** Esta entidad es la encargada de gestionar la conexión la base de datos que va a ser evaluada.
- **Entidad reporte:** En esta entidad se almacenan los reportes generados para luego ser mostrados o eliminados.
- **Entidad error_hallado:** Se almacenan todos los errores que se encuentren durante el proceso de revisión de los diferentes esquemas.
- **Entidad error_definido:** Se encuentran almacenados todos los errores definidos por el Departamento de Soluciones para la Aduana.
- **Entidad 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.
- **Entidad clasificacion_error:** En esta entidad se almacenan las clasificaciones de los errores, estas pueden ser crítico, menos crítico y leve.
- **Entidad tipo_db:** En esta entidad se almacenan los tipos de bases de datos, entre las que se encuentran Oracle, MySQL y PostgreSQL.
- **Entidad esquema:** En esta entidad se almacena el nombre de todos los esquemas que componen una base de datos.

2.7 Conclusiones

En este capítulo se describió de manera detallada la propuesta de solución del sistema a implementar. Se detalló la guía para evaluar un esquema. Se describieron los principales artefactos que según la metodología de desarrollo, se generan en el proceso de la planificación del proyecto, entre los que se encuentran la lista de reserva del producto, la descripción de las historias de usuarios, el diagrama de clases del diseño y el modelo de datos 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 mostraran además el diagrama de componente y de despliegue para una mayor comprensión del sistema. Además se realizará una breve descripción de las pruebas realizadas al sistema.

3.1 Diagrama de Componente del sistema.

El diagrama de componentes modela el empaquetado físico del sistema en unidades reutilizables llamadas componentes. Estos se relacionan entre ellos y pueden poseer dependencias de agentes externos a dichos componentes. Un componente es un bloque físico de implementación que encapsula una o más clases. Ejemplos de componentes son tablas, archivos de datos, ejecutables, bibliotecas de vínculos dinámicos, documentos. (29)

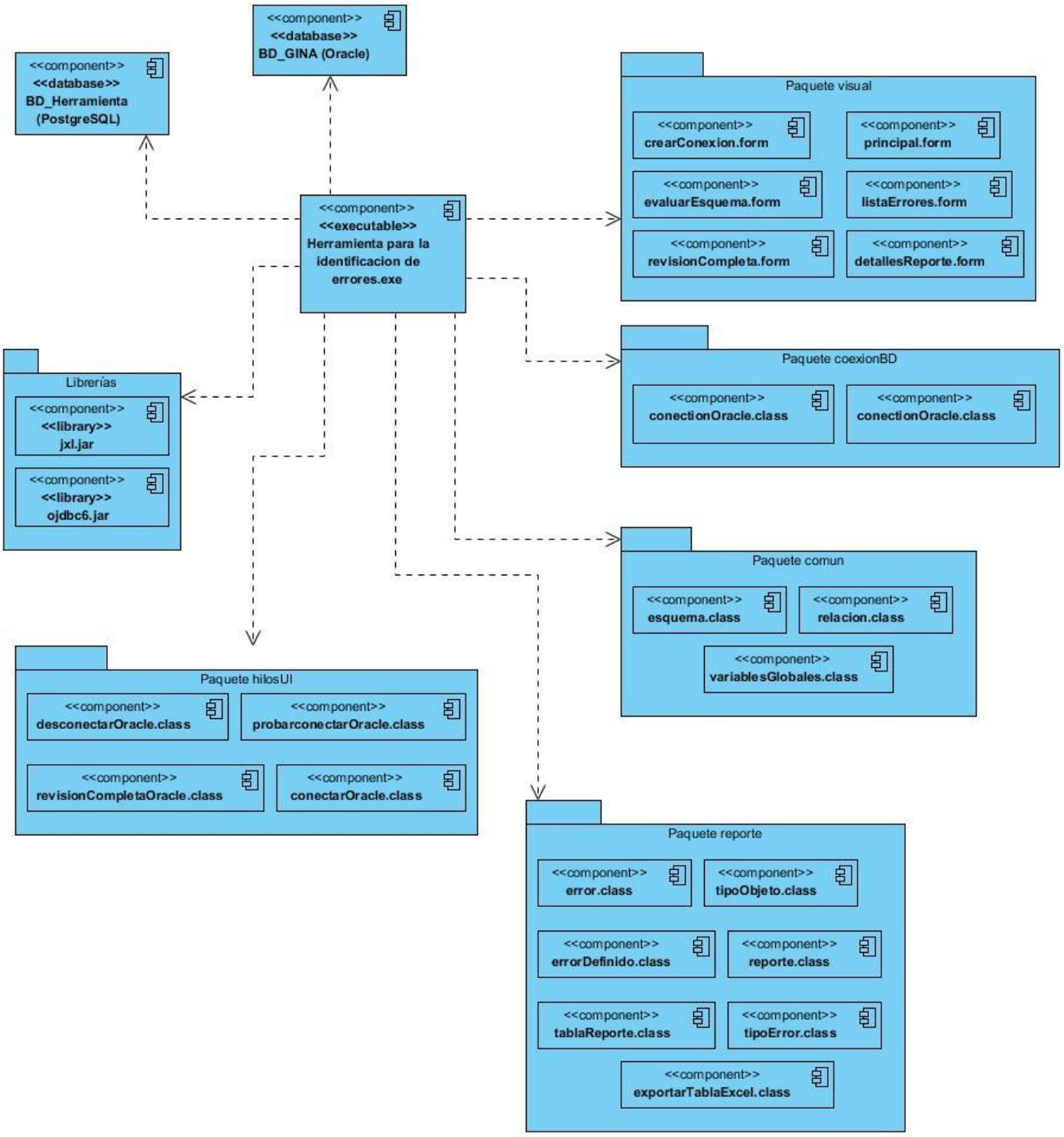


Figura 4: Diagrama de componentes del sistema.

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. (29)

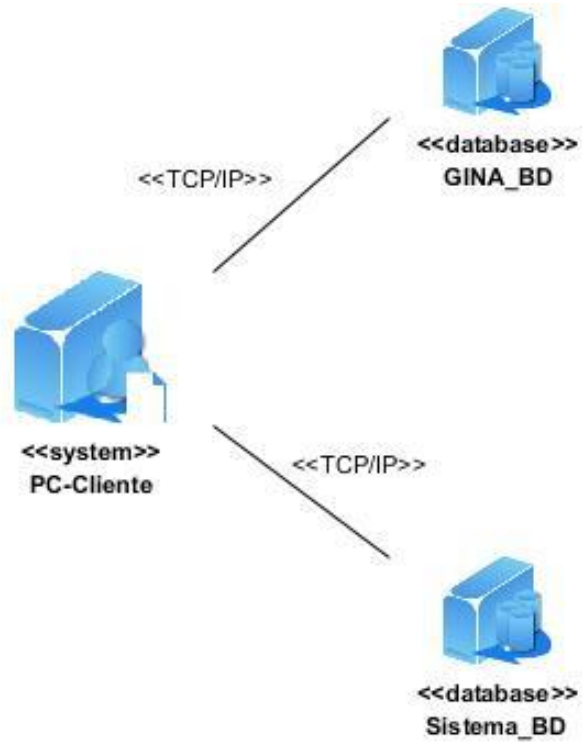


Figura 5: Diagrama de despliegue del sistema.

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 las PC-Clientes.

GINA: Es la base de datos a la que se va a conectar la herramienta para realizar las revisiones necesarias. En este caso se llama **GINA** que no es más que la base de datos del **si**stema de **G**estión **I**ntegral **A**duanera. 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.

3.2.2 Descripción del protocolo utilizado.

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

3.3 Pruebas.

El único instrumento adecuado para determinar el estado de la calidad de un producto software es el proceso de pruebas. 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. (30)

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.

3.3.1 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. El mejor instrumento para esta determinación es 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. (31)

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

3.3.1.1 Prueba asociada a la historia de usuario crear conexión a la base de 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.

Caso de prueba de aceptación	
Número de caso de prueba : 01	**Número de Historia de Usuario:** 01
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.	

Tabla 13: Prueba: Crear conexión con campos incorrectos.

3.3.1.2 Prueba asociada a la historia de usuario modificar conexión a la base de datos.

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

Caso de prueba de aceptación	
Número de caso de prueba : 02	**Número de Historia de Usuario:** 02
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.	

Tabla 14: Prueba: Modificar conexión a la base de datos con datos correctos.

3.3.1.3 Prueba asociada a la historia de usuario eliminar conexión a la base de datos.

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

Caso de prueba de aceptación	
Número de caso de prueba : 03	**Número de Historia de Usuario:** 04
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.	

Tabla 15: Prueba: Eliminar conexión de la base de datos.

3.3.1.4 Prueba asociada a la historia de usuario revisar errores en las restricciones.

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

Caso de prueba de aceptación	
Número de caso de prueba : 04	**Número de Historia de Usuario:** 07
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.

Tabla 16: Prueba: Revisar restricciones de un esquema sin seleccionar el mismo.

3.3.2 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.
- No deberá tener más de 500 líneas de código. (32)

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. (33)

A continuación se muestran dos casos de pruebas unitarias a las clases conetionOracle y conectionPostgres realizadas con el framework Junit.

Caso de prueba 1: Nombre clase: conectionOracle.

Tipo de prueba: Prueba de unidad.

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.

Tabla: 17: Métodos a evaluar de la clase conectionOracle.

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 conexión.	Lanza SQLException.	OK	OK
establecerConexion	timeout	Datos correctos de una conexión.	Establece conexión en menos de 50 segundos.	OK	OK
contarTablasEsquema	assertNotNull	Objeto null	Devuelve una lista vacía.	OK	OK
contarIndiceEsquema	assertEquals	Esquema 'TC'	78	OK	OK

Ejecución de la prueba con Junit:

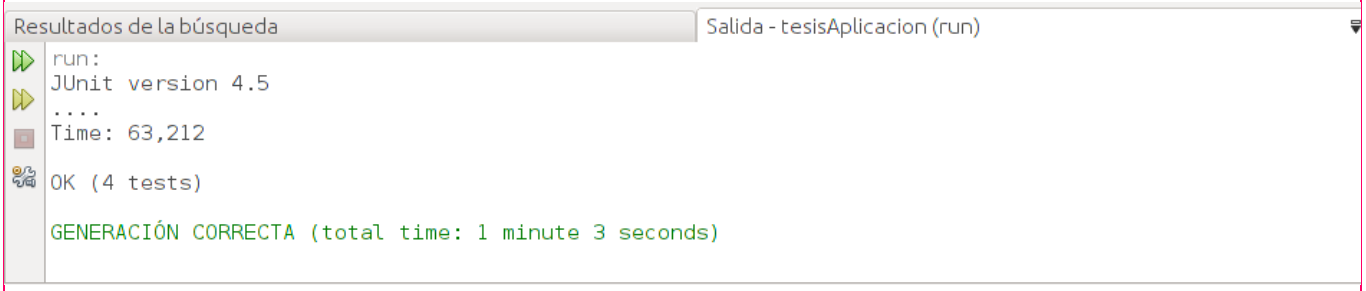


Tabla: 18: Pruebas con Junit a los métodos de la clase conectionOracle.

3.4 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 en el departamento.

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 Departamento de Soluciones para la Aduana en cuanto al diseño de esquemas de base de datos 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 el Departamento de Soluciones para la 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.
- El sistema obtenido es robusto, resistente a cambios y cumple con los requisitos planteados.
- SXP es una metodología flexible que se ajustó satisfactoriamente al desarrollo del sistema.
- Se obtuvo un sistema que facilitará la inclusión de nuevas funcionalidades.

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 los errores de diseño de una base de datos en el menor tiempo posible, se necesita de un proceso investigativo más amplio, y este trabajo es solo la primera parte del mismo. Teniendo en cuenta lo antes planteado se pueden hacer las siguientes recomendaciones:

- Estudio para agregar a la aplicación las funcionalidades de corregir automáticamente los errores en los casos posibles.
- Estudio para agregar a la herramienta una alternativa manual en la que se le permita al usuario escoger la mejor solución para corregir un error en caso de que exista mas de una solución vía para la solución del mismo.
- Estudio para agregar a la herramienta la verificación de las 3 primeras Formas Normales de ser posible.

Referencias bibliográficas.

1. ISACA. Trust in, and value from information systems. [En línea] ISACA, 2012. [Citado el: 20 de nobiembre de 2011.] http://www.isaca.org/Knowledge-Center/cobit/Pages/Downloads.aspx.

2. REAL ACADEMIA ESPAÑOLA. [En línea] 2012. [Citado el: 20 de noviembre de 2011.] http://www.rae.es/rae/gestores/gespub000039.nsf/voTodosporId/6994DD89C871EF0AC12579C700269C47?OpenDocument&i=8.

3. **Riordan, Rebecca M.** *Diseño de base de datos relacionales.* España : Compuesto en Evolution, Artes Grificas, S. L., 2000. ISBN: 84-481-2770-6.

4. **Arias, A. A. Rol Diseñador de la Base de Datos.** Sistema Integrado de Gestión Estadística (SIGE). Ciudad de La Habana. Cuba : s.n., 2007. Universidad de las Ciencias Informáticas.

5. **Ramakrishnan, Raghu y Gehrke, Johannes.** *Database Management Systems.* California : Osborne/McGraw-Hill Berkeley, 2000. ISBN:0072440422.

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

7. FFIEC. *IT Examination HandBook InfoBase.* [En línea] 2011. http://ithandbook.ffiec.gov/it-booklets/development-and-acquisition/development-procedures/software-development-techniques/computer-aided-software-engineering.aspx.

8. EcuRed, Conocimiento con todos y para todos. [En línea] 2012. www.embarcadero.com/products/er-studio.

9. fabFORCE.net, Fabulous Force Database Tools. [En línea] 2003. http://www.fabforce.net/dbdesigner4/.

10. **Infomación, Sistemas de.** Web asiganturas de la Universidad de La Habna. [En línea] 3 de enero de 2012. http://www.fec.uh.cu/websasignaturas/GI/Clases/Sistemas%20de%20Informacion/easy%20case.pdf.

11. Oracle. *Oracle Corporation.* [En línea] 2012. http://www.oracle.com/technetwork/developer-tools/designer/overview/index-082236.html.

12. **Telelogic, an IBM Company.** Telelogic, an IBM Company. [En línea] 2012.

http://www.opengroup.org/togaf/cert/showcase/participants/telelogic/index.htm.

13. **BERTINO, E. A. y MARTINO,.** *Sistemas de bases de datos orientadas a objetos. .* Los Ángeles, California : s.l. : Ediciones Díaz de Santos, 1995.

14. PostgreSQL. [En línea] 1996-2012. http://www.postgresql.org/about/.

15. **Cobo, Ángel, y otros, y otros.** *PHP y MySQL: Tecnología para el desarrollo de aplicaciones web.* España : Ediciones Díaz de Santos, 2005. ISBN: 84-7978-706-6.

16. Toad World. [En línea] 2012. http://www.toadsoft.com/.

17. **ORACLE.** Conozca mas sobre java. [En línea] 2012. http://www.java.com/es/about/.

18. desarrolloWeb.com. [En línea] 26 de enero de 2005. http://netbeans.org/community/releases/69.

19. **affiliates, Oracle Corporation and/or its.** NetBeans. [En línea] 2012. http://www.netbeans.org/community/releases/69/.

20. PostgreSQL. [En línea] 1996-2012. http://www.postgresql.org/about/.

21. PostgreSQL. [En línea] 2012. http://www.postgresql.org/ftp/pgadmin3/.

22. fabFORCE.net, Fabulous Force Database Tools. [En línea] 2003. http://www.fabforce.net/dbdesigner4/.

23. *Metodologías tradicionales vs metodologías ágiles.* **Figeroa, Roberth G. , Solis, Camilo J. y Cabrera, Armando A.** Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación : s.n., 2012.

24. **Romero, Ing. Gladys Marsi Peñalver.** *Release_0.2_SXP.* La Habana, Cuba : s.n., 2012.

25. **Sommerville, Ian.** *Ingenieria del Software.* s.l. : Prentice Hall, enero 2005, 7ma. Edición. ISBN: 8478290745.

26. **ROMERO, GLADYS MARSI PEÑALVER.** *Release_0.2_SXP.* La Habana, Cuba : s.n., 2012.

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

28. **Date, C.J.** *Introducción a los Sistemas de Base de Datos.* s.l. : Prentice Hall.

29. **Laman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México :

PRENTICE HALL, 1999. ISBN: 970-17-0261-1.

30. pruebasdesoftware. [En línea] 2005. http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

31. prubasdesoftware. [En línea] 2005. http://pruebasdesoftware.com/pruebadeaceptacion.htm.

32. **Juristo, Natalia, Moreno, Ana M. y Vegas, Sira.** *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* 2005.

33. **Usaola, Macario Polo.** Pruebas de programas Java mediante junit. [En línea] Escuela Superior de Informática Paseo de la Universidad, 2010. http://www.inf-cr.uclm.es/www/mpolo/tutorial/proposito.html.