Universidad de las Ciencias Informáticas

"Facultad 6"

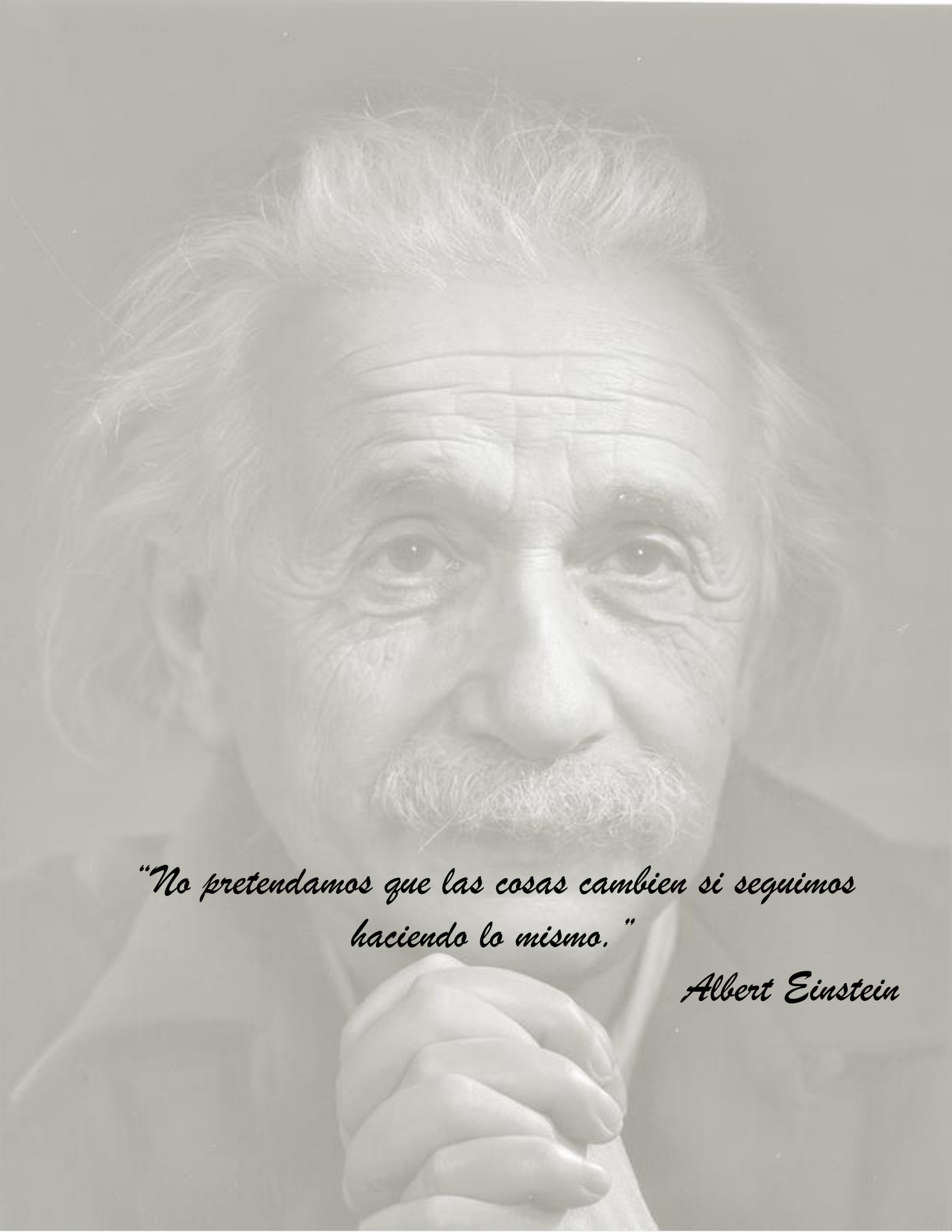
Título: "Herramienta para el enmascarado de datos en bases de datos PostgreSQL, MySQL y SQL Server"

Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas.

Autores: Josué Mojena Marín
Oswaldo Victor Melo Pérez

Tutor: Ing. Mikel Díaz Hernández
Ing. Flavio Enrique Roche Rodríguez

"18 junio de 2014".
"Año 56 de la Revolución"



"No pretendamos que las cosas cambien si seguimos haciendo lo mismo."

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis que tiene por título: **Herramienta para el enmascarado de datos en bases de datos PostgreSQL, MySQL y SQL Server** y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Josué Mojena Marín

Firma del Autor

_____

Oswaldo Victor Melo Pérez

Firma del Autor

_____

Ing. Mikel Díaz Hernández

Firma del Tutor

_____

Ing. Flavio E. Roche Rodríguez

Firma del Tutor

DATOS DE CONTACTO

Tutor:

Ing. Mikel Díaz Hernández

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: mdiazhdez@uci.cu

Tutor:

Ing. Flavio E. Roche Rodríguez

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: feroche@uci.cu

AGRADECIMIENTOS

Es inevitable pensar que se ha hecho realidad uno de mis más importantes sueños, sin tener en cuenta cada una de las personas que se vieron involucradas en él, con su apoyo, su presencia o con solo su preocupación de cómo iba el proceso. Es por ello que quiero agradecer:

A mi mamá por haberme regalado el placer que es la vida, por haber guiado mis primeros pasos y luego darme la confianza plena para seguir adelante solo. A ella que tuvo plena confianza en mis decisiones y compromiso. Por ser la amiga, la compañera y la mujer más importante en mi vida. Por su amor, cariño, dedicación y sentimiento de madre que la hace distinta de todas.

A mi papá por ser mi ejemplo a seguir, por su constante exigencia y comprensión. Por ser un hombre sabio a la hora de dar un consejo o un regaño. Por su preocupación por mi progreso como estudiante, por su educación y estimulación para que me convirtiera en un buen profesional.

A mi hermana Yudicel por haberme acogido en sus brazos como su hijo hace mucho tiempo, por su gran apoyo moral, espiritual y material. Por ser la amiga que ha estado dispuesta a apoyarme incondicionalmente. Por su amor, paciencia y ese gran corazón que la convierte en la mejor persona del mundo.

A mi hermana Yuleymis por haberme legado el reto de sellar una familia de ingenieros. Por sus sabias críticas y enseñanzas, que me han llevado a ser una mejor persona, por ese sentimiento grande y autoprotector que tiene para con los suyos. Por su amor, su apoyo, sus aportes y exigencias en mi vida universitaria.

A una persona que se ha colado en mi vida llenándome de felicidad con su amor, cariño y dedicación. A ella que se ha dispuesto a protagonizar junto a mí el último capítulo del libro mi vida. A ti Mariannys que te llegue mi amor y mayor agradecimiento.

A mis cuñados Jesús y Yaser, que han sido partícipes de cada momento vivido y sufrido en estos cinco años dispuestos a dar lo mejor de si cuando ha sido necesario. Agradecer a toda mi familia en general.

A mi hermano de la vida Orioldis que ha estado presente en los momentos duros, manteniendo su fidelidad en todas circunstancias. Por su apoyo, dedicación y además ser el mejor compañero de fiestas y parrandas. A mi amigo Héctor que ha pesar que la vida de estudiantes nos separó, siempre ha mantenido su amista inigualable, compartiendo muy buenos momentos juntos.

A mi compañero de tesis que ha pesar de sus rabietas se ha mantenido firme en las noches de trabajo y de fiestas.

A los tutores que se han se han estresado tanto como nosotros, dando lo mejor de si y aportando su granito de arena. A nuestro oponente por sus críticas constructivas que nos ayudaron a perfeccionar este trabajo.

Al piquete de los aventureros, encabezados por la loca Laura, la flaca de Anet, el bachatero de Eduard, y el que no se pierde nada Yadián, además a mis compañeros Darián, Juanmi, Pedro, Erichel, a mis amigos Lester y Damián, que llegaron para quedarse.

A Jany y Daniela, que escribieron un pedacito muy importante de mi historia. A todos mis compañeros de aula.

En fin a todos los que de una manera u otra, ayudaron a aliviar el gran peso de ser un graduado universitario, a todos gracias.

Josué Mojena Marín.

Agradezco con todo el corazón a esa persona que me dio la vida, aquella que siempre ha cuidado y preocupado por mi bienestar. A esa que aún me trata como su pequeño y para la cual todos los días tengo un beso. A ella la que nunca ha dejado de darme fuerza y razones de triunfar en la vida. Eres especial, te amo mamá.

A mi padre quien ha sido mi intensivo y un gran amigo. A el que me ha mostrado como ser un mejor profesional y ser un triunfador en la vida. Te quiero papa.

A mi hermosa y cariñosa hermana, mi eterna gran amiga. Por ser alguien con quien siempre he podido contar y a pesar de nuestras disputas de hermanos siempre nos querremos. Por siempre juntos sister.

A mis tías Natacha y Humbe, por ayudarme y apoyarme en todo lo que he necesitado. Las quiero.

A mis primos Harold y Carlitos, a ellos que con su amistad han logrado alegrar mis días. Ustedes son mis hermanos, los quiero.

A los grandes compañeros y amigos Eduard, el Beny, Javier, el Carli, Pedrito y el mas cómicos de todos, el coge lucha del Laffi, con quienes compartí muy buenos momentos estos cinco años.

A Mario y Alfredo, grandes compañeros del futbol que nunca olvidaré los buenos momentos que pasamos juntos, al fin campeones! En todo.

A mi dúo de tesis Josué que contribuyó y se esforzó para la realización de este trabajo. Por ser un gran compañero de fiestas.

A todos los de mi grupo por soportarme y ser una gran familia para mí. A ustedes que han sido grandiosos compañeros.

A los colegas del rugby, espero que tengamos mejores resultados.

A mis tutores por ser tan comprensivos y ayudar en todo lo que estuvo a su alcance en el desarrollo de este trabajo.

A Glennis por ayudarme e impulsar a ser mejor cada día durante el tiempo que pudo, hoy soy mejor y veo la vida de manera diferente, a ti muchas gracias.

Al tribunal y el oponente por las recomendaciones y críticas tan constructivas, todo para lograr un mejor trabajo de diploma.

A todos los compañeros que he tenido durante estos cinco anos, gracias nunca los olvidaré.

Oswaldo Victor Melo Pérez.

DEDICATORIA

Dedico este trabajo a mi abuelo Argeo Mojena Montero, por haber sido un hombre de luz larga, creador de una maravillosa familia, donde abunda la unidad y el amor. A él que representa la raíz y columna vertebral de todos nosotros.

Josué Mojena Marín.

A mis padres, mi hermana y mi pequeña sobrina Karla.

A mis padres por ser figuras muy representativa en mi vida, a ellos que han sido mis grandes propulsores en todos los entornos que he enfrentado hasta hoy. A mi fabulosa hermana que nunca ha dejado de creer en mí. Al último y más querido miembro de mi familia, mi pequeña Pití (Karla) que me ha alegrado cada día desde su llegada al mundo. Los quiero.

Oswaldo Victor Melo Pérez.

RESUMEN

En el Centro de Tecnologías de Gestión de Datos (DATEC) existe la necesidad de mantener la integridad y confidencialidad de la información sensible contenida en las bases de datos (BD) PostgreSQL, MySQL y SQL Server; a partir de la cual surge el presente trabajo para desarrollar una herramienta que permita el enmascarado de dichas bases de datos. Con esta herramienta se evitaría la pérdida o sustracción inapropiada de información sin que los datos pierdan sus propiedades originales.

Para darle solución al problema se realizó una investigación de las herramientas ya existentes que realizan el enmascarado de datos (técnica comúnmente utilizada para este tipo de acciones) con prestigio internacional y Mask-PG (herramienta enmascaradora de datos para PostgreSQL desarrollada en el centro DATEC). Además se caracterizaron las diferentes arquitecturas y técnicas más utilizadas para este tipo de procedimiento, con el objetivo de definir cuáles utilizar en el sistema a desarrollar.

La aplicación desarrollada,que lleva por nombre Brookesia, permite el enmascarado de datos en las BD PostgreSQL, MySQL y SQL Server;la misma cuenta con las técnicas baraja, anulación, variación de número y fechas, sustitución y enmascarado de datos de salida. Con este sistema se garantiza la integridad y confidencialidad de los datos sensibles almacenados en BD de PostgreSQL, MySQL y SQL Server.

Palabras claves: base de datos, enmascarado de datos, MySQL, PostgreSQL, SQL Server.

ABSTRACT

At the Center for Data Management Technologies (DATEC) there is a need to maintain the integrity and confidentiality of sensitive data in databases (DB) PostgreSQL, MySQL and SQL Server. The present work arises to develop a tool that allows the masking of such databases to avoid the loss or theft of information inappropriately, without data losing their original properties.

For solving the problem an investigation of existing tools that perform the masked data (commonly used technique for this type of action) with international prestige and Mask-PG (data masking tool for PostgreSQL developed in DATEC center) was performed. Besides the different architectures and techniques used for this type of procedure, in order to define which ones to use in the system to be developed were characterized.

The developed application, which is called Brookesia, allows the masked data in the DB PostgreSQL, MySQL and SQL Server; it has the technical shuffles, cancellation, variation in number and date substitution and masked output data. With this system the integrity and confidentiality of sensitive data stored on DB PostgreSQL, MySQL and SQL Server is guaranteed.

Keywords: database, data masking, MySQL, PostgreSQL, SQL Server.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El desarrollo de las Tecnologías de la Información y la Comunicación (TIC) en el mundo, ha implicado que el manejo de la información sea de manera digital, lo anterior ha permitido facilitar el crecimiento, divulgación y tratamiento de grandes volúmenes de datos, propiciando la utilización de los medios y soportes informáticos para su almacenamiento. Como consecuencia de esta evolución de la información ha aumentado el uso de los medios informáticos en todas las esferas de la sociedad y con ello la falta de seguridad, ya que su manipulación se realiza mediante el uso de las bases de datos (BD), donde estas representan el principal objetivo de ataque para el robo y la sustracción de datos.

En la actualidad las instituciones son conscientes de la importancia de la seguridad y protección de los datos e informaciones que almacenan. Por ello generalmente tienen un estricto control mediante procedimientos y estándares de seguridad, implementados para la protección de los datos en sus entornos productivos. Sin embargo, estos sistemas de seguridad no suelen ser efectivos cuando son aplicados en ambientes no productivos, en los cuales los desarrolladores y probadores tienen acceso a los datos reales. Es habitual que en los entornos de desarrollo se utilicen datos reales para las pruebas que se le realizan a las aplicaciones con intenciones de conocer su rendimiento y funcionalidad; de esta manera las compañías se ven expuestas ya que si los datos sensibles no son protegidos, corren el riesgo de que la información crítica, con alto valor en otras compañías, no sea utilizada con fines positivos, provocando una afectación directa a la empresa.

Existen diversas técnicas para la protección de los datos, dos de las más utilizadas mundialmente son la encriptación y el enmascarado. La encriptación es de vital importancia en el intercambio de información y en la exposición directa de BD con información sensible, pero esta técnica se vuelve muy complicada de utilizar en ambientes de desarrollo y pruebas. Cuando la encriptación es utilizada sobre BD que se utilizan en dichos entornos, los desarrolladores y probadores interactúan con estos datos sensibles en sus formas naturales, teniendo acceso a información que pudieran ser copiadas o robadas, pudiendo provocar un efecto negativo sobre la institución. De manera que, el enmascarado de datos se vuelve más práctico en estos ambientes, donde la información contenida en las BD deja de ser sensible, pero sin dejar de tener características reales, o sea, se modifican los datos de manera que dejan de tener un valor como información, pero siguen teniendo sentido.

Cuba se encuentra inmersa en un desarrollo tecnológico, donde el principal objetivo es informatizar todas

las esferas de la sociedad y lograr la soberanía tecnológica. Por consiguiente, los Sistemas Gestores de Bases de Datos (SGBD) que se utilizan generalmente son los que están bajo la licencia de código abierto, como son los gestores PostgreSQL y MySQL, este último está en un esquema de licenciamiento dual; ya pesar de la gran migración que existe en el país hacia el software libre, aún quedan instituciones que tienen sus BD alojadas en gestores como SQL Server.

La Universidad de las Ciencias Informáticas (UCI) tiene como compromiso y objetivo fundamental contribuir a la informatización de la sociedad, propiciando que el país avance hacia una soberanía tecnológica. Por lo que el Centro de Tecnologías de Gestión de Datos (DATEC), tiene como fundamento de trabajo potenciar las tecnologías de BD enfocado a dicho objetivo.

En el departamento PostgreSQL del centro DATEC, se ha dificultado el desarrollo de proyectos por no contar con una herramienta que realice la modificación de BD con información sensible para PostgreSQL, MySQL y SQL Server, garantizando su integridad y confidencialidad. Con intenciones de solucionar lo anterior, en el curso 2011-2012 fue desarrollada la herramienta Mask-PG con la limitante de ser solo funcional para bases de datos PostgreSQL. Es por esto, que el presente trabajo tiene como **problema de investigación:** ¿Cómo garantizar con el enmascarado la integridad y confidencialidad de los datos sensibles almacenados en bases de datos PostgreSQL, MySQL y SQL Server?

Para dar solución al problema planteado se define como **objeto de estudio:** El proceso de enmascarado de datos, enmarcado en el **campo de acción:** Enmascarado de datos en bases de datos PostgreSQL, MySQL y SQL Server.

Para dar solución al problema de investigación se traza como **objetivo general:** Desarrollar una herramienta que garantice con el enmascarado la integridad y confidencialidad de los datos en bases de datos PostgreSQL, MySQL y SQL Server.

A partir del objetivo general se derivan los siguientes objetivos específicos:

- ✓ Caracterizar las herramientas y estrategias de enmascarado de datos existentes.
- ✓ Implementar la herramienta de enmascarado de datos para bases de datos PostgreSQL, MySQL y SQL Server.
- ✓ Validar la herramienta para realizar el enmascarado de datos a bases de datos PostgreSQL, MySQL y SQL Server.

Para cumplir con los objetivos planteados se trazaron las siguientes **tareas de la investigación**:

- ✓ Caracterización de las herramientas existentes que realizan el enmascarado de datos.
- ✓ Caracterización de los tipos de enmascarado de datos más utilizados.
- ✓ Identificación de las funcionalidades de la herramienta para el enmascarado de datos.
- ✓ Modelación y diseño de la herramienta para el enmascarado de datos.
- ✓ Implementación de la herramienta para el enmascarado de datos.
- ✓ Definición del tipo de prueba para la validación dela herramienta de enmascarado de datos.
- ✓ Realización de las pruebas definidas para validar la herramienta de enmascarado de datos.

Para el desarrollo de la investigación se utilizarán los siguientes **métodos científicos de la investigación:**

- ✓ Método Histórico – Lógico: Este método presupone los antecedentes y trayectoria que tiene el tema en investigación. Fue necesario para estudiar la necesidad que implicó el surgimiento del enmascarado de bases de datos, e identificar y caracterizar las herramientas de enmascarado existentes en el mundo.
- ✓ Método Análisis – Sintético: Este método permite la división o descomposición del tema, para garantizar un mejor entendimiento y comprensión del estudio a realizar, después de haber realizado este análisis, la síntesis establece la unión de las partes previamente analizadas. Este es el método a seguir para el análisis de las estrategias y elementos fundamentales del enmascarado de datos, de manera que se pueda adquirir un conocimiento del tema de una forma comprensible.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

En este capítulo se fundamenta la investigación definiendo un conjunto de conceptos y elementos asociados con el enmascarado de datos. Se caracterizan algunas de las herramientas de enmascarado de datos más importantes utilizadas en el mundo, así como las estrategias o algoritmos que utilizan. Se definen las herramientas y tecnologías necesarias para el desarrollo de la aplicación, así como la metodología que va a guiar el proceso de desarrollo.

CAPÍTULO 2: DESCRIPCIÓN DEL DESARROLLO DE LA HERRAMIENTA BROOKESIA.

En este capítulo se aborda la descripción del desarrollo de la herramienta de enmascarado de datos, caracterizando los patrones que se utilizarán para modelar el diseño de la implementación, además de ejemplificar y describir los artefactos generados por la metodología de desarrollo.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DE LA HERRAMIENTA BROOKESIA.

En este capítulo se describen los principales artefactos generados en las fases de implementación y prueba, detallando los estándares de codificación e interfaces de la solución. Además se especifican las pruebas a las que será sometida con el fin de garantizar un funcionamiento eficiente de la misma.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se hace referencia a los elementos fundamentales de la presente investigación, abarcando conceptos como: bases de datos, sistemas gestores de bases de datos y enmascarado de datos. Se enuncian y explican algunas técnicas de enmascarados que existen, así como las herramientas que se utilizan en el mundo para realizar este proceso. Se describen la metodología y herramientas a utilizar para el desarrollo de la aplicación, para lograr una organización y planificación de dicha solución, enfocado a obtener un resultado con calidad.

1.1 Bases de Datos

Las bases de datos surgen a partir de la necesidad de almacenar grandes volúmenes de información, de tal manera que un sistema informático pueda manipularla. Su utilización ha propiciado consistencia, integridad, seguridad, flexibilidad y rapidez al obtener datos. Algunas definiciones de bases de datos son:

Una base de datos se define como un conjunto exhaustivo de datos estructurados, fiables y homogéneos, organizados independientemente de su utilización e implementación en una computadora, accesibles en tiempo real, que pueden compartir varios usuarios con necesidades de información diferentes y no predecibles en el tiempo (1).

También se puede definir como un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada (2).

En conclusión,una base de datos es un almacén de datos organizados y relacionados entre sí de manera estructurada, a los que los usuarios pueden acceder para ingresar, actualizar o visualizar, según los permisos que les hayan asignado.

1.2 Sistemas Gestores de Bases de Datos

Para un mejor manejo y administración de las bases de datos surgen herramientas especializadas para estos fines. Estas herramientas, que hoy en día se han vuelto muy populares y necesarias son los SGBD.

Entre las herramientas de software desarrolladas para crear y gestionar una base de datos, sin duda, la más importante es el SGBD. Este programa es el encargado de poner a disposición de los distintos usuarios las técnicas de bases de datos: descripción centralizada de los datos, posibilidad de definir vistas de dichos datos, etc. Estos sistemas tienen como objetivo fundamental mantener independencia, integridad y seguridad de los datos, o sea, es el encargado de mantener las virtudes que se proponen en la definición de bases de datos(3).

Otra definición para un SGBD es *"software o conjunto de programas que permite crear y mantener una base de datos. El SGBD actúa como interfaz entre los programas de aplicación y el sistema operativo. El objetivo principal es proporcionar un entorno eficiente a la hora de almacenar y recuperar la información de las base de datos. Este software facilita el proceso de definir, construir y manipular bases de datos para diversas aplicaciones"*(4).

Los SGBD que se tratan en esta investigación son PostgreSQL, MySQL y SQL Server,a continuación se hace una pequeña caracterización de cada uno de ellos.

1.2.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto – relacional, distribuido bajo licencia Berkeley Software Distribution (BSD) y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (5).

En la actualidad el proyecto PostgreSQL ha ganado en experiencia gracias a la característica de ser código abierto, permitiendo esto que existan instituciones y programadores independientes interesados en colaborar y apoyar el desarrollo del mismo, mediante extensiones y contribuciones que le han permitido ganarse el protagonismo como SGBD.

En el mundo actual PostgreSQL es utilizado por grandes compañías de negocio, y otras cuantas más que están en proceso de migración. Incluso, existen empresas como McAfee**,** Sony Onliney Deutsche Bank que utilizan bases de datos comerciales y han migrado, debido a la gran cantidad de ventajas que le proporciona el software libre. Este SGBD en particular, proporciona grandes ventajas como son: un mejor

soporte, ahorros considerables en costos de operación, estabilidad, confiabilidad, extensible, multiplataforma, está diseñado para ambientes de grandes volúmenes de datos y cuenta con potentes herramientas gráficas de diseño para administrar sus bases de datos como son pgAdmin y pgAccess.

1.2.2 MySQL

MySQL es la base de datos número uno para Web y es una excelente base de datos embebida. Más de 3.000 Proveedores de Software Independientes (ISVs por sus siglas en inglés) y Fabricantes de Equipos Originales(OEMs por sus siglas en inglés), incluyendo 8 de los 10 mayores y 17 de los 20 principales proveedores de software, de todo el mundo, confían en MySQL como base de datos de sus productos (6).El sistema de base de datos operacional MySQL es hoy en día uno de los más importantes en lo referente al diseño y programación de bases de datos de tipo relacional. Cuenta con millones de aplicaciones y aparece en el mundo informático como una de las más utilizadas. El programa MySQL se usa como servidor, a través del cual pueden conectarse múltiples usuarios y utilizarlo al mismo tiempo(7).

MySQL se encuentra en proceso evolutivo, donde es muy utilizado por pequeñas y grandes empresas, estas últimas generalmente la usan en ámbitos departamentales, como prueba de su eficiencia. Posee un gran soporte y es de código abierto.

1.2.3 SQL Server

Microsoft SQL Server es un sistema de administración y análisis de bases de datos relacionales de Microsoft para soluciones de comercio electrónico, línea de negocio y almacenamiento de datos (8). En las últimas versiones de este sistema se añadieron grandes funcionalidades que le dan cierta distinción frente a otros, como son: alta disponibilidad y recuperación de desastres mediante clústeres AlwaysOn, para lograr una gran velocidad en ejecución de consultas está xVelocity en almacenamiento de memoria y una exploración de datos rápida. Este SGBD tiene como desventaja fundamental que no es de código abierto, por lo que su soporte solo depende de Microsoft, lo que hace que pierda protagonismo frente a otros de los sistemas. Además de que es específico para sistemas Windows, o sea, no es multiplataforma, y con la revolución de software libre queda limitado.

1.3 Enmascarado de Datos

Las instituciones adoptan medidas extremas para la protección de la información sensible en ambientes de producción. Sin embargo, de forma frecuente se descuidan los no productivos como los de desarrollo, pruebas y capacitación. Estos ambientes no productivos necesitan de datos reales y con frecuencia operan con una copia de los datos sensibles. Teniendo en cuenta esto, dichas áreas son un objetivo alentador para usuarios malintencionados. La prestigiosa empresa de asesoramiento Forrester Research[1]recomienda que "*todas las empresas deberían analizar la tecnología para el enmascaramiento de datos que puede ayudarles a proteger datos privados en ambientes de prueba o cuando se envían datos a un proveedor externo o geográficamente distante*"(9).

Des – identificar o enmascarar es una vía para asegurar que el robo, la exposición o pérdida de datos no pueda ser utilizada por personas no autorizadas. Es una solución que transforma la información sensible en datos que no son verdaderos pero sí verídicos, es decir, que mantienen un aspecto similar al real, conservando las propiedades de los originales. Estas técnicas permiten alterar los datos sin perder sus atributos, de tal modo que garantice la confidencialidad de la información enmascarada(10).

Enmascarado de datos se refiere al proceso de cambiar ciertos elementos de datos dentro de un almacén de datos de manera que la estructura sigue siendo similar, mientras que la información en sí misma se cambia para proteger la información sensible(11).

Es el proceso de des – identificación (enmascaramiento) de elementos específicos dentro de los almacenes de datos, mediante la aplicación de algoritmos de un solo sentido a los datos. El proceso garantiza que los datos confidenciales se sustituyen con datos verídicos, pero no reales, por ejemplo, realizar variación de los dígitos de un número de Seguro Social, preservando el formato de la información. La naturaleza unidireccional del algoritmo significa que no hay necesidad de mantener las claves para restaurar los datos, como se haría con la encriptación(12).

1.3.1 Importancia del enmascarado de datos

El enmascaramiento de datos es un aspecto importante a tener en cuenta en el marco general de la evaluación de riesgos de las empresas. Los datos enmascarados son utilizados por diferentes entidades

[1]Más información consultar:www.forrester.com

privadas y gubernamentales en el desarrollo de aplicaciones, pruebas, control de calidad, apoyo y análisis de negocio. Este proceso de des – identificar asegura que los datos confidenciales sean reemplazados con datos fidedignos pero no reales, logrando que la información sensible no esté disponible fuera del ambiente autorizado. La creación de pruebas y desarrollo de copias en un proceso automatizado reduce la exposición de datos sensibles. El enmascaramiento de los datos es una estrategia efectiva para reducir el riesgo de exposición de los datos desde dentro y fuera de una organización, debe ser considerada una práctica recomendada para evitar la fuga de datos confidenciales en las bases de datos de los ambientes no productivos; proporciona una mejor vía para la realización de operaciones de información dependientes, sin poner estos en riesgo de exposición(13).

El uso de las tecnologías de enmascarado de datos trae consigo una serie de beneficios para las instituciones que hacen uso de ellas. Entre las principales ventajas que posibilitan estas herramientas se pueden destacar:

- Incrementan la protección y control del robo de datos.
- Reducen las restricciones sobre los datos.
- Proporcionan datos realistas para los ambientes de pruebas, desarrollo, capacitación, minería de datos o de investigación.
- Reducen el riesgo de violación de datos y de cumplimiento mediante la protección de datos sensibles y privados.
- Reducen costos de desarrollo gracias a que requiere un menor esfuerzo para la protección de información sensible.
- Aceleran el tiempo de comercialización al aumentar la calidad del desarrollo y de las pruebas.
- Permiten el desarrollo de software y el intercambio de datos, en ambientes con baja seguridad.
- Apoyan el cumplimiento de las políticas y legislaciones sobre privacidad de datos.
- Mejoran la confianza del cliente.
- Proporcionan una mayor sensación de seguridad a los clientes, empleados y proveedores(14).

1.3.2 Arquitecturas de enmascarado de datos

Fundamentalmente, existen dos tipos básicos de arquitecturas que se utilizan en el diseño de herramientas para el enmascarado de datos. Estas constan de dos bases de datos, una de origen

(contenedora de la información a enmascarar) y otra de destino que será el resultado final que se obtendrá luego de realizar el proceso de enmascarado de los datos (10). A continuación son descritas dichas arquitecturas.

Sobre la marcha, de servidor a servidor

En esta arquitectura los datos no existen en la base de datos de destino antes del enmascarado de los datos.

Ventajas

- ✓ Los datos nunca están presentes en una forma desenmascarada en la base de datos destino.

Desventajas

- ✓ Cualquier error en el proceso interrumpiría la transferencia de los datos.
- ✓ La capacidad de enmascarar los datos después de que finaliza la transferencia puede ser problemática.

En el lugar (In- Situ)

En esta arquitectura, el clon de la base de datos a ser enmascarado se crea por otros medios y el software simplemente opera en la base de datos clonada.

Ventajas

- ✓ Las operaciones de enmascarado son independientes del proceso de clon de la base de datos destino.
- ✓ Es posible aplicar operaciones de enmascaramiento adicionales en cualquier momento.

Desventajas

- ✓ Los datos se presentan en su estado original en la base de datos destino y será necesario adoptar medidas de seguridad adicionales durante ese tiempo.

Se escoge utilizar la arquitectura en el lugar (In-situ) debido a que el uso de ella proporciona independencia entre los procesos de clonaje de la base de datos y el enmascarado de la información, favoreciendo la aplicación de otras técnicas de enmascarado en cualquier momento deseado.

1.3.3 Técnicas de Enmascarado de Datos

Existen diversas técnicas utilizadas para el enmascaramiento de datos, las más comunes que se utilizan incluyen el cifrado, baraja, el enmascarado de datos de salida, variación de números y fechas, la sustitución, y anulación de salida o eliminación(15). A continuación se describen dichas técnicas:

Sustitución

Esta técnica consiste en reemplazar al azar el contenido de una columna de datos con información que es similar pero no tiene relación alguna con los datos reales(10).

La sustitución es muy eficaz en términos de preservar el aspecto y la sensación existente en los datos. La desventaja es que en una base de datos bastante grande será necesaria una base de conocimientos lo suficientemente voluminosa, pues debe existir una sustitución para cada columna de datos. La Tabla 1 muestra un ejemplo de la utilización del algoritmo sustitución.

Tabla 1Ejemplo del algoritmo sustitución.

Tabla original			Tabla resultante	
Nombre	**Apellidos**		**Nombre**	**Apellidos**
Josué	Mojena Marín		Flavio	Pérez González
Oswaldo	Melo Pérez		María	Martínez Gato

Baraja

La técnica de baraja es similar a la sustitución, salvo que en este método los datos se derivan de la propia columna, es decir, los datos de una columna se trasladan al azar entre las filas hasta que no haya ninguna correlación razonable con el resto de la información en la fila(10). La Tabla 2 muestra un ejemplo de la utilización del algoritmo baraja.

Tabla 2Ejemplo del algoritmo baraja.

Tabla original			Tabla resultante	
Nombre	**Apellidos**		**Nombre**	**Apellidos**
Josué	Mojena Marín		Josué	Melo Pérez
Oswaldo	Melo Pérez		Oswaldo	Mojena Marín

Enmascarado de datos de salida

Consiste en la sustitución de determinados campos con un carácter de máscara (por ejemplo, una X). Este método oculta el contenido de los datos y conserva el mismo formato en las pantallas de interfaz e informes(10). La Tabla 3 muestra un ejemplo de la utilización del algoritmo enmascarado de datos de salida.

Tabla 3Ejemplo del algoritmo enmascarado de datos de salida.

Tabla original		Tabla resultante
Tarjeta crédito		**Tarjeta crédito**
4346 6454 0020 5379		4346 XXXX XXXX 5379

Variación de números y fechas

Esta técnica es útil en datos numéricos o de fecha, implica la modificación de cada número o fecha en una columna por algunos al azar dependiendo de un porcentaje de variación de su valor real(10).

Esta técnica tiene la ventaja de proporcionar un disfraz razonable mientras que los datos siguen manteniendo el rango y la distribución de los valores en la columna dentro de los límites existentes. La Tabla 4 muestra un ejemplo de la utilización del algoritmo variación de números y fechas.

Tabla 4Ejemplo del algoritmo variación de números y fecha.

Tabla original			Tabla resultante	
Edad	**Fecha inscripción**		**Edad**	**Fecha inscripción**
23	12/04/1990		38	25/11/1999
40	06/09/1974		26	16/04/2000

Cifrado

Esta técnica consiste en cifrar los datos con cierta clave ofreciendo la opción de dejar los datos en su lugar y visibles para las personas que conozcan dicha clave. Esta parece ser una muy buena opción, sin embargo, para las bases de datos de pruebas anónimas, es una de las técnicas menos útiles(10). La Tabla 5 muestra un ejemplo de la utilización del algoritmo cifrado.

Tabla 5Ejemplo del algoritmo cifrado.

Tabla original			Tabla resultante	
Edad	**Fecha inscripción**		**Edad**	**Fecha inscripción**
23	12/04/1990		Ask56	5d5d4d4as5s
40	06/09/1974		S4sd5	4747s7w4d85

Anulación de salida o eliminación

Consiste en la simple supresión de una columna de datos mediante su sustitución con valores NULL. Es una eficaz manera de asegurar y controlar la visibilidad inapropiada de la información sensible en los entornos de desarrollo y prueba(10). La Tabla 6 muestra un ejemplo de la utilización del algoritmo anulación de salida o eliminación.

Tabla 6Ejemplo del algoritmo anulación de salida o eliminación.

Tabla original			Tabla resultante	
Edad	**Fecha inscripción**		**Edad**	**Fecha inscripción**
23	12/04/1990			
40	06/09/1974			

1.4 Herramientas de Enmascarado Existentes

A nivel mundial el cumplimiento de requisitos legales y de parámetros de seguridad para información y confidencialidad, ha aumentado el interés entre las empresas por la utilización de las técnicas de enmascarado de datos, favoreciendo al auge del desarrollo de aplicaciones que permitan la puesta en práctica de dicha técnica para los diferentes gestores de bases de datos, tal es el caso de:

1.4.1 Informática Data Masking

Es un software de enmascaramiento dinámico de datos que proporciona funcionalidades en tiempo real para evitar que los usuarios no autorizados tengan acceso a información confidencial. Aplica numerosas acciones de seguridad en tiempo real, como enmascarar, cifrar, ocultar, bloquear y auditar datos, así como emitir alertas de uso no autorizado, lo que aumenta la seguridad del enmascaramiento dinámico de datos. Elige automáticamente la técnica de enmascarado basándose en políticas como el cifrado de nombres o el ocultamiento de la información sobre tarjetas de crédito y salarios. Admite solo aplicaciones, informes y

herramientas de desarrollo que se ejecute en todas las versiones de Oracle y Microsoft SQL Server. Informática Dynamic Data Masking emplea una amplia variedad de técnicas de enmascaramiento de datos, incluyendo el enmascarado de datos de salida, cifrado, función hash, baraja, variación de números y fechas, sustitución, bloqueo y ocultación, que conservan las características originales de los datos y mantienen su integridad.

Entre sus mayores beneficios se encuentran:

- Disminuye drásticamente el riesgo de una filtración de datos durante el enmascaramiento dinámico de datos.
- Se personalizan fácilmente las soluciones de enmascaramiento para una normativa o unos requisitos de negocio diferentes(16).

1.4.2 DataMaskingSuite

Es un software que ayuda a ocultar la identidad de los datos y ofrece un alto nivel de seguridad de la información en muchas fuentes de datos diferentes. Data Masking Suite ofusca los datos críticos después de exportarlos desde sistemas de planificación de recursos empresariales o sistemas internos. La información anónima puede ser utilizada por el equipo interno del proyecto, sin riesgo de pérdida de datos o el uso indebido.

Algunas de sus características

- Enmascara datos de cualquier fuente.
- Enmascarado de todas las bases de datos (importación de fuentes de datos Excel, CSV SharePoint conexión abierta a todas las bases de datos (Microsoft SQL Server, MySQL, Oracle, DB2)).
- La ofuscación de datos de archivos como Excel o CSV.
- Codificación para grandes volúmenes de datos.
- Ofrece más de 20 métodos de enmascaramiento.
- Bibliotecas de datos y conjuntos de sustitución para crear datos de ejemplo de prueba para los nombres de empresas.
- Flexible "Buscar y reemplazar" métodos y enmascaramiento basado en patrones.

- Utiliza los métodos de baraja, reemplazar, sustituir, cifrar o anulación de salida o eliminación para proteger los datos sensibles(17).

1.4.3 Mask-PG

Es una herramienta desarrollada en la UCI que permite el enmascarado de datos en bases de datos de PostgreSQL, evitando que los datos sean expuestos ante usuarios no autorizados, hace posible además, el cumplimiento de las normativas de privacidad de los datos en todas las organizaciones y la reducción de riesgo para evitar la filtración de los datos. La herramienta está desarrollada con una arquitectura In-situ (en el lugar) debido a que el uso de esta proporciona independencia entre los procesos de clonaje de la base de datos y el enmascarado de la información. Su funcionamiento se basa en los métodos de sustitución, baraja y variación de números y fechas, debido a que estas técnicas aseguran que la información sensible sea sustituida con datos reales, manteniendo un aspecto similar al real y conservando las propiedades de los datos originales. Posee como deficiencia el lento proceso de enmascaramiento de datos para bases de datos con un volumen de datos importante.

El principal inconveniente de estas herramientas es que ninguna realiza el proceso de enmascarado de datos para el SGBD PostgreSQL, Microsoft SQL Server, MySQL de forma integrada, además las herramientas Informática Dynamic Data Masking y DataMaskingSuite son software propietarios, es decir que no cumplen con el proceso de migración a software libre en el que está encaminado el país. La solución Mask-PG solo implementa tres técnicas de enmascarado de datos, presentando también deficiencias de rendimiento en el proceso de enmascaramiento; de estas premisas surge la necesidad de elaborar una herramienta que integre dichos gestores y permita realizar esta importante tarea, siendo de software libre y que no presente problemas de rendimiento durante el proceso de enmascarado.

1.4.4 Selección de las técnicas a implementar

Tabla 7 Tabla de comparación de las técnicas utilizadas por las herramientas investigadas.

Técnicas	Informática Dynamic Data Masking	DataMaskingSuite	Mask-PG
Sustitución	x	x	x
Baraja	x	x	x

Cifrado	x	x	
Enmascarado de datos de salida	x	x	
Función hash	x		
Variación de números y fechas	x	x	x
Anulación de salida o eliminación	x	x	
Anagrama		x	
Ajuste proporcional inversa		x	

A nivel mundial cinco de las técnicas más utilizadas en las herramientas de enmascarado de datos son sustitución, baraja, variación de números y fecha, enmascarado de datos de salida y anulación de salida o eliminación. En la tabla anterior se evidencia como estos métodos son utilizados en herramientas de prestigio internacional (Informática Dynamic Data Masking y Data Masking Suite) en el proceso de enmascarado. En el caso de la herramienta Mask-PG solo implementa tres de todas las técnicas existentes; por consecuente se decide que en el proceso de desarrollo de la nueva herramienta de enmascarado de datos serán implementadas estas cinco técnicas, ya que están presentes en casi todas las herramientas y además son algoritmos que aseguran la confidencialidad de los datos sensibles. Estos algoritmos aseguran con su uso en el proceso de enmascarado de datos que la información sensible sea sustituida por datos que mantienen un aspecto igual al real, conservando las propiedades de los datos originales, y en el caso de la técnica de anulación de salida o eliminación que permite al cliente controlar la visibilidad inapropiada de la información sensible en los diferentes entornos donde puedan ser utilizados dichos datos.

1.5 Metodología de Desarrollo

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayuda a la documentación para el desarrollo de productos de software(18).Son las encargadas de orientar cada actividad a realizar para el desarrollo de una aplicación informática, definiendo los involucrados en dichas actividades y el papel que van a desempeñar en cada una de ellas, detallando un conjunto de información del comienzo y resultado de las mismas.

Las metodologías de desarrollo se clasifican en dos tipos, las tradicionales y las ágiles, estas últimas le dan un mayor valor a los involucrados, tanto a los desarrolladores como a la colaboración del cliente, siendo más práctica para aplicaciones donde el desarrollo es incremental con iteraciones muy cortas. Este tipo de metodología tiene gran efectividad en proyectos donde los requisitos son variables, el tiempo de desarrollo es pequeño y se exige una alta calidad en el producto.

Programación Extrema o Extreme Programming conocida comúnmente por sus siglas en inglés XP, es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico(19).

Se define esta metodología ya que el equipo de trabajo es pequeño, donde es necesario reducir el esfuerzo de desarrollo para lograr una rápida implementación de software, no existe un contrato tradicional, existen pocos roles, el cliente es parte del equipo de trabajo y además, se cuenta con una vasta experiencia en la utilización de la misma.

1.6 Herramientas y tecnologías

Las herramientas y tecnologías que se seleccionaron para el desarrollo de la aplicación de enmascarado de datos, tienen el objetivo de facilitar, flexibilizar y organizar dicho proyecto.

1.6.1 Herramientas CASE

Las herramientas CASE (Ingeniería de Software Asistida por Computadora, en inglés Computer Aided Software Engineering) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero(20).

Visual Paradigm for UML 8.0

Herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue (para metodologías de ciclo completo). El software de modelado UML agiliza la construcción de aplicaciones de calidad y a un menor costo. Propicia

un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, además permite dibujar todos los tipos de diagramas de clases, realizar ingeniería inversa, generar documentación y código desde diagramas(21).

Se seleccionó la herramienta CASE Visual Paradigm for UML 8.0 pues soporta el ciclo completo del desarrollo de software, genera código fuente a partir del diseño realizado para el lenguaje de Java y otros. Es una herramienta gratuita y además se encuentra disponible para distribución Linux. Permite la integración con el entorno de desarrollo NetBeans, la capacidad de ingeniería directa e inversa y el uso de un lenguaje estándar que facilita la comunicación entre todo el equipo de desarrollo.

1.6.2 Lenguaje de Programación

Un lenguaje de programación es un lenguaje formal que permite comunicarnos con una máquina, es la vía para darle instrucciones de una forma entendible a las computadoras, a través de algoritmos y reglas lógicas. Como todos los lenguajes está constituido por reglas gramaticales, que se dividen en sintácticas y semánticas que definen su estructura, diferenciándolo de otros lenguajes. Dentro de los lenguajes de programación más usados según el índice TIOBOE[2]en octubre 2013 se encuentra Java, posicionado entre los primeros lugares.

Java

Es un lenguaje de programación concurrente orientado a objetos, desarrollado por Sun Microsystems. Se deriva de C y C++, eliminando parte de las complejidades que muestran estos lenguajes, es independiente de plataforma, es ejecutado sobre una máquina virtual que le permite no verse afectado por la arquitectura de la computadora(22).

Java cuenta con procedimientos que garantizan la seguridad durante la ejecución, no viola las restricciones de seguridad del sistema donde se ejecute, este gestor de seguridad que posee permite controlar el acceso de la aplicación a los recursos del sistema. Su licencia es bajo la Licencia Pública General de GNU, dándole más popularidad frente a otros lenguajes.

Debido a la simplicidad, robustez, evidenciada al eliminar la preocupación de manejo de memoria al programador; ser un lenguaje con licencia pública, lo que su utilización no genera costo. Por su portabilidad permite que pueda ser compilado en cualquier computadora que tenga el interprete de java.

[2]Más información en: http://www.tiobe.com

También por ser dinámico, es decir, no necesita para que funcione compilar todas las clases de un programa, y por tener la capacidad de ser multiproceso; se selecciona java como lenguaje para la implementación de la herramienta.

1.6.3 Entorno de Desarrollo Integrado (IDE)

Un IDE es un programa informático compuesto por herramientas de programación, puede dedicarse para un solo lenguaje de programación o bien, poder utilizarse para varios. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de NetBeans que soporta una gran variedad de lenguajes de programación(23).

NetBeans 7.3

Es un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones web, empresariales, de escritorio y aplicaciones móviles utilizando la plataforma Java, así como PHP, JavaScript, Ajax, C / C++, HTML5, JavaScript y normas CSS3. NetBeans dispone de soporte para crear interfaces gráficas de forma visual, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis, entre otros(24).

Características:

- Instalación y actualización simple.
- Compatibilidad para múltiples lenguajes.
- Características visuales para el desarrollo web.
- Desarrollo rápido de interfaz de usuario.
- Multi – plataforma.

Debido a que soporta el lenguaje de programación java, ser una plataforma de aplicaciones y permitir un desarrollo rápido de las interfaces de usuarios, lo que agiliza el proceso de implementación de la herramienta. Ser una herramienta de código abierto y poseer soporte para control de versiones, lo que

permite la colaboración entre varios desarrolladores. Se decide utilizar NetBeans 7.3 como IDE a utilizar en el proceso de desarrollo de la aplicación.

Conclusiones parciales

En el presente capítulo fue realizado un estudio y caracterización de los conceptos de bases de datos, sistemas gestores de bases de datos y enmascarado de datos, que consolidan un mejor entendimiento del trabajo. A partir del estudio de las herramientas Informática Data Masking, DataMaskingSuite y Mask-PG, se logró constatar, que según la bibliografía estudiada no existe un sistema que sea capaz de enmascarar los datos para todas las bases de datos PostgreSQL, Microsoft SQL Server y MySQL; además se identificaron las técnicas de enmascarado de datos: sustitución, baraja, anulación de salida o eliminación y variación de números y fecha. De las arquitecturas básicas utilizadas en el diseño de sistemas de enmascarado de datos se seleccionó la arquitectura In-Situ para el diseño de la aplicación, fue identificada como metodología de desarrollo XP y se seleccionó Java y NetBeans, como lenguaje e IDE que se utilizarán en el desarrollo del sistema.

CAPÍTULO 2: DESCRIPCIÓN DEL DESARROLLO DE LA HERRAMIENTA BROOKESIA

Introducción

En el presente capítulo se hace una descripción del desarrollo de la herramienta de enmascarado de datos, detallando los patrones que se utilizan para modelar el diseño de la solución y una breve explicación y ejemplificación de los artefactos utilizados para el desarrollo.

2.1 Propuesta dela herramienta a desarrollar

Para darle cumplimiento al objetivo general se desarrollará una herramienta que implementará las técnicas de enmascarado de datos siguientes: baraja, sustitución, variación de números y fechas, anulación de salida o eliminación y enmascarado de datos de salida, de manera que garantice que los datos sensibles almacenados en BD de PostgreSQL, MySQL y SQL Server mantengan la seguridad y confidencialidad de la información.

La herramienta llevará por nombre Brookesia, en representación de una especie de camaleón de la Isla de Madagascar que cambia de color y simula su presencia en el ambiente, una forma eficiente de lograr un enmascarado. La misma contará con un módulo de conexión, que permitirá que el usuario se conecte a la BD deseada, seguido de esto se generará un árbol jerárquico representativo de la BD conectada, donde el usuario puede configurar cada objeto del árbol (BD, esquema, tabla y columna) que se desea enmascarar según la estrategia de enmascarado pertinente, y tendrá una configuración automática que posibilitará hacer un enmascarado predefinido. Además contará con un módulo de visualización que permitirá tener una vista previa de los datos antes de realizar el proceso de enmascarado y después de realizado.

2.2 Modelo de dominio

El Modelo de Dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real, no de los componentes de software. Una clase conceptual puede ser una idea o un objeto físico (símbolo, definición y extensión)(25).

La metodología XP no incorpora en sus artefactos el modelo de dominio, pero se decide realizar para proporcionar una mejor representación de los principales conceptos que se manejan en el negocio y se tenga una mejor comprensión del problema que se pretende resolver.

En el diagrama de dominio el usuario interactúa con las bases de datos, donde estas pueden ser de PostgreSQL, MySQL o SQL Server. Estas BD hasta el momento solo se encriptaban o borraban para garantizar la seguridad de la información que almacenan. En la fig. 1 se muestra el modelo propuesto para la solución a implementar.

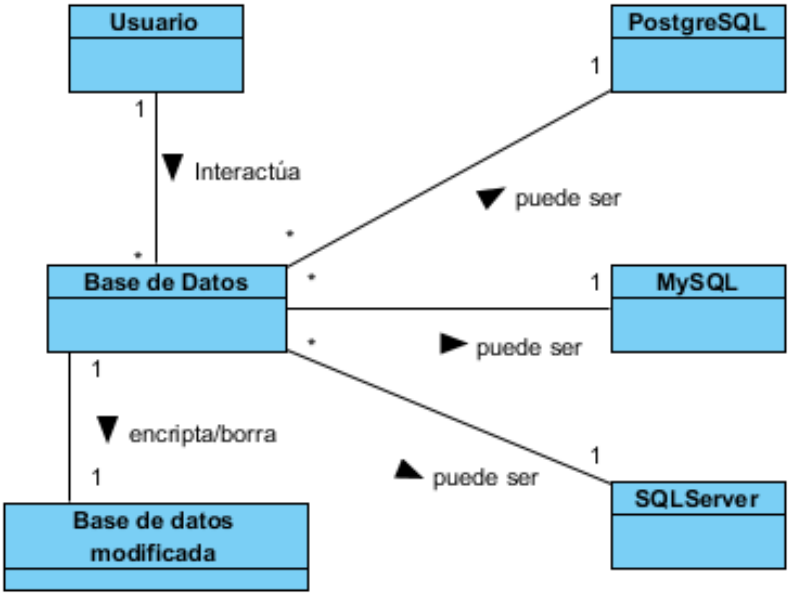


Fig. 1 Modelo de dominio.

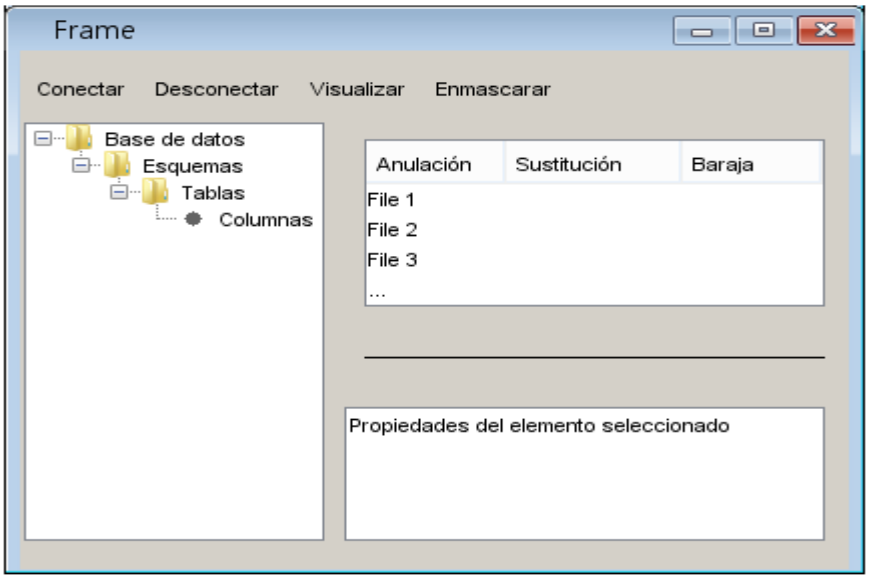
2.3 Historias de usuario

Las historias de usuarios (HU) son la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales se describe brevemente las características que el sistema debe poseer. El tratamiento de las HU es muy dinámico y flexible, cada una de ella es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas(26). A continuación se muestra una de las HU generada para la implementación de la herramienta de enmascarado de datos Brookesia.

Tabla 8 Historia de usuario Enmascarar datos con la estrategia de anulación en base de datos de PostgreSQL.

Historia de Usuario	
Número:1	**Nombre de la Historia de Usuario:** Enmascarar datos

	con la estrategia de anulación en base de datos de PostgreSQL.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Josué Mojena Marín	**Iteración asignada:** 1
Prioridad en negocio: Muy Alta	**Puntos estimados:**0.8 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 0.8 semanas
Descripción: Aplicar a los datos seleccionados de una base de datos de PostgreSQL el algoritmo de anulación, el cual remplazará los campos correspondientes por datos de tipo null o elimina completamente la información.	
Observaciones: En caso de que se aplique el algoritmo y exista algún error en el proceso se mostrará un mensaje de error, en caso de realizar la operación correctamente se mostrará un mensaje de que se ha enmascarado satisfactoriamente.	
Prototipo de interfaces:	



2.4 Lista de reserva del producto

La lista de Reserva del Producto (LRP) es una tabla que contiene los requisitos funcionales ordenados por la prioridad de implementación y los requisitos no funcionales con una breve descripción de cada uno.

Además se refleja la estimación de la implementación por semanas de cada uno de ellos, y se define el rol que hizo la estimación. A continuación se muestra la LRP del sistema.

Tabla 9 Lista de reserva del producto.

Ítem *	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Enmascarar datos de bases de datos de PostgreSQL con la estrategia de anulación.	0.8 Semanas	Analista
2	Enmascarar datos de bases de datos de MySQL con la estrategia de anulación.	0.4 Semanas	Analista
3	Enmascarar datos de bases de datos de SQL Server con la estrategia de anulación.	0.4 Semanas	Analista
4	Enmascarar datos de bases de datos de PostgreSQL con la estrategia de variación de números o fechas.	1 Semanas	Analista
5	Enmascarar datos de bases de datos de MySQL con la estrategia de variación de números o fechas.	0.4 Semanas	Analista
6	Enmascarar datos de bases de datos de SQL Server con la estrategia de variación de números o fechas.	0.4 Semanas	Analista
7	Enmascarar datos de bases de datos de PostgreSQL con la estrategia de sustitución.	1.4 Semanas	Analista
8	Enmascarar datos de bases de datos de MySQL con la estrategia de sustitución.	0.4 Semanas	Analista

9	Enmascarar datos de bases de datos de SQL Server con la estrategia de sustitución.	0.4 Semanas	Analista
10	Enmascarar datos de bases de datos de PostgreSQL con la estrategia de baraja.	1.2 Semanas	Analista
11	Enmascarar datos de bases de datos de MySQL con la estrategia de baraja.	0.4 Semanas	Analista
12	Enmascarar datos de bases de datos de SQL Server con la estrategia de baraja.	0.4 Semanas	Analista
13	Enmascarar datos de bases de datos de PostgreSQL con la estrategia de enmascarado de datos de salida.	1 Semanas	Analista
14	Enmascarar datos de bases de datos de MySQL con la estrategia de enmascarado de datos de salida.	0.4 Semanas	Analista
15	Enmascarar datos de bases de datos de SQL Server con la estrategia de enmascarado de datos de salida.	0.4 Semanas	Analista
Prioridad: Alta			
16	Conectarse a una base de datos de PostgreSQL.	0.6 Semanas	Analista
17	Conectarse a una base de datos de MySQL.	0.4 Semanas	Analista
18	Conectarse a una base de datos de SQL Server.	0.4 Semanas	Analista
Prioridad: Media			
19	Mostrar los datos de las bases de datos.	0.4Semanas	Analista

20	Mostrar el árbol jerárquico de las bases de datos.	0.4 Semanas	Analista
21	Mostrar el progreso de los procesos de la aplicación.	0.6 Semanas	Analista
22	Mostrar las propiedades de los elementos en las bases de datos.	0.2 Semanas	Analista
Prioridad: Baja			
23	RNF - El usuario de la aplicación debe tener un conocimiento básico de PostgreSQL, MySQL y/o SQL Server, en dependencia de la base de datos que se vaya a enmascarar. (Usabilidad)		Analista
24	RNF - La herramienta contará con un enmascarado por paginación e hilos de ejecución, que permitirá que el tiempo de respuesta disminuya, según las prestaciones de la PC. (Rendimiento)		Analista
25	RNF - La herramienta será multiplataforma, lo que le permite ser ejecutada sobre cualquier sistema operativo. (Portabilidad)		Analista
26	RNF - Para su ejecución será necesario tener instalada la máquina virtual de java JVM. (Software)		Analista
27	RNF - La computadora debe tener como características mínimas 1 GB de RAM así como un espacio libre en el disco duro de 30 MB. (Hardware)		Analista
28	RNF - Solo podrá acceder a los datos sensibles el usuario que tenga privilegios de lectura en la BD dada. (Confidencialidad)		Analista
29	RNF - No se podrá enmascarar la BD, a menos que el usuario		Analista

	tenga privilegios para la modificación de la misma. (Integridad)		
30	RNF - Se podrá hacer uso de la aplicación siempre que esté instalada y exista una BD para enmascarar. (Disponibilidad)		Analista

2.5 Plan de iteraciones

Esta fase incluye varias iteraciones de desarrollo sobre el sistema donde ninguna excederá de tres semanas. Una vez descritas e identificadas las Historias de Usuarios y estimado el esfuerzo propuesto para la realización de cada una de ellas, se procede a la planificación de la etapa de implementación del sistema. En esta se especifica cuáles son las HU que van a ser implementadas para cada iteración y se determinan las posibles fechas de entrega(27).

Se definieron 5 iteraciones para el desarrollo del sistema, las cuáles se detallan a continuación:

Tabla 10 Plan de iteraciones.

Iteraciones	Orden de las HU a implementar	Duración de las iteraciones
Iteración 1	Enmascarar datos de bases de datos de PostgreSQL con la estrategia de anulación.	3 Semanas
	Enmascarar datos de bases de datos de MySQL con la estrategia de anulación.	
	Enmascarar datos de bases de datos de SQL Server con la estrategia de anulación.	
	Enmascarar datos de bases de datos de PostgreSQL con la estrategia de variación de números o fechas.	
	Enmascarar datos de bases de datos de MySQL con la estrategia de variación de números o fechas.	
Iteración 2	Enmascarar datos de bases de datos de SQL Server con la estrategia de variación de números o fechas.	2.6 Semanas
	Enmascarar datos de bases de datos de PostgreSQL con la	

	estrategia de sustitución.	
	Enmascarar datos de bases de datos de MySQL con la estrategia de sustitución.	
	Enmascarar datos de bases de datos de SQL Server con la estrategia de sustitución.	
Iteración 3	Enmascarar datos de bases de datos de PostgreSQL con la estrategia de baraja.	2.6 Semanas
	Enmascarar datos de bases de datos de MySQL con la estrategia de baraja.	
	Enmascarar datos de bases de datos de PostgreSQL con la estrategia de enmascarado de datos de salida.	
Iteración 4	Enmascarar datos de bases de datos de SQL Server con la estrategia de baraja.	3 Semanas
	Enmascarar datos de bases de datos de MySQL con la estrategia de enmascarado de datos de salida.	
	Enmascarar datos de bases de datos de SQL Server con la estrategia de enmascarado de datos de salida.	
	Conectarse a una base de datos de PostgreSQL.	
	Conectarse a una base de datos de MySQL.	
	Conectarse a una base de datos de SQL Server.	
	Mostrar los datos de las bases de datos.	
Iteración 5	Mostrar el árbol jerárquico de las bases de datos.	1.2 Semanas
	Mostrar el progreso de los procesos de la aplicación.	
	Mostrar las propiedades de los elementos en las bases de datos.	
Total		12.4 Semanas

2.6 Diseño del sistema

Para el diseño de aplicaciones informáticas la metodología XP no requiere la presentación del sistema mediante diagrama de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC (Clase, Responsabilidad y Colaboración). El uso de estos diagramas puede aplicarse siempre y cuando intervengan en el mejoramiento de la comunicación, tratando que no sean un peso en su mantenimiento, no sean extensos y que enfoquen la información de mayor importancia(28).

2.6.1 Tarjetas CRC

Para poder diseñar el sistema se debe cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas CRC permiten que el equipo completo contribuya en la tarea del diseño. Una tarjeta CRC representa un objeto, el nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente.

Nombre de las clases: Es cualquier persona, cosa, evento, concepto, pantalla o reporte.

Responsabilidades: Las responsabilidades de una clase son las cosas que conoce y las que realiza, sus atributos y métodos.

Colaboradores: Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

En la Tabla 14 se muestra un ejemplo de la tarjeta CRC Conexión:

Tabla 11 Tarjeta CRC Conexión.

Tarjeta CRC	
Clase: Conexión.	
Responsabilidades	**Colaboraciones**
- Conectarse a una BD de PostgreSQL.	

- Conectarse a una BD de MySQL. -Conectarse a una BD de SQL Server.	

2.6.2 Diagrama de clases

La flexibilidad de la metodología ágil XP permite hacer usos de cualquier artefacto que sea necesario para el desarrollo exitoso del sistema, por lo que se modeló un diagrama de clases para observar el comportamiento de los patrones de diseño y las relaciones entre las clases del sistema. El diagrama de clases es una abstracción de una o varias clases en el diseño dela solución. El diagrama se encuentra en el expediente del proyecto en forma de imagen debido a su amplio tamaño. El mismo cuenta con 12 paquetes y 67 clases organizados según los patrones utilizados.

2.7 Patrones de arquitectura

Un patrón es una solución de diseño de software a un problema, aceptada como correcta, a la que se ha dado un nombre y que puede ser aplicada en otros contextos(29).

Un patrón es una solución a un problema, que por ser recurrente puede utilizarse en soluciones a otros problemas. Su propósito fundamental es la reutilización eficiente y diseminaciones de soluciones, representan un conjunto de literatura de resoluciones a problemas habituales.

El patrón modelo vista controlador (MVC) separa la lógica del negocio de la interfaz de usuario, donde ellas interactúan entre sí, a través de las clases controladoras. Este patrón facilita la evolución del sistema por separado, además de incrementar la reutilización y flexibilidad del código (Fig.2)(30).

Vista: Representa la interfaz de usuario y todas las herramientas con las cuales el usuario hace uso del sistema. Dentro de la vista se encuentran los paquetes HandlerStrategy, Vistas, TComponentes, Remplace, Ttree, DataBaseTree, TableProperty y el MainProject.

Modelo: Contiene toda la lógica del negocio, la representación de todo el sistema incluido la interacción con una base de datos. Dentro del modelo están los paquetes Conection, Querys, Factory y Catálogos.

Controlador: Este componente es el que responde a la interacción (eventos) que hace el usuario en la interfaz y realiza las peticiones al modelo para pasar estos a la vista. Dentro de controlador están los paquetes Controls, Hilos y Strategy.

La siguiente figura muestra cómo se modela dicho patrón en el diseño de la solución:

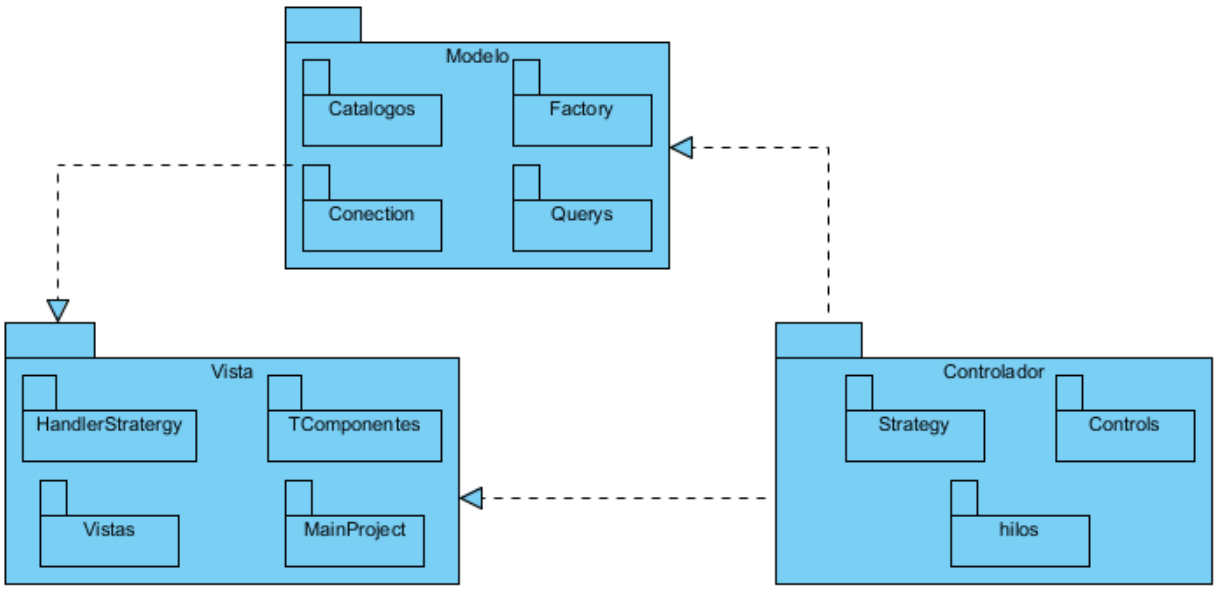


Fig. 2Patrón Modelo Vista Controlador.

2.8 Patrones de diseño

Un patrón de diseño es una forma conveniente para la reutilización de código orientado a objetos entre los proyectos y entre los programadores. La idea detrás de los patrones de diseño es simple: para catalogar comunes interacciones entre los objetos que los programadores a menudo han encontrado útiles.

Los patrones de diseño se han convertido en un elemento básico del diseño orientado a objetos y la programación, los mismos proporcionan soluciones elegantes, fáciles de reutilizar y de mantener(31).

Los patrones de diseños que se evidencian en la implementación están dentro de la Banda de Cuatro (GOF por sus siglas en inglés) y en los Patrones de Software para la Asignación General de Responsabilidad (GRASP).

2.8.1 Patrones GOF

Los patrones GOF describen soluciones simples y de una manera elegante a problemas en un contexto particular, dentro del diseño de software orientado a objetos. En la modelación del sistema se han tenido en cuenta los siguientes patrones GOF:

Abstract Factory: El patrón Abstract Factory es utilizado cuando se tienen familias de múltiples objetos que se manipulan de una manera similar. Su propósito es proporcionar una interfaz, que permita crear familias de objetos relacionados o dependientes, sin especificar sus clases concretas. Este patrón se pone en práctica en la familia de clases PgFactory, SqlServerFactory y MySQLFactory que heredan de la clase SGFactory (Fig.3).

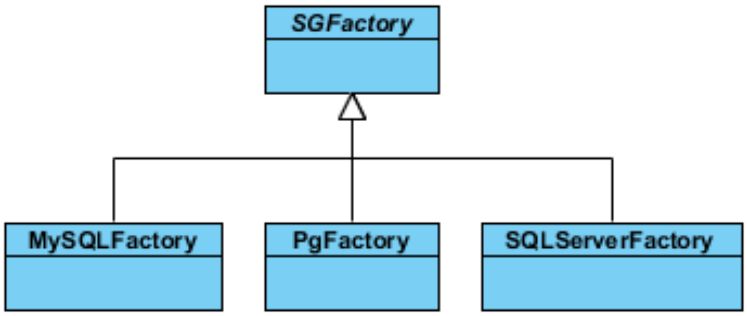


Fig. 3Patrón abstract factory.

Visitor: El patrón visitor representa una operación que se lleva a cabo sobre los elementos de una estructura de objetos. Permite definir nuevas operaciones para cada tipo de objeto sobre los que opera, evitando incluir estas operaciones en las clases objetos. Este patrón se evidencia en la Clase DMControl donde se hace una secuencia de visita entre los componentes de una BD para realizar el enmascarado (Fig.4).

```
67      public void maskBD(Tnode_dataBase data_base) throws Exception {
68          LinkedList<Tnode> squemas = data_base.getObjetos();
69          maskSchema(squemas);
70      }
71
72      public void maskSchema(LinkedList<Tnode> squemas) throws Exception {
73          for (Tnode esquema : squemas) {
74              if (esquema.isMask()) {
75                  LinkedList<Tnode> tablas = ((Tnode_Schema) esquema).getObjetos();
76                  maskTable(tablas);
77              }
78          }
79      }
80
81      public void maskTable(LinkedList<Tnode> tables) throws SQLException, Exception {
82          for (Tnode tabla : tables) {
83              if (tabla.isMask()) {
84                  LinkedList<Tnode> columnas = ((Tnode_Table) tabla).getObjetos();
85                  maskColumn(columnas);
86              }
87          }
88      }
```

Fig. 4 Patrón visitor.

Strategy: El patrón strategy define una familia de algoritmos, de tal manera que encapsula cada uno de ellos y los hace intercambiables entre sí. Garantiza total independencia entre las operaciones de los algoritmos y quien los usa. Este patrón se evidencia en la familia de clases de los algoritmos que heredan de la interfaz MaskStrategy (Fig.5).

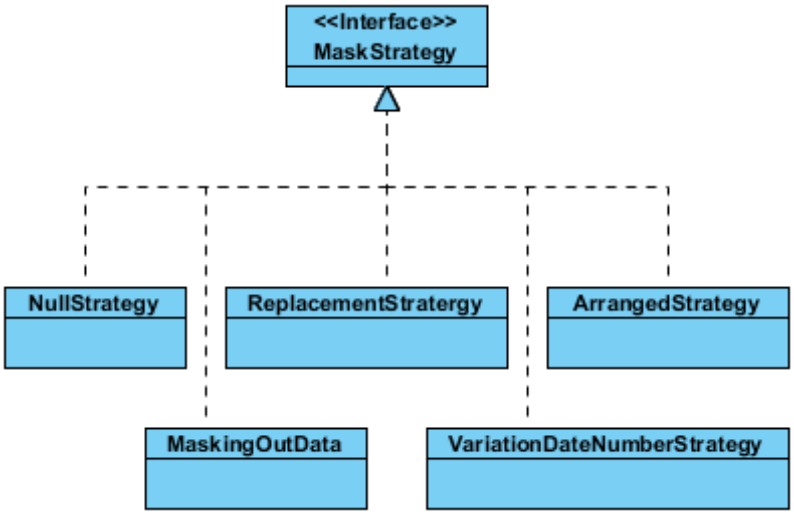


Fig. 5Patrón strategy.

Handler: El patrón Handler es utilizado para manejar identificadores de objetos manteniendo la independencia con su implementación. Este patrón fue modelado para manejar la visualización de las configuraciones de los algoritmos de enmascarado (Fig.6).

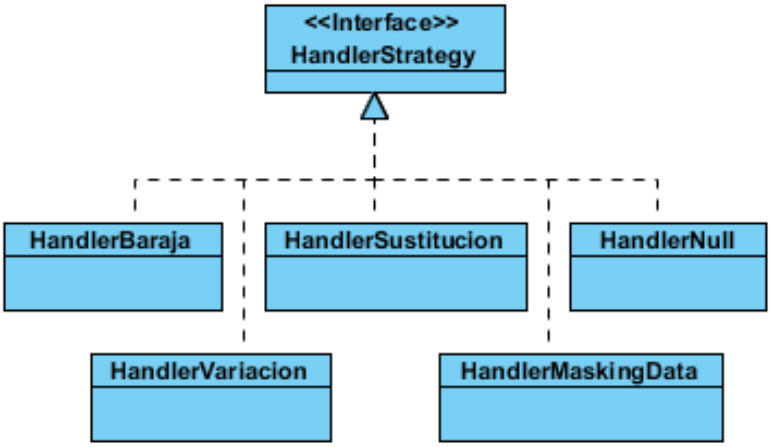


Fig. 6Patrón handler.

2.8.2 Patrones GRASP

Los patrones GRASP definen básicamente quien hace que cosa en un instante dado. Para la modelación de la solución se han tenido en cuenta los siguientes patrones GRASP.

Experto: Este patrón define que la clase encargada de realizar alguna función, es la que contiene o puede contener los datos necesarios para realizarla, asigna la responsabilidad directa a la clase experta, que contiene dicha información. Este patrón fue modelado para manejar la conexión de las BD, a través de la clase Conection, que asigna la responsabilidad de conexión a las clases ConectionPostgres, ConectionMySql y ConectionSQLServer según sea necesario (Fig.7).

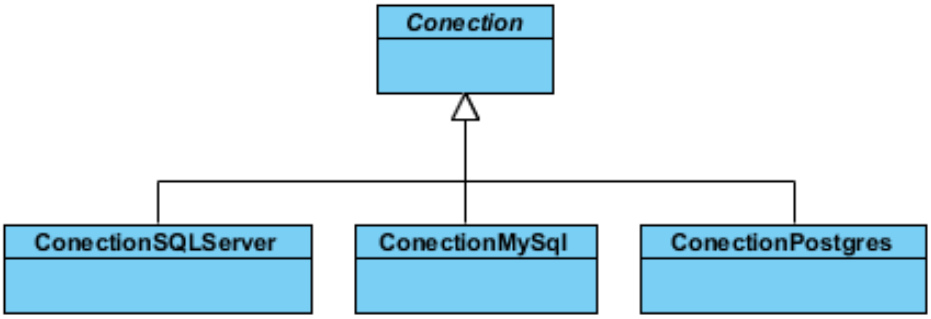


Fig. 7Patrón experto.

Creador: Este patrón define que clase crea que objeto, para ello la clase creadora debe contener, agregar, componer, registrar, tener los datos de inicialización o utilizar muy de cerca otra clase. Para manejar las funciones de creación de conexión y de catálogos, se tuvo en cuenta la modelación de este patrón para las clases SGFactory, PgFactory, MySQLFactory y SQLServerFactory (Fig.8).

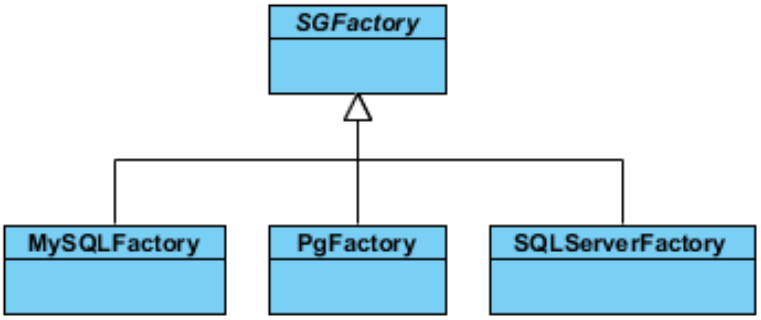


Fig. 8Patrón creador.

Bajo Acoplamiento: Es necesario que existan pocas dependencias entre las clases, evitar que todas las clases estén relacionadas entre sí, para ello es necesario que no exista una herencia muy profunda. Esto garantiza una mejor reutilización del código en otros proyectos. Este patrón se evidencia en la relación de la clase DMControl con la clase Catalogo (Fig.9).

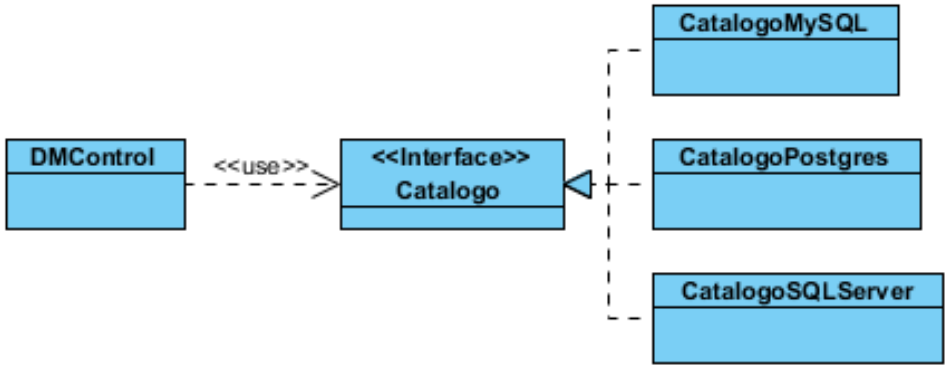


Fig. 9Patrón bajo acoplamiento.

Alta cohesión: Para que haya una alta cohesión es necesario que cada elemento o clase en nuestro diseño realice una única función en la solución, y que solo sea realizada por ella. Para lograr esto es necesario realizar una asignación adecuada de responsabilidades y evitar que las clases estén saturadas de métodos. Este patrón se puede evidenciar en las Clases Querys donde PostgreSQLQuerys, mySQLQuerys y SQLServerQuerys realizan específicamente las consultas para su gestor en cuestión (Fig.10).

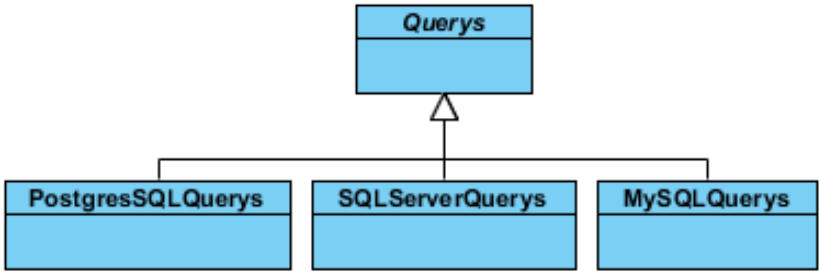


Fig. 10Alta cohesión.

Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión(32). En el diseño de la solución la clase DMControl evidencia dicho patrón (Fig. 11).

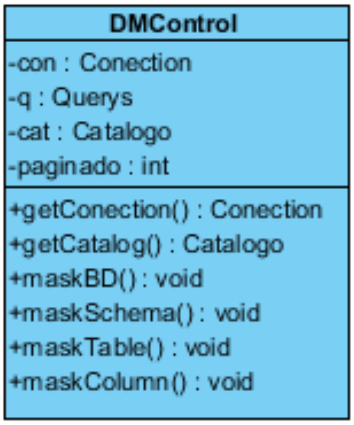


Fig. 11Patrón controlador.

Conclusiones parciales

Se identificaron 22 HU, en las cuales se agrupan un mismo número de requisitos, se planificó su implementación para 5 iteraciones comenzando por las de mayor prioridad en un tiempo de 12.4 semanas. Se definieron los requisitos no funcionales de usabilidad, portabilidad, software, hardware, apariencia o interfaz externa y de seguridad con los que debe cumplir el sistema. Se hizo uso del patrón arquitectónico MVC y los patrones de diseño GOF como abstract factory, visitor, strategy y handler; también los GRASP como el experto, creador, bajo acoplamiento, alta cohesión y controlador, todos empleados en el diseño de la solución. Se confeccionaron un total de 8 tarjetas CRC. Se definió como estándar de codificación a emplear durante la implementación del sistema el que dictan las convenciones de código para el lenguaje de programación Java.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DE LA HERRAMIENTA BROOKESIA

Introducción

En el presente capítulo se describe la implementación de la herramienta Brookesia. Además se especifican las pruebas a las que fue sometida con el fin de garantizar una alta calidad y satisfacción por parte de los clientes, llevando a cabo las pruebas según lo descrito en los casos de pruebas basados en historias de usuario.

3.1 Tareas de la Ingeniería

De las HU se derivan una o varias tareas de ingeniería, estas son pasos lógicos a seguir por los desarrolladores, cada tarea tiene relacionada una HU, tiempo estimado, programador asignado y una descripción breve de la misma. A continuación se presentan las tareas de ingeniería para la HU enmascarar datos con la estrategia de anulación en base de datos de PostgreSQL:

Tabla 12 Tarea de ingeniería 1.

Tarea de Ingeniería	
Número Tarea:1	**Número Historia de Usuario: 1**
Nombre Tarea: Implementar anulación de datos.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0,4 semanas
Fecha Inicio: 3 / 2 / 2014	**Fecha Fin:** 4 / 2 / 2014
Programador Responsable: Oswaldo Victor Melo Pérez.	
Descripción: Anula un conjunto de datos en forma de columna.	

Tabla 13 Tarea de ingeniería 2.

Tarea de Ingeniería	
Número Tarea:2	**Número Historia de Usuario: 1**
Nombre Tarea: Actualizar los datos modificados.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0,2 semanas
Fecha Inicio: 5 / 2 / 2014	**Fecha Fin:** 5 / 2 / 2014
Programador Responsable: Josué Mojena Marín.	
Descripción: Ejecuta una actualización en la base de datos en PostgreSQL.	

Tabla 14 Tarea de ingeniería 3.

Tarea de Ingeniería	
Número Tarea:3	**Número Historia de Usuario: 1**
Nombre Tarea: Crear interfaz del algoritmo de anulación.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0,2 semanas
Fecha Inicio: 6 / 2 / 2014	**Fecha Fin:** 6 / 2 / 2014
Programador Responsable: Oswaldo Victor Melo Pérez.	
Descripción: Se desarrolla la interfaz del algoritmo de anulación. Se debe asegurar que el formulario para el algoritmo permita la selección de la base de datos, tablas y columnas que se desean enmascarar, además debe proporcionar los tipos de datos de las columnas así como la cantidad de filas que posean las tablas.	

3.2 Estándares de codificación

La implementación se realizará bajo los estándares de codificación que dictan las Convenciones de Código para el lenguaje de programación Java. Son de gran importancia debido a las siguientes razones:

- ✓ El 80% del coste del código de un programa va a su mantenimiento.
- ✓ Casi ningún software lo mantiene toda su vida el autor original.
- ✓ Las convenciones de código mejoran la lectura del software, permitiendo entender código nuevo mucho más rápidamente y más a fondo.
- ✓ Si se distribuye el código fuente como un producto, es necesario asegurarse de que está bien hecho y presentado como cualquier otro producto(33).

Los tipos de comentarios que se utilizarán en la implementación serán los siguientes:

- ✓ **Comentarios de documentación (JavaDoc):** Solo son utilizados en Java, pueden exportarse a ficheros HTML. Tienen gran utilidad ya que explican las características de los métodos, ¿cómo funciona?, ¿qué parámetros recibe? y ¿qué devuelve? Permite la lectura de la descripción de un método desde una instancia donde se le hace una llamada.

```
7   /**
8    * @return devuelve el nombre de las base de datos que se encuentran en el
9    * servidor específico
10   * @throws java.lang.Exception
11   */
```

Fig. 12 Comentarios de documentación.

- ✓ **Comentarios de bloque:** Estos comentarios pueden ser usados para describir ficheros, métodos, estructuras de datos y algoritmos. Pueden utilizarse para describir algún proceso dentro de un método.

```
7    /*
8     * Set the Nimbus look and feel
9     * If Nimbus (introduced in Java SE 6) is not available, stay with the defau
10    * details see
11    * http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
12    */
```

Fig. 13 Comentarios de bloque.

- ✓ **Comentarios de una línea:** Comentario pequeño que solo abarca una línea y describe el código que le sigue.

```
7    /*llenando la tabla*/
```

Fig. 14 Comentarios de una línea.

- ✓ **Comentarios de remolques:** Comentarios que aparecen en la misma línea de código que describen.

```
56   for (int i = 0; i < cant; i++) {/* para crear todos los esquema que contiene la BD*/
```

Fig. 15 Comentarios de remolque.

- ✓ **Comentarios de fin de línea:** Este comentario no debe ser usado para comentarear varias líneas consecutivas, con excepciones de comentar secciones de código consecutivas.

```
42           // GenerateProperties();
```

Fig. 16 Comentarios de fin de línea.

Las declaraciones que estarán presentes en el código del sistema tendrán las siguientes características:

- ✓ **Cantidad por línea:** Es recomendable una sola declaración por línea.

```
86           String tableName;
87           String schemaName;
```

Fig. 17 Cantidad de declaraciones por línea.

- ✓ **Inicialización:** Inicializar las variables donde se declaran, a no ser que su valor inicial dependa de algún cálculo o procedimiento.

```
85          Map mp = new HashMap();
```

Fig. 18 Inicialización.

- ✓ **Colocación:** es recomendable hacer las declaraciones al principio del bloque, no esperar su primer uso. Para evitar la confusión de programadores y facilitar la reutilización.

```
    public void GenerateProperties() throws Exception {
85      Map mp = new HashMap();
86      String tableName;
```

Fig. 19 Colocación.

- ✓ **Declaraciones de clases e interfaces:** No debe de haber ningún espacio en blanco entre el nombre de un método y el paréntesis "(". La llave de apertura "{" debe aparecer después de la declaración y la llave de cierre "}" en una nueva línea alineada justo a su sentencia de apertura, excepto cuando no haya código implementado entre ellas.

```
16  public class PgFactory extends SGFactory {
17
        private ConectionPostgres con;
19
20      public PgFactory() {
21          con = new ConectionPostgres();
22      }
23
24      @Override
        public Conection createConection() {
26          return con;
27      }
28
29      @Override
        public Catalogo createCatalog() {
31          return new CatalogoPostgres(con);
32      }
33
34  }
```

Fig. 20 Declaraciones de clases e interfaces.

Las sentencias se implementarán manteniendo las siguientes normas:

- ✓ **Sentencias Simples:** Una sentencia por línea.

```
21          con = new ConectionPostgres();
```

Fig. 21 Sentencias simples.

- ✓ **Sentencias compuestas:** Son sentencias que contienes listas de sentencias. La sentencia encerrada entre llave debe estar alineada un nivel más. La llave de apertura "{" debe aparecer al final de la línea de la sentencia compuesta y la de cierre "}" en una línea nueva al nivel de la sentencia compuesta.

```
56    for (int i = 0; i < cant; i++) {
57        Tnode_Schema aux = new Tnode_Schema(Type_key.SCHEMAS, nomSchemas.get(i), conection, catalog, preces);
58        aux.setData_base_parent(this);
59        aux.setParent(this);
60        addObjeto(aux);
61        aux.GenerateObject();
62    }
```

Fig. 22 Sentencias compuestas.

El estilo de los nombres será de la siguiente manera:

- ✓ **Clases e Interfaces:** Los nombres de las clases tendrán la primera letra mayúscula, y si es compuestos la primera letra de cada palabra en mayúscula, nombres simples y de alguna manera que describa el contenido, usar palabras completas, a no ser que la abreviatura sea muy conocida, ej. (UML, BD).

```
16    public class PgFactory
11    public class Column
```

Fig. 23 Nombre de clases e interfaces.

- ✓ **Métodos y variables:** En el caso de los métodos será en minúscula, en caso de ser un nombre compuesto la inicial de la primera palabra en minúscula y la de las otras palabras que lo componen en mayúscula. Los nombres de las variables no deben comenzar con los caracteres guión bajo "_"

o signo de peso “$”, deben ser cortos pero con significados lógicos, capaz de permitir a un observador identificar su función.

```
31      public void createCatalog() {
32          int contMax;
33      }
34
35       public void create() {
36          int cont;
37      }
```

Fig. 24 Nombre de métodos y variables.

3.3 Interfaces principales de la aplicación

El sistema desarrollado consta de dos interfaces, una para realizar la configuración de la conexión a una BD que será enmascarada (Fig. 25) y otra para administrar el enmascarado a realizar en la misma (Fig. 26).

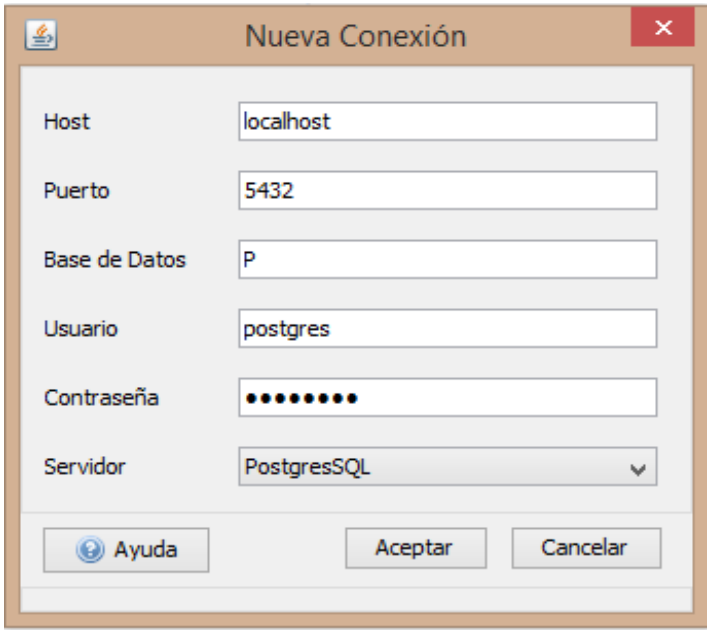


Fig. 25Configuración de la conexión a una BD.

La fig. 25 muestra la vista de cómo se configura la conexión a la BD que será enmascarada, donde se deben especificar los parámetros: servidor, puerto, nombre de la BD, usuario, contraseña y gestor de base de datos.

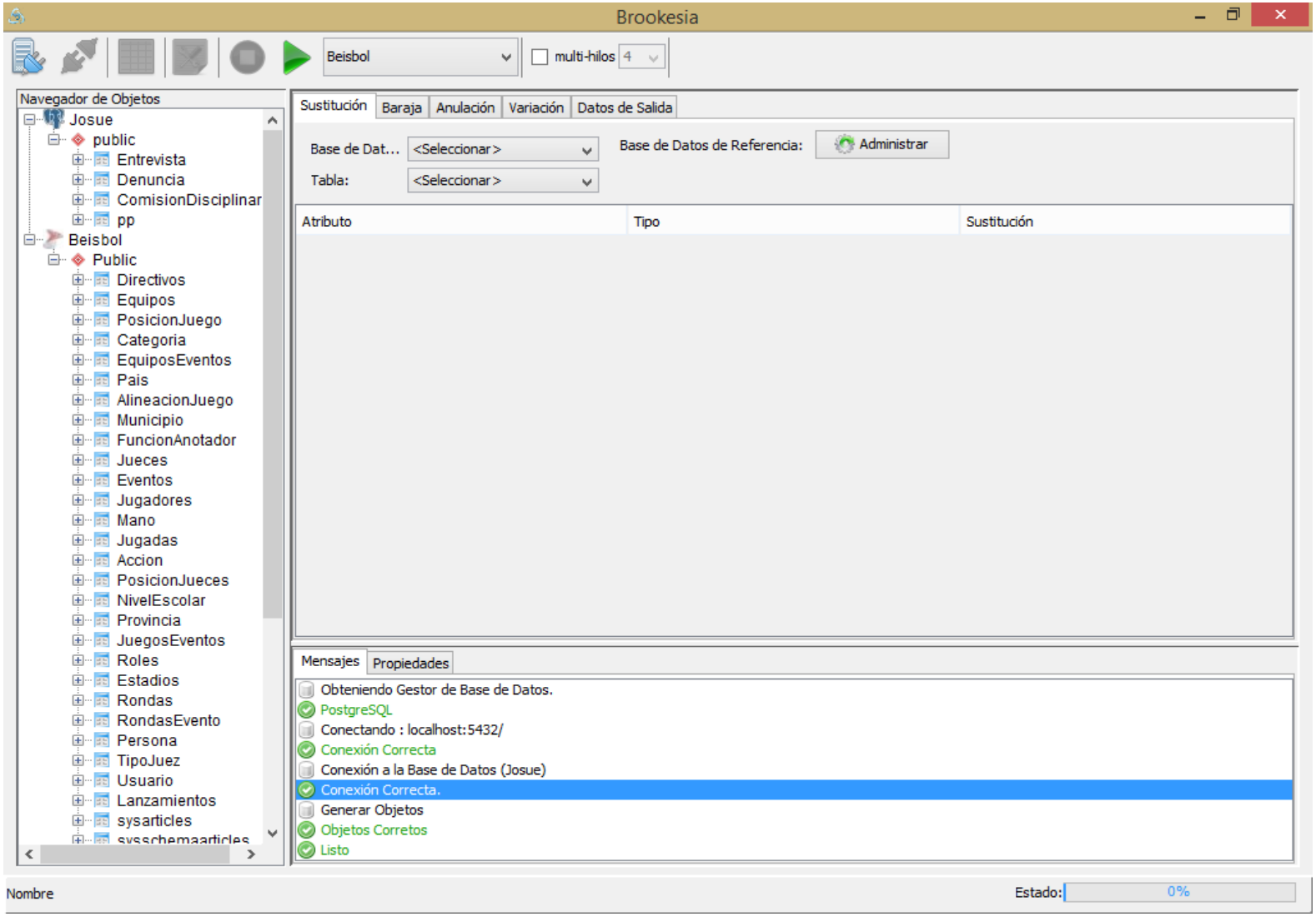


Fig. 26Administrar el enmascarado.

En la Fig. 27 se puede apreciar la barra de tareas donde se realizan todas las configuraciones pertinentes para el enmascarado de datos. Se brindan varias opciones en botones para la interacción con la herramienta. De izquierda a derecha, está conectarse a una BD, que mostrará una interfaz para la configuración de la conexión; luego la desconexión de una BD, que desconectará de la aplicación la que se encuentre seleccionada en el panel de navegación. El botón visualizar tabla permite la visualización de los datos contenidos en una tabla, previamente seleccionada en el panel de navegación. El de configuración, brinda las alternativas de realizar la configuración del enmascarado de datos de forma manual o automática. Por último, el control del proceso de enmascarado tanto iniciar como parar, el cual inicia o termina la ejecución del enmascarado.

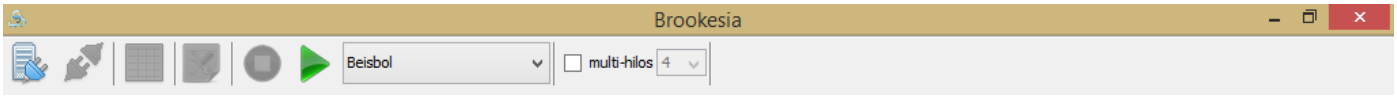


Fig. 27 Barra de tareas.

En la parte izquierda de la vista está el panel de navegación, donde se permite navegar y ser seleccionadas las BD conectadas, así como toda la estructura que las componen (Fig. 28). Además, cuando es seleccionado algún objeto del árbol jerárquico, se despliegan en el panel de notificaciones (Fig. 30), todas las propiedades correspondientes al objeto seleccionado.

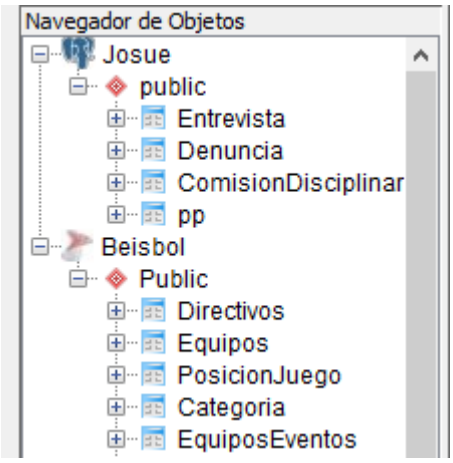


Fig. 28 Panel de navegación.

Otro panel que conforma la interfaz principal del sistema, es el de trabajo. En este se selecciona la BD, tabla y columna a enmascarar; también es donde se realizan todas las configuraciones pertinentes para el enmascarado de datos (Fig. 29).

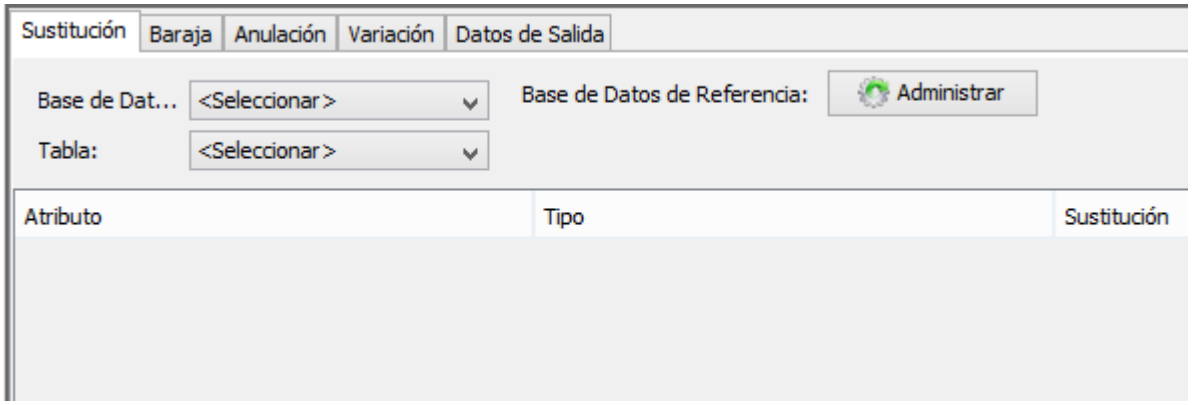


Fig. 29 Área de trabajo.

En caso de seleccionar manual la configuración para ejecutar el enmascarado, se podrá elegir por cada tabla contenida en la BD, los campos a enmascarar y la técnica a utilizar en cada caso. También en la parte inferior derecha muestra una barra de progreso, que reflejará en porcientos el cumplimiento de cada tarea realizada en la aplicación, desde su ejecución hasta su cierre (Fig. 30).

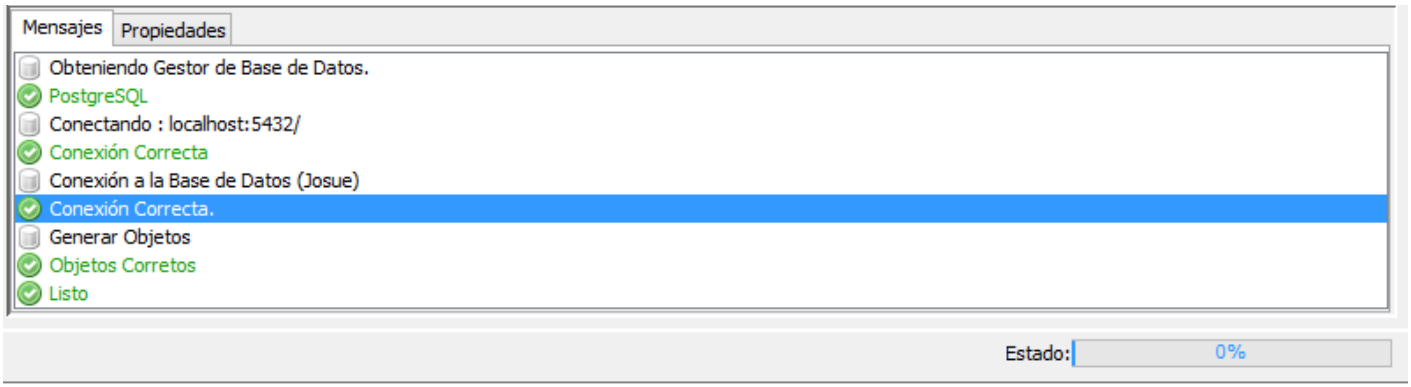


Fig. 30 Área de notificaciones.

3.4 Pruebas

El proceso de pruebas o validación de un sistema, es una de las etapas fundamentales del desarrollo del mismo. Es donde se valida la calidad y la aceptación que tiene la herramienta para el cliente.

"Cualquier pieza de software completo, desarrollado o adquirido, puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento"(34).

3.4.1 Estrategias de Prueba

"La Estrategia de Prueba de software integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba: la planificación, el diseño de casos de prueba, la ejecución y los resultados, tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software"(35).

La estrategia de pruebas se hace con el objetivo de que el producto de software en desarrollo, reúna todos los requisitos planteados por el cliente mediante la lógica del negocio. La estrategia debe ser definida claramente, pues comprende las ideas más representativas del proceso de pruebas que se llevará a cabo. Si se tiene en cuenta una correcta planificación, puede proporcionar menor tiempo de ejecución y disminuir los costos que se generan.

"Una estrategia de prueba del software debe ser lo suficientemente flexible como para promover un enfoque personalizado. Al mismo tiempo, debe ser lo adecuadamente rígido como para promover una planeación razonable y un seguimiento administrativo del avance del proyecto"(35).

En XP se ha desarrollado un enfoque que reduce la probabilidad de producir nuevos incrementos del sistema que introduzcan errores en el software existente (36). XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones(37).

XP como metodología ágil y enfocada principalmente en las necesidades del cliente, propone para validar el correcto funcionamiento del producto y dirigir la implementación, la realización de pruebas unitarias y pruebas de aceptación.

Pruebas unitarias

Las pruebas unitarias son pruebas definidas por los programadores. Estas se le hacen a pequeñas porciones de código por separados, para garantizar que determinado módulo satisfaga un comportamiento esperado antes de integrarse al sistema. Deben ser definidas antes de realizar el código y se pueden ir ejecutando a medida que se está implementando, no es indispensable esperar al final de la implementación del software(38).

Estas pruebas tienen como objetivo fundamental asegurar el correcto funcionamiento de las interfaces o flujos de datos entre componentes. Las mismas son empleadas cuando la interfaz de un método no es clara y la implementación es complicada.

Pruebas de aceptación

Las pruebas de aceptación son supervisadas por el cliente basándose en los requisitos tomados de las historias de usuario. En todas las iteraciones, cada una de las historias de usuario deberá tener una o más pruebas de aceptación, de las cuales se determinarán los casos de prueba e identificarán los errores que serán corregidos.

Estas son escritas en fases tempranas del desarrollo antes de que sea iniciada la implementación, para validar las necesidades del cliente y dirigir la implementación. Las pruebas de aceptación son consideradas “pruebas de caja negra” (“*Black box system tests*”). Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución(38).

Entre otros tipos de pruebas que se realizan en un sistema está la que evalúa la funcionalidad de este, denominada prueba funcional, que tiene por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para las cuales han sido creados. Por lo que será un tipo de prueba a utilizar en la estrategia de pruebas definida para el sistema.

Se seleccionó la estrategia pruebas de aceptación para validar el sistema, debido a que estas responden a las necesidades del cliente. Este tipo de prueba garantiza que la herramienta cumpla con sus especificaciones funcionales.

3.4.2 Técnica de caja negra

"Las pruebas de caja negra son las que se aplican a la interfaz del software. Una prueba de este tipo examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software"(35).

Las pruebas de caja negra, también denominadas pruebas de comportamiento, se concentran en los requisitos funcionales del software. Es decir, permiten al ingeniero derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías:

- ✓ Funciones incorrectas o faltantes.
- ✓ Errores de interfaz.
- ✓ Errores de estructuras de datos o en acceso a bases de datos externas.
- ✓ Errores de comportamiento o desempeño.
- ✓ Errores de inicialización y término(35).

La técnica de caja negra posee varios métodos de pruebas, entre ellos: análisis de valores límites, partición de equivalencia y prueba de tabla ortogonal.

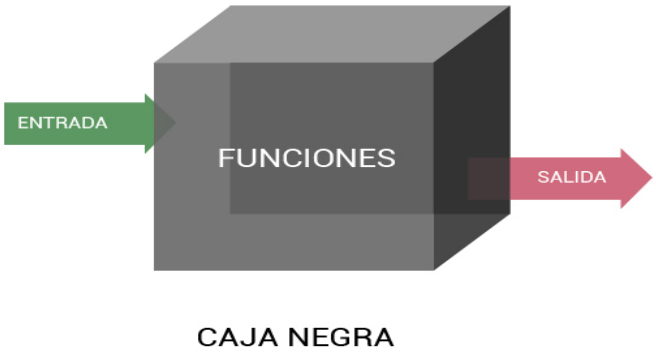


Fig. 31 Tipo de prueba caja negra

Debido a que las pruebas de aceptación requieren de la técnica de caja negra, esta técnica es seleccionada para validar las funcionalidades de la herramienta. En la cual se va a aplicar el método partición de equivalencia, ya que divide el dominio de entrada de un programa en clases de datos para probar todas las variantes posibles. Este método se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse.

3.4.3 Casos de pruebas basados en historias de usuarios

Los casos de prueba son un conjunto de condiciones o variables bajo las cuales se pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y se produzca un resultado correcto. Esta prueba permite ejercitar de forma completa todos los requisitos funcionales de un programa, es un enfoque complementario que intenta descubrir diferentes tipos de errores, por ejemplo: errores de interfaz y errores en estructuras de datos.

En el proceso de pruebas en cuestión se hace uso del método partición de equivalencia, de ahí el motivo por el que se diseña un caso de prueba por cada HU del sistema. El siguiente es un ejemplo del caso de prueba Enmascarar datos con la estrategia de anulación.

Tabla 15 Caso de Prueba Enmascarar datos con la estrategia anulación.

Escenario	Descripción	Base de Datos	Esquema	Tabla	Columna Anulación	Respuesta del sistema	Flujo central
EC 1.1 Enmascarar por anulación	En este escenario se configura la estrategia anulación	V Brookesia	V Public	V Entrevista	V SI	El sistema pone en rojo la letra del nombre del elemento configurado.	1. Seleccionar el elemento a configurar en el árbol jerárquico. 2. Seleccionar en la columna Anulación Verdadero o falso.
		V Brookesia	V Public	V Entrevista	V NO	El sistema pone en negro la letra del nombre del elemento desconfigurado.	1. Seleccionar el elemento a configurar en el árbol jerárquico. 2. Seleccionar en la columna Anulación Verdadero o falso.

Tabla 16Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Base de Datos	Campo de selección	No	Campo que contiene una lista de las bases de datos conectadas.
2	Esquema	Campo de selección	No	Campo que contiene una lista de los esquemas de la base de datos seleccionada.
3	Tabla	Campo de selección	No	Campo que contiene una lista de las tablas en el esquema seleccionado.
4	Anulación	Campo de selección	No	Campo de selección booleana.

3.4.4 Presentación de los resultados de las pruebas funcionales

Con el objetivo de validar la herramienta Brookesia se aplicaron los casos de pruebas a cada HU. Este proceso se realizó al finalizar cada iteración de implementación del sistema.

Primera iteración de desarrollo:

En esta iteración se identificaron dos no conformidades en dos iteraciones, las dos significativas y una recomendación, estas identificadas y solucionadas en la primera iteración de prueba, siendo comprobado en la segunda la inexistencia de no conformidades (Fig. 32).

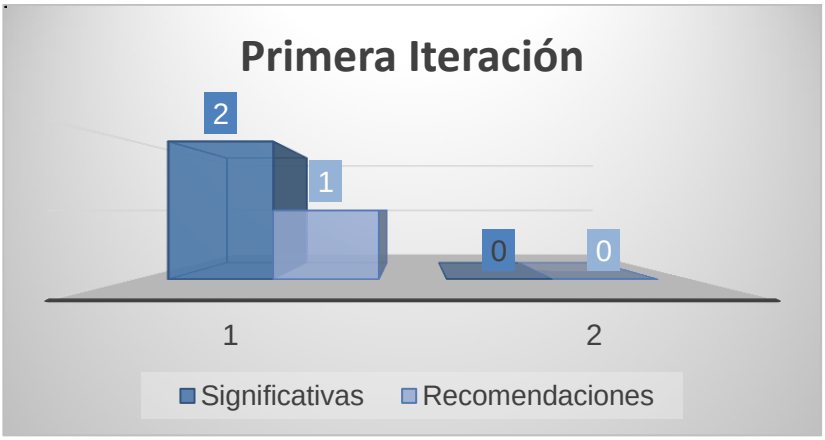


Fig. 32 Pruebas 1era iteración de desarrollo.

Segunda iteración de desarrollo:

En la validación realizada al finalizar la segunda iteración de desarrollo, se detectaron tres no conformidades en tres iteraciones, en la primera se identificaron dos no conformidades significativas; en la

segunda una no significativas, siendo comprobado en la tercera iteración que todas las deficiencias encontradas habían sido solucionadas y no existían no conformidades (Fig. 33).

Fig. 33Pruebas 2da iteración de desarrollo.

Tercera iteración de desarrollo:

Al finalizar la tercera iteración de desarrollo se identificaron en dos iteraciones de prueba dos no conformidades, ambas significativas e identificadas en la primera iteración, las cuales fueron solucionadas para la segunda iteración, no encontrando ninguna en esta (Fig. 34).

Fig. 34 Pruebas 3era iteración de desarrollo.

Cuarta iteración de desarrollo:

En la cuarta iteración de desarrollo se identificaron cuatro no conformidades en tres iteraciones de prueba, en la primera se identificaron dos no conformidades significativas, ninguna no significativa y no hubo recomendaciones; en la segunda se detectaron dos no conformidades, una significativa, una no

significativa y una recomendación; en la tercera iteración se pudo comprobar que estaban resueltas las no conformidades encontradas y no se detectó alguna nueva (Fig. 35).

Fig. 35Pruebas 4ta iteración de desarrollo.

Quinta iteración de desarrollo:

Una vez terminada la implementación, se realizaron las últimas pruebas al sistema, donde se identificaron un total de ocho no conformidades en tres iteraciones (Fig. 28), en la primera se identificaron cinco no conformidades, tres de ellas significativas, dos no significativas y una recomendación; en la segunda se identificaron dos significativas y una no significativas, siendo comprobado en la tercera iteración que se habían resuelto todos los problemas (Fig. 36).

Fig. 36 Pruebas 5ta iteración de desarrollo.

Conclusiones parciales

Se implementó el 100% de las 22 HU definidas en la fase de análisis. Se definió como estándar de codificación a emplear durante la implementación del sistema, las convenciones de código para el lenguaje de programación Java. Se utilizó para las pruebas la técnica de caja negra, con el método partición de equivalencia. Se realizaron las pruebas de aceptación a través del diseño de casos de pruebas; donde fueron identificadas 8 no conformidades en dos iteraciones, siendo estas solucionadas en una tercera, demostrando que la herramienta desarrollada cumple con las características especificadas por el cliente.

CONCLUSIONES GENERALES

La realización del presente trabajo posibilitó cumplir con los objetivos y tareas propuestas para su desarrollo, por lo que se concluye:

- ✓ Se implementó la herramienta de enmascarado de datos con las técnicas sustitución, variación de números y fechas, baraja, anulación de salida o eliminación y enmascarado de datos de salida, garantizando así la integridad y confidencialidad de las bases de datos en PostgreSQL, MySQL y SQL Server.
- ✓ Al realizar el análisis se obtuvieron 22 HU que permitieron definir todas las funcionalidades que requería el sistema.
- ✓ La utilización del patrón arquitectónico MVC y los patrones de diseño GRASP y GOF posibilitó hacer un diseño de la herramienta que permite la evolución y mantenimiento de la misma.
- ✓ Se validó la solución en tres iteraciones mediante los casos de prueba, uno por cada Historia de Usuario, demostrando que la aplicación cumple con las funcionalidades identificadas.

RECOMENDACIONES

- ✓ Adicionar nuevas técnicas de enmascarado de datos a Brookesia como: función hash, anagrama y ajuste proporcional inversa.
- ✓ Ampliar el enmascarado de datos a bases de datos no relacionales como MongoDB y CouchDB.
- ✓ Aumentar el soporte de Brookesia a otros gestores relacionales como SQLite y Oracle.

REFERENCIAS BIBLIOGRÁFICAS

1. **Camellea.***Gestión de Base de Datos con ADO.NET.* Ciudad de La Habana : Científico Técnica, 2004.

2. **J., Date C.***Introducción a los Sistemas de Bases de Datos.* 2001. Séptima Edición.

3. **Gómez Ballester, Eva, y otros.***Apuntes Bases de datos 1.* Dpto. de Lenguajes y Sistemas Informáticos Escuela Politécnica Superior Universidad de Alicante : s.n., 2007. pág. http://www.dlsi.ua.es/asignaturas/bd.

4. **Yera, Angel Cobo.** [En línea] http://books.google.com.cu/books?id=anCDr9N-kGsC&pg=PA7&dq=definicion+de+sistemas+gestores+de+base+de+datos&hl=es&ei=VgDcTMHyGovOngesrJQX&sa=X&o..

5. **Martínez, Rafael.** PosgreSQL-es -. [En línea] 2013. http://www.postgresql.org.es/sobre_postgresql.

6. **MySQL.** [En línea] 2013. http://www.mysql.com/why-mysql/white-papers/las-10-razones-principales-para-usar-mysql-como-base-de-datos-integrada/.

7. Definición ABC. [En línea] 2013. http://www.definicionabc.com/tecnologia/mysql.php#ixzz2ll9vdiwf.

8. Microsoft. [En línea] http://www.microsoft.com/es-xl/sqlserver.

9. Forrester Research. Forrester. [En línea] 2013. www.forrester.com.

10. Net 2000 Ltd. Data Masker. [En línea] 1998. http://www.datamasker.com/DataMasking_WhatYouNeedToKnow.pdf .

11. Techopedia. [En línea] 2010. http://www.techopedia.com/definition/13602/data-masking.

12. Linda Musthaler, Brian Musthaler. CIO. [En línea] 10 de 2009. http://www.cio.com/article/505843/Data_Masking_Secures_Sensitive_Data_in_Non_Production_Environments.

13. Help Net Security. [En línea] 25 de 8 de 2011. http://www.net-security.org/secworld.php?id=11512.

14. Slides Hare. [En línea] 2012. http://www.slideshare.net/sesa78/data-masking-14021097.

15. Cobb, Michael. Search Security. [En línea] 2010. http://searchsecurity.techtarget.com/tip/Unmasking-data-masking-techniques-in-the-enterprise.

16. Informatica. [En línea] 2008. http://www.informatica.com/es/products/data-masking/dynamic-data-masking/.

17. Orpheus GmbH. [En línea] 2014. http://www.data-masking-tool.com/features.

18. Informática Aplicada a la Gestión Pública. Facultad Derecho UMU. *Ingeniería del software. Metodologías de desarrollo.* [En línea] http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html.

19. Beck, K. *Extreme Programming Explained. Embrace Change. Pearson Education.* [trad.] Addison Wesley.

20. Ciencia y Técnica Administrativa. [En línea] 2008. http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm..

21. Visual Paradigm. [En línea] 2014. http://www.visual-paradigm.com/.

22. Meza, José María Dueñas. *Lenguajes de programación.* 2013.

23. Ruíz, Alberto. Observatorio Tecnológico. [En línea] 2009. http://recursostic.educacion.es/observatorio/web/eu/software/programacion/911-monografico-java.

24. ORACLE CORPORATION. *NetBeans IDE 7.3 Release Information.* [En línea] [Citado el: 27 de 11 de 2013.] http://netbeans.org/community/releases/73/.

25. Santisteban, Luis Angel Quintana. *Análisis y diseño de una herramienta para modelación de bases de datos.* La Habana, Cuba : s.n., 2008.

26. José H. Canós, Patricio Letelier y Mª Carmen Penadés. *Métodologías Ágiles en el Desarrollo de Software.* DSIC -Universidad Politécnica de Valencia, Camino de Vera : s.n. 46022 Valencia.

27. Delgado, Raidel Páez LLopiz. Ariel Labrada. *Herramienta para la creación de Sistemas de Información Geográfica.* La Habana, Cuba. : s.n., 2012.

28. Extreme Programing, Agile Process. Agile Software Development: A gentle introduction. [En línea] 2009. http://www.agile-process.org.

29. Molpeceres, Alberto. Diseño de software con patrones. [En línea] 01 de 07 de 2001. http://www.programacion.com/java/articulo/joa_patrones1/#joa_patrones1_software.

30. Mestras, Juan Pavón.*Estructura de las Aplicaciones Orientadas a Objetos, El patrón Modelo-Vista-Controlador (MVC), Programación Orientada a Objetos.* Dep. Ingeniería del Software e Inteligencia Artificial, Universidad Complutense Madrid : s.n.

31. Cooper, James W. *Java™ Design Patterns: A Tutorial.* [ed.] Addison Wesley. January 28, 2000. pág. 352. ISBN: 0-201-48539-7.

32. Mora, Roberto Canales. *Patrones de GRASP.* Madrid : s.n., 2003-2004.

33. Scott Hommel, Alberto Molpeceres. Convenciones de Código para el lenguaje de programación JAVA™ . *Sun Microsystems Inc.* [En línea] 2001 . http://www.javahispano.com.

34. Peña, Dr. Juan Manuel Fernández.*Pruebas de software: experiencias en su aplicación. Capítulo 6 Pruebas de sistema.* México : Facultad de Estadística e Informática Universidad Veracruzana, 2006.

35. *Ingeniría del software. Un enfoque práctico.*Pressman, Roger. Vol. 6ta edición.

36. Sommerville, Ian.*Ingeniería de Software.* [trad.] Antonio Botía Martínez, Francisco Mora Lizán, José Pascual Trigueros Jover Maria Isabel Alonso Galipienso. s.l. : Dpto Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante. Vol. Septima edición.

37. *PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA.* J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres. s.l. : Department de Lenguajes y Sistemas Informáticos, University of Sevilla.

38. Joskowicz, José. *Reglas y Prácticas en eXtreme Programming.* 2008.

BIBLIOGRAFÍA

1. **Camellea.***Gestión de Base de Datos con ADO.NET.* Ciudad de La Habana : Científico Técnica, 2004.

2. **J., Date C.***Introducción a los Sistemas de Bases de Datos.* 2001. Séptima Edición.

3. **Gómez Ballester, Eva, y otros.***Apuntes Bases de datos 1.* Dpto. de Lenguajes y Sistemas Informáticos Escuela Politécnica Superior Universidad de Alicante : s.n., 2007. pág. http://www.dlsi.ua.es/asignaturas/bd.

4. **Yera, Angel Cobo.** [En línea] http://books.google.com.cu/books?id=anCDr9N-kGsC&pg=PA7&dq=definicion+de+sistemas+gestores+de+base+de+datos&hl=es&ei=VgDcTMHyGovOngesrJQX&sa=X&o..

5. **Martínez, Rafael.** PosgreSQL-es -. [En línea] 2013. http://www.postgresql.org.es/sobre_postgresql.

6. **MySQL.** [En línea] 2013. http://www.mysql.com/why-mysql/white-papers/las-10-razones-principales-para-usar-mysql-como-base-de-datos-integrada/.

7. Definición ABC. [En línea] 2013. http://www.definicionabc.com/tecnologia/mysql.php#ixzz2ll9vdiwf.

8. Microsoft. [En línea] http://www.microsoft.com/es-xl/sqlserver.

9. Forrester Research. Forrester. [En línea] 2013. www.forrester.com.

10. Net 2000 Ltd. Data Masker. [En línea] 1998. http://www.datamasker.com/DataMasking_WhatYouNeedToKnow.pdf .

11. Techopedia. [En línea] 2010. http://www.techopedia.com/definition/13602/data-masking.

12. Linda Musthaler, Brian Musthaler. CIO. [En línea] 10 de 2009. http://www.cio.com/article/505843/Data_Masking_Secures_Sensitive_Data_in_Non_Production_Environments.

13. Help Net Security. [En línea] 25 de 8 de 2011. http://www.net-security.org/secworld.php?id=11512.

14. Slides Hare. [En línea] 2012. http://www.slideshare.net/sesa78/data-masking-14021097.

15. Cobb, Michael. Search Security. [En línea] 2010. http://searchsecurity.techtarget.com/tip/Unmasking-data-masking-techniques-in-the-enterprise.

16. Informatica. [En línea] 2008. http://www.informatica.com/es/products/data-masking/dynamic-data-masking/.

17. Orpheus GmbH. [En línea] 2014. http://www.data-masking-tool.com/features.

18. Informática Aplicada a la Gestión Pública. Facultad Derecho UMU. *Ingeniería del software. Metodologías de desarrollo.* [En línea] http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html.

19. Beck, K. *Extreme Programming Explained. Embrace Change. Pearson Education.* [trad.] Addison Wesley.

20. Ciencia y Técnica Administrativa. [En línea] 2008. http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm..

21. Visual Paradigm. [En línea] 2014. http://www.visual-paradigm.com/.

22. Meza, José María Dueñas. *Lenguajes de programación.* 2013.

23. Ruíz, Alberto. Observatorio Tecnológico. [En línea] 2009. http://recursostic.educacion.es/observatorio/web/eu/software/programacion/911-monografico-java.

24. ORACLE CORPORATION. *NetBeans IDE 7.3 Release Information.* [En línea] [Citado el: 27 de 11 de 2013.] http://netbeans.org/community/releases/73/.

25. Santisteban, Luis Angel Quintana. *Análisis y diseño de una herramienta para modelación de bases de datos.* La Habana, Cuba : s.n., 2008.

26. José H. Canós, Patricio Letelier y Mª Carmen Penadés. *Métodologías Ágiles en el Desarrollo de Software.* DSIC -Universidad Politécnica de Valencia, Camino de Vera : s.n. 46022 Valencia.

27. Delgado, Raidel Páez LLopiz. Ariel Labrada. *Herramienta para la creación de Sistemas de Información Geográfica.* La Habana, Cuba. : s.n., 2012.

28. Extreme Programing, Agile Process. Agile Software Development: A gentle introduction. [En línea] 2009. http://www.agile-process.org.

29. Molpeceres, Alberto. Diseño de software con patrones. [En línea] 01 de 07 de 2001. http://www.programacion.com/java/articulo/joa_patrones1/#joa_patrones1_software.

30. Mestras, Juan Pavón.*Estructura de las Aplicaciones Orientadas a Objetos, El patrón Modelo-Vista-Controlador (MVC), Programación Orientada a Objetos.* Dep. Ingeniería del Software e Inteligencia Artificial, Universidad Complutense Madrid : s.n.

31. Cooper, James W. *Java™ Design Patterns: A Tutorial.* [ed.] Addison Wesley. January 28, 2000. pág. 352. ISBN: 0-201-48539-7.

32. Mora, Roberto Canales. *Patrones de GRASP.* Madrid : s.n., 2003-2004.

33. Scott Hommel, Alberto Molpeceres. Convenciones de Código para el lenguaje de programación JAVA™ . *Sun Microsystems Inc.* [En línea] 2001 . http://www.javahispano.com.

34. Peña, Dr. Juan Manuel Fernández.*Pruebas de software: experiencias en su aplicación. Capítulo 6 Pruebas de sistema.* México : Facultad de Estadística e Informática Universidad Veracruzana, 2006.

35. *Ingeniría del software. Un enfoque práctico.*Pressman, Roger. Vol. 6ta edición.

36. Sommerville, Ian.*Ingeniería de Software.* [trad.] Antonio Botía Martínez, Francisco Mora Lizán, José Pascual Trigueros Jover Maria Isabel Alonso Galipienso. s.l. : Dpto Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante. Vol. Septima edición.

37. *PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA.* J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres. s.l. : Department de Lenguajes y Sistemas Informáticos, University of Sevilla.

38. Joskowicz, José. *Reglas y Prácticas en eXtreme Programming.* 2008.

39. Rafael Camps Paré, Luis Alberto Casillas Santillán, Dolors Costal Costa, Marc Gibert Ginestà, Carme Martín Escofet, Oscar Pérez Mora. *Software libre.* Barcelona : Universitat Oberta de Catalunya, 2005. ISBN: 84-9788-269-5.

40. *ORACLE ENTERPRISE MANAGER 10g.* **Corporation, Oracle.** 2007.

41. Solutions, Focus Business. La Protección de los Datos en los entornos. [En línea] 2009. www.focusbs.com.

42. Isaías Carrillo Pérez, Rodrigo Pérez González, Aureliano David Rodríguez Martín. METODOLOGIA DE DESARROLLO DEL SOFTWARE. [En línea] 2008.

43. PENROSE, ROGER. *La mente nueva del emperador. En torno a la cibernética, la mente y las leyes de la física.* MÉXICO : CONSEJO NACIONAL DE CIENCIA Y TECNOLOGIA. FONDO DE CULTURA ECONÓMICA, 2004. ISBN 0-19-851973-7.