Universidad de las Ciencias Informáticas
FACULTAD 6

Título: Monkript: Biblioteca para el cifrado y compresión de datos sensibles en MongoDB.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autores:
Ana Yesmín Serrano Escalona.
Enier Alarcón Barbán.

Tutores:
Ing. Marcos Luis Ortiz Valmaseda.
Ing. Frank Alberto Rodríguez Solana.

La Habana, junio del 2014.
Año 56 de la Revolución.

DECLARACIÓN DE AUTORÍA

Declaramos ser los autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Ana Yesmín Serrano Escalona

_____

Firma del autor

Enier Alarcón Barbán

_____

Firma del autor

Ing. Marcos Luis Ortiz Valmaseda

_____

Firma del tutor

Ing. Frank Alberto Rodríguez Solana

_____

Firma del tutor

DATOS DE CONTACTO

Autores:
Ana Yesmín Serrano Escalona
Universidad de las Ciencias Informáticas (UCI)
e-mail: ayserrano@estudiantes.uci.cu

Enier Alarcón Barbán
Universidad de las Ciencias Informáticas (UCI)
e-mail: ealarcon@estudiantes.uci.cu

Tutores:
Ing. Marcos Luis Ortiz Valmaseda
Universidad de las Ciencias Informáticas (UCI)
e-mail: mlortiz@uci.cu

Ing. Frank Alberto Rodríguez Solana
Universidad de las Ciencias Informáticas (UCI)
e-mail: frankalberto@uci.cu

AGRACEDIMIENTOS

A mis padres que me han acompañado desde mis primeros pasos y me han guiado durante mi vida al éxito y la felicidad. Por ser lo más grande que tengo en el mundo, por todos los valores que me han inculcado que han hecho de mí una mejor persona. Por soportar mis malacrianzas. Por enseñarme con su ejemplo y protegerme con su amor.

A mi familia: abuelas y abuelo; mis tías por ser para mí mis segundas madres, mis tíos; mis primos por ser mis hermanos y mi hermano por ser tan insoportable, gracias a todos por su apoyo incondicional durante estos 5 años de estudio y a lo largo de mi vida.

A Rainer por aparecer en el momento que más lo necesitaba, quedarse a mi lado en las buenas y malas, por su comprensión, paciencia, por compartir conmigo las noches de desvelos y por su apoyo sin límites, por ser la maravillosa persona que eres.

A las personas que me ha demostrado que hay lazos que unen mucho más que la sangre, por estar junto a mi desde el 1er día y durante estos 5 años de la carrera, por contribuir a convertirme en una mejor persona porque sin ustedes no lo hubiera logrado y la UCI no hubiese sido igual: Yadira(Negra) mi hermanita nunca cambies, Alfre(Mi titi) mi negrito eres el regalo más grande que me dio la universidad y Pedro(Piter) eres la persona de mejores sentimientos que he conocido.

A mi amigo y dúo de tesis, por ser el hombre que es, por ser mi justiciero y confidente al mismo tiempo, por su carácter irresistible, por soportarme. Por ser la persona excepcional que es.

A mis grandes amigos y amigas de siempre Liz Mavis, La Pote, Ana del Carmen, Julio, Fragah, Daniel y Maikel Zuñiga, por compartir conmigo los buenos y malos momentos de la universidad. Por convertirse en personas tan especiales para mí, brindarme su amistad sin esperar nada a cambio, gracias por haberlos conocido.

A mis madres en la UCI: Ivette mientras estuvo y Gladys. Por creerse el personaje, por su apoyo, sus regaños y ayuda en todo momento.

A mis hermanotes Bernardo y Yohander que para mí son como de sangre, por estar siempre pendientes y ocupados de mí, a Vladi que llegó de último pero siempre estuvo cuando lo necesité.

A los tutores Frank y Marcos porque aunque lejos siempre estuvo pendiente y preocupado por la evolución de este trabajo.

A las profes Yanelis, Luidmila, Arianna, Aislen y los profes Jorge Luis, Luis y Guerra por dejar una huella en mi vida para siempre, por ser ejemplos a seguir y porque como educadores nunca sabré donde termina su influencia sobre mí.

A Maylen, Ismel, José Lázaro, Larramendi y el Yoe por tantos buenos y malos momentos compartidos.

A Osmel y su familia por su apoyo y ayuda en los primeros años de mi carrera.

Al piquete de la cooperativa Arlas, Franco, Isnel, Celia, Deylert y Tan, a los “Trigueños” y Rosita.

A todos aquellos que de una forma u otra han contribuido con el desarrollo de este trabajo, mi carrera como profesional y a mí como persona.

Ana.

Sin dudas esta es la parte por la que quieres sentirte ingeniero, para en esa muestra de altruismo reflejar con lágrimas todos los que contribuyeron con esta meta que desde niño quise alcanzar:

(No hay grandes obras sin grandes amigos)

Agradecer:

A ti mami que siempre estuviste en esa fila sentada, para ti este título que seguro estoy estas más orgullosa que yo y sé que el sacrificio fue de los dos por igual, te amo.

A mi abuela, por su cariño, sus palabras de alientos, sus consejos prudentes con las asignaturas aun sin entender de qué le hablaba, hoy te digo que tu pinocho ya es ingeniero.

A mi hermanita, que esto le sirva de ejemplo y dentro de 4 años también nos regale su título, empínate.

Al ruso, ese otro papá. Y a mis tíos ingenieros Maykel, Frank y Alex por ser ejemplo y estar ahí para todos los problemas. A mis tías Leti, Xiomara y Deisy.

Para mamá el título y para mí un baúl lleno de amigos y recuerdos.

A David Rojas Pupo ese gran amigo con el que compartí tanto en tan poco tiempo.

A la familia Hernández-González por todo el cariño brindado en especial a Yulina otra hermana.

A Adisnelis Lorente, por sincerarse con la causa y toda la gratitud que mostró en Camagüey, a María Antonieta mi eterna novia, a Mavis Carcases mi sabia consejera en la UCI, a Jorge Enrique Hurtado gracias por tu experiencia.

A alguien que no está acostumbrada que la reconozcan, mi amiga y compañera de Tesis Ana Yesmin Serrano Escalona fuiste lo mejor de mis últimos dos años en la UCI.

A Liudmila o sencillamente “chupi” en nuestros duros tiempos de convivencia cuando había que planificarse. A Noraima y Lulu las amigas que quedaron.

A una de las personas más talentosas que he conocido, tanto así que a veces llega a inflar, pero que fue otro compañero de tesis en cada hora que se le dedicó a esta investigación Rainer Segura.

A Martica Acosta mi chiquitica cosa apapachable que siempre estuvo ahí y a Geito (Geonel) que se sumó sin hacer preguntas.

A Adriana Tomey la primera amiga que tuve en la UCI junto con Grether, Bermudes y Nuris.

A los colegas del grupo 9103 donde empecé a andar en esta universidad, especialmente para Pavel y Batista.

Al chino por las menticas y por crear espacios para quitar el estrés.

A Victor Manuel Peregrino, por la ayuda en el proyecto de compiladores, sin el cual nunca hubiese pasado de 2do año.

A José Carlos Pérez Zamora por soportarme todos mis trastornos, a Lionel mi hijo en esta escuela y al loco de Alcides por tenerme presente y la confianza que me brindo.

A Lorenzo por permitirme aprender de él y respetarme siempre, junto con Javier, Daniel Zaldívar y Jorge.

A la FEU por darme la oportunidad de ser una mejor persona y ese piquete del secretariado que fue en todo momento un grupo más: Carlos Arce por las charlas de las que siempre aprendí, Luis Manuel por el ejemplo de radicalización, Daniel A. Barrio por verme siempre como un amigo, a los tiempos de Janet, Liniusca , Camue, Julito, a Darlon el gruñon, a Gabriel el mejor divulgador del ejército libertador.

A Código y Letra por darme la oportunidad de acercarme al periodismo y todos los colegas de los medios.

A Froilan que siempre estuvo dispuesto a ayudar.

No se me olvida a los viejos y buenos amigos Chegui mi Orisel, la amiga del pre y de aquí.

A Luis Ángel Sosa Rivero que sabe que este fue el sueño de los dos.

A Gabriel David y Mayi por estar ahí en las buenas y en las malas.

A Mavis y Yadira (la negra) por compartirnos mutuamente los tormentos de la tesis.

Al piquete de la cooperativa.

A mis maestros María del Carmen Betancourt, Grisel Delgado, Margarita Céspedes por ser mi ejemplo y enseñarme que lo mejor que tiene el macho de nuestra especie es ser hombre.

A los tutores Marcos y Frank por el constante empeño en el perfeccionamiento de la investigación

A mis profesores Yoandris Quintana, Nara Lidia, mi queridísima Yanelis Benites, Yordanis Piñeiro por ser mis evangelios vivos.

A Jorge Luis y Gladis por las constantes revisiones a este trabajo

Al grupo 7 que siempre me valoró mucho, especialmente a Mariño, Julito, Thondi, Andrés y el feíto de Omar. También al epicentro de este grupo Ana del Carmen Espinosa Robert por ser tan especial y siempre estar ahí para escucharme.

Enier

DEDICATORIA

A mis padres porque a ellos se los debo.

A mi Papi-Chuli chiquitico, que te sirva de ejemplo y tú también lo puedas lograr.

A mis amigos Fragah y Daniel que por cuestiones de la vida no lo pudieron lograr, esta también es su tesis.

Ana Yesmín Serrano Escalona

A la memoria de Osvaldo Burgos Milán quien en estos días si la vida le hubiese permitido se estaría haciendo ingeniero en la facultad 4 de esta universidad. Cuando un amigo se va algo se muere en el alma, hoy le dedico este título y donde quiera que esté me sentiré orgulloso de haber sido su amigo. ¡Chulí Viejo!

A Eimy de la Caridad esa sobrina que llegó hace algunos meses que esta dedicatoria le sirva de inspiración para que ella también lo logre.

Al dúo musical Buena Fe por enseñarme que la Universidad es un estado del alma.

Enier Alarcón Barbán

RESUMEN

El desarrollo de las tecnologías de la informática y las comunicaciones ha provocado un enorme crecimiento de información. Lo cual ha generado la necesidad de mejorar las tecnologías existentes, percibiéndose una tendencia al uso de las bases de datos no relacionales, entre ellas MongoDB. La seguridad que brinda MongoDB para tratar información sensible es muy vulnerable. El presente trabajo se planteó como objetivo principal desarrollar una biblioteca para el cifrado y la compresión de información en MongoDB. Para lograrlo se analizaron los fundamentos teóricos del cifrado y la compresión de datos, teniendo en cuenta las herramientas, tecnologías y metodologías apropiadas para su desarrollo. Se obtuvo como resultado una biblioteca con una arquitectura sólida diseñada sobre la base de los patrones GRASP y validada con diferentes técnicas de pruebas de software.

PALABRAS CLAVE: bases de datos no relacionales, biblioteca, cifrado, compresión, MongoDB.

ABSTRACT

The development of information technologies and communications has led to tremendous growth of information. This has generated the need to improve existing technologies, perceiving a trend to the use of non-relational databases, including MongoDB. The security provided to treat sensitive information MongoDB is very vulnerable. This paper was presented as main objective to develop a library for encryption and compression of information in MongoDB. To achieve the theoretical foundations of encryption and compression of data were analyzed, taking into account the tools, technologies and methodologies appropriate for their development. The result was a library with a solid architecture designed based on the GRASP patterns, validated with different software testing techniques.

KEYWORDS: non-relational database, library, encryption, compression, MongoDB.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

Desde el surgimiento de la humanidad el hombre ha tenido la necesidad de almacenar la información, recopilando datos en soportes que han ido evolucionando desde piedras, maderas, papel, cintas magnéticas, hasta discos. Estos datos por su importancia y utilidad debían ser administrados de forma responsable y eficaz. Aparejado con el surgimiento de las primeras computadoras y la necesidad de almacenar grandes volúmenes de información o datos surge el uso de las bases de datos.

En la actualidad, en esta sociedad de la “era digital”, su uso se hace indispensable donde su utilidad se debe no sólo al almacenamiento de datos de una forma determinada sino a que en una base de datos también existen una cantidad de elementos que ayudan a organizar, relacionar, proteger, y administrar de manera sistemática y eficiente los datos. Con la utilización de las nuevas Tecnologías de la Informática y las Comunicaciones (TIC) casi toda colección de información puede convertirse en una base de datos constituyendo el fundamento a los sistemas de información y el soporte para la gestión de los datos y la toma de decisiones. Surge entonces en el mundo un gran uso de estas encausado por las necesidades competitivas de las empresas.

Las bases de datos facilitan no sólo el registro de gran cantidad de datos sino también el acceso a los mismos, lo cual permite ahorrar tanto espacio físico como tiempo al momento de consultar información contenida en ellas. (1) Una de las más usadas son las relacionales o SQL, aunque con los adelantos tecnológicos en la web, el software como servicio y los servicios en la nube, se ha hecho compleja la escalabilidad por lo que aparecen las bases de datos no relacionales o noSQL. Estas últimas intentan resolver el problema proponiendo una estructura de almacenamiento más versátil.

Las empresas cubanas apuestan por la informatización de sus procesos y desembocan en la necesidad de recopilar y almacenar su información para un oportuno análisis de la misma. La aplicación de medios electrónicos e informáticos ha revolucionado la gestión de estas empresas, lo cual explica las razones que progresivamente obligan a que las organizaciones desarrollen las tecnologías de bases de datos. El crecimiento de la información, la complejidad de los mercados, la heterogeneidad de los consumidores y las múltiples combinaciones producto-mercado son sólo algunas de las preocupaciones de estas empresas. (2)

La Universidad de las Ciencias Informáticas (UCI) como soporte de la industria cubana de informática usa esta tecnología en variables que apuesten por el desarrollo sostenible del país y potencien el desarrollo del software libre. A pesar de que todos sus centros de producción emplean las bases de datos ya que el uso de estas está estrechamente vinculado a la informática, es específicamente el Centro de Tecnología de Gestión de Datos (DATEC), el que se dedica al estudio y análisis de las tecnologías de bases de datos sosteniendo estrechas colaboraciones con entidades, organizaciones y empresas así como comunidades internacionales de desarrollo. La arquitectura organizacional de este centro cuenta con el Departamento de PostgreSQL creado por la necesidad de dominar las tecnologías existentes para que sean capaces de almacenar, procesar y analizar grandes volúmenes de datos.

El constante desarrollo tecnológico en el mundo de las Tecnologías de la Informática y las Comunicaciones, ha provocado un enorme crecimiento en el tamaño de información generado. Lo cual ha creado la necesidad de mejorar las tecnologías existentes percibiéndose una tendencia en el uso de bases de datos no relacionales para gestionar la información de una manera rápida y eficiente con el menor costo posible. Dentro de los sistemas de bases de datos no relacionales, o noSQL, se encuentran entre las más usadas MongoDB, HBase, CouchDB, Redis, Apache Cassandra, Neo4J, entre otras.

MongoDB es un sistema de bases de datos orientado a documentos. Este sistema no guarda los datos en tablas como en las bases de datos relacionales, en cambio los almacena en estructuras de datos en documentos de tipo JSON con un esquema dinámico, al cual denominan BSON, haciendo que los datos sean más fáciles de integrar.

El cifrado de datos en MongoDB para tratar información sensible, puede ocurrir en múltiples niveles. Los dos principales niveles son:

1. “datos-en-trasmisión”: la información es protegida cifrándola con SSL/TSL mientras se trasmite por la red.

2. “datos-almacenado”: se protege cifrando la información ya almacenada.

En este último nivel se puede cifrar la información utilizando alguna de las vías siguientes:

- Cifrando todo el disco.

- Cifrando archivos o bases de datos individuales en el disco.

- Cifrando todo el documento (filas en SQL) o los atributos individuales (columnas en SQL).

Las vías anteriormente mencionadas tienen sus inconvenientes: por ejemplo durante el proceso de cifrado del disco los datos son descifrados luego de extraerse del disco lo que provoca que se manejen datos en texto plano y los mismos quedan reflejados en los logs del sistema por lo que, tanto estos como las copias de respaldo tienen que implementar su propio sistema de cifrado. Similar ocurre con las restantes vías, sumado a que en el cifrado de una base de datos se corre el riesgo de que si ocurre un fallo en el proceso de cifrado puede perderse información valiosa ya que estos sistemas de bases de datos son capaces de almacenar TBs de información con un crecimiento exponencial. Se hace necesario que la información nunca se trasmita ni se almacene en texto plano para evitar que quede registrada en los logs del sistema y pueda ser utilizada por potenciales atacantes.

Por lo anterior expuesto se identifica el siguiente **problema de la investigación:** ¿Cómo proteger y disminuir el tamaño de información en MongoDB?

En correspondencia con el problema de la investigación señalado se define como **objeto de estudio:** el cifrado y compresión de información.

El objeto delimita el **campo de acción:** el cifrado y compresión de información en MongoDB.

Para dar solución al problema planteado se trazó como **objetivo general**: desarrollar una biblioteca para el cifrado y compresión de información en MongoDB.

Objetivos específicos:

- Definir el algoritmo de cifrado a utilizar.

- Definir el algoritmo de compresión a utilizar.

- Implementar la biblioteca Monkript.

- Validar la biblioteca Monkript.

Tareas de la investigación:

- Caracterización de los algoritmos de cifrado.
- Caracterización de los algoritmos de compresión.
- Caracterización de las técnicas y metodología a utilizar en el desarrollo de la investigación.
- Selección de las tecnologías a utilizar en el desarrollo de Monkript.
- Desarrollo de los artefactos de la metodología utilizada en la construcción de Monkript.
- Implementación de Monkript.
- Definición de los casos de pruebas para Monkript.
- Validación de la implementación en base a las pruebas realizadas.

Posibles resultados: biblioteca para el cifrado y compresión de información en MongoDB.

Para el desarrollo de la presente investigación se tuvo en cuenta los siguientes métodos **de la investigación científica:**

Métodos teóricos

Analítico – sintético: Utilizado para estudiar y analizar documentos, libros, artículos y otras fuentes bibliográficas de diferentes autores para poder realizar una amplia investigación sobre las características de los principales algoritmos de cifrado y compresión, así como las tendencias actuales de los gestores de bases de datos noSQL.

Modelación: Empleado para crear el proceso de diseño mediante la abstracción de sus elementos fundamentales utilizando un lenguaje de modelado y así desarrollar un modelo para la biblioteca a desarrollar a partir de la situación problemática.

Análisis Documental: Mediante este método se hizo un estudio de gran cantidad de documentación referente a la seguridad de MongoDB así como de la necesidad de compresión de los datos en el

mercado actual para reducir los requisitos de almacenamiento. El uso de este método posibilitó además el estudio de las tecnologías de desarrollo para implementar la biblioteca y la experiencia de algunos programadores en el tema.

Capítulo 1: Fundamentos teóricos de la investigación, metodología y tecnologías empleadas.

Introducción

En el presente capítulo se abordan conceptos asociados al cifrado y compresión de datos de distintos autores. Se analizan productos de software existentes a nivel mundial que dan solución a problemas similares. Se relacionan además un conjunto de particularidades de las tecnologías, metodología y herramientas a utilizar.

1.1 Conceptos asociados al dominio del problema

Para un mejor entendimiento del problema a continuación se definen los conceptos fundamentales a tener presentes en el desarrollo de una biblioteca de cifrado y compresión de datos. Partiendo de la ciencia que se dedica a cifrar y proteger la información.

Criptografía:

Rama inicial de las Matemáticas y en la actualidad de la Informática y la Telemática, que hace uso de métodos y técnicas con el objeto principal de cifrar y proteger un mensaje o archivo por medio de un algoritmo, usando una o más claves. Esto da lugar a diferentes tipos de sistemas de cifrado que permiten asegurar estos cuatro aspectos de la seguridad informática: la confidencialidad, la integridad, la disponibilidad y el no repudio de emisor y receptor. (3)

Emisor:

Aquél que transmite la información (un individuo, un grupo o una máquina). (4)

Receptor:

Aquél, que recibe la información (un individuo, un grupo o una máquina). (4)

Cifrado:

Técnica por la que la información se hace ilegible para terceras personas. Para poder acceder a ella es necesaria una clave que sólo conocen el emisor y/o el receptor. Se usa para evitar el robo de información sensible. (5)

Cifrado Simétrico:

Técnica que permite que la información sea cifrada y descifrada utilizando la misma clave de cifrado en el extremo de la conexión que cifra la comunicación y en el extremo en el que se descifra. La simetría se refiere a que las partes tienen la misma llave tanto para cifrar como para descifrar. (6)

Cifrado Asimétrico:

Técnica que permite que la información sea cifrada y descifrada sin tener que entregar llaves de seguridad. Utiliza como elemento necesario una clave diferente en el extremo que cifra de la que utiliza el extremo que descifra, en esto consiste la asimetría. (6)

Una cadena de texto que se cifra genera una cadena de texto de mayor longitud. Por lo que es conveniente sumar el proceso de compresión una vez cifrada la información.

Compresión:

Es la acción y efecto de comprimir. La compresión de datos consiste en reducir el tamaño de un archivo digital a partir de una reducción del tamaño de los datos, lo que se traduce en que el archivo en cuestión, tras la compresión, ocupará menos espacio. (7)

1.2 Algoritmos de cifrado

Criptografía simétrica

Los algoritmos simétricos pueden ser divididos en Cifrado de Flujo y Cifrado de Bloques. El cifrado por flujo cifra un texto plano bit a bit, mientras que el cifrado por bloques toma un número de bits (generalmente 64 bit en cifrados modernos), y lo cifran como una unidad simple. La seguridad en clave simétrica reside en la propia clave secreta, y por tanto el principal problema es la distribución de esta clave a los distintos usuarios para cifrar y descifrar la información. La misión del emisor y receptor es mantener la clave en secreto. Si cae en manos equivocadas ya no se podría considerar que la comunicación sea

segura y se debe generar una nueva clave. Otro problema reside en que las claves secretas a guardar son proporcionales al número de canales seguros que se desea mantener. Esto no es un problema en sí, pero se debe administrar bien las llaves. (8) La principal ventaja de los algoritmos simétricos es la velocidad y son muy usados para el cifrado de grandes cantidades de datos.

Criptografía asimétrica

Se trata de criptosistemas más modernos y complejos que los simétricos. A pesar de ser computacionalmente mucho más complejos, son el estándar hoy en día, utilizados en sistemas que combinan cifrado asimétrico y simétrico (Pretty Good Privacy). La criptografía asimétrica está basada en la utilización de números primos muy grandes. Si multiplicamos entre sí dos números primos muy grandes, el resultado obtenido no puede descomponerse eficazmente, utilizando los métodos aritméticos más avanzados en los ordenadores más avanzados sería necesario utilizar durante miles de millones de años tantos ordenadores como átomos existen en el universo. El proceso será más seguro cuanto mayor sea el tamaño de los números primos utilizados. El problema de las claves asimétricas es que cuando el texto a tratar es largo el proceso de codificación es muy lento.

Se decide utilizar un algoritmo simétrico para el cifrado de los datos teniendo en cuenta que son más sencillos de implementar. Los asimétricos pueden llegar a ser 1000 veces más lentos, tanto que no pueden ser utilizados para cifrar grandes cantidades de información. Si analizamos un cifrado asimétrico con uno simétrico, se nota que utiliza pocas operaciones y estas operaciones son generalmente lineales, por lo que se computan rápidamente.

1.2.1 Algoritmos de cifrado simétrico

Data Encryption Standard (DES)

Utiliza bloques de 64 bits, los cuales codifica empleando claves de 56 bits y aplicando permutaciones a nivel de bit en diferentes momentos (mediante tablas de permutaciones y operaciones XOR). Es una red de Feistel [1] de 16 rondas, más dos permutaciones, una que se aplica al principio y otra al final. La flexibilidad de DES reside en que el mismo algoritmo puede ser utilizado tanto para cifrar como para descifrar, simplemente invirtiendo el orden de las 16 subclaves obtenidas a partir de la clave de cifrado. La

[1] Método de cifrado en bloque con una estructura particular. Debe su nombre al criptógrafo Horst Feistel.

empresa española sin fines de lucro llamado Electronic Frontier Foundation (EFF) construyó en Enero de 1999 una máquina capaz de probar las 2^{56} claves posibles en DES y romperlo sólo en tres días con fuerza bruta.

Triple DES

Se basa en aplicar el algoritmo DES tres veces, la clave tiene una longitud de 128 bits. Si se cifra el mismo bloque de datos dos veces con dos llaves diferentes (de 64 bits), aumenta el tamaño de la clave. Parte de una llave de 128 bits, que es divida en dos llaves, A y B. Al recibir los datos, aplicamos el algoritmo DES con la llave A, a continuación se repite con la llave B y luego otra vez con la llave A (de nuevo). 3DES aumenta de forma significativa la seguridad del sistema de DES, pero requiere más recursos del ordenador.

International Data Encryption Algorithm (IDEA)

Es un algoritmo de cifrado por bloques de 64 bits iterativo. La clave es de 128 bits. El cifrado precisa 8 rotaciones complejas. El algoritmo funciona de la misma forma para cifrar que para descifrar. El algoritmo es fácil de implementar en hardware y software, aunque algunas de las operaciones que realiza no son eficientes en software, por lo que su eficiencia es similar a la del DES. La única debilidad conocida es un conjunto de 251 claves débiles, pero dado que el algoritmo tiene 2^{128} claves posibles no se considera un problema serio.

Advanced Encryption Standard (AES)

Este cifrado puede implementar tanto en sistemas hardware como en software. Opera con bloques y claves de longitudes variable de 128bits, de 192 bits y de 256 bits. El resultado intermedio del cifrado constituye una matriz de bytes de cuatro filas por cuatro columnas. A esta matriz se le vuelve a aplicar una serie de bucles de cifrado basado en operaciones matemáticas (sustituciones no lineales de bytes, desplazamiento de filas de la matriz, combinaciones de las columnas mediante multiplicaciones lógicas y sumas XOR en base a claves intermedias). (9)

Algunos criptógrafos muestran preocupación sobre la seguridad del AES. El margen entre el número de rondas especificado en el codificador y los mejores ataques conocidos es muy pequeño. Otra

preocupación es la estructura de AES. A diferencia de la mayoría de cifradores de bloques, tiene una descripción matemática muy ordenada.

Blowfish

Es un algoritmo de cifrado rápido y fuerte. Su creador es Bruce Schneier, uno de los más prestigiosos criptógrafos en el mundo. Es una red de Feistel, que itera una función de cifrado simple 16 veces. El tamaño del bloque es de 64 bits, y la clave puede ser cualquier longitud de hasta 448 bits. Es sólo apto para aplicaciones en las que la clave no cambia a menudo, como un enlace de comunicaciones o un codificador de archivos automático. Es significativamente más rápido que DES cuando se implementa en microprocesadores de 32 bits con grandes cachés de datos. (10)

Luego de un análisis de los algoritmos simétricos más usados se decidió escoger como algoritmo de cifrado a Blowfish teniendo en cuenta que es fácil de implementar y más rápido que los algoritmos anteriormente descritos, con licencia de software libre y uno de los más seguros pues su seguridad no se ha roto nunca y puede correr en 5K de memoria. Es de los algoritmos más usados actualmente para el cifrado de archivos por las aplicaciones existentes. Cuenta con una fuerte aceptación como algoritmo de cifrado.

1.3 Algoritmos de compresión

En los algoritmos de compresión, la compresión consiste en tomar una trama de símbolos y transformarlos en códigos/claves. Si ésta es eficiente, las claves resultantes ocuparán menor espacio que los símbolos originales. La decisión de obtener una codificación a partir de ciertos símbolos (o conjunto de ellos) está basada en un modelo. El modelo es simplemente una colección de datos y reglas usados para procesar a la entrada símbolos y determinar su correspondiente codificación a la salida. Dentro de las técnicas de compresión de datos, y atendiendo a la reversibilidad de la información original, hay dos grandes familias:

- Técnicas de compresión "lowless" o sin pérdida (para datos en los que es imprescindible que no se pierda nada de información, como registros de bases de datos, ficheros ejecutables, hojas de cálculo, etc.).

- Técnicas de compresión "lossy" o con pérdida (para datos en los que se permite cierta pérdida de información "sin que se note demasiado", como por ejemplo en ficheros en MP3, imágenes en JPEG, PNG u otros. Aquí una pequeña disminución en la calidad final no se nota demasiado, pero influye muy positivamente en la reducción del tamaño del fichero).

1.3.1 Algoritmos de compresión sin pérdida de datos

Los algoritmos de compresión sin pérdidas son procedimientos de compresión para archivos informáticos, son los más eficientes para el trabajo con datos dentro de los cuales encontramos dos grandes grupos:

- **Algoritmos estadísticos en los que se encuentran:**

Shannon-Fano: debe su nombre a sus creadores Claude Shannon y Robert Fano, fue el primer algoritmo para construir un conjunto de los mejores códigos de tamaño variable. Partimos de un conjunto de n símbolos con probabilidades conocidas (o frecuencias) de ocurrencia. Los símbolos se organizan por primera vez en orden descendente de sus probabilidades. El conjunto de símbolos se divide entonces en dos subconjuntos que tienen la mismas (o prácticamente las mismas) probabilidades. Todos los símbolos en un subconjunto consiguen códigos que comienzan con un 0 asignados, mientras que los códigos de los símbolos en el otro subconjunto empiezan con un 1. Cada subconjunto de forma recursiva divide en dos subconjuntos de probabilidades aproximadamente iguales, y el segundo bit de todos los códigos se determina de una manera similar. Cuando un subconjunto contiene sólo dos símbolos, sus códigos se distinguen por la adición de un poco más de cada uno. El proceso continúa hasta que no hay más subconjuntos. (11)

Huffman: es similar al método de Shannon-Fano. Por lo general produce mejores códigos, al igual que el método de Shannon-Fano, produce el mejor código cuando las probabilidades de los símbolos son potencias negativas de 2. La principal diferencia entre los dos métodos es que Shannon-Fano construye sus códigos de arriba a abajo (desde el más a la izquierda de los bits de la derecha), mientras que Huffman construye un árbol de código de abajo hacia arriba (construye los códigos de derecha a izquierda). (11)

- **Algoritmos basados en diccionario en los que se encuentran:**

RLE (**Run-length encoding):** es una forma muy simple de compresión de datos en la que secuencias de datos consecutivas con el mismo valor son almacenadas como un único valor más su recuento. Demuestra gran eficiencia cuando hay un alto número de repeticiones consecutivas de un determinado byte. La unidad básica serían dos bytes, el primero indica el número de veces que se repite el segundo. (11)

En este grupo se encuentran también la familia de algoritmos LZ que se caracterizan por ser sencillos, rápidos, no necesitan entrenamiento previo y consumen pocos recursos ejemplo de alguno de ellos son:

LZ77: mantiene un registro de los últimos caracteres procesados de la entrada. En cada momento, el algoritmo se encuentra procesando en un punto de la entrada los "n" caracteres anteriores que forman la "historia" del algoritmo o "ventana". Los caracteres posteriores al punto actual forman el llamado "buffer de adelantamiento". En cada paso, la cadena que comienza en el punto actual de la entrada se busca hacia atrás en la "historia". Si se encuentra una coincidencia que sea lo suficientemente larga como para tenerla en cuenta, a la salida se sustituye la cadena coincidente por un par que indica el "desplazamiento hacia atrás" y la "longitud" de la coincidencia con la cadena hacia atrás. Como los pares "desplazamiento, longitud" ocupan menos que la cadena que coincidió, se obtiene compresión. Si no se encuentra una coincidencia, la salida es una copia literal de la entrada. Posteriormente se avanza la ventana (es decir, se avanza en la entrada) bien de longitud si hubo coincidencia, bien un carácter si no la hubo. (11)

LZ78: usa un diccionario pre-inicializado que se construye sobre la base de la secuencia de datos de entrada. Logra la compresión mediante la sustitución de las ocurrencias repetidas de datos con referencias al diccionario. Cada entrada de diccionario es de la forma diccionario = {índice, carácter}, en donde el índice es el índice a una entrada de diccionario anterior, y el carácter se añade a la cadena representada por diccionario. (11)

Snappy: la codificación es orientada a bytes (bytes enteros sólo se emiten o se consumen de una secuencia). El formato no utiliza ningún codificador de entropía, como árbol de Huffman o codificador aritmético. Los primeros bytes de la secuencia son la longitud de los datos no comprimidos almacenados, que permite la codificación de longitud variable. Los siete bits más bajos de cada byte se utilizan para los datos y el primer bit es una bandera que indica si el siguiente byte se utiliza para el mismo entero. (12)

Los bytes restantes de la corriente se codifican utilizando uno de los cuatro tipos de elementos. El tipo de elemento se codifica en el primer byte (byte de etiqueta) del elemento. Los dos bits más bajos de este byte es el código de tipo:

00 - literal - datos sin comprimir; los 6 bits superiores se utilizan para la longitud del almacén de datos, y si la longitud de los datos es más de 60 bytes, se añade la codificación de longitud variable adicional.

01 - Copia con la longitud almacenada como 3 bits y desplazamiento almacenados como 11 bits.

10 - Copia de longitud almacenada como 6 bits de byte etiqueta.

11 - Copia de longitud almacenada como 6 bits de byte etiqueta.

La copia se refiere al diccionario (sólo datos descomprimidos). El desplazamiento es el cambio de la posición actual a la secuencia ya descomprimida. La longitud es el número de bytes a copiar del diccionario.

Luego de un análisis de los algoritmos de compresión más usados actualmente se decide utilizar Snappy teniendo en cuenta: su rapidez, velocidad de compresión de 250 MB / segundo; estabilidad y fiabilidad, ha comprimido y descomprimido PBs de datos en el entorno de producción de Google. El formato de flujo de bits es estable y no va a cambiar entre versiones. También es robusto y es de código abierto.

1.4 Bibliotecas de cifrado y compresión existentes.

Molebox: es capaz de comprimir el tamaño de tu aplicación y protegerla, cifrar ficheros individuales y por lotes. Permite crear un archivo ejecutable que contenga toda la información que se desea comprimir o proteger de la intromisión de terceros. (13) Comprime tanto a la aplicación como a las bibliotecas necesarias para su ejecución brindando la posibilidad de acceder a la información sin necesidad de descomprimirla. Es de gran utilidad para las aplicaciones de Windows. (14)

Xbundler: es un especial de plug-in para Themida y WinLicense que permite a bibliotecas de vínculos dinámicos (DLL Win32 y .NET DLL) y archivos de datos a ser embebido dentro de una aplicación protegida, lo que simplifica la distribución de la aplicación a sus clientes. Comprime y cifra todos los

archivos incrustados sin afectar a la capacidad de la aplicación para que funcione correctamente, y no requiere codificación adicional de su parte. (15)

SharpZipLib: es una biblioteca de clases para la plataforma .NET que permite trabajar con formatos de compresión de archivos (ZIP, GZIP, Tar y BZip2). El componente es completamente gratuito y es software libre distribuido bajo licencia GPL. Se encuentra programada en C# e incluye el código fuente. Con dicha biblioteca se puede comprimir y descomprimir archivos desde cualquier lenguaje soportado por la plataforma .NET de Microsoft, como páginas desarrolladas en ASP.NET, C#, VB.NET compatible para versiones de Windows. (16)

Una vez concluido el estudio de sistemas homólogos existentes se puede afirmar que estos dan solución a la seguridad y reducción del tamaño de archivos y lo hacen de forma eficiente, pero son bibliotecas compatibles únicamente con las tecnologías .NET, sin permitir que puedan funcionar en cualquier ordenador independientemente del sistema operativo que utilice y que puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales, por lo que no están disponibles para sistemas como MAC o las distribuciones GNU/Linux[2] . Ninguna de estas soluciones satisfacen la necesidad de cifrar la información y reducir el tamaño de la misma para los sistemas gestores de base de datos NoSQL.

En la búsqueda realizada no se encontraron sistemas que solucionen el problema de la investigación por lo que es necesario comenzar a desarrollar la biblioteca propuesta como un sistema con calidad. Para su desarrollo es necesario contar con una guía que describa el proceso de elaboración del software.

1.5 Metodologías de Desarrollo

En la actualidad el software de computadora es la tecnología individual más importante en ámbito mundial. A medida que la importancia del software ha crecido, la comunidad del software ha intentado de manera continua desarrollar tecnologías que hagan más fácil, más rápida y menos cara la construcción y el mantenimiento de programas de computadora de alta calidad. Para esto surge la ingeniería que permite realizar un producto de calidad utilizando herramientas, métodos y procesos adecuados. (17)

[2] GNU/Linux es uno de los términos empleados para referirse a la combinación del núcleo o kernel libre similar a Unix denominado Linux.

Las metodologías de desarrollo de software, son específicamente, quienes ayudan a los desarrolladores a partir de un conjunto de técnicas, procedimientos y herramientas a elaborar el nuevo producto. Detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla, o sea, describe paso a paso el proceso de conformación del producto informático deseado. (18)

Existen aquellas metodologías tradicionales que se centran especialmente en el control del proceso, cumpliendo con un detallado plan y documentando exhaustivamente todo el proyecto, entre estas se encuentra el Proceso Unificado de Racional (RUP), adaptable para proyectos de largo plazo; Microsoft Solution Framework (MSF), que se adapta a proyectos de cualquier dimensión y tecnología. Por otra parte, aparecen las metodologías ágiles, que dan mayor valor al individuo, el cliente colabora en todo momento, se basa en desarrollar un software incremental, sencillo y adaptable. Destacándose Programación Extrema (XP) la cual se recomienda para proyectos de corto plazo y Open Up que se centra en articular la arquitectura para facilitar la colaboración técnica.

1.5.1 Metodología Programación Extrema (XP)

XP básicamente plantea que se trabaje sobre la base de la colaboración mutua con el cliente y donde no exista más documentación que el propio código. Este proceso define cuatro actividades del marco de trabajo: planeación, diseño, codificación y prueba

- **Planeación** donde se definen las características y funcionalidades que tendrá el software, en un documento llamado historias, que es creado por el cliente el cual le da valor de prioridad a las tareas. Los miembros del equipo entonces evalúan cada historia y le asignan un costo, el cual se mide en semanas de desarrollo, si la historia requiere de mucho tiempo se le pide al cliente que la divida en dos o más, de forma que quede más pequeña y se vuelve a definir el costo. Una vez que se fija el compromiso básico y la fecha de entrega de las primeras historias, el equipo ordena las historias en el orden se desarrollaran.
- **Diseño** que no es más que una guía de implementación para cada una de las historia. Este diseño se caracteriza por ser simple, es un artefacto que puede ser modificado. También se usarán las tarjetas CRC (Clase – Responsabilidad - Colaboración) que permite una mejor organización del diseño en este tipo de proyecto.

– **Codificación** antes de comenzar a codificar se recomienda realizar pruebas de unidad. Posibilitando que en vez de que los programadores desarrollen cada uno en su propio estilo lo hagan sobre uno solo, el que está definido por la metodología, logrando uniformidad y organización. Otro concepto clave de la XP es la programación en parejas.

– **Pruebas** es la última actividad y existen varios tipos de ellas, las mencionadas anteriormente como pruebas de unidad, las pruebas de aceptación que son las que el cliente específica y se centran en funcionalidades y características que el sistema debe tener. Las pruebas se van realizando constantemente lo que ayuda al equipo de XP a ir perfeccionando detalles. (19) Un proyecto XP ocurre como se muestra en la figura 1.

Fig. 1 Proyecto XP

1.5.2 Metodología Proceso Unificado Abierto (Open Up)

La metodología Open UP -en inglés: Open Unified Process- es un proceso unificado que aplica propuestas de gestión ágil, tratando de ser manejable en relación con RUP pues mantiene sus características esenciales: centrado en la arquitectura, dirigido por Casos de Uso (CU) y un desarrollo iterativo e incremental, dentro del ciclo de vida de un proyecto de software. No provee lineamientos para todos los elementos que se manejan en un proyecto, pero contiene el conjunto mínimo de práctica que ayuda a los equipos de desarrollo a ser más eficientes. Es un proceso iterativo que es mínimo, completo y extensible que puede utilizarse tal cual o ampliarse para tratar una amplia variedad de tipos de proyecto. Esta metodología excluye la mayoría de las piezas opcionales de RUP, y muchos artefactos han sido fusionados. Está organizada dentro de cuatro áreas principales de contenido: Comunicación-Colaboración, Intención, Solución y Administración.

Entre sus principios básicos se encuentra la colaboración para alinear los intereses y desarrollar buenas prácticas colaborativas que generen un buen ambiente de trabajo, el enfoque para reducir riesgos y articular la arquitectura, el balance para confortar las prioridades (necesidades y costo técnico) y la evolución para dividir el proyecto en iteraciones cortas obteniendo retroalimentación temprana.
Las fases del ciclo de vida de Open Up son las que se muestran en la figura 2:

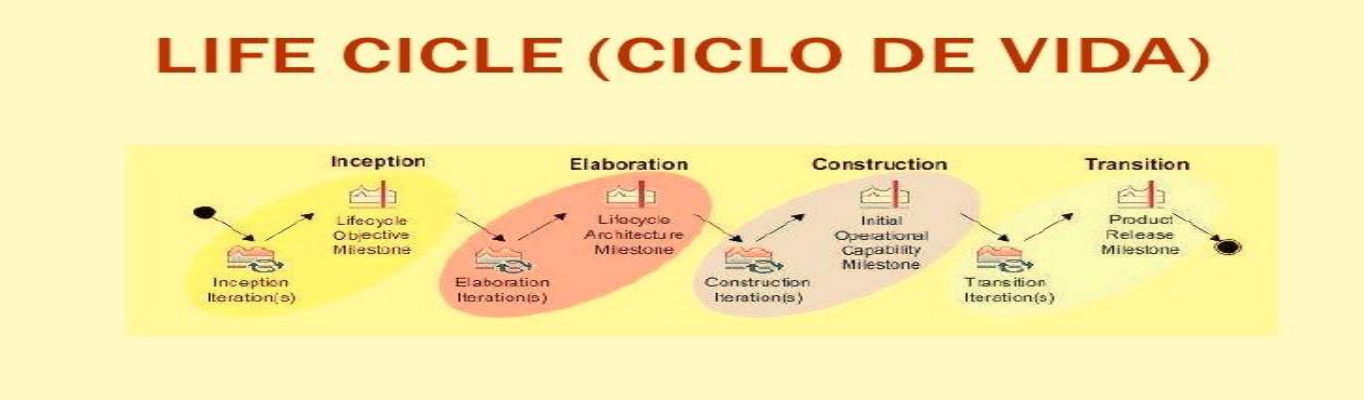


Fig. 2 Ciclo de vida de la metodología Open Up

- **Fase de inicio:** En esta fase, las necesidades de cada participante del proyecto son tomadas en cuenta y plasmadas en objetivos del proyecto. Se definen para el proyecto: el ámbito, los límites, el criterio de aceptación, los casos de uso críticos, una estimación inicial del coste y un boceto de la planificación.
- **Fase de elaboración:** En esta fase se realizan tareas de análisis del dominio y definición de la arquitectura del sistema. Se debe elaborar un plan de proyecto, estableciendo, unos requisitos y una arquitectura estables. Por otro lado, el proceso de desarrollo, las herramientas, la infraestructura a utilizar y el entorno de desarrollo también se especifican en detalle en esta fase.
- **Fase de construcción:** Todos los componentes y funcionalidades del sistema que falten por implementar son realizados, probados e integrados en esta fase. Los resultados obtenidos en forma de incrementos ejecutables deben ser desarrollados de la forma más rápida posible sin dejar de lado la calidad de lo desarrollado.
- **Fase de transición:** Esta fase corresponde a la introducción del producto en la comunidad de usuarios, cuando el producto está lo suficientemente maduro. La fase de la transición consta de las sub-fases de pruebas de versiones beta, pilotaje y capacitación de los usuarios finales y de los encargados del mantenimiento del sistema. En función de la respuesta obtenida por los usuarios

puede ser necesario realizar cambios en las entregas finales o implementar alguna funcionalidad más. (20)

1.5.3 Selección de la metodología a utilizar

Después de haber analizado las características fundamentales de las metodologías de desarrollo XP y Open Up se determinó que para un mejor desarrollo del proyecto se seleccionará Open Up. A pesar de las ventajas que ofrece XP, como su diseño hace que apenas se le dé importancia al análisis como fase independiente, debido a que se trabaja exclusivamente en función de las necesidades del momento no es oportuno porque es muy costosa en caso de fallar. Es necesario utilizar Open Up atendiendo a que está es apropiada no solo para proyectos pequeños sino también de bajos recursos y permite disminuir las probabilidades de fracaso e incrementar las probabilidades de éxito, es una metodología de desarrollo de software de código abierto diseñada para pequeños equipos. Además es preciso centrarse en la arquitectura para lograr una eficiente integración de la solución que se propone y que esta sea escalable y flexible.

1.6 Herramientas

Las herramientas de desarrollo de software son varios productos informáticos que dan soporte a una tarea concreta dentro de las actividades de desarrollo de software, facilitando y asegurando entregar un sistema con calidad.

1.6.1 Lenguajes de programación

Java: es un lenguaje orientado a objeto, diseñado para crear software altamente fiable y para soportar aplicaciones que serán ejecutadas en los más variados entornos de red. El lenguaje en sí mismo toma mucho de la sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Con respecto a la memoria, su gestión no es un problema debido que es gestionada por el propio lenguaje y no por el programador. (21)

Gracias al API de java se puede ampliar el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas y crear aplicaciones visuales al estilo Windows.

Algunas de las principales características se muestran a continuación:

Orientado a objetos: fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos.

Robusto: fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución.

Multiplataforma: permite que los programas desarrollados en él, puedan funcionar en cualquier ordenador independientemente del sistema operativo que utilice y que puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

Portable: la indiferencia a la arquitectura representa sólo una parte de su portabilidad. Especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas. Estas dos últimas características se conocen como la Máquina Virtual Java (JVM).

Dinámico: el lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas.

Multihilo: Permite la ejecución de varias tareas a la vez.

Python: es un lenguaje de programación de alto nivel cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Usa tipado dinámico y conteo de referencias para la administración de memoria. Permite dividir su programa en módulos reutilizables desde otros programas Python.

Viene con una gran colección de módulos estándar que se pueden utilizar como base para los programas o como ejemplos para empezar a aprender Python. Permite la herencia múltiple así como incorporar rutinas compiladas en C y tiene incluidos tipos de datos de alto nivel, como matrices flexibles y diccionarios. Resolución dinámica de nombres, es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado ligadura dinámica de métodos). (22)

Funciona en Windows, Linux / Unix, Mac OS X, y ha sido adaptado a las aplicaciones Java y .NET. Es libre de usar, incluso para los productos comerciales, debido a su licencia de código abierto. (23)

Algunas de las principales características se muestran a continuación:

Interpretado: No se debe compilar el código antes de su ejecución. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador.

Orientado a Objetos: La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

Funciones y bibliotecas: Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc.

Sintaxis clara: tiene una sintaxis muy visual, gracias a una notación indentada (con márgenes) de obligado cumplimiento.

Multiparadigma: lo que significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional.

C++: Es un lenguaje orientado a objeto al que se le añadieron características y cualidades de las que carecía el lenguaje C. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción. (24)

Depende mucho del hardware, es uno de los lenguajes más potentes porque permite programar a alto y a bajo nivel pero es complicado porque los programadores deben hacerlo casi todo por ellos mismos. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. Por otra parte, este lenguaje es particularmente difícil debido a que los programadores necesitan una gran disciplina para no cometer errores y además son estos los encargados de planificar la gestión de memoria y los errores, por este concepto suelen ser graves.

Para el desarrollo de este sistema se decidió utilizar el lenguaje de programación Java porque es un lenguaje altamente fiable para el desarrollo de aplicaciones de escritorio, presenta un tiempo de ejecución relativamente bajo, es multiplataforma y contiene bibliotecas que facilitan al programador desarrollar una aplicación segura. Es uno de los lenguajes actualmente más usados y conocidos por los programadores.

1.6.2 Herramienta Case

CASE por sus siglas en inglés corresponde a las iniciales de: Computer Aided Software Engineering; y en su traducción al español significa Ingeniería de Software Asistida por Computación. Las herramientas CASE representan una forma que permite Modelar los Procesos de Negocios. Un elemento importante conveniente de destacar, es que las herramientas CASE, son eso: "HERRAMIENTAS", y que como tales permiten aumentar la productividad en el desarrollo de un proyecto y como herramientas que son, deben ser aplicadas a una metodología determinada.” (25)

“**Visual Paradigm** es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software para el modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Soporta aplicaciones web. Se caracteriza por el uso de un lenguaje estándar común al equipo de trabajo que facilita la comunicación entre sus integrantes, es una herramienta fácil de instalar y actualizar. Se utilizará en su versión 8.0 para el modelado de los artefactos que genere la metodología seleccionada en el epígrafe anterior.

Esta herramienta permite aumentar la calidad del software, a través de la mejora en el desarrollo y mantenimiento del mismo, de igual forma potencia la reutilización del software y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería del Software. (26)

Dentro de sus características fundamentales están:

Multiplataforma: soportada en plataformas Java para Sistemas Operativos Windows, Linux y Mac OS X.

Interoperabilidad: intercambia diagramas UML y modelos con otras herramientas.

Generación de Documentación: comparte y genera los diagramas y diseños en formatos como PDF20, HTML, JPG y Microsoft Word.

Ingeniería de Código: permite generación de código e ingeniería inversa en lenguajes como Java, C++, CORBA, IDL, PHP, XML Schema, Ada, Python, C#, VB .NET, ODL, Flash ActionScript, Perl y Rugby.

Integración con Entornos de Desarrollo: apoyo al ciclo de vida completo de desarrollo del software: análisis, diseño e implementación, en IDE como Eclipse, Microsoft Visual Studio, NetBeans, Sun ONE, Oracle JDeveloper, JBuilder y otros.

Modelamiento de Bases de Datos: generación de bases de datos, conversión de diagramas entidad-relación a tablas de base de datos, mapeos de objetos y relaciones, ingeniería inversa desde gestores de bases de datos.

1.6.3 Entorno de Desarrollo Integrado (IDE) NetBeans

Un **IDE** 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.

NetBeans es un entorno de desarrollo integrado, escrito en Java y multiplataforma. Producto que permite el ciclo de desarrollo de aplicaciones completo (compilación/depuración/perfil) y en la creación de proyectos facilita una guía para la organización del código fuente. Dentro de la comunidad de desarrolladores cuenta con un elevado apoyo por la amplia documentación y recursos de capacitación que presenta. El editor de código fuente que presenta en estos momentos es más ágil y a la vez robusto e integra conjuntamente lenguajes como HTML, JavaScript y CSS, características que lo convierten en una excelente herramienta para el desarrollo de soluciones informáticas. (27)

Se ejecuta en Windows, Linux, Mac OS X y Solaris y además es un producto de código abierto y gratuito sin restricciones de uso.

Contiene todos los módulos necesarios para el desarrollo de aplicaciones Java. Permite la ingeniería inversa, así como la integración con diferentes frameworks. Soporta el desarrollo de todos los tipos de

aplicación Java (J2SE, web, EJB y aplicaciones móviles). Se ha convertido en un IDE apto para la mayoría de los lenguajes de programación de código abierto modernos.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. Se usará la versión 7.4 para la implementación de la biblioteca propuesta.

1.6.4 Sistema gestor de bases de datos MongoDB

MongoDB es un potente, flexible y escalable gestor de bases de datos. Combina la capacidad de escalar con muchas de las características más útiles de las bases de datos relacionales, como los índices secundarios, consultas de rangos de clasificación y de ordenamiento.

Simplifica la administración de bases de datos por lo que los servidores se administran a sí mismos tanto como sea posible. Aparte de poner en marcha el servidor de bases de datos, muy poco es necesario en la administración. Si el servidor maestro se cae, MongoDB puede conmutar por error automáticamente a un esclavo copia de seguridad y promover el esclavo a un maestro. En un entorno distribuido, el grupo necesita que le digan sólo que existe un nuevo nodo para integrar automáticamente y configurarlo, esta propiedad se conoce como conjuntos de réplicas (Replica Sets) como se muestra en la figura 3.

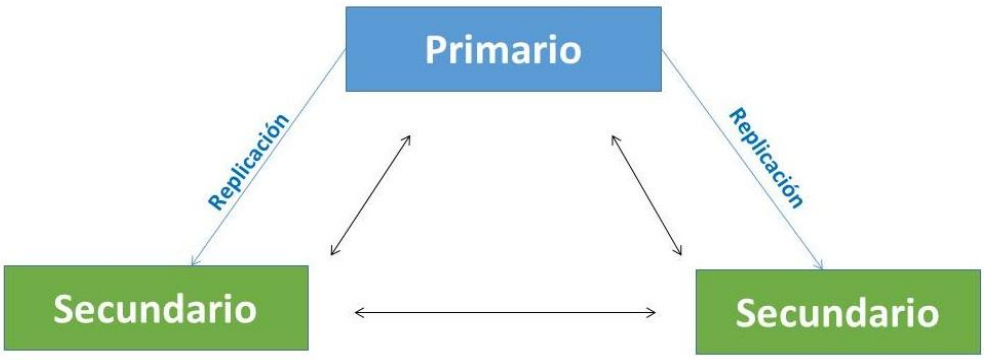


Fig. 3 Graficación de la característica conjuntos de réplicas.

Este gestor remplaza el concepto de filas por documentos y bases de datos por colecciones de datos. No tiene esquemas: las claves de un documento no están predefinidas o fijas. Sin un esquema para cambiar, las migraciones masivas de datos suelen ser innecesarias. Las llaves nuevas o que faltan pueden ser tratadas a nivel de aplicación, en lugar de obligar a todos los datos a que tengan la misma forma. Esto le garantiza una gran flexibilidad en su funcionamiento con la evolución de modelos de datos.

El enfoque orientado a documentos hace posible representar relaciones jerárquicas complejas con un solo registro. Esto le ofrece a los desarrolladores en lenguajes orientados a objetos que puedan pensar la forma en que quieren sus datos. La característica Auto-Sharding le permite además dividir automáticamente los datos a través de múltiples servidores. Se puede equilibrar los datos y carga a través de un clúster, lo que permitiría centrarse en la programación de la aplicación. (28)

Dentro de sus características fundamentales están:

Indexación: soporta índices secundarios genéricos, lo que permite una gran variedad de consultas rápidas y las capacidades de indexación geoespaciales también.

Almacenado JavaScript: en lugar de los procedimientos almacenados, los desarrolladores pueden almacenar y utilizar las funciones y valores de JavaScript en las capacidades de indexación geoespaciales del lado servidor.

Agregación: soporta MapReduce [3] y otras herramientas de agregación.

Colecciones de tamaño fijo: útiles para ciertos tipos de datos, tales como troncos.

Almacenamiento de archivos (GridFS): soporta un protocolo fácil de usar para el almacenamiento de grandes archivos y metadatos de archivos.

Conclusiones parciales del capítulo

En este capítulo se analizaron los principales conceptos asociados a la compresión y cifrado de datos. Se seleccionó Blowfish como algoritmo de cifrado de datos y Snappy para la compresión. Se expusieron

[3] MapReduce es un modelo de programación y una aplicación asociada para el procesamiento y la generación de grandes conjuntos de datos.

características generales de sistemas de cifrado y compresión de archivos existentes en el mundo, que no representan una solución al problema pero sirvieron de base para elaborar una idea general de cómo se desea representar la solución final. Se realizó un análisis detallado de las herramientas y metodología a utilizar, definiendo Open UP como metodología de desarrollo. Se seleccionó además Visual Paradigm 8.0 como herramienta de modelado, NetBeans 7.4 como plataforma de desarrollo y Java 1.7 como lenguaje de programación para la implementación del sistema.

Capítulo 2: Descripción y diseño de la solución propuesta: Monkript

Introducción

En el presente capítulo se describen las características de la biblioteca a desarrollar, realizando un análisis de la propuesta de solución. Se realizará el modelo conceptual que permitirá identificar los conceptos asociados al entorno de Monkript, detallando los requisitos que se deben satisfacer para dar cumplimiento a los objetivos propuestos. Se obtendrá el diagrama de clase del diseño, fundamentando la arquitectura seleccionada y los patrones de diseño utilizados. Al finalizar el capítulo se tendrá una mejor comprensión del negocio y del sistema a desarrollar en general.

2.1 Modelo Conceptual

Dado que con lo explicado hasta el momento no existe total claridad de los conceptos asociados a los proceso del negocio se decide realizar un Modelo Conceptual que fundamente los conceptos significativos para un mejor dominio del problema y las relaciones entre ellos. (29)

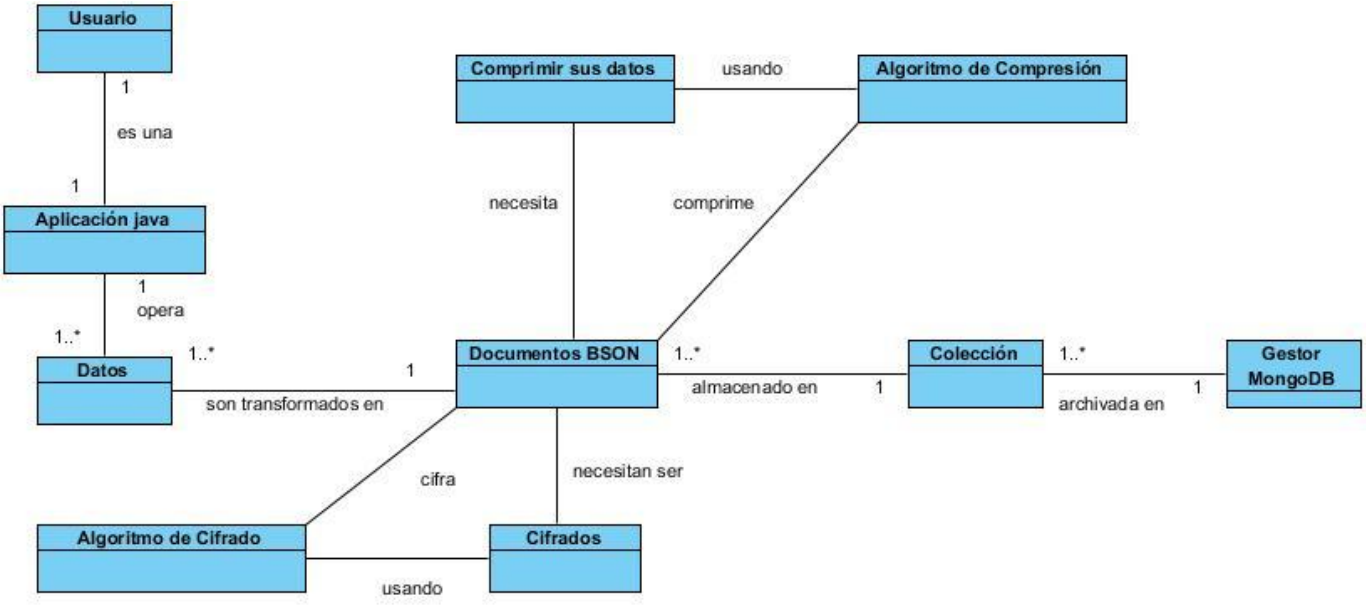


Fig. 4 Modelo Conceptual.

En el Modelo Conceptual propuesto anteriormente un **Usuario** será una **Aplicación java** que trabajará con MongoDB actuando como usuario de este gestor realizando operaciones sobre los **Datos** que en este

caso son transformados en estructuras denominadas **Documentos BSON** que necesitan ser **Cifrados** para una mayor seguridad de la información mediante un **Algoritmo de Cifrado** y también necesitan ser **Comprimidos** mediante un **Algoritmo de Compresión**, finalmente esta estructura se almacena en una **Colección** en el **Gestor MongoDB.**

Algunos de los conceptos que no se han definido aún en la investigación son los siguientes:

Usuario: Puede ser un humano, un programa u otra computadora que manipulen de manera directa un producto de software.

Aplicación java: Es un software implementado sobre la base del lenguaje de programación Java.

Datos: Información dispuesta de manera adecuada para su tratamiento por un ordenador.

Colección: Es un conjunto ordenado de cosas, por lo común de una misma clase y reunidas por su especial interés o valor. (4)

El concepto **Documentos BSON** aunque se describe en la introducción es necesario destacar que es un formato de serialización utilizado para almacenar documentos y hacer llamadas a procedimientos remotos en MongoDB. "BSON" es un acrónimo de las palabras "binarios" y "JSON". Se puede conceptualizar BSON como una representación binaria de documentos JSON (JavaScript Object Notation). (30)Donde la información se guarda con la estructura clave valor.

2.2 Requerimientos del sistema

Para lograr el desarrollo de un sistema correcto es necesario establecer de una forma clara los requisitos, quienes reflejan las necesidades de un cliente o de un usuario. Los requisitos pueden tener dos clasificaciones: requisitos funcionales y requisitos no funcionales.

2.2.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe propiciar el sistema o sea definen las capacidades, condiciones o funciones que el sistema debe ser capaz de satisfacer. (31)A continuación son enumerados los ochos requisitos identificados para el desarrollo de la biblioteca de cifrado y

compresión Monkript:

RF1: Cifrar documentos BSON. La biblioteca debe ser capaz de cifrar los documentos BSON que el usuario desee insertar en MongoDB.

RF2: Insertar documentos cifrados en una colección en MongoDB. La biblioteca debe ser capaz de permitir que se adicionen los documentos cifrados en una colección de MongoDB.

RF3: Actualizar documentos cifrados almacenados en una colección en MongoDB. La biblioteca debe ser capaz de permitir que se actualicen los atributos de un documento cifrado en una colección de MongoDB.

RF4: Eliminar un documento cifrado almacenado en una colección de MongoDB. La biblioteca debe ser capaz de permitir que se eliminen los documentos cifrados almacenados en una colección de MongoDB.

RF5: Descifrar documentos almacenados en una colección de MongoDB. La biblioteca debe ser capaz de permitir que se descifren los documentos almacenados en una colección de MongoDB**.**

RF6: Buscar todos los documentos cifrados de una colección almacenada en MongoDB. La biblioteca debe ser capaz de permitir la búsqueda sobre los documentos cifrados en una colección de MongoDB.

RF7: Comprimir un documento almacenado en una colección de MongoDB. La biblioteca debe ser capaz de permitir que se compriman los documentos que se almacenan en una colección de MongoDB**.**

RF8: Descomprimir un documento almacenado en una colección de MongoDB. La biblioteca debe ser capaz de permitir que se descompriman los documentos almacenados en una colección de MongoDB.

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos requerimientos que no se refieren directamente a las funciones

específicas que ofrece el sistema, sino a las propiedades emergentes de éste. (31) A diferencia de los funcionales estos definen cuales son las propiedades con la que debe contar el sistema, así como sus restricciones fundamentales. Teniendo en consideración que Monkript será una biblioteca para el cifrado y compresión de datos sensibles en MongoDB el espectro de los requerimientos no funcionales se reduce de la siguiente manera:

RNF Software:

- Se requiere tener instalada la Máquina Virtual de Java 1.7.

- Se requiere tener instalado el Sistema Gestor de Bases de Datos (SGBD) MongoDB 2.4.4.

RNF Soporte: La biblioteca debe contar con la documentación actualizada para su uso (en este caso basada en el formato Javadoc de java) para facilitar la reutilización y el mantenimiento de las funcionalidades implementadas.

RNF Portabilidad: La biblioteca podrá ser ejecutada en cualquier sistema operativo siempre que se haya instalado la Máquina Virtual de Java 1.7.

RNF Disponibilidad: Se podrá hacer uso de la biblioteca siempre que la aplicación que desee utilizarla esté conectada a ella.

RNF Diseño e implementación:

- Java Development Kit (JDK) en su versión 1.7.
- NetBeans 7.4 como entorno de desarrollo integrado.
- Visual Paradigm para UML 8.0 como herramienta CASE.
- MongoDB 2.4.4 como gestor de bases de datos no relacional.

2.3 Descripción de la solución que se propone

La solución propuesta está diseñada para aquellas instituciones o usuarios que almacenen sus datos usando el gestor MongoDB mediante una aplicación java. Permitiéndoles una mayor protección de los datos garantizando el cifrado de estos en el nivel de aplicación, dado que este nivel mantiene todo el control de la información, así esta nunca se trasmitirá ni almacenará en texto plano. Ninguna parte de la

capa de datos puede revelar valores en texto plano a potenciales atacantes y los logs del sistema solo contendrán información cifrada. Monkript al trabajar en el nivel de aplicación no incluirá el proceso de generación automática ni gestión de la clave que se utilice para el cifrado de los datos, siendo responsable la aplicación que haga uso de la biblioteca del tratamiento de esta llave.

Monkript posibilitará además la compresión de esa información cifrada, ya que el tamaño de un documento cifrado es mayor que un documento sin cifrar. El cifrado y la compresión se realizarán como procesos inseparables, siempre que se cifre un documento este será comprimido.

Es importante destacar que la biblioteca que se propone permitirá que se realicen sobre los datos cifrados y comprimidos la mayoría de las operaciones que se pueden ejecutar en MongoDB. El proceso anteriormente mencionado será adicional al que normalmente realiza MongoDB con los datos en texto plano, sin afectarse las operaciones sobre los mismos, es decir que permitirá almacenar y operar sobre datos en texto plano y datos cifrados y comprimidos.

La aplicación que haga uso de la biblioteca, la cual se encuentra incluida en el driver de MongoDB para java, deberá importar el mismo y realizar la conexión apoyándose en la documentación generada por Monkript. (Ver Anexo 1)

2.3.1 Actores del sistema

Un actor no es más que aquella persona que interactúa con el sistema, quien inicializa los casos de uso y puede ser un humano, una máquina o un software. En el caso específico de la biblioteca que se propone sería una **aplicación java,** que representa la aplicación encargada de realizar solicitudes de tareas a la biblioteca para su procesamiento.

2.3.2 Casos de Uso del Sistema

Un caso de uso identifica las características claves del sistema agrupando los requerimientos del usuario, por lo que cada caso de uso expresa una meta que el sistema debe lograr. De ahí a que el modelo de caso de uso del sistema que contiene actores, casos de uso y sus relaciones describa las funcionalidades propuesta del nuevo sistema.

A continuación se muestran los casos de uso identificados:

Tabla 1 Casos de Uso del Sistema

Referencia a requisitos	**Nombre del caso de uso**	**Prioridad**
RF1	Cifrar documentos BSON	Crítico
RF2, RF3, RF4, RF6	Gestionar documentos cifrados en una colección de MongoDB.	Crítico
RF5	Descifrar documentos almacenados en una colección de MongoDB	Crítico
RF7	Comprimir un documento almacenado en una colección de MongoDB.	Crítico
RF8	Descomprimir un documento almacenado en una colección de MongoDB.	Crítico

La tabla 1 muestra los 5 casos de usos identificados así como la prioridad de estos y la referencia a los requisitos funcionales que responden. El comportamiento de estos casos de uso debe ser estructurado y organizado por patrones que capturan las mejores prácticas para su modelación. Entre las clasificaciones de los patrones de caso de uso se encuentran:

- Concordancia.
- Extensión Concreta o Inclusión.
- Múltiples Actores.
- CRUD (acrónimo de las palabras en inglés Create, Read, Update, Delete que traducidas al español son crear, leer, actualizar y eliminar). (32)

2.3.3 Diagrama de Casos de Uso del Sistema.

Los diagramas de Casos de Uso del Sistema (CUS) constituyen una representación gráfica de los proceso como se muestra en el siguiente diagrama:

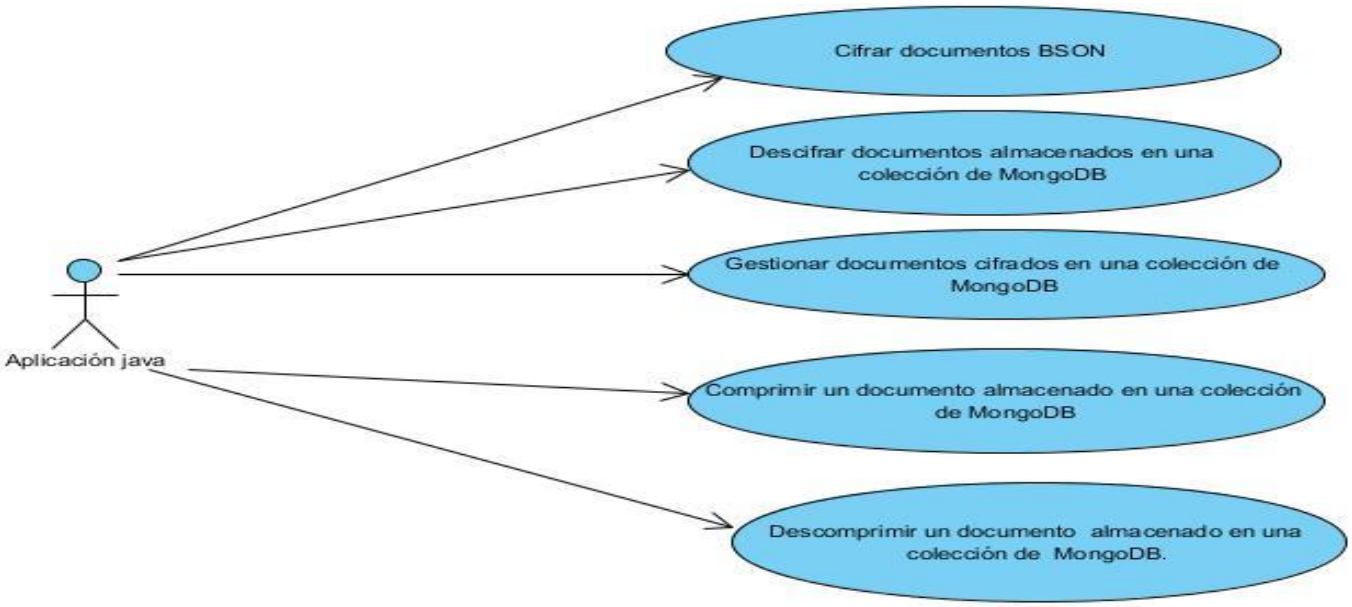


Fig. 5 Diagrama de Casos de Uso del Sistema.

La figura 5 muestra los 5 casos de uso del sistema en los que se agruparon los 8 requisitos funcionales, aplicando el patrón CRUD en el CUS *Gestionar documentos cifrados en una colección de MongoDB*, recomendado para los casos de uso donde se quiere realizar cambios y consultas a alguna entidad del sistema.

2.3.4 Descripción de los Casos de Uso del Sistema

La descripción de los casos de uso del sistema recoge el objetivo del CUS así como un resumen del proceso que alberga el mismo. Las siguientes tablas hacen una descripción textual breve de los casos de uso Cifrar documentos BSON y Gestionar documentos cifrados en una colección de MongoDB. (Ver Anexo 2 para observar la descripción de los restantes CUS)

Tabla 2 Descripción de CUS Cifrar documento BSON

Caso de Uso	Cifrar documento BSON.
Actores	Aplicación Java “(Inicia)”.
Objetivo	Cifrar un documento que posteriormente será almacenado en una colección.
Resumen	El caso de uso inicia cuando la aplicación una vez conectada al gestor MongoDB, haciendo uso del driver, desea cifrar un documento; finalizando el mismo cuando la acción haya sido realizada.
Precondiciones	Que exista la conexión con el gestor.
Referencias	RF 1

Prioridad	Crítico

Tabla 3 Descripción del CUS Gestionar documentos cifrados en una colección de MongoDB.

Caso de Uso	Gestionar documentos cifrados en una colección de MongoDB.
Objetivo	El objetivo principal es insertar, actualizar, buscar y eliminar un documento cifrado en MongoDB.
Actores	Aplicación Java “(Inicia)”.
Resumen	El caso de uso inicia cuando la aplicación una vez conectada al gestor MongoDB, haciendo uso del driver, desea insertar un nuevo documento o decide modificar alguno existente, puede también realizar una búsqueda o desear eliminar alguno de los documentos almacenados; finalizando el mismo cuando la acción haya sido realizada.
Precondiciones	Que exista la conexión con el gestor.
Referencias	RF2, RF3, RF4, RF6
Prioridad	Crítico

2.4 Diseño de la biblioteca Monkript

Según IEEE el diseño es el proceso de definir la arquitectura, los componentes, interfaces y las otras características de un sistema o componente. Visto así como un proceso, entonces esta actividad del ciclo de vida de ingeniería de software debe describir la arquitectura de software. También debe describir los componentes a un nivel de detalles que permitan su construcción.

2.4.1 Arquitectura propuesta

La arquitectura de software puede considerarse entonces como el “puente” entre los requerimientos del sistema y la implementación. (33) En la que se especifica la estructura del sistema, entendida como la organización de componentes y relaciones entre ellos, aunque según Pressman también reduce los riesgos asociados con la construcción del software. (17) En este trabajo la arquitectura brindará una organización básica para la implementación de la biblioteca a desarrollar permitiéndole una futura fácil integración con otros sistemas.

Estilo Arquitectónico

El estilo arquitectónico describe el esqueleto estructural y general de la aplicación, es independiente de otros estilos expresando la relación concreta entre los componentes. En el caso de la construcción de la biblioteca se usará el estilo llamada retorno ya que éste refleja la estructura del lenguaje de programación, permitiéndole al diseñador del sistema elaborar una estructura relativamente fácil de modificar fundamentada en un razonamiento jerárquico. (31) Emplear este estilo posibilitará además la comunicación, la coordinación y cooperación entre los componentes y las restricciones que definen como se integran los componentes para conformar la biblioteca, además de los modelos semánticos que facilitan al diseñador el entendimiento de todas las partes del sistema, evitando que las variaciones realizadas a funcionalidades o componentes específicos afecten el funcionamiento general de la biblioteca.

Patrón Arquitectónico

El patrón arquitectónico es quien define la estructura básica de la aplicación siendo una plantilla de construcción que incluye reglas y pautas para su organización, el patrón de arquitectura en n capas, es el que se aplicará en este diseño ya que brinda una organización jerárquica tal que cada capa proporciona servicios a la capa inmediata superior y se sirve de las prestaciones que le brinda la inmediata inferior. Este patrón es un sub-estilo dentro del estilo llamada y retorno anteriormente definido, en el cual las restricciones topológicas del patrón pueden incluir una limitación que exige a cada capa operar solo con la adyacente y a elementos de una capa entenderse sólo con otros elementos de la misma capa. (29) En el diseño de la biblioteca a desarrollar se definen las capas de **Lógica de Negocio** y la de **Acceso a Datos**, por lo que se empleará el patrón en la variante dos capas.

Lógica de Negocio: es la capa que sirve de mediador entre las aplicaciones que usen la biblioteca y es donde se alojan mayoritariamente sus clases. En ella se reciben las peticiones del usuario y se envían las respuestas tras el procesamiento y la comunicación con la capa de acceso a datos. Debe su nombre a que es aquí donde se establecen todas las reglas del negocio que deben cumplirse.

Acceso a Datos: es la capa diseñada para que residan los datos y es la encargada de acceder a los mismos. Recibe solicitudes de almacenamiento o recuperación de información desde la capa de lógica de negocio.

También existirá un tratamiento vertical de las excepciones donde las clases alojadas en ambas capas brindarán un tratamiento constante de las excepciones.

2.4.2 Vista lógica de la arquitectura

La vista lógica de la arquitectura describe la arquitectura del sistema presentando varios niveles de refinamiento. Indica los módulos lógicos principales así como sus relaciones. Describe el diseño más importante de las clases y su organización en paquetes y subsistemas, y la organización de éstos en capas.

La vista lógica de la arquitectura de la biblioteca que se propone estará conformada por dos capas, la Capa de Lógica de Negocio y la Capa de Acceso a Datos. Ambas capas quedan expuestas en la figura 6 detallando las clases del diseño que las componen y sus relaciones evidenciando la comunicación entre estas.

Diagrama de clase del diseño

Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema, y además se obtiene como resultado del refinamiento del modelo conceptual. Las clases del diseño representan una abstracción de una o varias clases en la implementación del sistema. A continuación se presenta el diagrama de clase del diseño acoplado en las capas de la arquitectura.

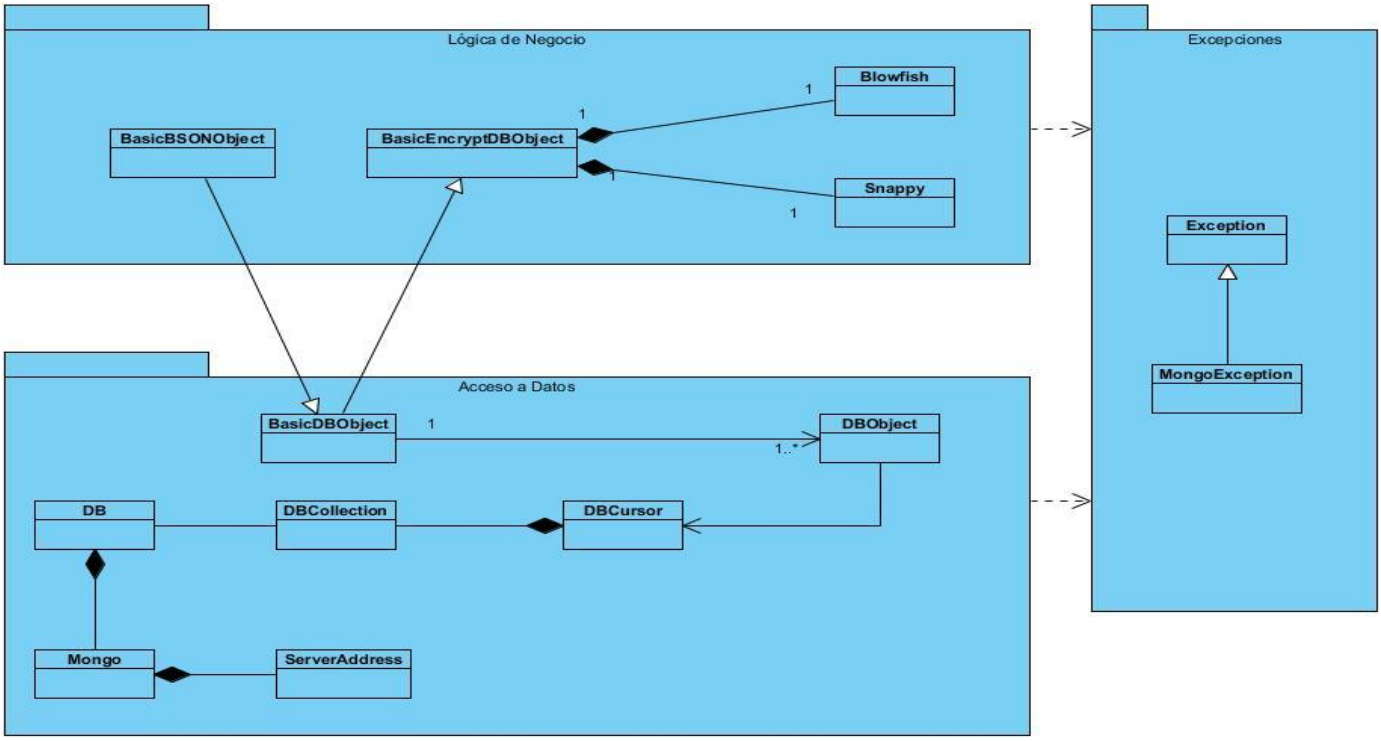


Fig. 6 Diagrama de la Vista Lógica de las Clases del Diseño.

2.4.3 Descripción de las clases del diseño

Capa Lógica de Negocio:

BasicEncryptDBObject: Es la clase responsable de instanciar las clases Blowfish y Snappy con el objetivo de utilizar estas instancias para cifrar y comprimir respectivamente cuando se hace uso del método *put()* adicionando clave/valor a partir de un objeto.

Blowfish: Es la clase responsable del cifrado haciendo uso de los métodos *Encrypt()* para el cifrado y *Decrypt()* para descifrar. Esta clase también cuenta con 2 atributos de tipo *string*: la clave con la que se desea cifrar y el texto que se quiere cifrar.

Snappy: Es la clase responsable de la compresión haciendo uso de los métodos *compress()* para la compresión de una colección y *uncompressString()* para descomprimir la misma.

BasicBSONObject: Es la clase que permite manejar documentos sin cifrar ni comprimir.

Capa Acceso a Datos:

BasicDBObject: Es la clase responsable de implementar la clase interfaz DBObject implementando los métodos de la misma y extiende de la clase BasicBSONObject heredando los métodos de la misma.

DBCursor: Es la clase responsable de iterar sobre los resultados de las colecciones de la base de datos haciendo uso de varios métodos entre ellos: *next()* para obtener el resultado siguiente y *hasNext()* para saber si existe un resultado siguiente.

DBCollection: Es la clase responsable de gestionar todo el trabajo con documentos e índices, hace uso de varios métodos entre ellos: *insert(), find(), update(), remove()* para insertar, buscar, actualizar y eliminar respectivamente un documento de la colección; *createIndex()* y *dropIndex()* para crear y eliminar índices respectivamente.

DB: Es la clase responsable de gestionar todo el trabajo con colecciones de datos haciendo uso de varios métodos entre ellos: *createCollection()* para crear una nueva colección y *getCollection()* para obtener la misma.

Mongo: Es la clase responsable de crear la conexión del driver con el gestor de bases de datos MongoDB y gestionar todo el trabajo con las bases de datos, haciendo uso de varios métodos entre ellos: *dropDatabase()* para eliminar una base de datos, *getDatabaseNames()* para obtener los nombres de todas las bases de datos.

2.4.4 Patrones de diseño empleados.

Los patrones de diseño refinan los componentes basados en la experiencia obtenida. Son una solución estándar para un problema común de programación.

La calidad del diseño de la interacción de los objetos y la asignación de responsabilidades presentan gran variación. Una mala decisión en el diseño trae como consecuencia componentes frágiles, que no se pueden mantener o reutilizar. Para mitigar estas malas decisiones existen los patrones GRASP, quienes describen los principios fundamentales de la asignación de responsabilidades a objetos expresados en forma de patrones. (29)

En el diseño de la solución propuesta se aplican algunos patrones GRASP de la siguiente manera:

Experto

Problema: ¿Cómo lograr que cada clase cumpla con la responsabilidad que le corresponde?

Solución: Asignar una responsabilidad a la clase que tiene la información necesaria para cumplirla.

La clase Blowfish contará con la información necesaria para cifrar y descifrar un documento que es la responsabilidad que le corresponde a esta clase. (Ver figura 7)

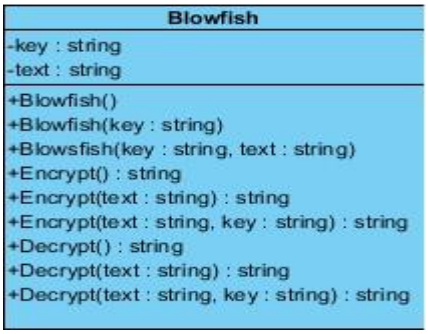


Fig. 7 Aplicación del patrón experto en la clase Blowfish.

Creador

Problema: ¿Cómo lograr relacionar una clase con la clase responsable de cifrar un documento?

Solución: Asignarle a una clase la responsabilidad de crear una instancia de otra clase.

Las clases BasicBEncryptDBObject, será la responsable de crear una nueva instancia de la clase Blowfish para el cifrado del documento. (Ver figura 8)

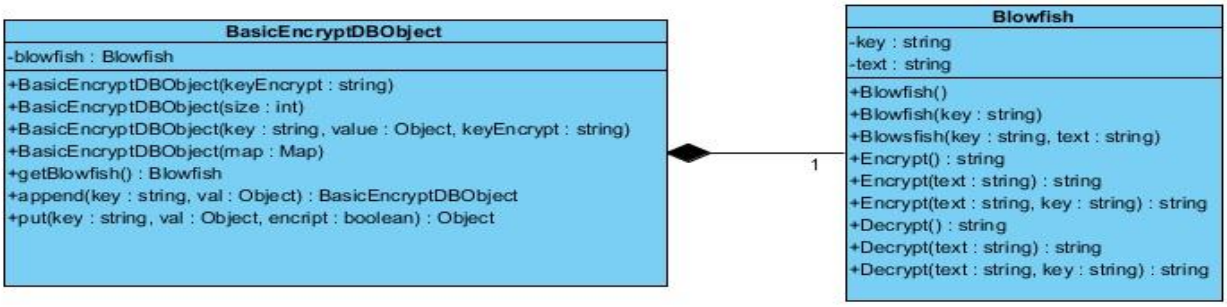


Fig. 8 Aplicación del patrón creador en la relación de la clase BasicEncryptDBObject y la clase Blowfish.

Alta cohesión:

Problema: ¿Cómo lograr que las clases trabajen en su misma área de aplicación?

Solución: Asignar a cada clase la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables, asumiendo solamente las responsabilidades que le corresponden evadiendo un trabajo excesivo.

La clase DBCollection tiene la responsabilidad para definir las acciones sobre los documentos y colabora con otras para realizar diferentes operaciones, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas (Ver figura 9). Así como la clase Blowfish y Snappy tienen la responsabilidad de definir las acciones para el cifrado y compresión respectivamente y colaboran con otras para realizar las mismas.

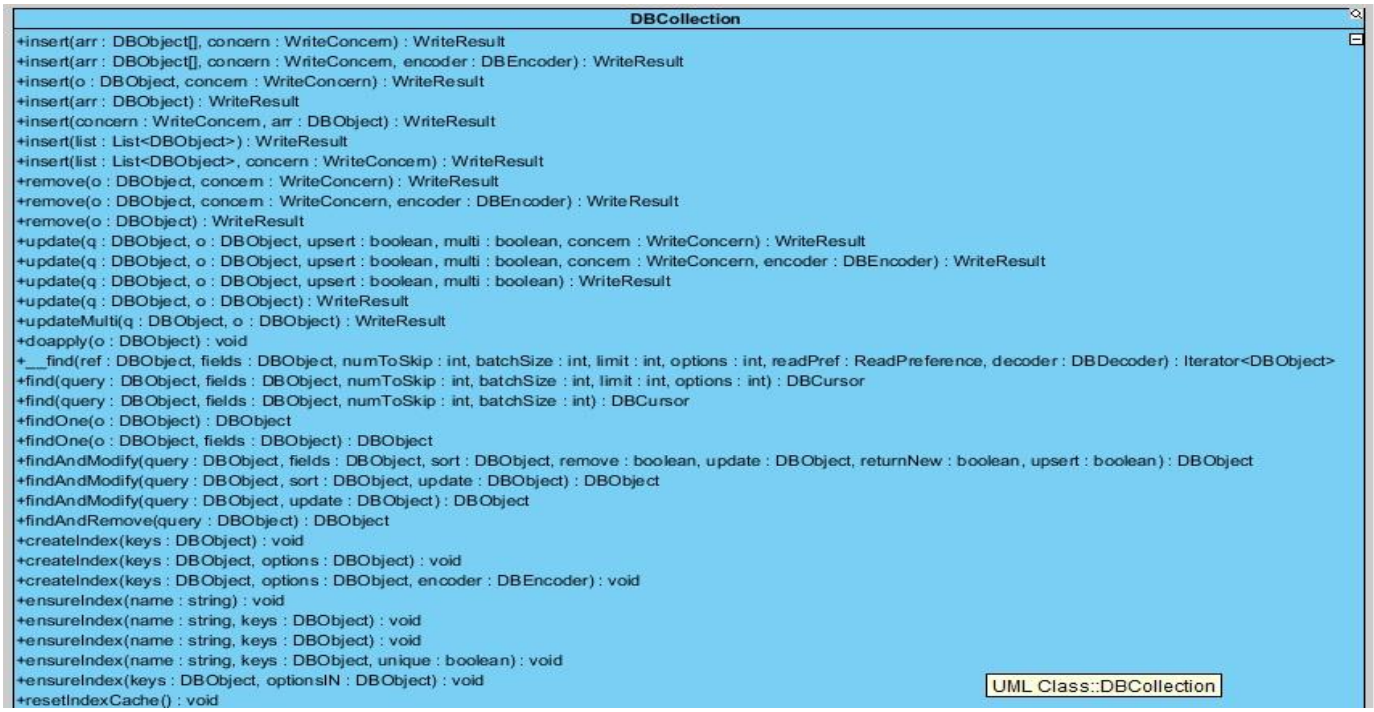


Fig. 9 Aplicación del patrón alta cohesión en la clase DBCollection.

Bajo acoplamiento:

Problema: ¿Cómo lograr que una clase no dependa de otras clases?

Solución: Asignar responsabilidades de manera tal que la fuerza con que una clase está conectada a otras clases sea baja. De tal forma que en caso de producirse una modificación en alguna, se tenga la mínima repercusión posible en el resto de clases, potenciando así la reutilización.

A la clase ServerAddress se le asignarán responsabilidades de forma tal que solo se comuniquen con la clase Mongo que se encarga de controlar el proceso de conexión al gestor de bases de datos MongoDB.(Ver figura 10)

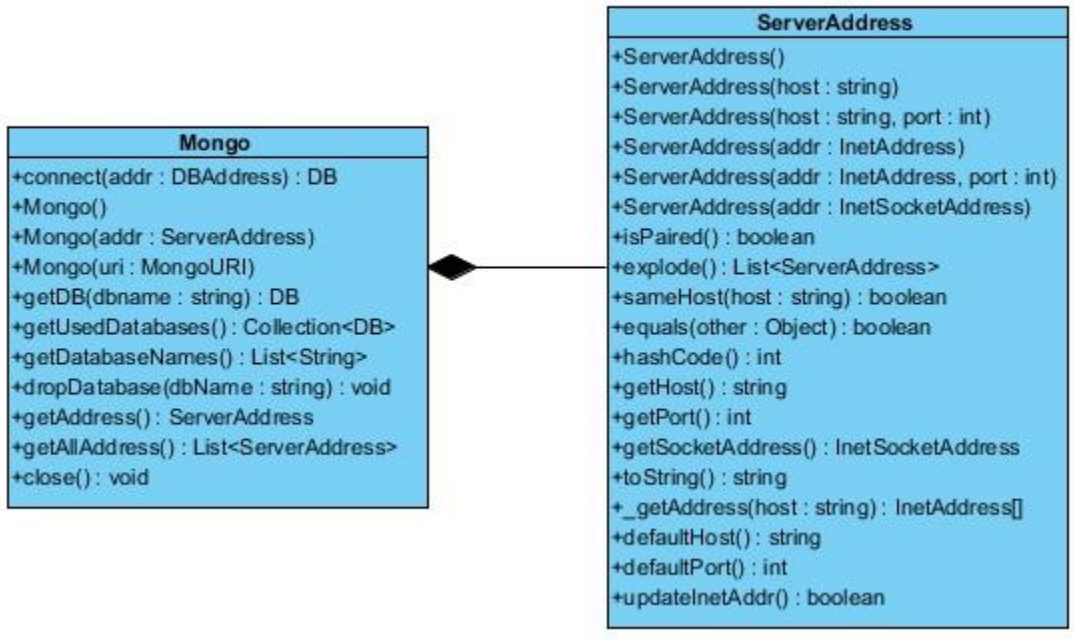


Fig. 10 Aplicación del patrón bajo acoplamiento en la relación de la clase Mongo y la clase ServerAddress.

Conclusiones parciales del capítulo

En el capítulo 2 se describieron las características generales del sistema a implementar. Se generaron algunos artefactos propuestos por la metodología Open Up para el desarrollo del software como el Modelo Conceptual que enmarcó en el contexto donde se usará la biblioteca, fueron capturados los requisitos funcionales y no funcionales y agrupados en Casos de Uso del Sistema generando una visión para la creación del producto. Se presentó el diseño de Monkript con una propuesta de solución que seleccionó la arquitectura en 2 capas. También se definió el diagrama de clase del diseño mostrando la estructura interna del sistema, fundamentando el uso de patrones del diseño GRASP para lograr una correcta implementación de la biblioteca a desarrollar.

Capítulo 3: Implementación y pruebas de la biblioteca

Introducción

Una vez terminado el diseño, se dispone a abordar los temas relacionados con los flujos de trabajo implementación y prueba. Es por esto que en el presente capítulo se desarrolla una descripción de la implementación. También se especifican las pruebas que se le realizaron a la biblioteca Monkript, para validar una correcta implementación de los casos de uso contribuyendo a elevar la calidad del producto desarrollado.

3.1 Implementación del sistema

Como se especificaba anteriormente con el resultado del diseño se comienza la implementación, logrando concebir la arquitectura y el sistema como un todo. Además se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código sea accesible para un futuro mantenimiento del software.

3.1.1 Estándares de codificación

El objetivo fundamental de los estándares de codificación empleados en Monkript es lograr una arquitectura y un estilo consistente, con lo cual el sistema resulte fácil de entender y por supuesto fácil de mantener. Tratando de entender el formato y el estilo utilizado en el código escrito por los desarrolladores, buscando aportar una mayor organización y limpieza.

Indentación: Las normas de indentación se centran en la lectura y comprensión del código por las personas. El indentado debe seguir las siguientes reglas:

- Siempre se indenta 4 espacios el código encerrado entre llaves '{' y '}'.
- El primer corchete está en el mismo renglón que la instrucción.
- A partir de la última llave '}' se termina la indentación.
- Las líneas deben tener una longitud menor a 80 caracteres.

La figura 11 muestra un ejemplo de la indentación del código presentada a continuación:

```
public static String genIndexName( DBObject keys ) {
    StringBuilder name = new StringBuilder();
    for ( String s : keys.keySet() ) {
        if ( name.length() > 0 )
            name.append ( '_' );
        name.append( s ).append( '_' );
        Object val = keys.get( s );
        if ( val instanceof Number || val instanceof String )
            name.append( val.toString().replace( ' ', '_' ) );
    }
    return name.toString();
}
```

Fig. 11 Indentación del código.

Comentarios: Los comentarios son notas cortas explicativas que se agregan al código para aportar mayor información a las personas que lo leen. Los comentarios de línea deben escribirse de forma clara, en el caso de las funcionalidades significativas o clases se usarán comentarios de bloque al inicio de estos. Debe evitarse comentar por cada línea de código. (Ver figura 12)

```
/** Borra todos los índices que aún no se han aplicado a esta colección.*/

public void resetIndexCache(){
    _createdIndexes.clear();
}
```

Fig. 12 Comentarios del código.

Convenciones de nombres: El objetivo fundamental es nombrar las clases y métodos de la biblioteca de cifrado y compresión de forma estándar. En ambos casos se usará el idioma inglés siguiendo la codificación empleada en el driver de Java para MongoDB, si es un nombre compuesto se empleará notación CamelCase[4]. (Ver figura 13)

```
public String encrypt(String key, String text) throws Exception {

    SecretKeySpec secretkey =  new SecretKeySpec(key.getBytes(), "Blowfish");
    Cipher cipher = Cipher.getInstance("Blowfish");
    cipher.init(Cipher.ENCRYPT_MODE, secretkey);
    byte[] hasil = cipher.doFinal(text.getBytes());
    return new BASE64Encoder().encode(hasil);
}
```

Fig. 13 Convenciones de nombre aplicadas en un método.

[4] CamelCase: Estilo de codificación que consiste en usar escritura en bloque para los identificadores de más de una palabra.

3.2 Principales métodos implementados

Para obtener los resultados esperados de la biblioteca Monkript, fue necesario que se implementaran métodos fundamentales para realizar el cifrado y compresión.

Para realizar el cifrado:

El siguiente fragmento de código pertenece a la clase Blowfish. El propósito de ésta es cifrar/descifrar un texto, implementando el algoritmo del mismo nombre, haciendo uso de la sobrecarga del método *Encrypt().* Recibe como parámetro el texto que se desea cifrar, obteniéndose como resultado el texto cifrado.

```
public String Encrypt(String text) throws Exception {

    SecretKeySpec secretkey =  new SecretKeySpec(this.key.getBytes(), "Blowfish");
    Cipher cipher = Cipher.getInstance("Blowfish");
    cipher.init(Cipher.ENCRYPT_MODE, secretkey);
    byte[] hasil = cipher.doFinal(text.getBytes());

    return new BASE64Encoder().encode(hasil);
}
```

Fig. 14 Implementación del método Encrypt().

El siguiente fragmento de código también pertenece a la clase Blowfish, cuyo propósito es permitir descifrar un texto, haciendo uso de la sobrecarga del método *Decrypt(),* que recibe como parámetro el texto que se desea descifrar, obteniéndose como resultado el texto descifrado.

```
public String Decrypt(String text) throws Exception {
    String decrypt = null;
    try
    {
        byte[] keyData = this.key.getBytes();
        SecretKeySpec secretKeySpec = new SecretKeySpec(keyData, "Blowfish");
        Cipher cipher = Cipher.getInstance("Blowfish");
        cipher.init(Cipher.DECRYPT_MODE, secretKeySpec);
        byte[] hasil = cipher.doFinal(new BASE64Decoder().decodeBuffer(text));
        decrypt = new String(hasil);
    }
    catch(Exception e)
    {
        decrypt = text;
    }
    return decrypt;
}
```

Fig. 15 Implementación del método Decrypt().

Para realizar la compresión:

El fragmento de código que se muestra a continuación pertenece a la clase Snappy. El propósito de ésta es comprimir/descomprimir un arreglo de byte, implementando el algoritmo del mismo nombre, haciendo uso de la sobrecarga del método *compress().* Recibe como parámetro el arreglo de byte que se desea comprimir, obteniéndose como resultado el arreglo comprimido.

```
public static byte[] compress(byte[] input) throws IOException {
    return rawCompress(input, input.length);
}
```

Fig. 16 Implementación del método compress().

El fragmento de código del método *uncompressString()* perteneciente a la clase Snappy se muestra en la figura 17. El propósito de este método es descomprimir un arreglo de byte, recibe como parámetro el arreglo de byte que se desea descomprimir, obteniéndose como resultado la cadena de texto inicialmente comprimida.

```
public static String uncompressString(byte[] input) throws IOException {
    try {
        return uncompressString(input, "UTF-8");
    }
    catch (UnsupportedEncodingException e) {
        throw new IllegalStateException("UTF-8 decoder is not found");
    }
}
```

Fig. 17 Implementación del método uncompressString().

El siguiente fragmento de código pertenece a la clase BasicEncryptDBObject. El propósito de esta clase es crear un documento de tipo BasicEncryptDBObject haciendo uso del método *put()* para agregar el par: clave-valor ya sea insertado por el usuario u obtenido de la base de datos. El método recibe como parámetros la clave y el valor del documento y un tercer parámetro para saber que operaciones hacer al documento cifrar/comprimir o descomprimir/descifrar.

```
public Object put(String key, Object val, boolean encrypt) throws IOException {
    String keys = null;
    Object value = null;
    /* Si encrypt es verdadero, se encripta la clave del documento */
    if(encrypt) {
        try {
            keys = blowfish.Encrypt(key);
        } catch (Exception ex) {
            Logger.getLogger(BasicEncryptDBObject.class.getName()).log(Level.SEVERE, null, ex);
        }
        /* Condición para encriptar y comprimir el valor en caso de que no sea otro objeto*/
        if(val instanceof DBObject) {
             value = val;
        } else {
             value = EncryptCompress(val.toString());
        }
    }
    /* Si encrypt fuese falso, se desencripta la clave del documento*/
    else {
        try {
            keys = blowfish.Decrypt(key);
        } catch (Exception ex) {
            Logger.getLogger(BasicEncryptDBObject.class.getName()).log(Level.SEVERE, null, ex);
        }
        /* En caso de que la clave no sea el id del documento y el valor sea otro documento
        se llama al método recursivamente */
        if(!key.equals("_id")) {
            if(val instanceof DBObject) {
                 BasicEncryptDBObject obj = new BasicEncryptDBObject(blowfish.getKey());
                 Set<String> claves = ((DBObject) val).keySet();
                    for (String p : claves) {
                          obj.put(p, ((DBObject) val).get(p), false);
                    }
                 value = obj;
            }
            /* Si el valor  no fuese otro documento, se descomprime y desencripta*/
            else {
                try {
                    String valueOp = Snappy.uncompressString((byte[]) val);

                    value = blowfish.Decrypt(valueOp);
                } catch (Exception ex) {
                    try {
                       value = blowfish.Decrypt(val.toString());
                    } catch (Exception ex1) {
                        Logger.getLogger(BasicEncryptDBObject.class.getName()).log(Level.SEVERE, null,
                    }
                }
            }
        }

    }
    /* En cualquier otro caso se llama al método recursivo pasando los valores
    previamente establecidos
    */
    super.put(keys, value);
    return this;
}
```

Fig. 18 Implementación del método put().

3. 3 Validación de la biblioteca implementada

La validación es el proceso de evaluación de un sistema para determinar si satisface los requisitos determinados. El desarrollo de un sistema informático implica una serie de actividades evidenciadas durante el desarrollo de la investigación donde pueden aparecer errores desde el inicio del proceso, dentro de los pasos del diseño o en la implementación. Por ello el desarrollo del software debe ir acompañado de una actividad que garantice la calidad del mismo. En estas actividades un sistema es ejecutado bajo condiciones o requerimientos determinados, los resultados son observados, registrados y analizados.

Según el propio Pressman *"La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación".* (17) Para validar la biblioteca Monkript se utilizaron pruebas de caja blanca ya que la biblioteca no cuenta con una interfaz visual y es imprescindible validar el correcto funcionamiento de los métodos implementados. La prueba de caja blanca como uno de los tipos de pruebas se considera dentro las más importantes que se le aplican a los softwares, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

3.3.1 Aplicación de la Técnica Camino Básico

La prueba del camino básico es una técnica de prueba del método de caja blanca. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. (34) El fragmento de código correspondiente a la funcionalidad "_lookForHints()" fue seleccionado para aplicar la prueba. (Ver figura 19).

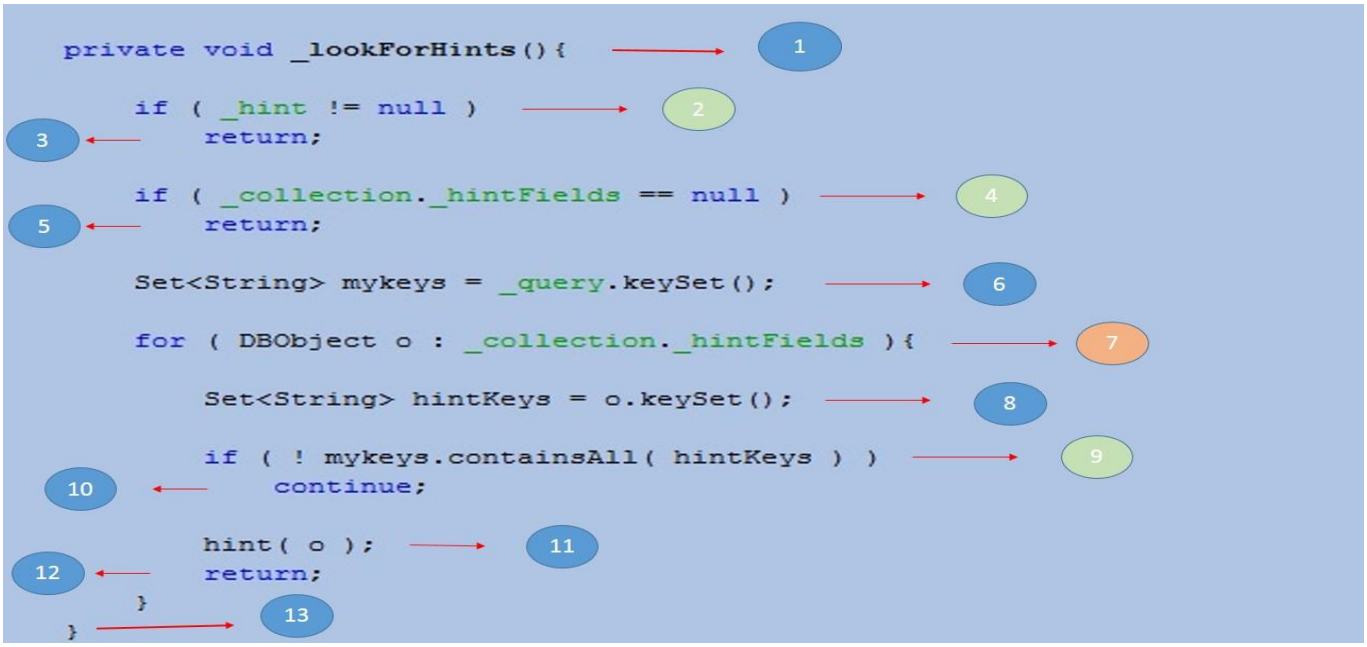


Fig. 19 Método _lookForHints().

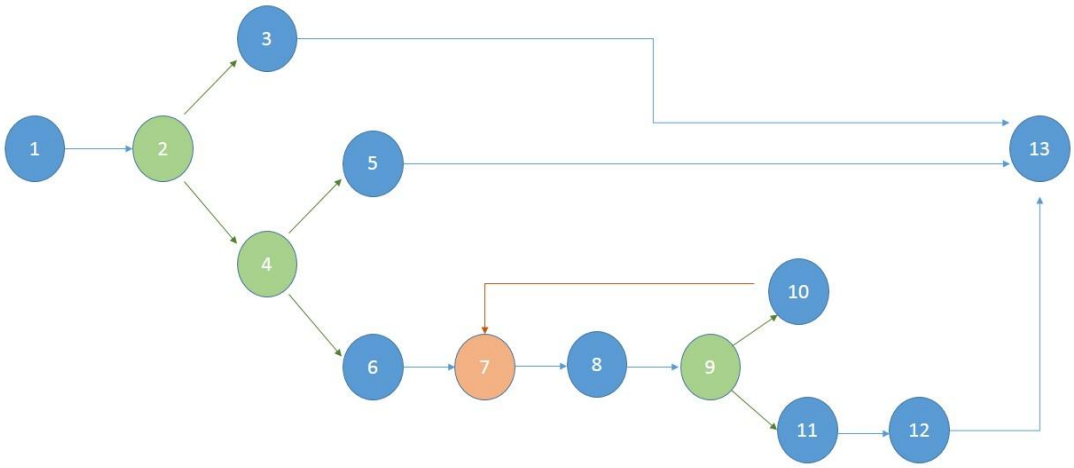


Fig. 20 Grafo de flujo del método _lookForHint().

Se identificaron 13 bloques de ejecución atendiendo a las dependencias procedimentales las cuales construyen el grafo de flujo del método representado en la figura 20. Teniendo esta información se procede al cálculo de la complejidad ciclomática.

La complejidad ciclomática *(V (G))* es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. Donde el valor calculado define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Esta operación se puede calcular de tres formas:

1. **V (G) = A - N + 2**, donde A es la cantidad total de aristas y N la cantidad total de nodos.
2. **V (G) = P + 1**, donde P es la cantidad total de nodos predicados (nodos de los cuales parten dos o más aristas).
3. **V (G) = R**, donde R es el número de regiones del grafo de flujo.

Luego de aplicar las tres fórmulas para obtener un resultado seguro se obtiene una complejidad ciclomática V (G) = 4. Por lo tanto se obtendrán 4 caminos de pruebas, que se muestran a continuación:
Camino I: 1-2-3-13
Camino II: 1-2-4-5-13
Camino III: 1-2-4-6-7-8-9-11-12-13
Camino IV: 1-2-4-6-7-8-9-10-7-8-9-11-12-13

Tabla 4 Descripción de los Casos de Pruebas

Camino	Descripción	Entrada	Salida
I	Mientras se cumpla la primera condicional este camino se ejecutará.	El valor del argumento analizado fue inicializado anteriormente.	Realiza el *return* saliendo del método.
II	Mientras se cumpla la segunda condicional y no se cumpla la primera este camino se ejecutará.	El valor del argumento analizado en la primera condición y la segunda son nulos.	Realiza el *return* saliendo del método.
III	Mientras no se cumplan las dos primeras condiciones y entre al ciclo donde no se cumpla la condición que este alberga, entonces el camino se ejecutará	Todos los valores de los argumentos de las condicionales son falsos.	Logra entrar al ciclo y en la primera iteración hace el *return* y sale del método.
IV	Mientras se esté cumpliendo la condicional que tiene el ciclo este camino se realiza.	El valor del argumento de la condicional alojada en el sitio es	Logra entrar al ciclo, realiza 2 o varias iteraciones hasta que realiza el *return* y

		verdadero.	sale del método.

Se aplicaron un total de 16 casos de pruebas para realizar una correcta validación del código fuente en 3 iteraciones. En la primera iteración se detectaron 6 no conformidades que fueron corregidas totalmente. En la segunda iteración se detectaron dos nuevas no conformidades igualmente corregidas En la tercera y última iteración no se detectaron errores. Estos resultados se muestran a continuación en la gráfica siguiente:

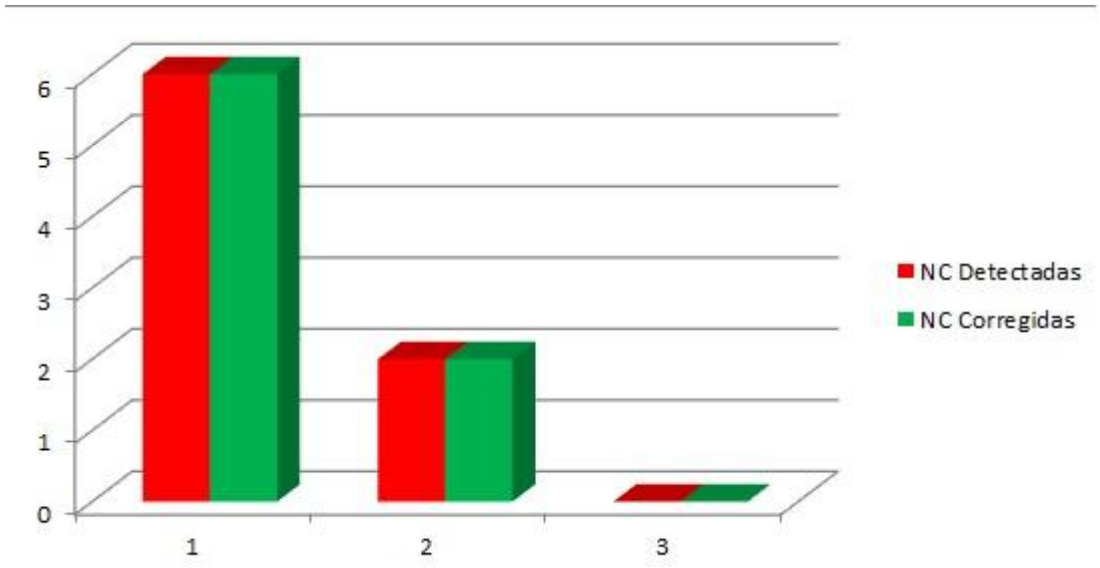


Gráfico 1: Relación de No Conformidades (NC) Detectadas y NC Corregidas por iteraciones.

3.3.2 Pruebas de Unidad

Las pruebas de unidad tienen el objetivo de verificar que el módulo, entendido como la unidad funcional de un programa independiente, está correctamente codificado. Para que una prueba de unidad sea buena debe ser independiente, completa, reutilizable y automatizable donde no se requiera de una intervención manual.

JUnit es una herramienta de software libre para la realización de pruebas de unidad de software desarrollado en java. (35)Para implementar las pruebas con JUnit 4.0 se selecciona la clase a probar, esta herramienta automáticamente crea el paquete de prueba y dentro de este genera la clase de prueba (Test). La clase Test tiene métodos homólogos a la clase probada que contienen los valores establecidos y una función "fail()" que lanza un fallo en el método. Si se ejecuta la clase Test esta muestra los errores en el código y en la barra superior el porciento de satisfacción de los casos de prueba.

A continuación se ejemplifica la prueba realizada a la clase BasicEncryptDBObject mediante la clase BasicEncryptDBObjectIT (Ver Anexo 3), donde se obtiene el siguiente resultado de un 100 % de satisfacción:

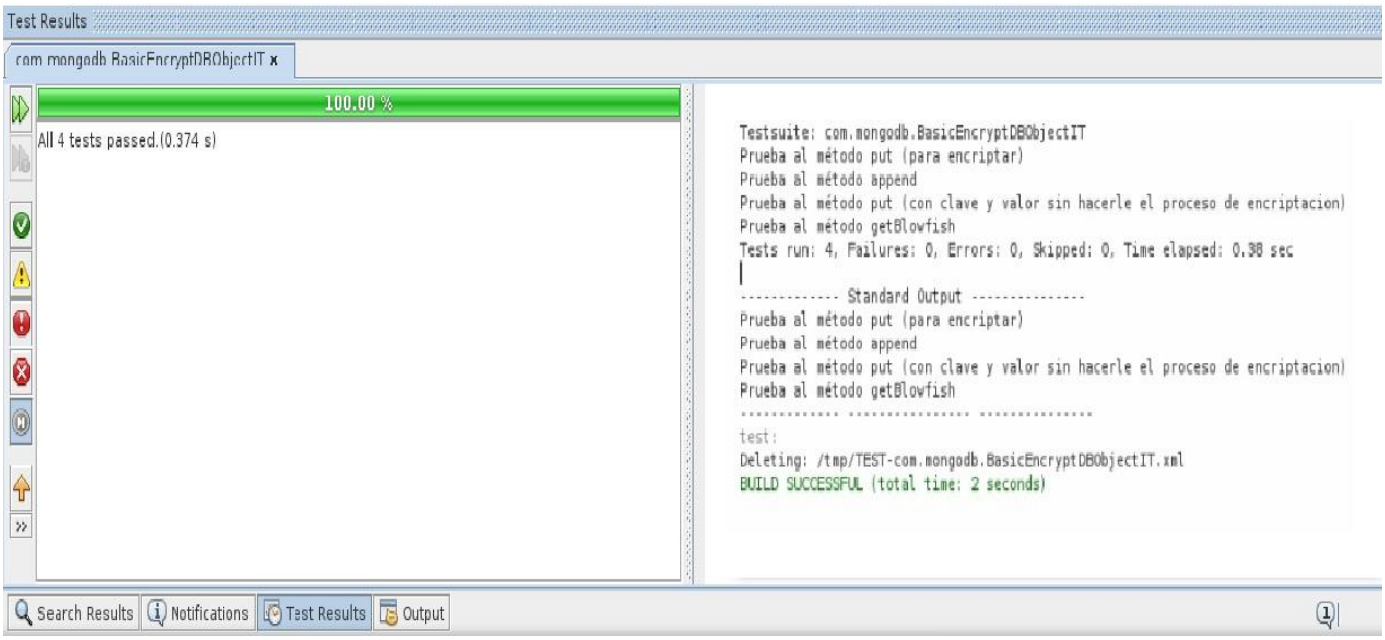


Fig. 21 Resultados de JUnit.

3.3.3 Pruebas de Funcionalidad

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. Un buen instrumento para esta determinación son las llamadas pruebas de funcionalidad. En estas pruebas se evalúa el grado de calidad del software con relación a si el sistema cubre sus necesidades de funcionamiento, acorde a las especificaciones del diseño.

A continuación se muestran 3 colecciones respectivamente almacenadas en el gestor MongoDB haciendo uso de la herramienta gráfica UMongo en las que se tienen insertados 100 documentos. En la figura 22 se observa la colección “Mongo” en la cual los documentos se encuentran en texto plano y tiene un tamaño de 6664 bytes. La colección “encrypt” contiene los documentos cifrados y su tamaño es de 19000 bytes

como se muestra en la figura 23. La colección “Encrp-Comp” contiene sus documentos cifrados y comprimidos como se observa en la figura 24 siendo su tamaño de 12944 bytes.

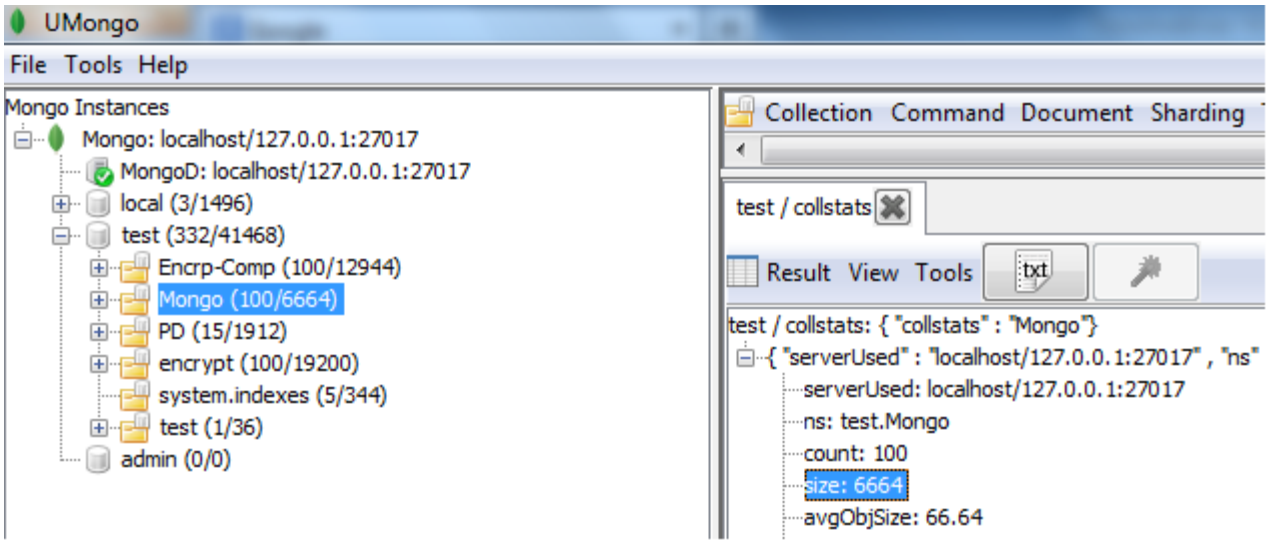


Fig. 22 Colección Mongo.

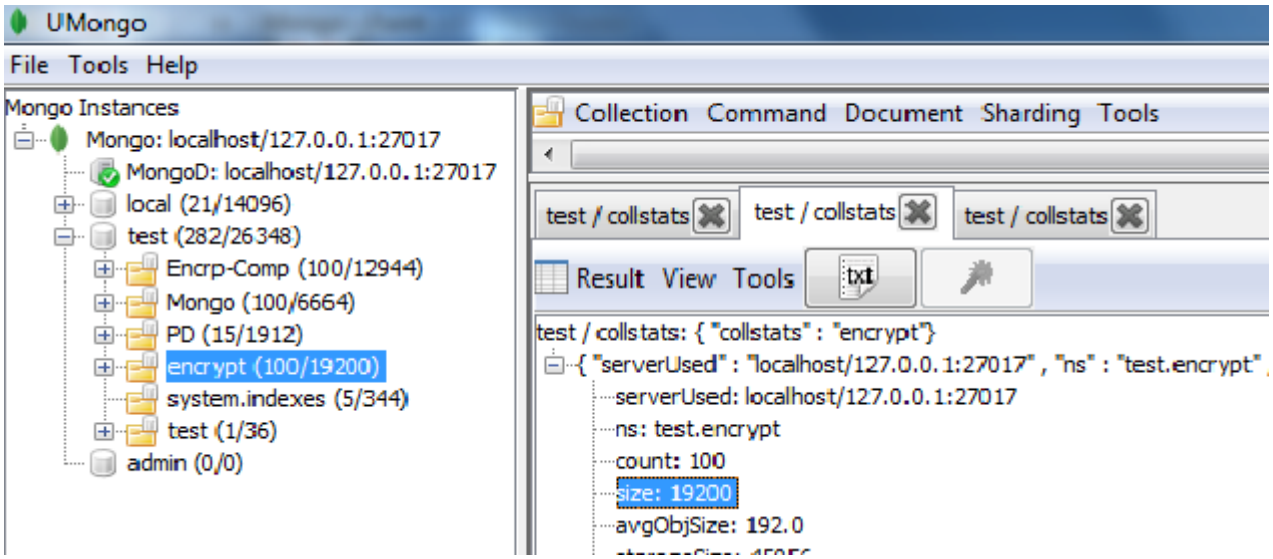


Fig. 23 Colección encrypt.

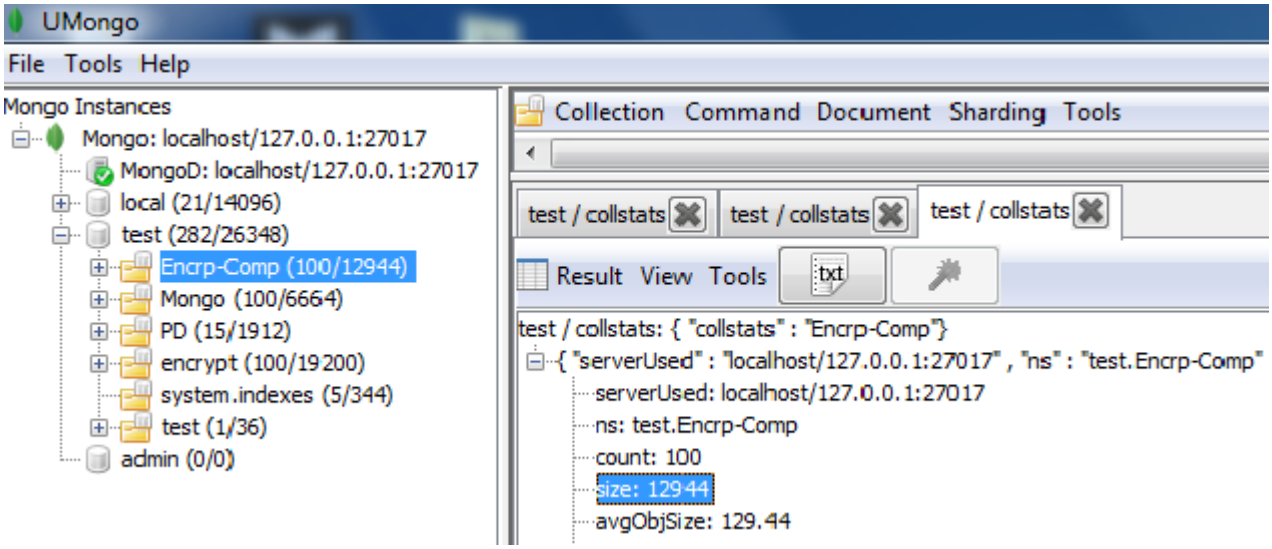


Fig. 24 Colección Encrp-Comp.

Para un mejor entendimiento de los resultados arrojados en las pruebas de funcionalidad se muestran los mismos en la siguiente tabla:

Tabla 5 Resultado de las pruebas de funcionalidad.

Colecciones	Cantidad de documentos	Tamaño promedio de cada documento(byte)	Tamaño total de la colección(byte)	Diferencia
Mongo	100	66.64	6664	-
encrypt	100	192.00	19 200	12 336
Encrp-Comp	100	129.44	12 944	6280

Luego de haber estudiado los resultados anteriores se concluye que los documentos cifrados tienen un aumento considerable con respecto a los documentos almacenados en texto plano el cuál se reduce en un 50% aproximadamente al utilizarse Monkript, existiendo aún una diferencia con respecto a los documentos sin cifrar pero siempre que se cifre un documento aumenta potencialmente el tamaño del mismo.

Después de someter el código de la biblioteca Monkript a distintas pruebas se pudo comprobar que tiene un buen funcionamiento. Rectificándose los errores arrojados en cada caso de prueba aplicado, quedando validada además la correcta implementación de los requisitos funcionales especificados.

Conclusiones parciales del capítulo

En este capítulo se sentaron las bases para la implementación definiendo un estándar de codificación adecuado que permitió el entendimiento y claridad del código. Se explicaron los principales métodos implementados. También se obtuvo en el capítulo los resultados de las pruebas de caja blanca mediante la técnica del camino básico realizada al código y los resultados de las pruebas aplicadas con la herramienta JUnit validando una correcta implementación del código fuente. Por último se verificaron las ventajas del producto mediante pruebas de funcionalidad.

Conclusiones Generales

Como resultado de la investigación se desarrolló el diseño y la implementación de una biblioteca de cifrado y compresión de datos que protege y disminuye el tamaño de información en MongoDB, arribando a las siguientes conclusiones:

- El estudio y desarrollo de la fundamentación teórica garantizó definir los algoritmos de cifrado Blowfish y compresión Snappy a utilizar, dando inicio a la elaboración de la biblioteca una vez seleccionadas las herramientas, tecnologías y metodología de desarrollo de software a utilizar.
- Se describieron las características de la biblioteca identificando 5 tipos de requisitos no funcionales y 8 funcionales agrupados en 5 casos de uso de alta criticidad.
- El modelo de diseño de Monkript, proporcionó una entrada para la implementación pues permitió describir cómo quedarían implementadas las clases y la comunicación existente entre ellas bajo una arquitectura en 2 capas y los patrones de diseño GRASP.
- Se validó la propuesta de biblioteca, a partir de las pruebas de caja blanca con la utilización de la herramienta JUnit permitiendo detectar los posibles errores y se realizaron además pruebas de funcionalidad, obteniendo un producto con calidad.

Recomendaciones

- Valorar la inclusión en la biblioteca Monkript de nuevos algoritmos de cifrado.

- Continuar profundizando esta investigación, con el objetivo de incorporar a la biblioteca las nuevas funcionalidades de las versiones superiores a la 2.4.4 de MongoDB.

Referencias Bibliográficas

1. **VILLAR Varela , Ana María y FERNÁNDEZ Pérez, Carlos .** Base de Datos Access: Aplicaciones Fundamentales y Manejo Básico de una Base de Datos. España : [s.n.], 2006. 39 p.

2. **CONTRERAS Escobar, Evis y VALDEZ Jiménez, Jorge Risquet.** EL SISTEMA DE INFORMACION DE MARKETING EN UNA EMPRESA CUBANA. Cuba : [s.n.], 2004.

3. ***RAMIÓ Aguirre , Jorge*.** SEGURIDAD INFORMÁTICA Y CRIPTOGRAFÍA. Madrid : [s.n.], 2006.

4.**Real Academia Española.** rae.es. [En línea] [Citado el: 20 de marzo de 2014.] http://www.rae.es.

5. **RIBAGORDA Garnacho, Arturo y AREITIO Bertolín, Javier.** Una breve panoramica de la criptografia. *Novática*, (172), Noviembre -diciembre 2004.

6. **SIMMONS, Gustavus J.** Contemporary Cryptology: The Science of Information Integrity. New York : [s.n.], 1994. ISBN:0879422777.

7. **ROMAN González , Avid.** Clasificación de datos basado en compresión. *ECIPeru* (9): 69-74, 2012.

8. **KAHATE, Atul.** CRYPTOGRAPHY AND NETWORK SECURITY. India : [s.n.], 2003. 62 p.

9. **STALLINGS, William.** Fundamentos de seguridad en redes: aplicaciones y estándares. 2da. ed. . 2004. pp 31-44.

10. **Mousa, A.** Data encryption performance based on Blowfish. En: International Symposium (47th: 2005: Zador). ELMAR. Zador : [s.n.], 2005. pp 131-134.

11. **SALOMÓN, David.** Data Compression. 4a. ed. London : Springer, 2007. pp 72-95.

12. **Saito, Taro L.** xerial.org. [En línea] [Citado el: 30 de noviembre de 2013.] http://xerial.org/snappy-java.

13. **DesaNova Ltda.** molebox.com. [En línea] [Citado el: 2 de diciembre de 2013.] http://www.molebox.com/molebox-vs.shtml.

14. **Network, Software Pick.** softpicks. [En línea] [Citado el: 30 de noviembre de 2013.] http://softpicks.com.es/software/Desarrollo/Miscelaneos/MoleBox-Pro_es-77394.htm.

15. **Oreans Technologies.** oreans.com. [En línea] [Citado el: 29 de noviembre de 2013.] http://www.oreans.com/es/xbundler.php.

16. **sharpziplib.** sharpziplib.com. [En línea] [Citado el: 30 de noviembre de 2013.] http://www.sharpziplib.com/.

17. **PRESSMAN, Roger S.** *Ingeniería de Software: Un enfoque práctico.* 6a. ed. s.l. : McGraw-Hill. pp 226,420.

18. Inteco:Informe Anual de Actividad. Madrid : [s.n.], 2009. pp 58-60.

19. **KENT Beck, Martin Fowler.** Planning Extreme Programming. USA: [s.n.], 2000. pp 79-84.

20. **BALDUINO, Ricardo**. Configuracion de la metodologia Open UP**.** Sao Paulo : [s.n.], Agosto 2007.

21. **Oracle.** java.com. [En línea] 2010. [Citado el: 20 de noviembre de 2013.] http://java.com/es/about/.

22. **H, Swaroop C.** A Byte of Python. USA: [s.n.], 2013. pp 17-18.

23. **Python Software Foundation.** www.python.org. *www.python.org.* [En línea] [Citado el: 20 de noviembre de 2013.] http://python.org.

24. **DACONTA , Michael C.** "C++ Pointers and Dynamic Memory Management". [ed.] John Wiley & Sons. ISBN 0-471-04998-0.

25. **Ciencia y Técnica Administrativa.** www.cyta.com.ar. [En línea] [Citado el: 22 de noviembre de 2013.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

26. **PRESSMAN, Roger S.** Software Engineerin: A Practitioner´s Approach. 7a. ed. New YorK: McGraw-Hill, 2010. ISBN 978-0-07-337 597-7.

27. **NetBeans.** www.netbeans.org. [En línea] [Citado el: 24 de noviembre de 2013.] http://netbeans.org/index_es.html.

28. **CHODOROW, Kristina y DIROLF, Michael.** MongoDB: The Definitive Guide. s.l. : O'Reilly Media, 2010. ISBN: 978-1-449-38156-1.

29. **LARMAN, Craig.** UML y Patrones. Introducción al análisis y diseño orientado a objetos. s.l. : Prentice Hall, 1999. pp 11,40,193 - 221. ISBN: 970-17-0261-1.

30. **Mongodb.** docs.mongodb.org. [En línea] [Citado el: 20 de marzo de 2014.] http://docs.mongodb.org/manual/reference/glossary/#term-document].

31. **SOMMERVILLE, Ian.** Ingeniería del Software. 7a. ed. Madrid : Pearson Educación, 2005. pp 109,233. ISBN: 81-7829-074-5.

32. **JACOBSON, Ivar.** El proceso unificado de desarrollo software. Madrid : Series Editors, 2000.

33. **HOFMEISTER, Christine, NORD, Robert y SONI, Dilip.** Applied Software Architecture. s.l. : Addison Wesley, 2000.

34. **BRUEGGE, Bernd y DUTOIT, Allen.** Ingeniería de software orientado a objetos. s.l. : Prentice Hall - Pearson Educación, 2002. pp 326-369.

35. **A Hunt, D Thomas.** Pragmatic unit testing in Java with JUnit. 2004.

Bibliografía

1. **VILLAR Varela , Ana María y FERNÁNDEZ Pérez, Carlos .** Base de Datos Access: Aplicaciones Fundamentales y Manejo Básico de una Base de Datos. España : [s.n.], 2006. 39 p.

2. **CONTRERAS Escobar, Evis y VALDEZ Jiménez, Jorge Risquet.** EL SISTEMA DE INFORMACION DE MARKETING EN UNA EMPRESA CUBANA. Cuba : [s.n.], 2004.

3. ***RAMIÓ Aguirre , Jorge*.** SEGURIDAD INFORMÁTICA Y CRIPTOGRAFÍA. Madrid : [s.n.], 2006.

4.**Real Academia Española.** rae.es. [En línea] [Citado el: 20 de marzo de 2014.] http://www.rae.es.

5. **RIBAGORDA Garnacho, Arturo y AREITIO Bertolín, Javier.** Una breve panoramica de la criptografia. *Novática*, (172), Noviembre -diciembre 2004.

6. **SIMMONS, Gustavus J.** Contemporary Cryptology: The Science of Information Integrity. New York : [s.n.], 1994. ISBN:0879422777.

7. **ROMAN González , Avid.** Clasificación de datos basado en compresión. *ECIPeru* (9): 69-74, 2012.

8. **KAHATE, Atul.** CRYPTOGRAPHY AND NETWORK SECURITY. India : [s.n.], 2003. 62 p.

9. **STALLINGS, William.** Fundamentos de seguridad en redes: aplicaciones y estándares. 2da. ed. . 2004. pp 31-44.

10. **Mousa, A.** Data encryption performance based on Blowfish. En: International Symposium (47th: 2005: Zador). ELMAR. Zador : [s.n.], 2005. pp 131-134.

11. **SALOMÓN, David.** Data Compression. 4a. ed. London : Springer, 2007. pp 72-95.

12. **Saito, Taro L.** xerial.org. [En línea] [Citado el: 30 de noviembre de 2013.] http://xerial.org/snappy-java.

13. **DesaNova Ltda.** molebox.com. [En línea] [Citado el: 2 de diciembre de 2013.] http://www.molebox.com/molebox-vs.shtml.

14. **Network, Software Pick.** softpicks. [En línea] [Citado el: 30 de noviembre de 2013.] http://softpicks.com.es/software/Desarrollo/Miscelaneos/MoleBox-Pro_es-77394.htm.

15. **Oreans Technologies.** oreans.com. [En línea] [Citado el: 29 de noviembre de 2013.] http://www.oreans.com/es/xbundler.php.

16. **sharpziplib.** sharpziplib.com. [En línea] [Citado el: 30 de noviembre de 2013.] http://www.sharpziplib.com/.

17. **PRESSMAN, Roger S.** *Ingeniería de Software: Un enfoque práctico.* 6a. ed. s.l. : McGraw-Hill. pp 226,420.

18. Inteco:Informe Anual de Actividad. Madrid : [s.n.], 2009. pp 58-60.

19. **KENT Beck, Martin Fowler.** Planning Extreme Programming. USA: [s.n.], 2000. pp 79-84.

20. **BALDUINO, Ricardo**. Configuracion de la metodologia Open UP**.** Sao Paulo : [s.n.], Agosto 2007.

21. **Oracle.** java.com. [En línea] 2010. [Citado el: 20 de noviembre de 2013.] http://java.com/es/about/.

22. **H, Swaroop C.** A Byte of Python. USA: [s.n.], 2013. pp 17-18.

23. **Python Software Foundation.** www.python.o*rg. www.python.org. [En línea] [Citado el: 20 de noviembre de 2013.]* http://python.org.

24. **DACONTA , Michael C.** "C++ Pointers and Dynamic Memory Management". [ed.] John Wiley & Sons. ISBN 0-471-04998-0.

25. **Ciencia y Técnica Administrativa.** www.cyta.com.ar. [En línea] [Citado el: 22 de noviembre de 2013.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

26. **PRESSMAN, Roger S.** Software Engineerin: A Practitioner´s Approach. 7a. ed. New YorK: McGraw-Hill, 2010. ISBN 978-0-07-337 597-7.

27. **NetBeans.** www.netbeans.org. [En línea] [Citado el: 24 de noviembre de 2013.] http://netbeans.org/index_es.html.

28. **CHODOROW, Kristina y DIROLF, Michael.** MongoDB: The Definitive Guide. s.l. : O'Reilly Media, 2010. ISBN: 978-1-449-38156-1.

29. **LARMAN, Craig.** UML y Patrones. Introducción al análisis y diseño orientado a objetos. s.l. : Prentice Hall, 1999. pp 11,40,193 - 221. ISBN: 970-17-0261-1.

30. **Mongodb.** docs.mongodb.org. [En línea] [Citado el: 20 de marzo de 2014.] http://docs.mongodb.org/manual/reference/glossary/#term-document].

31. **SOMMERVILLE, Ian.** Ingeniería del Software. 7a. ed. Madrid : Pearson Educación, 2005. pp 109,233. ISBN: 81-7829-074-5.

32. **JACOBSON, Ivar.** El proceso unificado de desarrollo software. Madrid : Series Editors, 2000.

33. **HOFMEISTER, Christine, NORD, Robert y SONI, Dilip.** Applied Software Architecture. s.l. : Addison Wesley, 2000.

34. **BRUEGGE, Bernd y DUTOIT, Allen.** Ingeniería de software orientado a objetos. s.l. : Prentice Hall - Pearson Educación, 2002. pp 326-369.

35. **A Hunt, D Thomas.** Pragmatic unit testing in Java with JUnit. 2004.

36. recursos.cnice.mec.es. [En línea] 29 de noviembre de 2013. http://recursos.cnice.mec.es/lengua/profesores/eso1/t1/teoria_1.htm.

37. **MENEZES, A., van OORSCHOT, P. y VANSTONE, S.** Handbook of Applied Cryptography. s.l. : CRC Press, 1996.

38. **VISCONTI , Marcello y ASTUDILLO, Hernán.** Fundamentos de Ingenieria de Sofware. [Presentación] Chile : [s.n.] . Patrones de Diseño.

39. **MONNÉ Roque, Diana y MENDOZA Garnache, Alberto.** OpenUP como metodología desarrollo de sofware aplicada en el desarrollo de los productos del departamento Integración de Soluciones. DATEC (UCI). La Habana : [s.n.], 2012.

40. **BALDUINO, Ricardo** Introduction to OpenUP**.** 2007.

41. **GONZÁLEZ Arechavala, Yolanda y de CUADRA García, Fernando.** Calidad del Software (I). España : Asociación de ingenieros de ICAI. *Anales de mecánica y electricidad*, 78 (5): 54-60, Octubre 2001.

42. **VAUDENAY, Serge.** On the Weak Keys of Blowfish. Paris : [s.n.].

43. **ALCOVER, Pedro M., SUARDÍAZ, Juan y NAVARRO, Pedro J.** Adaptación de la docencia de una asignatura de criptografía a las recomendaciones del Espacio Europeo de Educación Superior. *IEEE-RITA*, 4 (2), Mayo 2009.

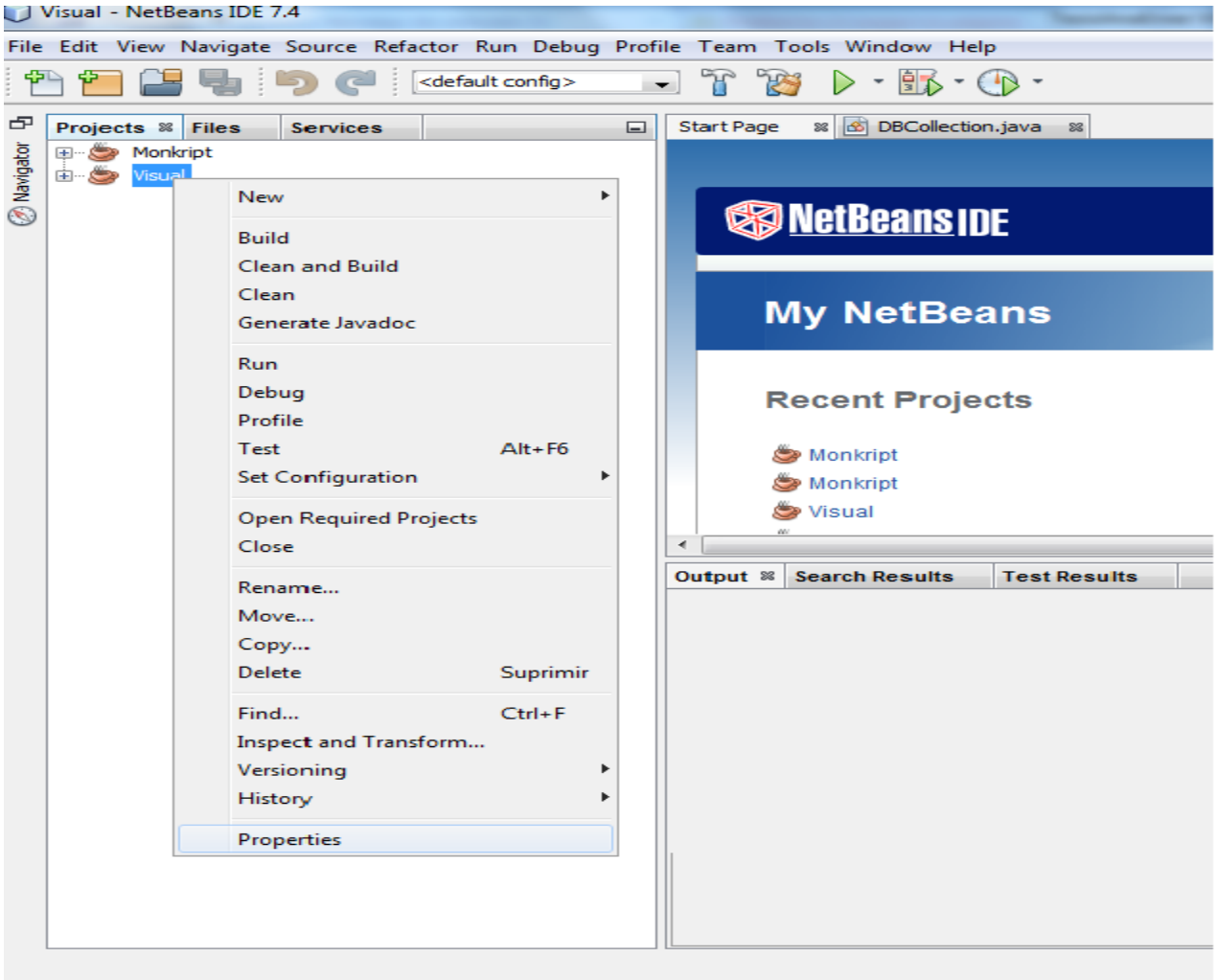
44. . **COELLO Coello, Carlos Artemio y HERNÁNDEZ de León, Héctor Ricardo.** Compresión de Base de Datos. Mexico : [s.n.].

Anexos

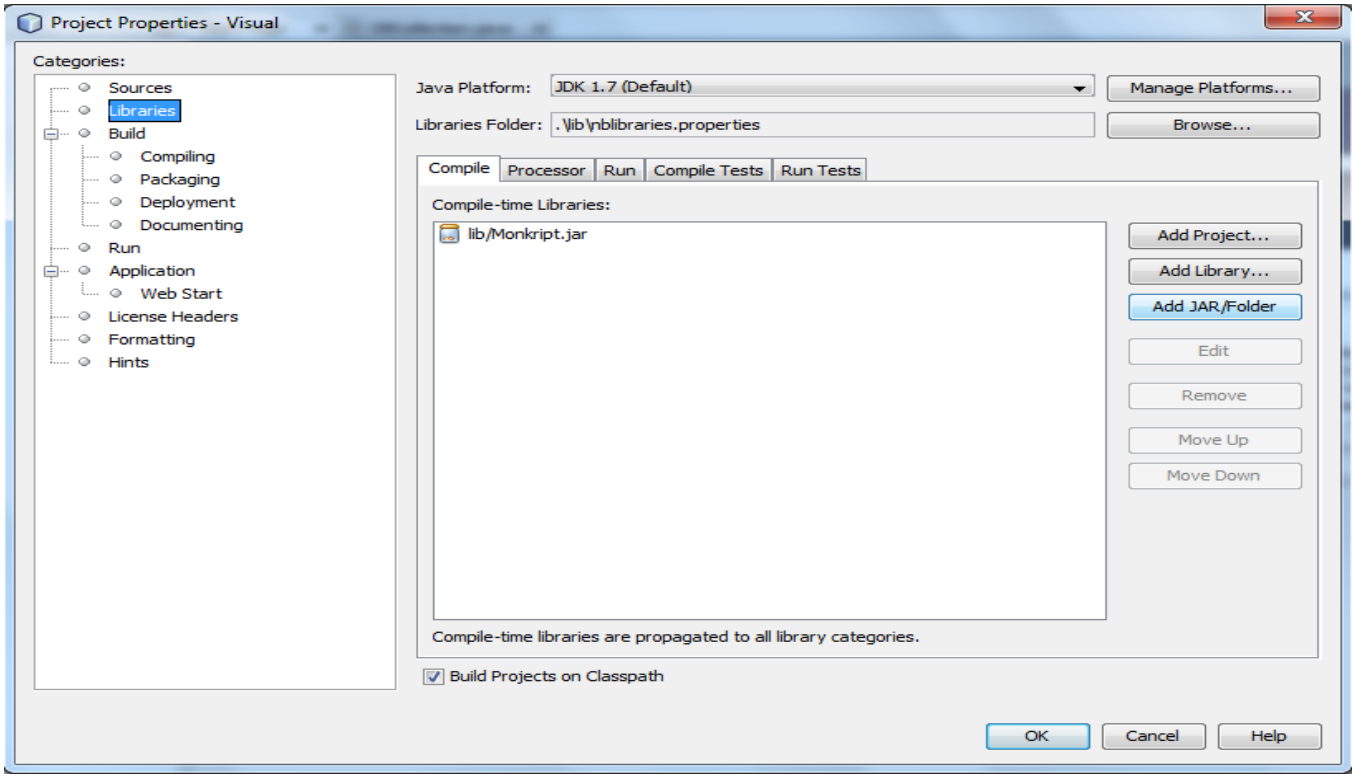
Anexo 1: Integración de Aplicación java con Monkript

Ejemplo realizado en Netbeans 7.4 y JDK 1.7

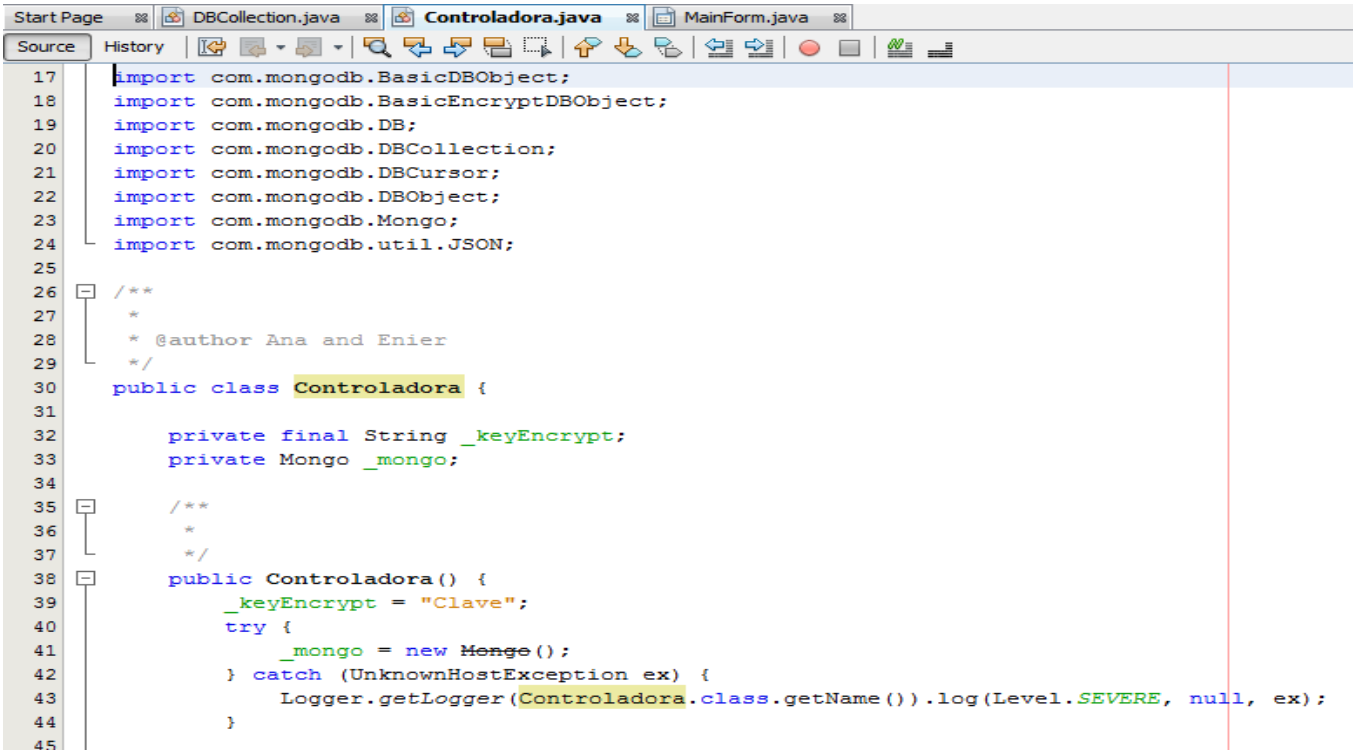
1. Adicionar la librería Monkript como se muestra en las siguientes imágenes:



Adicionar Monkript igualmente en las pestañas Processor y Run.



2. Importar las clases necesarias para el uso de la librería.



Anexo 2: Descripción de los Casos de Uso del Sistema

Caso de Uso	Descifrar documentos almacenados en una colección de MongoDB.
Actores	Aplicación Java “(Inicia)”
Objetivo	Descifrar un documento cifrado que este almacenado en una colección de MongoDB.
Resumen	El caso de uso inicia cuando la aplicación una vez conectada al gestor MongoDB, haciendo uso del driver, desee descifrar un documento; finalizando el mismo cuando la acción haya sido realizada.
Precondiciones	Que exista la conexión con el gestor y el documento cifrado.
Referencias	RF 5
Prioridad	Crítico

Caso de Uso	Comprimir un documento almacenado en una colección de MongoDB.
Actores	Aplicación Java “(Inicia)”
Objetivo	Comprimir un documento almacenado en una colección de MongoDB.
Resumen	El caso de uso inicia cuando la aplicación una vez conectada al gestor MongoDB, haciendo uso del driver, cifra un documento, entonces se comprime y se almacena cifrado y comprimido en una colección; finalizando el mismo cuando la acción haya sido realizada.
Precondiciones	Que exista la conexión con el gestor y que exista la colección creada.
Referencias	RF 7
Prioridad	Crítico

Caso de Uso	Descomprimir un documento almacenado en una colección de MongoDB.
Actores	Aplicación Java “(Inicia)”
Objetivo	Descomprimir un documento almacenado en una colección de MongoDB.
Resumen	El caso de uso inicia cuando la aplicación una vez conectada al gestor MongoDB, haciendo uso del driver, desea consultar un documento,

	entonces este se descomprime para poder operar con su información; finalizando el mismo cuando la acción haya sido realizada.
Precondiciones	Que exista la conexión con el gestor y que exista el documento.
Referencias	RF 8
Prioridad	Crítico

Anexo 3: Implementación de la clase BasicEncryptDBObjectIT

```
public class BasicEncryptDBObjectIT {

    public BasicEncryptDBObjectIT() {...2 lines }

    @BeforeClass
    public static void setUpClass() {...2 lines }

    @AfterClass
    public static void tearDownClass() {...2 lines }

    @Before
    public void setUp() {...2 lines }

    @After
    public void tearDown() {...2 lines }

    /** Test of append method, of class BasicEncryptDBObject ...3 lines */
    @Test
    public void testAppend() {
        System.out.println("Prueba al método append");
        String key = "name";
        Object val = "Monkript";
        String keyEncript = "key encript";
        BasicEncryptDBObject instance = new BasicEncryptDBObject(keyEncript);
        BasicEncryptDBObject expResult = new BasicEncryptDBObject(keyEncript).append(key, val
        BasicEncryptDBObject result = instance.append(key, val);
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        //fail("The test case is a prototype.");
    }
    /** Test of put method, of class BasicEncryptDBObject ...3 lines */
    @Test
    public void testPut_String_Object() throws IOException {
        System.out.println("Prueba al método put (con clave y valor sin hacerle el proceso de
        String key = "name";
        Object val = "Monkript";
        String keyEncript = "key encript";
        BasicEncryptDBObject instance = new BasicEncryptDBObject(keyEncript);
        Object expResult = new BasicEncryptDBObject(keyEncript).put(key, val);
        Object result = instance.put(key, val);
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        //fail("The test case is a prototype.");
    }

    /** Test of getBlowfish method, of class BasicEncryptDBObject ...3 lines */
    @Test
    public void testGetBlowfish() {
        System.out.println("Prueba al método getBlowfish");
        BasicEncryptDBObject instance = new BasicEncryptDBObject("Monkript");
        Blowfish expResult = instance.getBlowfish();
        Blowfish result = instance.getBlowfish();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        //fail("The test case is a prototype.");
    }
    /** Test of put method, of class BasicEncryptDBObject ...3 lines */
    @Test
    public void testPut_3args() throws Exception {
        System.out.println("Prueba al método put (para encriptar)");
        String key = "name";
        Object val = "Monkript";
        String keyEncript = "key encript";
        boolean encryct = true;
        BasicEncryptDBObject instance = new BasicEncryptDBObject(keyEncript);
        Object expResult = new BasicEncryptDBObject(key, val, keyEncript);
        Object result = instance.put(key, val, encryct);
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        //fail("The test case is a prototype.");
    }

}
```