Documentación

del Gestor de Contraseñas

Santiago Alejandro Chaparro Palacio

Ingeniería de Sistemas

Pontificia Universidad Javeriana

Bogotá D.C., Colombia

[santiagochaparro@javeriana.edu.co](mailto:santiagochaparro@javeriana.edu.co)

Diego Gerardo Barajas Suarez

Ingeniería de Sistemas

Pontificia Universidad Javeriana

Bogotá D.C., Colombia

[barajas-d@javeriana.edu.co](mailto:barajas-d@javeriana.edu.co)

Laura Marcela Sánchez Peláez

Ingeniería Informática del Software

Universidad de Oviedo

Oviedo, España

[sanchezplaura@javerina.edu.c](mailto:sanchezplaura@javerina.edu.co)o

***Resumen*—Este documento detalla la información técnica del gestor de contraseñas creado por Santiago Chaparro, Diego Barajas y Laura Sánchez a partir de los criterios dados para la aplicación.**

**Palabras clave — Gestor de Contraseñas, Interfaz, Paquetes, Contraseña, Seguridad, Cifrado**

1. Introducción

El presente documento tiene como objetivo presentar todo los aspectos técnicos que se tuvieron que tener en cuenta en el desarrollo del gestor de contraseñas, así como exponer como de que está compuesto el mismo, y como todas sus partes se relacionan entre sí.

II. Aspectos Tecnicos

El programa se ha realizado en lenguaje java siguiendo un patrón singleton o instancia única, es un patrón de diseño que permite restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.[2] Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

El proyecto sigue una arquitectura en capas, un modelo de desarrollo software en el que el objetivo primordial es la separación (desacoplamiento) de las partes que componen un sistema software: lógica de negocios, capa de presentación y capa de datos. De esta forma, por ejemplo, es sencillo y mantenible crear diferentes interfaces sobre un mismo sistema sin requerir cambio alguno en la capa de datos o lógica.[1]

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y en caso de que sobrevenga algún cambio, solo afectará al nivel requerido sin tener que revisar entre el código fuente de otros módulos, dado que se habrá reducido el Acoplamiento informático hasta una interfaz de paso de mensajes.

Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

Gestor de Contraseñas consta de tres capas, presentación( package igu), negocio ( package servicio), datos( package persistencia), además hay un package modelo que incluye los modelos con los que se va a trabajar, el Usuario y Registros, el usuario posee un conjunto de registros que tendrán un identificador (id), además de sus atributos.

Se tiene un package Utils que incluye clases que nos ayuda a la encriptación de la información sensible.

**A. Capa de presentación:** La que tiene contacto directo con el usuario, este le comunica la información y toma los datos del usuario para su proceso. Esta capa interactúa solo con la capa de negocio.

* Librerías Utilizadas para la GUI JFrame

En la parte de la interfaz gráfica del gestor de contraseñas se utilizó la librería **Swing**, el cual provee varias herramientas (en forma de clases) para realizar interfaces sencillas, pero suficientes para llevar a cabo el gestor de contraseñas. Cabe resaltar que para el manejo de información sensible, (las contraseñas) swing provee de una clase (parecida a JLabel) especial llamada JPassword, cuya presentación en la interfaz gráfica y funciones en el código tienen cierto grado de seguridad.

* Manejo de las contraseñas

Durante las operaciones que realiza el programa, se utiliza la contraseña maestra para acceder a la funcionalidad principal de la aplicación. Durante el inicio de sesión de la aplicación el usuario debe digitar su contraseña en el campo generado por el JPassword, que posee varias funciones para extraer lo que digito dicho usuario en el Label. Asimismo, el sistema en también tiene que descifrar la contraseña de los registros. En estos casos es importante tener en cuenta el aspecto técnico de la aplicación puesto a que la confidencialidad de esta información es importante. Para maximizar la seguridad de la aplicación se decidió :

1. Descifrar las contraseñas solo cuando realmente sea necesario. Por ejemplo, en el inicio de sesión, nunca se descifra la contraseña que está en la persistencia; lo que se hace es comparar el cifrado de la cadena de texto que el usuario introduce con el cifrado de la contraseña de la base de datos.
2. Manejar vectores de caracteres, no cadena de caracteres. En el lenguaje de programación seleccionado para el proyecto, java, los objetos creados no son posible borrarlos utilizando alguna linea de codigo, solo se borra cuando el sistema quiere limpiar memoria. Las cadenas de caracteres se manejan como objetos cuya clase es String, manejar las contraseñas en esa clase puede conllevar a que si se realiza un memory dump, se pueda ver en texto claro la contraseña. Para solucionar esta situación, el JPasswordField posee una función que convierte el texto que el usuario escribe en el Label en una vector de caracteres. Esto tiene como ventaja de que cuando se termine de utilizar la información dentro del vector, se pueda eliminar a voluntad de este.

**B. Capa de negocio:** Se establecen las clases y los métodos que se ejecutan, se reciben los datos del usuario y se envían las respuesta tras el proceso. Esta capa interacciona con con la capa de presentación y la capa de persistencia o datos, así hace la comunicación con la base de datos y desarrolla el CRUD de la aplicación.

La capa de negocio se encuentra en el package servicios, quien tiene la comunicación única con la capa de datos (package persistencia)

**C. Capa de datos:** Interactúa con la base de datos, encargada de extraerlos e introducirlos.

La capa de datos se encuentra en el package persistencia, quien tiene la comunicación con los modelos Registro y Usuario, además, capa de datos contiene todo el CRUD( CRUD es el acrónimo de "Crear, Leer, Actualizar y Borrar" (del original en inglés: Create, Read, Update and Delete), que se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.) [3] del sistema de gestión de contraseñas.

La base de datos se encuentra en un **archivo de texto plano .txt** donde todos los datos que son solicitados y registrados son cifrados con el algoritmo AES.

La base de datos queda cifrada de esta manera:



Para el proceso de encriptacion y desencriptacion se utilizan las librerias:

**Apache Commons Codec**: Codec se formó como un intento de centrar el esfuerzo de desarrollo en una implementación definitiva del codificador Base64. En el momento de la propuesta de Codec, había aproximadamente 34 clases Java diferentes que se ocupaban de la codificación Base64 distribuida en el repositorio CVS de la Fundación. [4]  **javax.crypto**: Proporciona las clases e interfaces para operaciones criptográficas. Las operaciones criptográficas definidas en este paquete incluyen el cifrado, la generación de claves y el acuerdo de claves, y la generación del Código de autenticación de mensajes (MAC). La compatibilidad con el cifrado incluye cifrados simétricos, asimétricos, de bloque y de flujo. Este paquete también admite flujos seguros y objetos sellados. [5]

**Cipher:** Esta clase proporciona la funcionalidad de un cifrado criptográfico para el cifrado y descifrado. Forma el núcleo del marco de Java Cryptographic Extension (JCE).

Para crear un objeto Cipher, la aplicación llama al **getInstancemétodo** Cipher y le pasa el nombre de la transformación solicitada . Opcionalmente, se puede especificar el nombre de un proveedor.

Una transformación es una cadena que describe la operación (o conjunto de operaciones) que se realizará en la entrada dada, para producir alguna salida. Una transformación siempre incluye el nombre de un algoritmo criptográfico (por ejemplo, AES ), y puede ser seguido por un modo de retroalimentación y un esquema de relleno.

Una transformación es de la forma:

" algoritmo / modo / relleno " o

" algoritmo "

**SecretKeySpec**

implementa KeySpec , SecretKey

Esta clase especifica una clave secreta de forma independiente del proveedor.

Se puede usar para construir un conjunto SecretKeyde bytes, sin tener que pasar por un (basado en el proveedor) SecretKeyFactory.

###### IvParameterSpec

##### implementa AlgorithmParameterSpec

##### Esta clase especifica un vector de inicialización (IV).

##### La primera línea corresponde con la contraseña maestra, la segunda con la fecha de creación de esa contraseña maestra y las demás con los registros. Los registros tienen sus atributos delimitados por un “#”, la estructura sería la siguiente:

*id#titulo#nombreUsuario#contraseña#URL*

##### Desktop

Clase java que nos permite utilizar varias funciones del ordenador directamente desde código.

###### Esta clase nos permitió obtener el navegador por defecto del ordenador y lanzar una URL conjunta de una petición http con formato GET, dado que clases como URL de java no permiten abrir este tipo de formatos por el FormatExeption.[6]

##### Inserción de datos en inputs

###### La estructura típica para un formulario en HTML es la siguiente:

###### <form action=""> <input id="GET-name" type="text" name="name"> </form>

###### En donde se muestra los atributos name de los input, en este caso name, al momento de lanzar una petición desde este formulario la URL se modificaría de la siguiente manera

###### <URL\_original>?<nombre\_input>=”<valor que va a tomar el input>”

###### al llamar a la URL directa en donde se encuentra en form con este formato se realizará la operación inversa y se rellenaron los formularios con la información especificada.[7]

ANEXOS

Veasen los diagramas de clase en el documento “Diagramas de clase gestor de contraseñas” disponible [aqui](Diagramas%20de%20clase%20gestor%20de%20contraseñas.docx).

Conclusiones

###### Mantener buenas prácticas para la seguridad de la información es absolutamente necesario en la era de la tecnología en la cual nos encontramos actualmente, dado el inmenso número de personas que manejan información, desde las personas del común con cuentas de correo o información de sus trabajos hasta grandes compañías o gobiernos con información demasiado importante o trascendente a nivel global.

###### Al implementar aplicaciones desde la más sencilla hasta la más compleja siempre es necesario tener en cuenta la seguridad como uno de los pilares, si no se mantiene la información de las aplicaciones segura puede acarrear grandes problemas tanto para los desarrolladores como para los usuarios.

###### Las fallas de seguridad son algo inevitable, pues nada es completamente seguro, esto conlleva el echo que siempre se debe estar mejorando las medidas de seguridad para la información o podría ser contraproducente para nosotros.

Bibliografía

[1] https://es.wikipedia.org/wiki/Programaci%C3%B3n\_por\_capas

[2]https://es.wikipedia.org/wiki/Singleton

[3]https://es.wikipedia.org/wiki/CRUD

[4]https://commons.apache.org/proper/commons-codec/download\_codec.cgi

[5]<https://docs.oracle.com/javase/>

[6]<https://docs.oracle.com/javase/7/docs/api/java/awt/Desktop.html>

[7]<https://tools.ietf.org/html/rfc2616>