

# Documentación proyecto 2 – Gestor de contraseñas

2<sup>nd</sup> Nicolás Miranda

Departamento de Ingeniería de Sistemas  
Pontificia Universidad Javeriana  
Bogotá, Colombia  
nicolasmiranda@javeriana.edu.co

3<sup>rd</sup> Miguel Romero

Departamento de Ingeniería de Sistemas  
Pontificia Universidad Javeriana  
Bogotá, Colombia  
miguel-romero@javeriana.edu.co

**Abstract:** This article specifically shows all the components that were used for the correct development of the password manager project, describing the classes and methods that were used and a test scenario to show the operation.

**Keywords-***Cryptography, AES, hash and password manager*

## I. INTRODUCCIÓN

En el presente proyecto, se pretende poner en práctica todos los conceptos aprendidos sobre criptografía, en particular, criptografía moderna, donde se utilizó el algoritmo AES para cifrar los archivos y también se aplicó las funciones de hash para verificar la integridad del archivo y que las contraseñas concuerden. Por otro lado, este ejemplo práctico de un gestor de contraseña se ve claramente la importancia de la seguridad de la información

## II. DESARROLLO

Para el desarrollo del proyecto, el lenguaje de programación que se decidió fue Java, ya que incluía varias bibliotecas para el uso de criptografía. En esta sección se describirá las bibliotecas que se utilizaron para el desarrollo del proyecto.

En primer lugar, la biblioteca que fue de más importancia para el desarrollo del proyecto fue *Bouncy Castle* [1] que es una API de Java criptográfica, bastante útil a la hora de crear los métodos necesarios para cifrar y descifrar la información de los archivos. Por otro lado, el paquete de *java.security* [2] que proporciona las clases e interfaces para el framework de seguridad en Java. Finalmente, en cuanto a las biblioteca de mayor relevancia, el paquete *javax.crypto* [3] el cual proporciona las clases e interfaces para realizar las operaciones criptográficas.

Para el proyecto se decidió crear 3 paquetes con el fin de garantizar una arquitectura adecuada muy similar al modelo MVC, que es una de las arquitecturas más utilizadas en proyectos de desarrollo de software y nos ayuda a la asignación de responsabilidades y seguir patrones que están estandarizadas en la industria [4]. La figura 1 muestra los 3 paquetes que contienen las clases. El paquete de *negocios* tiene las clases relacionadas con la lógica de la programación, el paquete de *persistencia* tiene la clase con todos los métodos para controlar los archivos y por último el paquete de *presentación* el cual contiene todas las pantallas utilizadas para mostrar las diferentes funcionalidades del proyecto.

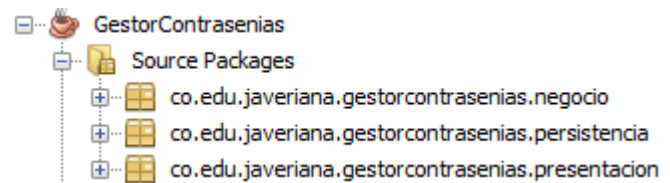


Figura 1. Paquetes del proyecto

## III. CLASES Y MÉTODOS

Como se comentó en la sección anterior, se utilizaron 3 paquetes para el desarrollo del proyecto. En la figura 2, se observa la clase *Sitio.java* que contiene los atributos que caracteriza a un sitio, los cuales son el nombre, url, user y contraseña

```
/**
 * Esta Clase contiene todos los atributos necesarios para caracterizar un sitio
 */
public class Sitio
{
    String nombre;
    String url;
    String user;
    String contraseña;

    public Sitio( String nombre, String url, String user, String contraseña )
    {
        this.nombre = nombre;
        this.url = url;
        this.user = user;
        this.contraseña = contraseña;
    }
}
```

Figura 2. Clase Sitio

La figura 3 y 4, se observa la clase *Utils* con algunos de los métodos que se implementaron como lo son *hash* y *cifrarAES*

```
/*
 * Método que cifra un archivo usando el algoritmo AES mediante CTR
 * (Counter) que convierte el cifrado de bloque en un cifrado de flujo
 */
public static byte[] cifrarAES( byte[] entrada, SecretKey key ) throws
{
    Security.addProvider( new BouncyCastleProvider() );
    Cipher aes = Cipher.getInstance( "AES/CTR/NoPadding", "BC" );

    SecureRandom random = new SecureRandom();
    byte[] iv = new byte[ aes.getBlockSize() ];
    random.nextBytes( iv );
    IvParameterSpec ivParametro = new IvParameterSpec( iv );

    aes.init( Cipher.ENCRYPT_MODE, key, ivParametro );
    byte[] cifrado = aes.doFinal( entrada );

    byte[] ivCifrado = Arrays.concatenate( iv, cifrado );

    return ivCifrado;
}
```

caracteriza a un sitio, los cuales son el nombre, url, user y contraseña

```
/**
 * Esta Clase contiene todos los atributos necesarios para caracterizar un sit
 */
public class Sitio
{
    String nombre;
    String url;
    String user;
    String contraseña;

    public Sitio( String nombre, String url, String user, String contraseña )
    {
        this.nombre = nombre;
        this.url = url;
        this.user = user;
        this.contrasenia = contraseña;
    }
}
```

Figura 2. Clase Sitio

La figura 3 y 4, se observa la clase *Utils* con algunos de los métodos que se implementaron como lo son *hash* y *cifrarAES*

```
/**
 * Método que cifra un archivo usando el algoritmo AES mediante CTR
 * (Counter) que convierte el cifrado de bloque en un cifrado de flujo
 */
public static byte[] cifrarAES( byte[] entrada, SecretKey key ) throws
{
    Security.addProvider( new BouncyCastleProvider() );
    Cipher aes = Cipher.getInstance( "AES/CTR/NoPadding", "BC" );

    SecureRandom random = new SecureRandom();
    byte[] iv = new byte[ aes.getBlockSize() ];
    random.nextBytes( iv );
    IvParameterSpec ivParametro = new IvParameterSpec( iv );

    aes.init( Cipher.ENCRYPT_MODE, key, ivParametro );
    byte[] cifrado = aes.doFinal( entrada );

    byte[] ivCifrado = Arrays.concatenate( iv, cifrado );

    return ivCifrado;
}
```

Figura 3. Clase Utils

```
import javax.crypto.*;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.KeySpec;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.spec.SecretKeySpec;

import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.util.Arrays;

/**
 * Clase con los métodos principales de criptografía disponibles en Java.
 * Adaptado de: https://github.com/kdenhartog/Password-Manager/blob/master/SecurityFunction.java
 */
public class Utils
{
    /**
     * Método que transforma un mensaje en una nueva serie de caracteres con
     * una longitud fija, en este caso se utiliza la función de hash SHA-512
     */
    public static byte[] hash( byte[] mensaje ) throws NoSuchAlgorithmException, NoSuchProviderException
    {
        Security.addProvider( new BouncyCastleProvider() );
        MessageDigest mda = MessageDigest.getInstance( "SHA-512", "BC" );

        return mda.digest( mensaje );
    }
}
```

Figura 4. Clase Utils

Por último, la figura 5 se observa la clase *ManejoArchivo.Java* que contiene todos los métodos

relacionados con la lectura, escritura y cifrado de los archivos, usando los métodos de la clase *Utils.java*

```
public class ManejoArchivos
{
    /**
     * Variables globales que permiten realizarlas funciones criptográficas que
     * se van a utilizar en cada uno de los métodos
     */
    public static SecretKey cifLlave;
    public static SecretKey macLlave;
    public static byte[] macSalt;
    public static byte[] cifSalt;

    public ManejoArchivos()
    { }

    /**
     * Método que valida si el archivo de contraseña maestra y el archivo de
     * sitios existe o no existe
     */
    public static boolean verificarExistenciaArchivo()
    {
        String archivoContraseniaMaestraPath = System.getProperty( "user.dir" );
        archivoContraseniaMaestraPath += "/contraseniaMaestra";

        String archivoSitiosPath = System.getProperty( "user.dir" );
        archivoSitiosPath += "/sitios";

        File archivoContraseniaMaestra = new File( archivoContraseniaMaestraPath );
        File archivoSitios = new File( archivoSitiosPath );

        return ( archivoContraseniaMaestra.exists() && archivoSitios.exists() );
    }
}
```

Figura 5. Clase ManejoArchivo

## IV. FUNCIONAMIENTO

La aplicación comienza revisando si tiene registro de alguna contraseña maestra. En caso de que haya una contraseña almacenada se le pide al usuario que ingrese su clave maestra. Si no hay registro de ninguna contraseña el programa solicita que cree una contraseña la cual se va a almacenar y será la contraseña la usará después. Cabe resaltar que esta contraseña debe cumplir con las buenas prácticas para las contraseñas y que en caso de que estos no cumplan con los requerimientos que se le piden para la creación de la contraseña no se dejará continuar al usuario. Para la verificación de las contraseñas se hizo uso de una expresión regular en la que se verifican que haya al menos una mayúscula, una minúscula, un dígito, un carácter especial y que tenga como mínimo 8 caracteres. La figura 6, muestra la pantalla inicial de registrarse y caso de no existir los archivos de contraseña maestra y sitios.

Figura 6. Pantalla registro

La figura 7, es la pantalla de inicio de sesión cuando el usuario ya dispone de una contraseña maestra, donde solo debe ingresar la contraseña y el programa automáticamente validará

los hashes, en caso de ser igual lo dejará ingresar y en caso de no, volverá a solicitar la contraseña.

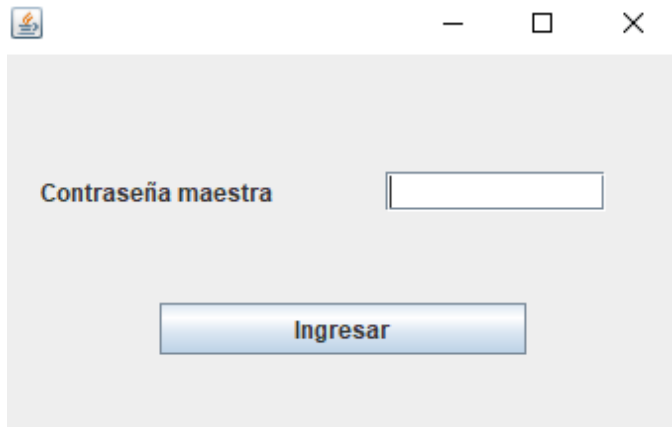


Figura 7. Pantalla inicio sesion

Una vez que el usuario ya tiene la contraseña será redirigido a la página de SesionIniciada, como se puede observar en la figura 8, en la cual podrá ver herramienta bastante útil con el tiempo y también la contraseña maestra.



| Sus datos son: |        |         |                 |                     |
|----------------|--------|---------|-----------------|---------------------|
| Nombre         | URL    | Usuario | Contraseña      | Tiempo de caduci... |
| Maestra        | NO URL | 0       | Comando\$2019-3 | 180000              |

Buttons: Crear Ingreso, Exportar a un archivo, Importar desde un archivo, Cambiar contraseña, Cerrar Sesion

Figura 8. Pantalla general

Dentro de la pantalla general, está la opción de crear un sitio, que se puede observar en la figura 9, donde se solicita la información relacionada con el sitio y con esta, se llena la tabla de sitios que el usuario ha registrado.

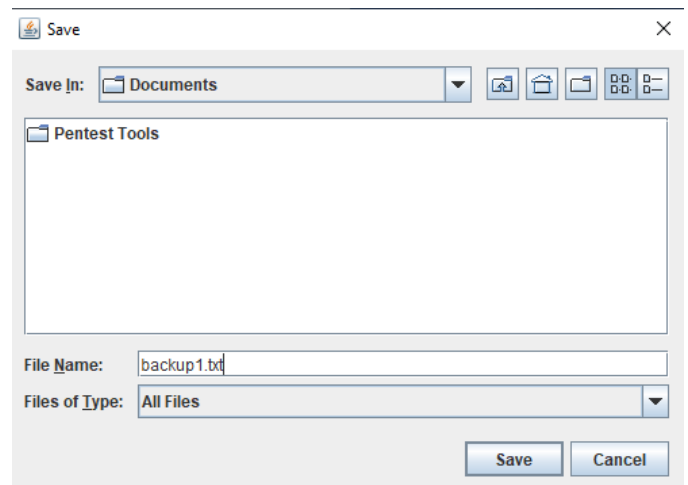


Form fields: Nombre (facebook), URL (facebook.com), Usuario (nicolasmirandam101), Contraseña (ISI-2019-3)

Button: Crear nuevo sitio

Figura 9. Pantalla crear sitio

La aplicación cuenta con las opciones para exportar, figura 10, e importar, figura 11, las contraseñas de las páginas web que el usuario agregue. El usuario por medio de la selección de carpetas puede escoger la ubicación para guardar los archivos según su preferencia y en el caso de importar no hay problema con la ubicación del archivo. Estos archivos van encriptados con por medio del cifrado AES. El cifrado y descifrado se realiza en base 64 y se guarda y lee desde un archivo de texto. Para este proceso es necesaria una llave maestra la cual es definida por nosotros y no por el usuario. Esta llave maestra no es necesario que cumpla con las buenas prácticas para las contraseñas ya que el cifrado AES se encarga de realizar el cambio de texto normal a texto cifrado. Todo esto es realizado desde la clase EncriptadorAES.



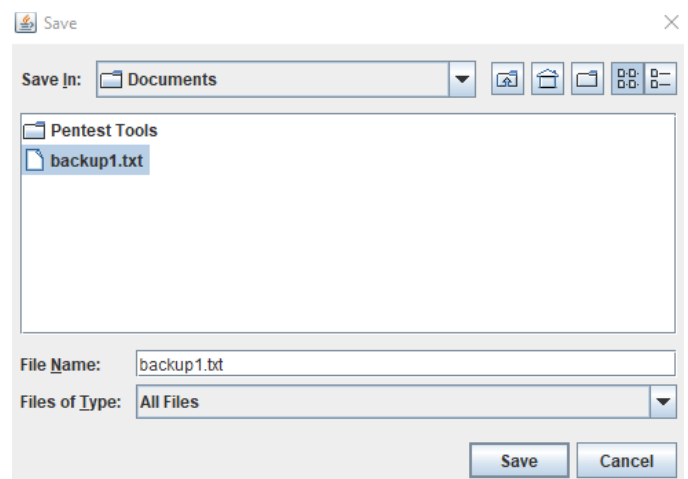
Save In: Documents

File Name: backup1.txt

Files of Type: All Files

Buttons: Save, Cancel

Figura 10. Pantalla importar archivo



Save In: Documents

File Name: backup1.txt

Files of Type: All Files

Buttons: Save, Cancel

Figura 11. Pantalla exportar archivo

Otra de las funcionalidades de la aplicación es la de generar un mensaje que le recuerde al usuario después de un tiempo determinado establecido por la aplicación que realice el cambio de la contraseña maestra como se puede observar en la figura 12 para lo cual la aplicación cuenta con un botón el cual abre otra ventana en la cual se le pide al usuario que ingrese la nueva contraseña. La nueva contraseña debe cumplir con las buenas prácticas para una contraseña segura.

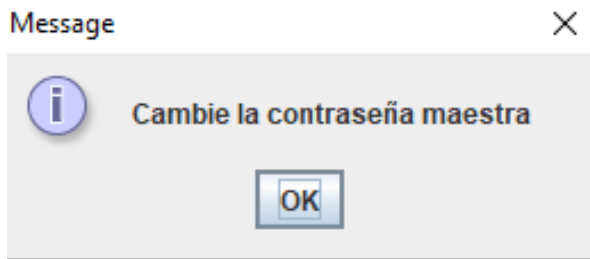


Figura 12. Aviso cambio contraseña

Por último, la pantalla de SesiónIniciada cuenta con un botón de cerrar sesión el cual al ser utilizado cierra la sesión y abre una ventana para que el usuario vuelva a Login. Los datos almacenados en la tabla sino fueron exportados a un archivo serán borrados, pero si se exportaron el archivo que se crea se mantiene en la carpeta seleccionada y al abrir la sesión se podrán importar. Un método que se implementó pero que no sirvió fue el del evento *mouseClicked* con el cual se pretendía tomar el campo de la URL para abrir el sitio que el usuario haya escogido. El método anterior se muestra en la figura 13.

```
private void tablaMouseClicked(java.awt.event.MouseEvent evt) {
    JTable table = (JTable) evt.getSource();
    Point pt = evt.getPoint();
    int ccol = table.columnAtPoint(pt);
    if (isURLColumn(table, ccol)) { // && pointInsidePreferredSize(table, pt)
        int crow = table.rowAtPoint(pt);
        URL url = (URL) table.getValueAt(crow, ccol);
        System.out.println(url);
        try {
            if (Desktop.isDesktopSupported()) { // JDK 1.6.0
                Desktop.getDesktop().browse(url.toURI());
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}

// TODO add your handling code here:
}
```

Figura 13. Método evento *mouseClicked*

## V. ESCENARIO DE PRUEBA Y LIMITACIONES

Entre las pruebas que se realizaron se tomo en cuenta si había una contraseña maestra existente, si no había contraseña maestra existente.

Para la verificación de la contraseña se ingresaron diferentes contraseñas entre las cuales unas cumplían con los requerimientos y otras que no los cumplían, también se validan los errores en digitación, para todos los casos en que haya fallos se pide que se ingrese la contraseña correcta.

Para la creación de sitios la aplicación le asigna para el tiempo por defecto 0 ya que las políticas de cambio de contraseña dependen de las aplicaciones.

Al momento de exportar el documento es necesario hacerlo con la extensión txt ya que al momento de importarlo la aplicación no pondrá error, mientras que si no se pone la extensión la ampliación tomara los caracteres con valores diferentes

## VI. CONCLUSIONES

Para finalizar, el gestor de contraseñas es una herramienta bastante útil a la hora de autenticarnos, porque, en primer lugar, solo se debe aprender una contraseña (contraseña maestra) y ya se tiene acceso a todos los sitios que tengamos guardados. Por otro lado, también evita uno de los problemas más grandes relacionados con las contraseñas que es memorizarnos con cantidad de las mismas y como es una tarea difícil, se incurre en una brecha de seguridad que es tener la misma contraseña para todos los sitios, lo que en términos de seguridad informática significa que somos muy vulnerables al tener un solo punto de fallo, es decir, que si un atacante descubre la contraseña, puede explotar todos los sitios y tener acceso a toda nuestra vida virtual. Por último, este Proyecto ayudó a afianzar los conocimientos teóricos sobre criptografía y ver como en el lenguaje de programación Java se puede utilizar bibliotecas criptográficas para garantizar mayor seguridad en la aplicación.

## VII. BIBLIOGRAFIA

- [1] Legion of the Bouncy Castle Inc, «The Legion of the Bouncy Castle,» 2013. [En línea]. Available: <http://www.bouncycastle.org/java.html>.
- [2] Oracle, «Package java.security,» [En línea]. Available: <https://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html>.
- [3] Oracle, «Package javax.crypto,» [En línea]. Available: <https://docs.oracle.com/javase/7/docs/api/javax/crypto/package-summary.html>.
- [4] Java Point, «Model 1 and Model 2 (MVC) Architecture,» [En línea]. Available: <https://www.javatpoint.com/model-1-and-model-2-mvc-architecture>.