Encriptación simétrica - Algoritmo AES con Java

- Versiones estables recomendadas Java OpenJDK17, 21 Y 23.

[Guía] Proceso de generación de llaves simétricas aleatorias

1. Los algoritmos criptográficos crypto básicos en Java esta disponibles en estas versiones.
2. Se recomienda usar el siguiente código en java:

```java
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.security.SecureRandom;

public class AesKeyGeneratorHex {

    public static void main(String[] args) throws Exception {
        // Generar clave AES de 32 bytes (256 bits)
        SecretKey aesKey = generarClaveAES();
        String claveHex = bytesToHex(aesKey.getEncoded());
        System.out.println("Clave AES (32 bytes - Hex): " + claveHex);

        // Generar IV de 16 bytes (128 bits)
        byte[] iv = generarIV();
        String ivHex = bytesToHex(iv);
        System.out.println("IV (16 bytes - Hex): " + ivHex);
    }

    // Método para generar una clave AES de 256 bits (32 bytes)
    public static SecretKey generarClaveAES() throws Exception {
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        keyGen.init(256); // 256 bits = 32 bytes
```

```
        return keyGen.generateKey();
    }

    // Método para generar un IV de 128 bits (16 bytes)
    public static byte[] generarIV() {
        byte[] iv = new byte[16]; // 16 bytes = 128 bits
        SecureRandom secureRandom = new SecureRandom();
        secureRandom.nextBytes(iv);
        return iv;
    }

    // Método para convertir un arreglo de bytes a una cadena en hexadecimal
    public static String bytesToHex(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : bytes) {
            hexString.append(String.format("%02x", b));
        }
        return hexString.toString();
    }
}
```

3. Al ejecutarse el archivo obtendrá una salida parecida a :

```
Clave AES (32 bytes - Hex): df79b7d35882413570d7a2fe99548f78ad1f698525baf8a4f3397d4fc
cf86a45
IV (16 bytes - Hex): be4187e4785487d0f0d4251687e2b194
```

Tanto la clave AES y el IV generados deben ser utilizado para la encriptación de campos string en el envío de notificaciones o autorizaciones según corresponda

[Guía] Proceso de encriptación simétrica AES

Cifrado Realizado

Se utiliza:

Utiliza	Descripción
Algoritmo	AES-256 (Advanced Encryption Standard)
Modo de operación	CBC (Cipher Block Chaining)
Padding	PKCS5Padding
Clave	Clave secreta proporcionada en formato hexadecimal (32 bytes)
Iv	Vector de inicialización para garantizar aleatoriedad entre bloques (16 bytes)
Salida	Texto cifrado en formato hexadecimal

Consideraciones:

- Se debe tener ya la llave AES-256 y el IV generados anteriormente.

<<<<<<< HEAD

1. Los algoritmos criptográficos crypto básicos en Java ha estado disponible desde Java 1.4 y posteriores.
   =======
2. Los algoritmos criptográficos crypto básicos en Java esta disponible desde Java 1.4 y posteriores.

398137d168e18d15f76504c1bda1422523cffe97

2. Se recomienda usar el siguiente código en java:
   - Nota: si se va a encriptar simétricamente objetos estos deben ser convertidos string antes, por ejemplo es el caso de notificador. Incluir la conversión en el código de encriptación:

```
import java.lang.reflect.Field;

public class Main {
    public static void main(String[] args) throws IllegalAccessException {
        // Crear un objeto
        Notificador notificador = new Notificador("5591607", "CI", "1977-07-13");

        // Convertir el objeto a un string JSON
        String jsonString = toJson(notificador);

        // Imprimir el string JSON
        System.out.println(jsonString);
    }
```

```
    // Método para convertir el objeto a JSON usando reflexión
    public static String toJson(Object obj) throws IllegalAccessException {
        StringBuilder json = new StringBuilder("{");

        // Obtener los campos del objeto
        Field[] fields = obj.getClass().getDeclaredFields();

        // Iterar sobre los campos y construir el JSON
        for (int i = 0; i < fields.length; i++) {
            fields[i].setAccessible(true); // Hacer los campos accesibles
            String key = fields[i].getName();
            String value = String.valueOf(fields[i].get(obj)); // Obtener el valor de
l campo

            // Formatear la clave y el valor como JSON
            json.append("\"").append(key).append("\":\"").append(value).append("\"");

            if (i < fields.length - 1) {
                json.append(", ");
            }
        }

        json.append("}");
        return json.toString();
    }
}

class Notificador {
    private String numeroDocumento;
    private String tipoDocumento;
    private String fechaNacimiento;

    // Constructor
    public Notificador(String numeroDocumento, String tipoDocumento, String fechaNaci
miento) {
        this.numeroDocumento = numeroDocumento;
        this.tipoDocumento = tipoDocumento;
        this.fechaNacimiento = fechaNacimiento;
    }
```

```
}
```

- Código de encriptación

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.spec.IvParameterSpec;

public class AesEncryption {

    public static void main(String[] args) throws Exception {
        // Clave AES de 32 bytes (256 bits) en formato hexadecimal
        String claveHex = "07fc3f82bf46557f7a4fad644833a72ca5a1751b83c24b306d42a54fbe
c388db";

        // IV de 16 bytes (128 bits) en formato hexadecimal
        String ivHex = "643cdf76a09a6efcb469d1949def66f7";

        // Texto a encriptar
        String textoOriginal = "descripcion";
        System.out.println("Texto original: " + textoOriginal);

        // Encriptar el texto
        String textoEncriptado = encriptarAES(textoOriginal, claveHex, ivHex);
        System.out.println("Texto encriptado (Hexadecimal): " + textoEncriptado);
    }

    // Método para encriptar texto usando AES en modo CBC con padding PKCS5
    public static String encriptarAES(String texto, String claveHex, String ivHex) th
rows Exception {
        // Convertir las cadenas hexadecimales a byte[]
        byte[] clave = hexStringToByteArray(claveHex);
        byte[] iv = hexStringToByteArray(ivHex);

        // Crear objeto Cipher para AES con modo CBC y padding PKCS5
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

        // Crear la clave y el IV
        SecretKeySpec claveSpec = new SecretKeySpec(clave, "AES");
        IvParameterSpec ivSpec = new IvParameterSpec(iv);
```

```
        // Inicializar el Cipher en modo ENCRYPT
        cipher.init(Cipher.ENCRYPT_MODE, claveSpec, ivSpec);

        // Encriptar el texto
        byte[] textoEncriptado = cipher.doFinal(texto.getBytes("UTF-8"));

        // Devolver el texto encriptado en formato hexadecimal
        return bytesToHex(textoEncriptado);
    }

    // Método para convertir una cadena hexadecimal a un arreglo de bytes
    public static byte[] hexStringToByteArray(String hexString) {
        int len = hexString.length();
        byte[] data = new byte[len / 2];
        for (int i = 0; i < len; i += 2) {
            data[i / 2] = (byte) ((Character.digit(hexString.charAt(i), 16) << 4)
                + Character.digit(hexString.charAt(i + 1), 16));
        }
        return data;
    }

    // Método para convertir un arreglo de bytes en una cadena hexadecimal
    public static String bytesToHex(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : bytes) {
            hexString.append(String.format("%02x", b));
        }
        return hexString.toString();
    }
}
```

3. Output:

```
Texto original: texto secreto
Texto encriptado (Hexadecimal): 5465d02c60fc5e61394e1cfca940125b
```

Este valor encriptado debe ser utilizado en el envío de notificaciones o autorizaciones según corresponda