UD6: Programación segura con java

PSP

Ana Alonso

Curso 2024-2025

Generación de HASH

- Se puede realizar mediante los algoritmos de la clase **MessageDigest** que se encarga de proporcionar el algoritmo deseado mediante el método getInstance().
- Métodos principales de la clase:
  - **getInstance()** : proporciona una instancia con la implementación del algoritmo de generación de resúmenes que se le indique.
  - **reset()** :elimina la posible información contenida en la instancia.
  - **update()** : actualiza los datos contenidos en la instancia a partir de los cuales se va a generar el resumen. Se puede invocar varias veces para añadir nuevos datos.
  - **digest()** : genera el resumen de los datos contenidos en el objeto.
  - **isEqual()** : compara 2 resúmenes para determinar si son iguales.

Cifrado con AES

- Paquetes :
  - java.security para la gestión de claves y de números aleatorios.
  - javax.crypto para el acceso a las implementaciones de AES
- Interfaces y clases para utilizar el algoritmo AES:
  - java.security.Key: interfaz de alto nivel para todo tipo de claves.
  - java.security.NoSuchAlgorithmException : excepción que se produce cuando se intenta instanciar un algoritmo del que no hay implementación.
  - java.security.SecureRandom : clase que proporciona un número aleatorio criptográficamente fuerte. Se utiliza para crear vectores de inicialización.
  - javax.crypto.spec.SecretKeySpec : Escifica una clave secreta. Puede ser generada aleatoriamente o a partir de una contraseña conocida del usuario. Implementa la interface java.security,Key.
  - Javax.cripto.Cipher: representa una implementación de un algoritmo de cifrado.

Cifrado con AES – principales métodos

- Métodos de la clase KeyGenerator:
  - **getInstance()** : proporciona un algoritmo de generación de claves.
  - **init()** : inicializa el generador de claves a una longitud determinada en bits.
  - **generateKey()** : genera una clave.
- Métodos de la clase SecureRandom:
  - **nextBytes():** proporciona un número aleatorio de bytes.
- Métodos de la clase Cipher:
  - **getInstance()** : proporciona una implentación de un algoritmo de cifrado y descifrado.
  - **init()** : inicializa el algoritmo de cifrado y descifrado.
  - **doFinal()** : cifra y descifra un conjunto de bytes.

Cifrado con RSA

- Paquetes :
  - java.security para la gestión de claves.
  - javax.crypto para el acceso al algoritmo de cifrado y descifrado.
- Interfaces y clases para utilizar el algoritmo RSA:
  - java.security.Key: interfaz de alto nivel para todo tipo de claves.
  - java.security.NoSuchAlgorithmException : excepción que se produce cuando se intenta instanciar un algoritmo del que no hay implementación.
  - java.security.KeyFactory: conversor de claves de formato byte a formato Key, y viceversa.
  - java.security.KeyPair: representa un par de claves pública y privada.
  - java.security.KeyPairGenerator: generador de claves pública y privada.
  - java.security.PrivateKey: representa una clave privada.
  - java.security.PublicKey: representa una clave pública.

Cifrado con RSA

- Interfaces y clases para utilizar el algoritmo RSA:
  - java.security.spec.EncodedKeySpec: clase abstracta que representa una clave pública o privada codificada.
  - java.security.spec.PKCS8EncodedKeySpec: representa una clave privada codificada con el estándar PKCS #8. Hereda de EncodedKeySpec.
  - java.security.spec.X509EncodedKeySpec: representa una clave pública codificada con el estándar X509. Hereda de EncodedKeySpec.
  - Javax.cripto.Cipher: representa una implementación de un algoritmo de cifrado.

Cifrado con RSA– principales métodos

- Métodos de la clase KeyPairGenerator:
  - **getInstance()** : proporciona un algoritmo de generación de claves.
  - **initialize()** : establece el tamaño de las claves.
  - **generateKeyPair()** : genera un par de claves privada y pública.
- Métodos de la clase KeyFactory:
  - **getInstance() :** proporciona una instancia del generador de conversor de claves.
  - **generatePublic():** genera una PublicKey a partir de un conjunto de bytes.
  - **generatePrivate():** genera una PrivateKey a partir de un conjunto de bytes.
- Métodos de la clase SecureRandom:
  - **nextBytes():** proporciona un número aleatorio de bytes.
- Métodos de la clase Cipher:
  - **getInstance() , init() , doFinal()**