**Documentación Completa del Módulo de Almacenamiento**

**Propósito General del Módulo**

El **módulo de almacenamiento** está diseñado para gestionar la **persistencia de datos** en archivos cifrados, garantizando la **seguridad** de la información almacenada mediante el uso de un cifrado simétrico (Fernet). Este módulo ofrece funciones para **guardar** y **cargar documentos** en un **formato personalizado** que solo puede ser interpretado por el programa, protegiendo los datos de accesos no autorizados.

Además, el módulo tiene soporte para **archivos de gran tamaño**, gracias al uso de **pickle** para serializar los objetos y convertirlos en un formato binario adecuado para el cifrado.

El principal objetivo de este módulo es ofrecer un **método seguro y eficiente** para almacenar y recuperar información sensible, utilizando un sistema de cifrado fuerte basado en **cryptography.Fernet**.

**Estructura del Módulo de Almacenamiento**

Este módulo contiene una **clase Almacenamiento**, que gestiona tanto el cifrado de datos como el almacenamiento y la recuperación de archivos cifrados. Las funcionalidades principales de este módulo incluyen:

1. **Generación de clave de cifrado**.
2. **Guardar documentos cifrados** en un archivo con un formato binario personalizado.
3. **Cargar y descifrar documentos cifrados**.
4. **Gestión de claves** para garantizar que los datos solo sean accesibles por este programa.

**Funciones del Módulo de Almacenamiento**

**1. \_\_init\_\_(self, clave=None)**

**Propósito:**

Inicializa el objeto Almacenamiento, que se encargará de gestionar el cifrado de documentos. Si no se proporciona una clave de cifrado, genera una nueva clave para el cifrado de los datos.

**Parámetros:**

* **clave** (str, opcional): Una clave secreta para cifrar y descifrar los documentos. Si no se proporciona, se genera una nueva clave.

**Descripción:**

* Si no se proporciona una clave, la función invoca el método **generar\_clave()** para generar una clave nueva para el cifrado.
* Si la clave es proporcionada, se utiliza la clave proporcionada para la instancia de **Fernet**.

**Flujo de Ejecución:**

1. El constructor recibe o genera una clave para el cifrado.
2. Crea una instancia de **Fernet** con la clave especificada, lo que permitirá cifrar y descifrar documentos.

**2. generar\_clave(self)**

**Propósito:**

Genera una nueva clave para el cifrado utilizando **cryptography.Fernet**.

**Descripción:**

* Este método utiliza la función **Fernet.generate\_key()** para generar una nueva clave aleatoria de 32 bytes que será utilizada para cifrar y descifrar los documentos.
* La clave generada es un objeto binario que luego se convierte a formato string mediante **.decode()** para facilitar su almacenamiento y uso.

**Salida:**

* **str**: La clave generada en formato string.

**Flujo de Ejecución:**

1. Se genera una clave aleatoria utilizando **Fernet.generate\_key()**.
2. Se convierte la clave a formato string utilizando **.decode()**.

**3. guardar\_documento(self, datos, archivo)**

**Propósito:**

Guarda los datos en un archivo cifrado, asegurando que solo este programa pueda acceder a ellos mediante la clave de cifrado.

**Parámetros:**

* **datos** (object): Los datos a almacenar, que pueden ser cualquier objeto serializable de Python (listas, diccionarios, etc.).
* **archivo** (str): El nombre del archivo donde se guardarán los datos cifrados.

**Descripción:**

1. **Serialización**: Los datos se serializan utilizando **pickle.dumps()** para convertir el objeto en un formato binario.
2. **Cifrado**: Los datos serializados se cifran utilizando la clave proporcionada a la instancia de **Fernet**, utilizando **Fernet.encrypt()**.
3. **Almacenamiento**: Los datos cifrados se guardan en un archivo utilizando el modo **'wb'** de apertura de archivos binarios en Python.

**Flujo de Ejecución:**

1. Los datos se serializan con **pickle**.
2. Los datos serializados se cifran utilizando **Fernet.encrypt()**.
3. Los datos cifrados se guardan en un archivo.

**Ejemplo de Uso:**

# Crear instancia de almacenamiento

almacen = Almacenamiento()

# Datos a guardar

datos = {"nombre": "Juan", "edad": 30}

# Guardar los datos en un archivo cifrado

almacen.guardar\_documento(datos, "documento\_seguro.dat")

**4. cargar\_documento(self, archivo)**

**Propósito:**

Carga un documento cifrado desde un archivo y descifra los datos almacenados para que puedan ser utilizados en el programa.

**Parámetros:**

* **archivo** (str): El nombre del archivo que contiene los datos cifrados.

**Salida:**

* **object**: Los datos descifrados y deserializados, listos para ser utilizados en el programa.

**Descripción:**

1. **Lectura del archivo**: Se abre el archivo en modo binario **'rb'** y se leen los datos cifrados.
2. **Descifrado**: Los datos leídos se descifran utilizando la clave mediante **Fernet.decrypt()**.
3. **Deserialización**: Los datos descifrados se deserializan usando **pickle.loads()** para convertirlos de nuevo en su formato original.

**Flujo de Ejecución:**

1. Se leen los datos cifrados desde el archivo.
2. Los datos se descifran utilizando **Fernet.decrypt()**.
3. Los datos descifrados se deserializan utilizando **pickle.loads()**.

**Ejemplo de Uso:**

# Cargar los datos desde un archivo cifrado

datos\_cargados = almacen.cargar\_documento("documento\_seguro.dat")

print(datos\_cargados) # {"nombre": "Juan", "edad": 30}

**5. obtener\_clave(self)**

**Propósito:**

Obtiene la clave de cifrado utilizada por el objeto **Fernet**.

**Salida:**

* **str**: La clave de cifrado en formato string.

**Descripción:**

Este método es útil si deseas obtener la clave de cifrado en algún punto del programa para almacenarla o usarla en otro contexto.

**Consideraciones Importantes**

1. **Cifrado Simétrico**: Utilizamos **cryptography.Fernet** para cifrar los documentos, lo que significa que la misma clave se usa tanto para cifrar como para descifrar. La seguridad de los documentos depende de la protección de esta clave.
2. **Formato de Documento Personalizado**: Los documentos se almacenan en un **formato binario** utilizando **pickle** para serializar los datos. Solo este programa, que entiende este formato, puede leer los archivos.
3. **Seguridad de la Clave**: La clave generada por **Fernet.generate\_key()** es sensible y debe ser protegida. Si se pierde la clave, los documentos cifrados no podrán ser descifrados. Es recomendable almacenar la clave de forma segura, preferiblemente en un lugar distinto a los archivos cifrados.
4. **Tamaño de los Archivos**: Este sistema es adecuado para manejar documentos de gran tamaño, ya que **pickle** permite serializar grandes cantidades de datos y **cryptography** es eficiente en términos de cifrado/descifrado.

**Comandos para Librerías de Cifrado**

Para cifrar y descifrar los documentos, necesitarás instalar la librería **cryptography**. Esto se puede hacer ejecutando el siguiente comando en la consola:

pip install cryptography

**Conclusión**

Este módulo de almacenamiento ofrece una solución segura y eficiente para **guardar** y **cargar documentos cifrados**. La combinación de **pickle** para serialización y **cryptography.Fernet** para cifrado asegura que los documentos sean inaccesibles sin la clave correcta. Con este enfoque, podemos almacenar documentos grandes de manera segura y garantizar que solo el programa pueda acceder a su contenido.

Este módulo es ideal para manejar **información sensible** o para **protección de datos** en proyectos donde la seguridad es crucial.