***PRÁCTICA: CRIPTOGRAFÍA Y SEGURIDAD INFORMÁTICA***

**ENTREGABLE 2**

**Grado en Ingeniería en Informática, Grupo 82, 07/10/2023**

Realizado por el grupo de prácticas 8, formado por:

* Raúl Sanz Belmar, 100472372@alumnos.uc3m.es
* Mario Rodríguez Román, 100472194@alumnos.uc3m.es

Enlace repositorio GitHub: <https://github.com/Rawwl03/Criptografia_Uc3m>

Índice de Contenidos

[**PROPÓSITO DE LA APLICACIÓN** 3](#_Toc152103624)

[**DISEÑO** 3](#_Toc152103625)

[**MEJORAS IMPLEMENTADAS** 4](#_Toc152103626)

[**FIRMA DIGITAL** 5](#_Toc152103627)

[**CERTIFICADOS DE CLAVE PÚBLICA** 5](#_Toc152103628)

[**COMPLEJIDAD DE LA APLICACIÓN** 5](#_Toc152103629)

# **PROPÓSITO DE LA APLICACIÓN**

El proyecto va a estar basado en la creación de una aplicación de servicios que te permita comprar entradas para el cine y gestionar tus entradas asociadas a tu cuenta personal. Las funcionalidades principales que se pueden realizar en el sistema son las siguientes:

* Modo sistema o admin para la gestión de entradas, usuarios y peticiones almacenadas
* Gestión de tarjetas (añadir o eliminar) para ser usadas en los pagos por parte del usuario
* Adquisición de entradas mediante un sistema de peticiones, que tendrán que ser aceptadas por un Admin o el sistema.
* Cambiar tu contraseña en cualquier momento

Todo esto contendrá una seguridad frente ataques proporcionada por la encriptación, de manera que vamos a guardar datos como contraseñas y tarjetas en una base de datos, pero estos estarán cifrados (y autenticados en el caso de las tarjetas) gracias a la librería cryptography. Además, a la hora de realizar peticiones, estas irán firmadas y serán verificadas, al igual que en el caso de que el sistema le envíe una entrada que ha comprado el usuario respectivo (lo veremos más adelante). Junto a todo esto, se ha desplegado una PKI, que será comentada en el diseño, consiguiendo así autenticidad a la hora de realizar acciones en el sistema.

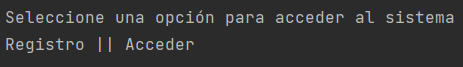
El diseño de la aplicación está comprendido por una serie de menús, cada uno con unas funcionalidades, que se manejan a través de la terminal (escribiendo inputs) y se puede navegar entre ellos. Aparte de la aplicación, se han añadido mejoras opcionales que se comentarán más adelante en profundidad, como implementación de base de datos, rotación de claves y acceso a un modo de sistema mediante reconocimiento facial.

## **DISEÑO**

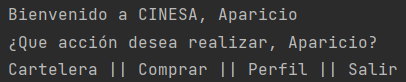
Como se ha mencionado antes, el diseño de la aplicación se basa en una serie de menús que están enlazados entre sí. Todos estos menús se generan en la clase Terminal, que representa a la terminal donde se va a mostrar todo, y se va a utilizar una variable global llamada ‘contrasena\_sys’, que almacenará la contraseña en claro del usuario actual. Como dato a mencionar, se crean datos para un usuario llamado “Sistema”, que no es almacenado en la tabla usuarios, pero si tiene claves asimétricas para poder realizar operaciones de gestión de datos, ya que es considerado el modo sistema que describiremos a continuación

Los menús principales que podemos encontrar son los siguientes:

1. **Menú acceso al sistema** -> este menú se trata del menú inicial que se muestra al iniciar la aplicación. Se pueden realizar tres cosas: registrarse, acceder al sistema con un usuario existente o acceder al menú sistema para gestionar datos mediante reconocimiento facial. La funcionalidad está recogida en sys\_inicio() {clase Terminal, línea 27}, y una vez accedido al sistema correctamente, irá al menú inicio del cine. Si el acceso al menú del sistema es permitido, entonces irá al menú de gestión del sistema.



1. **Menú inicio del cine** -> desde aquí podemos realizar 5 acciones: cartelera (podemos ver las películas disponibles y seleccionar una de ellas para ver más información), comprar (nos lleva al proceso de compra, que nos hace seleccionar película, seleccionar un asiento disponible y seleccionar una tarjeta válida para proceder al pago, donde se enviará una petición con la entrada que se desea comprar), perfil (nos lleva al menú perfil), peticiones(nos lleva al menú de hacer petición) y salir del sistema. Corresponde a la función accion\_cine(user\_accedido) {clase Terminal, línea 159}, que recibe el usuario que accedido al sistema, y proporciona las funcionalidades descritas anteriormente.



1. **Menú perfil** -> las funcionalidades que recoge este menú van asociadas a la interacción con la información de un perfil, que en este caso puede ser guardar una tarjeta, borrar una tarjeta, cambiar la contraseña del usuario, ver las entradas compradas, ver las tarjetas guardadas del usuario, ver las peticiones realizadas por el usuario y salir (volver al menú inicio del cine), y en el caso . La función que se encarga de realizar las funcionalidades de este menú es acc\_perfil(user\_accedido) {clase Terminal, línea 325}, que también necesita el user\_accedido.



Por lo tanto, estos son los 3 menús principales de la aplicación, el primero por ser el menú que sirve de entrada a la aplicación, y los otros 2 porque después de realizar cada acción se va a regresar a uno de ellos para seguir con la ejecución. Cada acción de las expresadas en cada menú va a llevar a cabo la creación de un menú de selección o de visualización, dependiendo de la acción escogida. Además, cada acción solicitada por input o valor dado tiene que ser válido, o de lo contrario el sistema no avanzará hasta que se haya introducido un input correcto.

## **MEJORAS IMPLEMENTADAS**

Entre todas las mejoras opcionales que se proponían en el enunciado, nosotros hemos decidido implementar 2 de ellas. Estas son las siguientes:

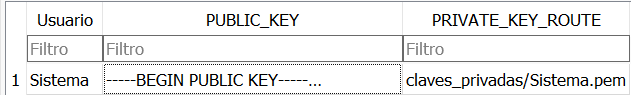
* **Implementación de base de datos:** para guardar los datos, hemos usado sqlite3 en Python, que nos permite crear bases de datos y gestionarlas para añadir los datos que querramos. Para ver la información de la base de datos, hemos usado DB Browser. La base de datos se corresponde con la clase Database, que contiene todas las acciones con querys SQL para realizar funcionalidades en cada caso. De esta manera, podemos tener los datos mejor organizados y con un formato adecuado, además de poder acceder fácilmente a nuestros datos mediante sentencias SQL.
* **Rotación de claves:** se ha decidido implementar la rotación de claves de dos maneras. La primera consiste en que cada vez que se realiza un acceso, se va a generar el cifrado de la contraseña con un salt diferente, de manera que el cifrado va cambiando (función rotación\_claves(user\_accedido, False, None) y así solo cambia el salt). La segunda manera consiste en la funcionalidad del menú perfil “cambiar contraseña”, que pedirá una nueva contraseña y generará un hash nuevo con un salt nuevo (función rotación\_claves(user\_accedido, True, contraseña\_antigua)). En estos dos casos, en rotación\_claves() {clase Terminal, línea 574}, se tendrán que descifrar las tarjetas con la clave antigua y cifrarlas de nuevo con la nueva clave generada, para que así los datos se conserven correctamente.

# **FIRMA DIGITAL**

La firma digital nos va a aportar dentro de nuestra aplicación autenticidad respecto a un mensaje. De esta manera, conseguimos verificar que un mensaje proviene de una persona concreta correctamente. Para realizar todo esto, primero hemos tenido que decidir dos cosas importantes: cómo gestionar las claves asimétricas que se utilizarán en este proceso, y cómo implementar la firma de manera que tenga utilidad. Además, es necesario decidir el algoritmo que se va a usar. En nuestro caso, el algoritmo usado es **RSA**, en el que se necesitará una clave privada y pública por cada usuario para que cada uno pueda firmar (con clave privada del emisor) y verificar (con clave pública del emisor) en su respectivo momento. A continuación, se van a explicar ambas cosas detalladamente.

* Gestión de claves asimétricas:

Por cada usuario que acceda por primera vez al sistema se creará un par de claves asimétricas (pública y privada) que se guardarán de la siguiente manera: en la base de datos, guardaremos la clave pública codificada en PEM y la ruta del fichero .pem donde estará la clave privada encriptada. A su vez, se creará un par de claves para un usuario llamado “Sistema”, que corresponde con el acceso en modo System. A continuación, vamos a mostrar fotos sobre cómo se ven estas claves en nuestra aplicación.



La función en la que se crean las claves se llama generar\_asymethric\_keys(user\_accedido) {clase Terminal, línea 1268}, que se invoca en el primer acceso de un usuario al sistema (en el futuro se ha cambiado la generación de una clave pública por un certificado de clave pública). Esta función genera mediante rsa una clave privada de longitud 2048 y con exponente público 65537 (valor recomendado). Esta clave es encriptada con la contraseña del usuario correspondiente para que sea privada suya, y es guardada en la ruta “claves\_privadas/”+<nombre\_user>+”.pem”. En cuanto al certificado de clave pública, que nos va a proporcionar la clave pública del usuario, se crea uno para el usuario que es firmado por la AC (“Sistema”), con una validez de 30 días. El formato de la firma se hablará más adelante, pues es el mismo para todos.

# **CERTIFICADOS DE CLAVE PÚBLICA**

# **COMPLEJIDAD DE LA APLICACIÓN**