# *Práctica 5 – S-TDD: Gestor de Credenciales*

Ingeniería del Software Seguro - Grado en Ciberseguridad e Inteligencia Artificial

Dado que casi todos los sistemas usan credenciales para el acceso básico de sus usuarios, vamos a diseñar un componente para SVAIA que se encargará de la gestión de credenciales. Este componente debe permitir almacenar y recuperar credenciales de usuarios de forma segura.

**Requisitos Funcionales:**

1. **Almacenamiento de Credenciales**: El sistema debe permitir añadir credenciales (usuario/contraseña) asociadas a un servicio (ej: "GitHub", "VulnDB", etc.).
2. **Recuperación de Credenciales**: Debe permitir recuperar la contraseña para un servicio y usuario dados.
3. **Eliminación de Credenciales**: Debe permitir eliminar credenciales existentes.
4. **Listado de Servicios**: Debe proporcionar un listado de todos los servicios almacenados.

**Requisitos de Seguridad:**

1. **Confidencialidad**: Las contraseñas nunca deben almacenarse sin cifrar.
2. **Integridad**: El sistema debe verificar que las contraseñas no pueden ser modificadas de forma consciente o accidental.
3. **Robustez**: El sistema debe verificar que las contraseñas cumplan con una política robusta.
4. **Autenticidad**: Solo debe permitir operar con el sistema si se proporciona una clave maestra correcta.
5. **Auditoría**: Debe mantenerse un log seguro de las acciones realizadas.
6. **Prevención de Logging Sensitivo**: No se debe registrar información sensible en logs.
7. **Protección contra Inyección**: Prevenir ataques de inyección en el UI (p. ej. en los nombres de servicio).

## Instrucciones

Debemos implementar el componente **`GestorCredenciales`** usando S-TDD, añadiendo tests de seguridad junto con los funcionales.

1. El código se organizará siguiendo la siguiente estructura:

```
GestorCredenciales/
├── src/
│   └── gestor_credenciales/
│       ├── __init__.py
│       ├── gestor_credenciales.py
│       ├── utils.py …
│       └── …
├── tests/
│   └── gestor_credenciales/
│       ├── test_ gestor_credenciales.py
│       ├── test_utils.py …
│       └── …
├── pyproject.toml
├── README.md
└── setup.cfg / setup.py (si se usa)
```

2. El código base se puede descargar en el Campus Virtual
3. Seguiremos en todo momento el ciclo S-TDD:

   tests (funcionales y de seguridad) → implementación → refactorización.
4. Los requisitos deben traducirse a código utilizando tanto pruebas como Diseño por Contrato. Usaremos **`unittest`** para las pruebas unitarias (principalmente comprobación del comportamientos) y Design-By-Contract (con **`icontract`** o decoradores) para validaciones adicionales (que incluyen tanto precondiciones **`-@require-`** como postcondiciones **`-@ensure-`**).
5. Usaremos *SonarQube for IDE* para analizar y refactorizar el código. Se recomienda usar *PyCharm* como IDE.
6. El proceso será en grupo e iterativo. El grupo define los requisitos a abordar, se distribuye el trabajo y se realiza mediante programación por pares. Los roles en el par cambian en cada ciclo.

## Tareas

1. Completar la implementación del módulo **`GestorCredenciales`** siguiendo S-TDD.

   Esto implica:

   1.1. Completar las pruebas que están ya incluidas;

   1.2. Desarrollar el código para que pase las pruebas. *Estos dos pasos los realizan pares diferentes.*

   1.3. Refactorizar el código usando SonarQube para mejorar la calidad, legibilidad y seguridad del mismo.

2. Añadir al menos 6 pruebas de seguridad adicionales a las proporcionadas (deben fallar tal como está el código)
    2.1. Implementar las pruebas e implementar las validaciones adicionales que sean necesarias siguiendo la técnica Design-By-Contract.
    2.2. Implementar el código adicional para que las nuevas pruebas pasen.
    2.3. Refactorizar el código usando SonarQube para mejorar la calidad, legibilidad y seguridad del mismo.
3. Registrar el trabajo de cada miembro del grupo en un informe de trabajo que indica todo lo que ha hecho cada uno. Se proporciona una plantilla en el Campus Virtual.
4. Documentar cualquier decisión de diseño relacionada con la seguridad.

### Criterios de Evaluación

1. **Corrección Funcional**: Todas las pruebas funcionales pasan.
2. **Pruebas de Seguridad**: Las pruebas de seguridad son completas y relevantes.
3. **Calidad de Código**: Uso adecuado de contratos y seguimiento de sugerencias de SonarQube.
4. **Documentación**: Explicación de medidas de seguridad implementadas.
5. **Manejo de Errores**: Correcto manejo de casos excepcionales.

### Entrega

La entrega se realiza por proyecto. Se debe entregar:

1. Archivo ZIP con el proyecto completo, incluyendo el código y las pruebas.
2. Capturas de pantalla de los análisis de SonarQube.
3. Informe de trabajo:
    3.1. Registro de trabajo de todos los miembros del grupo.
    3.2. Documentación breve explicando las decisiones de diseño.
    3.3. Conclusiones sobre la experiencia de uso de S-TDD.