UNIVERSIDAD PRIVADA DE TRUJILLO

UPRIT University

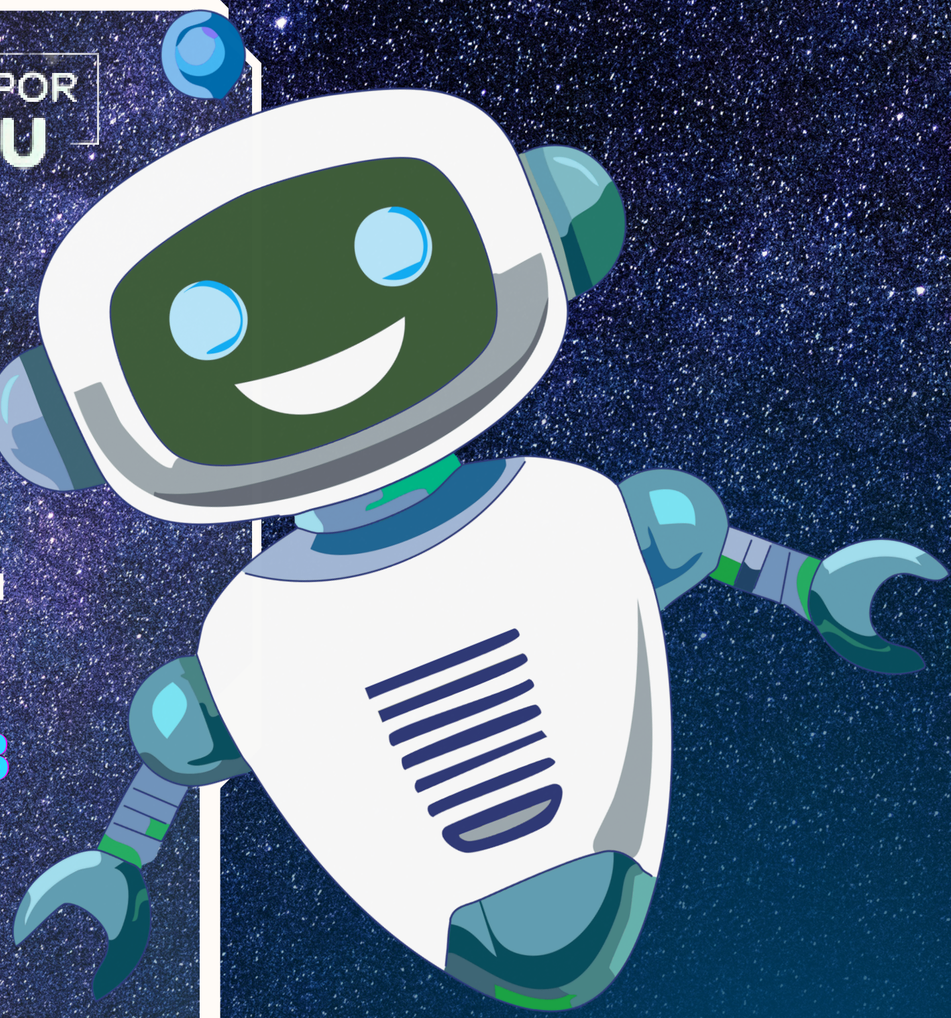
LICENCIADA POR SUNEDU

ACTIVIDAD FORMATIVA Nº 4

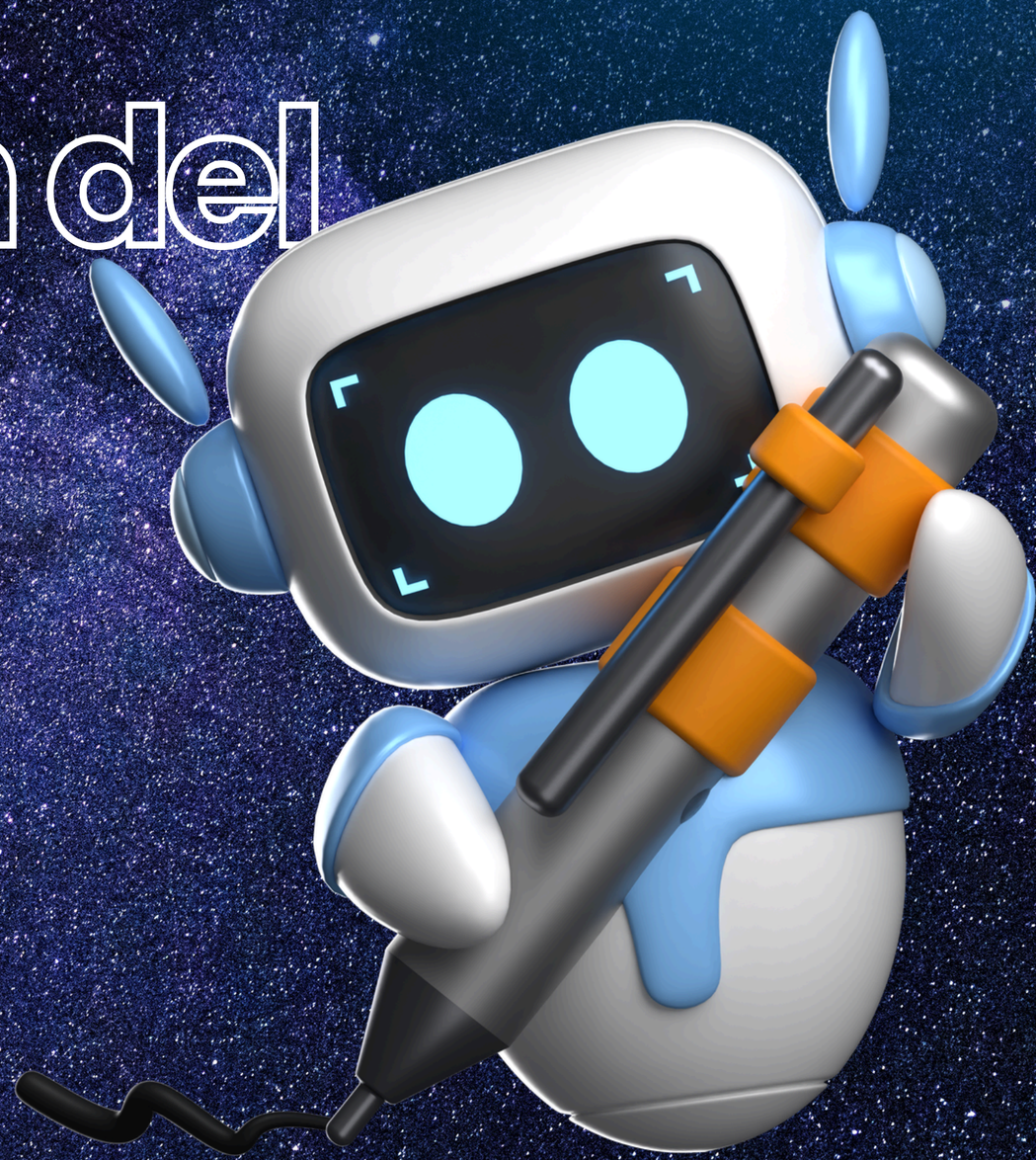
ARQUITECTURA DE ENTORNOS WEB

MG. ING. BLANCAS NUÑEZ MITCHELL PAULO

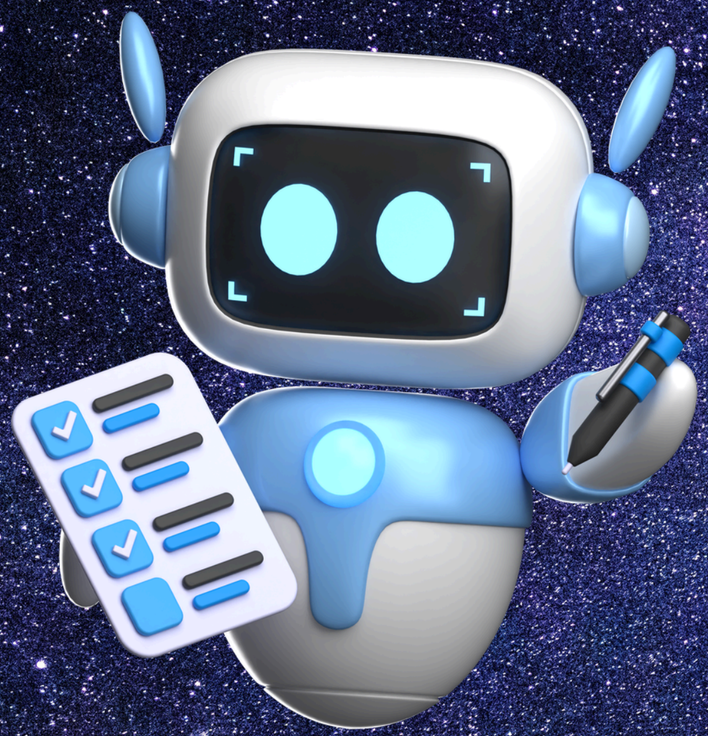
GRUPO Nº 1



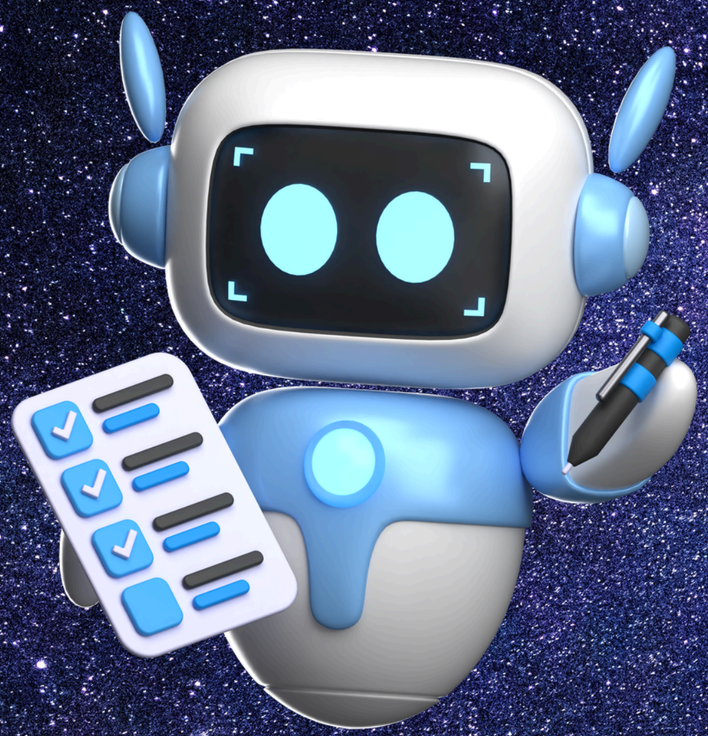
1. Identificación del problema

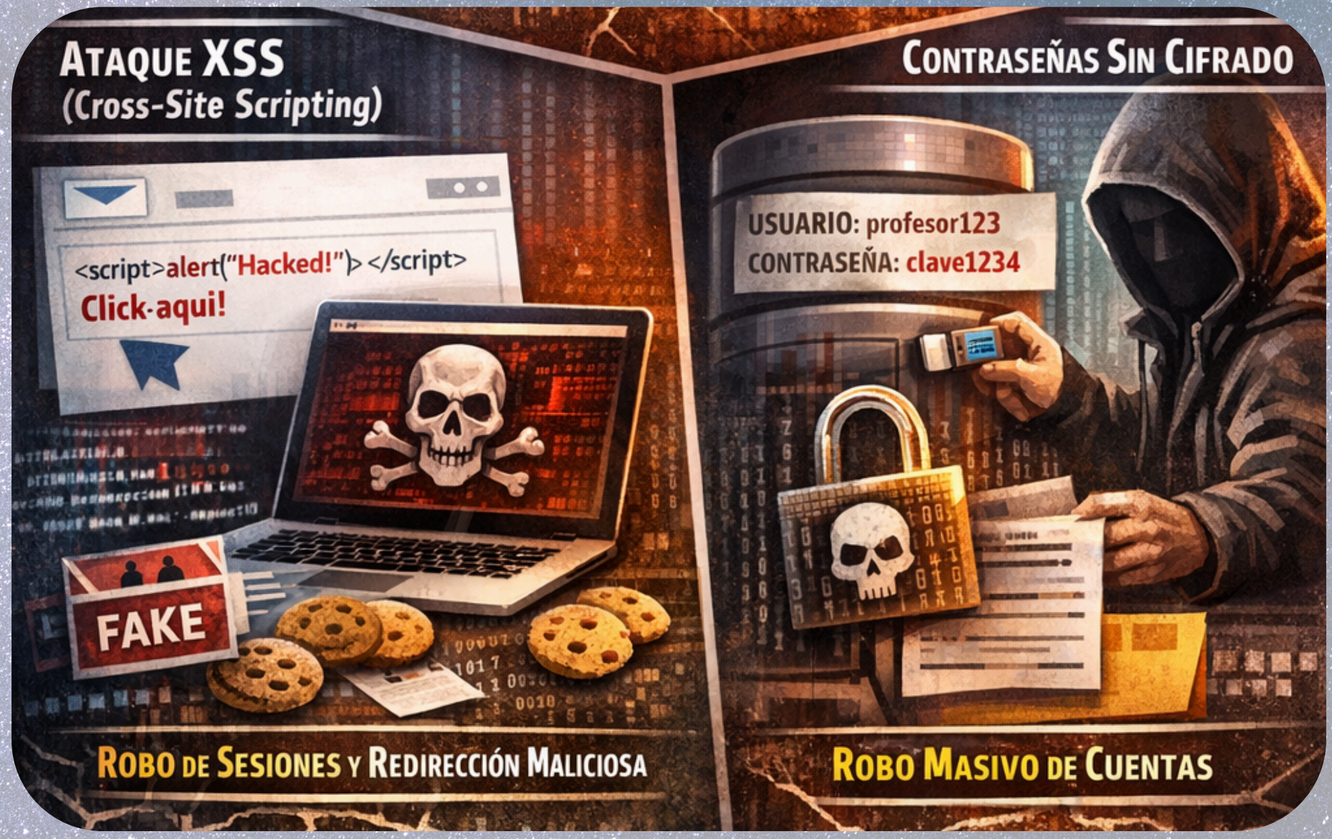


Posibles amenazas o ataques en el sistema

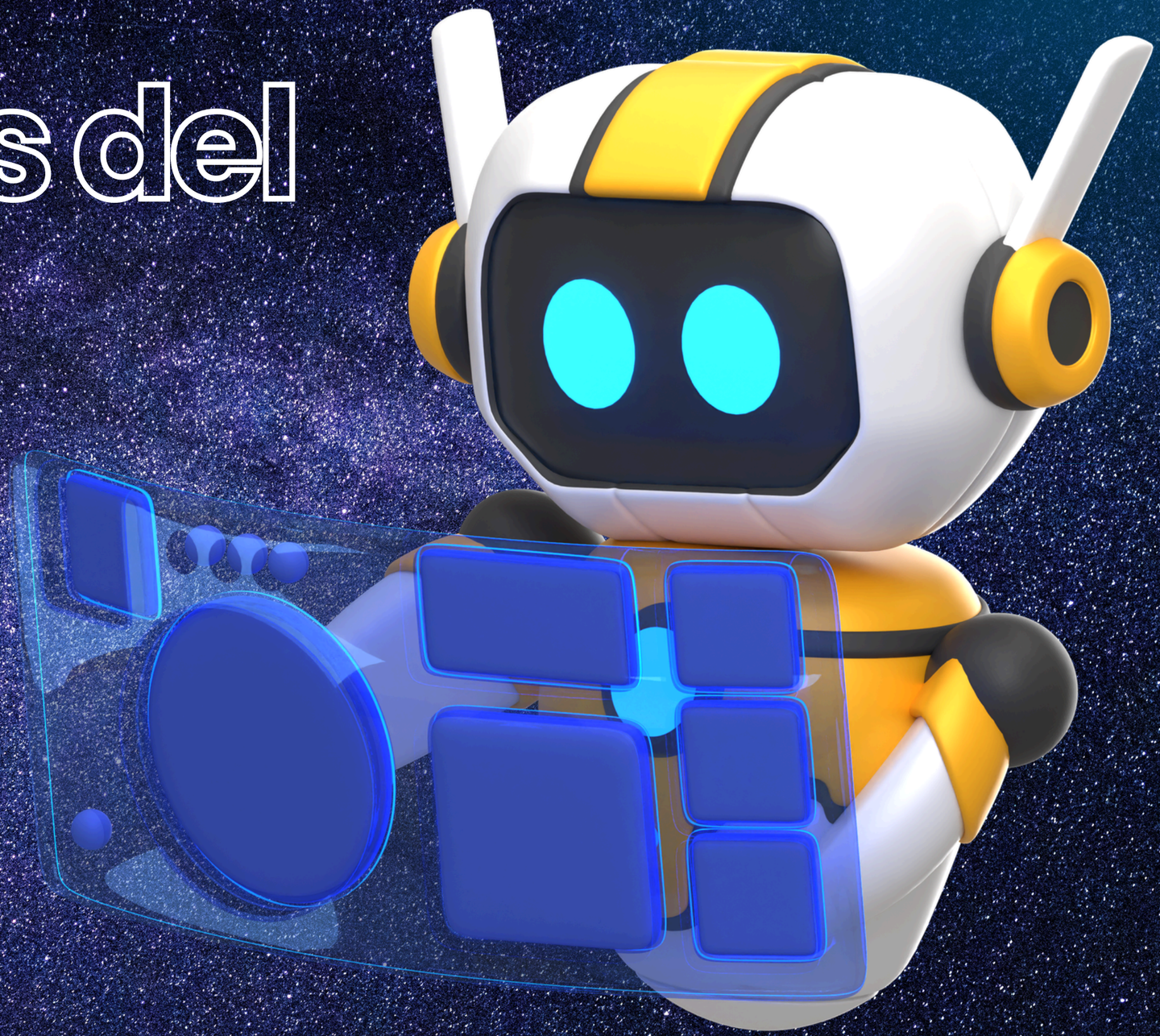


Posibles amenazas o ataques en el sistema





2. Análisis del sistema



Según las características del sistema, se identifican varias vulnerabilidades.

Vulnerabilidades de Seguridad

Vulnerabilidad 1 Sin Límite de Intentos de Login	Vulnerabilidad 2 Mensajes con Código HTML	Vulnerabilidad 3 Contraseñas Sin Cifrado
Problema Intentos ilimitados de inicio de sesión.	**Problema** <script> Mensajes con código HTML	**Problema** 123456 Contraseñas en Texto Plano
Consecuencia Ataques de Fuerza Bruta	**Consecuencia** XSS Ataques XSS	**Consecuencia** Robo Masivo de Cuentas
Impacto Robo de Cuentas Acceso a Información Académica Manipulación de Notas	**Impacto** Robo de Cookies Redirección a Páginas Falsas Instalación de Malware	**Impacto**

3. Propuesta de mejora

1. Limitar intentos de inicio de sesión

Problema que soluciona

Ataques de fuerza bruta.

Implementación

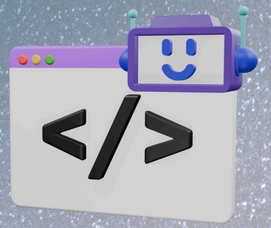
- Bloquear la cuenta después de 5 intentos fallidos.
- Usar CAPTCHA después de varios intentos.

Mejora en seguridad

Reduce significativamente la posibilidad de adivinar contraseñas.

2. Implementar cifrado de contraseñas

Problema que soluciona
Almacenamiento inseguro de contraseñas.



Implementación

- Utilizar algoritmos de hash seguros como:
- bcrypt
- Argon2
- SHA-256 con salt

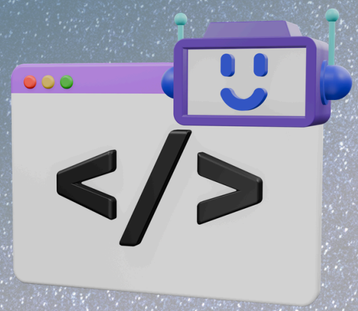
Ejemplo en Python:

```python
from werkzeug.security import generate_password_hash, check_password_hash

password_hash = generate_password_hash("mi_contraseña")

check_password_hash(password_hash, "mi_contraseña")
```

Mejora en seguridad
Aunque alguien acceda a la base de datos, no podrá ver las contraseñas reales.

3. Filtrar y sanitizar el contenido HTML

Problema que soluciona
Ataques XSS.



Implementación

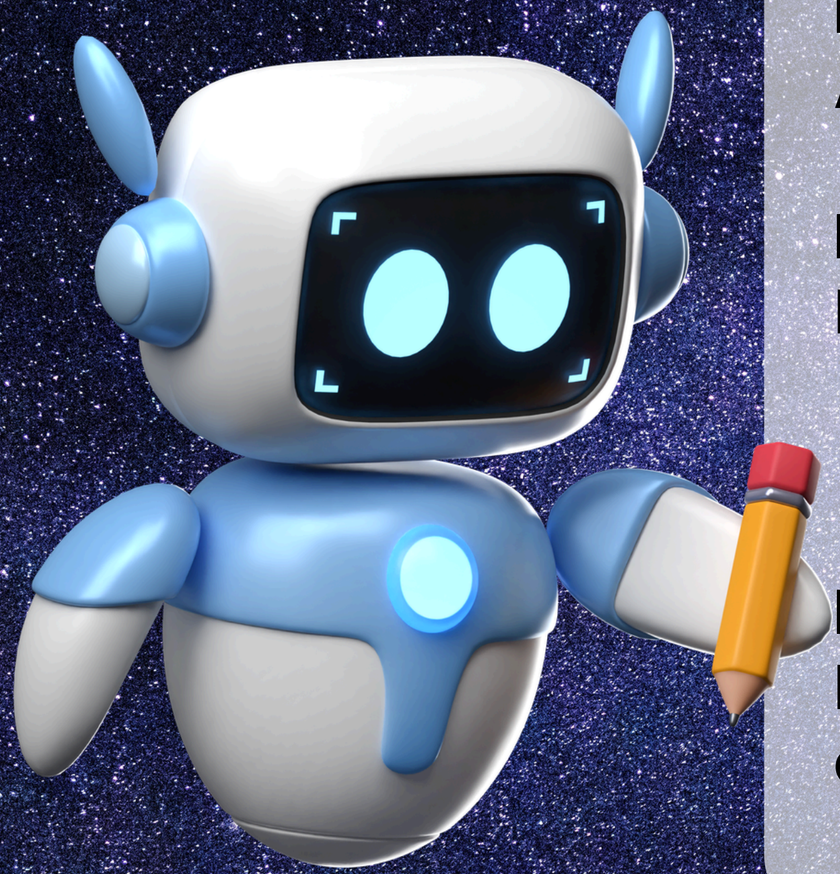
- Eliminar etiquetas HTML peligrosas.
- Usar librerías de sanitización.
- Mostrar texto como contenido plano.

Ejemplo en Python:

```python
import bleach


mensaje_seguro = bleach.clean(mensaje_usuario)
```

Mejora en seguridad
Evita la ejecución de scripts maliciosos en el navegador.

4. Implementar autenticación en dos factores (2FA)



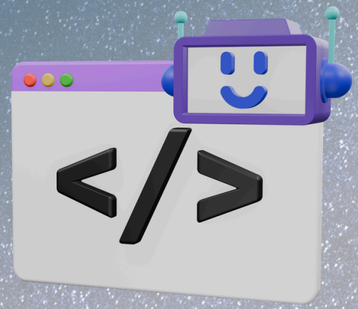
Problema que soluciona
Acceso no autorizado a cuentas.

Implementación
Después del login, pedir:

- código SMS
- aplicación autenticadora

Mejora en seguridad
Incluso si roban la contraseña, no podrán entrar sin el segundo factor.

5. Validación de consultas a la base de datos



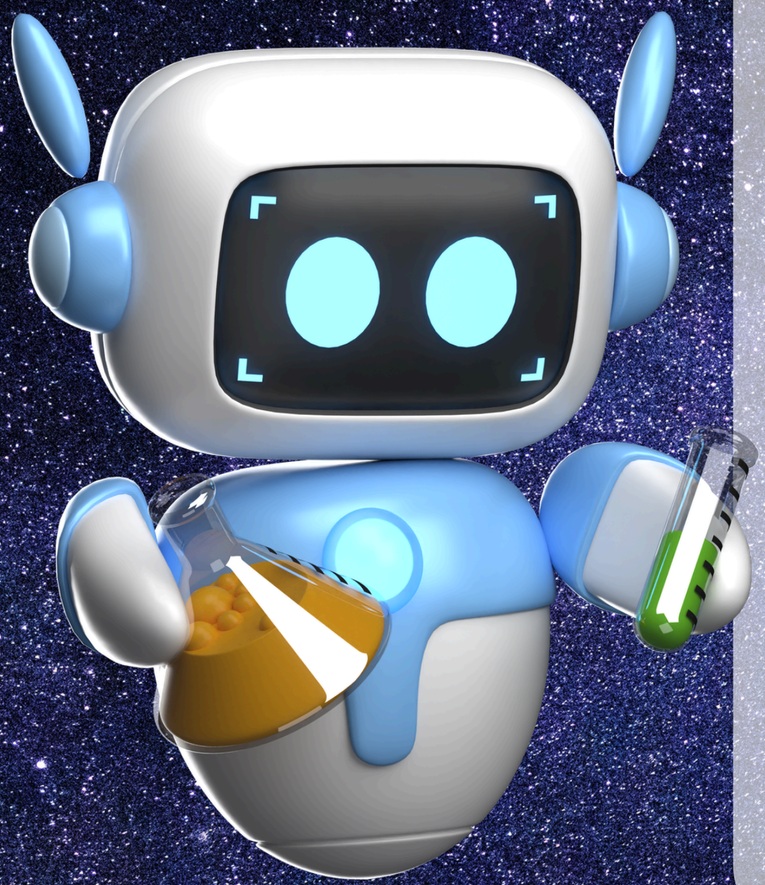
Problema que soluciona
Inyección SQL.

Implementación
Usar consultas parametrizadas.
Ejemplo:

```
cursor.execute("SELECT * FROM usuarios WHERE usuario=%s", (usuario,))
```

Mejora en seguridad
Evita que los atacantes manipulen las consultas.

6. Implementar monitoreo y registros de seguridad

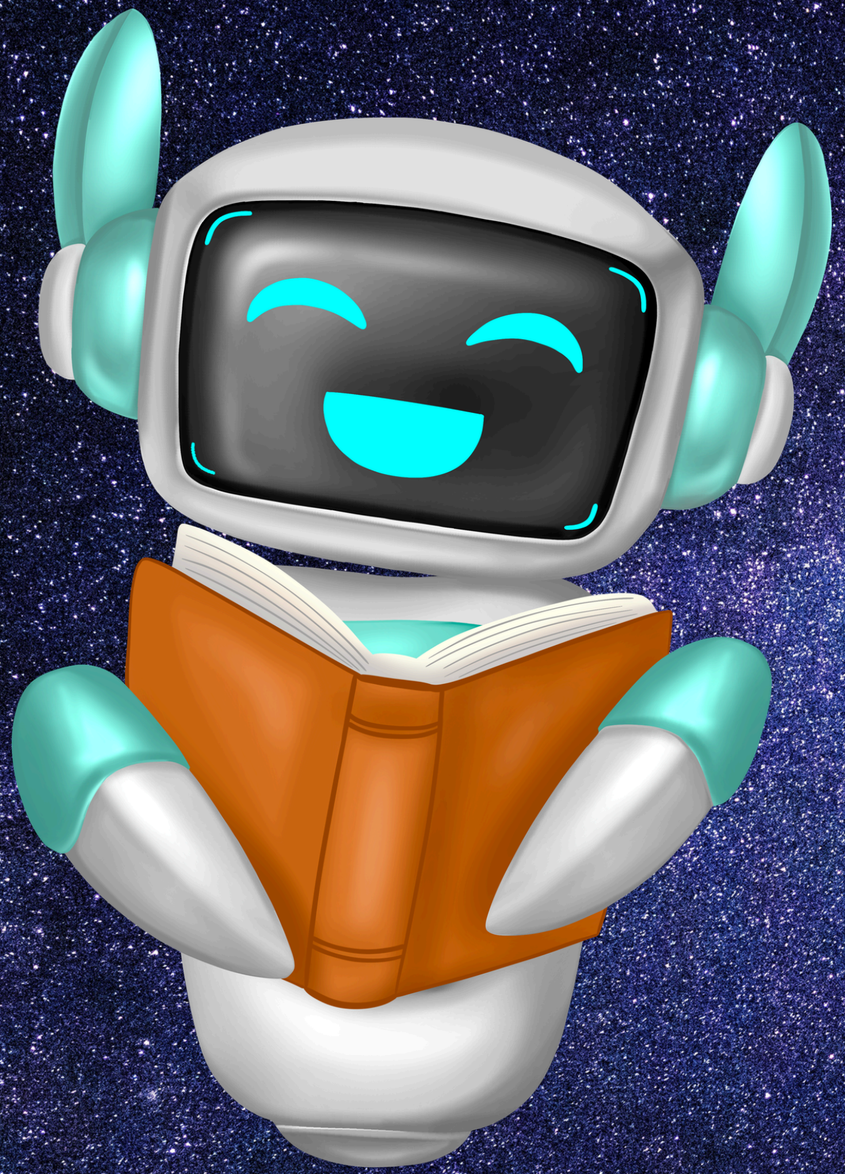


Problema que soluciona
Dificultad para detectar ataques.

Implementación
Registrar:

- intentos de login
- accesos sospechosos
- actividad inusual

Mejora en seguridad
Permite detectar ataques a tiempo.



Conclusión

EEl sistema EduPlatform presenta varias vulnerabilidades que podrían ser explotadas por atacantes, como la falta de límites en intentos de login, el almacenamiento inseguro de contraseñas y la posibilidad de insertar código HTML en los mensajes.

Para mejorar la seguridad del sistema, es necesario implementar medidas como cifrado de contraseñas, filtrado de contenido, autenticación en dos factores y control de accesos. Estas mejoras permitirán proteger la información de estudiantes y docentes, garantizando la integridad y confidencialidad del sistema.