### Ejercicio para detectar y evitar Inyecciones SQL

**Objetivo**:

Implementar un sistema de inicio de sesión básico en Java utilizando JDBC, primero con Statement y luego con PreparedStatement. Comparar ambos métodos y comprender cómo PreparedStatement previene ataques de inyección SQL.

**Enunciado**:

**Contexto**: Imagina que trabajas como desarrollador de una aplicación en Java que se conecta a una base de datos MySQL. Tu tarea es crear un sistema de inicio de sesión simple que valide las credenciales de los usuarios. La base de datos contiene una tabla usuarios con al menos dos campos: nombre (nombre de usuario) y contraseña.

1. **Implementación con** Statement:
   * Implementa un programa que solicite al usuario ingresar su nombre de usuario y contraseña.
   * Utiliza Statement para crear y ejecutar una consulta SQL que valide las credenciales ingresadas. Si el nombre de usuario y la contraseña coinciden con un registro en la tabla usuarios, muestra un mensaje de "Inicio de sesión exitoso". De lo contrario, muestra "Usuario o contraseña incorrectos".
   * Observa cómo este método **es vulnerable** a ataques de inyección SQL si se utilizan valores manipulados como entrada, por ejemplo, intenta iniciar sesión ingresando como nombre de usuario admin y como contraseña ' OR '1'='1.
2. **Implementación con** PreparedStatement:
   * Modifica el programa anterior para utilizar PreparedStatement en lugar de Statement para ejecutar la consulta SQL.
   * Asegúrate de que los valores del nombre de usuario y la contraseña se establezcan mediante parámetros en PreparedStatement.
   * Realiza pruebas con las mismas entradas que usaste en la Parte 1 y observa cómo PreparedStatement evita la ejecución de consultas manipuladas y protege contra inyecciones SQL.

**Ejemplo de Tabla de** usuarios y contraseñas:

|  |  |
| --- | --- |
| nombre | contraseña |
| admin | admin123 |
| user1 | password1 |
| user2 | password2 |