**UNIVERSIDAD NACIONAL DEL ALTIPLANO PUNO**

**INGENIERÍA DE SISTEMAS**

**INFORME DEL INYECCIÓN SQL**

**CURSO:**

BASE DE DATOS II

**DOCENTE:**

Ing. FERNANDEZ CHAMBI MAYENKA

**PRESENTADO POR:**

APAZA ANAHUI WALDIR ALEXANDER

MAMANI CUTIPA OWER LICENIO

CALDERON PILCO IVAN LEONIDAS

**SEMESTRE: V**

**PUNO – PERÚ**

**2025**

Contenido

[2.INTRODUCCIÓN 3](#_Toc195632529)

[3.Diseño y creación de base de datos en MySQL 3](#_Toc195632530)

[4.Estructura del Sistema 3](#_Toc195632531)

[4.1. Conexión de la base de datos 3](#_Toc195632532)

[4.2. OPERACIONES DE LOS CRUD 4](#_Toc195632533)

[2.3. INYECCIÓN SQL 8](#_Toc195632534)

[5.Conclusión 9](#_Toc195632535)

# INTRODUCCIÓN

En la actualidad es importante para el estudiante conocer las diferencias situaciones de seguridad que puede afrontar en el desarrollo de aplicaciones web u otras, es por tal motivo el objetivo del informe. El proyecto realizado es un CRUD diseñado con fines didácticos para ver de primera mano una de estas formas, en este caso, la inyección SQL. Para tal fin, se creó una interfaz gráfica con las 4 operaciones entrelazadas en el transcurso de él, para ello se usó Codeigniter 4, PHP, framework Bootstrap 5 y MySQL.

# Diseño y creación de base de datos en MySQL

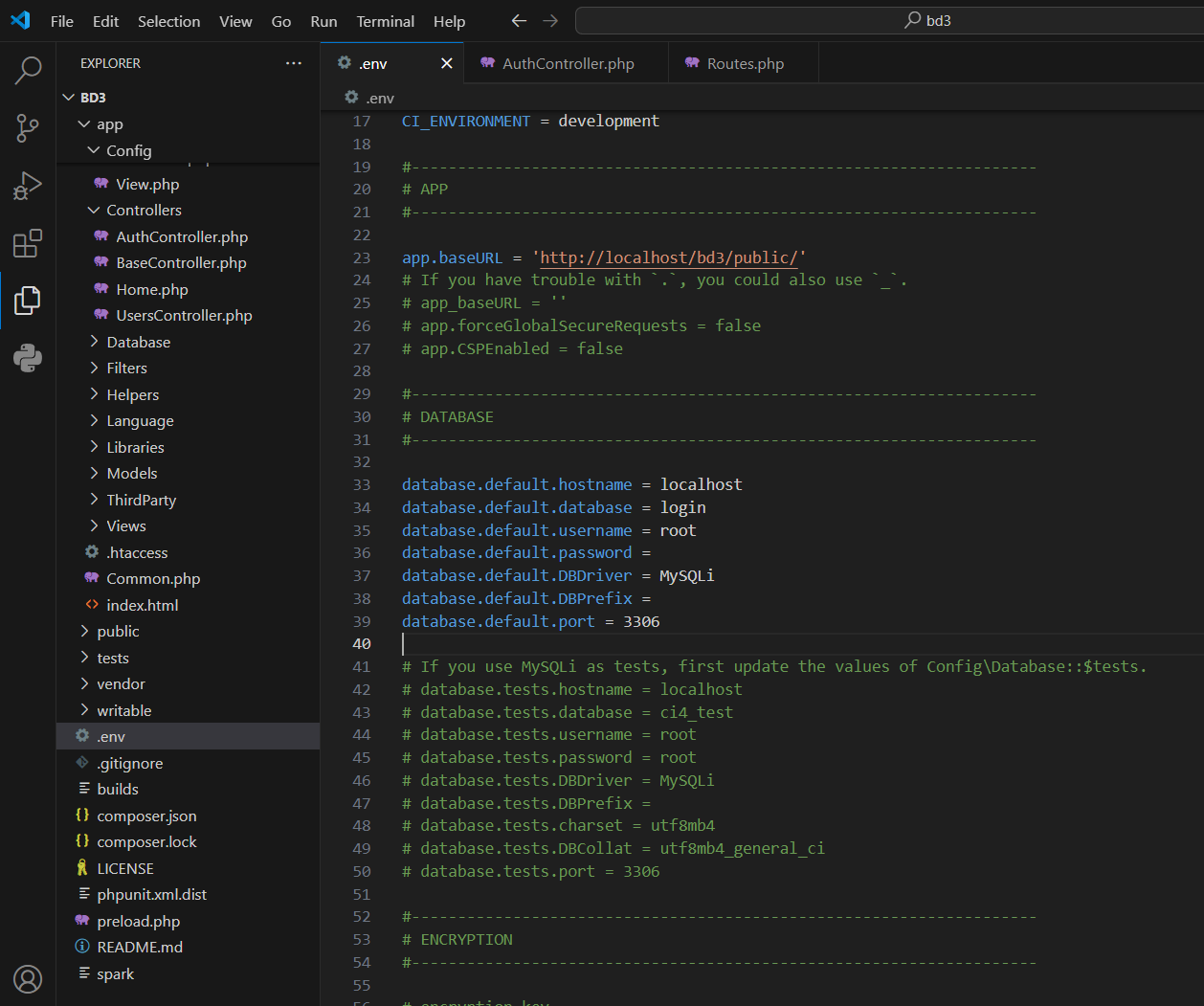
Dentro de esta base de datos, se ha diseñado una tabla llamada **users**, encargada de almacenar los datos principales de cada usuario. El diseño de esta tabla responde tanto a necesidades funcionales como a criterios de integridad y seguridad de los datos.

|  |
| --- |
| CREATE TABLE users **(**  id INT AUTO\_INCREMENT PRIMARY KEY**,**  photo\_name VARCHAR**(**255**)** DEFAULT **NULL,**  full\_name VARCHAR**(**100**)** NOT **NULL,**  username VARCHAR**(**50**)** UNIQUE NOT **NULL,**  email VARCHAR**(**100**)** UNIQUE NOT **NULL,**  password VARCHAR**(**255**)** NOT **NULL**  **);** |

# Estructura del Sistema

## Conexión de la base de datos

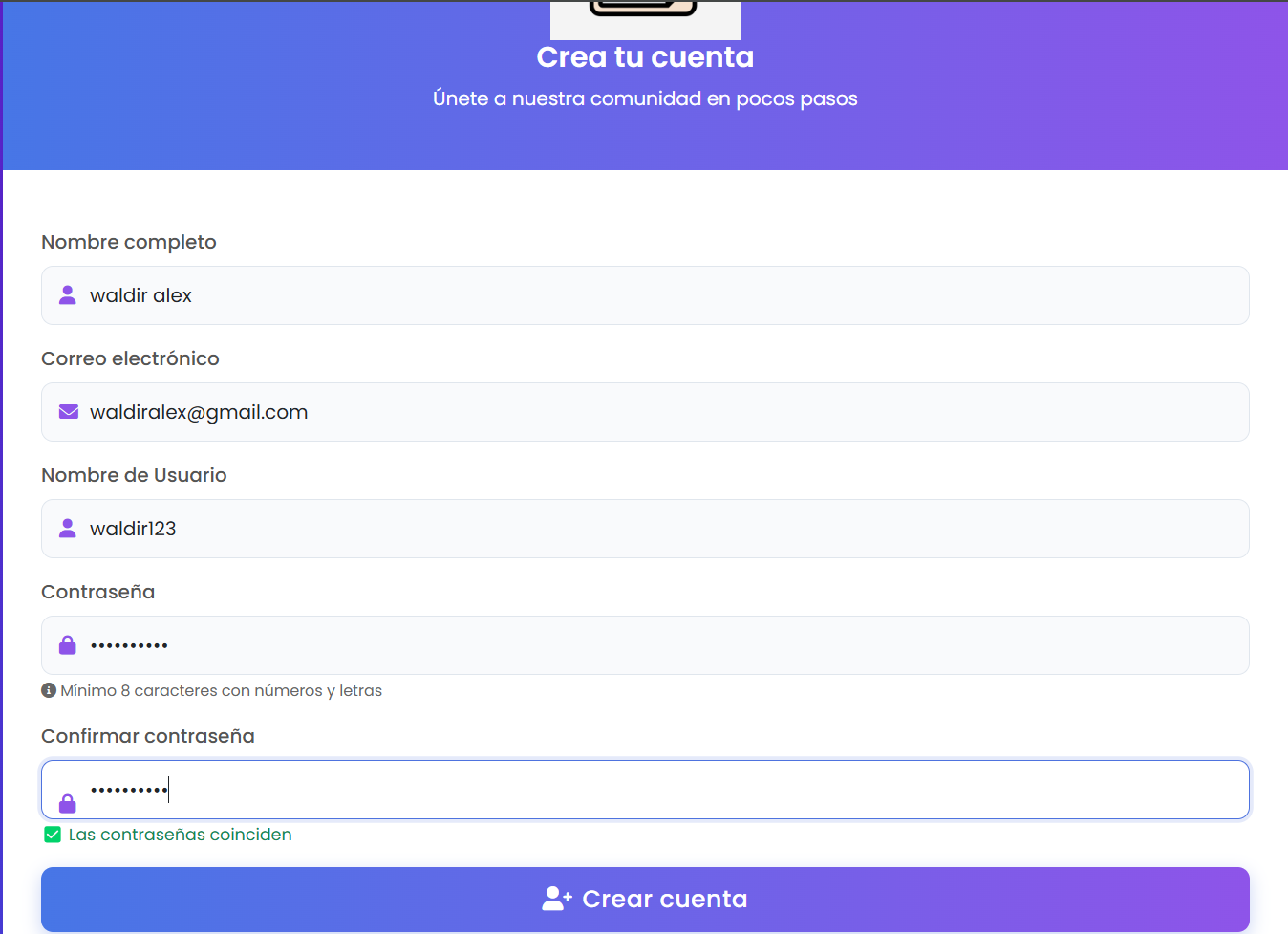
La conexión a la base de datos en Codeigniter 4 se da por medio del archivo de configuración ‘.env’ en el cual, se llenan segmentos concernientes al hostname, nombre de la base de datos, usuario y contraseña, driver y puerto.



## OPERACIONES DE LOS CRUD

**CREAR (INSERTAR USUARIO)**

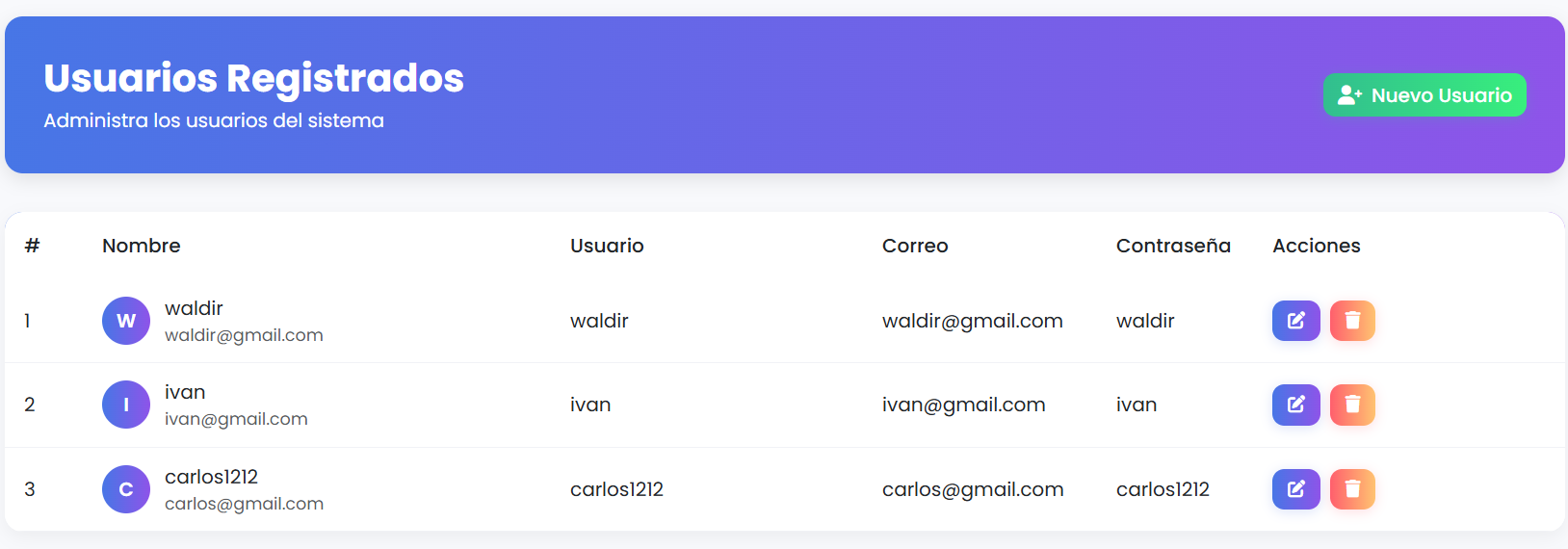
Primero se definen reglas de validación para asegurarse de que todos los campos requeridos estén completos y correctos, incluyendo la confirmación de contraseña. Si hay errores, se vuelve al formulario; si todo está bien, se instancia el modelo UserModel y se guarda la información en la base de datos usando el método save(). Este proceso garantiza que los datos del nuevo usuario se almacenen correctamente.



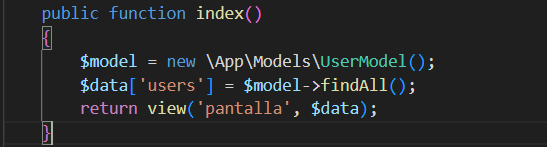


**LEER (MOSTRAR USUARIOS)**

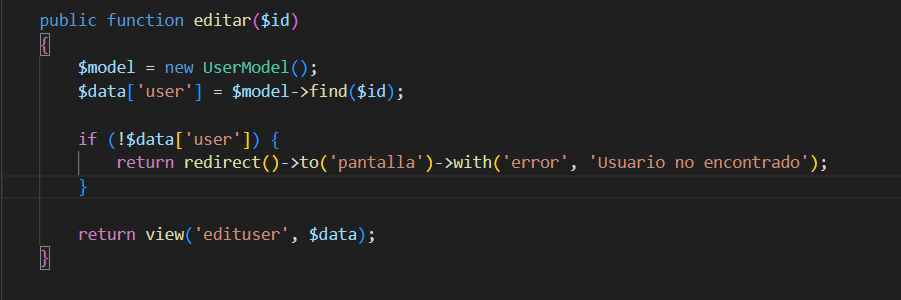
La lectura de usuarios se hace principalmente en el método index(), donde se consulta toda la lista de usuarios mediante findAll() del modelo UserModel. Luego, se pasan estos datos a una vista (pantalla) para ser mostrados, usualmente en una tabla. También el método editar($id) recupera un usuario en particular mediante find($id) para llenar un formulario con sus datos antes de editarlos.



Mostrar todos los usuarios

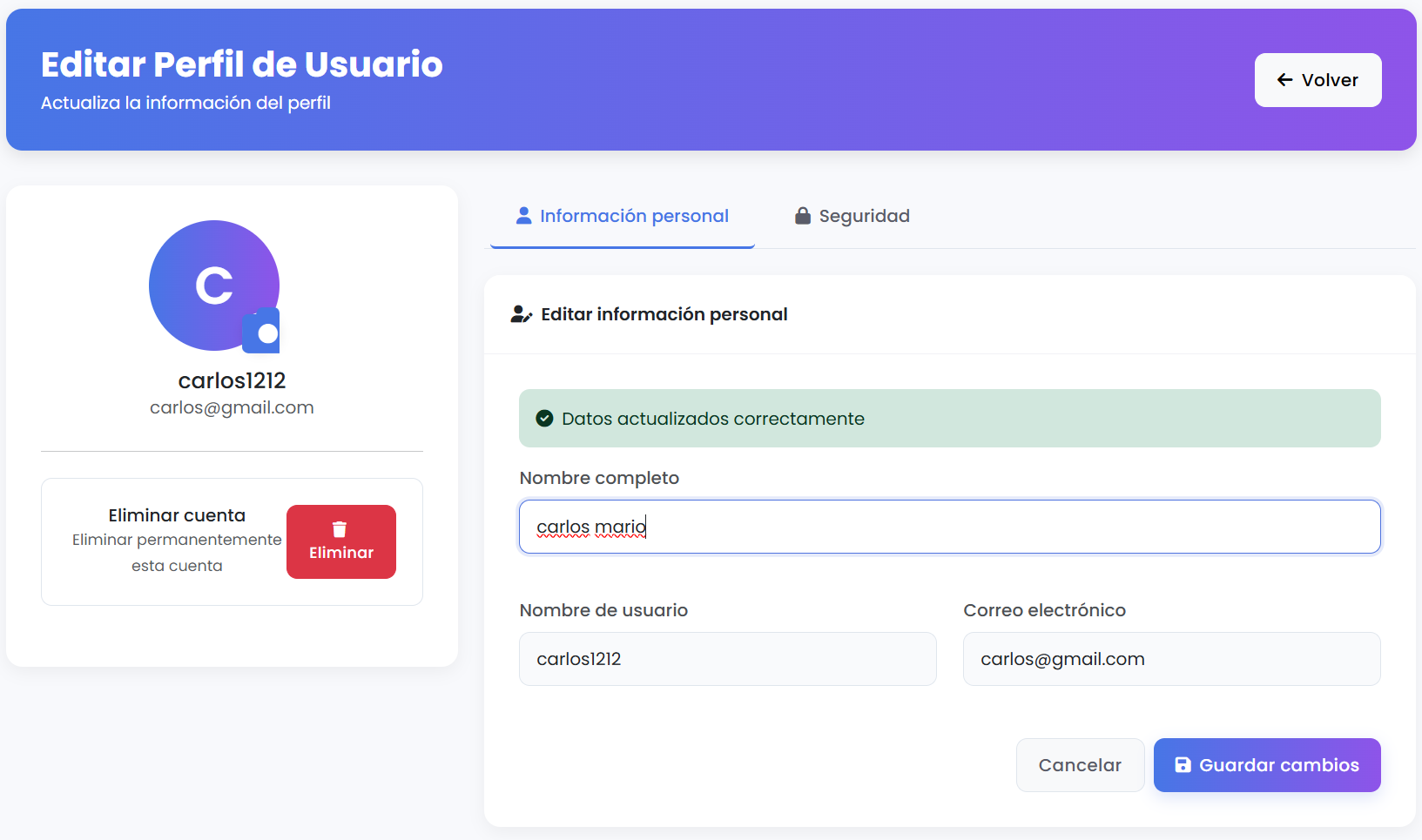


Mostrar usuarios especifico

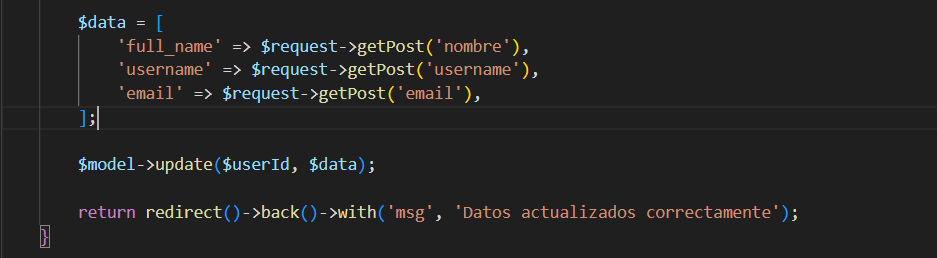


**ACTUALIZAR (MODIFICAR DATOS DEL USUARIO)**

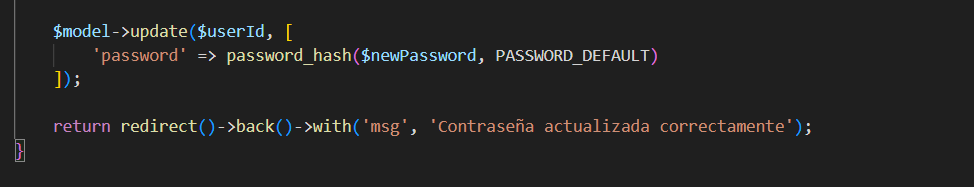
La actualización de un usuario se realiza en el método actualizar (). Se recibe el ID del usuario desde el formulario y se validan los nuevos datos. Si la validación es exitosa, se llama al método update($id, $data) del modelo para aplicar los cambios. También hay un método edit\_password() que permite cambiar la contraseña, comparando primero la contraseña actual ingresada con la que está en la base de datos, y si es correcta, actualiza a la nueva.



Actualizar dato generale de un usuario

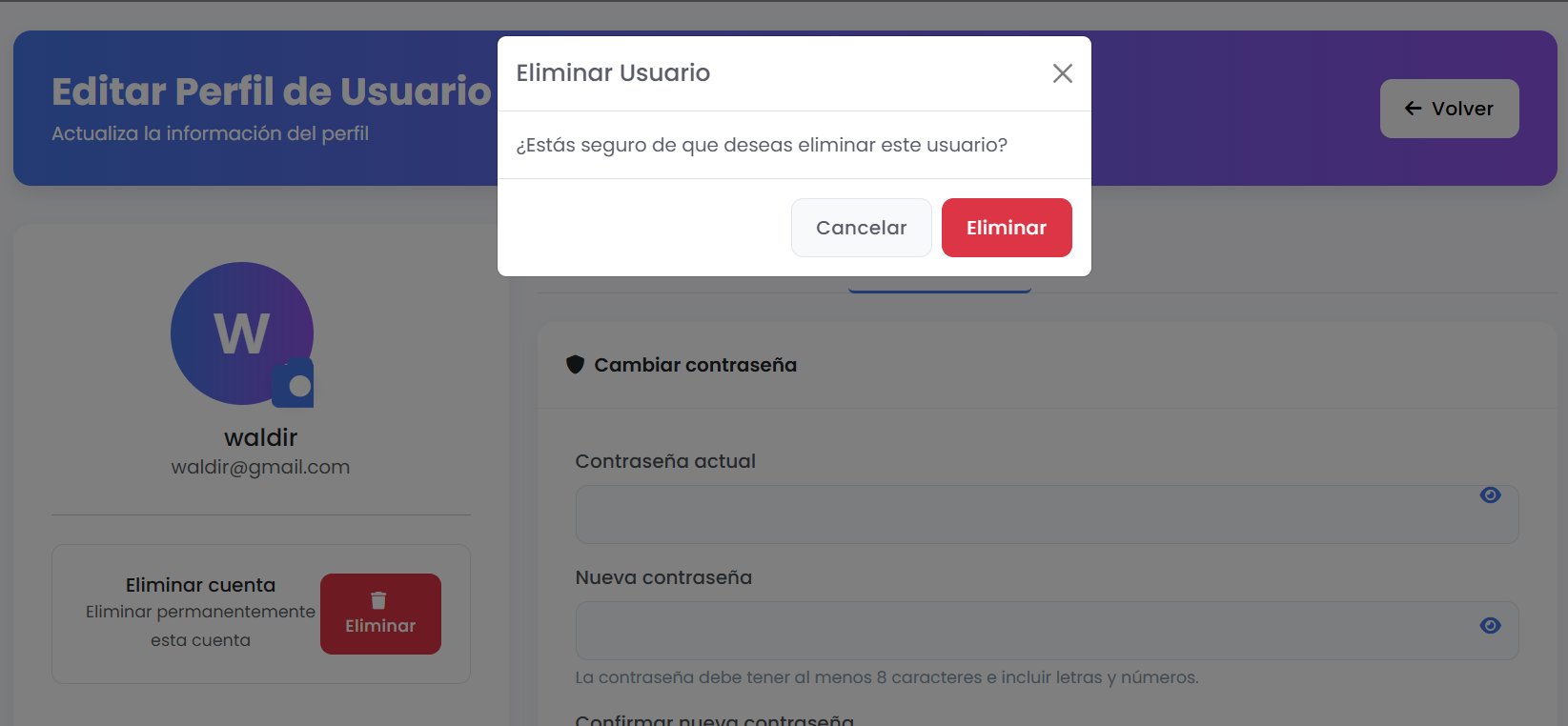


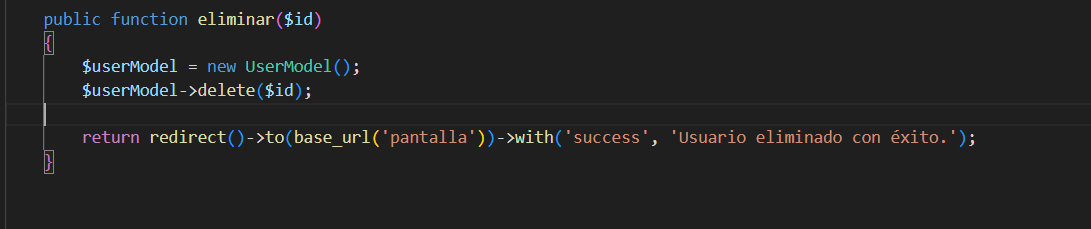
Actualizar contraseña del usuario



**ELIMINAR (ELIMINAR USUARIO)**

La eliminación se ejecuta con el método eliminar($id). Se recibe el ID del usuario a eliminar, se instancia el modelo y se utiliza el método delete($id) para borrarlo de la base de datos. Luego, se redirige al listado de usuarios mostrando un mensaje que confirma la eliminación. Esta operación es irreversible y elimina completamente el registro.





## 

## 2.3. INYECCIÓN SQL

|  |  |
| --- | --- |
| CÓDIGO VULNERABLE A INYECCIÓN SQL | EXPLICACIÓN |
|  | El código es vulnerable a inyección SQL porque concatena directamente los datos ingresados por el usuario sin validarlos ni utilizar sentencias preparadas. Le falta una separación adecuada entre los datos y la lógica de la consulta, lo que permite que un atacante inyecte condiciones maliciosas como ' OR '1'='1, alteren la verificación e ingresen sin credenciales válidas. |
| CÓDIGO CON PREVISIONES ANTE UN ATAQUE DE INYECCIÓN SQL | Este código utiliza Query Builder, específicamente métodos como where() y first(), los cuales internamente emplean consultas preparadas. Estas consultas separan los datos del usuario de la estructura de la instrucción SQL, evitando que cualquier texto malicioso modifique la lógica de la consulta. Gracias a esto, incluso si un atacante intenta inyectar código, no tendrá efecto porque los valores se tratan como simples datos, no como parte de la consulta SQL. Por lo tanto, este enfoque no es vulnerable a inyección SQL y representa una práctica segura en el manejo de credenciales. |
|  |

# Conclusión

La implementación del CRUD permitió conocer una de las situaciones de seguridad a tener en cuenta en el desarrollo de sistemas, como es la inyección SQL. Asimismo, la profundización y/o fomento de nuevos conocimientos tecnológicos, como filtros y algoritmos de protección, lo cual entabla en un mejor desarrollo de habilidades de lógica computacional y base de datos.

Nota: Añadir cuatro líneas de texto.

