Si no puedes modificar la base de datos, puedes seguir una estrategia alternativa para que el **token expire en 5 minutos** sin añadir una nueva columna.

**Estrategia: Token con Marca de Tiempo Codificada**

Dado que no puedes modificar la base de datos para agregar un campo de expiración, puedes incluir la **hora de generación dentro del propio token** y validarla al momento de la confirmación.

**¿Cómo Funciona?**

1. **Generar un token seguro**:
   * Incluye un **timestamp encriptado** junto con el token.
2. **Enviar el token por correo**.
3. **Validar el token al recibir la solicitud**:
   * Extraer el timestamp del token y verificar que no hayan pasado más de **5 minutos**.
   * Si ha expirado, rechazar la solicitud.

**Implementación en PHP**

Aquí está el código mejorado **sin modificar la base de datos**:

**1️⃣ Generar el Token con Expiración**

if (count($errores) === 0) {

$\_SESSION['pass'] = password\_hash($pass2, PASSWORD\_DEFAULT); // Hash de la nueva contraseña

// Generar token aleatorio

$randomToken = bin2hex(random\_bytes(16));

// Incluir timestamp actual en el token (para validarlo después)

$timestamp = time(); // Hora actual en segundos

$token = "$randomToken|$timestamp"; // Formato "token|timestamp"

// Guardar token en la base de datos (sin modificar estructura)

$conexion = conectarPDO($host, $user, $password, $bbdd);

$insertToken = $conexion->prepare('UPDATE usuarios SET token=?, activo="0", fecha=CURRENT\_TIMESTAMP WHERE email =?');

$insertToken->execute([$randomToken, $\_SESSION['email']]);

desconectarPDO($insertToken, $conexion);

if ($insertToken->rowCount() > 0) {

// Codificar el token en la URL

$tokenEncode = urlencode(base64\_encode($token));

// Enviar el correo

$headers = "From: dwes@php.com\r\nContent-type: text/plain; charset=utf-8";

$textoEmail = "

Hola!\n

Active su nueva contraseña pulsando en el siguiente enlace:\n

http://localhost:3000/DWS/bbdd\_rol\_usuario/restablecer/linkConfirmacion.php?token=$tokenEncode

";

mail($\_SESSION['email'], 'Restablece tu contraseña', $textoEmail, $headers);

header('Location: mensaje\_confirmacion.php');

exit();

}

}

**2️⃣ Validar el Token en linkConfirmacion.php**

Cuando el usuario hace clic en el enlace, extraemos el **timestamp del token** y verificamos si ha expirado.

if (isset($\_GET['token'])) {

$tokenDecode = base64\_decode($\_GET['token']); // Decodificar el token de la URL

list($randomToken, $timestamp) = explode('|', $tokenDecode); // Separar token y timestamp

$tiempoActual = time();

$diferencia = $tiempoActual - $timestamp; // Diferencia de tiempo en segundos

if ($diferencia > 300) { // Si han pasado más de 5 minutos (300 segundos)

die("❌ Enlace expirado. Solicita un nuevo restablecimiento de contraseña.");

}

// Validar el token en la base de datos

$conexion = conectarPDO($host, $user, $password, $bbdd);

$stmt = $conexion->prepare('SELECT email FROM usuarios WHERE token=?');

$stmt->execute([$randomToken]);

if ($stmt->rowCount() > 0) {

$usuario = $stmt->fetch(PDO::FETCH\_ASSOC);

$email = $usuario['email'];

// Actualizar la contraseña almacenada en la sesión

$updatePass = $conexion->prepare('UPDATE usuarios SET password=?, token=NULL, activo="1" WHERE email=?');

$updatePass->execute([$\_SESSION['pass'], $email]);

echo "✅ Contraseña restablecida con éxito.";

} else {

die("❌ Token inválido.");

}

}

**🔹 Ventajas de Esta Solución**

✅ **No modifica la base de datos**. Se usa la columna token existente.  
✅ **El token expira automáticamente** al incluir un timestamp dentro de él.  
✅ **Protege contra ataques de reutilización**.  
✅ **No se almacena la contraseña en la URL**, lo que mejora la seguridad.

Con esta solución, el enlace solo será válido **durante 5 minutos** sin cambiar la estructura de la base de datos. 🚀

¡Genial! Si decides seguir usando $tokenEncode = urlencode($timeToken);, asegúrate de que la estructura del token siga incluyendo el **timestamp**, ya que eso es lo que permite la expiración sin modificar la base de datos.

**🔹 Consideraciones Clave para tu Implementación**

1. **Mantén el formato del token** → "token|timestamp"
2. **Usa base64\_encode antes de urlencode**
3. **Verifica la expiración en linkConfirmacion.php**

**1️⃣ Generar el Token**

if (count($errores) === 0) {

$\_SESSION['pass'] = password\_hash($pass2, PASSWORD\_DEFAULT); // Almacenar hash de la contraseña

// Generar token aleatorio

$randomToken = bin2hex(random\_bytes(16));

$timestamp = time(); // Timestamp actual en segundos

// Crear el token con timestamp

$timeToken = "$randomToken|$timestamp";

// Guardar solo el token aleatorio en la base de datos (sin modificar estructura)

$conexion = conectarPDO($host, $user, $password, $bbdd);

$insertToken = $conexion->prepare('UPDATE usuarios SET token=?, activo="0", fecha=CURRENT\_TIMESTAMP WHERE email =?');

$insertToken->execute([$randomToken, $\_SESSION['email']]);

desconectarPDO($insertToken, $conexion);

if ($insertToken->rowCount() > 0) {

// Codificar el token para enviarlo en la URL

$tokenEncode = urlencode(base64\_encode($timeToken));

// Enviar el correo

$headers = "From: dwes@php.com\r\nContent-type: text/plain; charset=utf-8";

$textoEmail = "

Hola!\n

Active su nueva contraseña pulsando en el siguiente enlace:\n

http://localhost:3000/DWS/bbdd\_rol\_usuario/restablecer/linkConfirmacion.php?token=$tokenEncode

";

mail($\_SESSION['email'], 'Restablece tu contraseña', $textoEmail, $headers);

header('Location: mensaje\_confirmacion.php');

exit();

}

}

**2️⃣ Validar el Token en linkConfirmacion.php**

if (isset($\_GET['token'])) {

$tokenDecode = base64\_decode(urldecode($\_GET['token'])); // Decodificar el token

list($randomToken, $timestamp) = explode('|', $tokenDecode); // Separar token y timestamp

$tiempoActual = time();

$diferencia = $tiempoActual - $timestamp; // Diferencia de tiempo en segundos

if ($diferencia > 300) { // Si han pasado más de 5 minutos (300 segundos)

die("❌ Enlace expirado. Solicita un nuevo restablecimiento de contraseña.");

}

// Validar el token en la base de datos

$conexion = conectarPDO($host, $user, $password, $bbdd);

$stmt = $conexion->prepare('SELECT email FROM usuarios WHERE token=?');

$stmt->execute([$randomToken]);

if ($stmt->rowCount() > 0) {

$usuario = $stmt->fetch(PDO::FETCH\_ASSOC);

$email = $usuario['email'];

// Actualizar la contraseña almacenada en la sesión

$updatePass = $conexion->prepare('UPDATE usuarios SET password=?, token=NULL, activo="1" WHERE email=?');

$updatePass->execute([$\_SESSION['pass'], $email]);

echo "✅ Contraseña restablecida con éxito.";

} else {

die("❌ Token inválido.");

}

}

**🛡 Seguridad Adicional**

Si quieres hacer el sistema aún más seguro:

* 🔒 **Hashea el token antes de guardarlo en la BD** para evitar filtraciones:
* $hashedToken = password\_hash($randomToken, PASSWORD\_DEFAULT);
* $insertToken->execute([$hashedToken, $\_SESSION['email']]);
* ✅ **Usa password\_verify()** en linkConfirmacion.php para comparar el token enviado:
* if (password\_verify($randomToken, $hashedTokenDB)) {
* // Token válido
* }

Así mantienes la **máxima seguridad posible** sin modificar la base de datos. 🚀

¡Buena elección de estrategia! Si necesitas más ajustes, dime. 😊

FValidación de email

Esta línea de código en PHP está validando si una cadena de texto es una dirección de correo electrónico válida usando la función filter\_var() con el filtro FILTER\_VALIDATE\_EMAIL. Vamos a desglosarla:

return (filter\_var($texto, FILTER\_VALIDATE\_EMAIL) === FALSE) ? False : True;

**Explicación:**

1. **filter\_var($texto, FILTER\_VALIDATE\_EMAIL)**
   * filter\_var() es una función que filtra variables usando un filtro específico.
   * FILTER\_VALIDATE\_EMAIL verifica si el contenido de $texto es un correo electrónico válido.
   * Si $texto es un correo válido, la función devuelve el correo validado.
   * Si no es válido, devuelve FALSE.
2. **Comparación estricta con === FALSE**
   * filter\_var($texto, FILTER\_VALIDATE\_EMAIL) === FALSE verifica si la validación falló.
   * Si es FALSE, significa que $texto no es un correo válido.
3. **Operador ternario ? :**
   * Si la validación devuelve FALSE, el operador ternario retorna False.
   * Si la validación es exitosa (es un correo válido), retorna True.

**Forma simplificada:**

En PHP, la comparación con FALSE y el operador ternario son innecesarios en este caso, ya que filter\_var() ya devuelve FALSE o el valor validado (que se evalúa como true en un contexto booleano). Se podría escribir simplemente:

return filter\_var($texto, FILTER\_VALIDATE\_EMAIL) !== FALSE;

Este código hace exactamente lo mismo de una manera más limpia y eficiente.