🔐 Flujo de Recuperación de Contraseña - YigiCoin

📋 Resumen de Implementación

Se ha implementado un flujo completo de recuperación de contraseña con **3 pasos** que incluye envío real de emails usando **Resend** y actualización tanto de contraseña como de PIN.

✨ Cambios Implementados

1. Modificación en `/app/login/page.tsx`

Antes:

```
<button type="button" className="text-blue-600 hover:text-blue-800 cursor-pointer">
  ¿Olvidaste tu contraseña?
</button>
```

Después:

```
<a
  href="/recuperar-password"
  target="_blank"
  rel="noopener noreferrer"
  className="text-blue-600 hover:text-blue-800 cursor-pointer transition-colors"
>
  ¿Olvidaste tu contraseña?
</a>
```

✅ **Resultado:** El enlace “¿Olvidaste tu contraseña?” ahora abre la página de recuperación en una **nueva pestaña**.

---

2. Página `/app/recuperar-password/page.tsx`

La página ya estaba implementada con el flujo de 3 pasos completo:

Paso 1: Solicitar Email

- Usuario ingresa su correo electrónico
- Se llama a `POST /api/auth/forgot-password/send-code`
- Se envía un código de 6 dígitos al email del usuario
- Se muestra mensaje de éxito

Paso 2: Verificar Código

- Usuario ingresa el código recibido por email
- Se valida el formato (6 dígitos numéricos)

- Opción de reenviar el código
- Opción de volver al paso 1 para cambiar el email

Paso 3: Nueva Contraseña y PIN

- Usuario ingresa:
- Nueva contraseña (mínimo 8 caracteres)
- Confirmación de contraseña
- Nuevo PIN (4 dígitos)
- Confirmación de PIN
- Se muestran indicadores de seguridad en tiempo real:
- ✅ Contraseña: mínimo 8 caracteres
- ✅ Al menos una mayúscula
- ✅ Al menos un número
- ✅ Las contraseñas coinciden
- ✅ El PIN es de 4 dígitos y coincide
- Se llama a `POST /api/auth/forgot-password/reset`
- Tras éxito, **redirección automática a** `/login` después de 1.5 segundos

---

3. Helper `/lib/emailVerification.ts`

Se agregó la función `sendPasswordResetEmail`:

```
export async function sendPasswordResetEmail(email: string, code: string)
```

Características:
- ✉️ Envía emails profesionales con HTML estilizado
- 🎨 Diseño responsive y profesional
- ⏰ Incluye advertencia de expiración (10 minutos)
- 🔒 Nota de seguridad
- 📝 Footer con información de copyright
- 🔄 Fallback a console.log si Resend no está configurado

Email incluye:
- Título: "🔐 Recuperación de Contraseña"
- Código destacado en grande con estilo visual
- Advertencia de expiración
- Nota de seguridad
- Footer informativo

---

4. API `/api/auth/forgot-password/send-code/route.ts`

Mejoras implementadas:

```
// Importación de la nueva función
import { generateCode, hashCode, sendPasswordResetEmail } from '@/lib/emailVerifica-
tion';

// Envío real del email
try {
    await sendPasswordResetEmail(email, code);
    console.log(`✅ Código de recuperación enviado a ${email}`);
} catch (emailError) {
    console.error('Error al enviar email:', emailError);
    console.log(`⚠️ FALLBACK - Código de recuperación para ${email}: ${code}`);
}
```

Flujo:
1. ✅ Valida que el email sea válido
2. ✅ Verifica que el usuario exista en la base de datos
3. ✅ Genera código de 6 dígitos
4. ✅ Hashea el código con SHA-256
5. ✅ Guarda en tabla `PasswordReset` con expiración de 10 minutos
6. ✅ **Envía email con Resend**
7. ✅ Logs informativos para debugging
8. ✅ Manejo de errores con fallback

---

5. API `/api/auth/forgot-password/reset/route.ts`

Mejoras implementadas:

```
console.log(`✅ Contraseña y PIN actualizados exitosamente para ${email}`);
return NextResponse.json({
    success: true,
    message: 'Contraseña y PIN actualizados exitosamente'
});
```

Flujo completo:
1. ✅ Valida que todos los campos estén presentes (email, code, password, pin)
2. ✅ Valida formato del PIN (4 dígitos numéricos)
3. ✅ Verifica que el usuario exista
4. ✅ Busca el código más reciente no usado
5. ✅ Verifica que el código no haya expirado
6. ✅ Verifica que el código sea correcto (comparación de hashes)
7. ✅ Hashea la nueva contraseña con bcrypt
8. ✅ Hashea el nuevo PIN con bcrypt
9. ✅ Actualiza en una **transacción atómica**:
- Marca el código como usado
- Actualiza passwordHash y pinHash del usuario
10. ✅ Logs informativos para auditoría
11. ✅ Retorna mensaje de éxito

🗄️ Estructura de Base de Datos

Modelo `PasswordReset` (Prisma)

```
model PasswordReset {
  id        String   @id @default(cuid())
  email     String
  codeHash  String
  expiresAt DateTime
  used      Boolean  @default(false)
  createdAt DateTime @default(now())

  @@index([email])
}
```

Campos:
- `email`: Email del usuario
- `codeHash`: Hash SHA-256 del código de 6 dígitos
- `expiresAt`: Fecha de expiración (10 minutos)
- `used`: Flag para prevenir reutilización
- `createdAt`: Timestamp de creación

🔒 Seguridad Implementada

✅ Protecciones

1. **Códigos hasheados**: Los códigos se almacenan como SHA-256, no en texto plano
2. **Expiración**: Códigos expiran en 10 minutos
3. **Un solo uso**: Los códigos se marcan como `used` tras resetear
4. **Validación de formato**:
   - Email válido
   - Código de 6 dígitos
   - PIN de 4 dígitos numéricos
   - Contraseña mínimo 8 caracteres
5. **Bcrypt para passwords**: Contraseña y PIN se hashean con bcrypt (10 rounds)
6. **Transacciones atómicas**: Actualización de usuario y marcado de código en una transacción
7. **Logs de auditoría**: Todas las operaciones se registran

⚠️ Consideraciones de Producción

```
// Actualmente muestra error si el email no existe:
{ error: 'No existe una cuenta con este correo' }

// En producción, considerar:
{ success: true } // Siempre, para no filtrar usuarios válidos
```

📧 Configuración de Resend

Variables de Entorno

Asegúrate de tener configurado en `.env`:

```
RESEND_API_KEY=re_xxxxxxxxxxxxxxxxxxxxxxxx
```

Configuración del Dominio

En `lib/emailVerification.ts`, actualiza el dominio cuando tengas uno verificado:

```
from: 'YigiCoin <no-reply@yigicoin.com>', // Actualizar con tu dominio verificado
```

Fallback Sin Resend

Si `RESEND_API_KEY` no está configurado:
- ✅ El código se muestra en console.log
- ✅ La aplicación funciona normalmente
- ⚠️ Los emails NO se envían

🧪 Testing del Flujo

1. Test Manual del Flujo Completo

```
# 1. Iniciar la aplicación
npm run dev

# 2. Ir a http://localhost:3000/login
# 3. Hacer clic en "¿Olvidaste tu contraseña?"
# 4. Se abre nueva pestaña con /recuperar-password

# PASO 1: Ingresar email
# - Ingresar email de usuario existente
# - Clic en "Enviar Código de Verificación"
# - Verificar que llegó el email (o revisar logs)

# PASO 2: Verificar código
# - Ingresar código de 6 dígitos del email
# - Clic en "Verificar Código"

# PASO 3: Nueva contraseña y PIN
# - Ingresar nueva contraseña (min 8 caracteres)
# - Confirmar contraseña
# - Ingresar nuevo PIN (4 dígitos)
# - Confirmar PIN
# - Clic en "Actualizar Contraseña y PIN"
# - Verificar redirección automática a /login

# 5. Iniciar sesión con nueva contraseña y nuevo PIN
```

2. Test de APIs con cURL

Enviar código:

```
curl -X POST http://localhost:3000/api/auth/forgot-password/send-code \
  -H "Content-Type: application/json" \
  -d '{"email":"usuario@ejemplo.com"}'
```

Respuesta esperada:

```
{"success":true,"message":"Código enviado exitosamente"}
```

Resetear contraseña:

```
curl -X POST http://localhost:3000/api/auth/forgot-password/reset \
  -H "Content-Type: application/json" \
  -d '{
    "email":"usuario@ejemplo.com",
    "code":"123456",
    "password":"NuevaPassword123",
    "pin":"1234"
  }'
```

Respuesta esperada:

```
{"success":true,"message":"Contraseña y PIN actualizados exitosamente"}
```

3. Casos de Error a Verificar

Escenario	Comportamiento Esperado
Email no existe	Error: “No existe una cuenta con este correo”
Código incorrecto	Error: “Código incorrecto”
Código expirado	Error: “El código ha expirado, solicita uno nuevo”
PIN inválido	Error: “El PIN debe ser de 4 dígitos numéricos”
Contraseña corta	Error: “La contraseña debe tener al menos 8 caracteres”
Contraseñas no coinciden	Error: “Las contraseñas no coinciden”
PINs no coinciden	Error: “Los PIN no coinciden”
Código ya usado	Error: “No hay código activo para este correo”

📊 Logs para Debugging

Logs de Éxito

```
✅ Código de recuperación enviado a usuario@ejemplo.com
✅ Contraseña y PIN actualizados exitosamente para usuario@ejemplo.com
```

Logs de Fallback

```
⚠️ FALLBACK - Código de recuperación para usuario@ejemplo.com: 123456
Código de recuperación de contraseña para usuario@ejemplo.com: 123456
```

Logs de Error

```
Error al enviar email: [detalles del error]
Error en send-code: [detalles del error]
Error en reset: [detalles del error]
```

📱 UI/UX

Características de la Interfaz

1. **Indicador de Progreso Visual**
   - 3 pasos claramente marcados
   - Barra de progreso entre pasos
   - Labels descriptivos

2. **Validación en Tiempo Real**
   - ✅ Indicadores verdes para requisitos cumplidos
   - ❌ Indicadores grises para requisitos pendientes
   - Requisitos de seguridad visibles

3. **Mensajes al Usuario**
   - 🟢 Mensajes de éxito en verde
   - 🔴 Mensajes de error en rojo
   - ℹ️ Información de ayuda

4. **Experiencia de Usuario**
   - Botón “Reenviar código” disponible
   - Opción de volver al paso anterior
   - Spinner de carga durante peticiones
   - Campos con iconos descriptivos
   - Botones de mostrar/ocultar contraseña

5. **Responsive Design**
   - Diseño adaptable a móviles
   - Background con overlay
   - Logo de YigiCoin
   - Tarjetas con sombras y bordes

🔄 Flujo de Redirección

```
/login
  └─> [Click "¿Olvidaste tu contraseña?"]
       └─> Nueva pestaña: /recuperar-password
            └─> Paso 1: Ingresar email ✉️
                 └─> Paso 2: Verificar código 🔢
                      └─> Paso 3: Nueva contraseña y PIN 🔒
                           └─> [Success] Auto-redirect después de 1.5s
                                └─> /login 🔐
```

📝 Archivos Modificados

```
✅ /app/login/page.tsx
   - Cambio de button a <a> con target="_blank"

✅ /lib/emailVerification.ts
   - Nueva función: sendPasswordResetEmail()
   - Email HTML profesional y estilizado

✅ /app/api/auth/forgot-password/send-code/route.ts
   - Integración con sendPasswordResetEmail()
   - Logs mejorados
   - Manejo de errores con fallback

✅ /app/api/auth/forgot-password/reset/route.ts
   - Logs mejorados
   - Mensaje de éxito más descriptivo
```

✅ Checklist de Implementación

- [x] Enlace de recuperación abre en nueva pestaña
- [x] Flujo de 3 pasos implementado
- [x] Envío real de emails con Resend
- [x] Email HTML profesional y estilizado
- [x] Validación de formato de código (6 dígitos)
- [x] Validación de formato de PIN (4 dígitos)
- [x] Validación de contraseña (mínimo 8 caracteres)
- [x] Indicadores de seguridad en tiempo real
- [x] Códigos hasheados con SHA-256
- [x] Passwords hasheados con bcrypt
- [x] Expiración de códigos (10 minutos)
- [x] Prevención de reutilización de códigos
- [x] Transacción atómica en reset
- [x] Redirección automática a /login tras éxito

- [x] Manejo de errores completo
- [x] Logs para debugging y auditoría
- [x] Fallback si Resend no está configurado
- [x] UI/UX profesional y responsive
- [x] Opción de reenviar código
- [x] Opción de volver al paso anterior

🚀 Próximos Pasos Recomendados

1. **Configurar Resend en Producción**
   - Verificar dominio en Resend
   - Actualizar el `from` en emailVerification.ts
   - Testear envío de emails en producción

2. **Mejorar Seguridad**
   - Implementar rate limiting
   - Limitar intentos de código incorrecto
   - No revelar si el email existe (en producción)
   - Agregar CAPTCHA si hay abuso

3. **Monitoreo**
   - Configurar alertas para errores de envío
   - Trackear métricas de recuperación
   - Logs centralizados

4. **Testing Automatizado**
   - Tests unitarios para APIs
   - Tests de integración del flujo completo
   - Tests E2E con Playwright/Cypress

5. **Mejoras de UX**
   - Auto-copiar código desde email
   - Notificación push cuando llegue el email
   - Mostrar último email usado

📞 Soporte

Si tienes problemas con el flujo de recuperación:

1. Verifica que `RESEND_API_KEY` esté configurado
2. Revisa los logs de la consola
3. Verifica que el email del usuario exista en la BD
4. Confirma que el código no haya expirado
5. Verifica que el código no haya sido usado

Implementado por: DeepAgent - Abacus.AI
Fecha: Noviembre 2025
Versión: 1.0.0