# 🔐 Sistema de Restablecimiento de Contraseña

## 📋 Descripción General

Se ha implementado un sistema completo y seguro de restablecimiento de contraseña que permite a los usuarios recuperar el acceso a su cuenta cuando olvidan su contraseña.

## ✨ Características Principales

### 1. Solicitud de Restablecimiento

- Formulario intuitivo donde el usuario ingresa su email
- Validación de email
- Generación de token seguro único
- Envío de email con enlace de restablecimiento
- Mensajes informativos y amigables

### 2. Seguridad Robusta

- **Tokens hasheados**: Los tokens se almacenan hasheados con SHA-256 en la base de datos
- **Expiración temporal**: Los tokens expiran en 1 hora por seguridad
- **Tokens de un solo uso**: Cada token solo puede usarse una vez
- **Mensajes genéricos**: Por seguridad, siempre se muestra el mismo mensaje sin revelar si el email existe
- **Validación de contraseña**: Requiere mínimo 6 caracteres

### 3. Experiencia de Usuario

- Interfaz moderna y consistente con el resto de la aplicación
- Mensajes claros en cada paso del proceso
- Feedback visual inmediato
- Manejo de errores elegante
- Redirección automática tras éxito

## 🏗️ Arquitectura Técnica

### Archivos Creados/Modificados

#### 1. Servicio de Email (`lib/email.ts`)

```
// Funciones principales:
- sendEmail(): Envía emails (simulado en desarrollo)
- generatePasswordResetEmail(): Genera HTML del email
```

**Características:**
- En desarrollo: Muestra emails en consola
- En producción: Listo para integrar con SendGrid/Resend

- Emails HTML responsivos y profesionales
- Branding consistente con la aplicación

### 2. API - Solicitar Restablecimiento ( `app/api/auth/forgot-password/route.ts` )

```
POST /api/auth/forgot-password
Body: { email: string }
```

**Flujo:**
1. Recibe email del usuario
2. Verifica si el usuario existe (sin revelarlo en la respuesta)
3. Genera token único de 32 bytes
4. Hashea el token con SHA-256
5. Elimina tokens previos del mismo usuario
6. Crea nuevo token en la base de datos con expiración de 1 hora
7. Genera enlace de restablecimiento
8. Envía email al usuario

**Seguridad:**
- Siempre devuelve el mismo mensaje, exista o no el email
- Tokens criptográficamente seguros
- Limpieza automática de tokens antiguos

### 3. API - Restablecer Contraseña ( `app/api/auth/reset-password/route.ts` )

```
POST /api/auth/reset-password
Body: { token: string, email: string, password: string }
```

**Flujo:**
1. Recibe token, email y nueva contraseña
2. Valida que todos los campos estén presentes
3. Valida longitud de contraseña (mínimo 6 caracteres)
4. Hashea el token recibido para comparar
5. Busca el token en la base de datos
6. Verifica que no haya expirado
7. Verifica que el usuario existe
8. Hashea la nueva contraseña con bcrypt (12 rondas)
9. Actualiza la contraseña del usuario
10. Elimina el token usado

**Validaciones:**
- Token válido y no expirado
- Usuario existe
- Contraseña cumple requisitos mínimos
- Token de un solo uso

### 4. Página - Solicitar Restablecimiento ( `app/auth/forgot-password/page.tsx` )

**Características:**
- Formulario simple con campo de email
- Estado de “enviado” con mensaje de confirmación
- Consejos útiles (revisar spam, etc.)

- Enlaces a login y registro
- Opción para enviar a otro email
- Diseño responsive y accesible

**Estados:**
- **Inicial**: Formulario para ingresar email
- **Cargando**: Indicador mientras se procesa
- **Enviado**: Confirmación con instrucciones

### 5. Página - Nueva Contraseña (`app/auth/reset-password/page.tsx`)

**Características:**
- Lee token y email de parámetros URL
- Validación de enlace antes de mostrar formulario
- Campos para nueva contraseña y confirmación
- Validación en tiempo real
- Consejos de seguridad
- Redirección automática al login tras éxito
- Manejo de enlaces inválidos o expirados

**Estados:**
- **Enlace inválido**: Mensaje de error y opción para solicitar nuevo enlace
- **Formulario**: Campos para nueva contraseña
- **Cargando**: Indicador mientras se actualiza
- **Éxito**: Confirmación y redirección al login

### 6. Página de Login Actualizada (`app/auth/login/page.tsx`)

**Cambios:**
- Agregado enlace “¿Olvidaste tu contraseña?”
- Posicionado estratégicamente junto al campo de contraseña
- Estilo consistente con el diseño existente

## 🔄 Flujo Completo del Usuario

### Paso 1: Usuario olvida su contraseña

```
Usuario → Login → Click "¿Olvidaste tu contraseña?"
```

### Paso 2: Solicitud de restablecimiento

```
Usuario → Ingresa email → Submit
        ↓
API valida y genera token
        ↓
Email enviado con enlace único
        ↓
Mensaje de confirmación
```

### Paso 3: Email recibido

```
Usuario recibe email con:
- Saludo personalizado
- Explicación clara
- Botón/enlace para restablecer
- Link alternativo si el botón no funciona
- Advertencia de expiración (1 hora)
- Instrucciones si no solicitó el cambio
```

### Paso 4: Click en enlace

```
Usuario click en enlace → Página de nueva contraseña
        ↓
Validación de token
        ↓
Si es válido: Formulario
Si no es válido: Mensaje de error + Opción de solicitar nuevo
```

### Paso 5: Establecer nueva contraseña

```
Usuario → Ingresa nueva contraseña (2 veces)
        ↓
Validación en cliente
        ↓
Submit → API valida token y actualiza contraseña
        ↓
Confirmación de éxito
        ↓
Redirección automática al login (2 segundos)
```

### Paso 6: Login con nueva contraseña

```
Usuario → Login con nueva contraseña → Dashboard
```

## 🔐 Consideraciones de Seguridad

### Implementadas

1. **Tokens Seguros**
   - Generados con crypto.randomBytes (32 bytes)
   - Hasheados con SHA-256 antes de almacenar
   - Únicos e impredecibles

2. **Expiración Temporal**
   - Tokens válidos por 1 hora
   - Limpieza automática de tokens expirados

3. **Un Solo Uso**
   - Token eliminado después de usarse
   - No se pueden reutilizar

4. **Privacidad**
   - No se revela si un email existe en el sistema
   - Mensajes genéricos para evitar enumeración de usuarios

5. **Contraseñas Seguras**
   - Hasheadas con bcrypt (12 rondas)
   - Validación de longitud mínima
   - Confirmación requerida

6. **HTTPS Requerido**
   - En producción, todos los enlaces deben usar HTTPS
   - Protección contra interceptación

### Recomendaciones Adicionales

1. **Rate Limiting**
   - Implementar límite de solicitudes por IP
   - Prevenir ataques de fuerza bruta
   - Ejemplo: Máximo 3 solicitudes por hora

2. **Logging y Monitoreo**
   - Registrar intentos de restablecimiento
   - Alertas de actividad sospechosa
   - Auditoría de cambios de contraseña

3. **Notificaciones**
   - Email de confirmación tras cambio exitoso
   - Alertas de actividad no autorizada

4. **Validación de Contraseña Mejorada**
   - Requerir mayúsculas y minúsculas
   - Requerir números y símbolos
   - Longitud mínima de 8-12 caracteres
   - Verificar contra lista de contraseñas comunes

## 📧 Servicio de Email

### Estado Actual (Desarrollo)

Los emails se muestran en la consola del servidor:

```
═══════════════════════════════════════════════════
📧 EMAIL SIMULADO (DESARROLLO)
═══════════════════════════════════════════════════
Para: usuario@ejemplo.com
Asunto: Restablece tu contraseña - SpeaklyPlan
Contenido HTML: [HTML del email]
═══════════════════════════════════════════════════
```

## Para Producción

### Opción 1: SendGrid

```
// En lib/email.ts, descomentar y configurar:
const response = await fetch('https://api.sendgrid.com/v3/mail/send', {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${process.env.SENDGRID_API_KEY}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    personalizations: [{ to: [{ email: to }] }],
    from: { email: process.env.FROM_EMAIL },
    subject,
    content: [{ type: 'text/html', value: html }]
  })
})
```

**Variables de entorno requeridas:**

```
SENDGRID_API_KEY=tu_api_key_de_sendgrid
FROM_EMAIL=noreply@speaklyplan.com
```

### Opción 2: Resend

```
// Alternativa con Resend
import { Resend } from 'resend'

const resend = new Resend(process.env.RESEND_API_KEY)

await resend.emails.send({
  from: process.env.FROM_EMAIL,
  to,
  subject,
  html
})
```

**Variables de entorno requeridas:**

```
RESEND_API_KEY=tu_api_key_de_resend
FROM_EMAIL=noreply@speaklyplan.com
```

### Opción 3: Nodemailer (SMTP)

```
import nodemailer from 'nodemailer'

const transporter = nodemailer.createTransporter({
  host: process.env.SMTP_HOST,
  port: parseInt(process.env.SMTP_PORT || '587'),
  secure: process.env.SMTP_SECURE === 'true',
  auth: {
    user: process.env.SMTP_USER,
    pass: process.env.SMTP_PASS
  }
})

await transporter.sendMail({
  from: process.env.FROM_EMAIL,
  to,
  subject,
  html
})
```

## 🧪 Testing

### Testing Manual

#### 1. Test de Solicitud

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

**Respuesta esperada:**

```
{
  "message": "Si el email existe en nuestro sistema, recibirás un enlace de
restablecimiento."
}
```

#### 2. Test de Restablecimiento

```
curl -X POST http://localhost:3000/api/auth/reset-password \
  -H "Content-Type: application/json" \
  -d '{
    "token":"TOKEN_DEL_EMAIL",
    "email":"test@ejemplo.com",
    "password":"nuevaContraseña123"
  }'
```

**Respuesta esperada (éxito):**

```
{
  "message": "Contraseña actualizada exitosamente"
}
```

**Respuesta esperada (token inválido):**

```
{
  "error": "Token inválido o expirado"
}
```

### Testing en la UI

1. **Solicitar Restablecimiento**
   - Ir a `/auth/login`
   - Click en "¿Olvidaste tu contraseña?"
   - Ingresar email válido
   - Verificar mensaje de confirmación

2. **Verificar Email**
   - En desarrollo: Revisar consola del servidor
   - Copiar el enlace generado

3. **Restablecer Contraseña**
   - Pegar enlace en el navegador
   - Ingresar nueva contraseña (2 veces)
   - Verificar que coincidan
   - Click en "Actualizar contraseña"
   - Verificar mensaje de éxito
   - Verificar redirección al login

4. **Verificar Login**
   - Intentar login con contraseña antigua (debe fallar)
   - Intentar login con nueva contraseña (debe funcionar)

### Casos de Borde a Probar

1. **Email no existe**: Debe mostrar mensaje genérico
2. **Token expirado**: Debe mostrar error y opción de solicitar nuevo
3. **Token ya usado**: Debe mostrar error
4. **Contraseña muy corta**: Debe mostrar error de validación
5. **Contraseñas no coinciden**: Debe mostrar error
6. **Enlace malformado**: Debe mostrar error de enlace inválido
7. **Múltiples solicitudes**: Debe invalidar tokens anteriores

## 📊 Base de Datos

### Modelo Utilizado

```
model VerificationToken {
  identifier String
  token      String   @unique
  expires    DateTime

  @@unique([identifier, token])
}
```

**Uso:**
- `identifier`: Email del usuario
- `token`: Token hasheado (SHA-256)
- `expires`: Fecha/hora de expiración

### Consultas

#### Crear token

```
await prisma.verificationToken.create({
  data: {
    identifier: email,
    token: hashedToken,
    expires: new Date(Date.now() + 60 * 60 * 1000)
  }
})
```

#### Buscar token

```
const tokens = await prisma.verificationToken.findMany({
  where: { identifier: email }
})
```

#### Eliminar token

```
await prisma.verificationToken.deleteMany({
  where: {
    identifier: email,
    token: hashedToken
  }
})
```

## 🎨 UI/UX

### Páginas Creadas

1. `/auth/forgot-password`
   - Diseño limpio y moderno
   - Consistente con login/registro
   - Estados claros (inicial, cargando, enviado)
   - Iconos descriptivos
   - Mensajes informativos

2. `/auth/reset-password`
   - Validación de enlace automática
   - Formulario seguro
   - Consejos de seguridad
   - Confirmación visual
   - Redirección automática

### Elementos de UI

- **Cards**: Contenedores con sombra para formularios
- **Inputs**: Con iconos y validación

- **Buttons**: Con estados de carga
- **Alertas**: Info, success, warning, error
- **Icons**: Lucide React para consistencia
- **Gradientes**: Fondo consistente con la app

# 🚀 Próximos Pasos Recomendados

## Corto Plazo

1. **Configurar servicio de email real**
   - Elegir proveedor (SendGrid, Resend, etc.)
   - Obtener API key
   - Configurar dominio y DNS
   - Configurar variables de entorno

2. **Testing exhaustivo**
   - Probar todos los flujos
   - Verificar casos de borde
   - Testing en diferentes navegadores
   - Testing mobile

## Mediano Plazo

1. **Rate Limiting**
   - Implementar con Redis o memoria
   - Límite por IP y por email
   - Mensajes claros al alcanzar límite

2. **Notificaciones adicionales**
   - Email de confirmación tras cambio
   - Alertas de seguridad

3. **Analytics**
   - Tracking de solicitudes
   - Tasas de éxito/fallo
   - Tiempo promedio del proceso

## Largo Plazo

1. **Mejoras de seguridad**
   - 2FA opcional
   - Preguntas de seguridad
   - Verificación de identidad adicional

2. **Experiencia mejorada**
   - Personalización de emails
   - Múltiples idiomas
   - Mejoras de accesibilidad

# 📖 Documentación para Usuarios

## FAQ

**P: ¿Cuánto tiempo es válido el enlace de restablecimiento?**
R: El enlace es válido por 1 hora desde que lo solicitas por razones de seguridad.

**P: ¿Qué hago si el enlace expiró?**
R: Simplemente solicita un nuevo enlace desde la página de restablecimiento. El enlace anterior quedará invalidado automáticamente.

**P: ¿Puedo usar el mismo enlace varias veces?**
R: No, cada enlace solo puede usarse una vez. Después de restablecer tu contraseña, el enlace deja de funcionar.

**P: ¿Qué pasa si no recibo el email?**
R: Verifica tu carpeta de spam. Si aún no lo recibes, intenta solicitar un nuevo enlace o contacta a soporte.

**P: ¿Alguien más podría restablecer mi contraseña?**
R: No, solo tú puedes hacerlo ya que el enlace se envía únicamente a tu email registrado. Sin acceso a tu email, nadie puede restablecer tu contraseña.

# 🎯 Conclusión

El sistema de restablecimiento de contraseña está completamente implementado y funcional, con:

✅ Seguridad robusta con tokens hasheados y expiración temporal
✅ UX intuitiva con mensajes claros y feedback visual
✅ Código limpio y bien documentado
✅ Listo para desarrollo y testing
✅ Preparado para producción (solo falta configurar servicio de email)

El sistema sigue las mejores prácticas de seguridad y ofrece una experiencia de usuario fluida y profesional.