Recomendaciones de Seguridad - Inmova Platform

ACCION URGENTE REQUERIDA: Rotacion de Credenciales

Estado Actual

Se han identificado credenciales sensibles expuestas en el archivo `.env` que requieren rotacion inmediata:

- `DATABASE_URL` - Cadena de conexion a PostgreSQL
- `NEXTAUTH_SECRET` - Secreto para firmar tokens de sesion
- `ENCRYPTION_KEY` - Clave para cifrado de datos sensibles
- `CRON_SECRET` - Token para proteger endpoints de tareas programadas

ADVERTENCIA: Riesgos Identificados

1. **Acceso no autorizado a la base de datos**: Si DATABASE_URL esta comprometida, un atacante podria:
   - Leer datos sensibles de usuarios, contratos y pagos
   - Modificar o eliminar registros
   - Realizar ataques de escalada de privilegios

2. **Compromiso de sesiones de usuario**: Si NEXTAUTH_SECRET esta expuesta, un atacante podria:
   - Falsificar tokens de sesion
   - Acceder a cuentas de usuarios sin credenciales
   - Realizar acciones en nombre de otros usuarios

3. **Descifrado de datos sensibles**: Si ENCRYPTION_KEY esta comprometida:
   - Datos cifrados podrian ser descifrados
   - Informacion personal y financiera podria quedar expuesta

---

Plan de Rotacion de Credenciales

Fase 1: Rotacion de DATABASE_URL (Prioridad Alta - HOY)

Opcion A: Cambiar Contrasena de PostgreSQL

```
# 1. Conectarse a PostgreSQL como superusuario
psql -U postgres

# 2. Cambiar la contrasena del usuario
ALTER USER your_db_user WITH PASSWORD 'nueva_contrasena_segura_aqui';

# 3. Actualizar DATABASE_URL en produccion
# NO almacenar en .env del repositorio
# Usar variables de entorno del servidor o secrets manager
```

Requisitos de contrasena segura:
- Minimo 20 caracteres
- Mezcla de mayusculas, minusculas, numeros y simbolos
- Generada aleatoriamente (usar `openssl rand -base64 32`)

Opcion B: Crear Nuevo Usuario de Base de Datos

```
# 1. Crear nuevo usuario con contrasena segura
CREATE USER nuevo_usuario WITH PASSWORD 'contrasena_aleatoria_segura';

# 2. Otorgar permisos necesarios
GRANT ALL PRIVILEGES ON DATABASE tu_base_datos TO nuevo_usuario;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO nuevo_usuario;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO nuevo_usuario;

# 3. Actualizar DATABASE_URL con el nuevo usuario
# 4. Probar la conexion
# 5. Revocar acceso del usuario anterior
REVOKE ALL PRIVILEGES ON DATABASE tu_base_datos FROM usuario_anterior;
DROP USER usuario_anterior;
```

Fase 2: Rotacion de NEXTAUTH_SECRET (Prioridad Alta - HOY)

```
# Generar nuevo secreto seguro (32+ bytes)
openssl rand -base64 48

# El resultado sera algo como:
# abc123XYZ789def456GHI012jkl345MNO678pqr901STU234vwx567YZA890
```

Impacto de la rotacion:
- ADVERTENCIA: Todas las sesiones activas seran invalidadas
- Todos los usuarios deberan iniciar sesion nuevamente
- Planear esta rotacion fuera de horas pico

Procedimiento:
1. Comunicar a los usuarios sobre el mantenimiento programado
2. Actualizar NEXTAUTH_SECRET en produccion
3. Reiniciar la aplicacion
4. Verificar que el inicio de sesion funciona correctamente

Fase 3: Rotacion de ENCRYPTION_KEY (Prioridad Alta - HOY)

```
# Generar nueva clave de cifrado
openssl rand -hex 32
```

ADVERTENCIA CRITICA:
La rotacion de ENCRYPTION_KEY es compleja porque:
- Los datos existentes estan cifrados con la clave anterior
- Se requiere proceso de re-cifrado de datos

Procedimiento de Rotacion:

```
// Script de migracion: scripts/rotate-encryption-key.ts
import { prisma } from '../lib/db';
import * as crypto from 'crypto';

const OLD_KEY = process.env.OLD_ENCRYPTION_KEY!;
const NEW_KEY = process.env.ENCRYPTION_KEY!;

function decrypt(text: string, key: string): string {
  const decipher = crypto.createDecipher('aes-256-cbc', key);
  let decrypted = decipher.update(text, 'hex', 'utf8');
  decrypted += decipher.final('utf8');
  return decrypted;
}

function encrypt(text: string, key: string): string {
  const cipher = crypto.createCipher('aes-256-cbc', key);
  let encrypted = cipher.update(text, 'utf8', 'hex');
  encrypted += cipher.final('hex');
  return encrypted;
}

async function rotateEncryptionKey() {
  console.log('[SEGURIDAD] Iniciando rotacion de clave de cifrado...');

  // Identificar todos los campos cifrados en tu esquema
  // Ejemplo: DNI, IBAN, datos sensibles

  const tenants = await prisma.tenant.findMany();

  for (const tenant of tenants) {
    if (tenant.dni) {
      try {
        const decrypted = decrypt(tenant.dni, OLD_KEY);
        const reencrypted = encrypt(decrypted, NEW_KEY);

        await prisma.tenant.update({
          where: { id: tenant.id },
          data: { dni: reencrypted }
        });

        console.log(`[OK] Re-cifrado DNI para inquilino ${tenant.id}`);
      } catch (error) {
        console.error(`[ERROR] Procesando inquilino ${tenant.id}:`, error);
      }
    }
  }

  // Repetir para otros modelos/campos que usen cifrado

  console.log('[OK] Rotacion de clave completada');
}

rotateEncryptionKey()
  .catch(console.error)
  .finally(() => prisma.$disconnect());
```

Pasos:

1. Hacer backup completo de la base de datos
2. Generar nueva clave

3. Almacenar temporalmente la clave antigua como OLD_ENCRYPTION_KEY
4. Ejecutar script de re-cifrado
5. Verificar que todos los datos pueden ser descifrados
6. Eliminar OLD_ENCRYPTION_KEY

Fase 4: Rotacion de CRON_SECRET (Prioridad Media)

```
# Generar nuevo token
openssl rand -hex 32
```

Procedimiento:
1. Actualizar CRON_SECRET en el servidor
2. Actualizar configuraciones de cron jobs que llamen a endpoints protegidos
3. Reiniciar servicios afectados

Implementacion de Gestion Segura de Secretos

Opcion 1: Variables de Entorno del Servidor (Minimo Requerido)

```
# En el servidor de produccion (NO en .env del repositorio)
export DATABASE_URL="postgresql://..."
export NEXTAUTH_SECRET="..."
export ENCRYPTION_KEY="..."
```

Opcion 2: AWS Secrets Manager (Recomendado)

Configuracion Inicial

```
# Instalar AWS CLI y configurar credenciales
apt install awscli
aws configure

# Crear secreto
aws secretsmanager create-secret \
    --name inmova/production/database \
    --secret-string '{"DATABASE_URL":"postgresql://..."}'

aws secretsmanager create-secret \
    --name inmova/production/auth \
    --secret-string '{"NEXTAUTH_SECRET":"...","ENCRYPTION_KEY":"..."}'
```

Integracion en la Aplicacion

```
// lib/secrets.ts
import { SecretsManagerClient, GetSecretValueCommand } from '@aws-sdk/client-secrets-
manager';

const client = new SecretsManagerClient({ region: 'us-east-1' });

const secretCache = new Map<string, { value: any; expires: number }>();
const CACHE_TTL = 5 * 60 * 1000; // 5 minutos

export async function getSecret(secretName: string): Promise<any> {
  const cached = secretCache.get(secretName);
  if (cached && cached.expires > Date.now()) {
    return cached.value;
  }

  const command = new GetSecretValueCommand({ SecretId: secretName });
  const response = await client.send(command);

  const value = JSON.parse(response.SecretString!);

  secretCache.set(secretName, {
    value,
    expires: Date.now() + CACHE_TTL
  });

  return value;
}

// Uso en la aplicacion
export async function getDatabaseUrl(): Promise<string> {
  if (process.env.NODE_ENV === 'development') {
    return process.env.DATABASE_URL!;
  }

  const secrets = await getSecret('inmova/production/database');
  return secrets.DATABASE_URL;
}
```

Opcion 3: HashiCorp Vault (Para Empresas)

Vault proporciona:
- Rotacion automatica de secretos
- Auditoria completa de accesos
- Cifrado en reposo y en transito
- Control de acceso granular

Ver documentacion en: https://www.vaultproject.io/

Mejoras de Seguridad Adicionales Implementadas

1. Patron Singleton de PrismaClient [COMPLETADO]

Problema Resuelto:
- Multiples instancias de PrismaClient causaban agotamiento de conexiones
- Potencial vector de ataque de Denegacion de Servicio (DoS)

Solucion Implementada:
- Todos los archivos ahora importan `prisma` desde `lib/db.ts`
- Singleton garantiza una sola instancia global
- Previene fugas de conexiones

Archivos Corregidos:
- `lib/reminder-service.ts`
- `lib/services/sales-team-service.ts`
- `app/api/partners/*/route.ts` (6 archivos)

2. Validacion de Inputs con Zod [COMPLETADO]

Problema Resuelto:
- Datos sin validar permitian:
- Cross-Site Scripting (XSS) almacenado
- Corrupcion de datos
- Inyecciones indirectas

Solucion Implementada:
- Esquemas de validacion Zod centralizados en `lib/validations/index.ts`
- Validacion exhaustiva de todos los inputs en APIs criticas
- Sanitizacion automatica (trim, toLowerCase, toUpperCase)
- Validacion de formatos (email, DNI, telefono, fechas)
- Limites de longitud de cadenas
- Rangos numericos controlados

APIs Protegidas:
- [OK] Buildings (Edificios)
- [OK] Units (Unidades)
- [OK] Tenants (Inquilinos)
- [OK] Contracts (Contratos)
- [OK] Payments (Pagos)
- [OK] Maintenance (Mantenimiento - esquema creado)
- [OK] Providers (Proveedores - esquema creado)

Ejemplo de Validacion:

```
// Antes (INSEGURO)
const { nombre, direccion } = await req.json();
await prisma.building.create({ data: { nombre, direccion } });

// Despues (SEGURO)
const body = await req.json();
const validationResult = buildingCreateSchema.safeParse(body);

if (!validationResult.success) {
  return NextResponse.json(
    { error: 'Datos invalidos', details: validationResult.error.errors },
    { status: 400 }
  );
}

await prisma.building.create({ data: validationResult.data });
```

Monitorizacion y Auditoria

Logs de Seguridad Mejorados

Los siguientes eventos ahora se registran:

```
// Validaciones fallidas
logger.warn('Validation error creating building:', { errors });

// Operaciones exitosas
logger.info('Building created successfully:', { buildingId, companyId });

// Errores de sistema
logError(error, 'Error creating building');
```

Alertas Recomendadas

Configurar alertas para:

1. **Multiples validaciones fallidas desde la misma IP**
   - Umbral: 10+ intentos en 5 minutos
   - Accion: Bloqueo temporal de IP

2. **Intentos de inyeccion SQL**
   - Detectar patrones: `'; DROP TABLE`, `UNION SELECT`, etc.
   - Accion: Bloqueo permanente + notificacion

3. **Accesos no autorizados repetidos**
   - Umbral: 5+ errores 401 en 1 minuto
   - Accion: CAPTCHA o bloqueo temporal

Checklist de Accion Inmediata

Hoy (Proximas 2 horas)

- [ ] Hacer backup completo de la base de datos
- [ ] Rotar DATABASE_URL (Fase 1)
- [ ] Rotar NEXTAUTH_SECRET (Fase 2)
- [ ] Notificar a usuarios sobre cierre de sesiones
- [ ] Actualizar variables de entorno en produccion
- [ ] Reiniciar aplicacion
- [ ] Verificar que todo funciona correctamente

Esta Semana

- [ ] Planificar rotacion de ENCRYPTION_KEY (Fase 3)
- [ ] Desarrollar y probar script de re-cifrado
- [ ] Ejecutar rotacion de ENCRYPTION_KEY en horario de bajo trafico
- [ ] Rotar CRON_SECRET (Fase 4)
- [ ] Eliminar .env del repositorio si esta en control de versiones
- [ ] Agregar .env a .gitignore
- [ ] Verificar que no hay secretos en el historial de Git

Este Mes

- [ ] Implementar AWS Secrets Manager o alternativa
- [ ] Configurar rotacion automatica de credenciales
- [ ] Implementar sistema de alertas de seguridad
- [ ] Realizar auditoria de seguridad completa
- [ ] Capacitar al equipo en buenas practicas de seguridad

Contactos de Emergencia

En caso de incidente de seguridad:

1. **Aislar el sistema comprometido**
2. **Contactar al equipo de seguridad**
3. **Documentar el incidente**
4. **Seguir el plan de respuesta a incidentes**

Referencias

- OWASP Top 10 (https://owasp.org/www-project-top-ten/)
- AWS Secrets Manager Best Practices (https://docs.aws.amazon.com/secretsmanager/latest/userguide/best-practices.html)
- NIST Cybersecurity Framework (https://www.nist.gov/cyberframework)
- CIS Controls (https://www.cisecurity.org/controls)

Ultima actualizacion: 7 de diciembre de 2025
Responsable: Equipo de Desarrollo
Proxima revision: 7 de enero de 2026