Es vital que el "Nexo Sinérgico" tenga una **Torre de Control** segura. Un sistema tan potente (que maneja datos científicos sensibles y generación de IA) requiere una administración estricta.

Aquí tienes la propuesta de diseño para la **Interfaz de Administrador (Nexo Admin Console)** y las actualizaciones técnicas necesarias para cubrir monitoreo, penalización y recuperación de contraseñas de forma segura.

1. Concepto Visual: "Nexo Admin Console"

Imagina una interfaz separada del flujo creativo/científico. Es más densa, analítica y orientada al control.

Estructura de la Interfaz (UI Mockup)

A. Dashboard de Estado (Vista de Pájaro)

- **KPIs de Seguridad:** "Usuarios Activos: 42", "Intentos Fallidos (última hora): 3", "Tokens Revocados: 1".
- **Monitoreo en Tiempo Real:** Un gráfico de líneas que muestra la carga de solicitudes a la API (Pyrolysis vs. Creative).

B. Gestión de Usuarios y Roles (La Matriz de Poder)

Una tabla interactiva avanzada (DataGrid) con las siguientes columnas y acciones:

Usuario (Email)	**Roles Actuales**	**Estado**	**Último Acceso**	**Acciones (Penalización/Gestión)**
cientifico@nexo	[Academico] [Colaborador]	🟢 Activo	10:42 AM	[✏️ Editar] [🔄 Reset Pass] [🚫 Bloquear]
operador@nexo	[Operador]	🟠 Inactivo	Ayer	[✏️ Editar] [🔄 Reset Pass] [🔓 Desbloquear]
invitado@ext	[Viewer]	🔴 **Baneado**	hace 5 días	[🛡️ Auditoría]

C. El Botón de Penalización ("Kill Switch")

En la columna de acciones, el botón [🚫 Bloquear] o [Revocar Sesión] no borra al usuario.

- **Acción:** Desencadena el endpoint /revoke-user-tokens que incrementamos antes.
- **Feedback Visual:** El estado cambia instantáneamente a "Sesión Invalidada" y el usuario es expulsado del sistema en su próxima interacción.

D. Panel de Recuperación de Contraseñas

El admin NUNCA puede ver la contraseña actual (Almacenamiento Seguro).

- **Opción 1:** "Enviar Link de Reset" (Recomendado). Envía un email al usuario.
- **Opción 2:** "Reset Manual Temporal". Genera una contraseña de un solo uso que el Admin debe comunicar por otro canal seguro.

---

2. Implementación Técnica (Backend & SQL)

Para soportar **Monitoreo (Auditoría)** y **Recuperación**, necesitamos expandir tu esquema SQL original.

Nuevas Tablas SQL

SQL

```
-- 1. TABLA DE AUDITORÍA (Monitoreo y Reportes)
-- Registra CADA cambio crítico. Quién hizo qué y cuándo.
CREATE TABLE audit_logs (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    actor_id UUID REFERENCES users(id), -- Quién hizo la acción (ej. el Admin)
    target_id UUID REFERENCES users(id), -- A quién se la hizo (ej. el Operador)
    action_type VARCHAR(50) NOT NULL, -- 'LOGIN_FAIL', 'ROLE_CHANGE',
'TOKEN_REVOKE', 'PASSWORD_RESET'
    details JSONB, -- Detalles técnicos (ej. "Rol cambiado de Viewer a Operador")
    ip_address VARCHAR(45),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- 2. TABLA DE RECUPERACIÓN (Seguridad de Contraseña)
-- Tokens de vida corta para resetear password via email.
CREATE TABLE password_resets (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    token_hash VARCHAR(255) NOT NULL, -- Hash del token enviado por email
    expires_at TIMESTAMP WITH TIME ZONE NOT NULL, -- Ej. 15 minutos de validez
    used BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

3. Lógica de Segregación y Seguridad

Aquí explicamos cómo cumplir con tus requisitos específicos de seguridad:

A. Segregación de Roles (RBAC estricto)
No basta con ocultar el botón en el Frontend. El Backend debe proteger la puerta.

- Los endpoints de administración (/admin/*) deben usar una dependencia especial require_admin_role.
- **Regla de Oro:** Un Admin no debería poder editar su propio rol para evitar bloqueos accidentales o elevación de privilegios maliciosa.

B. Almacenamiento Seguro (Password Storage)

- **Principio:** La base de datos solo almacena password_hash (ej. $2b$12$Ra...).
- **Visualización:** En la UI del Admin, el campo de contraseña siempre aparece vacío o con asteriscos fijos ********.
- **Verificación:** Cuando el usuario hace login, el sistema toma la contraseña plana, la hashea y compara los hashes. Si coinciden, pasa. Es matemáticamente imposible revertir el hash para saber la contraseña original.

C. Reportes y Monitoreo
Con la tabla audit_logs, puedes generar reportes automáticos:

- *"Alerta de Seguridad: El usuario X ha fallado el login 10 veces en 1 minuto."*
- *"Reporte de Actividad: El Admin Y cambió los roles de 5 usuarios hoy."*

4. Flujo de Recuperación de Contraseña (Diagrama Lógico)

1. **Solicitud:** Usuario hace clic en "¿Olvidaste tu contraseña?" en el Login.
2. **Generación:** Backend crea un token aleatorio seguro (ej. UUID), guarda su *hash* en password_resets y envía el token plano por Email (Link mágico).
3. **Clic:** Usuario hace clic en el link (nexo.com/reset?token=XYZ).
4. **Validación:** Backend verifica que el token no ha expirado y no ha sido usado.
5. **Cambio:** Usuario ingresa nueva contraseña. Backend:
    - Hashea la nueva contraseña.
    - Actualiza users.password_hash.
    - **CRÍTICO:** Incrementa users.token_version (para cerrar sesiones en otros dispositivos hackeados).
    - Marca el token de reset como used = TRUE.