Prompt Detallado: Implementación de Roles Mecánico y Supervisor

> **Uso**: Copia y pega este prompt en Lovable, Cursor, GPT-4, Claude o cualquier agente de codegen para que genere el código necesario. Ajusta nombres de tablas/campos si tu esquema difiere.

🎯 Filosofía de Diseño UX

Principio Central

El mecánico es un **reportero de campo**, no un editor del sistema. Su trabajo es:

1. Ver qué equipos tienen mantenimientos pendientes
2. Realizar el trabajo físicamente
3. **Reportar** lo que hizo (fotos, partes, notas)
4. Esperar aprobación del admin para que se integre al sistema oficial

El admin es el **guardián de la verdad**: solo él decide qué entra al sistema oficial de mantenimiento.

Metáfora UX

Piensa en el mecánico como alguien que llena un "parte de trabajo" en papel y lo entrega al jefe. El jefe revisa, puede pedir correcciones, y cuando está conforme, lo archiva oficialmente. El mecánico nunca toca el archivo oficial.

Contexto del Proyecto

Estoy trabajando en una aplicación de gestión de mantenimiento de flotas llamada **ALITO Mantenimiento**. El stack es:

- **Frontend**: React 18 + Vite + TypeScript + Tailwind CSS + shadcn/ui
- **Backend/DB**: Supabase (PostgreSQL + Auth + Storage + Realtime)
- **Mobile**: Capacitor (Android) — misma base de código React con layouts móviles adaptativos
- **Rutas móviles**: `/mobile/*` con componentes en `src/pages/mobile/`
- **Autenticación**: Supabase Auth con roles personalizados en tabla `user_roles`
- **Permisos actuales**: Hook `useUserRoles` en `src/hooks/useUserRoles.ts` y utilidades en `src/lib/permissions.ts`

La app ya tiene:

- Dashboard con métricas
- Módulo de Equipos (CRUD)
- Módulo de Control de Mantenimiento (registro, historial, planes)
- Módulo de Inventario
- Sistema de notificaciones básico
- Vistas móviles adaptadas

Objetivo

Necesito implementar **dos nuevos roles** con sus respectivas interfaces y flujos:

🔧 ROL: MECÁNICO (mechanic)

Filosofía del Rol

El mecánico es un **usuario de campo con permisos muy limitados**. Su única función es:

1. **VER** qué trabajos tiene pendientes
2. **REPORTAR** lo que hizo (como un parte de trabajo)
3. **ESPERAR** que el admin apruebe su reporte

El mecánico NUNCA:

- ❌ Edita datos del sistema
- ❌ Modifica equipos
- ❌ Cambia estados de mantenimiento directamente
- ❌ Accede al inventario para modificarlo
- ❌ Ve información sensible (costos, proveedores, etc.)
- ❌ Elimina nada

Qué VE el Mecánico (Solo Lectura)

Módulo	Qué puede ver	Qué NO puede ver
Historial	Solo sus propios reportes enviados	Historial completo del sistema
Mantenimiento	Lista de equipos con mantenimientos pendientes/vencidos	Control de mantenimiento, planes, configuración
Inventario	Lista de partes disponibles (nombre, stock)	Precios, proveedores, costos
Equipos	Ficha básica del equipo que está trabajando	Edición, eliminación, datos financieros

Qué PUEDE HACER el Mecánico

1. **Crear un Reporte de Trabajo** (Submission)
    - Seleccionar el equipo en el que trabajó
    - Indicar fecha y hora del trabajo
    - Registrar horas/km actuales del equipo
    - Describir el trabajo realizado
    - Listar partes/repuestos utilizados
    - Subir fotos del trabajo (antes/después)
    - Agregar observaciones
2. **Ver Estado de sus Reportes**
    - Ver si está pendiente, aprobado, o rechazado
    - Ver feedback del admin si fue rechazado
    - Ver cuándo fue integrado al sistema
3. **Recibir Notificaciones**
    - Cuando su reporte es aprobado
    - Cuando su reporte es rechazado (con motivo)

📱 UX DETALLADO: INTERFAZ DEL MECÁNICO

Pantalla 1: Dashboard del Mecánico

Comportamiento:

- El dashboard es la pantalla de inicio del mecánico

- Muestra resumen de sus reportes (no del sistema)
- Acceso rápido a equipos pendientes
- Los números son SOLO de sus propios reportes

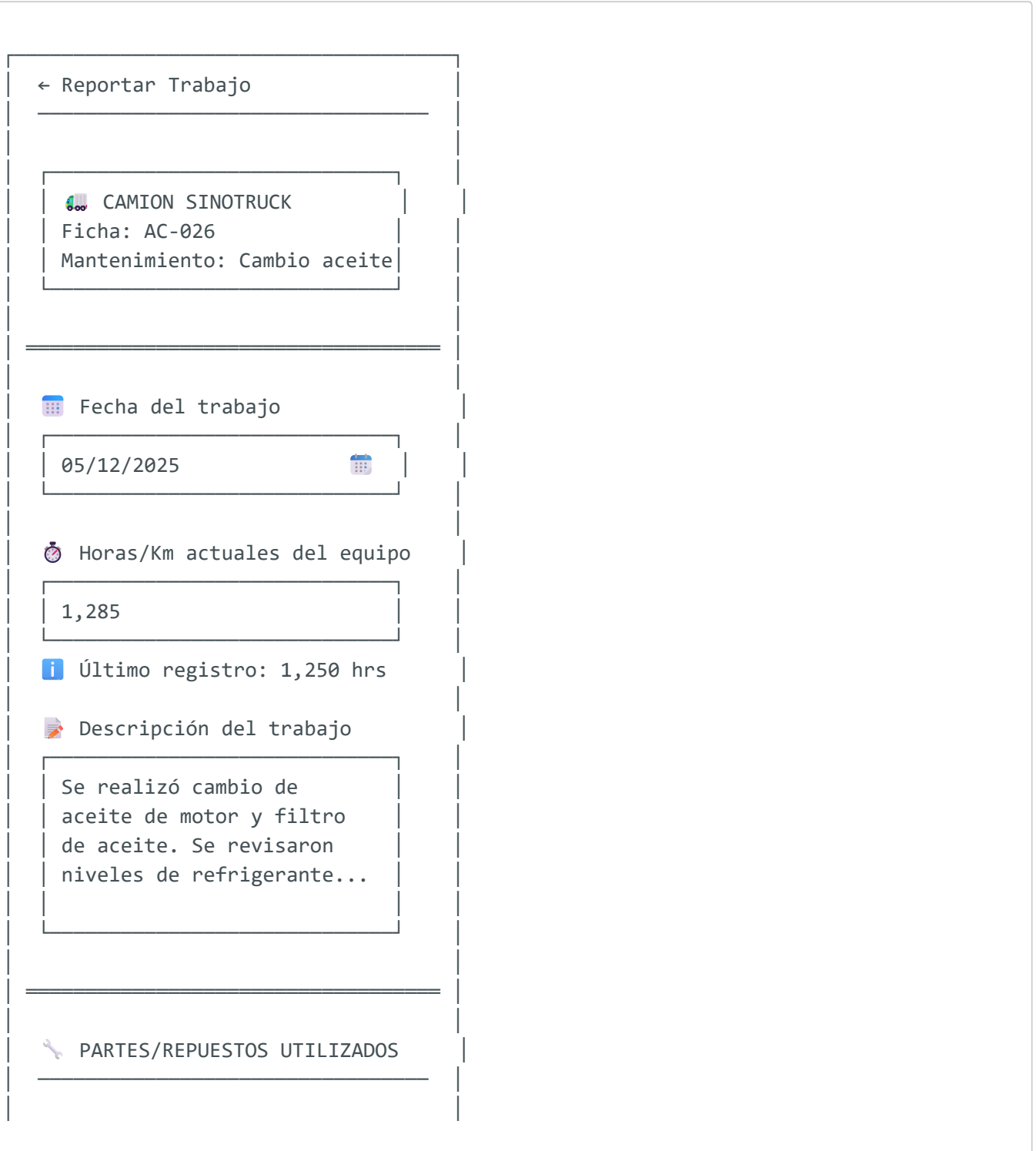
Pantalla 2: Lista de Equipos Pendientes

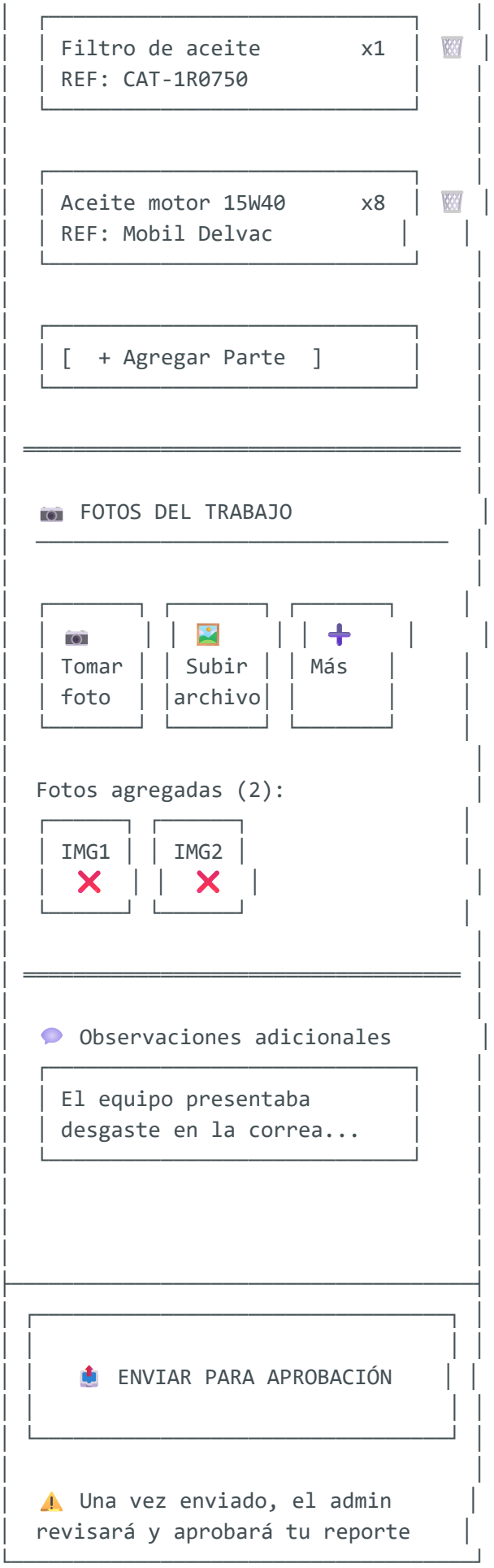
Comportamiento:

- Lista ordenada por urgencia (vencidos primero, luego próximos)
- Badge visual de estado (rojo=vencido, amarillo=próximo, verde=ok)
- Cada equipo tiene UN solo botón: "Reportar Trabajo"
- El mecánico NO puede ver detalles completos del equipo, solo lo necesario
- NO hay botones de editar/eliminar

Pantalla 3: Formulario de Reporte de Trabajo (CRÍTICA)

Esta es la pantalla más importante. Diseño mobile-first para 240x561:





Comportamiento detallado:

1. **Header fijo**: Muestra el equipo seleccionado (no editable)

2. **Campo Fecha**:

    - Default: fecha actual
    - Puede seleccionar fecha pasada (máx 7 días atrás)
    - NO puede seleccionar fecha futura

3. **Campo Horas/Km**:

    - Input numérico
    - Muestra el último registro como referencia
    - Validación: debe ser >= último registro
    - Tooltip: "Registra las horas/km actuales del equipo"

4. **Descripción del trabajo**:

    - Textarea multilínea
    - Placeholder con ejemplo: "Describa el trabajo realizado..."
    - Mínimo 20 caracteres requeridos

5. **Partes utilizadas**:

    - Lista dinámica (agregar/quitar)
    - Cada parte tiene: nombre, cantidad, referencia (opcional)
    - Botón "Agregar parte" abre modal/sheet con:
        - Opción 1: Buscar en inventario (autocomplete)
        - Opción 2: Escribir manualmente (si no está en inventario)
    - Si selecciona del inventario, se guarda el `inventario_id`
    - El admin verá esto y podrá descontar del stock al aprobar

6. **Fotos**:

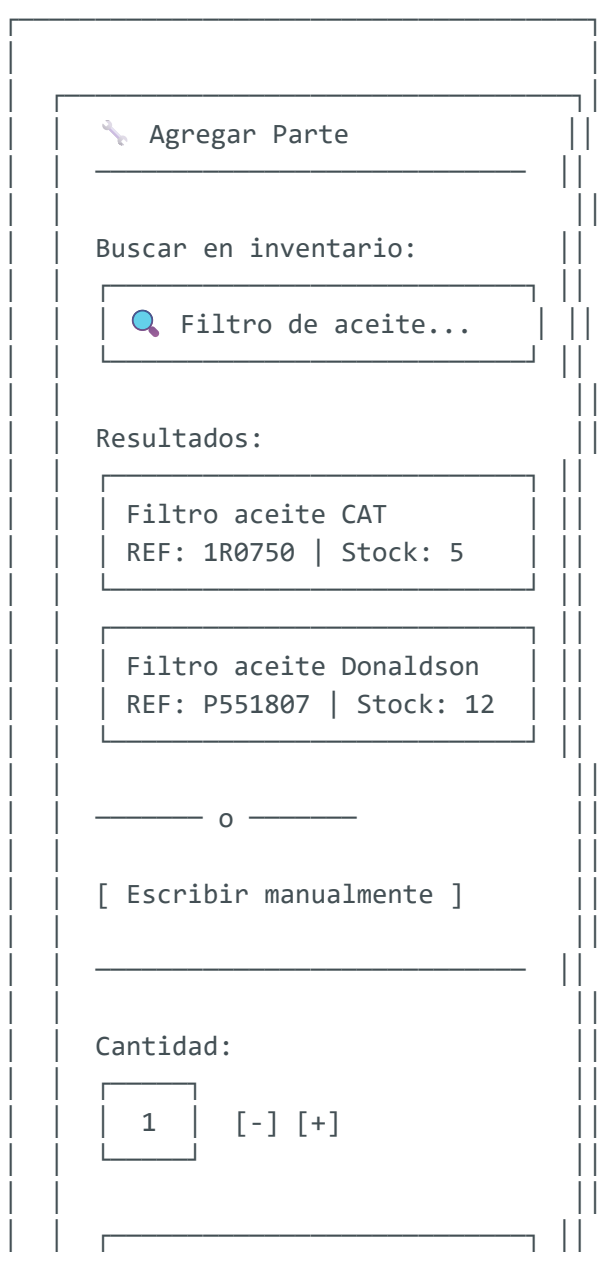
    - Mínimo 1 foto requerida
    - Máximo 5 fotos
    - Opciones: Tomar foto (cámara) o Subir archivo
    - Preview con opción de eliminar
    - Compresión automática a max 1MB
    - Formatos: JPG, PNG, WEBP

7. **Observaciones**:

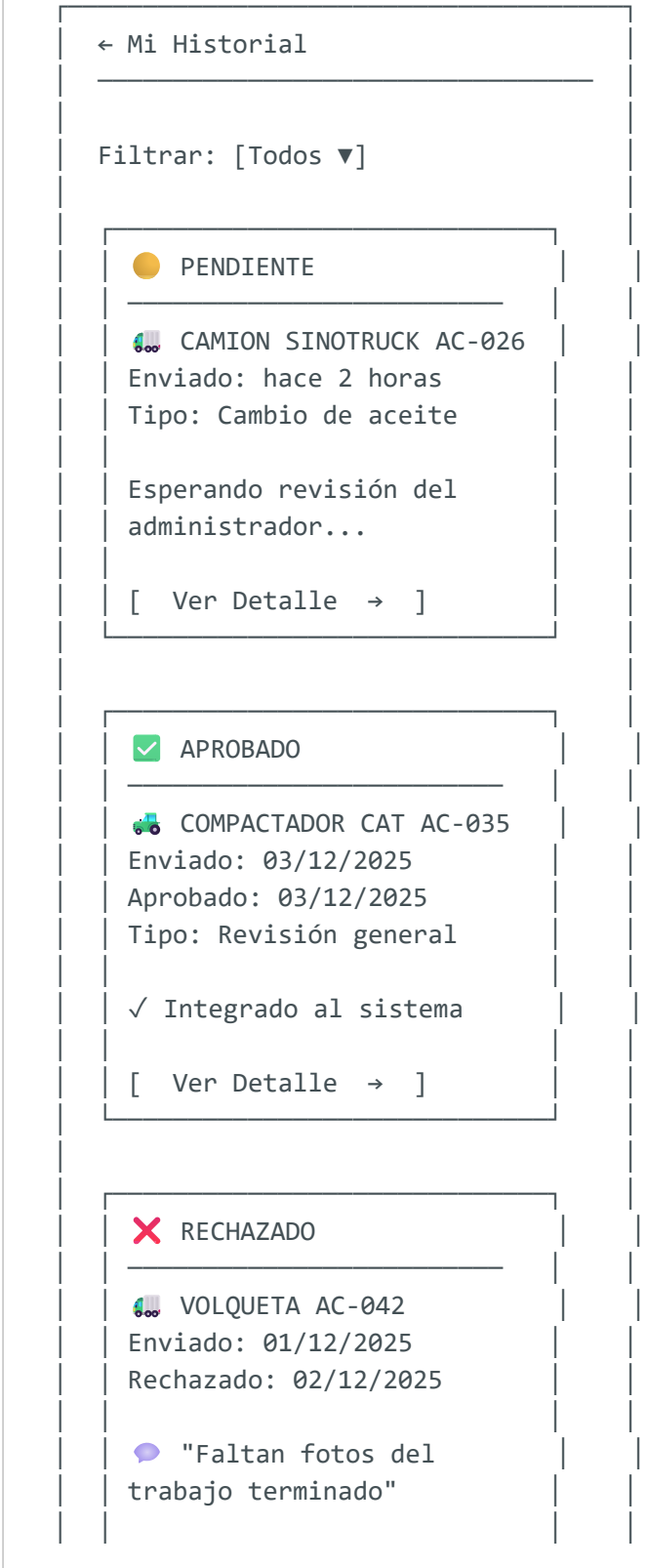
    - Opcional
    - Para notas adicionales, alertas, sugerencias

8. **Botón Enviar**:
    - Fixed en bottom (sticky)
    - Color primario, grande (min 48px alto)
    - Al pulsar:
        - Validar todos los campos
        - Mostrar loading
        - Subir fotos a Storage
        - Crear registro en `maintenance_submissions`
        - Mostrar confirmación
        - Redirigir al historial

Pantalla 4: Modal de Agregar Parte



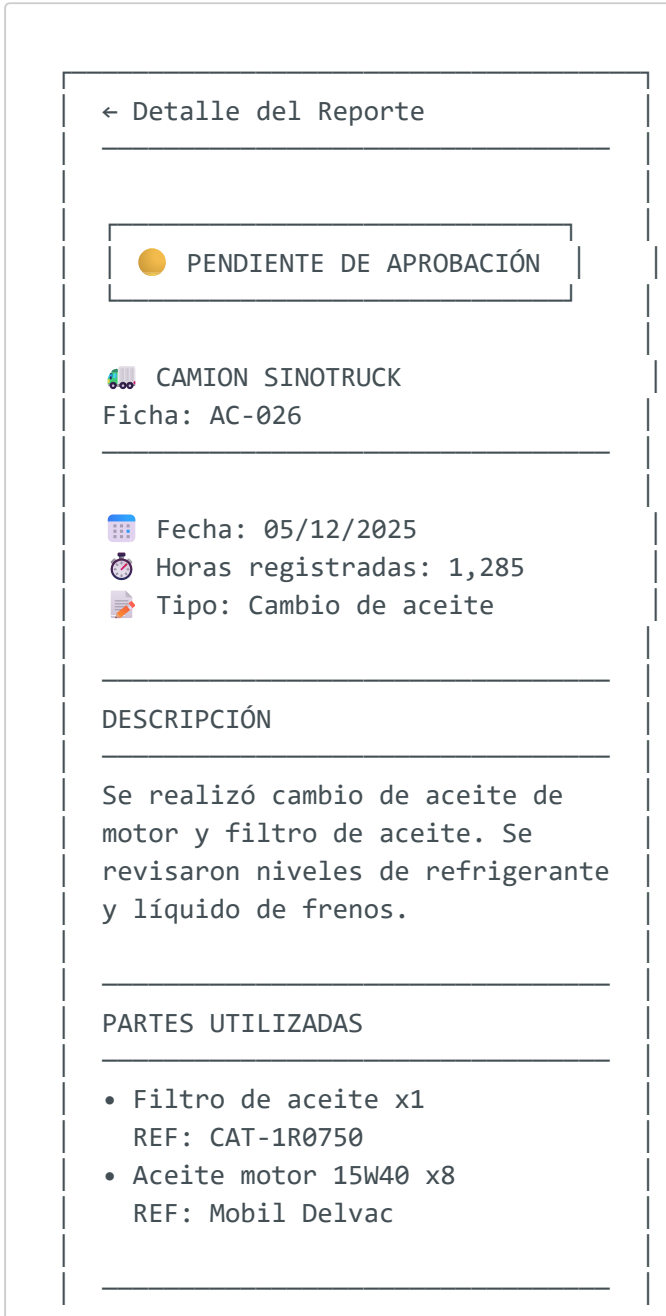
Pantalla 5: Historial del Mecánico



Comportamiento:

- Lista de TODOS los reportes del mecánico
- Filtro por estado
- Badge de color según estado
- Si está rechazado, muestra el feedback del admin
- Opción de "corregir" solo para rechazados (crea nueva submission)

Pantalla 6: Detalle de Reporte

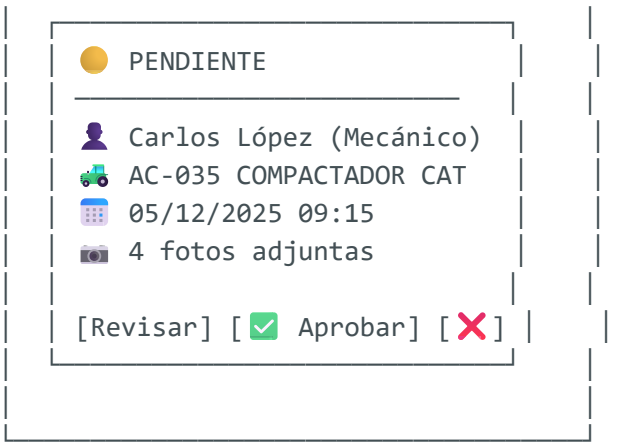


```
FOTOS
──────────────────────────────────

┌──────┐ ┌──────┐
│      │ │      │
│ IMG1 │ │ IMG2 │  (tap to zoom)
│      │ │      │
└──────┘ └──────┘

──────────────────────────────────
OBSERVACIONES
──────────────────────────────────
El equipo presentaba desgaste en
la correa del alternador.
Recomiendo revisión pronto.

──────────────────────────────────
TIMELINE
──────────────────────────────────
📤 Enviado: 05/12/2025 10:30
⏳ En espera de aprobación...
```

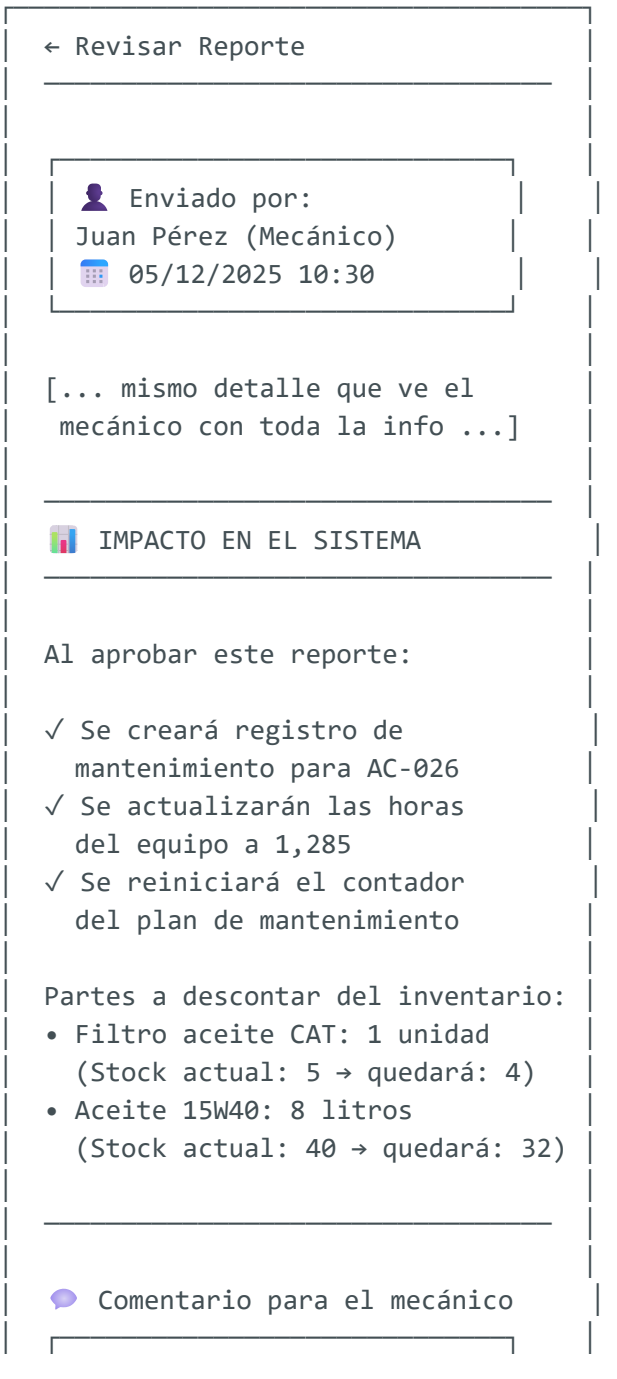
👨‍💼 VISTA DEL ADMIN: Gestión de Reportes

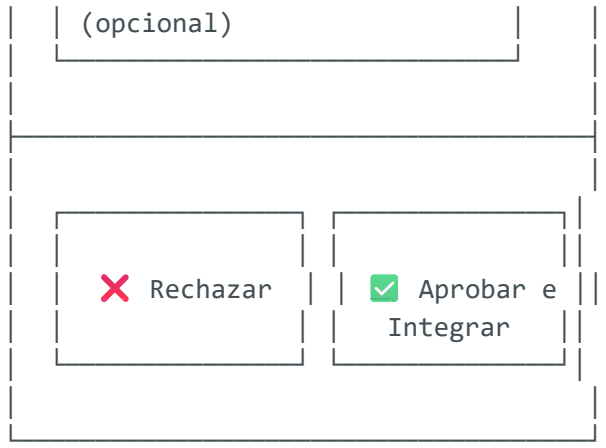
Pantalla Admin: Lista de Reportes Pendientes

```
📋 Reportes de Mecánicos
──────────────────────────────────

🔔 Tienes 3 reportes pendientes

Filtros: [Pendientes ▼] [Todos ▼]

┌──────────────────────────┐
│ 🟡 PENDIENTE             │
│ ──────────────────────   │
│ 👤 Juan Pérez (Mecánico) │
│ 🚚 AC-026 CAMION SINOTRUCK │
│ 🗓️ 05/12/2025 10:30      │
│ 📷 2 fotos adjuntas      │
│                          │
│ [Revisar] [✅ Aprobar] [❌] │
└──────────────────────────┘
```



Pantalla Admin: Revisar y Aprobar Reporte





Comportamiento del botón "Aprobar e Integrar":

1. Muestra confirmación: "¿Estás seguro de aprobar este reporte?"
2. Al confirmar:
   - Llama a función RPC `approve_and_integrate_submission`
   - Crea registro en `historial_mantenimiento`
   - Actualiza horas/km del equipo
   - Descuenta partes del inventario (si aplica)
   - Marca submission como `integrated`
   - Notifica al mecánico
   - Redirige al admin a Control de Mantenimiento mostrando el nuevo registro

Comportamiento del botón "Rechazar":

1. Abre modal pidiendo motivo (obligatorio)
2. Al confirmar:
   - Llama a función `reject_submission`
   - Marca submission como `rejected`
   - Guarda feedback
   - Notifica al mecánico con el motivo
   - El mecánico puede crear un nuevo reporte corregido

👔 ROL: SUPERVISOR (supervisor)

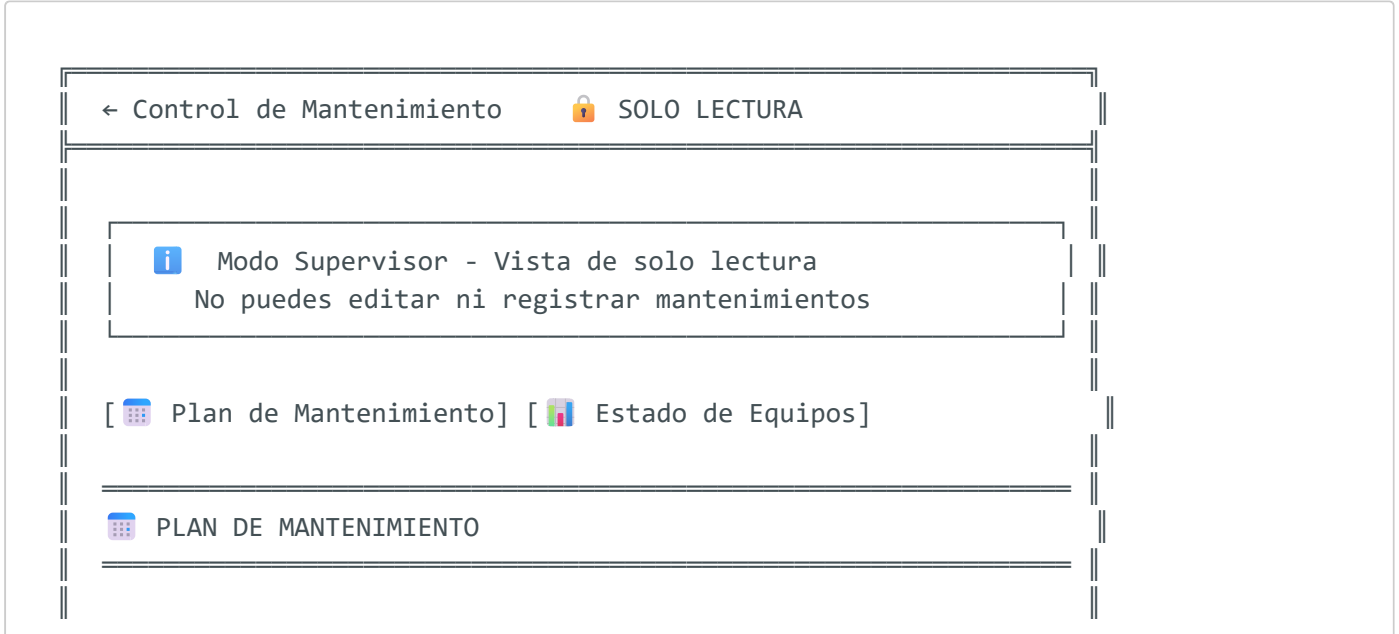
Permisos:

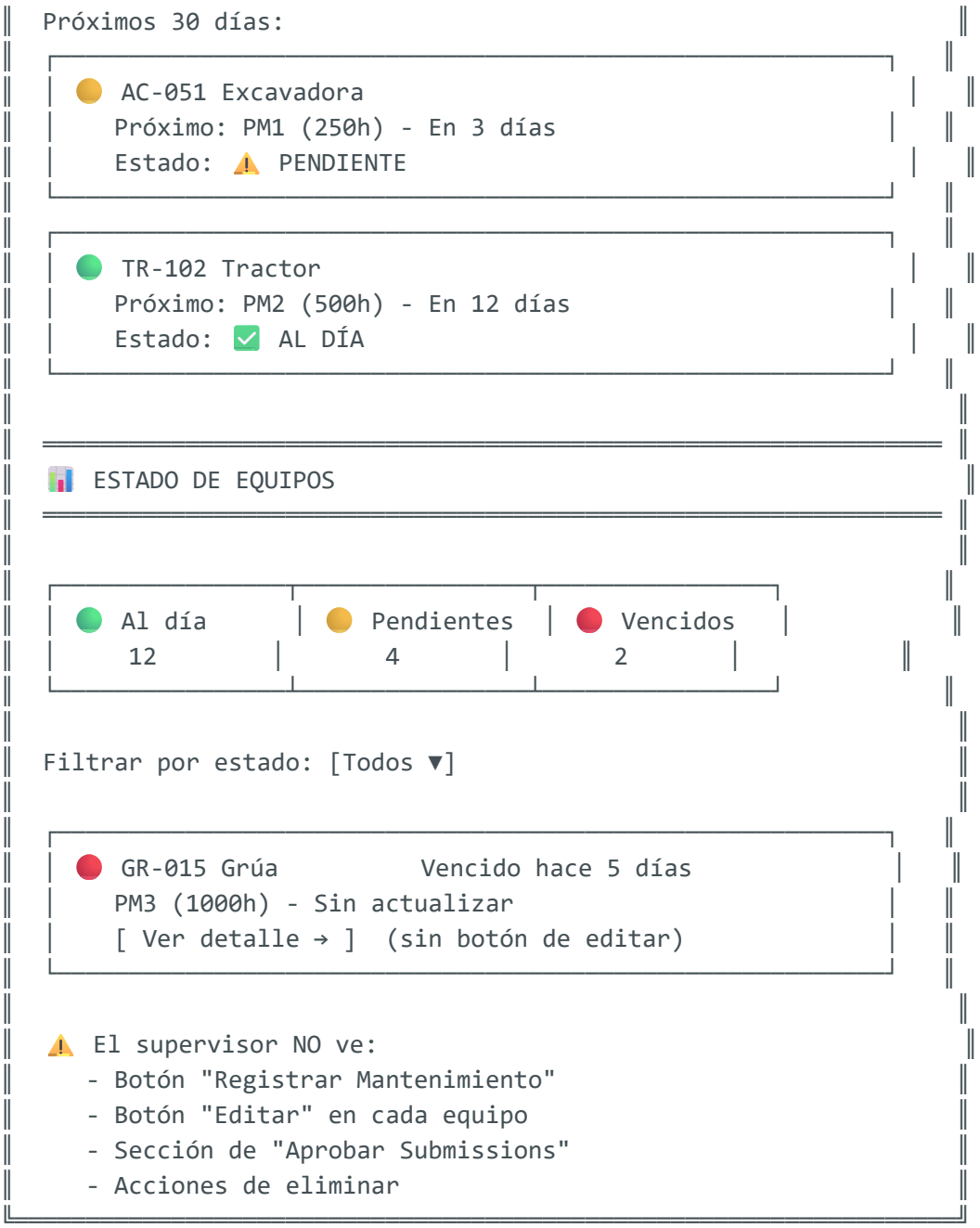
Módulo	Ver	Crear	Editar	Eliminar
Dashboard	✅	❌	❌	❌
Equipos	✅	❌	❌	❌
Control Mantenimiento (Plan y Estado)	✅	❌	❌	❌
Control Mantenimiento (Acciones)	❌	❌	❌	❌
Historial	✅	❌	❌	❌
Inventario	✅	❌	❌	❌
Reportes	✅	❌	❌	❌
Configuración	❌	❌	❌	❌
Usuarios	❌	❌	❌	❌

Diferencia clave con Admin:

- El supervisor **solo ve** la sección de "Plan de Mantenimiento" y "Estado" (pendientes/actualizados)
- **NO ve** los botones de acción (registrar, editar, eliminar)
- **NO ve** la sección de "Registrar Mantenimiento" del módulo Control
- **NO puede** aprobar submissions de mecánicos

Wireframe: Vista del Supervisor en Control Mantenimiento





Componentes a modificar para Supervisor:

```
// En cada componente que tenga acciones, verificar rol:

const { hasPermission } = useUserRoles();

// En ControlMantenimiento.tsx
{hasPermission('maintenance:write') && (
  <Button onClick={openRegistrarForm}>
    Registrar Mantenimiento
  </Button>
)}

// En EquipmentCard.tsx
{hasPermission('equipment:write') && (
  <DropdownMenuItem onClick={handleEdit}>Editar</DropdownMenuItem>
```

```
)}
{hasPermission('equipment:delete') && (
  <DropdownMenuItem onClick={handleDelete}>Eliminar</DropdownMenuItem>
)}
```

Banner de Solo Lectura:

```
// components/ui/ReadOnlyBanner.tsx
export function ReadOnlyBanner() {
  const { isRole } = useUserRoles();

  if (!isRole('supervisor')) return null;

  return (
    <div className="bg-blue-50 border-b border-blue-200 px-4 py-2">
      <div className="flex items-center gap-2 text-blue-700 text-sm">
        <Lock className="h-4 w-4" />
        <span>Modo Supervisor - Vista de solo lectura</span>
      </div>
    </div>
  );
}
```

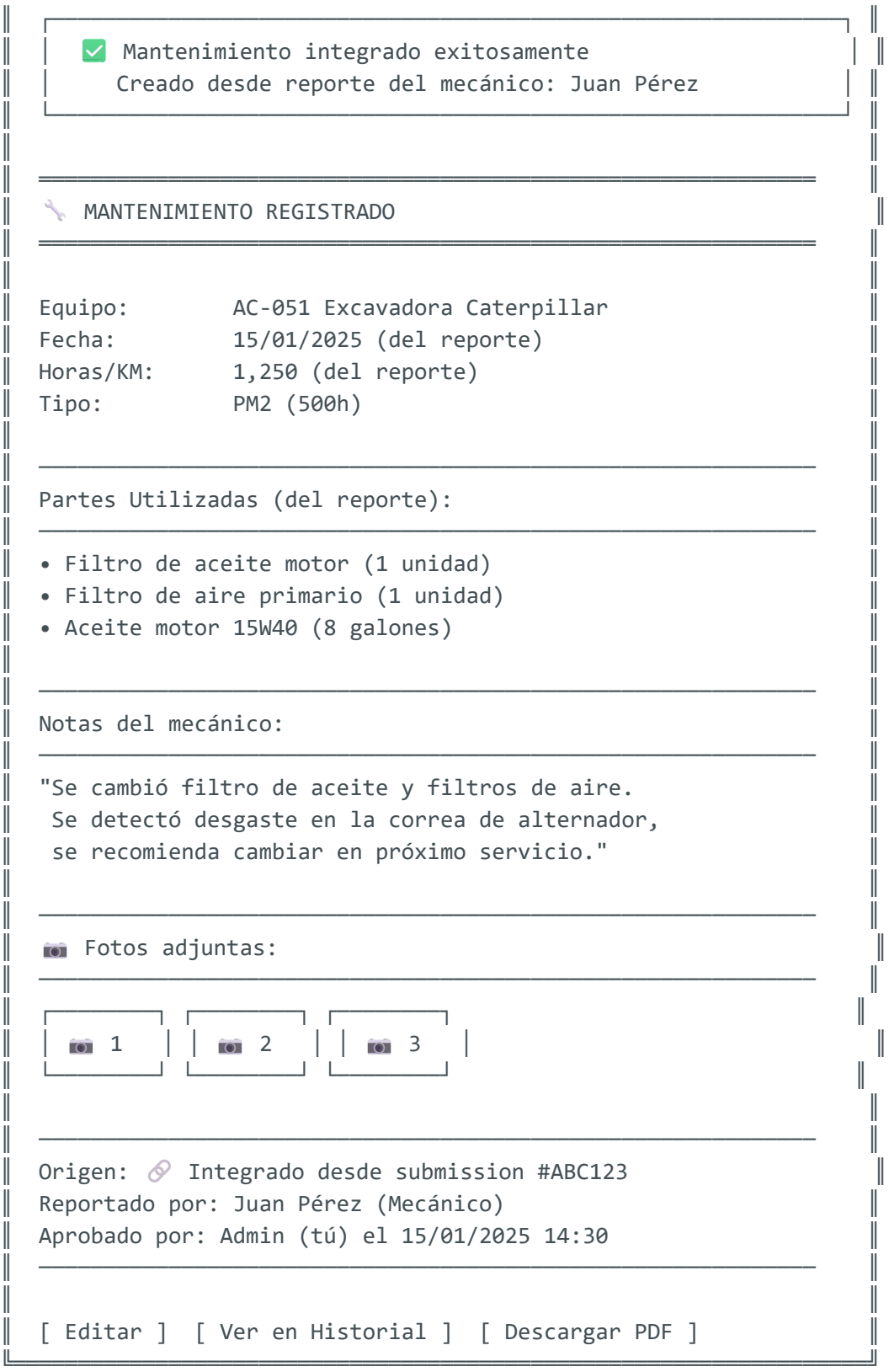
🔗 POST-INTEGRACIÓN: Cómo se ve el registro integrado

Cuando el admin hace click en "Aprobar e Integrar", sucede esto:

1. **Se crea automáticamente** un registro en la tabla `mantenimientos` (la oficial)
2. **Se redirige** al admin a la vista de ese mantenimiento en Control
3. **Los datos aparecen prellenados** del reporte del mecánico

Wireframe: Control Mantenimiento después de integrar

```
╔═══════════════════════════════════════════════════════════╗
║  ← Control de Mantenimiento                               ║
╠═══════════════════════════════════════════════════════════╣
```



Datos que se copian automáticamente:

Campo en Submission	→	Campo en Mantenimiento Oficial
equipo_id	→	equipo_id
fecha_mantenimiento	→	fecha

Campo en Submission	→	Campo en Mantenimiento Oficial
`horas_km_actuales`	→	`horas_km`
`tipo_mantenimiento`	→	`tipo`
`descripcion`	→	`descripcion`
`notas_adicionales`	→	`observaciones`
`partes_utilizadas[]`	→	Se crean registros en `mantenimiento_partes`
`fotos[]`	→	Se copian URLs a `mantenimiento_fotos`
`created_by` (mecánico)	→	`realizado_por` (referencia)
(nuevo)	→	`aprobado_por` (admin que aprobó)
(nuevo)	→	`submission_id` (referencia a origen)

Trazabilidad:

El registro oficial de mantenimiento mantiene referencia al submission original:

```
-- En la tabla mantenimientos, agregar columnas de trazabilidad:
ALTER TABLE mantenimientos ADD COLUMN IF NOT EXISTS
  submission_id UUID REFERENCES maintenance_submissions(id);

ALTER TABLE mantenimientos ADD COLUMN IF NOT EXISTS
  realizado_por UUID REFERENCES auth.users(id);

ALTER TABLE mantenimientos ADD COLUMN IF NOT EXISTS
  aprobado_por UUID REFERENCES auth.users(id);
```

Esto permite:

- Ver qué mantenimientos vinieron de reportes de mecánicos
- Auditar quién hizo el trabajo vs quién lo aprobó
- Mantener las fotos y evidencias originales
- Generar reportes de productividad por mecánico

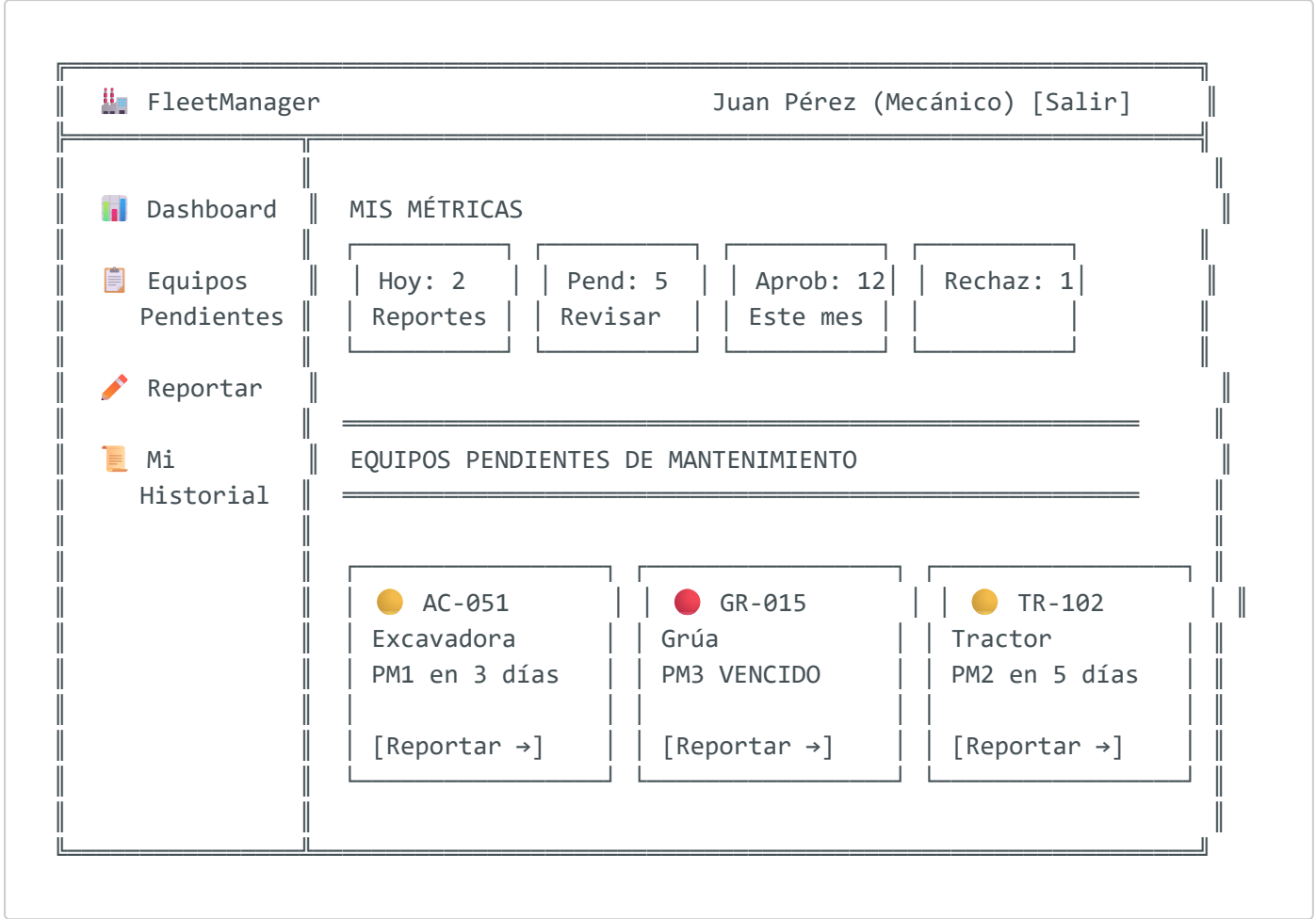
📱 VERSIÓN WEB DEL MECÁNICO

El mecánico puede acceder desde web también (no solo móvil). La interfaz se adapta:

Breakpoints:

Viewport	Diseño
< 640px (mobile)	Diseño móvil, BottomNav, tarjetas apiladas
640px-1024px (tablet)	Sidebar colapsada, grid de 2 columnas
> 1024px (desktop)	Sidebar expandida, grid de 3 columnas

En desktop, el mecánico ve:



Modelo de Datos (SQL para Supabase)

Genera las siguientes migraciones SQL:

```sql
-- ============================================
-- MIGRACIÓN: Nuevos roles y tablas para mecánico
-- ============================================

-- 1. Asegurar que existan los roles en la tabla roles
INSERT INTO roles (name, description) VALUES
  ('mechanic', 'Mecánico - puede registrar mantenimientos y enviar para
aprobación'),
  ('supervisor', 'Supervisor - acceso de solo lectura a dashboards y reportes')
ON CONFLICT (name) DO NOTHING;

-- 2. Tabla de submissions de mantenimiento (propuestas del mecánico)
CREATE TABLE IF NOT EXISTS maintenance_submissions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  created_by UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,
  equipo_id UUID NOT NULL REFERENCES equipos(id) ON DELETE CASCADE,

  -- Datos del mantenimiento
  fecha_mantenimiento TIMESTAMPTZ NOT NULL DEFAULT NOW(),
  horas_km_actuales INTEGER NOT NULL,
  tipo_mantenimiento TEXT,
  descripcion_trabajo TEXT,
  observaciones TEXT,

  -- Partes utilizadas (JSON array)
  partes_usadas JSONB DEFAULT '[]'::jsonb,
  -- Formato: [{"nombre": "Filtro aceite", "cantidad": 1, "referencia": "ABC123",
"del_inventario": true, "inventario_id": "uuid"}]

  -- Estado del flujo
  status TEXT NOT NULL DEFAULT 'pending' CHECK (status IN ('pending', 'approved',
'rejected', 'integrated')),

  -- Revisión del admin
  reviewed_by UUID REFERENCES auth.users(id),
  reviewed_at TIMESTAMPTZ,
  admin_feedback TEXT,

  -- Vinculación con mantenimiento oficial (después de integrar)
  linked_maintenance_id UUID REFERENCES historial_mantenimiento(id),

  -- Timestamps
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- 3. Tabla de adjuntos (fotos, documentos)
CREATE TABLE IF NOT EXISTS submission_attachments (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  submission_id UUID NOT NULL REFERENCES maintenance_submissions(id) ON DELETE
CASCADE,

  storage_path TEXT NOT NULL,  -- path en Supabase Storage
  filename TEXT NOT NULL,
  mime_type TEXT,
  file_size INTEGER,
```

```
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- 4. Tabla de notificaciones (si no existe)
CREATE TABLE IF NOT EXISTS notifications (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,

  type TEXT NOT NULL,  -- 'submission_received', 'submission_approved',
'submission_rejected'
  title TEXT NOT NULL,
  message TEXT,
  payload JSONB DEFAULT '{}'::jsonb,

  read BOOLEAN DEFAULT FALSE,
  read_at TIMESTAMPTZ,

  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- 5. Índices para performance
CREATE INDEX IF NOT EXISTS idx_submissions_created_by ON
maintenance_submissions(created_by);
CREATE INDEX IF NOT EXISTS idx_submissions_status ON
maintenance_submissions(status);
CREATE INDEX IF NOT EXISTS idx_submissions_equipo ON
maintenance_submissions(equipo_id);
CREATE INDEX IF NOT EXISTS idx_attachments_submission ON
submission_attachments(submission_id);
CREATE INDEX IF NOT EXISTS idx_notifications_user ON notifications(user_id);
CREATE INDEX IF NOT EXISTS idx_notifications_unread ON notifications(user_id) WHERE
read = FALSE;

-- 6. Trigger para updated_at
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
  NEW.updated_at = NOW();
  RETURN NEW;
END;
$$ language 'plpgsql';

CREATE TRIGGER update_submissions_updated_at
  BEFORE UPDATE ON maintenance_submissions
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

-- ============================================
-- POLÍTICAS RLS (Row Level Security)
-- ============================================

ALTER TABLE maintenance_submissions ENABLE ROW LEVEL SECURITY;
ALTER TABLE submission_attachments ENABLE ROW LEVEL SECURITY;
ALTER TABLE notifications ENABLE ROW LEVEL SECURITY;

-- Submissions: mecánico puede crear y ver las suyas; admin puede ver y actualizar
todas
```

```sql
CREATE POLICY "Mechanics can insert own submissions" ON maintenance_submissions
  FOR INSERT TO authenticated
  WITH CHECK (auth.uid() = created_by);

CREATE POLICY "Users can view own submissions" ON maintenance_submissions
  FOR SELECT TO authenticated
  USING (
    auth.uid() = created_by
    OR EXISTS (
      SELECT 1 FROM user_roles ur
      JOIN roles r ON ur.role_id = r.id
      WHERE ur.user_id = auth.uid() AND r.name IN ('admin', 'supervisor')
    )
  );

CREATE POLICY "Admins can update submissions" ON maintenance_submissions
  FOR UPDATE TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM user_roles ur
      JOIN roles r ON ur.role_id = r.id
      WHERE ur.user_id = auth.uid() AND r.name = 'admin'
    )
  );

-- Attachments: mismo patrón
CREATE POLICY "Users can manage own attachments" ON submission_attachments
  FOR ALL TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM maintenance_submissions ms
      WHERE ms.id = submission_id AND ms.created_by = auth.uid()
    )
    OR EXISTS (
      SELECT 1 FROM user_roles ur
      JOIN roles r ON ur.role_id = r.id
      WHERE ur.user_id = auth.uid() AND r.name IN ('admin', 'supervisor')
    )
  );

-- Notifications: solo el usuario destinatario
CREATE POLICY "Users can view own notifications" ON notifications
  FOR SELECT TO authenticated
  USING (auth.uid() = user_id);

CREATE POLICY "Users can update own notifications" ON notifications
  FOR UPDATE TO authenticated
  USING (auth.uid() = user_id);

-- ============================================
-- FUNCIÓN RPC: Aprobar e integrar submission
-- ============================================

CREATE OR REPLACE FUNCTION approve_and_integrate_submission(
  p_submission_id UUID,
  p_admin_feedback TEXT DEFAULT NULL
)
```

```
RETURNS JSON
LANGUAGE plpgsql
SECURITY DEFINER
AS $$
DECLARE
  v_submission maintenance_submissions%ROWTYPE;
  v_new_maintenance_id UUID;
  v_mechanic_id UUID;
BEGIN
  -- Verificar que el usuario es admin
  IF NOT EXISTS (
    SELECT 1 FROM user_roles ur
    JOIN roles r ON ur.role_id = r.id
    WHERE ur.user_id = auth.uid() AND r.name = 'admin'
  ) THEN
    RAISE EXCEPTION 'Solo administradores pueden aprobar submissions';
  END IF;

  -- Obtener submission
  SELECT * INTO v_submission FROM maintenance_submissions WHERE id =
p_submission_id;

  IF v_submission IS NULL THEN
    RAISE EXCEPTION 'Submission no encontrada';
  END IF;

  IF v_submission.status != 'pending' THEN
    RAISE EXCEPTION 'Solo se pueden aprobar submissions pendientes';
  END IF;

  v_mechanic_id := v_submission.created_by;

  -- Crear registro oficial en historial_mantenimiento
  INSERT INTO historial_mantenimiento (
    equipo_id,
    fecha_mantenimiento,
    horas_km_al_momento,
    tipo_mantenimiento,
    descripcion,
    observaciones,
    realizado_por,
    created_at
  ) VALUES (
    v_submission.equipo_id,
    v_submission.fecha_mantenimiento,
    v_submission.horas_km_actuales,
    v_submission.tipo_mantenimiento,
    v_submission.descripcion_trabajo,
    v_submission.observaciones,
    v_mechanic_id,
    NOW()
  ) RETURNING id INTO v_new_maintenance_id;

  -- Actualizar submission
  UPDATE maintenance_submissions SET
    status = 'integrated',
    reviewed_by = auth.uid(),
```

```
    reviewed_at = NOW(),
    admin_feedback = p_admin_feedback,
    linked_maintenance_id = v_new_maintenance_id
  WHERE id = p_submission_id;

  -- Notificar al mecánico
  INSERT INTO notifications (user_id, type, title, message, payload)
  VALUES (
    v_mechanic_id,
    'submission_approved',
    'Mantenimiento Aprobado',
    'Tu registro de mantenimiento ha sido aprobado e integrado al sistema.',
    jsonb_build_object('submission_id', p_submission_id, 'maintenance_id',
v_new_maintenance_id)
  );

  RETURN jsonb_build_object(
    'success', true,
    'maintenance_id', v_new_maintenance_id,
    'message', 'Submission aprobada e integrada correctamente'
  );
END;
$$;

-- Función para rechazar
CREATE OR REPLACE FUNCTION reject_submission(
  p_submission_id UUID,
  p_feedback TEXT
)
RETURNS JSON
LANGUAGE plpgsql
SECURITY DEFINER
AS $$
DECLARE
  v_submission maintenance_submissions%ROWTYPE;
BEGIN
  -- Verificar admin
  IF NOT EXISTS (
    SELECT 1 FROM user_roles ur
    JOIN roles r ON ur.role_id = r.id
    WHERE ur.user_id = auth.uid() AND r.name = 'admin'
  ) THEN
    RAISE EXCEPTION 'Solo administradores pueden rechazar submissions';
  END IF;

  SELECT * INTO v_submission FROM maintenance_submissions WHERE id =
p_submission_id;

  IF v_submission IS NULL OR v_submission.status != 'pending' THEN
    RAISE EXCEPTION 'Submission no válida para rechazo';
  END IF;

  UPDATE maintenance_submissions SET
    status = 'rejected',
    reviewed_by = auth.uid(),
    reviewed_at = NOW(),
    admin_feedback = p_feedback
```

```
  WHERE id = p_submission_id;

  -- Notificar al mecánico
  INSERT INTO notifications (user_id, type, title, message, payload)
  VALUES (
    v_submission.created_by,
    'submission_rejected',
    'Mantenimiento Rechazado',
    COALESCE(p_feedback, 'Tu registro de mantenimiento ha sido rechazado.'),
    jsonb_build_object('submission_id', p_submission_id, 'feedback', p_feedback)
  );

  RETURN jsonb_build_object('success', true, 'message', 'Submission rechazada');
END;
$$;
```

Componentes Frontend a Crear

Para el Mecánico (Mobile-first)

Archivos nuevos:

1. `src/pages/mobile/MechanicDashboard.tsx`
    - Dashboard del mecánico con:
        - Resumen: submissions pendientes, aprobadas, rechazadas
        - Acceso rápido a "Registrar Mantenimiento"
        - Lista de equipos asignados/pendientes
2. `src/pages/mobile/MechanicPendingList.tsx`
    - Lista de equipos que necesitan mantenimiento
    - Cada item muestra: ficha, nombre, horas/km actuales, fecha último mantenimiento
    - Botón "Registrar" que abre el formulario
3. `src/pages/mobile/MechanicSubmissionForm.tsx`
    - Formulario completo para registrar mantenimiento:
        - Select de equipo (si no viene preseleccionado)
        - Input fecha (default: hoy)
        - Input horas/km actuales (numérico)

        - Select tipo de mantenimiento
        - Textarea descripción del trabajo
        - **Sección "Partes Utilizadas"**: lista dinámica con campos:
            - Nombre de la parte
            - Cantidad
            - Referencia/código (opcional)
            - Checkbox "Del inventario" (para descontar stock)
        - **Sección "Fotos/Adjuntos"**:
            - Botón "Tomar foto" (usa cámara en móvil)
            - Botón "Subir archivo"
            - Preview de imágenes con opción de eliminar
            - Compresión automática antes de subir
        - Textarea observaciones
        - Botón principal sticky bottom: "Enviar para Aprobación"

4. `src/pages/mobile/MechanicHistory.tsx`
    - Historial de submissions del mecánico
    - Filtros por estado (pending, approved, rejected, integrated)
    - Cada item muestra estado con badge de color

5. `src/pages/mobile/MechanicSubmissionDetail.tsx`
    - Vista detalle de una submission
    - Muestra todos los datos, fotos, feedback del admin si existe

Componentes compartidos:

6. `src/components/mechanic/PartsUsedInput.tsx`
    - Input dinámico para agregar/quitar partes
    - Autocompletado desde inventario (opcional)

7. `src/components/mechanic/PhotoUploader.tsx`
    - Componente de upload con:
        - Acceso a cámara (capture="environment")
        - Preview de imágenes
        - Compresión client-side (max 1MB por imagen)
        - Progress bar de upload
        - Integración con Supabase Storage

8. `src/components/mechanic/SubmissionCard.tsx`
    - Card para mostrar una submission en listas
    - Badge de estado con colores semánticos

Para el Admin

9. `src/pages/admin/SubmissionsList.tsx`
    - Lista de todas las submissions pendientes de revisión
    - Filtros por estado, mecánico, equipo
    - Quick actions: ver detalle, aprobar, rechazar
10. `src/pages/admin/SubmissionReview.tsx`
    - Vista completa de una submission para revisar
    - Galería de fotos con zoom
    - Lista de partes usadas
    - Botones: "Aprobar e Integrar" y "Rechazar" (con modal para feedback)

Para el Supervisor

11. Modificar componentes existentes para detectar rol supervisor y:
    - Ocultar botones de acción (crear, editar, eliminar)
    - Mostrar banner "Vista de solo lectura"
    - Deshabilitar formularios

Archivos a modificar:

- `src/hooks/useUserRoles.ts` - agregar checks para 'mechanic' y 'supervisor'
- `src/lib/permissions.ts` - agregar constantes y funciones para nuevos permisos
- `src/components/BottomNav.tsx` - mostrar navegación diferente según rol
- `src/App.tsx` - agregar rutas protegidas para cada rol

Hooks y Utilities

```
// src/hooks/useMechanicSubmissions.ts
export function useMechanicSubmissions() {
  // Fetch submissions del mecánico actual
  // Create new submission
  // Upload attachments
}

// src/hooks/useAdminSubmissions.ts
export function useAdminSubmissions() {
  // Fetch all pending submissions
  // Approve submission
  // Reject submission
}

// src/hooks/useNotifications.ts
export function useNotifications() {
  // Fetch unread notifications
  // Mark as read
  // Subscribe to realtime updates
}
```

Especificaciones de UI Mobile (240x561)

```
/* Reglas clave para pantallas ultra-pequeñas */
.mechanic-form {
  @apply flex flex-col min-h-screen;
}

.mechanic-form__content {
  @apply flex-1 overflow-y-auto p-2 space-y-3;
}

.mechanic-form__actions {
  @apply sticky bottom-0 p-2 bg-background border-t safe-area-pb;
}

/* Inputs más compactos */
@screen 3xs {
  .form-input {
    @apply text-sm py-2;
  }

  .form-label {
    @apply text-xs;
  }
}

/* Touch targets mínimos */
.touch-target {
```

```
  @apply min-h-[44px] min-w-[44px];
}

/* Preview de fotos */
.photo-grid {
  @apply grid grid-cols-3 gap-1;
}

.photo-grid__item {
  @apply aspect-square rounded overflow-hidden relative;
}

/* Parts list compacta */
.parts-list__item {
  @apply flex items-center gap-2 p-2 bg-muted rounded;
}
```

Supabase Storage Setup

```
// Crear bucket 'submissions' con las siguientes políticas:

// 1. Bucket config
const bucket = 'submissions';
const maxFileSize = 5 * 1024 * 1024; // 5MB
const allowedMimeTypes = ['image/jpeg', 'image/png', 'image/webp',
'application/pdf'];

// 2. Storage policies (en Supabase Dashboard o SQL):
/*
-- Mecánicos pueden subir a su carpeta
CREATE POLICY "Mechanics can upload own files"
ON storage.objects FOR INSERT
WITH CHECK (
  bucket_id = 'submissions'
  AND auth.uid()::text = (storage.foldername(name))[1]
);

-- Usuarios pueden ver sus propios archivos y admins pueden ver todos
CREATE POLICY "Users can view authorized files"
ON storage.objects FOR SELECT
USING (
  bucket_id = 'submissions'
  AND (
    auth.uid()::text = (storage.foldername(name))[1]
    OR EXISTS (
      SELECT 1 FROM user_roles ur
      JOIN roles r ON ur.role_id = r.id
      WHERE ur.user_id = auth.uid() AND r.name IN ('admin', 'supervisor')
    )
  )
```

```
);
*/
```

Flujo de Navegación

```
MECÁNICO:
/mobile/mechanic
  ├── /dashboard (MechanicDashboard)
  ├── /pending (MechanicPendingList)
  ├── /submit/:equipoId? (MechanicSubmissionForm)
  ├── /history (MechanicHistory)
  └── /submission/:id (MechanicSubmissionDetail)

ADMIN (adicional a rutas existentes):
/admin/submissions (SubmissionsList)
/admin/submissions/:id (SubmissionReview)

SUPERVISOR:
- Mismas rutas que admin pero con permisos de solo lectura
- /dashboard, /equipos, /mantenimiento (vista), /reportes
```

Criterios de Aceptación

1. ✅ Mecánico puede crear cuenta y ser asignado rol `mechanic`
2. ✅ Mecánico ve solo sus equipos asignados o todos los pendientes (configureable)
3. ✅ Formulario de submission funciona en móvil 240x561 sin cortes
4. ✅ Fotos se comprimen antes de subir y se muestran en preview
5. ✅ Partes usadas se registran correctamente en JSON
6. ✅ Admin recibe notificación cuando hay nueva submission
7. ✅ Admin puede aprobar con un click y el registro se crea automáticamente
8. ✅ Admin puede rechazar con feedback obligatorio
9. ✅ Mecánico recibe notificación del resultado
10. ✅ Supervisor puede ver todo pero no puede modificar nada
11. ✅ RLS policies funcionan correctamente
12. ✅ Tests pasan para el flujo completo
13. ✅ El mecánico NO puede editar NADA del sistema, solo reportar

14. ✅ El mecánico NO puede ver su reporte pendiente y modificarlo (inmutable una vez enviado)
15. ✅ El historial del mecánico es de SOLO LECTURA de sus propios reportes

---

🚫 REGLAS UX CRÍTICAS - LO QUE EL MECÁNICO NO PUEDE HACER

En la UI del Mecánico NO debe existir:

Elemento	Razón
Botón "Editar" en ningún lado	El mecánico no edita, solo reporta
Botón "Eliminar" en ningún lado	El mecánico no puede borrar nada
Acceso a "Control de Mantenimiento"	Es módulo exclusivo de admin
Acceso a "Configuraciones"	Es módulo exclusivo de admin
Acceso a "Planes de Mantenimiento"	Es módulo exclusivo de admin
Ver costos/precios en inventario	Información sensible
Modificar datos de equipos	Solo admin
Cambiar estados de mantenimiento	Solo mediante aprobación de admin
Editar un reporte ya enviado	Una vez enviado es inmutable
Ver reportes de otros mecánicos	Solo ve los suyos

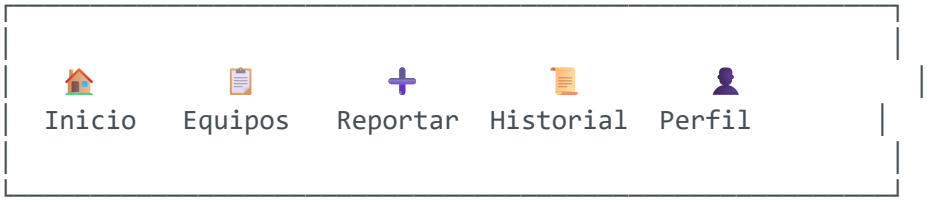
Flujo de "Corrección" cuando es Rechazado:

Cuando un reporte es rechazado, el mecánico NO lo edita. En su lugar:

1. Ve el feedback del admin
2. Presiona "Crear Nuevo Reporte"
3. Se abre un formulario NUEVO prellenado con los datos anteriores
4. Puede modificar lo necesario
5. Envía como un NUEVO reporte
6. El reporte rechazado queda en historial como referencia

Esto mantiene la trazabilidad completa.

📱 NAVEGACIÓN DEL MECÁNICO (BottomNav)



Tab	Pantalla	Descripción
🏠 Inicio	MechanicDashboard	Resumen personal, accesos rápidos
📋 Equipos	MechanicPendingList	Lista de equipos pendientes de mantenimiento
➕ Reportar	MechanicSubmissionForm	Formulario para crear nuevo reporte (puede preseleccionar equipo)
📜 Historial	MechanicHistory	Lista de todos sus reportes con estados
👤 Perfil	MechanicProfile	Datos del usuario, cerrar sesión

Rutas:

```
// Rutas del mecánico (protegidas con RequireRole(['mechanic']))
/mobile/mechanic                    // Dashboard
/mobile/mechanic/equipos            // Lista equipos pendientes
/mobile/mechanic/reportar           // Form nuevo reporte
/mobile/mechanic/reportar/:equipoId // Form con equipo preseleccionado
/mobile/mechanic/historial          // Lista de sus reportes
/mobile/mechanic/historial/:id      // Detalle de un reporte
/mobile/mechanic/perfil             // Perfil del usuario
```

🔔 SISTEMA DE NOTIFICACIONES

Notificaciones que recibe el ADMIN:

Evento	Título	Mensaje	Acción
Nuevo reporte	"Nuevo Reporte de Mantenimiento"	"[Mecánico] ha enviado un reporte para [Equipo]"	Ir a revisar
Reporte corregido	"Reporte Corregido"	"[Mecánico] ha enviado una corrección para [Equipo]"	Ir a revisar

Notificaciones que recibe el MECÁNICO:

Evento	Título	Mensaje	Acción
Aprobado	"Reporte Aprobado ✅"	"Tu reporte para [Equipo] ha sido aprobado e integrado al sistema"	Ver detalle
Rechazado	"Reporte Rechazado ❌"	"Tu reporte para [Equipo] necesita correcciones: [feedback]"	Crear corrección

Implementación:

```
// Trigger en Supabase cuando se crea submission
// Notifica a todos los usuarios con rol 'admin'
CREATE OR REPLACE FUNCTION notify_admins_new_submission()
RETURNS TRIGGER AS $$
BEGIN
  INSERT INTO notifications (user_id, type, title, message, payload)
  SELECT
    ur.user_id,
    'submission_received',
    'Nuevo Reporte de Mantenimiento',
    'Un mecánico ha enviado un nuevo reporte',
    jsonb_build_object('submission_id', NEW.id, 'equipo_id', NEW.equipo_id)
  FROM user_roles ur
  JOIN roles r ON ur.role_id = r.id
  WHERE r.name = 'admin';

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER on_submission_created
  AFTER INSERT ON maintenance_submissions
  FOR EACH ROW EXECUTE FUNCTION notify_admins_new_submission();
```

📊 MÉTRICAS DEL DASHBOARD DEL MECÁNICO

El dashboard del mecánico muestra SOLO métricas personales:

```
interface MechanicDashboardMetrics {
  // Mis reportes
  reportesPendientes: number;      // status = 'pending'
  reportesAprobados: number;       // status = 'approved' o 'integrated'
  reportesRechazados: number;      // status = 'rejected'
  reportesEsteMes: number;         // created_at en mes actual

  // Equipos
  equiposPendientes: number;       // Equipos con mantenimiento vencido/próximo
  equiposVencidos: number;         // Mantenimientos vencidos

  // NO mostrar:
  // - Métricas globales del sistema
  // - Información de otros mecánicos
  // - Costos o valores monetarios
}
```

Comandos de Desarrollo

```
# Aplicar migración
npx supabase db push

# Crear bucket de storage (si no existe)
npx supabase storage create submissions

# Correr en desarrollo
npm run dev

# Build y sync Android
npm run android:sync
```

```
# Probar en viewport pequeño
# Abrir DevTools > Toggle device > Custom: 240x561
```

Notas Adicionales

- Las imágenes deben comprimirse en cliente usando `browser-image-compression` o canvas nativo
- Considerar modo offline: guardar draft en localStorage si no hay conexión
- El mecánico NO puede editar una submission una vez enviada (inmutable)
- Añadir confirmación antes de enviar para evitar envíos accidentales
- Los badges de estado usan colores: pending=yellow, approved=green, rejected=red, integrated=blue
- El botón "Reportar" debe ser el más prominente en la UI del mecánico

---

🎨 ESPECIFICACIONES VISUALES

Colores de Estado:

```
/* Estados de submission */
.status-pending { @apply bg-yellow-100 text-yellow-800 border-yellow-300; }
.status-approved { @apply bg-green-100 text-green-800 border-green-300; }
.status-rejected { @apply bg-red-100 text-red-800 border-red-300; }
.status-integrated { @apply bg-blue-100 text-blue-800 border-blue-300; }
```

Iconos:

Estado	Icono	Color
Pendiente	🟡 o ⏳	Amarillo
Aprobado	✅	Verde
Rechazado	❌	Rojo
Integrado	🔗	Azul

Touch Targets (Mobile):

```
/* Mínimo 44x44px para todos los elementos interactivos */
.touch-target {
  min-height: 44px;
  min-width: 44px;
}

/* Botón principal sticky */
.sticky-button {
  @apply fixed bottom-0 left-0 right-0 p-4 bg-background border-t;
  padding-bottom: env(safe-area-inset-bottom, 16px);
}
```

📝 CHECKLIST DE IMPLEMENTACIÓN

Fase 1: Base de Datos (1 día)

- ☐ Crear migración SQL con tablas (`maintenance_submissions`, `submission_attachments`, `notifications`)
- ☐ Agregar columnas de trazabilidad a `mantenimientos` (`submission_id`, `realizado_por`, `aprobado_por`)
- ☐ Configurar RLS policies para mecánico y supervisor
- ☐ Crear funciones RPC (`approve_and_integrate_submission`, `reject_submission`)
- ☐ Crear triggers de notificaciones
- ☐ Crear bucket de storage `submissions`

Fase 2: UI Mecánico (2-3 días)

- ☐ `MechanicDashboard.tsx` — métricas personales
- ☐ `MechanicPendingList.tsx` — lista de equipos pendientes
- ☐ `MechanicSubmissionForm.tsx` — formulario de reporte
- ☐ `MechanicHistory.tsx` — historial de mis reportes
- ☐ `MechanicSubmissionDetail.tsx` — detalle con timeline
- ☐ `PhotoUploader.tsx` — subir fotos (cámara/archivo/pegar)

- ☐ `PartsInput.tsx` — agregar partes del inventario
- ☐ `MechanicBottomNav.tsx` — navegación móvil
- ☐ `MechanicSidebar.tsx` — navegación web/desktop
- ☐ Rutas: `/mobile/mechanic/*`

Fase 3: UI Admin - Aprobaciones (1-2 días)

- ☐ `SubmissionsList.tsx` — lista de submissions pendientes
- ☐ `SubmissionReview.tsx` — revisar submission con fotos y partes
- ☐ `ApproveModal.tsx` — modal de confirmación de aprobación
- ☐ `RejectModal.tsx` — modal con campo de motivo
- ☐ Integración con sistema de notificaciones
- ☐ Badge de notificaciones en header
- ☐ Redirigir a Control Mantenimiento post-integración

Fase 4: Supervisor (0.5 días)

- ☐ `ReadOnlyBanner.tsx` — banner "Vista de solo lectura"
- ☐ Modificar `ControlMantenimiento.tsx` para ocultar acciones si es supervisor
- ☐ Modificar `EquipmentCard.tsx` para ocultar editar/eliminar
- ☐ Verificar que no hay fugas de permisos en ninguna vista
- ☐ Limitar acceso a solo: Dashboard, Equipos, Plan/Estado, Historial, Reportes

Fase 5: Integraciones (1 día)

- ☐ Actualizar `useUserRoles.ts` con permisos granulares
- ☐ Actualizar `permissions.ts` con nuevos roles
- ☐ Agregar rutas protegidas en `App.tsx`
- ☐ Notificaciones en tiempo real con Supabase Realtime
- ☐ Compresión de imágenes en cliente

Fase 6: Testing (1 día)

- ☐ Test flujo completo mecánico→admin→integración
- ☐ Test responsive 240x561 (móvil pequeño)
- ☐ Test RLS policies (mecánico solo ve sus datos)

- ☐ Test notificaciones en tiempo real
- ☐ Test subida de fotos y partes
- ☐ Test supervisor no puede modificar nada

✅ RESUMEN DE COBERTURA DE REQUISITOS

#	Requisito Original	Estado	Sección
1	Mecánico ve historial (solo sus reportes)	✅	Pantalla 5
2	Mecánico ve equipos pendientes	✅	Pantalla 2
3	Mecánico ve inventario (para seleccionar partes)	✅	Modal de partes
4	Mecánico sube fotos (copiar/pegar/archivo)	✅	PhotoUploader
5	Mecánico reporta partes/repuestos/filtros usados	✅	PartsInput
6	Admin recibe notificación de nuevos reportes	✅	Trigger SQL + Notificaciones
7	Admin aprueba e integra con 1 click	✅	SubmissionReview + RPC
8	Campos prellenados en Control Mantenimiento	✅	approve_and_integrate_submission
9	Funciona en Android y Web	✅	Capacitor + responsive
10	Historial registra todo	✅	Trazabilidad con submission_id
11	Supervisor ve dashboard (read-only)	✅	Permisos supervisor
12	Supervisor ve lista de equipos	✅	Permisos supervisor

#	Requisito Original	Estado	Sección
13	Supervisor ve Plan y Estado en Control	✅	Wireframe supervisor
14	Supervisor NO puede editar/eliminar	✅	RLS + UI condicional
15	Supervisor NO aprueba submissions	✅	Solo admin aprueba

FIN DEL PROMPT

Usa este documento completo para implementar la funcionalidad. Si necesitas dividirlo en partes más pequeñas, empieza por:

1. Migración SQL y políticas RLS
2. Componentes del mecánico (mobile)
3. Vista de admin para aprobar
4. Ajustes de permisos para supervisor