Análisis Completo del Proyecto `import-dash`

Fecha: 2025-12-04 Autor: Informe generado automáticamente y revisado manualmente

---

Resumen ejecutivo

Este documento ofrece un análisis detallado del estado actual del repositorio `import-dash`, problemas detectados, causas raíz, evidencia técnica, y un plan de acciones priorizadas (con comandos, archivos a modificar y estimaciones de tiempo). El objetivo

es dejar el proyecto en un estado reproducible para desarrollo, correcto para builds nativos (Electron y Android) y con mejora de rendimiento y experiencia móvil — sobre todo en pantallas ultra-pequeñas (ej. 240×561).

Resumen rápido de hallazgos clave:

- La aplicación web compila correctamente con Vite (`npm run build`).
- El servidor de desarrollo se puede ejecutar (se configuraron puertos alternativos), pero hay conflictos iniciales con puertos ocupados.
- Integración móvil con Capacitor requiere regeneración del proyecto Android en esta máquina — ya se regeneró con `npx cap add android` y sincronizó correctamente.
- Packaging de Electron presenta fragilidad: instalación de `electron-builder` falló por un script postinstall (dependencia `electron-winstaller` que busca binarios 7z-ia32.exe). Se evitó temporalmente con `--ignore-scripts` y se instaló solo `electron` para dev.
- El bundle contiene chunks muy grandes (> 500 KB) que impactan rendimiento; requiere code-splitting y análisis de dependencias.
- Problemas de responsive en Dashboard móvil: en pantallas muy pequeñas los componentes se recortan. Requiere reglas CSS y revisiones de layout.
- Falta de automatización CI/CD para builds nativos y releases; documentos y procedimientos incompletos (keystore Android, firmas Windows para Electron).

Cómo auditamos (comandos y archivos revisados)

Comandos ejecutados durante el análisis:

- `npm install`
- `npm run build` (Vite build)
- `npm run dev` (varias veces, probando puertos alternativos)
- `npx cap sync android`
- `npx cap add android` (regenerar plataforma Android)
- `npx cap open android` (abrir Android Studio)
- `git pull origin main`, `git stash`, `git stash pop` (gestión de cambios locales)

Archivos/áreas revisadas manualmente:

- `package.json`
- `tailwind.config.ts`
- `src/pages/mobile/DashboardMobile.tsx` (conflictos y responsive)
- `src/components/mobile/*` (MobileLayout, MobileCard, otros)
- `capacitor.config.ts` y `android/app/src/main/assets/capacitor.config.json`
- `android/` (se regeneró la plataforma y comprobó presencia de `gradle`, `gradlew` y `app/`)
- `dist/` generado por `vite build`

Logs y outputs consultados: salida de `npm install`, `npm run build`, `npx cap sync android` y `npx cap add android`.

Hallazgos detallados y evidencia

1. Build web

- `vite build`: OK — se generan `dist/` y activos. Evidencia: salida de build con lista de chunks.
- Advertencias: algunos chunks mayores a 500 KB (ej. `index-*.js` ~553 KB). Esto sugiere que librerías grandes (jspdf, html2canvas, etc.) están en bundle principal.

Impacto: mayores tiempos de carga y uso de recursos, sobre todo en móviles y WebView (capacitor).

2. Dependencias y problemas en instalación

- Error inicial al ejecutar `npm install electron electron-builder electron-updater --save-dev`: fallo por script postinstall en `electron-winstaller` buscando `vendor/7z-ia32.exe` (ENOENT).
- Causa probable: combinación de versiones de `node` (v22) y paquetes que esperan artefactos o scripts no compatibles; `electron-winstaller` es antiguo y frágil.
- Solución temporal aplicada: `npm install ... --ignore-scripts` y posteriormente `npm install electron --save-dev`.

Impacto: Builds de instalador (nsis / nsis-web) pueden fallar en CI o en máquinas sin los binarios adecuados.

3. Capacitor / Android

- `android/` inicialmente no tenía Gradle (error). Se eliminó y se regeneró con `npx cap add android`.
- `npx cap sync android` copió `dist/` a `android/app/src/main/assets/public` y creó `capacitor.config.json`.

Impacto: después de regenerar, el proyecto Android tiene archivos de Gradle listos. Sin embargo, para generar APK release será necesario configurar keystore y posiblemente ajustar `gradle.properties` y versiones de AGP/JDK.

4. Responsive (Dashboard móvil)

- Observación del problema: en pantallas ultra-pequeñas (240×561) ciertos `Cards` y contenedores se recortan; texto y botones no se adaptan.
- Causas frecuentes detectadas en código:
  - Contenedores con anchuras mínimas o sin `min-w-0` en flex/cols.
  - Uso de paddings/gaps fijos para tamaños pequeños.
  - Iconos/elementos con tamaño fijo que no escalan.
  - Falta de breakpoints por debajo de 320px.

Impacto: mala UX móvil en dispositivos con anchuras pequeñas o WebViews (Android) pequeñas.

5. CI/CD y Releases

- No existe pipeline robusto (aunque hay archivos no trackeados en `.github/workflows` detectados en el workspace). No se observó un workflow completo que realice builds de Electron o Android y suba artefactos.

Impacto: releases manuales, riesgo de errores humanos, builds no reproducibles.

6. Seguridad y signing

- No hay keystore ni instrucciones para generar el signing key de Android en repo. `capacitor.config.ts` tiene placeholders para `keystorePath`.
- Para Windows, no hay configuración de firma de código (certificado) en `package.json` ni en `electron-builder` settings.

Impacto: no se puede publicar versiones firmadas sin añadir secretos y configuraciones de firma en CI.

7. Testing y calidad

- No hay test suites visibles (Vitest/Jest/E2E) en el repo.
- Linter existe (`eslint`) pero no está integrado en CI.

Impacto: regresiones no detectadas, menor cobertura de calidad.

8. Dependencias y vulnerabilidades

- `npm audit` reportó vulnerabilidades moderadas; algunas dependencias están marcadas como deprecated.

Impacto: riesgo de seguridad y mantenimiento futuro.

Recomendaciones y plan de acción (priorizado)

A continuación propongo un plan en fases con acciones concretas, comandos y estimaciones de tiempo. Las estimaciones son aproximadas para un desarrollador con acceso al repositorio y privilegios locales.

Fase 0 — Preparación (0.5 días)

- Definir y documentar la versión de Node recomendada: usar Node LTS 18.x o 20.x.
  - Añadir `.nvmrc` con `18.20.1` o similar.
  - Añadir `engines` en `package.json`.

Comandos:

```
# Añadir .nvmrc
echo 18.20.1 > .nvmrc
# En package.json (manualmente) añadir:
# "engines": { "node": "18.x || 20.x" }
```

Motivo: reproducibilidad y compatibilidad con `electron-builder` y herramientas nativas.

Prioridad Alta (1–3 días)

1. Fix: Packaging Electron reproducible

- Recomendación: forzar Node LTS en dev y CI. Evitar `--ignore-scripts` salvo como workaround temporal.
- Pasos:
  - Instalar `nvm-windows` en máquinas Windows del equipo. Ejecutar `nvm install 18.20.1` y `nvm use 18.20.1`.
  - Ejecutar `npm ci` y `npm run electron:build`.
- Alternativa si se necesita seguir con Node 22: instalar `7z` en la máquina (o proveer los binarios requeridos) y repetir install.

Riesgos: cambiar Node localmente puede afectar otros proyectos; comunicar al equipo.

2. Fix: Responsive crítico para Dashboard (240×561)

- Implementar cambios en `src/pages/mobile/DashboardMobile.tsx`, `src/components/mobile/MobileLayout.tsx`, y `src/components/mobile/MobileCard.tsx`:
  - Añadir `min-w-0` en contenedores flex/cols.
  - Añadir `overflow-hidden` y `truncate` en texto que puede crecer.
  - Añadir breakpoints ultra-small en `tailwind.config.ts` (ej. `xxs: '320px', '3xs': '240px'`).
  - Reducir paddings, gaps y tamaños de fuente en esos breakpoints.
- Ejemplos CSS/utility: `className="min-w-0 overflow-hidden truncate"`, `xxs:text-xs`, `3xs:p-1`.

Comandos de prueba:

```
npx vite --port 5173
# Abrir http://localhost:5173/ y en DevTools simular 240x561
```

3. Fix: Android reproducible y signing (preparación)

- Añadir documento `docs/ANDROID_RELEASE.md` con pasos para generar keystore y configurar `gradle.properties` y GitHub Secrets.
- Comandos (generar keystore):

```
keytool -genkey -v -keystore alito-release.keystore -alias alito_key -keyalg RSA -
keysize 2048 -validity 10000
```

- Subir `ALITO_KEYSTORE` (base64 o archivo) y variables `KEYSTORE_PASSWORD`, `KEY_ALIAS`, `KEY_PASSWORD` a secrets de CI.

4. Quick: `npm audit` and fixes

- Ejecutar `npm audit`; aplicar `npm audit fix` y revisar `npm audit fix --force` solo tras pruebas.

Prioridad Media (3–7 días)

1. Bundle analysis y code-splitting

- Generar reporte de bundle con plugin visualizer o `source-map-explorer`.
- Identificar paquetes pesados y cargarlos dinámicamente con `import()` en rutas donde se usan (p. ej. export PDF/Reportes, jspdf, html2canvas, grafías pesadas).

Comandos ejemplo:

```
npm run build
npx rollup --config node_modules/vite/dist/node/chunks/dep-build.js # (o usar
plugin viz)
# O usar vite-plugin-visualizer
```

2. Añadir CI minimal (GitHub Actions)

- Pipeline mínimo: checkout, setup-node, `npm ci`, `npm run build`, `npx cap sync android` (opcional), artefactos (dist).
- Luego extender para Electron y Android signing.

3. Tests: añadir Vitest y una suite básica para lógica crítica.

Prioridad Baja / Largo plazo (2–6 semanas)

1. Full CI multi-platform: Web + Electron (nsis) + Android (assembleRelease), subir a GitHub Releases.
2. Integrar Sentry y logging centralizado.
3. Automatizar dependabot y escaneo de seguridad programado.
4. Mejoras UX y performance continuas (Lighthouse, A11y)

Implementaciones concretas (snippets, archivos y comandos)

A continuación se listan cambios concretos que puedes aplicar inmediatamente. Si quieres, puedo aplicar estos cambios en el repositorio.

1) Añadir `.nvmrc` y `engines`

Archivo: `.nvmrc`

```
18.20.1
```

Agregar en `package.json` (ya recomendado):

```
"engines": {
  "node": "^18 || ^20"
}
```

2) Tailwind: añadir breakpoints ultra-pequeños

Archivo: `tailwind.config.ts` (ejemplo para agregar `xxs` y `3xs`):

```
export default {
  theme: {
```

```
    extend: {},
    screens: {
      '3xs': '240px',
      'xxs': '320px',
      'xs': '375px',
      // ...rest
    }
  }
}
```

3) Ejemplo de cambio en `MobileLayout` o `DashboardMobile`

- Asegurar contenedores flex con `min-w-0` y `overflow-hidden`:

```
<div className="flex flex-col min-w-0 overflow-hidden">
  <h3 className="truncate">Titulo largo que no debe forzar el ancho</h3>
</div>
```

- Ajustes de tipografía y padding para `3xs`:

```
<div className="p-3 3xs:p-1 xxs:p-2">
  <p className="text-sm 3xs:text-xs">...</p>
</div>
```

4) Code-splitting con rutas dinámicas (ejemplo React Router)

```
// Antes
import { ReportPage } from '@/pages/ReportPage'
// Después
const ReportPage = React.lazy(() => import('@/pages/ReportPage'))

// Uso con Suspense
<Suspense fallback={<Spinner/>}>
  <Route path="/report" element={<ReportPage/>} />
</Suspense>
```

5) GitHub Actions - pipeline de ejemplo (workflow básico)

Archivo: `.github/workflows/ci.yml`

```yaml
name: CI
on: [push, pull_request]
jobs:
  build-web:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Use Node
        uses: actions/setup-node@v4
        with:
          node-version: '18'
      - run: npm ci
      - run: npm run build
      - uses: actions/upload-artifact@v4
        with:
          name: dist
          path: dist
```

6) Android signing steps (README / `docs/ANDROID_RELEASE.md`)

Contenido mínimo:

- `keytool` command to generate keystore
- How to set `gradle.properties` with `MYAPP_UPLOAD_STORE_FILE`, `MYAPP_UPLOAD_KEY_ALIAS`, etc.
- How to configure GitHub Actions secrets and use signing in `gradlew assembleRelease`.

Estimaciones de tiempo (orientativas)

- Preparación (.nvmrc, engines, README): 0.5 día
- Fix responsive crítico (Dashboard 240×561): 0.5–1 día
- Electron packaging reproducible (cambio Node y reintentos): 0.5–1 día

- Android signing doc y prueba de `assembleDebug`: 0.5–1 día
- Bundle analysis y code-splitting (primera pasada): 1–2 días
- CI básico + tests iniciales: 2–4 días

Riesgos y mitigaciones

- Cambiar versión de Node puede afectar otros proyectos: mitigación -> documentar `.nvmrc` y usar `nvm`.
- Actualizar dependencias mayores puede romper código: mitigación -> usar `npm audit fix` solo para parche/semver compatible y probar.
- Firma y keys en CI: usar GitHub Secrets y no subir keystore al repo.

Checklists (qué hacer ahora)

- ☐ Añadir `.nvmrc` y documentar Node LTS en README
- ☐ Corregir responsive en `DashboardMobile` (aplicar `min-w-0`, `truncate`, breakpoints nuevos)
- ☐ Preparar `docs/ANDROID_RELEASE.md` y generar keystore localmente
- ☐ Agregar workflow de CI (web) y artefactos
- ☐ Analizar bundle y mover librerías pesadas a imports dinámicos
- ☐ Configurar Sentry (opcional en etapa siguiente)

Apéndices

Comandos útiles (resumen)

```
# Desarrollo
npm ci
npm run dev
npx vite --port 5173

# Build web
npm run build
```

```
# Capacitor Android
npm run android:sync
npm run android:open
npm run android:build
# o desde android/
cd android; .\gradlew assembleDebug

# Electron (recomendado en Node LTS)
nvm use 18
npm ci
npm run electron:build

# Audit
npm audit
npm audit fix
```

Ejemplo: `.nvmrc`

```
18.20.1
```

Si quieres que lo añada como archivo en el repositorio y haga commit de inmediato lo puedo hacer. También puedo aplicar ya los cambios en `DashboardMobile.tsx` y en `tailwind.config.ts` para arreglar el responsive 240×561 — dime si quieres que empiece por eso (lo implemento y pruebo con `npx vite --port 5173`).

Fin del documento.