Especificación de Software — Control de Gastos, Facturación y Materiales de Obra (Tamivar)

Fecha: 29/09/2025 | Autor: Equipo de desarrollo | Estado: Propuesta inicial (v0.1)

1. Resumen ejecutivo

La empresa Tamivar enfrenta desorganización y falta de visibilidad en tres frentes clave: control de gastos, seguimiento de facturación y gestión de materiales de obra. El objetivo es implementar una aplicación de escritorio (Electron) con capacidades PWA (funciona offline) y base de datos local embebida (SQLite), que estructure procesos, centralice datos, facilite auditoría y reporte, y mejore la toma de decisiones.

2. Problemática

- No existe un sistema adecuado para el control de gastos: registros manuales, incompletos y poco confiables.
- Falta de claridad del origen, propósito y justificantes de cada desembolso.
- En facturación no hay mecanismo de seguimiento: difícil distinguir facturas emitidas vs pendientes; retrasos y omisiones.
- Gestión de materiales de obra sin registro sistemático: duplicidad de compras, desperdicio y pérdidas.
- Inconsistencias por la falta de integración entre áreas (administrativa/operativa) y demoras en comunicar información.
- Riesgos financieros/administrativos, afectación a la credibilidad, imagen poco profesional y menor competitividad.

3. Solución propuesta (síntesis)

Desarrollar un sistema integral local con: - App de escritorio (Electron) con UI moderna (React + TypeScript) y capacidades PWA (Service Worker, cache, offline-first). - Base de datos SQLite embebida, accesible solo a través de la app. - Módulos: Gastos, Facturación, Materiales, Proveedores, Proyectos/Obras, Reportes y Auditoría. - Flujos con evidencia

(fotos/PDF de comprobantes), estatus y aprobaciones. - Roles/Permisos, bitácora de cambios, respaldos cifrados y controles de acceso.

4. Alcance

- Versión inicial (MVP) enfocada en: captura y control de gastos con justificantes, seguimiento básico de facturas, inventario de materiales con movimientos, reportes operativos.
- Datos locales en el equipo del usuario. Sin sincronización multi-dispositivo en v0.1 (extensible en roadmap).
- Sin timbrado fiscal CFDI en v0.1; se contemplan campos para registrar folios/estatus y anexar XML/PDF de terceros.

5. Objetivos

- Reducir errores y tiempos en registros administrativos (>50%).
- Aumentar la trazabilidad y auditoría (100% de operaciones con responsable, fecha y evidencia).
- Mejorar visibilidad de estatus de facturas y consumo de materiales.
- Proveer reportes para decisiones de compra y control presupuestal por obra.

6. Usuarios y roles

- Admin: configuración, usuarios, permisos, catálogos, respaldo/restauración.
- Contabilidad: gastos, facturas, reportes contables, conciliaciones.
- Gestoría: validación documental, seguimiento de trámites, revisión de facturas.
- Obra/Almacén: solicitud de compras, entradas/salidas de materiales, inventario.
- Dirección: panel ejecutivo y reportes.

7. Requisitos funcionales

7.1 Gastos

- Capturar gasto con: fecha, obra/proyecto, categoría, proveedor, monto, forma de pago, centro de costo, descripción.
- Adjuntar comprobantes (foto/PDF/XML), validación de obligatoriedad según categoría.
- Flujo de estatus: Borrador → En revisión → Aprobado/Rechazado → Reembolsado/Contabilizado.
- Políticas: topes por categoría, obligatoriedad de justificación, validación de duplicados por folio/importe/fecha.
- Reportes: gastos por obra/categoría/periodo, vs presupuesto, gastos pendientes de aprobación/reembolso.

7.2 Facturación

- Registro de facturas: cliente/proveedor, serie/folio, fecha emisión, subtotal/IVA/total, UUID (si aplica), estatus (Pendiente/Emitida/Pagada/Cancelada), vencimiento.
- Adjuntos: XML/PDF (cuando exista). Validación de duplicados por UUID/folio.
- Seguimiento: listado por estatus, alertas de vencimiento (notificaciones locales).
- Relación con gastos/obras para trazabilidad.

7.3 Materiales de obra (Inventario)

- Catálogo de materiales (SKU, descripción, unidad, costo estándar, categoría).
- Entradas: compras/ajustes/iniciales; Salidas: consumo en obra, devoluciones.
- Kardex por material y por obra. Existencias por almacén/obra.
- Solicitudes de material y órdenes de compra simples (MVP).

7.4 Proyectos/Obras y Proveedores

- Obras: datos generales, responsables, presupuesto base, fechas, estado (Activa/Cerrada).

- Proveedores: razón social, RFC (opcional), contactos, condiciones de pago, documentos.

7.5 Reportes y paneles

- Gastos por obra/categoría/mes; facturas por estatus; consumo de materiales vs presupuesto.
- Exportación a CSV/PDF. Filtros y segmentación por fechas/obra/usuario.

7.6 Auditoría y bitácora

- Registro de operaciones (quién, qué, cuándo, antes/después para campos clave).
- Trazabilidad de adjuntos (hash/huella digital) para detectar cambios.

8. Requisitos no funcionales

- Rendimiento: operaciones comunes < 200 ms; listados paginados/virtualizados.
- Offline-first total; arranque sin red. Archivos adjuntos almacenados localmente.
- Seguridad: RBAC, cifrado de respaldos, hashes de contraseñas (Argon2/bcrypt), bloqueo de sesión.
- Confiabilidad: respaldo manual/automático programable; restauración validada.
- Usabilidad: interfaz en español, accesible (WCAG AA básico), atajos de teclado.
- Portabilidad: macOS inicialmente; empaquetado con electron-builder.

9. Arquitectura y tecnologías

- Contenedor: Electron (main process) + Renderer (React + TypeScript + Vite).
- PWA: Service Worker para cache de UI; modo offline; actualización controlada.
- Datos: SQLite (better-sqlite3) con migraciones (Kysely/Drizzle). Adjuntos en filesystem app-data.
- Capa de acceso a datos: Kysely (TS) con driver better-sqlite3 para tipado fuerte.
- Estado y UI: React Query para datos; Zustand/Redux para estado UI; MUI/Tailwind para componentes.

- IPC/Seguridad: Context Isolation, preload seguro; API interna (Electron IPC) como fachada.
- Empaquetado: electron-builder (dmg/pkg para macOS). Auto-update opcional (siguiente fase).

Diagramas de arquitectura y UML

Diagrama de arquitectura lógica (alto nivel):

```
flowchart LR
  U[Usuario] --> UI[Renderer Electron (React + TS)]
  UI -->|IPC seguro (preload)| IPC[Canales IPC]
  IPC --> MAIN[Proceso Main (Electron)]
  MAIN --> DAL[Capa de Datos (Kysely + better-sqlite3)]
  DAL --> DB[(SQLite embebido)]
  MAIN --> FS[Almacenamiento de adjuntos (Filesystem App-Data)]
  UI --> SW[Service Worker]
  SW --> CACHE[Cache UI/estáticos]
  MAIN --> BACKUP[Backups cifrados AES-256]
  UI --> AUTH[RBAC/Sesión]
```

Diagrama de componentes (módulos principales):

```
flowchart TB
  subgraph UI[UI Renderer]
    G[Gastos]
    F[Facturación]
    M[Materiales]
    P[Proyectos/Obras]
    V[Proveedores]
    R[Reportes]
    A[Auditoría/Backups]
  end
  UI --> API[API Interna (IPC)]
  API --> SVCG[Servicio Gastos]
  API --> SVCF[Servicio Facturación]
  API --> SVCM[Servicio Materiales]
  API --> SVCCAT[Servicios Catálogos]
  API --> SVCSEC[Servicio Seguridad]
  SVCG & SVCF & SVCM & SVCCAT & SVCSEC --> DAL2[(DAO/Migraciones)]
```

```
    DAL2 --> SQL[(SQLite)]
    SVCG & SVCF --> FILES[(Adjuntos)]
```

Secuencia: Captura de gasto con adjuntos y aprobación

```
sequenceDiagram
    participant U as Usuario
    participant UI as UI (Renderer)
    participant P as Preload/IPC
    participant S as Servicio Gastos (Main)
    participant DB as SQLite
    participant FS as Filesystem

    U->>UI: Completa formulario + adjunta archivos
    UI->>P: submitExpense(data, files)
    P->>S: submitExpense(data, files)
    S->>DB: BEGIN; INSERT expense...
    S->>FS: Guardar adjuntos (ruta app-data)
    S->>DB: INSERT expense_attachments + checksum
    DB-->>S: COMMIT OK
    S-->>P: Respuesta {id, status: Borrador}
    P-->>UI: Mostrar confirmación
    UI->>P: requestApproval(expenseId)
    P->>S: approve(expenseId)
    S->>DB: UPDATE status=Aprobado; log auditoría
    DB-->>S: OK
    S-->>P: OK
    P-->>UI: Estado actualizado
```

Secuencia: Seguimiento de factura y alerta de vencimiento

```
sequenceDiagram
    participant U as Usuario
    participant UI as UI
    participant P as IPC
    participant S as Servicio Facturas
    participant DB as SQLite
    participant OS as Notificaciones OS

    U->>UI: Registra factura (folio/UUID, vencimiento)
    UI->>P: createInvoice(data)
```

```
    P->>S: createInvoice(data)
    S->>DB: INSERT invoices
    S-->>P: OK
    P-->>UI: Mostrar en listado (Pendiente)
    Note over S: Job local revisa vencimientos
    S->>DB: SELECT invoices vencidas/próximas
    S->>OS: Enviar notificación local
```

Secuencia: Movimiento de materiales (entrada/salida)

```
sequenceDiagram
    participant U as Usuario Almacén
    participant UI as UI
    participant P as IPC
    participant S as Servicio Inventario
    participant DB as SQLite

    U->>UI: Registrar entrada (compra)
    UI->>P: addStock(material, qty, cost)
    P->>S: addStock(...)
    S->>DB: INSERT stock_moves(IN, qty, cost)
    S->>DB: UPDATE stock (existencia, costo promedio)
    S-->>P: OK
    P-->>UI: Actualizar existencias
    U->>UI: Registrar salida a obra
    UI->>P: consume(material, obra, qty)
    P->>S: consume(...)
    S->>DB: INSERT stock_moves(OUT, qty)
    S->>DB: UPDATE stock (existencia)
    S-->>P: OK
    P-->>UI: Kardex actualizado
```

10. Componentes principales (módulos)

- Autenticación y usuarios: login local, gestión de roles/permisos.
- Gastos: formularios, visores de comprobantes, flujo de aprobación.
- Facturación: registro/seguimiento, alertas de vencimiento.
- Inventario: catálogo, movimientos, kardex, existencias.

- Proyectos/Obras: alta/edición, presupuesto, estado.
- Proveedores: alta/edición, adjuntos, scoring básico (opcional).
- Reportes: constructor de filtros y exportación.
- Auditoría y respaldos: ver logs, crear/restaurar backup cifrado.

11. Modelo de datos (esquema propuesto SQLite)

Nota: claves primarias como id INTEGER AUTOINCREMENT; campos created_at/updated_at en todos los catálogos.

- users: id, username, password_hash, role_id, display_name, last_login_at, is_active
- roles: id, name, permissions (json)
- projects (obras): id, code, name, budget_amount, start_date, end_date, status
- vendors (proveedores): id, rfc, name, contact, email, phone, notes
- expense_categories: id, name, policy_limit, require_receipt (bool)
- expenses: id, project_id, category_id, vendor_id, amount, currency, date, payment_method, description, status, created_by
- expense_attachments: id, expense_id, path, mime_type, checksum, size
- invoices: id, project_id, vendor_id_or_client_id, type (AP/AR), series, folio, uuid, issue_date, due_date, subtotal, tax, total, status
- invoice_attachments: id, invoice_id, path, mime_type, checksum
- materials: id, sku, name, unit, std_cost, category
- warehouses (opcional simple): id, name
- stock: id, material_id, warehouse_id, qty
- stock_moves: id, material_id, project_id, warehouse_id, type (IN/OUT/ADJ), qty, unit_cost, date, reference
- approvals (opcional): id, entity (expense/invoice), entity_id, action, by_user, at

- audit_log: id, entity, entity_id, action, by_user, at, before (json), after (json)
- settings: key, value

Índices sugeridos: por uuid, series+folio, project_id, date, material_id.

12. Flujos clave (UML textual)

- Captura de gasto: Usuario crea → adjunta comprobante → valida reglas → envía a revisión → aprobar/rechazar → (si aprobado) marcar contabilizado/reembolsado.
- Factura: Registrar datos básicos → adjuntar PDF/XML → estatus y vencimiento → alertas → marcar pagada/cancelada.
- Materiales: Entrada por compra → aumenta stock → salida por consumo en obra → kardex y costo promedio.

13. Seguridad

- Autenticación local con almacenamiento de hash Argon2/bcrypt y bloqueo por intentos.
- RBAC por roles (Admin/Contabilidad/Gestoría/Obra/Dirección) y permisos granularizados por módulo/acción.
- Aislamiento de contexto en Electron, deshabilitar nodeIntegration en renderer; whitelisting de canales IPC.
- Cifrado de respaldos (AES-256) con contraseña; validación de integridad (checksum).
- Opción de base cifrada con SQLCipher en roadmap; en v0.1, cifrado de archivos de adjuntos en repositorio local.
- Sanitización de entradas y validación estricta; protección contra inyección SQL (Kysely + parámetros).
- Auditoría inmutable (append-only) y hash de adjuntos.

14. Privacidad y cumplimiento

- Datos locales en el dispositivo; sin telemetría por defecto.

- Política de retención configurable (p. ej., 5 años para documentos fiscales almacenados por referencia).
- Cumplimiento fiscal: sin timbrado en v0.1; soporta registro de folios/UUID y archivos oficiales.

15. Experiencia de usuario

- Diseño responsive dentro de la ventana Electron; atajos (nuevo gasto: ⌘N, buscar: ⌘F).
- Campos y validaciones guiadas; estados visibles con etiquetas (chips) y filtros guardados.
- Soporte de arrastrar/soltar para adjuntos; visor embebido de PDF e imágenes.

16. Instalación y operación

macOS

- Distribución .dmg/.pkg firmada; almacenamiento en ~/Library/Application Support/Tamivar/.
- Copias de seguridad manuales/automáticas programables a directorio elegido.
- Herramienta de mantenimiento: reparar DB (VACUUM), verificar integridad de adjuntos.

Windows

- Sistemas soportados: Windows 10/11 (x64).
- Requisitos de build: Node.js LTS, Python 3.x, Visual Studio Build Tools (C++), para compatibilidad con módulos nativos (better-sqlite3/node-gyp).
- Empaquetado con electron-builder: targets nsis (instalador) y/o portable.
- Almacenamiento de datos: %APPDATA%/Tamivar/ para DB y configuración. Respaldos sugeridos en Documents/Tamivar/Backups.
- Notificaciones locales integradas con Windows; atajos usan Ctrl (p. ej., Ctrl+N) en lugar de ⌘.

- Seguridad opcional: cifrado de secretos con DPAPI (vía librerías como keytar) para proteger credenciales locales; backups cifrados (AES-256) con contraseña.
- Consideraciones: antivirus/copia en la nube pueden bloquear la DB; excluir %APPDATA%/Tamivar/ o usar locking amigable. Evitar rutas de red.
- Firma de código recomendada para SmartScreen.

17. Plan de pruebas

- Unitarias: reglas de validación (duplicados, límites), cálculos (totales, kardex).
- Integración: flujos de gastos/facturas/materiales, adjuntos y auditoría.
- UI: pruebas de smoke y regresión visual (basics) en las vistas principales.
- Aceptación: casos de uso por rol (checklist) y criterios de éxito por módulo.

17.1 Matriz de casos de prueba (MVP)

- TC-EXP-001: Prevenir gastos duplicados
    - Precondición: Existe gasto con folio F123 por $1,000 el 2025-09-01.
    - Pasos: Capturar otro gasto con mismo folio/importe/fecha.
    - Resultado esperado: Validación bloquea y muestra mensaje; no se inserta registro.
- TC-EXP-002: Flujo de aprobación de gasto
    - Pasos: Crear gasto con adjunto obligatorio; enviar a revisión; aprobar.
    - Resultado: Estatus cambia Borrador→En revisión→Aprobado; se registra en auditoría.
- TC-INV-001: Registro de factura con UUID duplicado
    - Pasos: Registrar factura con UUID X; intentar registrar otra con mismo UUID.
    - Resultado: Rechazo por duplicado; solo una en DB.
- TC-INV-002: Alerta de factura próxima a vencer
    - Precondición: Factura con vencimiento +2 días.

  - Pasos: Ejecutar job de alertas.
  - Resultado: Notificación local visible y bandera en UI.
- TC-STK-001: Entrada de material actualiza existencias
  - Pasos: Agregar entrada 100 u.; consultar existencias/kardex.
  - Resultado: Existencia +100; movimiento IN en kardex.
- TC-STK-002: Salida por consumo en obra
  - Pasos: Consumir 30 u. para Obra A.
  - Resultado: Existencia -30; movimiento OUT asociado a Obra A.
- TC-AUD-001: Bitácora registra cambios críticos
  - Pasos: Editar monto de gasto aprobado.
  - Resultado: Registro en audit_log con before/after y usuario.
- TC-SEC-001: RBAC restringe acciones
  - Pasos: Usuario Obra intenta aprobar gasto.
  - Resultado: Operación denegada; UI deshabilita acción.

17.2 Estrategia y herramientas

- Unitarias/Integración: Vitest/Jest (TS) + pruebas sobre capa de datos (Kysely con SQLite en modo test).
- E2E/UI: Playwright (perfiles para macOS/Windows; atajos específicos por OS).
- Cobertura objetivo: ≥80% en reglas de negocio y servicios; smoke en pantallas clave.

17.3 Datos y fixtures

- Semillas para: usuarios por rol, 2 obras, 3 proveedores, 5 materiales.
- Adjuntos de prueba livianos (PDF/imagen) con checksums conocidos.

17.4 Carga/Resiliencia

- Escenarios con 50k gastos, 10k facturas, 100k movimientos de inventario.

- Validar paginación/virtualización y tiempos de consulta (<500 ms consultas indexadas).

18. Riesgos y mitigaciones

- Pérdida de equipo → Respaldos programables cifrados.
- Corrupción de DB por cierre abrupto → Transacciones + backup incremental.
- Complejidad fiscal (CFDI) → Postergar timbrado a fase 2 con proveedor especializado.
- Carga de adjuntos pesada → Límite de tamaño, compresión/thumbnail.

19. Roadmap sugerido

- Fase 0: Diseño detallado, esquema DB y prototipo UI (2-3 semanas).
- Fase 1 (MVP 6-8 semanas): Gastos + Proveedores + Obras + Reportes básicos + Auditoría + Respaldos.
- Fase 2: Inventario de materiales completo (kardex, almacenes), alertas facturas.
- Fase 3: Auto-update, cifrado SQLCipher, importaciones/exportaciones avanzadas.
- Fase 4: Integración CFDI (consulta/validación), multi-equipo con sincronización.

20. Criterios de aceptación (MVP)

- Registrar, aprobar y reportar gastos con adjuntos, sin errores críticos.
- Ver reportes por obra/categoría/periodo y exportar CSV.
- Registrar y seguir estatus de facturas, con alertas de vencimiento.
- Gestionar materiales con entradas/salidas y ver kardex por obra/material.
- Operar completamente offline; respaldar y restaurar datos con verificación.

21. Anexos

- Glosario: AP (cuentas por pagar), AR (cuentas por cobrar), Kardex (movimientos de inventario), RBAC (control de acceso basado en roles).

- Referencia del documento origen: “FORMATODELPROYECTOMARIA.md” (secciones Problema/Justificación).