Propuesta modernización TR-Contabilidad

El plan propone una migración gradual hacia una arquitectura moderna en tecnología .NET 8/9, diseñada para ser segura, flexible, funcionar on premise pero preparada para la nube y sin interrumpir la operación actual de los clientes. La estrategia será iterativa y controlada, reduciendo riesgos técnicos y humanos.

Además, la modernización incorpora buenas prácticas como arquitectura desacoplada por features, integración y entrega continua (CI/CD), pruebas automatizadas y mecanismos de actualización silenciosa. Esto permitirá optimizar los costos de soporte, mejorar la experiencia del usuario y garantizar la evolución tecnológica de la solución.

Situación actual

Aspecto	Descripción
Tecnología	Aplicación de escritorio desarrollada en **Visual Basic 6**. Código fuente presenta aproximadamente 250 archivos de formularios (`.frm`), componentes y dependencias. Se distribuye como instalador y requiere un ejecutable adicional para las actualizaciones.
Módulos	**Módulo 1 (Administrador):** gestiona empresas y datos transversales. **Módulo 2 (Cliente):** gestiona cuentas corrientes, activo fijo, libros legales e impuestos del artículo 14D.
Base de datos	Bases de datos locales en **Microsoft Access** o **SQL Server Express** instaladas **on-premise**.
Usuarios	Algunos utilizan versiones antiguas de Windows lo que limita la capacidad de actualizar componentes.
Problemas	Dependencia de VB6 (plataforma obsoleta sin soporte), baja compatibilidad con ambientes actuales, dificultades para instalar en sistemas modernos, mantenimiento costoso, dependencia de actualizador externo.

Riesgos estado actual

- **Soporte limitado y riesgos de seguridad:** VB6 tiene un riesgo por sus vulnerabilidades y costes de mantenimiento.
- **Escasa modularidad:** el código está distribuido en muchos formularios y módulos con dependencias.
- **Dependencia de sistemas operativos antiguos:** Algunos clientes usan versiones antiguas, limitando la adopción de nuevas tecnologías.

Situación deseada

- Utilizar tecnología moderna.
- Mantener la aplicación on-premise, pero preparada para escenarios cloud.
- En lo posible, minimizar dependencias de software externas en los equipos de clientes.
- Organizar código de forma desacoplada, por features y preparado para migrar a la nube.
- Modernizar la interfaz de usuario para mejorar la experiencia.
- Incluir actualizaciones automáticas con opción de rollback.
- Cumplir con políticas de seguridad y compliance de la compañía.

Tecnologías a considerar:

- **Lenguaje y runtime:** C#.NET y .NET 9 como base para aplicaciones modernas, autocontenidas o framework-dependent.
- **Base de datos:** SQLite/SQL Server LocalDB según escenario.
- **Arquitectura de código basada en features y cloud-ready:** se organizará el código por funcionalidades, encapsulando comandos, consultas, lógica de negocio, acceso a datos y controladores en cada feature. Esto minimiza el acoplamiento y mejora la cohesión. Código preparado para posible migración a cloud.
- **Interfaz modernizada:** la capa de presentación podría implementarse en **MAUI** para compatibilidad multiplataforma, **WPF**/WinForms autocontenida o **Blazor WebView** para un enfoque web local. Un backend **API REST** en ASP.NET Core permitirá desacoplar la lógica y facilitar probable migración a la nube.
- **Actualizaciones automáticas:** la app se distribuirá mediante instaladores MSIX/WiX Toolset/Squirrel o ejecutables autocontenidos y contará con un servicio interno que compruebe nuevas versiones y se actualice de manera silenciosa.
- **CI/CD:** Se configurarán pipelines de CI/CD en GitHub Actions para compilar, probar y publicar cada release.

- **Pruebas unitarias e integración continua:** adoptar pruebas desde el inicio y utilizar herramientas de IA para generar documentación y casos de prueba.
- **Participación y motivación del equipo:** fomentar una cultura de **mejora continua** y aprendizaje. Mostrar beneficios del nuevo enfoque y oportunidad de utilización de nuevas tecnologías.

Estrategia por etapas

1. **Análisis y planificación**
    - **Inventario del código**: evaluar código VB6, componentes externos y dependencias. Clasificar módulos por criticidad y dificultad.
    - **Entrevistas con usuarios y equipo de soporte**: identificar funcionalidades críticas y flujos de trabajo. Involucrar a usuarios para obtener su perspectiva.
    - **Evaluación de tecnologías**: realizar pruebas de concepto con .NET 9/8 para validar compatibilidad y rendimiento. Si se determina que el tamaño de las aplicaciones autocontenidas es problemático, considerar publicarlas como *framework-dependent* (instalador más ligero) o utilizar técnicas de *trim*.
2. **Diseño de la arquitectura deseada**
    - **Backend API**: implementar una API REST con ASP.NET Core organizada en vertical slices (por ejemplo, `Features/Empresas`, `Features/Activos`, etc.). Cada slice contendrá comandos, consultas, validaciones, servicios y controladores.
    - **Capa de datos**: usar **Entity Framework Core** para la persistencia. Los modelos se pueden generar a partir de la base actual, adaptando los tipos de datos.
    - **Capa de presentación**: elegir WPF/MAUI/Blazor WebView según los requisitos de experiencia de usuario y hardware. La UI consumirá la API REST local. Se recomienda la participación de área de productos para colaborar con mockups.
    - **Servicios de actualización**: integrar un módulo que verifique versiones en un servidor central y realice actualizaciones silenciosas. Para los clientes que no puedan tener conexión permanente, evaluar ofrecer un instalador.
3. **Refactorización y desacoplamiento**
    - **Estrategia incremental**: comenzar con la creación de nuevas funcionalidades en la arquitectura moderna y rodear el sistema VB6 existente. Migrar gradualmente los módulos de VB6 a C# y retirar el código antiguo. Esto evita el enfoque *big bang* y reduce el riesgo.
    - **Servicios compartidos**: extraer utilidades y lógica común a bibliotecas independientes para evitar duplicación.
    - **Pruebas automatizadas**: al migrar cada módulo, crear pruebas unitarias que comparen resultados con la versión VB6 para asegurar equivalencia.
4. **Herramientas de IA y automatización**

- **Análisis de código VB6**: emplear herramientas de Generative AI para mapear las dependencias, generar documentación y sugerir traducciones a C#. Esto puede reducir los tiempos de modernización y mejorar la comprensión del código.
- **Generación de pruebas**: usar IA para crear casos de prueba y documentación. La automatización de pruebas y documentación reduce esfuerzo y riesgo de errores.
- **Copilots y asistentes**: integrar herramientas como GitHub Copilot para la refactorización y adopción de patrones de diseño.

5. **Gestión del cambio y motivación de desarrolladores**
   - **Cultura de mejora continua**: fomentar la mentalidad de que el cambio es un proceso evolutivo y no una interrupción. Comunicar que la modernización ofrece oportunidades de aprendizaje y crecimiento profesional.
   - **Reconocimiento y participación**: reconocer las contribuciones del equipo y asignar responsabilidades en decisiones de diseño. Escuchar y demostrar beneficios con pruebas de concepto.
6. **Estrategias de migración y despliegue**
   - **Migración iterativa**: mover funcionalidades módulo por módulo. Cada iteración incluirá análisis, desarrollo, pruebas y despliegue piloto.
   - **Paralelismo**: permitir que las versiones VB6 y .NET coexistan durante un periodo para validar resultados.
   - **Pruebas de usuario y feedback**: liberar versiones beta a usuarios seleccionados para obtener retroalimentación temprana.
   - **Plan de despliegue final**: una vez que la mayoría de funcionalidades estén migradas, planificar la entrega general. Proporcionar soporte para incidencias.

Estimaciones

El siguiente backlog está pensado considerando **solo una persona** y utiliza estimaciones en horas (1 día = 7 horas). Se contemplan tres escenarios: Optimista (O), Normal (N) y Pesimista (P). Además, se identifican tareas que pueden ejecutarse en paralelo.

Feature / Historia de usuario	Resumen de tareas	O (h)	N (h)	P (h)	Paralelismo posible
Feature 1: Diagnóstico y planificación					

Feature / Historia de usuario	Resumen de tareas	O (h)	N (h)	P (h)	Paralelismo posible
Como analista quiero inventariar el código VB6	Ejecutar herramientas de análisis, catalogar formularios y dependencias, documentar riesgos.	28	42	70	Secuencial
Como product owner quiero entrevistar usuarios y equipo de soporte	Planificar entrevistas, realizar reuniones, recoger feedback y expectativas.	28	42	56	Secuencial
Como product owner quiero definir la visión de la situación deseada	Establecer objetivos, priorizar funcionalidades, crear mapa de ruta.	14	21	35	Secuencial
Feature 2: Diseño y preparación					
Como arquitecto quiero diseñar la arquitectura de vertical slices	Crear estructura de carpetas por feature, definir dependencias, documentar normas de codificación.	21	35	56	Paralelo con otras de preparación
Como DBA quiero diseñar el esquema SQLite y las estrategias de migración de datos desde Access/SQL Express	Definir modelo de datos, escribir scripts de migración, validar integridad.	35	49	84	Paralelo con otras de preparación
Como dev quiero configurar CI/CD con GitHub Actions	Crear workflows de build, test, análisis estático y empaquetado.	21	35	49	Paralelo con otras de preparación

Feature / Historia de usuario	Resumen de tareas	O (h)	N (h)	P (h)	Paralelismo posible
Como dev quiero configurar herramientas de IA	Seleccionar herramientas, integrarlas en el repositorio, documentar su uso.	14	21	35	Paralelo con otras de preparación
Como product owner quiero definir estrategias de formación y motivación	Diseñar plan de capacitación, coaching y comunidades de práctica; preparar sesiones de comunicación.	21	35	49	Paralelo con otras de preparación
Feature 3: Migración iterativa (por módulo)					
Como dev quiero migrar el módulo Administrador a C#/.NET	Diseñar API y UI para gestiones de empresas; migrar lógica; crear pruebas; desplegar versión piloto.	105	140	210	Secuencial
Como dev quiero migrar el módulo Cuentas Corrientes a C#/.NET	Implementar slice `CuentasCorrientes`, reescribir cálculos, pruebas, despliegue.	105	140	210	Secuencial
Como dev quiero migrar el módulo Activo Fijo a C#/.NET	Migrar cálculos de depreciación, reportes; implementar slice correspondiente.	84	112	175	Secuencial
Como dev quiero migrar el módulo Libros Legales e Impuestos a C#/.NET	Migrar generación de libros e impuestos artículo 14D; validar cálculos.	105	140	210	Secuencial

Feature / Historia de usuario	Resumen de tareas	O (h)	N (h)	P (h)	Paralelismo posible
Como dev quiero desacoplar la lógica de negocio de la UI implementando API REST y servicios	Crear comandos y consultas, separar reglas de dominio, exponer endpoints.	56	84	126	Secuencial
Como dev quiero integrar el servicio de actualizaciones automáticas en la nueva aplicación	Diseñar mecanismo de comprobación, descarga e instalación de nuevas versiones.	35	49	84	Secuencial
Como dev quiero empaquetar la aplicación en instaladores autocontenidos o framework-dependent según el análisis de tamaño y soporte	Configurar MSIX o ejecutables, firmar certificados y validar en clientes.	21	35	56	Secuencial
Feature 4: Pruebas y estabilización					
Como tester quiero ejecutar pruebas unitarias e integración para asegurar la paridad con VB6	Configurar proyectos de prueba, escribir casos y comparar resultados.	35	49	70	Paralelo con otras de pruebas
Como usuario clave quiero validar la aplicación migrada para garantizar que satisface las necesidades	Planificar pruebas piloto, recopilar feedback, reportar fallos.	28	42	56	Paralelo con otras de pruebas
Como dev quiero optimizar rendimiento y	Analizar informes de rendimiento, refactorizar y	21	35	56	Paralelo con otras de

Feature / Historia de usuario	Resumen de tareas	O (h)	N (h)	P (h)	Paralelismo posible
corregir errores detectados durante las pruebas	ajustar consultas.				pruebas
Como product owner quiero preparar manuales y material de capacitación para los usuarios finales	Crear documentación, videos de entrenamiento, planes de adopción.	28	42	56	Paralelo con otras de pruebas
Como product owner quiero planificar el despliegue final a los clientes de forma segura	Definir ventanas de instalación, comunicación, contingencias y soporte.	21	35	49	Paralelo con otras de pruebas
Feature 5: Soporte y mejora continua					
Como dev quiero atender incidencias y liberar correcciones después del lanzamiento	Mantener un sistema de tickets, asignar prioridades, publicar parches.	14	28	42	Secuencial
Como arquitecto quiero explorar la migración del backend a la nube en el futuro	Evaluar plataformas, analizar costos y prototipos.	35	56	84	Secuencial
Como product owner quiero medir métricas de uso y satisfacción para guiar mejoras	Implementar telemetría, analizar feedback y ajustar backlog.	21	35	49	Secuencial
Total		896	1302	1967	

Consideraciones y riesgos

- **Complejidad de componentes de terceros:** identificar controles o librerías VB6 sin soporte y evaluar alternativas modernas o desarrollar reemplazos.
- **Tamaño y actualizaciones de aplicaciones autocontenidas:** las aplicaciones autocontenidas incluyen el runtime y todas las librerías, por lo que su tamaño y requisitos de almacenamiento son mayores. Se debe evaluar si un despliegue *framework-dependent* (que requiere instalar .NET en el cliente) reduce el tamaño y simplifica actualizaciones.
- **Gestión de datos:** al migrar a SQLite se deben considerar las limitaciones (concurrencia, tamaño) y las implicancias de migrar datos desde Access/SQL Server. Hacer copias de seguridad y pruebas de integridad.
- **Planificación incremental:** evitar un enfoque *big bang* y adoptar migración gradual para minimizar riesgos.

Beneficios

- **Reducción de costos de soporte:** menos incidencias en instalación y compatibilidad.
- **Mayor productividad del usuario:** interfaz moderna, tiempos de respuesta más rápidos.
- **Cumplimiento normativo:** mitigación de riesgos legales y de compliance.
- **Escalabilidad técnica:** la solución está preparada para soportar nuevas normativas tributarias o cambios de negocio con ajustes localizados, evitando reescrituras masivas.
- **Menor riesgo de regresiones:** cada módulo está encapsulado y probado de manera independiente.
- **Facilidad de implementación de nuevas características:** gracias a la arquitectura desacoplada por features, es posible agregar o modificar funcionalidades de manera ágil, con bajo impacto en el resto del sistema.

Conclusión

Este plan no solo moderniza la aplicación contable, sino que además establece una base tecnológica escalable. La arquitectura desacoplada y preparada para la nube asegura que la inversión no quede obsoleta, mientras que la estrategia iterativa y la gestión del cambio reducen riesgos técnicos y humanos.
De esta manera, TR-Contabilidad no solo asegura su continuidad operativa, sino que se transforma en una plataforma flexible, preparada para responder rápidamente a nuevas necesidades regulatorias, tecnológicas y de los clientes.