

Playbook: de VB6 a .NET

Guía de decisión y ruta de migración

Muchas empresas siguen operando con aplicaciones de escritorio hechas en Visual Basic 6. Funcionan, pero cada año cuesta más mantenerlas, integrarlas con otros sistemas y encontrar a alguien que las conozca. Este playbook te ayuda a decidir qué hacer con tu aplicación VB6 y a planear una migración a .NET por etapas, sin apostar la operación a un único corte.

1. Por qué importa ahora

Microsoft trata por separado el entorno de desarrollo de VB6 y su runtime. Según su declaración oficial de soporte:

- El IDE de Visual Basic 6.0 no tiene soporte desde el 8 de abril de 2008. No existe una forma soportada de crear o mantener aplicaciones VB6.
- El runtime sigue soportado durante el ciclo de vida de las versiones de Windows con soporte (la lista incluye Windows 11 y Windows Server 2025), pero solo para regresiones graves y problemas críticos de seguridad de aplicaciones existentes.
- El runtime es solo de 32 bits: en Windows de 64 bits las aplicaciones VB6 se soportan únicamente en la capa de emulación WOW.
- Microsoft no da soporte a controles OCX/ActiveX de terceros; eso depende de cada fabricante.
- La propia Microsoft recomienda enfáticamente reemplazar estas aplicaciones con tecnología moderna.

En la práctica, la aplicación puede seguir abriendo, pero cada cambio se hace con herramientas sin soporte y el riesgo crece con cada dependencia de terceros y cada persona que deja la empresa.

2. Señales de que es momento de actuar

- Solo una o dos personas entienden el código y nadie más se atreve a hacer cambios.
- Dependes de controles OCX o DLL de proveedores que ya no existen o no responden.
- Cada actualización de Windows, antivirus o equipos genera temor a que algo deje de funcionar.
- El negocio pide integraciones (web, móvil, APIs, facturación electrónica, analítica) y la respuesta es "no se puede" o un puente frágil.
- Seguridad o auditoría piden autenticación centralizada, trazabilidad o cifrado que la aplicación no ofrece.
- Los reportes y conciliaciones terminan en Excel porque el sistema no los resuelve.
- Instalar la aplicación en un equipo nuevo requiere un procedimiento que casi nadie recuerda.

3. Opciones y cuándo encaja cada una

No hay una única respuesta correcta. La mejor opción depende del valor del sistema para el negocio, de cuánto va a seguir cambiando y de qué tan bien se entiende hoy. Las opciones se pueden combinar.

Retener y contener

Cuándo encaja: el sistema es estable, cambia poco y su reemplazo ya tiene fecha. Se aísla (red y permisos), se documentan dependencias y se prueban los respaldos.

Ten en cuenta: no reduce la deuda técnica; solo compra tiempo mientras decides.

Rehost (mover sin cambios)

Cuándo encaja: lo urgente es el hardware o el centro de datos, no el código. La aplicación y su base de datos pasan a servidores virtualizados o escritorios remotos.

Ten en cuenta: sigue siendo VB6 de 32 bits sin IDE soportado; es un paso intermedio, no un destino.

Envolver con APIs

Cuándo encaja: otros sistemas necesitan datos o procesos del VB6 y la lógica está en la base de datos o en componentes COM reutilizables. Se exponen servicios .NET sobre lo existente.

Ten en cuenta: da integración rápido, pero suma una capa que mantener y no elimina VB6.

Conversión asistida con herramientas (replatform)

Cuándo encaja: el código está razonablemente ordenado y quieres llegar pronto a .NET conservando la estructura. Las herramientas traducen formularios y código a C# o VB.NET.

Ten en cuenta: el resultado suele arrastrar la misma estructura y deuda; requiere ajuste manual y probar todos los flujos, en especial controles de terceros, impresión y reportes.

Refactor por módulos (patrón strangler fig)

Cuándo encaja: el sistema es crítico, seguirá evolucionando y puede convivir con uno nuevo durante un tiempo. Se construye un backend .NET con APIs y un frontend web módulo por módulo hasta apagar el VB6.

Ten en cuenta: necesitas el código fuente y poder separar módulos; durante la transición conviven dos sistemas y hay que coordinar los datos.

Reescribir o reemplazar (desarrollo nuevo o SaaS)

Cuándo encaja: el proceso de negocio cambió tanto que el sistema ya no lo representa, o un producto de mercado lo cubre con ajustes razonables.

Ten en cuenta: es la opción con más incertidumbre: las reglas no documentadas se pierden con facilidad y, si se hace de una sola vez, el valor llega tarde.

Una ruta combinada que conviene evaluar

Contener el VB6 mientras haces el diagnóstico, exponer APIs para las integraciones urgentes y migrar módulo por módulo hacia una versión LTS de .NET. Según la política oficial de Microsoft, las versiones LTS reciben soporte gratuito durante tres años (por ejemplo, .NET 10 hasta noviembre de 2028), así que planea también cómo vas a actualizar de versión.

4. Ruta por fases

1 Diagnóstico

Inventario de formularios, módulos, clases, controles OCX/DLL, reportes, base de datos, integraciones y usuarios. Entrevistas para ubicar las reglas de negocio críticas y los puntos de mayor dolor.

Entregable: mapa del sistema, riesgos y opción recomendada por módulo.

2 Módulo piloto

Un módulo acotado y visible para validar la arquitectura .NET, el acceso a datos, la autenticación, el despliegue y la forma de trabajo entre equipos.

Entregable: módulo en uso por usuarios reales y lecciones aprendidas para el resto.

3 Migración incremental

Módulo por módulo según prioridades del negocio, con pruebas que comparan resultados del sistema viejo y el nuevo, y reglas claras sobre qué sistema es dueño de cada dato.

Entregable: en cada iteración, módulos migrados, reglas documentadas y pruebas automatizadas.

4 Retiro del VB6

Cuando ningún usuario ni proceso dependa de la aplicación: congelar cambios, archivar código y datos históricos según tus políticas, desinstalar y retirar dependencias.

Entregable: acta de retiro y documentación del sistema nuevo.

Ejemplo ilustrativo (ficticio)

Distribuidora Demo factura y controla inventario con un sistema VB6 sobre SQL Server. Tras el diagnóstico, empieza por la consulta de inventario como módulo piloto en .NET y web, sobre la misma base de datos. Luego migra pedidos y facturación, compara resultados entre ambos sistemas durante cada salida y apaga el VB6 cuando el último módulo queda en uso.

5. Riesgos y cómo mitigarlos

Riesgo	Cómo mitigarlo
Reglas de negocio ocultas en los eventos de los formularios	Extraerlas y validarlas con usuarios clave antes de reescribir; pruebas de caracterización que fijen el comportamiento actual.
Controles de terceros sin equivalente (grillas, impresión, reportes)	Inventariarlos en el diagnóstico, decidir pronto su reemplazo y probar los reportes con datos reales.
Migración de datos	Perfilar la calidad de los datos, usar scripts repetibles, conciliar conteos y totales y ensayar la migración antes de cada corte.
Dos sistemas sobre la misma base de datos	Definir qué sistema escribe cada dato, probar transacciones y bloqueos y monitorear durante la convivencia.

Riesgo	Cómo mitigarlo
Operación en paralelo	Acordar un periodo en que ambos sistemas procesan lo mismo, comparar resultados y dejar por escrito los criterios para dar el paso.
Pruebas insuficientes	Casos de prueba basados en flujos reales, automatización en el sistema nuevo y pruebas de aceptación con usuarios.
Resistencia al cambio	Involucrar a los usuarios desde el piloto, capacitar y dar soporte cercano en cada salida.

6. Checklist de preparación

- ☐ Tienes el código fuente completo y compila en un entorno controlado.
- ☐ Hay un inventario de controles OCX/DLL, reportes e integraciones.
- ☐ Sabes qué módulos son críticos y quiénes son sus usuarios clave.
- ☐ Las reglas de negocio importantes están identificadas, aunque no estén documentadas del todo.
- ☐ Hay respaldos probados de la base de datos y un entorno de pruebas.
- ☐ Un patrocinador del negocio tiene tiempo para decidir prioridades.
- ☐ Está definido qué significa "terminado" para cada módulo (criterios de aceptación).
- ☐ Existe un plan para la convivencia de ambos sistemas y para el retiro final.

7. Cómo trabajamos la modernización

En Nextsoft no ejecutamos solos la migración de sistemas legacy: la hacemos junto con aliados estratégicos especializados. El aliado realiza la migración del sistema VB6; Nextsoft construye los módulos nuevos, las integraciones, el frontend y el QA, y es tu único punto de contacto durante todo el proyecto.

Siguiente paso

Haz el autodiagnóstico legacy gratuito: 10 preguntas sobre tecnología, operación, seguridad y negocio, con el resultado calculado en tu navegador. Si quieres conversar sobre tu caso, escríbenos.

Autodiagnóstico: nextsoftcorp.com/es/autodiagnostico-legacy

Correo: contacto@nextsoftcorp.com

Fuentes (consultadas en septiembre de 2026)

Microsoft Learn, declaración de soporte de Visual Basic 6.0: <https://learn.microsoft.com/es-es/previous-versions/visualstudio/visual-basic-6/visual-basic-6-support-policy>

Microsoft, política de soporte de .NET: <https://dotnet.microsoft.com/es-es/platform/support/policy/dotnet-core>

Microsoft Azure Architecture Center, patrón Strangler Fig: <https://learn.microsoft.com/es-es/azure/architecture/patterns/strangler-fig>