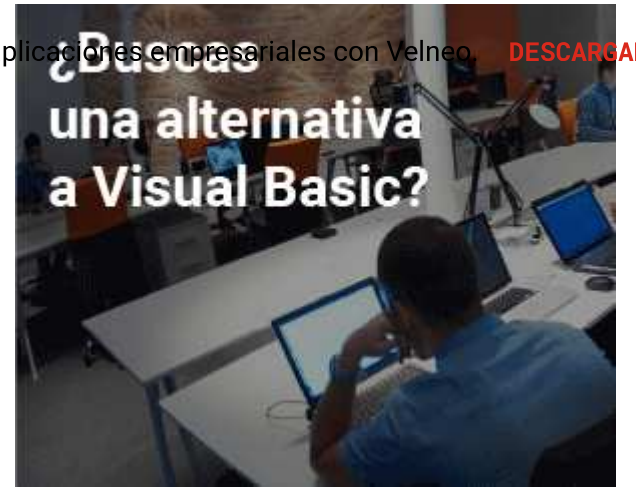
¿Por qué Microsoft descontinuó Visual Basic? ¡Descúbrelo!

POR [N1] FRED EL 24 FEBRERO 2016 | 8 COMMENTS

En este artículo vamos a repasar un poco la **historia** reciente de **Microsoft** adaptando un post muy interesante de **Tim Anderson**, adentrándonos en las **razones por las cuales Visual Basic 6 ha sido abandonado por Microsoft**. En su día, Visual Basic fue el lenguaje de programación más extendido y popular del mundo y aún así Microsoft optó por congelar su continuidad a favor de un Visual Basic nuevo y diferente. A continuación explicaremos **por qué Microsoft descontinuó Visual Basic**. Hay detalles del artículo original que han quedado desfasados, pero la argumentación de base y toda la polémica en torno a la discontinuidad de Visual Basic



Era una historia que sonaba perfecta. Microsoft tenía quizás la mayor comunidad de desarrolladores del mundo enganchados a un lenguaje que a su vez estaba enganchado a Windows. Aún así, Microsoft cogió este activo de valor incalculable y aparentemente lo echó a un lado. Allá por 2002 anunció que el lenguaje iba a ser reemplazado por algo nuevo, diferente e incompatible. Y desde ese día hasta hoy ese hecho ha provocado un estruendo constante. Muchos desarrolladores de Visual Basic reaccionaron con sentimientos que van desde la frustración hasta el resentimiento. Muchos se sintieron incluso traicionados. Ahora viene la explicación.

Para analizar este suceso tenemos que tener en cuenta **tres cosas**:

#1 La API de Windows

En primer lugar está el tema de la API de Windows. Es el interfaz de programación de bajo nivel en Windows, tal y como se explica en los manuales

de Windows.

#2 COM, el *Component Object Model*

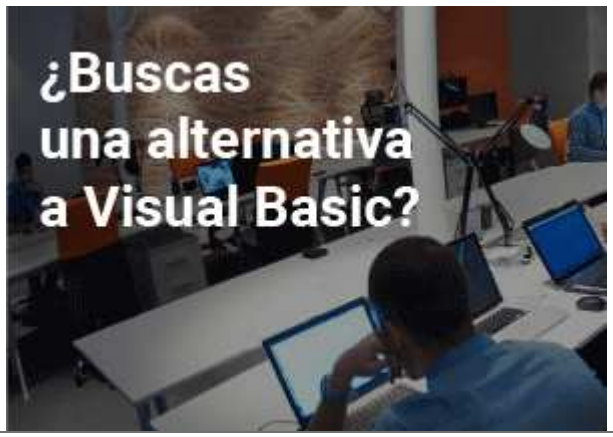
En segundo lugar, está COM, el acrónimo de *Component Object Model*. ¿Qué es COM? Es en esencia un mecanismo para vincular y unir componentes de software. Es un estándar binario, así que funciona con código compilado en runtime. En realidad COM es una familia de tecnologías. Una de ellas son los controles ActiveX que se encuentran tanto en Internet Explorer y Visual Basic. También está COM *automation*, que se usa en Microsoft Office y demás para controlar una aplicación de otra. Un tercer estándar COM es OLE (Object Linking and Embedding), que se usa cuando insertas una hoja de cálculo Excel en un documento Word.

#3 El framework .NET

El entorno .NET es el reemplazo de Microsoft para COM. Todos sabemos la lógica detrás de .NET: está un poco desfasado, pero en general es de gran utilidad. COM fue substituido porque fallaba. Es un estándar binario con un acoplado alto, lo que lo hace débil para aplicaciones web. Es altamente complejo, el cual era uno de los motivos principales por los que muchos desarrolladores se estaban pasando de Visual Basic a Java. También tenía problemas de versiones, provocando fallos en el software. En contraste, .NET tiene una arquitectura de acoplamiento bajo, ideal para Internet y aplicaciones

Es difícil subestimar lo que realmente supone el cambio de Microsoft de COM a .NET. Creo que debemos asumir que la empresa no lo hubiera hecho si hubiera tenido una alternativa mejor. Si la política de la industria lo hubiese permitido, podría haber cambiado a JAVA en vez de a .NET, pero el movimiento de alejarse de COM era necesario para que la plataforma de Microsoft siguiese siendo viable.

Hoy en día, con la familia de tecnologías denominadas Indigo, el alcance de esta medida se hace más aparente. Indigo subsituye a COM+, el servidor de transacciones base de COM que es clave en aplicaciones de empresa distribuidas que funcionan en Windows. Indigo además es el nuevo estándar para *web services* XML, MSMQ (*message queuing* o cola de mensajes), gestión de transacciones y objetos remotos, e incluso comunicaciones entre procesos. Indigo está construida en la segunda versión del framework .NET.



Asumo que en algún momento, allá por el año 2000, cuando los planes para sacar .NET estaban en ciernes, Microsoft se centró en Visual Basic 6.0 y valoraron qué hacer con él. VB6 fue lanzado en septiembre de 1998, así que en aquél año 2000 ya le tocaba una actualización. En ese momento, VB6 era un producto extendido, pero también fuente de un descontento considerable. Los desarrolladores se quejaban de su falta de orientación a objetos de forma completa y muchas de sus anomalías. Otro problema era el muro que suponía VB en forma de obstáculo. Algunas de las cosas que se podían hacer en C++ o en Delphi no se podían hacer en VB, a no ser mediante unos *hacks* monstruosos.

La verdad es que Visual Basic nunca fue concebida para ser un lenguaje de desarrollo completo. Se había diseñado para ser un lenguaje de composición de alto nivel para componentes de bajo nivel, un lenguaje de cohesión por decirlo de alguna manera. Por esta razón alrededor de Visual Basic se crearon una serie de componentes de mucho éxito por terceros, en us mayoría controles ActiveX. Estos componentes se programaron principalmente usando C++, pero utilizados principalmente desde Visual Basic. Sin ActiveX, Visual Basic tendría una falta de potencia grave.

Visual Basic obtiene las habilidades de sus componentes de COM. De hecho, Visual Basic se hizo usando COM. No es un simplemente un buen cliente o servidor COM; es un producto COM. La

PublicNotCreatable o *GlobalSingleUse*? Estos términos tan extraños son características COM. En otras palabras, la tecnología sobre la cual se hizo Visual Basic fue la tecnología que .NET estaba substituyendo. De ninguna forma fácil se podría adaptar VB para convertirse en un lenguaje de .NET.

Artículo relacionado: ¿Por qué cuesta tanto dar el salto de VB6 a VB.NET?

Claramente Microsoft tuvo que implementar un nuevo Visual Basic. Sin embargo, tomó la única decisión factible desde mi punto de vista. La empresa creó un producto totalmente nuevo, haciéndolo compatible con VB6 únicamente en aquellos aspectos que se podían sin dañar al nuevo lenguaje. En uno o dos casos mantuvo funciones rotas en VB6 por el bien de la compatibilidad (me viene a la mente el extraño caso del dimensionado de *arrays*), pero en general hizo todo de cero. El nuevo producto solucionó muchos de los problemas que se daban en VB6. Se carga la mayoría de las anomalías, soporta la orientación a objetos totalmente, elimina la dependencia de un solo entorno de desarrollo (VB.NET tiene un compilador de comando de linea) y de manera general elimina los muros y los obstáculos, equiparando a VB con cualquier otro lenguaje .NET.

El problema de la compatibilidad

Todo esto no ha venido gratis. Se ha pagado un

cientos de líneas de código de VB6. La arquitectura de la BBDD está basada en ADO, el último modelo de BBDD basado en COM. Y ahora viene Microsoft y dice que VB6 es un callejón sin salida. ¿Qué haces?

Es un escenario malo, y no poco común. La recomendación oficial de Microsoft es que migres a .NET, o que congeles la aplicación y solo la mantengas, y que hagas las nuevas funciones en .NET usando técnicas de interoperabilidad para integrar lo viejo con lo nuevo. Ninguna de las dos opciones es muy atractiva. Migrar es un esfuerzo mayúsculo, y tus desarrolladores tienen conocimientos en VB6 y COM, no .NET. La interoperabilidad es otra posibilidad, pero a menudo acarrea problemas de rendimiento al igual que otros imprevistos de programación ingeniosa.

Microsoft ofrece unas herramientas de migración… Francamente pueden ayudar un poco, pero hay muchos problemas intrínsecos. El mayor obstáculo no está en el lenguaje de programación, que convierte bastante bien en la mayoría de los casos. La madre del cordero está en la librería de clases, componentes y el interfaz gráfico. El motor de formularios de Visual Basic no tiene nada que ver con la librería de formulario de .NET Windows. Los controles ActiveX funcionan hasta un punto en .NET, pero no son óptimos y a menudo causan problemas. Y peor aún, los programas avanzados en VB hacen un uso considerable de ingeniosos trucos y llamadas API que evidentemente van a tropezar con

Factores Mitigantes

Puede haber factores mitigantes. En general, las aplicaciones que no son visuales se convierten o interoperan más fácilmente que los programas que sí son visuales. Los programas que siguen las mejores prácticas en términos de separar la lógica del negocio de la presentación del código, y que hacen uso de un diseño orientado a objetos son más fáciles de migrar o mantener que aquellas que no sean así. Sin embargo, incluso las aplicaciones mejor programadas siguen teniendo un problema…

COM no está del todo muerto

Personalmente me gusta mucho .NET. Mi instinto general a la hora de pensar en el futuro de una aplicación que tengo hecha en VB es hacerla de nuevo en .NET o quizá una aplicación en JAVA para

en marzo de 2005.

Nota: en el año 2014 se publicó que el *runtime* de VB6 tendrá soporte hasta el año 2024, Microsoft informó concretamente que el *runtime* es aún un componente del sistema operativo Windows 8.1 hasta como mínimo 2024.

Sin embargo, el verdadero soporte para VB6 está en la comunidad y en la web. A día de hoy se sabe absolutamente todo sobre el producto y sabemos también que VB es muy ampliable: puedes hacer llamadas a la API de Windows, puedes tirar de controles ActiveX, puedes crear DLLs en otros lenguajes y hacer llamadas desde Visual Basic.

Hasta no hace poco los programadores en Visual Basic se temían que alguna futura versión de Windows no soportara Visual Basic, atando a los desarrolladores de Visual Basic a sistemas operativos antiguos de Windows. Sin embargo, esto ya ha quedado solventado hasta al menos 2024. Microsoft informó concretamente que el *runtime* es aún un componente del sistema operativo Windows 8.1 hasta como mínimo 2024. En Microsoft no son estúpidos, ¿por qué iban a limitar la adopción de futuras versiones de Windows haciendo que no soportaran aplicaciones hechas en Visual Basic?

Además, en Microsoft aún se usa Visual Basic. VBA sigue siendo el lenguaje macro de Microsoft Office. Y en realidad Office sigue estando basado en COM y

menos.

Tomemos como ejemplo en paralelo el soporte de aplicaciones de 16-bits. Las aplicaciones de 16-bit aún corren en Windows XP, incluso aplicaciones DOS. Sin embargo, no son ejecutables en Windows de 64-bits. No era práctico darle tres niveles de soporte simultáneamente a 16-bits, 32-bits y 64-bits en Windows. Creo que las aplicaciones hechas en VB6 funcionarán mientras exista soporte para aplicaciones de 32-bits en Windows. También creo que Windows de 32-bits tendrá un recorrido más largo que el de 16-bits ya que hay muchísimas aplicaciones por ahí de 32-bits y las ventajas de 64-bits sobre 32-bits no son tan importantes para la mayoría de los usuarios. Las aplicaciones de VB6 funcionarán hasta el 2024 como mínimo, por ejemplo.

Yo no quiero decir que quedarse en VB6 ahora mismo sea una salida ideal. Está ya muy desfasado en alguna áreas y eso irá a más. Ya los últimos temas de Windows XP no eran soportados por aplicaciones en VB6. Es difícil a día de hoy seguir con VB6. Hacer nuevos desarrollos en VB6 no tiene mucho sentido ya. Otra cosa es que los clientes y las empresas con aplicaciones hechas ya en VB6 no les preocupe mucho todo esto. Solo quieren que su software y sus programas funcionen y punto.

¿Qué más podría haber hecho Microsoft?

Microsoft podría haber continuado con COM y no haber creado .NET. Personalmente pienso que esta opción no era viable. Si lo hubiera hecho, la migración de Microsoft a Java y a otras herramientas hubiese sido enorme.

También podría haber creado un VB7 compatible de forma separada a .NET, como una forma de desarrollo aparte. Esto es lo que sucedió con FoxPro (y el final del camino también le ha llegado). La principal desventaja aquí es no se hubiese ofrecido una senda de migración clara para aquellos que sí optaron por dar el salto de VB6 a .NET. Si Microsoft hubiera hecho esto nadie se hubiese tomado su estrategia de .NET en serio. O directamente podría haber creado en paralelo VB7 y VB.NET, lo que hubiese conducido a la confusión más absoluta. Podría haber sido una medida para apagar fuegos, pero no hubiese cambiado la dura realidad: un Visual Basic basado en COM no tiene cabida en el mundo de .NET.

Creo que Microsoft tomó una decisión acertada al congelar VB6. La decisión acertada no implica que no haya creado angustia. Microsoft estaba en una encrucijada, y VB no le iba a ayudar a elegir el buen camino. Esta historia siempre iba a terminar en llanto.

Si has llegado hasta aquí y estás buscando una alternativa a Visual Basic 6 (VB6) te recomendamos

8 Responses to "¿Por qué Microsoft descontinuó Visual Basic? ¡Descúbrelo!"

[N2] MATCAS DICE:

17 MARZO 2016 A LAS 19:17

muy buen articulo

[N1] ALBERTO LINARES DICE:

25 MAYO 2017 A LAS 23:47

excelente articulo!!!

[N1] AUDESER DICE:

28 MAYO 2017 A LAS 19:37

Aquí un desarrollador “avanzado” de

en posicionamiento de su lenguaje de programación en escritorio, y en otros entornos críticos que aparecieron (léase, todo desarrollo web o el campo de los móviles).
Tampoco justifica porqué una nueva versión, que arreglase el problema de COM o el caos de ActiveX, debería ser un nuevo lenguaje totalmente incompatible con lo anterior. Cuando programo, me fuerzo a hacerlo declarando todas las variables, refiriendo a los objetos sin valores por defecto y llamando a todas las librerías de forma explícita. Ese fue el gran cambio que forzó a los que se pasaron a .NET… y podría conseguirse de mil formas sin requerir un VB .NET "distinto" al VB previo.
Personalmente, creo que el enfoque de MS ha sido completamente errado. Si tienes una comunidad grande (con un lenguaje en posición dominante), que te está reportando fallos y te advierte de las limitaciones, lo que debes hacer es intentar corregirlos, bien con Service Packs o con una nueva versión que amplíe a las nuevas necesidades requeridas. Lo último que debes hacer es cambiarles el lenguaje y hacerlo no retrocompatible. Así fue que, muchos no lo siguieron, o forzados a aprender algo nuevo, migraron a otras alternativas que ya se les estaban

puesto MS para mantener .NET a flote.
Y con los bandazos que da para acabar con los últimos irreductibles, retirando otorgar nuevas licencias de VBA a las aplicaciones (Arc-Gis, AutoDesk, Corel…) para forzar el paso a VSTO o .NET, Python se está convirtiendo en el lenguaje de referencia para desarrollos de scripts en aplicaciones de escritorio. No entiendo la estrategia, lo siento.
Si finalmente deciden cerrar la vía de VBA en el futuro, que es uno de los puntos sobre el que se soporta el apabullante dominio de Office en el entorno empresarial (y que luego extiende al ámbito doméstico), no imagino el nivel de enfado que pueden recibir. Espero que VBA tenga una larga vida, aunque sea como está, y con ello el soporte a aplicaciones VB6 se continue en Windows.

[N1] NERY SEGOVIA DICE:

6 JULIO 2017 A LAS 18:13

Realmente comparto AUDESER, si esta es la explicación del porque del descontinuo del VB6 no lo entiendo. Para mi que fue un error o sea UN GRAN ERROR de Microsoft el cortar VB6, si Office esta respaldado por VB y lo han mantenido es de esperarse

sito, win 8 para el olvido y ahora con win 10 repuntando (remando)…..
De todas maneras la cantidad de profesionales que estaba con VB6 con sus errores fue de lo mejor, al pasar al .NET se perdió todo los profesionales pasaron a otros lenguajes menos al .NET y .ASP eso es bien sabido. Ademas cuanto dinero ha invertido para .NET que después de 8 años no puede igualar y me atrevo a decir; competir con JAVA, PHP etc.
De todas maneras sería bueno un VB7 web.

[N1] LEONARDO_STURNIOLO DICE:

21 OCTUBRE 2018 A LAS 14:44

creo que muchos invirtieron mucho tiempo aprendiendo mucho vb6 y perdieron todo su capital en esa conversión.

[N1] GEOVANI_MARTINEZ DICE:

9 FEBRERO 2019 A LAS 0:10

La estregia de Microsoft tuvo aciertos y desaciertos. Aciertos lograron consolidar uno de los mejores IDES para desarrollo web , potabilidad, el CLR.framework y acceso a datos

“transparente” con COM. Funcionaba muy bien y con vb6 alcanzó su más alta madurez. Microsoft pudo haber seguido los 2 caminos dedicar vb a aplicaciones de escritorio COM y. Net para la web. De esta manera la transición hubiese sido mas rápida hacia las nuevas tecnologías en aquel entonces. Las aplicaciones de escritorio de .Net no han sido tan exitosas como se esperaba al punto que este tipo de aplicaciones tienen más acogida con access y su gran número de seguidores que viven cobijados bajo office hasta el día de hoy y su poderoso VBA.

[N1] ALORES DICE:

11 FEBRERO 2019 A LAS 10:53

Hola Geovani,

Estoy muy de acuerdo con tu comentario.

A mayores, y cambiando de tema, parece que la estrategia de Microsoft ahora ya no pasa por la coexistencia entre los lenguajes de programación VB y C# en .NET.

El director del programa de Microsoft, Mads Torgersen, ha publicado sobre la estrategia del lenguaje de programación de la compañía

Aunque Torgersen insiste en que su equipo “hará todo lo necesario para mantener a Visual Basic como un ciudadano de primera clase del ecosistema.NET”, las perspectivas para el lenguaje de programación no son buenas. Más desarrolladores de VB cambiarán ahora que Microsoft ha sido claro sobre sus planes.

[N1] RADBASIC DICE:

10 JULIO 2019 A LAS 0:50

Hola,

Si os interesa, hay una alternativa 100% compatible al Visual Basic 6. Se llama RAD Basic. Podéis consultar en su página web:
http://www.radbasic.dev

Carga directamente los proyectos de VB6 (ficheros vbp) y puede compilar a 64 bits.

Visual Basic 6 todavía no está muerto y creo que dará mucha guerra en el futuro.

Saludos

Deja un comentario

Lo siento, debes estar conectado para publicar

(+34) 986 93 21 63

life is soft