# Aplicaciones con el uso de ADO. NET y LINQ

Hoy en día, muchos programadores empresariales deben usar dos (o más) lenguajes de programación: un lenguaje de alto nivel para las capas de presentación y lógica de negocios (por ejemplo, Visual C# o Visual Basic) y un lenguaje de consulta para interactuar con la base de datos (como Transact-SQL). Esto requiere que el programador tenga conocimientos de varios idiomas para ser efectivo y también causa discrepancias de idiomas en el entorno de desarrollo. Por ejemplo, una aplicación que utiliza API de acceso a datos para ejecutar una consulta en una base de datos especifica la consulta como un literal de cadena usando comillas. Esta cadena de consulta es ilegible y no se comprueba si contiene errores, tales como una sintaxis no válida o si existen las columnas o las filas a las que hace referencia. No hay ninguna comprobación de tipo de los parámetros de consulta y tampoco hay compatibilidad con IntelliSense.

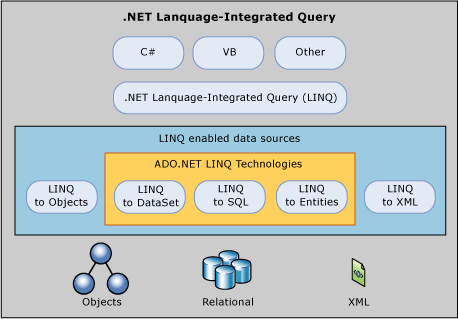
Language-Integrated Query (LINQ) permite a los programadores formar consultas basadas en conjuntos en el código de su aplicación, sin tener que usar un lenguaje de consulta independiente. Se puede escribir consultas de LINQ en varios orígenes de datos enumerables (es decir, un origen de datos que implementa la interfaz [IEnumerable](https://docs.microsoft.com/es-es/dotnet/api/system.collections.ienumerable)), como estructuras de datos en memoria, documentos XML, bases de datos SQL y objetos [DataSet](https://docs.microsoft.com/es-es/dotnet/api/system.data.dataset). Aunque esos orígenes de datos enumerables se implementan de varias formas, todos revelan las mismas construcciones de lenguaje y sintaxis. Como las consultas se pueden formar en el lenguaje de programación mismo, no es necesario utilizar otro lenguaje de consultas que esté incrustado como literales de cadena que el compilador no pueda entender o comprobar. También integración de consultas en el lenguaje de programación permite a los programadores de Visual Studio ser más productivos proporcionando comprobación de sintaxis y tipo en tiempo de compilación y IntelliSense. Estas características reducen la necesidad de depuración y corrección de errores de consultas.

La transferencia de datos de las tablas de SQL a objetos de memoria a menudo es una tarea tediosa y propensa a errores. El proveedor de LINQ implementado por LINQ to DataSet y LINQ to SQL convierte el origen de datos en recopilaciones de objetos basadas en [IEnumerable](https://docs.microsoft.com/es-es/dotnet/api/system.collections.ienumerable). El programador siempre ve los datos como una recopilación de [IEnumerable](https://docs.microsoft.com/es-es/dotnet/api/system.collections.ienumerable) cuando se realiza la consulta y la actualización. Se proporciona compatibilidad completa con IntelliSense para escribir consultas en esas colecciones.

Existen tres tecnologías Language-Integrated Query (LINQ) de ADO.NET independientes: LINQ to DataSet, LINQ to SQL y LINQ to Entities. LINQ to DataSet Proporciona consultas más ricas y optimizadas en el [DataSet](https://docs.microsoft.com/es-es/dotnet/api/system.data.dataset) y LINQ to SQL le permite consultar directamente los esquemas de base de datos de SQL Server, y LINQ to Entities permite consultar un Entity Data Model.

# Terminología y manejo de ADO. NET y LINQ

El siguiente diagrama proporciona una visión general de cómo se relacionan las tecnologías ADO.NET LINQ con lenguajes de programación de alto nivel y orígenes de datos habilitados para LINQ.



# Resumen de video

**¿QUÉ ES ADO.NET?**

ADO .NET es la nueva versión del modelo de objetos ADO (ActiveX DataObjetos), es decir, la estrategia que ofrece Microsoft para el acceso a datos. ADO.NET ha sido ampliado para cubrir todas las necesidades que ADO no ofrecía, ADO .NET está diseñado para trabajar con conjuntos de datos desconectados, lo que permite reducir el tráfico de red. ADO .NET utiliza XML como formato universal de transmisión de los datos. ADO .NET posee una serie de objetos que son los mismos que aparecen en la versión anterior de ADO, como pueden ser el objeto Connection o Command, e introduce nuevos objetos tales como el objeto DataReader, DataSet o DataView.ADO .NET se puede definir como:

Un conjunto de interfaces, clases, estructuras y enumeraciones que permiten el acceso a los datos desde la plataforma.NET de Microsoft

Que permite un modo de acceso desconectado a los datos que pueden provenir de múltiples fuentes de datos de diferente arquitectura de almacenamiento.

Y que soporta un completo modelo de programación y adaptación basado en el estándar XML.

**Arquitectura ADO.NET**

La mayoría de las aplicaciones escritas en Visual Basic y Visual C# giran en torno a la lectura y actualización de información de bases de datos. Para permitir la integración de datos en aplicaciones distribuidas y escalables, Visual Studio .NET es compatible con una nueva generación de tecnología de acceso a datos: ADO.NET.

El problema es comunicar un programa o aplicación con una base de datos y más que comunicar se pretende que el programa o aplicación realice una serie de procesos u operaciones con la base de datos o mejor aún con el conjunto de tablas que contiene una base de datos.

La primera nota por recordar es que una base de datos puede estar físicamente en el servidor y en algún fólder o directorio del disco duro de dicha máquina. Otra cosilla que debemos recordar es que así como existen servidores de páginas (Web Server), servidores de correo (Mail Server), servidores de ftp (ftp Server), también existen servidores de bases de datos (DataBase Server), los más comunes son el SqlServer de Microsoft, Oracle, MySql, y muchos más, estos servidores también pueden crear, administrar y procesar una base de datos.

El modo de comunicación entre nuestra aplicación y la base de datos implica que ambos manejen un lenguaje de programación común, es decir no se puede mandar una instrucción en csharp .net, o en visual Basic .net o en Basic o pascal...o en cualquier otro lenguaje, a la base de datos y además esperar que esta última la entienda. Para entender esto, una razón muy sencilla es que la base de datos tendría que conocer o comprender todos los lenguajes de programación, ahora dime, no sería más fácil que exista un lenguaje común...?, entonces para resolver este problema de comunicación es que se usa un lenguaje común de bases de datos que tanto los lenguajes de programación existentes como las bases de datos entienden, este lenguaje común de bases de datos es el SQL (Structured Query Languaje) o lenguaje estructurado de consultas.

La pregunta es ahora como mandamos las instrucciones SQL a la base de datos, la respuesta es mediante los OBJETOS ADO.NET, las cuales proporcionan acceso coherente a orígenes de datos como Microsoft SQL Server, así como a orígenes de datos expuestos mediante OLE DB y XML.

En la actualidad ADO.NET ya es parte del .NET Framework, esto quiere decir que es, de alguna manera, parte del sistema operativo y no más un redistribuible de 4 ó 5 MB que se necesita alojar junto al cliente o junto al instalador de una aplicación. Esto significa que nosotros, como desarrolladores, estaremos enfocados más al acceso a datos y a la lógica para manipular estos datos, y no tendremos porqué preocuparnos en cómo a los clientes la librería.

Si se prevé que la aplicación va a tener un ciclo de vida largo, entonces debe considerar la posibilidad de rediseñar la tecnología de acceso a datos de la aplicación y utilizar ADO.NET en aplicaciones administradas o ADO en aplicaciones nativas. A largo plazo, el uso de las tecnologías más modernas de acceso a datos reduce el tiempo de desarrollo, simplifica el código y proporciona un rendimiento excelente.