6. Plataformas para el Desarrollo en .NET

6.1. Introducción y objetivos

En este tema se estudian las plataformas relacionadas con la tecnología .NET. En primer lugar, estableceremos el contexto de la tecnología para conocer cómo ha ido evolucionando a lo largo de los años y cuál es su situación actual, así como las ventajas e inconvenientes de su uso.

Posteriormente veremos algunos de los principales entornos de desarrollo integrado que facilitan la escritura, compilación, depuración y despliegue de soluciones basadas en .NET, así como algunas plataformas que facilitan la automatización de la construcción y el despliegue de este tipo de aplicaciones. Por último, expondremos algunas de las plataformas que aceleran el desarrollo de soluciones basadas en tecnologías .NET.

Con el estudio de este tema pretendemos alcanzar los siguientes objetivos:

* Comprender el contexto, el valor y la utilidad que tiene el desarrollo con una tecnología con una importante presencia en el mercado como es .NET*.*
* Conocer los principales entornos de desarrollo integrado para .NET*.*
* Introducir algunas herramientas de automatización para la gestión de dependencias y paquetes sobre .NET*.*
* Conocer algunas plataformas aceleradoras del desarrollo a nivel de interfaz de usuario y servidores empresariales*.*

6.2. Contexto de las plataformas para el desarrollo en .NET

Bajo el nombre de .NETincluimos el conjunto de plataformas de desarrollo de software que Microsoft ha ido lanzando a lo largo de los años, fruto de la evolución propia de la tecnología, así como de las necesidades que como proveedor de plataformas ha querido implementar.

Para poder llegar a entender la situación actual es necesario conocer la historia desde su aparición. La historia de .NET se remonta a 1996, cuando el ingeniero danés Anders Hejlsberg abandona Borland, donde era el arquitecto del equipo de desarrollo de los productos Turbo Pascal y Delphi y se incorpora a Microsoft. Bill Gates pretendía conseguir una plataforma similar a Java —que por aquel entonces se estaba convirtiendo en la plataforma de desarrollo de referencia para muchos programadores—, pero mejorada, aprendiendo de sus fallos y limitaciones.

Hejlsberg comienza su andadura en Microsoft asumiendo el rol de arquitecto principal de la construcción de Visual J++, una implementación mezcla de C++ y Java 1.1 que tenía como principal característica la creación de interfaces gráficas. Dicha implementación no llegó a ver la luz puesto que Microsoft fue demandado por Sun y tuvo que paralizar su desarrollo. Al abandonarse dicho proyecto, surge uno nuevo: el proyecto .NET. El objetivo de este proyecto consistía en ofrecer las herramientas necesarias para crear software de propósito general con la más alta calidad.

A finales del 2000 se lanza la primera versión beta y es en febrero de 2002 cuando se libera oficialmente la versión 1.0 de lo que se conoció como .NET Framework. Desde esa primera versión, se ha ido liberando diferentes versiones mejoradas estructuralmente y de acceso a los recursos hardware disponibles.

Actualmente, .NET es una plataforma de desarrollo gratuita, de código abierto y multiplataforma con el que se pueden implementar diferentes tipos de aplicaciones, entre las que destacan, tal y como se muestran en la Figura 1, los siguientes:

* Aplicaciones de consola.
* Aplicaciones de escritorio, incluyendo Window Presentation Foundation (WPF), Windows Forms y la plataforma universal de Windows (UWP).
* Aplicaciones móviles.
* Aplicaciones web y microservicios.
* Aplicaciones nativas de la nube.
* Funciones sin servidor en la nube.
* Otras, como pueden ser juegos, IoT (Internet de las cosas) o servicios de Windows.

A diagram of a network

Description automatically generated with medium confidence

Figura 1. Tipos de aplicaciones desarrolladas con .NET. Fuente: Díaz Alcolea, 2020.

Con .NET, el código y los archivos del proyecto tienen el mismo aspecto independientemente del tipo de aplicación con que se compile. Y cada aplicación tiene acceso a las mismas funcionalidades del lenguaje, API y tiempo de ejecución. Por otro lado, al ser multiplataforma es posible generar aplicaciones para diferentes sistemas operativos (Windows, macOS, Linux, iOS, Android, etc.) y permite, además, usar funcionalidades específicas de la plataforma como son las API propias del sistema operativo.

.NET ofrece lo que se conoce como implementaciones, que ofrecen características diferentes en función del tipo de aplicación que se desea desarrollar y el sistema operativo al que va dirigido. Cada implementación incluye un entorno de ejecución y una biblioteca de clases, lo que permite incorporar también marcos de aplicaciones y herramientas de desarrollo.

Dentro de las implementaciones, actualmente conviven:

* .NET Framework: es la implementación original de .NET que existe desde el 2002, solo se ejecuta en Windows y está optimizado para crear aplicaciones de escritorio de Windows. A partir de la versión 4.5 implementa el .NET Standard.
* .NET 5+ (incluido .NET Core): es la implementación más reciente y se ejecuta en cualquier plataforma. Es la implementación más reciente y se ejecuta en cualquier plataforma. Se ejecuta en Windows, macOS y Linux, e implementa .NET Standard.
* Mono: utilizada cuando se requiere un tiempo de ejecución pequeño. Se suele emplear como un compilador Just-In-Time, aunque también incluye un compilador estático completo. Se activa desde las aplicaciones Xamarin para dispositivos móviles y también para juegos creados con el motor Unity. Admite todas las versiones de .NET Standard publicadas actualmente.
* Universal Windows Platform: se utiliza para compilar aplicaciones de IoT. Se ha diseñado para unificar los diferentes tipos de dispositivos de destino, como tabletas, teléfonos e incluso la consola Xbox. Proporciona un entorno de ejecución (AppContainer) y un conjunto de API de Windows para usar en lugar del Win32 (WinRT). Pueden escribirse aplicaciones C++, C#, Visual Basic y JavaScript.

La implementación más reciente de .NET es .NET 5+, que ya dispone de un calendario de versiones programado y que permite el desarrollo de aplicaciones que se ejecutan en cualquier sistema operativo.

Por otro lado, Microsoft desarrolló .NET Standard no como una implementación, sino como una especificación de API que establece los criterios comunes que permiten desarrollar bibliotecas de clases para varias implementaciones de .NET. Hasta su aparición de la especificación de .NET Standard, las distintas implementaciones tenían el esquema que aparece en la Figura 2.

A diagram of a computer program

Description automatically generated with medium confidence

Figura 2. Implementaciones previas de .NET. Fuente: Díaz Alcolea, 2020.

El grado de acoplamiento de la librería de clases respecto de la implementación hacía más complicada su evolución, por lo que, una vez definido y lanzado el .NET Standard, el esquema que de dichas implementaciones sería el que aparece en la Figura 3.

A diagram of a computer program

Description automatically generated with medium confidence

Figura 3. Incorporación de .NET Standard a las implementaciones. Fuente: Díaz Alcolea, 2020.

Las bibliotecas de clases son el concepto que introduce .NET para dividir las funcionalidades útiles en módulos y que puedan ser utilizados por varias aplicaciones.

Hay tres tipos de bibliotecas de clases:

* Las específicas de la plataforma, que permiten el acceso a las API de una plataforma determinada, pero que solo pueden ser usadas por las aplicaciones que van a ejecutarse en dicha plataforma.
* Las portables, que tienen acceso a subconjunto de API y las pueden usar aplicaciones que tienen como destino distintas plataformas.
* Las de .NET Standard, que ofrece lo mejor de las anteriores. Son específicas de la plataforma, ya que exponen toda la funcionalidad de la plataforma subyacente, y portables, puesto que funcionan en todas las plataformas compatibles. .NET Standard expone un conjunto de contratos y las implementaciones de .NET deben cumplir el contrato por completo o no admitirlo.

.NET es independiente del lenguaje. Esto significa que, como desarrollador es posible trabajar en uno de los muchos lenguajes que tienen como destino las implementaciones de .NET.

La independencia del lenguaje implica una interoperabilidad entre estos, es decir, es posible tener acceso a los tipos y miembros de bibliotecas de clases desarrolladas para implementaciones de .NET sin tener que conocer el lenguaje en el que se escribieron originalmente y sin tener que seguir las convenciones del lenguaje original. Para que esto sea posible, solo se deben exponer a aquellos que tengan las características comunes a todos los lenguajes y que se encuentran descritos en el *common language specification*(CLS), que es el conjunto de reglas que se aplican a los ensamblados generados y se encuentra definido en el estándar ECMA-335 (ECMA International, 2019).

Los tres principales lenguajes de programación que puedes utilizar para programar una aplicación .NET son C#, F# y Visual Basic. Los tres lenguajes de programación son compatibles con Windows, así como con macOS y Linux. Los usuarios de Windows tienen otra opción adicional que es C++/CLI, una variante del C++ clásico desarrollada por Microsoft.

* C# es un lenguaje de programación moderno, basado en objetos y con seguridad de tipos. C# tiene sus raíces en la familia de lenguajes C y a los programadores de C, C++, Java y JavaScript les resultará familiar inmediatamente.
* F# es un lenguaje de programación interoperable, multiplataforma y de código abierto para escribir código conciso, sólido y de gran rendimiento. El foco permanece en el dominio del problema, en lugar de en los detalles de la programación. La programación de F# está orientada a datos, donde el código implica la transformación de datos con funciones.
* Visual Basic es un lenguaje clásico que, entre los lenguajes de .NET, es el que presenta una sintaxis lo más parecida al lenguaje humano normal, lo que facilita su aprendizaje. A diferencia de C# y F#, para los que Microsoft desarrolla nuevas características de forma activa, el lenguaje Visual Basic es estable. Visual Basic no es compatible con las aplicaciones web, pero sí con las API web.

6.3. Valor y utilidad de las implementaciones de la plataforma .NET

El valor y utilidad del desarrollo en .NET se centran en las características propias de su desarrollo y evolución, en particular:

* Independencia de la plataforma: cuando Microsoft lanzó .NET en 2002 estaba pensado solo para usuarios de productos de Microsoft, lo que significaba que no se podía codificar ninguna aplicación para otro sistema operativo. Sin embargo, con la aparición de .NET Core, y actualmente con la versión .NET 6, los principales sistemas operativos son compatibles con ellas.
* Interoperabilidad de los lenguajes: uno de las principales beneficios y razones por las que los desarrolladores se decantan por el uso de .NET como plataforma es la posibilidad de programar en varios lenguajes, de forma que, de cara a desarrollar herramientas que cumplan requisitos de negocio concretos, se pueda hacer uso de lo mejor de cada lenguaje.
* Desarrollo de varios tipos de aplicaciones: esto es algo especialmente fluido en .NET, ya que no es necesario conocer el sistema operativo en el que se ejecutará una aplicación a la hora de desarrollarla, puesto que será posible programar de la misma forma independientemente de la plataforma destino y el tipo de aplicación: web, escritorio, aplicaciones para dispositivos móviles, aplicaciones de consola, etc.
* Biblioteca común: al hilo de la interoperabilidad de los lenguajes, se dispone de una biblioteca de clases muy amplia y en diferentes lenguajes de programación. Además, existe una amplia comunidad de usuarios y documentación sólida por parte de Microsoft.
* Gestión de la memoria: no hay necesidad de reservar memoria para las variables o gestionar punteros a espacios de memoria. El *runtime* se encarga de la gestión de la memoria, al asignar la necesaria requerida para variables y objetos. Cuenta con un recolector de basura (*garbage collector*, GC) que detecta si el programa no está siendo utilizado y procede a liberar el espacio de memoria que se encontraba dispuesto para su uso.
* Seguridad: permite manejar niveles de seguridad en el código, lo que posibilita la lectura pero no la escritura, por ejemplo, de aquellos accesos que procedan de la web.
* Mayor eficiencia en ejecución: las aplicaciones en .NET se compilan inicialmente en un lenguaje intermedio y, en tiempo de ejecución, el compilador Just-In-Time (JIT) convierte el lenguaje intermedio en código nativo. Además, para aplicaciones UWP, Microsoft ha desarrollado la tecnología .NET Native que permite la compilación directa en código nativo, con lo que aumenta todavía más su rendimiento.

Algunos de los inconvenientes del uso de .NET pueden ser:

* Soporte para objetos relacionales limitado: .NET funciona con el enfoque de programación orientada a objetos, que se basa en objetos en lugar de acciones y datos en lugar de lógica. Su Entity Framework proporciona el soporte para el desarrollo de aplicaciones de software orientado a datos en .NET Framework. La entidad actúa como puente entre el .NET Framework orientado a objetos y las bases de datos SQL. Sin embargo, algunos desarrolladores creen que Entity Framework no ofrece la flexibilidad requerida y no es compatible con todos los diseños de bases de datos disponibles.
* *Vendor lock-in*: .NET es un marco de Microsoft. Si bien Xamarin y .NET Core son fuentes abiertas, todo el ecosistema está lejos de ser impulsado por la comunidad. Esto significa que sus productos dependerán de las decisiones que tome Microsoft.
* Fugas de memoria: el problema de las fugas de memoria es común con todas las tecnologías. .NET también es una de esas plataformas que con frecuencia ha sido criticada por problemas relacionados con la memoria y la fuga de memoria. Aunque .NET posee un recolector de basura para este problema, los ingenieros aún deben invertir esfuerzos adicionales en la gestión de recursos.
* Dificultad en la transición a .NET Core: la transición a .NET Core podría ser larga y difícil. Las aplicaciones creadas con versiones anteriores de .NET son mucho más que las creadas con .NET Core. La transición de estas aplicaciones a las últimas versiones de .NET puede ser un poco complicada.

6.4. Entornos de desarrollo integrados

Como se ha introducido anteriormente, los entornos de desarrollo integrado (IDE) permiten a los desarrolladores trabajar en un solo entorno y mejorar su flujo de trabajo de programación. Cada IDE tiene diferentes recursos, pero todos incluyen un editor de texto, herramientas de compilación y un depurador.

Los IDE para el desarrollo con .NETmás populares son:

A blue and white text box

Description automatically generated

Figura 4. IDE para .NET. Fuente: elaboración propia.

Visual Studio

Es un entorno de desarrollo integrado (IDE) desarrollado por Microsoft. Se trata de un conjunto de herramientas que permiten desarrollar aplicaciones de escritorio, aplicaciones móviles, aplicaciones web ASP.NET, y servicios web XML. Tienen las siguientes características:

* Es multiplataforma: aunque es un producto desarrollado por Microsoft, es posible utilizarlo en Mac, Android, iOS y Linux. Así como también tanto en la web como en la nube.
* Promete eficiencia pura: no sólo se puede escribir códigos más rápidamente sino que permite depurar y emitir diagnósticos de forma mucho más sencilla.
* Es compatible con los siguientes lenguajes de programación: C#, C++, Visual Basic .NET, Python, Java, JavaScript, TypeScript, Node.js, Ruby, PHP y F#. También es compatible en las siguientes plataformas y flujos de trabajo: .NET, Web, Nube, escritorio, juegos, móvil (Android/iOS), ciencia de datos y aprendizaje automático/inteligencia artificial.
* Puedes crear aplicaciones nativas como híbridas: tanto para Windows como para Android e iOS.
* Utiliza gráficos vanguardistas.
* Incluye soporte técnico de primerísima categoría para el desarrollo web

Visual Studio es una herramienta de desarrollo muy completa, ya que le permite tanto a desarrolladores con años de experiencia como a desarrolladores novatos escribir código de forma sencilla. Visual Studio ofrece las siguientes características y funcionalidades:

* Ofrece opciones para depurar el código, generar perfiles y emitir diagnósticos de forma fácil.
* Permite integración de pruebas.
* Permite acceder a un *marketplace* para instalar extensiones.
* Permite utilizar el control de versiones de Git para rastrear y guardar los cambios en los ficheros.
* Permite desarrollar e implementar bases de datos de SQL Server y Azure SQL fácilmente.
* Permite desarrollar extensiones propias.
* Permite crear, administrar e implementar aplicaciones de escala de nube en Azure.
* Permite desarrollar tanto aplicaciones nativas como híbridas para Android, iOS y Windows.

Esta herramienta es muy útil para programadores, principalmente gracias a su versatilidad, ya que les permite:

* Crear softwares y programas relativos al universo web, como es el caso de aplicaciones (videojuegos incluso) o móvil, incluso instrumentos de *big data* y analítica.
* Editar, depurar y compilar código para posteriormente publicar una aplicación.
* Reducir no solo el esfuerzo en actividades rutinarias, sino además mejorar la eficiencia de las tareas esenciales de los desarrolladores.
* Instalar integraciones y extensiones relacionadas con *machine learning*(aprendizaje automático) o GitHub.
* Desarrollar extensiones propias en Visual Studio.

A screenshot of a computer program

Description automatically generated

Figura 4. Captura de los componentes para instalar en Visual Studio. Fuente: elaboración propia

Con más de una docena de versiones públicas, su formato actual (que es Visual Studio 2022, en su versión 17.1.1) ofrece una serie de mejoras y se posiciona como el mejor IDE completo para desarrolladores .NET y C++ en Windows. Las alternativas que ofrece Visual Studio son las siguientes:

* Community: IDE eficiente y gratis para estudiantes, colaboradores de código abierto y usuarios particulares. Permite crear aplicaciones para Android, iOS, Windows y la Web.
* Professional: IDE profesional adecuado para equipos pequeños.
* Enterprise: solución completa y escalable para equipos de cualquier tamaño.
* Team Explorer: una solución gratuita para que los usuarios que no son desarrolladores interactúen con Azure DevOps Server y Azure DevOps.

En este vídeo, *Introducción al desarrollo de aplicaciones con .NET y Visual Studio*, aprenderemos a crear varios proyectos con .NET y a editarlos, ejecutarlos y depurarlos sobre Visual Studio.

Las últimas características incorporadas son:

* Herramientas remotas: permite la implementación de aplicaciones, la depuración y la comprobación remotas, la generación de perfiles de rendimiento y la ejecución de pruebas unitarias en equipos que no tengan instalado Visual Studio.
* IntelliTrace *stand-alone collector:* permite recopilar datos de diagnóstico de sus aplicaciones en servidores de producción sin instalar Visual Studio ni volver a implementar la aplicación.
* Agentes: se puede usar para realizar pruebas de carga, funcionales y automatizadas.
* *Build tools*: permite compilar proyectos de Visual Studio desde una interfaz de la línea de comandos. Algunos proyectos admitidos son: ASP.NET, Azure, escritorio de C++, ClickOnce, contenedores, .NET Core, .NET Desktop, Node.js, Office y SharePoint, Python, TypeScript, pruebas unitarias, UWP, WCF (Windows Communication Foundation) y Xamarin.

A screenshot of a computer program

Description automatically generated

Figura 5. Captura de pantalla de Visual Studio 2022. Fuente: Lee y Olprod, 2022.

OmniSharp

Permite el desarrollo de aplicaciones .NET multiplataforma en editores como Atom, Brackets, Sublime Text, Emacs y Vim.

OmniSharp es un conjunto de herramientas, integraciones de editor y bibliotecas que juntas crean un ecosistema que le permite tener una gran experiencia de programación sin importar cuál sea su editor y sistema operativo de elección.

El proyecto OmniSharp ha sido creado por la comunidad para la comunidad. Si bien algunos de nosotros trabajamos para Microsoft, este proyecto es un esfuerzo de la comunidad y no está patrocinado ni respaldado por Microsoft.

.NET CLI

La interfaz de línea de comandos (CLI) es una nueva herramienta de plataforma cruzada para crear, restaurar paquetes, construir, ejecutar y publicar aplicaciones multiplataforma en Linux, macOS y Windows. Se instala con el SDK de .NET, por lo que no necesitamos instalarlo por separado en la máquina de desarrollo.

Visual Studio Code

Es un editor proporcionado por Microsoft de código fuente ligero pero potente que se ejecuta en su escritorio y multiplataforma, ya que está disponible para Windows, macOS y Linux. Dispone de soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C++, C#, Java, Python, PHP, Go) y *runtimes* (como .NET y Unity). El entorno no dispone de ningún alarde, centrándose exclusivamente en ofrecer flexibilidad y simpleza para precisamente promover esa compatibilidad en distintas plataformas que ofrece, más allá de soporte para repositorios Git o posibilidad de abrir varias instancias de archivos en una misma ventana.

Este editor tiene, según la última encuesta para profesionales de Stack Overflow (Stack Overflow, 2021), un liderazgo significativo en cuanto a los IDE de desarrollo.

A screenshot of a computer

Description automatically generated

Figura 6. Captura de pantalla de Visual Sutdio Code. Fuente: Visual Studio Code, s. f.

JetBrains Rider

Rider de JetBrains es un IDE .NET multiplataforma basado en la plataforma IntelliJ y ReSharper. Ofrece soporte para aplicaciones .NET y .NET Core en todas las plataformas.

Rider es compatible con proyectos basados en .NET Framework, el nuevo .NET Core multiplataforma y en Mono. Esto le permite desarrollar una amplia variedad de aplicaciones, incluidos servicios, bibliotecas y aplicaciones para escritorio .NET, juegos Unity, aplicaciones Xamarin y aplicaciones web ASP.NET y ASP.NET Core.

Cuenta con más de 2200 inspecciones de código en directo, cientos de acciones de contexto y refactorizaciones aportadas por ReSharper y combina todo esto con las sólidas funcionalidades de IDE de la plataforma IntelliJ. A pesar de un conjunto de funcionalidades pesado, Rider se ha diseñado para ser rápido y con capacidad de respuesta.

A screenshot of a computer program

Description automatically generated

Figura 7. Captura de pantalla de JetBrains Rider. Fuente: JetBrains, s. f. a.

6.5. Plataformas para la gestión de paquetes

El soporte de administración de paquetes permite flujos de trabajo de entrega continua al alojar nuestros paquetes, módulos y bibliotecas, para ponerlos a disposición del equipo de desarrollo o en los procesos de construcción y compilación o en los de despliegue. Entre las principales soluciones de gestión de paquetes para .NET se encuentran: NuGet, su NuGet Package Explorer y ProGet.

NuGet es el administrador de paquetes de referencia para .NET. Los paquetes NuGet son una forma para que los desarrolladores creen, compartan y consuman código reutilizable. La galería NuGet es el repositorio central de paquetes que utilizan todos los autores y consumidores de paquetes. Tiene alrededor de 150 mil paquetes únicos y más de un millón quinientos mil de versiones de paquetes. Es posible utilizarlo de forma integrada desde el IDE Visual Studio, pero también viene con una función de línea de comandos. Es equivalente a npm o *yarn* para Node, *brew* para macOS,apt-get para Linux o Maven para Java (NuGet, 2022).

En pocas palabras, un paquete NuGet es un único archivo ZIP con la extensión.nupkg que contiene código compilado (DLL), otros archivos relacionados con ese código y un manifiesto descriptivo que incluye información como el número de versión del paquete. Los desarrolladores con código para compartir crean paquetes y los publican en un *host* público o privado. Los consumidores de paquetes obtienen esos paquetes de *hosts* adecuados, los agregan a sus proyectos y luego llaman a la funcionalidad de un paquete en su código de proyecto. Luego, NuGet se encarga de todos los detalles intermedios.

Dado que NuGet admite *hosts* privados junto con el *host* público de nuget.org, puede usar paquetes de NuGet para compartir código exclusivo de una organización o un grupo de trabajo. También puede usar los paquetes de NuGet como una manera conveniente de factorizar su propio código para usarlo únicamente en sus propios proyectos. En resumen, un paquete NuGet es una unidad de código que se puede compartir, pero no requiere ni implica ningún medio particular de compartir.

Las principales funcionalidades que aporta NuGet son:

* Proporciona el repositorio central nuget.org con soporte para hospedaje privado.
* Proporciona las herramientas que los desarrolladores necesitan para crear, publicar y consumir paquetes.
* Lo que es más importante, mantiene una lista de referencia de los paquetes usados en un proyecto y la capacidad de restaurar y actualizar esos paquetes de esa lista.

Para que estos procesos funcionen de manera eficiente, administra una caché de paquetes y una carpeta de paquetes globales para atajar la instalación y la reinstalación. La caché evita descargar un paquete que ya se ha instalado en la máquina. La carpeta de paquetes globales permite que varios proyectos compartan el mismo paquete instalado, lo que reduce el espacio total de NuGet en la computadora. La caché y la carpeta de paquetes globales también son muy útiles cuando se restaura con frecuencia una gran cantidad de paquetes, como en un servidor de compilación.

Dentro de un proyecto individual, administra el gráfico de dependencia general, que nuevamente incluye la resolución de múltiples referencias a diferentes versiones del mismo paquete. Es bastante común que un proyecto dependa de uno o más paquetes que tienen las mismas dependencias. Muchos otros paquetes emplean algunos de los paquetes de utilidades más útiles de nuget.org. Para evitar traer varias versiones de ese paquete a la propia aplicación, NuGet determina qué versión única pueden usar todos los consumidores.

NuGet Package Explorer (NPE)

Es una aplicación que facilita la creación y exploración de paquetes NuGet. Puede cargar un archivo .nupkg o .snupkg desde el disco o directamente desde una fuente como nuget.org (GitHub, 2022a).

A screenshot of a computer

Description automatically generated

Figura 8. Vista de NuGet Package Explorer. Fuente: GitHub, 2022a.

ProGet

Es un sistema de gestión de paquetes, diseñado por la empresa de software Inedo. Permite a los usuarios alojar y administrar paquetes, aplicaciones y componentes personales o de toda la empresa. Originalmente, fue diseñado como un administrador privado de NuGet y un servidor fuente y de símbolos. A partir de 2015, ProGet amplió el soporte, agregó funciones de nivel empresarial y está diseñado para adaptarse a una metodología de desarrollo y operaciones (DevOps). Las empresas utilizan ProGet para «empaquetar aplicaciones y componentes» (Inedo, s. f.) con el objetivo de garantizar que el software se cree solo una vez y se implemente de manera uniforme en todos los entornos.

Las principales funcionalidades que aporta ProGet son:

* Permite utilizar paquetes universales para distribuir uniformemente sus aplicaciones y componentes o imágenes de Docker y gráficos de Helm (Kubernetes) para su software en contenedores.
* Facilita compartir código creado y probado previamente entre equipos de desarrollo, ya sea que use C#/.NET (NuGet), Python (PyPI o Conda), JavaScript/Node.js (npm), Ruby (Gems) o Java (Maven).
* Permite administrar software de Windows de terceros (Chocolatey), paquetes de PowerShell/recursos DSC y paquetes de Linux (RPM/YUM y Debian/apt).
* Soporta el almacenamiento de Amazon S3 o Azure Blob para escalar de forma económica y con pago por uso el almacenamiento de archivos de paquetes y una recuperación ante desastres fiable.
* Permite agregar tantos nodos de servidor web como necesite para equilibrar la carga del tráfico de varios equipos y crear servidores en todo el mundo.
* Da soporte para la detección y el bloqueo de licencias de código abierto que no puedan ser usadas a nivel comercial.
* Escanea los paquetes en busca de vulnerabilidades y bloquee el acceso a aquellos que se consideren demasiado riesgosos o inseguros según las reglas que configure.
* Registra los despliegues de otras herramientas, incluidas Otter, Octopus Deploy o incluso sus propios *scripts* personalizados, para que pueda ver qué paquetes se desplegaron en qué servidores.

6.6. Plataformas aceleradoras del desarrollo en .NET

Adicionalmente a las plataformas anteriormente presentadas, existen otras herramientas que extienden las funcionalidades del IDE, apoyan el análisis de rendimiento o ayudan a obtener el código cuando solo disponemos de ficheros compilados entre otras. Algunas de las principales plataformas aceleradoras del desarrollo en .NET son:

A screenshot of a computer

Description automatically generated

Tabla 2. Plataformas aceleradoras del desarrollo en .NET. Fuente: elaboración propia.

ReSharper

Es una extensión de Visual Studio creada por JetBrains para ayudar a los desarrolladores en la codificación de .NET. Es una de las herramientas de .NET para desarrolladores de .NET que ofrece la capacidad de evaluar la calidad del código de .NET y descubrir y reparar problemas rápidamente. También tiene muchos atajos para refactorizar y navegar rápidamente. Permite a los usuarios brindar soporte sin igual a todos los lenguajes tecnológicos desde C#, XAML, JavaScript, C++, TypeScript, JSON, ASP.NET MVC, XML, HTML, CSS, Protobuf, NAnt y secuencias de comandos MSBuild, incluida la funcionalidad integral entre idiomas (JetBrains, s. f. b).

Es una herramienta de productividad de código cuyo trabajo principal es ofrecer optimización del rendimiento de los códigos a través del rendimiento. Ayuda a los desarrolladores a monitorear perfiles para eliminar pérdidas de memoria. Es un corredor de unidades, un descompilador que se libera por sí solo. La suscripción a Resharper Ultimate también proporciona dotCover (un ejecutor de pruebas unitarias de .NET y una herramienta de cobertura de código), así como otros como dotTrace, dotMemory y dotPeek.

Algunas de sus funcionalidades son:

* Análisis de código: amplía Visual Studio con más de 2200 inspecciones de código sobre la marcha para C#, VB.NET, ASP.NET, JavaScript, TypeScript y otras tecnologías. Para la mayoría de las inspecciones, ReSharper proporciona arreglos rápidos (bombillas) para mejorar el código.
* Refactorizaciones: dispone de más de sesenta refactorizaciones y más de 450 acciones de contexto ayudan a organizar el código de forma segura y moverlo por la solución, distribuir la responsabilidad, desacoplar, disminuir la complejidad o simplemente usar una sintaxis de lenguaje alternativo.
* Navegación y búsqueda: ayuda a llegar instantáneamente a cualquier código en una solución, sin importar cuán grande sea la solución. También puede navegar desde cualquier símbolo a su código relacionado, como implementaciones de una interfaz dada, métodos de extensión de una clase o usos de un campo.
* Formateo y limpieza de código: permite configurar y aplicar el estilo de código según sus preferencias personales o el estándar de su equipo. La configuración de formato y estilo de código para C#, VB.NET, ASP.NET, JavaScript, TypeScript y otros lenguajes se puede aplicar en cualquier ámbito, desde una selección hasta una solución.
* Generación de código: puede crear una gran cantidad de código para usted, desde nuevos archivos y clases hasta bloques condicionales y GUID y desde métodos y propiedades requeridas por una interfaz que se implementa hasta comprobaciones de igualdad y formateo de miembros.

NDepend

Es una extensión de Visual Studio que ayuda en la herramienta de análisis de código administrado de .NET estática utilizada por un desarrollador de .NET. La aplicación permite la medición de la calidad del códigoal utilizar múltiples métricas de código, así como la visualización de su diseño. También proporciona una perspectiva de vista de pájaro de la organización de la aplicación y el código. Proporciona un lenguaje de consulta personalizado especializado para examinar el acoplamiento de la aplicación, así como la capacidad de evaluar la eficacia de la adhesión al código (NDepend, s. f.).

El programa también compara instantáneas de código base, gráficos y criterios de calidad. Las consultas LINQ se pueden usar para crear reglas definidas por el usuario. CQLinq es el nombre de esta función. Además, la herramienta incluye un número significativo de reglas de código CQLinq preestablecidas. En Visual Studio, las reglas de código se pueden verificar automáticamente o durante la integración continua.

A screenshot of a software

Description automatically generated

Figura 9. Vista del analizador NDepend. Fuente: NDepend, s. f.

LINQPad

Es un *playground* para programadores .NET, entendido como una aplicación liviana de .NET que se usa para escribir código C# de forma interactiva sin un IDE. La herramienta puede ejecutar no solo consultas simples, sino también una serie de consultas. Tiene un depurador integrado y capacidades de autocompletar (LINQPad, 2022).

La herramienta facilita que los desarrolladores puedan usarlo inmediatamente para probar fragmentos de código y consultar bases de datos usando LINQ o SQL Server Express. Es popular debido a su extenso formato de salida, depuración integrada, autocompletado opcional y ayuda a la creación de prototipos con comentarios instantáneos, soportando C#, F# o Visual Basic .NET.

A screenshot of a computer

Description automatically generated

Figura 12. Vista del *playground* LINQPad. Fuente: LINQpad, s. f.

.NET Reflector

Es un conocido descompilador y analizador estático de .NET. También está disponible como extensión de Visual Studio y es una de las herramientas más útiles para los desarrolladores de .NET. El beneficio de usar la herramienta reflectora .NET es depurar su programa y descubrir problemas. Incluso puede entrar en código de terceros escrito en Visual Studio.

ILSpy

Es el navegador y descompilador de ensamblados .NET de código abierto. Es una buena alternativa libre a .NET Reflector, que ya no tiene una versión gratuita. Las herramientas pueden descompilar cualquier archivo DLL de .NET a Visual Basic o C#. En caso de que haya perdido su código fuente o no lo haya tenido en primer lugar, ILSpy le permite ver cómo funciona el código detrás de las DLL del marco.

DotPeek

es un descompilador confiable basado en ReSharper, útil para depurar ensamblajes y códigos de terceros. Con dotPeek, puede examinar problemas que ocurren en componentes de terceros, incluso sin tener acceso directo a su código fuente. El descompilador admite múltiples formatos, incluidas bibliotecas (.dll), ejecutables (.exe) y archivos de metadatos de Windows (.winmd).

PerfCollect y PerfView

Son dos soluciones enfocadas para que las herramientas de desarrollo de Dotnet sean más eficientes. PerfCollect es un *script bash* que recopila datos y los automatiza. Los *scripts bash* son una serie de comandos contenidos en un archivo simple para su ejecución. Mientras que, por otro lado, PerfView es una herramienta de rendimiento que se centra principalmente en Windows. PerfView es una herramienta de análisis de rendimiento y PerfCollect automatiza la recopilación de datos.

PerfView también se puede utilizar para ajustar la CPU y la asignación de memoria. También es como una caja de herramientas de ajuste de rendimiento de *.NET*. No requiere ninguna instalación para comenzar a usarlo. Su funcionamiento simplemente copiando la aplicación a la PC con Windows donde se requieren datos de rendimiento. Después de eso, debe configurarse sobre lo que necesita ser perfilado, como una aplicación, y comenzará a capturar sin problemas.

DotTrace y dotMemory

Son herramientas de creación de perfiles para realizar un seguimiento de los problemas de rendimiento y memoria en las aplicaciones .NET. Capaces de procesar grandes volúmenes de datos, los muestran al usuario de forma integral.

Las funciones de dotTrace incluyen detección de cuellos de botella en el rendimiento, análisis de pilas de llamadas, informes de líneas de tiempo de rendimiento y muchas otras acciones de creación de perfiles (JetBrains, s. f. c).

DotMemory tiene perfiles remotos, inspecciones automáticas para cualquier problema de memoria, análisis de tráfico de memoria, comparación de instantáneas de memoria, etc. (JetBrains, s. f. d).

A screenshot of a computer

Description automatically generated

Figura 13. Vista del analizador de memoria dotMemory. Fuente: JetBrains, s. f. d

BenchmarkDotNet

Es un proyecto de código abierto que elimina las suposiciones de rendimiento y le brinda métricas al aislar automáticamente cada punto de referencia y ejecutar varias iteraciones, incluido el calentamiento. Permite escribir pruebas de rendimiento, para que puedas medir y comparar el rendimiento de tus algoritmos. Ayuda a transformar métodos en puntos de referencia, realizar un seguimiento de su rendimiento y compartir experimentos de medición reproducibles (GitHub, 2022b).

Los resultados se presentan en un formato fácil de usar que destaca todos los hechos importantes sobre su experimento. La biblioteca es adoptada por más de 6800 proyectos que incluyen .NET Runtime y es compatible con .NET Foundation.

A collage of graphs and diagrams

Description automatically generated

Figura 14. Vista del analizador de rendimiento de BenchmarkDotNet. Fuente: GitHub, 2022b.