Máster en Cloud Apps

Desarrollo y despliegue de aplicaciones en la nube

Curso académico 2019/2020

Trabajo de Fin de Máster

Análisis e implementación de partes del máster implementadas con tecnologías .net y afines

Autor: Javier Vela Aylón

Tutor: Francisco Gotázar Bellas

Esta página ha sido intencionalmente dejada en blanco

Contents

Introducción

El máster tiene como objetivo el adquirir conocimientos que permitan desarrollar y desplegar aplicaciones de internet utilizando las últimas tendencias del sector.

En el recorrido del master se han utilizado tecnologías como java, node, AWS,…. El objetivo trabajo fin de máster es el analizar e implementar varias partes del master con tecnologías .net y afines.

A no ser que se indique lo contrario, todas las librerías indicadas en la memoria son open source y son soportadas por .net Foundation.

Historia .NET framework

.NET framework es un SDK desarrollado por Microsoft que permite el desarrollo de aplicaciones para sistemas operativos Windows a partir de varios lenguajes de programación: C#, F# o visual basic .NET.

El framework se compone de dos partes: FCL (framework class library) y CLR (Commmon language runmitme)

- **FCL**: Ofrece interoperabilidad entre los diferentes lenguajes de programación e incluyen una serie de librerías que
    - Utilidades: Listas, diccionarios,…
    - Acceso a recursos del sistema operativo: red, ficheros,…
    - Otros frameworks: ASP.NET, WPF, ADO .NET

- **CLR**: ES la máquina virtual que maneja la ejecución de aplicaciones .NET, la compilación just-in-time (JIT) convierte el código manejado en instrucciones que puede ejecutar la CPU del ordenador. Aparte ofrece otros servicios tales como el manejo de memoria, ejecución

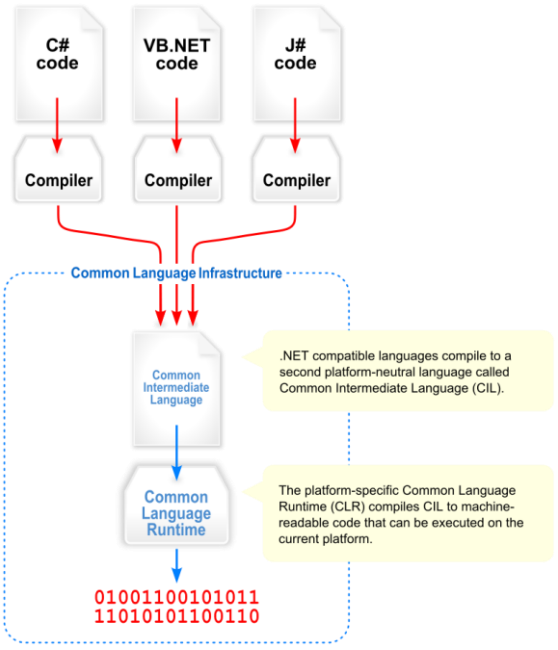


Figure 1.- Módulos .net framework: FCL y CLR

En el año 2000 se lanza la primera versión beta del framework y se trabaja en la estandarización de CLI y del lenguaje de programación C#, ECMA lo ratifica en 2001 e ISO en el año 2003. Durante los siguientes años se siguen lanzando nuevas versiones del framework ampliando y mejorando significativamente las funcionalidades del mismo.

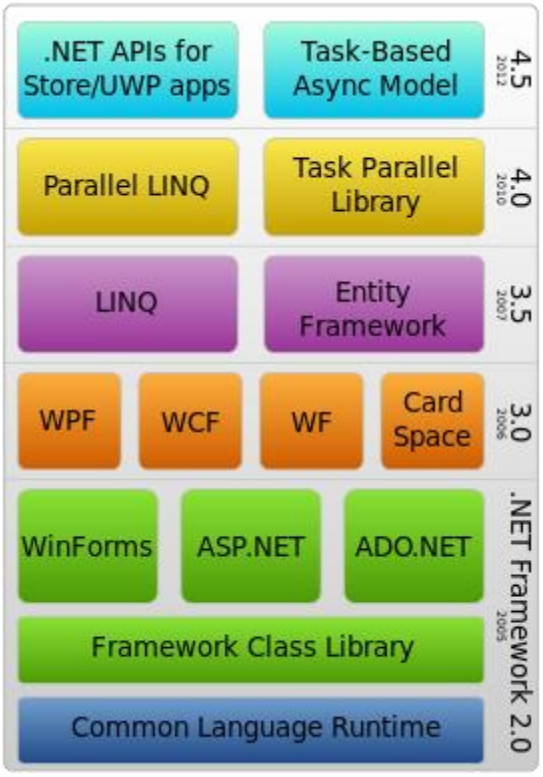


Figure 2.- Librerías / versión framework disponible

En 2004 ve la luz un proyecto denominado mono que implementa los estándares definidos para .NET framework. Su objetivo no es solo la implementación de estos estándares sino también es que sea multiplataforma: Android, Linux, Windows, PS3, Wii o Xbox 360. Mono es adquirido por Novell y en el año 2011 se funda una nueva empresa llamada Xamarin con el fin de desarrollar aplicaciones móviles con un único lenguaje. Microsoft compró Xamarin en el año 2016, convirtiendo Xamarin en un SDK open source y gratuito integrado dentro de sus herramientas de desarrollo.

En el año 2014 es anunciado .NET Core. Es un nuevo framework open source reescrito desde cero, se reaprovechan algunas APIs ya estandarizadas y será el framework sustituirá en los proximos años a .NET framework (4.8 se lanzó en 2019). El proyecto pertenece a la .NET Foundation donde cualquier desarrollador puede participar en el diseño y desarrollo del framework.

A diferencia de .NET framework, .NET Core es multiplataforma: SDKs y runtimes están disponibles para una gran variedad de sistemas operativos y sus correspondientes distribuciones. También es modular y está compuesto de varios paquetes distribuidos a través de NuGet.

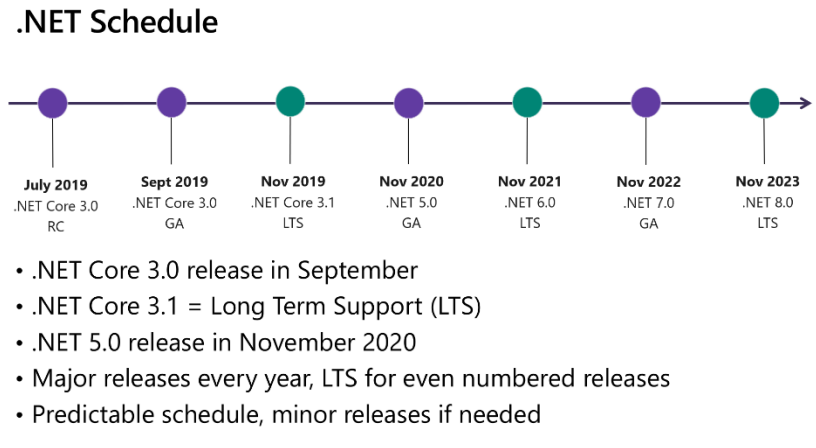


Figure 3.- Fechas lanzamiento versiones .NET Core

Desde la primera beta del nuevo framework hasta hoy (.NET Core 5 – noviembre 2020) se han lanzado diferentes versiones para unificar en un único framework toda la funcionalidad necesaria para poder construir aplicaciones para escritorio, aplicaciones móviles o aplicaciones para la nube con un alto rendimiento.

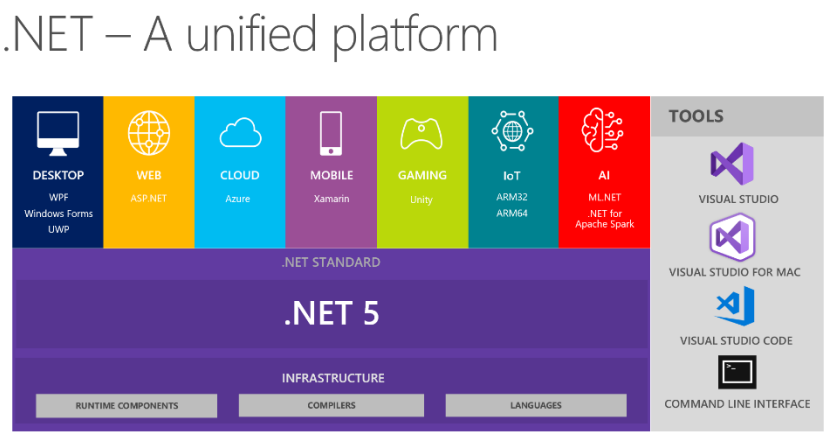


Figure 4.-sdfsdfasdf

Aplicaciones .NET

El framework .NET permite crear diferentes tipos de aplicaciones: aplicaciones de línea de comandos, escritorio, móvil, web, IOT, librerías… en el presente trabajo fin de máster únicamente nos centraremos en las aplicaciones web.

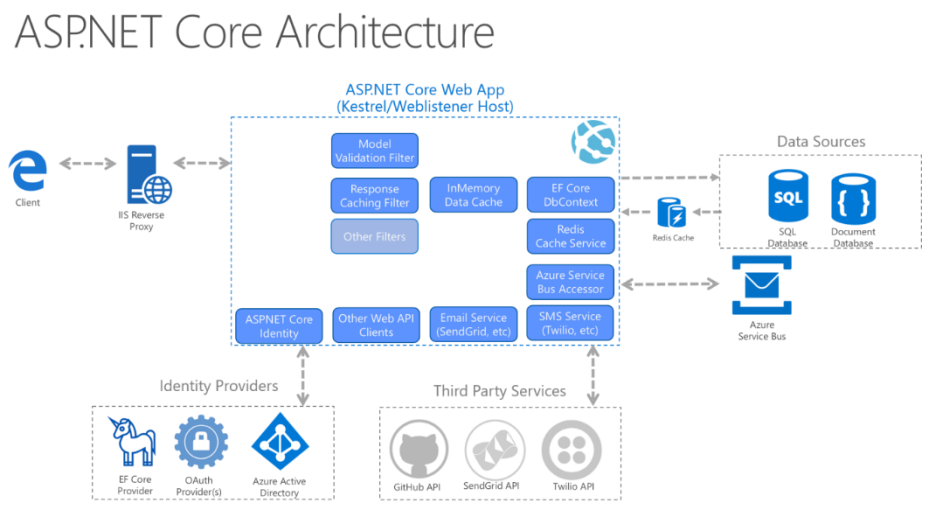


Figure 5.- Posible arquitectura aplicación ASP.NET Core

ASP.NET

ASP.NET es la extensión del framework .NET con herramientas y librerías para construir aplicaciones y servicios web con .NET y C# para cualquier plataforma (Linux, Windows, macOS…)

ASP.NET Web Apps

Permite el desarrollo de aplicaciones web basadas en HTML5, CSS y javascript. Estas aplicaciones son escalables, rápidas y seguras y se caracterizan por:

- Autenticación multifactor (MFA)
- Integración con autenticación externas como Google o Twitter
- Soporte para los protocolos estándar de autenticación
- Protección contra ataques XSS (cross-site scripting) o CSFR (cross-site request forgery)

A la hora de inicializar un nuevo proyecto ASP.Net permite:

- Utilización de UI frameworks como angular o react
- Blazor Webassembly (wasm) es un framework open source Web UI, y multi plaforma que permite el desarrollo de aplicaciones SPA utilizando .NET y C# en sustitución de javascript, generando aplicaciones que siguen los estándares web

En caso no utilizar un UI framework como angular o react, ASP.NET nos permite crear aplicaciones web utilizando el patrón de diseño MVC (Model-View-Controller) y utilizar sintaxis razor (sintaxis de marcado para insertar código .NET en páginas web).

ASP.NET Web APIs:

Con ASP.Net Web APIs se pueden desarrollar APIs de una manera sencilla para diferentes tipos de clientes con un mismo lenguaje. Sus principales características son:

- **Serialización**: Por defecto todas las respuestas de todos los endpoints se serializan en formato JSON pudiendo personalizar esta configuración
- **Definición de rutas utilizado varios tipos de configuración**: controladores, razor pages o middlewares
- **Autenticación y autorización**: los endpoints seguros definidos en la API utilizan mecanismos estándar como JWT. Pudiendo ser configurados y extendidos a través de ASP.NET Core Security
- **Procolo HTTPS por defecto**

Real-time

SignalR está incluido dentro del framework ASP.NET y permite utilizar comunicación en tiempo real (dashboards, juegos, notificaciones, chats,…) en nuestra aplicación:

Su desarrollo se ha basado en dos objetivos: rapidez y escalabilidad utilizando Redis, SQL Server o Azure Service Bus.

Igual que el resto del framework ASP.NET es de código abierto y la especificación del protocolo de comunicación es un protocolo abierto.

Testing

Una cuestión importante a la hora de desarrollar una aplicación es incluir los test necesarios a fin de asegurarnos que la aplicación funciona correctamente. Dentro del ecosistema .NET podemos utilizar las siguientes librerías xUnit, moq y coverlet.

xUnit permite escribir test unitarios o test funcionales para diferentes lenguajes como C#, F# o VB.NET. Proporciona dos atributos para identificar un método de test, sin ser necesario identificar la clase que los contiene:

- **Fact**: la ejecución del test siempre es “true”
- **Theory**: la ejecución del test solo es “true” en algunos escenarios

No existen atributos específicos para ejecutar código antes de la ejecución de los test o después de su ejecución. En caso de que sea necesario se debe utilizar un constructor sin parámetros e implementar la interfaz IDisposable.Dispose respectivamente.

Para verificar el resultado del test contra el resultado esperado, xUnit ofrece diferentes métodos en la clase Assert (Constraint Model): Equal, NotEqual, NotSame….

Moq es una librería (no pertenece a .NET foundation) que permite implementar mocks de métodos, propiedades, eventos o callbacks de manera sencilla.

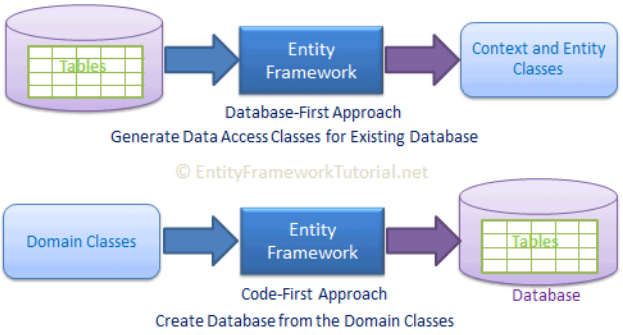
Coverlet: es una librería y perteneciente a la .NET foundation que permite obtener la cobertura de código de los test de una aplicación.

Persistencia - EntityFramework (EF)

EF es un framework disponible para acceder a diferentes motores de bases de datos (sql y no sql): mysql, postgresql, mongo, firebird…

Soporta dos flujos de desarrollo:

- **Code first**: EF crea la base de datos y elementos relacionados a partir de la configuración realizada por el desarrollador
- **Database first**: A partir de la base de datos, EF genera las clases necesarias para poder acceder a la base de datos



El acceso a los datos se realiza a través del modelo (model), un modelo es un conjunto de clases y objetos que representan a la base de datos y permite consultar y guardar datos.

El modelo puede crearse a partir de una base de datos existente o se programar el modelo manualmente para que coincida con la base datos. La configuración de un modelo se puede realizar de datos maneras:

- **Fluent API**: Sobrescribiendo el método OnModelCreating. Para evitar el tamaño del fichero se puede seperar en varias clases
- **Data annotations (atributos)**: el desarrollador puede añadir atributos a las propiedades de sus clases. En el caso de existir una configuración fluent API y data annotation, siempre se aplica la configuración fluent API

En el caso de que el modelo cambie, EF proporciona migrations para gestionar los cambios de datos. Cuando existe un cambio, el desarrollador utiliza EF Core tolos para crear las actualizaciones necesarias en la base de datos para mantener el esquema de base de datos actualizado.

Consulta de datos

LINQ es una de las opciones para realizar consultas a través de EF, permite utilizar el lenguaje de programación para escribir consultar fuertemente tipadas. EF envía la consulta LINQ al proveedor de base de datos que es el encargado en transformar la consulta en sintaxis SQL, ejecutarla y devolver los resultados obtenidos.

Guardado de datos

Cada instancia del contexto contiene un ChangeTracker que es el encargado de monitorizar los cambios que deben guardarse en la base de datos cuando se ejecuta el comando SavesChanges.

Feature toggle - Esquio

La librería esquio ha sido desarrollada por varios desarrolladores españoles y permite utilizar feature toggle en una aplicación .NET de manera rápida y sencilla así como en otros servicios como Azure devops.

Una vez referenciada la librería en el proyecto a utilizar Esquio ofrece varios toggles por defecto:

- **Environment variable**: Habilita la feature dependiendo del valor de una variable de entorno
- **Between dates:** Habilita la feature entre las fechas y horas indicadas
- **Country**: Habilita la feature si el país de usuario está incluido en la alista
- **Identity name**: Habilita la feature si el nombre del usuario esta incluido en la lista
- **Http Header value**: Habilita la feature si la clave existe y si el valor esta incluido en la lista

La configuración puede almacenarse en un fichero de configuración en formato json o en una base de datos. Para realizar cambios de configuración se proporciona esquio CLI y una interfaz web.

Computación en la nube

ARM (Azure resource manager)

Durante los últimos años muchos equipos de desarrollo han adoptado metodologías agiles en sus procesos de desarrollo, esto ha provocado la necesidad de poder desplegar sus aplicaciones en la nube.

Para ello se pueden automatizar los despliegues utilizando infraestructura como código (IaC) pudiendo definir la infraestructura necesaria para desplegar y ejecutar una aplicación. De esta manera se puede almacenar y versionar la infraestructura dentro de un repositorio de código fuente.

Azure resource manager (ARM) es la solución utilizada por Microsoft para implementar infraestructura como código para las soluciones que utilizan la plataforma Azure. Una plantilla ARM es un JSON (javascript object anotation) que define la infraestructura y configuración necesitaría para crear la infraestructura necesaria para ejecutar una o varias aplicaciones.

Las ventajas de utilizar ARM frente a otras tecnologías (ej. Terraform o ansible):

- Sintaxis declarativa
- Resultados repetibles
- Orquestación
- Ficheros modulares
- Creación de cualquier recurso de Azure
- Extensible
- Testing
- Preview
- Validación previa al despliegue
- Histórico de despliegues

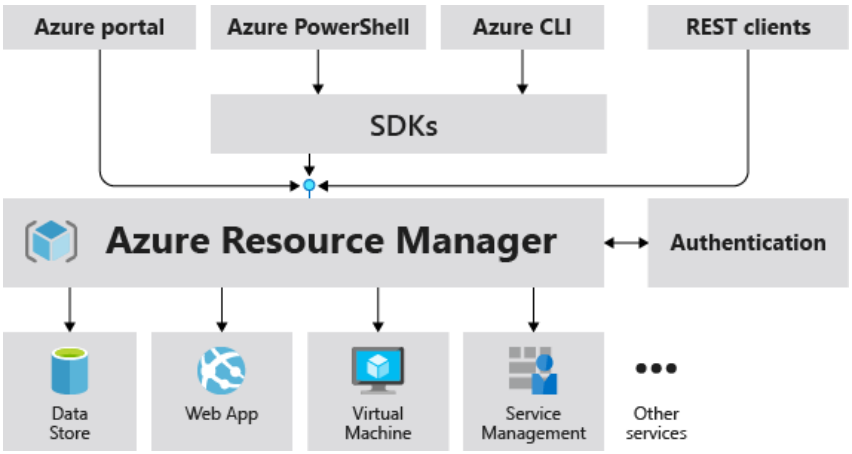


Figure 6.- ARM arquitectura

Azure serverless

Microsoft ofrece varias soluciones serverless dentro de su catálogo de servicios a fin de minimizar el uso de infraestructura para las aplicaciones: Azure Functions o Serverless Kubernetes.

Azure functions

Permite ejecutar código .net, node, java, PowerShell o Python a partir de un evento. Algunos de estos eventos son los siguientes:

- **Blob storage**: Ejecuta una función ante un cambio en un blog storage
- **Azure cosmos DB**: Ejecuta una función ante cambios en una base de datos Azure Cosmos DB
- **Http**: La ejecución de la función se inicia por una petición http
- **RabbitMQ, Azure queue storage**: Ejecuta una función cuando se crea un mensaje en una cola RabbitMQ o Azure queue storage

Con Azure Functions podemos cubrir diferentes escenarios tales como procesar archivos cuando se almacenan, ejecutar tareas programadas, analizar datos o procesar información sin necesidad de tener un proceso arrancado consumiendo diferentes recursos.

Las ventajas de utilizar Azure Funtions frente a otras soluciones:

- Escalado automatizado y flexible
- No es necesario mantener servidores
- Integración basada en eventos
- Experiencia de desarrollo completa: compilación, depuración, despliegue y monitorización
- Conectividad con otros servicios

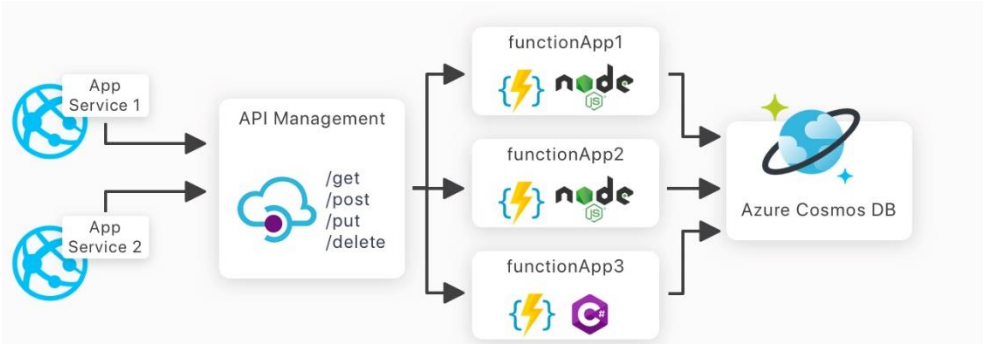


Figure 7.- Posible escenario Azure functions

Serverless kubernetes

KEDA (Kubernetes-based Event Driven Autoscaling) es una herramienta desarrollada por Microsoft y RedHat que ofrece un mecanismo de escalado desde cero a cientos de instancias de manera proactiva en un clúster de kubernetes.

Puede utilizar diferentes fuentes de datos (eventos) como apache Kafka, AWS CloudWath, Azure Monitor, Promethus, Redis o mysql entre otros.

KEDA realiza dos funciones:

- Activa o desactiva un deployment en el caso de existan o no exista uno o varios eventos
- Proporciona un KMS (kubernetes Metris Server) con información recopila a partir de las fuete de datos

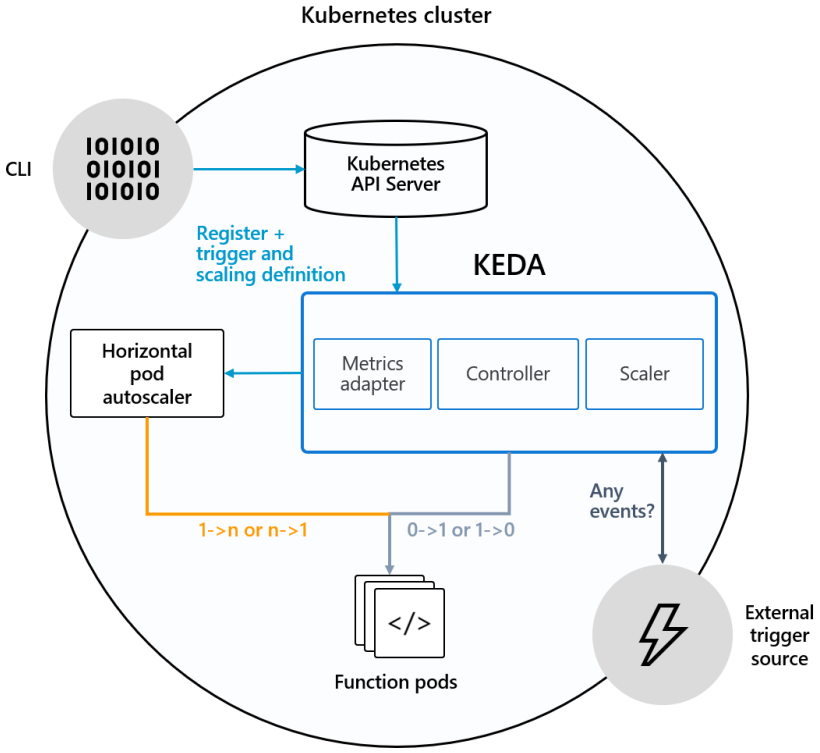


Figure 8.- Arquitectura KEDA

Para realizar sus funciones utiliza tres componentes:

- **Scaler**: Realiza la conexión y monitorización de eventos
- **Operator (agent)**: Encargado de activar un deployment y configurar el HPA

- **Metrics**: ofrece al HPA la información necesaria para que pueda auto escalar los deployments de manera correcta

Para realizar la configuración del escalado se utiliza custom resource (CRD) de tipo ScaledObject donde se parametriza la fuente de datos, los parámetros de escalado y el deployment que debe escalar.

```yaml
apiVersion: keda.k8s.io/v1alpha1
kind: ScaledObject
metadata:
  name: redis-scaledobject
  namespace: default
  labels:
    deploymentName: processor
spec:
  scaleTargetRef:
    deploymentName: processor
  pollingInterval: 20
  cooldownPeriod:  200
  minReplicaCount: 0
  maxReplicaCount: 50
  triggers:
  - type: redis
    metadata:
      address: redis:6739
      listName: jobs
      listLength: "20"
    authenticationRef:
      name: redis-auth-secret
```

Figure 9.- yaml configuración KEDA

Microservicios

A día de hoy una de las arquitecturas más utilizadas a la hora de desarrollar aplicaciones en la nube es la arquitectura de microservicios. Se puede definir un microservicio como un servicio pequeño y autónomo que interactúa con otros microservicios.

Uno de los primeros stack de microservicios fue el stack de Netflix y fue adotado por Spring bajo el nombre spring-cloud-netflix. Está compuesto por los siguientes elementos:

- **config-server**: servicio que externaliza la toda la configuración de los microservicios
- **eureka**: servicio para el registro de microservicios
- **zuul**: servicio que actuala como api gateway y que actúa como punto de entrada
- **hystrix**: servicio para la gestión de errors utilizando el patron circuit breaker

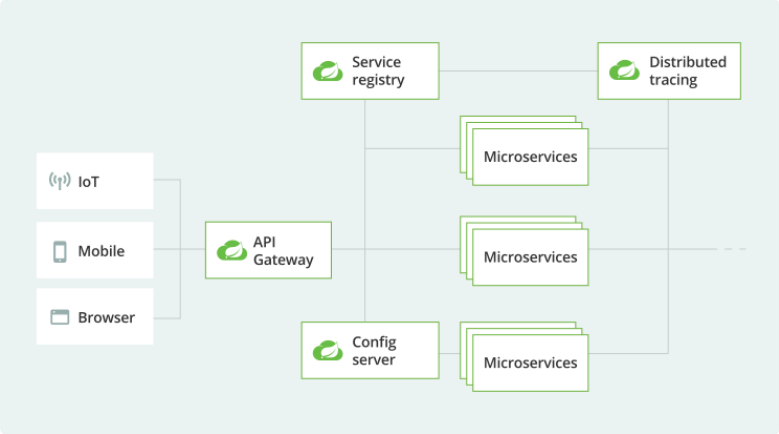


Figure 10.- Arquitectura microservicios Spring Cloud

Steeltoe

Ofrece una serie de librerías cuyo objetivo es integrarse con los servicios Spring Cloud necesarios en una arquitectura de microservicios:

- Configuration providers (Spring Cloud, Vault, etc.)
- Service Discovery client (Netflix Eureka, etc.)
- CircuitBreaker (Netflix Hystrix, etc.)
- Management

Azure Spring Cloud

Azure Spring Cloud (preview – noviembre de 2020) es un servicio que facilita el despliegue de soluciones basadas en microservicios (java spring boot y ASP.NET Core Steeltoe) sin realizar ningún cambio en el código Fuente de las aplicaciones.

- **Spring Cloud Config Server**: provee de un servicio externo distribuido con la configuracion para los diferentes microservicios
- **Blue/green deployments**: Los desarrolladores pueden desplegar nuevos cambios en los microservicios configurando una estrategia de despliegue blue / green
- **CI/CD pipeline automation**: Integracion con el sistema CI/CD Azure develops utilizando Azure CLI
- **Application scaling**: Permite escalar y configurar la memoria y CPU de manera sencilla un microservicio
- **Application monitoring**: El sistema de logs distribuido permite monitorizar las comunicaciones entre los diferentes microservicios. Spring Cloud Sleuth se integra con Azure Application Insights

Contenedores y orquestadores

Tye

Es un proyecto de código abierto y experimental, el objetivo de esta herramienta es el facilitar el desarrollo, testeo y despliegue de microservicios y aplicaciones distribuidas basadas en NetCore.

Algunas funcionalidades que nos ofrece:

- Ejecutar varios servicios con un único comando
- Utilizar dependencias
- Contenerización automática de aplicaciones .NET
- Creación de manifiestos de kubernetes con un mínimo de configuración
- Utilización de la misma configuración para desarrollo / producción (consistencia)

Una de las mayores ventajas que presenta tye frente a docker compose es que no es necesario escribir ningún fichero Dockerfile o yaml para utilizarlo, tye ejecutará todas las aplicaciones NetCore de la solución en puertos aleatorios.

En caso de necesitar infraestructura adicional como redis o un sqlserver se pueden utilizar contenedores docker y realizar las configuraciones correspondientes a través de un fichero yaml.

Tye ofrece un dashboard donde se pueden comprobar todos los servicios levantados asi como su configuración o logs. Tambien hay disponible un servicio de discovery en el que podemos obtener las urls de los diferentes servicios que hay levantados

Container instances (ACI)

Container instances permite desarrollar y desplegar aplicaciones sin necesidad de administrar máquinas virtuales ni aprender otras tecnologías para desplegar y desarrollar aplicaciones en la nube.

ACI puede arrancar un contenedor en cuestión de segundos utilizando imágenes Linux o Windows desde docker hub o desde un registro de contenedores privado. El acceso a los contenedores puede realizarse a través de una dirección IP o un nombre de dominio, también se puede acceder a la consola del contenedor a fin de revisar o solucionar problemas durante el desarrollo.

Se puede realizar el despliegue desde diferentes herramientas como az cli, PowerShell, ARM template o la propia línea de comandos de docker.

Container groups: es un grupo de contenedores que ejecutan en la misma maquina host y que comparten el mismo ciclo de vida, red o recursos.

Seguridad: Microsoft recomienda diferentes acciones a tener en cuenta por un usuario cuando utiliza ACI (basado en Azure Security Benchmark version 1.0,):

- Protección de los recursos a través de reglas de red o utilizando Azure firewall
- Monitorización del tráfico de red (Azure security center monitoring)
- Protección de las aplicaciones web (Web application firewall - WAF)
- Gestiona del tráfico de red: utilización application Gateway, utilización https,…

Red: Azure virtual network proporciona una red privada para interconectar diferentes servicios en Azure y conectar servicios on-premise

ACI puede utilizarse en diferentes escenarios tales como procesado de datos, integración continua o un sistema de eventos.

AKS

Azure Kubernetes Service (AKS) facilita el despliegue de un clúster kubernetes como servicio en la nube de Azure reduciendo la complejidad de su administración y facilitando las tareas de mantenimiento.

La creación de un clúster AKS se puede realizar desde el portal de Azure, Azure CLI, ARM o terraform pudiendo configurar los parámetros de red, integración con directorio activo o monitorización.

AKS puede integrarse con el directorio activo y utilizar RBAC (kubernetes role-based Access control) para gestionar el acceso y permisos a los diferentes recursos y namespaces del clúster. La monitorización de los contenedores, nodos, controladores como la información de los logs de los nodos master puede realizarse a través de Azure monitor.

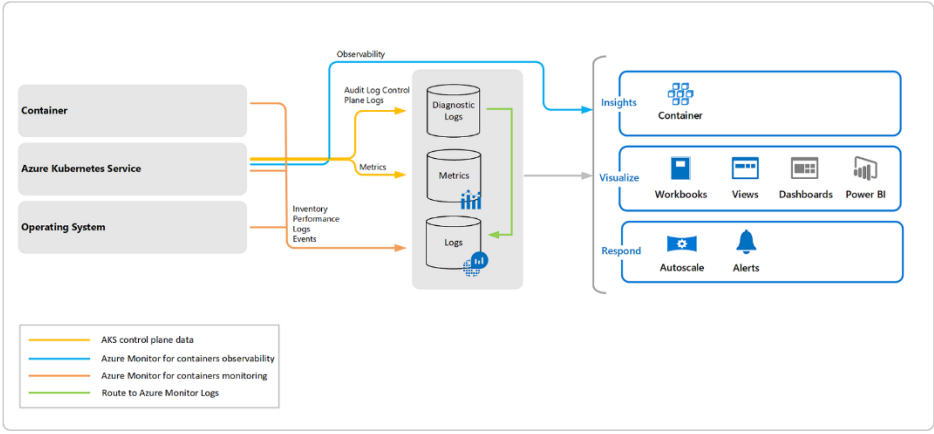


Figure 11.- Monitorización AKS service

Los nodos del clúster son ejecutados sobre máquinas virtuales a los que se pueden conectar a diferentes configuraciones de almacenamiento, configurar la utilización de GPUs para uso intensivo de procesamiento, gráficos o visualización. Los sistemas operativos disponibles para un node pool son Windows Server 2019, Linux o Ubuntu.

Actualmente en modo preview (noviembre de 2020) se encuentra la utilización de nodos confidenciales basados en la tecnología Intel SGX.

A la hora de crear un nuevo clúster, se permite realizar cambios de configuración de red o almacenamiento. A nivel de red existen dos opciones:

- **Kubenet**: Es la opción por defecto a la hora de crear un nuevo clúster. A cada nodo se le asigna una dirección IP de la red y cada uno de los pods desplegados se le asigna una dirección IP en un rango diferente. Para comunicar los diferentes elementos se utiliza un NAT (network address translation)
- **Azure Container Networking Interface (CNI) networking:** Todos los recursos desplegados en el clúster utilizan los recursos y configuraciones de red asociadas a red asignada

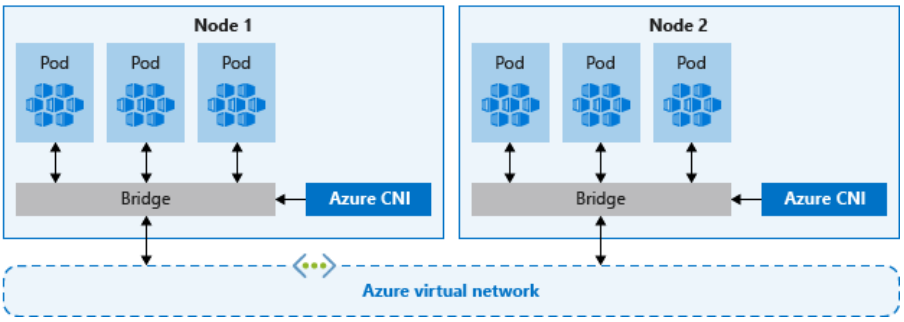


Figure 12.- AKS Azure CNI

AKS proporciona dos tipos de almacenamiento para que las aplicaciones desplegadas en el clúster puedan almacenar información en caso de que lo necesiten:

- **Azure disks**: Son utilizados cuando se crean recursos del tipo Kubernetes DataDisk. Estos pueden ser tipo Premium que por defecto utilizan discos SSD de alto rendimiento
- **Azure files**: Permite compartir información entre los diferentes pods a través de SMB 3.0

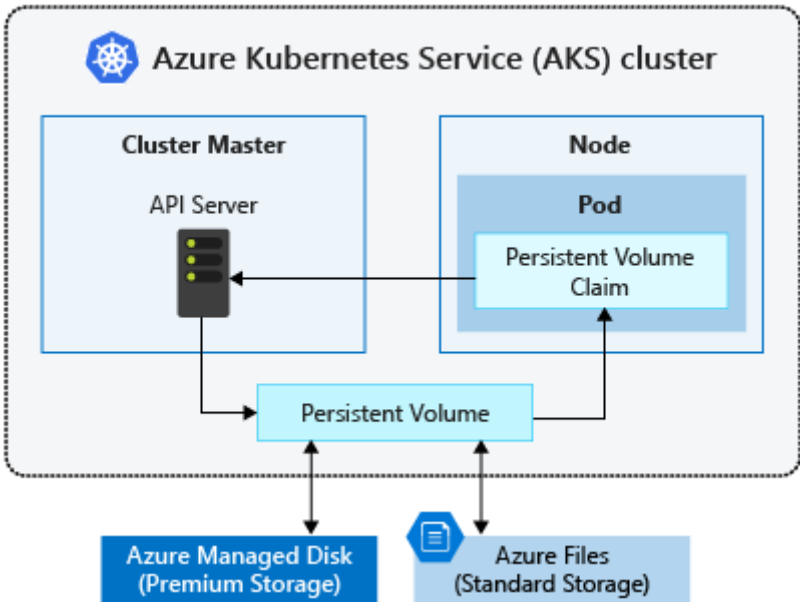


Figure 13.- AKS tipos almacenamiento

Herramientas desarrollo

A día de hoy las aplicaciones NetCore pueden desarrollarse con diferentes herramientas y diferentes sistemas operativos (Linux, MacOS o Windows):

- **Línea comandos**: CLI (Command Line Interface) permite desarrollar aplicaciones (crear, compilar, …) en cualquier sistema operativo
- **Visual Code**: Disponible para Linux, MacOS y Windows. Requiere la instalación de las extensiones necesarias
- **Visual Studio**: IDE solo disponible para Windows. Su principal característica es que tiene integración completa para el desarrollo de aplicaciones .NET
- **OmniSharp**: Permite el desarrollo de aplicaciones .NET desde los editores: Atom, Brackets, Slublime Text, Emacs o Vim

Bridge to kubernetes

Permite redirigir tráfico desde el clúster de kubernetes donde está desplegada nuestra aplicación hasta la máquina del desarrollador, de esta manera podemos desarrollar y depurar un servicio localmente mientras interactúa con el resto de servicios desplegados en el clúster. Algunas de las ventajas que ofrece bridge to kubernetes son:

- **Simplifica el desarrollo de microservicios**: no es necesario desplegar en la máquina del desarrollador todos los elementos de la aplicación
- **Fácil depuración**: Los desarrolladores pueden utilizar sus herramientas y perfiles de desarrollo ya que la depuración se realiza en la máquina del desarrollador
- **Desarrollo y testeo en un escenario end-to-end: Los** desarrolladores pueden realizar pruebas end-to-end utilizando la aplicación desplegada en el clúster

Para utilizar esta funcionalidad es necesario utilizar:

- Visual Studio 2019 16.7 preview 4 o superior o Visual Code
- Extensión bridge to kubernetes

- Azure CLI

Cuando se inicia el servicio Bridge to Kubernetes:

- Se selecciona el servicio que va a reemplazarse el clúster
- El agente instalado en el clúster redirige el tráfico desde el clúster hasta la máquina del desarrollador. Para ello ejecuta y abre una conexión desde la máquina de desarrollador hasta el agente utilizando kubectl port-forward
- Posteriormente, se configura la herramienta de desarrollo con la información recopilada en el clúster y en la máquina de desarrollo, actualizándose así el fichero host files– ¡Importante! Son necesarios privilegios de administrador
- Finalmente, se arranca la aplicación en modo depuración

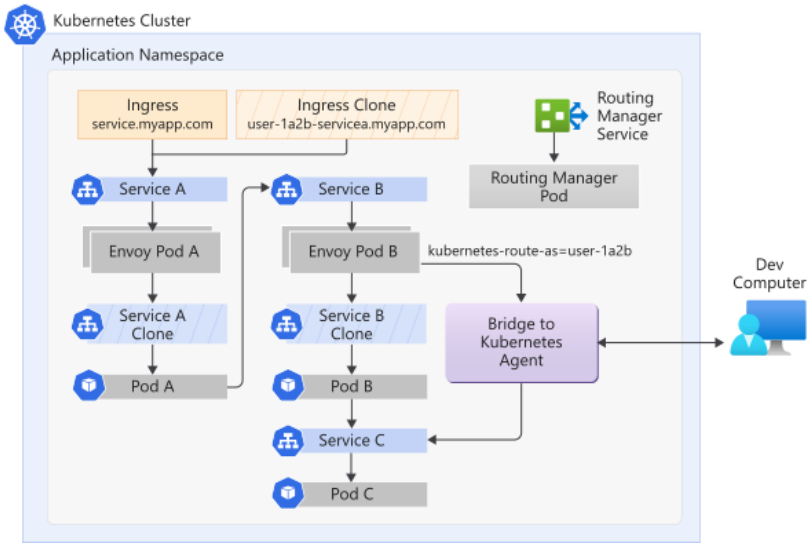


Figure 14.- Diagrama funcionamiento bridge to kubernetes

Integración y despliegue continuo

Azure devops es una herramienta cubre todo el ciclo de vida del desarrollo de software.

Azure pipelines

Es el servicio cloud que compila, prueba y despliega de manera automática cualquier tipo de aplicación desarrollada en cualquier lenguaje.

La definición de una pipeline se realiza en sintaxis yaml y está versionada junto con el código fuente de la aplicación.

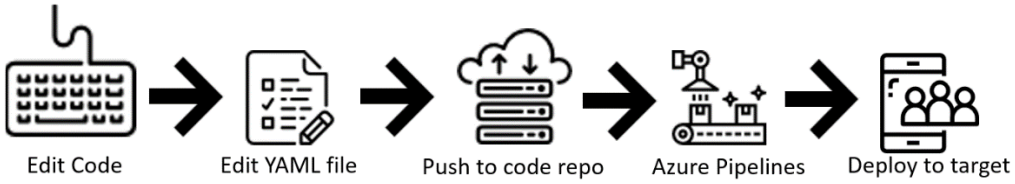


Figure 15.- CI/CD Azure pipelines

Los elementos que componen una pipeline son:

- **Trigger**: es el desencadenador que indica la ejecución de una pipeline (ej.: push a un repositorio)
- **Pipeline**: define el proceso de integración continua (CI) y está compuesta de una o más stages
- **Stage**: está compuesto de uno o más jobs
- **Job**: agrupa un conjunto de steps y se ejecutan en un mismo agente
- **Step**: es el elemento más pequeño, puede ser un script o una tarea. La tarea puede ser predefinida o puede ser una tarea personalizada
- **Artefact**: colección de ficheros generados por una ejecución de una pipeline (Run)
- **Run**: Ejecución de una pipeline
- **Enviroment**: es una colección de recursos: contenedores, máquinas virtuales, web apps, etc… donde la aplicación será desplegada

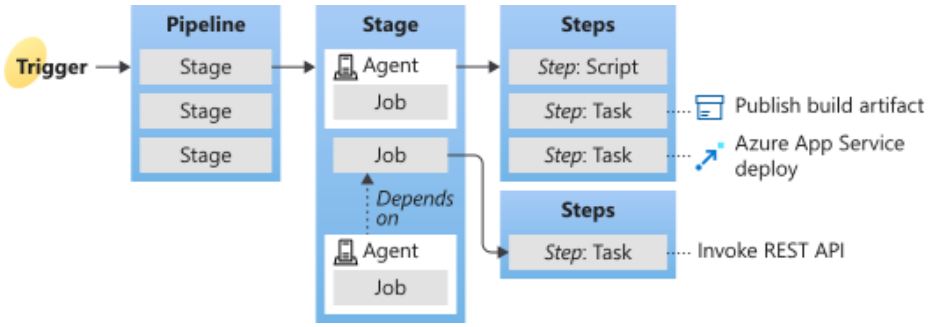


Figure 16.- Diagrama ejemplo Azure pipeline (CI/CD)

Azure boards

Permite a los equipos de desarrollo tener una única herramienta para gestionar todo el ciclo de desarrollo de software. Azure boards permite personalizar Scrum y Kanban con sus flujos de trabajo personalizados. Otras ventajas de su uso son:

- Facilidad de uso y personalización
- Personalización de dashboards
- Integración con otras herramientas: github, office…

Azure Artifacts

Permite crear y compartir librerías NuGet, npm o Maven con los equipos de desarrollo e integrarse con procesos CI/CD

Azure test plans

Provee de una herramienta donde los diferentes equipos pueden realizar seguimiento de los planes de test, test de aceptación o test exploratorios. Para la utilización de Azure test plan, es necesario realizar una suscripción de pago.

- **Planes manuales de test**: tests desarrollados por los equipos de desarrollo
- **Test de aceptación (UAT):** test de aceptación desarrollados para comprobar que la funcionalidad entregada coincide con los requerimientos definidos
- **Test de exploración y Stakeholder feedback**: es responsabilidad de todo el equipo de desarrollo (desarrolladores, product owners, UX,…) colaborar en la creación y

mantenimiento de los test. A través de un plugin instalado en el navegador pueden crear test diferentes test de exploración navegando a través de la aplicación

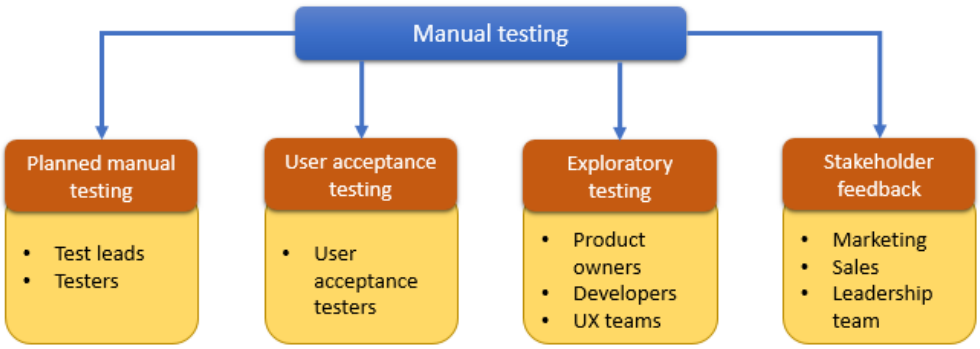


Figure 17.- Tipos de test y usuarios

Conclusión y próximos pasos

En la ejecución del trabajo de fin de master hemos podido analizar e implementar todas las partes del master con tecnologías .net y afines:

- Aplicaciones web y aplicaciones nativas en la nube: web api, web apps
- Acceso a datos: bases de datos sql y no sql
- Testing
- Feature flags
- Computación en la nube
- Contenedores y orquestadores
- Devops

Tan solo hemos encontrado un punto donde Microsoft ya no ofrece ninguna herramienta, pruebas de carga. Esta herramienta no continúa desarrollándose desde el año 2019 y Microsoft recomienda la utilización de otras herramientas como Apache JMeter.

La implementación de estas tecnologías puede encontrarse en el repositorio de GitHub MasterCloudApps-Projects/DotNetFullMasterStack: https://github.com/MasterCloudApps-Projects/DotNetFullMasterStack donde se puede encontrar la documentación técnica necesaria para ejecutar la aplicación desarrollada.

Podemos destacar que a día hoy Microsoft ha cambiado su rumbo y está apostando fuertemente por el open source y por la comunidad por lo que los frameworks y herramientas de desarrollo como visual code cada día son más populares.

Algunas herramientas que no se han podido analizar y que son muy interesantes para poder ampliar el trabajo fin de master serian:

- Dapr (Distributed application runtime)
- Microservices, Domain-Driven Design and Entity Framework Core
- Azure service operator
- Identity server
- Xamarin CI/CD
- Appcenter

Bibliography

[1] «NET_Framework,» 11 2020. [En línea]. Available: https://en.wikipedia.org/wiki/.NET_Framework.

[2] «What is dotnet,» 11 2020. [En línea]. Available: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework.

[3] «NET_Core,» 11 2020. [En línea]. Available: https://en.wikipedia.org/wiki/.NET_Core.

[4] «Mono_sw,» 11 2020. [En línea]. Available: https://en.wikipedia.org/wiki/Mono_(software) .

[5] 11 2020. [En línea]. Available: https://dotnetfoundation.org/.

[6] «ASP.NET,» 11 2020. [En línea]. Available: https://dotnet.microsoft.com/apps/aspnet .

[7] «ASP.NET blazor,» 11 2020. [En línea]. Available: https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor .

[8] «Razor,» 11 2020. [En línea]. Available: https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-5.0 .

[9] «Security,» 11 2020. [En línea]. Available: https://docs.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-5.0.

[10] «techempower,» 11 2020. [En línea]. Available: https://www.techempower.com/benchmarks/#section=data-r19&hw=ph&test=plaintext.

[11] «Moq,» 11 2020. [En línea]. Available: https://github.com/moq.

[12] 11 2020. [En línea]. Available: https://xunit.net/.

[13] «cmdline,» 11 2020. [En línea]. Available: https://xunit.net/docs/getting-started/netcore/cmdline.

[14] «compare,» 11 2020. [En línea]. Available: https://xunit.net/docs/comparisons .

[15] «nunit Constraint-Model,» 11 2020. [En línea]. Available: https://github.com/nunit/docs/wiki/.

[16] «Unit testing coverage,» 11 2020. [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-code-coverage?tabs=windows.

[17] «coverlet,» 11 2020. [En línea]. Available: https://dotnetfoundation.org/projects/coverlet.

[18] «vstest datacollector,» 11 2020. [En línea]. Available: https://github.com/Microsoft/vstest-docs/blob/master/docs/extensions/datacollector.md.

[19] «EF core,» 11 2020. [En línea]. Available: https://docs.microsoft.com/en-us/ef/core/.

[20] «EF core cli,» 11 2020. [En línea]. Available: https://docs.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli.

[21] «Esquio,» 11 2020. [En línea]. Available: https://github.com/Xabaril/Esquio.

[22] 11 2020. [En línea]. Available: https://keda.sh/ .

[23] «Announcing Keda,» 11 2020. [En línea]. Available: https://cloudblogs.microsoft.com/opensource/2019/05/06/announcing-keda-kubernetes-event-driven-autoscaling-containers/.

[24] «Smart Scaling of Event-Driven Applications in Azure Kubernetes,» 11 2020. [En línea]. Available: https://www.youtube.com/watch?v=x6VfMcLFs4o&list=PLhiJRVuS9xnLZnnkWLrnkraRPDz2i6HES&index=5 .

[25] «Serverless,» 11 2020. [En línea]. Available: https://azure.microsoft.com/en-us/solutions/serverless/#solutions.

[26] «Spring cloud,» 11 2020. [En línea]. Available: https://docs.microsoft.com/en-us/azure/spring-cloud/spring-cloud-quickstart?tabs=Azure-CLI&pivots=programming-language-java.

[27] «steeltoe,» 11 2020. [En línea]. Available: https://thenewstack.io/steeltoe-modernize-net-apps-for-a-microservices-architecture/.

[28] «steeltoe,» 11 2020. [En línea]. Available: https://www.infoq.com/br/articles/netflix-oss-steeltoe/.

[29] «Tye,» 11 2020. [En línea]. Available: https://devblogs.microsoft.com/aspnet/introducing-project-tye/.

[30] «Tye,» 11 2020. [En línea]. Available: https://github.com/dotnet/tye.

[31] «Container instances,» 11 2020. [En línea]. Available: https://azure.microsoft.com/en-gb/services/container-instances/.

[32] «Azure container instances now generally available,» 11 2020. [En línea]. Available: https://azure.microsoft.com/es-es/blog/azure-container-instances-now-generally-available/.

[33] «AKS,» 11 2020. [En línea]. Available: https://azure.microsoft.com/en-us/services/kubernetes-service/.

[34] «ASP.net core architecture overview,» 11 2020. [En línea]. Available: asp-dot-net-Core-architecture-overview.

[35] «dotnet architecture,» 11 2020. [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures.

[36] «Introducing net 5,» 11 2020. [En línea]. Available: https://devblogs.microsoft.com/dotnet/introducing-net-5/.

[37] «.NET Core tools,» 11 2020. [En línea]. Available: https://dotnet.microsoft.com/platform/tools.

[38] «AKS - Network,» 11 2020. [En línea]. Available: https://docs.microsoft.com/en-us/azure/aks/concepts-network.