Microservicios en Azure

- Jesús Villalobos
- Responsable de desarrollo
- jvillalobos@certia.net
- @CertiaJesus

Agenda

- Arquitectura de microservicios
- Microservicios en Azure
- Servicios orientados a diseño
  - DEMO: Creación de una Logic App
- Servicios orientados a código
  - DEMO: Creación de una Azure Function
- Creación de microservicios desde Visual Studio
  - DEMO: Crear una Azure Function desde Visual Studio

ARQUITECTURA DE MICROSERVICIOS

Aplicaciones distribuidas en Internet

- Existe la necesidad de interconectar diversos sistemas heterogéneos.
- Los móviles son el cliente de servicios más utilizado en todo el mundo.
- La comunicación ha de ser rápida, ligera, concreta.
- La seguridad ha de ser simple pero efectiva.
- Los servicios web basados en SOAP y XML fueron un buen comienzo pero es necesario superar sus limitaciones.

Microservicios

- Pequeñas piezas de software que realizan tareas muy concretas y que se interconectan entre sí.
- La arquitectura de microservicios busca reunir ciertas características:
  - Implementación en cualquier sistema
  - Concisión, rapidez
  - Han de hablar entre ellos independientemente del lenguaje
  - Han de poder comunicarse con sus plataformas si es necesario
- La intercomunicación se realiza mediante JSON y HTTP(S)

Infraestructura

- Cualquier sistema operativo puede albergar microservicios si puede exponer un puerto HTTP(S).
- Hay consideraciones que son obligatorias
  - Seguridad de acceso
  - Protección ante ataques de DoS o de cualquier otro tipo
  - Escalado vertical y horizontal
  - Registro, gestión, entornos de desarrollo…

Desarrollo

- Los microservicios deben comunicarse entre sí sin importar el lenguaje en que estén desarrollados.
- Deben poder desarrollarse en cualquier lenguaje moderno.
- Deben permitir el acceso a APIs del sistema
- Referencias externas mediante paquetes (NuGet, NPM…)
- Deben contemplar la posibilidad de realizar ejecuciones asíncronas (p. ej. para cálculos largos)

Comunicaciones

- HTTP(S) como protocolo de transmisión
- JSON como soporte de información
- Accesibles a toda la Internet para que se puedan utilizar desde móviles, IoT, etc.
- Seguridad mediante tokens de acceso y encriptación por certificado
- Los microservicios deben poder comunicarse entre sí y orquestarse también mediante JSON y HTTP(S)

Escenarios

- Aplicaciones para móviles, IoT, etc.
- Publicación de APIs para clientes
- Flujos de información
- Interacción con servicios de terceros (redes sociales, DevOps, etc.) mediante captura o disparo de eventos.
- Escalado de servicios clave
- Cálculos largos, IA, procesado de *big data*.

MICROSERVICIOS EN AZURE

Servicios sin servidor

- Azure nos permite crear máquinas virtuales con cualquier sistema operativo, pero para una arquitectura de microservicios tal vez sea demasiado.
- Azure nos permite crear arquitecturas de microservicios “sin servidor”
  - En realidad sí que hay servidor
- Colocaremos los servicios en grupos de recursos.

Grupos de recursos

- Cualquier elemento de Azure, desde una BD hasta una VM, es un recurso.
- Los recursos pueden agruparse para unificar la facturación por uso o por otros criterios (geográficos, etc.)
- El recurso solo se paga cuando se usa
- La facturación puede variar mucho en función del HW subyacente.
  - El HW asignado puede escalarse horizontalmente o verticalmente.

Infraestructura

- Azure maneja internamente los detalles de implementación de los recursos
  - El disco duro, los procesadores, la RAM, las redes
- También nos proporciona mecanismos de seguridad mediante tokens de acceso.
- Tenemos un registro exacto del funcionamiento y del uso de cada recurso.
- Sabemos cuánto cuesta cada recurso y tendremos previsiones de gasto

Tipos de microservicios (I)

- **Orientados al diseño**
  - Los elementos se crean mediante diagramas de flujo o asistentes.
  - El código subyacente no es accesible o no suele emplearse.
  - Idóneos para expertos en lógica de negocio que no tienen conocimientos de programación.
  - Se crean a partir de plantillas y existen multitud de plantillas
  - **Logic Apps**, **Power Automate**

Tipos de microservicios (II)

- **Orientados al código**
  - Los elementos se crean mediante código
    - C#, JS, Python, Powershell
  - Requieren conocimientos de desarrollo
  - El código puede desarrollarse en la propia plataforma de Azure o en Visual Studio.
  - **Azure Functions, WebJobs**

SERVICIOS ORIENTADOS A DISEÑO

Logic Apps

- Mediante Logic Apps podemos crear servicios que respondan a eventos desencadenados por otros servicios o bien sujetos a una temporización.
- Logic Apps dispone de plantillas prediseñadas para muchos tipos de operación, y también hay cientos de conectores para enlazar con otros servicios.
  - Desde servicios de correo electrónico a bases de datos pasando por calendarios, sistemas de ficheros, etc.

Diseñador de flujo

- La Logic App se desarrolla principalmente en un diseñador de flujo.
- Podemos poner elementos de control
  - Cláusulas condicionales, bucles
- Podemos crear ejecuciones en paralelo
- Los elementos poseen una salida que podemos detectar y crear conmutadores basados en ellas.

Código y configuración

- Podemos ver el código de la Logic App, pero no es más que objetos de JSON.
  - No hay código imperativo asociado
- También podemos crear parámetros específicos
- Los parámetros son también clases JSON

Desencadenantes

- La App se disparará como consecuencia de un desencadenante (*trigger*).
- El desencadenante puede ser un evento de un servicio, una llamada HTTP, un webhook de Azure o de otro servicio…
- También podemos ejecutar la Logic App de forma programada
  - Las programaciones en Azure emplean sintaxis CRON para determinar la programación.

Power Automate

- Power Automate se parece a Logic Apps pero está pensada sobre todo para usuarios sin ningún conocimiento de programación.
- Son pequeños servicios que ejecutan una tarea determinada como respuesta a un suceso.
- También poseen estructuras de control sencillas.
- Existen docenas de conectores, la mayor parte gratuitos.

Logic Apps vs. Power Automate

- Power Automate solo se puede utilizar desde la web, mientras que Logic Apps puede usarse desde Visual Studio.
- PA pertenece a Office 365 mientras que LA está dentro de Azure. Power Automate puede exportarse a Logic Apps.
- La integración con aplicaciones en Office es más sencilla con PA ya que podemos crear un botón para ejecutarla.
  - Es más integrable con Sharepoint o Teams.
- Logic Apps es más rápido y es más integrable con servicios B2B

Demo: Creación de Logic Apps

SERVICIOS ORIENTADOS A CÓDIGO

Azure Functions

- AF es el mecanismo más potente para la creación de microservicios en Azure.
- Una AF es una función escrita en cualquiera de los lenguajes aceptados que se ejecuta como consecuencia de un evento que la desencadena.
- Podemos escribirla en C#, JS, Powershell, Java, Python…
- Podemos crear la AF directamente en el portal de Azure o bien crearla en Visual Studio y luego desplegarla.

Creación e implementación

- Las AF se agrupan en una aplicación de funciones que puede tener más de una AF.
  - Eso permite compartir recursos o tokens de acceso
- La AF se crea a partir de una plantilla estándar que determina el desencadenador que la disparará.
- La creación genera dos ficheros que pueden editarse desde el portal:
  - El fichero de código en el lenguaje que corresponda.
  - Un fichero llamado **function.json** que contiene la configuración y la parametrización de la función.

Ejecución y depuración

- Podemos ejecutar la AF directamente desde el portal.
  - Se nos proporciona un interfaz para ejecutar el desencadenador si es una llamada HTTP o una respuesta a un evento.
- Existen métodos específicos para dejar un registro de la ejecución o para rastrear los errores.
- Podemos saber el tiempo de ejecución, el tiempo de respuesta, detenerla, etc.

Funciones duraderas

- Por lo general las AF no poseen estado.
  - Las funciones no conservan datos entre llamadas.
- Es posible crear **funciones duraderas** que conservan el estado.
- Estas AF se emplean en escenarios muy concretos.
  - Flujos de supervisión.
  - Encadenamiento y orquestación de funciones.
  - Llamadas asincrónicas.
  - Instanciación Singleton.

Orquestación y comunicaciones

- Una AF puede comunicarse, tanto de entrada como de salida, con cualquier otro elemento.
  - Logic Apps, Power Automate, WebJobs, otras Azure Functions.
- La comunicación se realiza mediante parámetros de entrada y salida que se mapean como objetos JSON.

WebJobs

- Los WebJobs son servicios integrados dentro del servicio genérico AppService, que permite poner en marcha aplicaciones web sin necesidad de “servidor”.
  - Un WebJob sería una aplicación web especializada en dar un servicio concreto o en ejecutar una tarea en segundo plano.
  - De momento no es utilizable en Linux o .NET Core pero es previsible que lo será.

Estructura de un WebJob

- Un WebJob es un fichero que se ejecuta dentro del contexto de una App Service de Azure.
- El ejecutable puede ser casi de cualquier tipo
  - *cmd, bat, exe*
  - ¡También en VB.NET! ☺
  - Script de Powershell
  - js, php, java, py, etc…
- El proceso puede ejecutarse de manera continua o como consecuencia de un desencadenador temporal.
  - Mediante una expresión CRON

Azure Functions vs. WebJobs

- Normalmente es mejor utilizar Azure Functions, pero en algunos casos podemos querer crear en su lugar un WebJobs
  - Queremos personalizar la ejecución del servidor para ofrecer características concretas, como la posibilidad de realizar reintentos.
  - Queremos integrar el WebJobs en una web existente que se está ejecutando en un App Service.

Demo: Creación de una Azure Function

CREACIÓN DE MICROSERVICIOS DESDE VISUAL STUDIO

Microservicios desde Visual Studio

- Cualquier elemento de Azure puede crearse, configurarse y desplegarse desde Visual Studio.
  - Eso incluye bases de datos, máquinas virtuales, grupos de recursos…
- Es importante emplear Visual Studio 2019 ya que es la versión que incorpora más características de integración con Azure.
  - Hay un explorador de Azure especializado que permite realizar cualquier operación.

Desarrollo y depuración

- Debemos instalar el SDK de Azure en nuestra instalación de Visual Studio.
- Dependiendo del proyecto que queramos crear tendremos que crear alguna infraestructura en local o no.
  - Por ejemplo, una Azure Function puede necesitar una cuenta de almacenamiento.
- La depuración se realiza ejecutando el código en local.
  - Podemos poner break points.

Ventajas del uso de Visual Studio

- Podemos emplear versionado de código y control de código fuente.
  - Azure DevOps, Git
- No hay un enlace fijo entre el proyecto y la implantación, así que podemos desplegar un mismo proyecto en distintas cuentas de Azure.
- En el caso de las Azure Functions, no se sube el código fuente sino el compilado, lo cual mejora la seguridad (aunque no se pueda editar luego el código desde el portal de Azure).

Demo: Crear una Azure Function desde Visual Studio

¡Gracias por asistir!

No os perdáis nuestro próximo webinar sobre Visualizaciones en Power BI
https://www.certia.net/webinar-power-bi/

- Jesús Villalobos
- Responsable de desarrollo
- jvillalobos@certia.net
- @CertiaJesus