Windows Azure

La nube, versión Microsoft

El pasado 4 de enero, Microsoft oficializó el lanzamiento de Windows Azure, su apuesta en el terreno de la computación en la nube (Cloud Computing), ampliando así la oferta existente en el mercado, por parte principalmente de Amazon y Google. En este artículo intentamos desgranar la propuesta de la compañía de Redmond, apuntando los conceptos más importantes desde un punto de vista práctico y con ojos de desarrollador.


Toni Recio y Gerard López

Project Manager y Analista Programador, ambos en Pasiona Consulting

Azure, ampliando posibilidades

Hasta ahora, cuando desarrollábamos aplicaciones a desplegar en servidores Windows teníamos dos opciones: usar infraestructura propia (tanto equipos como canales de comunicación) o contratarlos mediante un servicio de *hosting*. Azure nos abre una tercera vía, ampliándonos las posibilidades a la hora de distribuir nuestros desarrollos y situándonos en un escenario donde:

- Podemos dimensionar el hardware que aloja nuestros desarrollos sin intermediarios administrativos, siendo posible, por ejemplo, configurar la potencia de los servidores o el número de los mismos en cualquier momento; se nos permite, incluso, optar a programar la cantidad de recursos por fecha y hora según las previsiones de tráfico a las que queramos dar respuesta.
- Pagamos solo por lo que consumimos, con aplicación de tarifas competitivas.
- Disponemos de la fiabilidad y potencia de los *Data Centers* de Microsoft, a la vez que podemos decidir entre los diferentes centros para, por ejemplo, optimizar las comunicaciones dependiendo de la ubicación geográfica de los usuarios de nuestros servicios, o por cuestiones legales asociadas a dicha zona.
- La gestión del balanceo de carga entre servidores nos es totalmente transparente.
- Nos olvidamos del mantenimiento de los equipos, la redundancia de datos o la actualización del software de los servidores.
- Los costes de licencia se reducen considerablemente y se simplifican los modelos de licenciamiento.

En resumen, disponemos de la potencia necesaria, sea cual sea el escenario, tanto de proceso como a nivel de ancho de banda, a la vez que gestionamos todo ello con nuestras propias manos. Si eso no fuera suficiente, la plataforma implementa una serie de servicios (muchos de los cuales iremos descubriendo en éstas páginas) de los que nuestros desarrollos podrán aprovecharse. Azure, por tanto, no es solo una plataforma donde ejecutar nuestro código, sino que también nos provee a los desarrolladores de un elenco de herramientas en formato **SaaS** (*Software As A Service*) y **DaaS** (*Data As A Service*) a explotar desde nuestras soluciones.

Una vez ubicados en las posibilidades de la plataforma, veamos cómo podemos realizar nuestra primera toma de contacto con el nuevo Windows en la nube.

Preparando el entorno

Al contrario de lo que muchos piensan y aunque parezca paradójico, podemos empezar a jugar con Azure sin estar tan siquiera conectados a Internet.

Para ello tan solo necesitaremos a nuestro querido Visual Studio, en sus versiones 2008 con SP1 ó 2010, sobre el que instalaremos un conjunto de herramientas de Windows Azure que, aparte de la integración con el entorno de desarrollo, incluye también su SDK. La forma más sencilla de instalar dichas herramientas es mediante el **Web Platform Installer** (`http://www.microsoft.com/web/downloads/platform.aspx`); dentro del apartado "Herramientas del desarrollador", pulsamos en la acción "Personalizar" de la opción "Visual Studio Tools", para por último seleccionar "**Windows Azure Tools para Microsoft Visual Studio**" y pulsar el botón "Instalar". Es importante destacar que deberemos contar con IIS 7.0 y SQL Server instalados en nuestro equipo local.

Creando nuestro primer "Hola nube"

Vamos a tratar de hacer un ejemplo mínimo, para centrarnos en Azure como plataforma y que por otro lado nos permita ver "nubes" de inmediato. Es por ello que nuestro ejemplo será una simple página Web que muestre la última fotografía del satélite meteorológico **Meteosat**. En ella podremos introducir observaciones sobre la imagen y guardar los datos mediante un botón de comando, pero de momento nos conformaremos con mostrar la fotografía, dejando para más adelante la implementación de la manipulación y almacenamiento de datos.

Para empezar, arrancamos Visual Studio con privilegios de administrador, dado que el entorno de simulación local de la plataforma requerirá de ciertos permisos sobre nuestra máquina. Creamos un proyecto de la nueva categoría "**Cloud**", seleccionando como plantilla "**Windows Azure Cloud Service**", y asignamos los nombres `WeatherService` y `CloudMania` al proyecto y la solución, respectivamente.

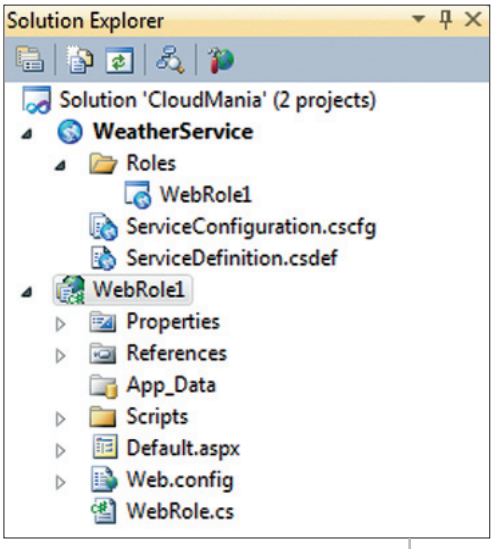

Figura 1. Estructura de la solución

Una vez que aceptamos el diálogo, se nos consultará qué roles debe implementar el servicio que queremos desarrollar. Podemos ver los roles como las máquinas virtuales que vamos a utilizar para desplegar nuestra aplicación en la nube. Cada rol nos presenta una configuración distinta de máquina orientada a unas tareas concretas. Los roles que se nos proponen son los siguientes:

- **ASP.NET Web Role**, destinado a aplicaciones ASP.NET tradicionales que también podremos utilizar, por ejemplo, para alojar aplicaciones Silverlight. Destacar que cuando hagamos uso de este rol para agregar una aplicación que ya teníamos desarrollada previamente, ésta no podrá ser del tipo "*Web Site*", ya que por ahora dichas plantillas no están soportadas, por lo que tendremos que migrar nuestro sitio Web a un proyecto de tipo "*Web Application*" para poder incluirlo en el *Cloud Service*.
- **ASP.NET MVC 2 Web Role**, orientado a desarrollos de aplicaciones con el nuevo *framework* MVC.
- **WCF Service Web Role**, para programar servicios basados en WCF.
- **Worker Role**, enfocado al desarrollo de servicios sin interfaz de usuario.
- **CGI Web Role,** para el despliegue de aplicaciones basadas en el protocolo FastCGI, como pueden ser gestores de contenido desarrollados en PHP.

> En la actualidad, Azure también da soporte a desarrollos realizados con **Ruby**, **Python** o **Java**, pero éstos requieren de la instalación de *software* adicional en nuestra máquina de desarrollo.

Dada la naturaleza de nuestro ejercicio, tan solo necesitamos un rol Web ASP.NET. Al indicarlo, se nos generará la estructura de nuestra solución (figura 1), donde nos encontraremos con un nuevo proyecto: al proyecto `WeatherService`, que contiene la configuración y metadatos del nuevo servicio, se añadirá otro llamado `WebRole1`, que contiene una implementación prácticamente igual a lo que sería un proyecto ASP.NET tradicional.

Centrémonos de momento en `Default.aspx` para mostrar la última imagen del Meteosat (listado 1); bastará con insertar una etiqueta (*tag*) de imagen en la página y ubicar un par de controles con el espacio para los comentarios y el botón de guardar.

Ahora ya estamos en disposición de arrancar nuestra solución; no almacena datos, pero ya mostrará la última fotografía del satélite. Al ejecutar, advertiremos que nos aparece un nuevo icono de notificación, ubicado en la esqui-

na inferior derecha de la pantalla, que muestra el estado de los servicios asociados a nuestro entorno de desarrollo. En estos momentos se están levantando los entornos de desarrollo locales de los dos servicios clave de la plataforma Azure: **Development Fabric** y **Development Storage**.

- **Colas**: Funcionalidad orientada a comunicar diferentes procesos mediante mensajes de menos de 8 Kb. No se trata tanto de almacenar información, sino de ponerla a disposición de varios servicios, tanto en la nube como fuera de ella.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="WebRole1._Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <div style="float:left; margin-right: 20px;">
      <img ID="Imagen" runat="server" alt="Foto Meteosat"
       src="http://www.meteosat.com/imagenes/meteosat/sp/ultima_meteosat-visible.jpg"/>
   </div>
   <div>
      <asp:TextBox ID="txtObservaciones" runat="server" TextMode="MultiLine"
         Height="100px" Width="300px"/>
      <br/>
      <asp:Button ID="btnGuardar" runat="server" Text="Almacenar"
         onclick="btnGuardar_Click" />
   </div>
   </form>
</body>
```

Listado 1. Mostrar última imagen del Meteosat (Default.aspx)

Development Fabric es el servicio que nos simula el *Fabric Controller* de Azure, un servicio vital del sistema, dado que es el encargado de monitorizar, mantener y provisionar el entorno donde se ejecuten nuestras aplicaciones en la nube.

Por otro lado, con Development Storage dispondremos en local de todas las herramientas disponibles en Azure para que nuestros desarrollos puedan almacenar datos no relacionales accesibles vía REST. El servicio contempla tres modelos de almacenamiento:

- **Tablas**. Se trata de un modelo prácticamente idéntico al de una tabla de una base de datos convencional, pero sin la capacidad de relacionarse con otras tablas: información estructurada, pero no relacionada.
- **Blobs**: Estructura orientada a almacenar archivos binarios de gran tamaño (hasta un total de 200 Gb. En nuestro entorno local, el tamaño máximo se reduce a 2 Gb).

Para implementar los tres modelos, y aunque parezca paradójico, la plataforma local se sustenta sobre SQL Server. De hecho, lo primero que veremos antes de que se nos muestre nuestra aplicación es la ventana de inicialización del *Storage*, la cual, una vez aceptada, dará paso al navegador con nuestra aplicación.

Ahora que ya tenemos desarrollada la primera parte de nuestra "nube local", tendremos la posibilidad de interactuar con los servicios de *Fabric* y *Storage* mediante los administradores accesibles en el menú contextual del icono de notificación. También podremos configurar nuestro servicio editando en el proyecto **WeatherService** los archivos **ServiceConfiguration.cscfg** y **ServiceDefinition.csdef** o, sin necesidad de tocar XML, configurar directamente vía **WebRole1** (ver figura 2). Es de destacar con qué facilidad podremos definir el número de máquinas virtuales que soportarán nuestra aplicación, la potencia de las mismas (tabla 1) o la sencilla configuración de los *endpoints*.

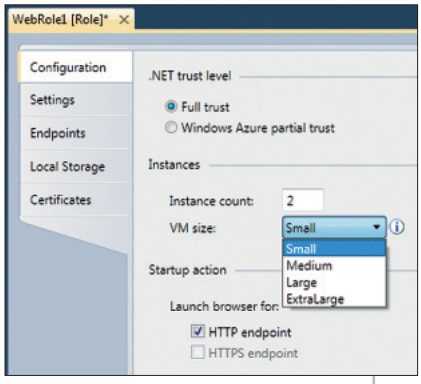

Figura 2. Configurando el rol Web

Tamaño de instancia	CPU	Memoria	Almacenamiento	Rendimiento E/S
Small	1.6 GHz	1.75 GB	225 GB	Moderado
Medium	2 x 1.6 GHz	3.5 GB	490 GB	Elevado
Large	4 x 1.6 GHz	7 GB	1,000 GB	Elevado
Extra large	8 x 1.6 GHz	14 GB	2,040 GB	Elevado

Tabla 1. Características de las máquinas virtuales.

Para probar, ¿primero hay que pasar la tarjeta?

¿Y cómo podemos hacer pruebas reales en la nube? Microsoft ofrece dos modalidades gratuitas en promoción, una para suscriptores MSDN y otra para el público general. Las dos modalidades constan de una serie de recursos limitados, y en caso de superar los consumos establecidos deberemos abonar los costes adicionales.

Todas las labores administrativas relacionadas con Azure se gestionan desde

...just Microsoft®

Gestió de Qualitat
ISO 9001:2000
▸ Somàs a controls periòdics voluntaris
DEKRA Certification
DEKRA
Certificado

www.pasiona.com

Microsoft®
GOLD CERTIFIED
Partner

el portal Microsoft Online Services (**https://mocp.microsoftonline.com**), al que accederemos para pedir nuestra cuenta.

> Si deseamos usar la oferta para suscriptores MSDN, deberemos acceder primero desde el enlace que aparece en el menú de suscripción del propio portal de MSDN (**http://msdn.microsoft.com/es-es/default.aspx**). En la zona "Suscripciones" accederemos a la opción "Plataforma Windows Azure" para poder disfrutar de la oferta con mayores recursos disponibles y durante un período de tiempo más extenso.

Si es la primera vez que accedemos al portal de servicios en línea de Microsoft, se nos requerirá una serie de datos de contacto; una vez rellenados los formularios, dependiendo del tipo de promoción iremos por un camino u otro. En el caso de la promoción pública, nos dirigimos a la pestaña "Servicios", seleccionamos "Plataforma Windows Azure" y en "Ofertas especiales" elegimos "Comprar ahora" (figura 3); mientras que los suscriptores MSDN se encontrarán ya en el carro de la compra y tendrán que pulsar "Pagar" para continuar. En ambos casos, entre los datos que nos piden está el número de tarjeta de crédito que es obligatorio facilitar, dado que si excedemos los recursos asignados se nos podrá cobrar.

Cuando hayamos facilitado todos los datos que se nos requieren, Microsoft nos enviará un par de correos avisándonos de la activación del servicio y dándonos instrucciones para el acceso al portal de desarrollo de Azure.

Dos detalles importantes a tener en cuenta: primero, si no queremos descuidarnos y exceder los límites gratuitos de las promociones, podemos consultar la cantidad de recursos consumidos hasta un momento determinado (la información no es instantánea, tarda unas horas en actualizarse) desde el apartado "Acciones", pulsando el enlace "Ver mis facturas". Y segundo, el hecho de que por defecto nuestra suscripción a la plataforma Azure está activada para renovarse automáticamente cuando finalice el plazo de prueba. Si no queremos pasar a un servicio de pago sin nuestra intervención, podemos cancelar la renovación automática en la pestaña de suscripciones mediante la acción "Cancelar suscripción de renovación automática".

> El sistema implementa una serie de avisos, vía correo electrónico, que nos advierten cuando estamos a punto de sobrepasar los límites establecidos en las cuentas de evaluación.

Figura 3. Comprando en Microsoft Online Services

Subiendo nuestra aplicación a una nube real

Una vez que hemos realizado nuestras pruebas en entorno local (podríamos decir que nuestra aplicación está en la niebla ;-) y hemos obtenido una cuenta con acceso a Azure, estamos en condiciones de publicar nuestro "Hola nube" en los *Data Centers* de Microsoft.

Para publicar nuestro proyecto **WeatherService**, volveremos a Visual Studio y utilizaremos la opción "Publish" del menú "Build", lo que nos abrirá una carpeta local donde están ubicados nuestros dos únicos ficheros necesarios para hacer la publicación, y también nos lanzará nuestro navegador predeterminado mostrando el portal de desarrollo de Azure (**https://windows.azure.com**). Si es la primera vez que accedemos al portal, tendremos que crear un servicio mediante el enlace "New Service" y seleccionar entre crear una cuenta *Storage* o un *Hosted Service*. Para el ejemplo que nos ocupa, elegiremos el servicio, introduciremos un nombre y descripción para el mismo, y después tendremos que elegir una URL para nuestra aplicación, que será un subdominio de **cloudapp.net**. Observe que tenemos un botón para comprobar que la dirección está disponible. Hemos de tener en cuenta que ni el nombre ni la URL podrán ser modificados posteriormente.

Otro aspecto importante a decidir será en qué zona geográfica queremos que sea alojada nuestra aplicación. Aquí también podemos seleccionar un grupo de afinidad, porque si vamos a subir varias aplicaciones a la nube que interactúan entre ellas, esto nos asegura que se mantengan en una misma región geográfica. Si cuando creamos un servicio no especificamos un grupo de afinidad, después no podremos agregárselo, con lo que tendríamos que eliminar el servicio y volver a crearlo.

Una vez que tenemos el servicio creado, nos aparece un panel de gestión (figura 4). En dicho panel podemos controlar el despliegue en producción y en el entorno de *staging* (entorno de pruebas). Si pulsamos en la flecha de la derecha veremos los dos entornos; esto nos sirve para poder mantener una versión publicada en la nube pero en un entorno de pruebas al que podremos acceder con una URL diferente (podemos restringir el acceso), y cuando hayamos probado nuestra versión, pasarla a producción con

un simple clic (figura 5). Este paso entre entornos nos desplegará la versión de *staging* en producción, y a su vez hará una copia de la versión de producción en *staging* para mantenerla de *backup*.

Figura 4. Selección del entorno de despliegue

Figura 6. Servicio desplegado en producción

Ahora ya solo nos faltaría arrancar la aplicación utilizando el botón "Run"; paciencia de nuevo, esto puede tardar unos minutos. Cuando el proceso haya finalizado, veremos que el estado de nuestra aplicación es **Ready**; a partir de este momento ¡estamos en la nube!, y podemos visitar nuestra aplicación desde la URL asociada.

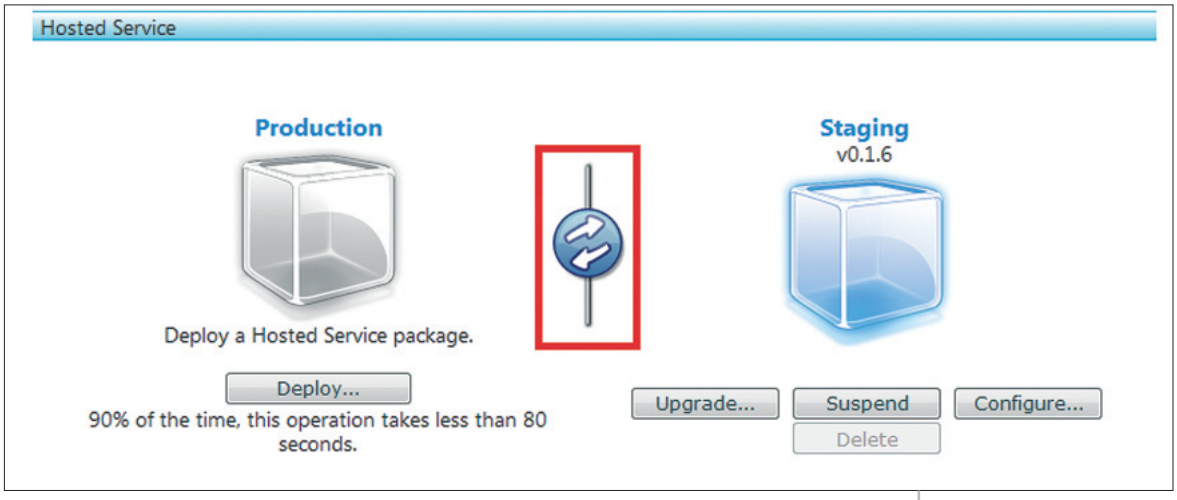

Figura 5. Paso entre entornos

Para realizar la publicación, deberemos pulsar el botón "Deploy" (en nuestro ejemplo, lo haremos sobre el entorno de producción) e indicar la ubicación de los ficheros que nos habia generado Visual Studio al publicar: el **Application Package** (fichero `.cspkg`) y el **Configuration Settings** (fichero `.cscfg`). Entonces pulsaremos de nuevo el botón "Deploy" y esperaremos unos minutos hasta que la aplicación se haya desplegado por completo.

Una vez desplegada, el cubo que representa nuestra aplicación se nos mostrará en azul, y veremos que se nos habilitan algunas nuevas opciones (figura 6). Desde la opción "Configure", podremos modificar el fichero `.cscfg`, donde por ejemplo podríamos establecer nuevamente el número de instancias levantadas para balancear la carga de nuestra aplicación.

Debemos tener claro que siempre que tengamos una aplicación desplegada, ésta estará consumiendo recursos de proceso, incluso aunque estuviera en estado **Stopped**. Si queremos dejar de consumir recursos para no sobrepasar las horas establecidas en las cuentas gratuitas, debemos eliminar el despliegue (¡el cubo debe quedar gris!).

Todo el proceso de publicación puede realizarse de una manera automatizada mediante **Windows Azure Service Management API** (`http://msdn.microsoft.com/en-us/library/ee460799.aspx`). Los detalles de esta API basada en REST se escapan del alcance de este artículo; puede ver un escenario completo y funcional de su utilización en uno de los ejemplos publicados por Microsoft, **CSManage** (`http://code.msdn.microsoft.com/Rele-`

ase/ProjectReleases.aspx?ProjectName=windowsazuresamples&ReleaseId=3233).

Trabajando con el Storage

Ahora que ya sabemos lo necesario para jugar un poco con Azure, vamos a interactuar con el servicio *Storage*, a modo de simple pincelada introductoria, para que nuestro pequeño ejemplo trabaje con datos.

```
using System;
using Microsoft.WindowsAzure.StorageClient;

namespace WebRole1
{
  public class imgMetaData : TableServiceEntity
  {
     public imgMetaData()
     {
        PartitionKey = "meteosat";
        RowKey =  Guid.NewGuid().ToString();
     }

     public string Observacion { get; set; }
     public DateTime Fecha { get; set; }
     public string ContainerRef { get; set; }
     public string BlobRef { get; set; }
  }
}
```

Listado 2. Entidad de metadatos (imgMetaData.cs)

> Al trabajar con .NET, podemos utilizar la librería **Microsoft.WindowsAzure.StorageClient**, que nos encapsulará las llamadas vía REST para que nos resulte más cómodo trabajar con *Storage*.

Para almacenar los datos en una tabla, lo primero que vamos a necesitar es crear una estructura que represente los metadatos de la imagen que deseamos almacenar. Lo haremos mediante una nueva clase que implemente una entidad heredando de **TableServiceEntity**, clase que ubicaremos en un nuevo fichero **imgMetaData.cs** dentro del proyecto de rol Web (listado 2). En el constructor de nuestra nueva entidad se inicializan las propiedades heredadas **PartitionKey** y **RowKey**, que implementan claves de agrupación e identificación de datos en la nube.

Ahora que ya tenemos nuestra entidad, deberemos generar el código mínimo necesario para interactuar con colecciones de dichas entidades. Para ello, también en el rol Web, creamos otra nueva clase en el fichero **imgMetaDataServiceContext.cs**, esta vez heredando de **TableServiceContext** (listado 3). En nuestro ejemplo nos centraremos en dos métodos: uno para recuperar entidades de metadatos y otro para agregarlas.

```
using System;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.StorageClient;

namespace WebRole1
{
   public class imgMetaDataServiceContext : TableServiceContext
   {
      public imgMetaDataServiceContext(string baseAdress,
         StorageCredentials credentials) : base(baseAdress, credentials)
      {
      }

      public IQueryable<imgMetaData> imgMetaData
      {
         get { return this.CreateQuery<imgMetaData>("imgMetaData"); }
      }

      public void AddImage(string observacion, DateTime fecha,
         string containerRef, string blobRef)
      {
         AddObject("imgMetaData", new imgMetaData {
            Observacion = observacion, Fecha = fecha,
            ContainerRef = containerRef, BlobRef = blobRef });
         SaveChanges();
      }
   }
}
```

Listado 3. Servicio de contexto (imgMetaDataServiceContext.cs)

> Es necesario agregar una referencia a la librería **System.Data.Services.Client** para poder compilar el código.

Vamos ahora a crear la cadena de conexión, que de momento apuntará a nuestro entorno local: dentro de la carpeta **Roles** del proyecto **WeatherService**, accedemos a las propiedades del **WebRole1** (tal y como veíamos en la figura 2), y en el apartado "Settings" añadimos el elemento **DataConnectionString** con valor **UseDevelopmentStorage=true**.

A continuación, nos centraremos en el evento **OnStart** del rol Web. En él mapeamos la configuración del servicio y nos conectamos al *Storage*, para así poder crear el contenedor BLOB de imágenes y la tabla que contendrá los metadatos. Todo ello puede verse en el listado 4.

Por último, nos queda implementar el evento **Click** del botón **btnGuardar**, cuyo cuerpo se muestra en el listado 5.

Con estos sencillos pasos, nuestra aplicación ya almacena datos en el *Storage*; pero deberemos tener en cuenta que los datos se almacenan en local. Para inte-

```
// Creamos la conexión con el servicio de Storage
CloudStorageAccount.SetConfigurationSettingPublisher(
   (configName, configSetter) => {
      configSetter(ConfigurationManager.ConnectionStrings[configName].ConnectionString);
   });
var storageAccount =
   CloudStorageAccount.FromConfigurationSetting("DataConnectionString");

// Creamos el contenedor BLOB para almacenar las imágenes
CloudBlobClient blobStorage = storageAccount.CreateCloudBlobClient();
CloudBlobContainer container = blobStorage.GetContainerReference("meteosat");
container.CreateIfNotExist();

// Configuramos el contenedor con acceso público
var permissions = container.GetPermissions();
permissions.PublicAccess = BlobContainerPublicAccessType.Container;
container.SetPermissions(permissions);

// Creamos la tabla para almacenar los metadatos
CloudTableClient.CreateTablesFromModel(typeof(imgMetaDataServiceContext),
       storageAccount.TableEndpoint.AbsoluteUri, storageAccount.Credentials);
```

Listado 4. Evento OnStart en WebRole.cs

```
// Inicializamos la cuenta de Storage
var storageAccount =
   CloudStorageAccount.FromConfigurationSetting("DataConnectionString");

// El nombre del contenedor no puede tener mayúsculas
string containerName = "meteosat";

// Almacenamos la imagen
CloudBlobClient blobStorage = storageAccount.CreateCloudBlobClient();
CloudBlobContainer container = blobStorage.GetContainerReference(containerName);
string uniqueBlobName = string.Format("Imagenes/{0}", Guid.NewGuid().ToString());
CloudBlockBlob blob = container.GetBlockBlobReference(uniqueBlobName);

WebRequest req = WebRequest.Create(Imagen.Src);
WebResponse response = req.GetResponse();
Stream stream = response.GetResponseStream();

blob.UploadFromStream(stream);

// Almacenamos los metadatos en la tabla
var operations =
   new imgMetaDataServiceContext(storageAccount.TableEndpoint.ToString(),
      storageAccount.Credentials);
operations.AddImage(txtObservaciones.Text, DateTime.Now,
   containerName,uniqueBlobName);
```

Listado 5. BtnGuardar_Click en Default.aspx.cs

ractuar con la nube, tenemos que cambiar nuestra cadena de conexión con los datos, que obtendremos como resultado de crear un servicio de *Storage* en el portal de desarrollo de Azure (del mismo modo que antes hemos creado un *Hosted Service*).

SQL Azure, almacenamiento relacional por fin en la nube

Uno de los factores diferenciales de la plataforma Azure con respecto a otras alternativas de computación en la nube es la posibilidad de trabajar con almacenamiento relacional. Microsoft, recogiendo el *feedback* de la comunidad, decidió apostar por un SQL Server en la nube que nos permita trasladar nuestros desarrollos ya realizados a Azure con un simple cambio en la cadena de conexión. Pese a que **SQL Azure** cuenta con algunas limitaciones (ver tabla 2) con respecto a su versión tradicional, puede satisfacer plenamente las necesidades de la mayoría de los proyectos.

La curva de aprendizaje de SQL Azure es mínima, y podemos utilizar **SQL Server Management Studio** (*SSMS*) para su administración, mientras que nuestros desarrollos accederán a los datos mediante el mismo código ADO.NET. Para obtener los datos del servidor y poder realizar las conexiones pertinentes, debemos recurrir al portal de gestión de SQL Azure (**https://sql.azure.com**) y pulsar en el apartado "DataBase". Pese a todo, no nos resultará inmediato conectarnos mediante SSMS, que nos mostrará un mensaje de error al tratar de conectar con el servidor. El problema no es otro que el hecho de que SQL Azure tiene cerradas por defecto todas las puertas de acceso. Para solventar esta situación, volvemos al apartado "Database" del portal y en él seleccionamos la pestaña "Firewall Settings"; añadimos una nueva regla en el cortafuegos indicando nuestra dirección IP y esperamos unos minutos a que la regla se replique.

Una vez solventado el escollo de la seguridad, nos encontraremos con otro error derivado del hecho de que SSMS espera conectarse a un servidor de SQL

Soportado	No soportado
Tablas, índices y vistas	Consultas y transacciones distribuidas
Procedimientos almacenados	Tipos de datos espaciales
Triggers	Service Broker
Constraints	CLR
Variables de tabla y tablas temporales de sesión	Servidor físico o catálogo DDL y vistas

Tabla 2. Principales características de SQL Azure.

> Microsoft ha puesto gran empeño en no hacer traumático el camino de nuestras aplicaciones hacia la nube, que en muchos casos puede requerir solo un simple cambio en la cadena de conexión

Server tradicional, no compatible al 100% con SQL Azure, lo que puede provocar algunas incompatibilidades con herramientas de administración como el propio SSMS. Para sortear este último escollo, cancelamos la ventana de conexión y pulsamos el botón "Nueva consulta". De nuevo se visualizará la ventana de conexión, en la que pulsamos el botón "Opciones" e indicamos, en la pestaña "Propiedades de Conexión", el nombre de la base de datos con la que deseemos trabajar. En nuestro caso, al no haber creado ninguna, indicaremos la base de datos `master`. Pese a no disponer de toda la potencia de gestión de SSMS, podremos ahora ejecutar sentencias de Transact-SQL contra nuestra base de datos en la nube. Es importante destacar que estos primeros problemas de compatibilidad ya se encuentran solventados en el recientemente publicado SQL Server 2008 R2.

> Será de vital importancia optimizar las consultas y los accesos a la base de datos, ya que es importante recordar que en el entorno de la nube pagamos por su uso; ¡los **SELECT *** siempre son más caros!

Por último, es importante destacar que existen dos versiones de SQL Azure: la **Web Edition**, de 1 Gb de capacidad, y la **Business Edition**, de 10 Gb. Si necesitamos superar los 10 Gb de información relacional, deberemos hacer uso de más de un servidor de bases de datos. Pese a todo, en el pasado MIX de Las Vegas se anunció, con disponibilidad inmediata, una nueva edición de 50 Gb.

Un amplio horizonte de posibilidades

En la actualidad, Azure ya ofrece algunos otros interesantes servicios en producción, además de los que hemos mencionado hasta ahora, así como otros servicios de momento disponibles en evaluación:

- **AppFabric** (no confundir con el *Fabric Controller*). Se trata de un conjunto de potentes servicios orientados a las comunicaciones, como el **Service Bus**, que mediante una implementación del patrón *Enterprise Service Bus* nos permite registrar y exponer servicios entre diferentes redes a través de dispositivos de control de accesos como *firewalls* o NAT, o el **Access Control**, con el que podremos exponer dichos servicios de forma segura.
- **Windows Azure Drive**. Es un servicio que nos da acceso a una API NTFS, mediante la cual podemos hacer uso de un espacio de disco permanente, implementado con **Windows Azure Page Blob**. No está pensado para que lo utilicemos como almacenamiento de datos, pues para eso ya tenemos las herramientas de Storage y SQL Azure, sino que está orientado a facilitarnos la migración de desarrollos existentes que hacen uso de recursos locales (por ejemplo, archivos locales de configuración). Cuenta con un límite de 1 TB de información por *page blob*. Actualmente se encuentra en fase beta.
- **Codename Dallas**. Es un servicio de datos (*DaaS*) que nace con la idea de englobar fuentes de datos para que puedan ser consumidas por programadores e *information workers*. Nosotros también dispondremos de la posibilidad de proveer nuestra información creando servicios propios, que podremos publicar para que sean consumidos por terceros. Este producto se encuentra en CTP, pero para que nos hagamos una idea y clarificar el concepto, entre los servicios actuales se encuentran *NASA Mars Orbital Images* y *UNESCO UIS Data*.

Un buen modo de estar al día con relación a la aparación de nuevos servicios y utilidades alrededor de Azure es el portal **PinPoint** de Microsoft (`http://pinpoint.microsoft.com`) o el centro de desarrollo para Azure en MSDN (`http://msdn.microsoft.com/es-ES/windowsazure`).

Conclusiones

La conclusión más importante que podemos extraer acerca del tema de este artículo es el empeño que ha puesto Microsoft en no hacer traumático el camino de nuestras aplicaciones hacia la nube, que en muchos casos puede producirse con un simple cambio en la cadena de conexión. Por añadidura, nos ofrece una plataforma de despliegue profesional asequible tanto para nuestro día a día como para los servicios destinados a ser los nuevos casos de éxito de la red; todo ello de una forma sencilla, y aislándonos de la problemática de gestión de servidores y comunicaciones.

¿Y tú? ¿Cuánto vas a tardar en subirte a la nube?