# **Introducción**

Completado

100 XP

* 3 minutos

Imagine que trabaja para una empresa de escaleras mecánicas que ha realizado grandes inversiones en tecnología de IoT para supervisar su producto sobre el terreno. Se encarga de supervisar el procesamiento de los datos del sensor de temperatura de los engranajes de accionamiento de las escaleras mecánicas. Debe supervisar los datos de temperatura y agregar una marca de datos para indicar si los engranajes están demasiado calientes. En sistemas posteriores, estos datos ayudarán a determinar cuándo se requiere mantenimiento.

Su empresa recibe datos de sensores de varias ubicaciones y de diferentes modelos de escaleras mecánicas. Los datos llegan en diferentes formatos, como cargas de archivos por lotes, extracciones programadas de bases de datos, mensajes en una cola y datos entrantes de un centro de eventos. Desea desarrollar un servicio reutilizable que pueda procesar los datos de temperatura de todos estos orígenes.

Al diseñar un servicio como este con estrategias de arquitectura empresarial tradicionales, tendría que tener en cuenta la infraestructura de servidor y el mantenimiento anticipado: investigar el hardware necesario, planear su instalación, coordinarse con TI para administrarlo, etc. Una alternativa a todo ese trabajo es la informática sin servidor. Con la informática sin servidor, el proveedor en la nube administra el aprovisionamiento y el mantenimiento de la infraestructura, lo que le permite centrarse totalmente en la creación de la aplicación lógica. Azure Functions es un componente clave de la oferta de informática sin servidor de Azure y le permite ejecutar fragmentos de código o *funciones*, escritos en el lenguaje de programación que prefiera, en la nube.

## **Objetivos de aprendizaje**

En este módulo, aprenderá a:

* Decidir si la informática sin servidor es adecuada para sus necesidades empresariales.
* Crear una aplicación de función de Azure en Azure Portal.
* Ejecutar una función mediante desencadenadores.
* Supervisar y probar la función de Azure desde Azure Portal.

# **Decidir si la informática sin servidor es adecuada para sus necesidades empresariales**

Completado

100 XP

* 5 minutos

Para ayudarle a decidir si la informática sin servidor satisface sus necesidades, primero vamos a ver en qué consiste.

## **¿Qué es la informática sin servidor?**

La [informática sin servidor](https://azure.microsoft.com/solutions/serverless/) se puede considerar una función como servicio (FaaS), o bien un microservicio que se hospeda en una plataforma de nube. La lógica de negocios se ejecuta como funciones y no necesita aprovisionar ni escalar manualmente la infraestructura. El proveedor de nube administra la infraestructura. La aplicación se escala horizontalmente o se reduce verticalmente de manera automática en función de la carga. Azure tiene varias formas de crear este tipo de arquitectura. Los dos métodos más comunes son Azure Logic Apps y Azure Functions, que son los métodos en los que nos centramos en este módulo.

## **¿Qué es Azure Functions?**

Azure Functions es una plataforma de aplicaciones sin servidor. Permite que los desarrolladores hospeden la lógica de negocios que se puede ejecutar sin aprovisionar infraestructura. Functions proporciona escalabilidad intrínseca y solo se cobra por los recursos usados. Puede escribir el código de la función en el lenguaje que prefiera, como C#, F#, JavaScript, Python y PowerShell Core. También se admiten los administradores de paquetes como NuGet y NPM, por lo que es posible usar bibliotecas populares en la lógica de negocios.

## **Ventajas de una solución de informática sin servidor**

La informática sin servidor es una opción excelente para hospedar código de lógica de negocios en la nube. Con las ofertas sin servidor, como Azure Functions, puede escribir la lógica de negocios en el lenguaje que prefiera. El escalado se realizará automáticamente, no tendrá servidores que administrar y solo se le cobrará por lo que use, no por el tiempo reservado. Estas son algunas características adicionales de una solución sin servidor que se deben tener en cuenta.

### **Prevención de la sobreasignación de infraestructura**

Suponga que aprovisionó servidores de máquina virtual y los configuró con recursos suficientes para controlar los tiempos de carga máxima. Cuando la carga es ligera, es posible que pudiera pagar por infraestructura que no usa. La informática sin servidor ayuda a resolver el problema de la asignación mediante el escalado o la reducción vertical automáticamente y solo se facturará cuando la función procese trabajo.

### **Lógica sin estado**

Las funciones sin estado son excelentes candidatos para el proceso sin servidor; las instancias de funciones se crean y destruyen a petición. Si es necesario el estado, se puede almacenar en un servicio de almacenamiento asociado.

### **Controladas por eventos**

Las funciones se *basan en eventos*. Esto significa que solo se ejecutan en respuesta a un evento (denominado "desencadenador"), como recibir una solicitud HTTP o agregar un mensaje a una cola. Configure un desencadenador como parte de la definición de la función. Este método simplifica el código al permitirle declarar de dónde provienen los datos (enlace de entrada/desencadenador) y dónde van (enlace de salida). No es necesario escribir código para ver colas, blobs, centros, etc. Puede centrarse únicamente en la lógica de negocios.

### **Las funciones pueden utilizarse en entornos de proceso tradicionales**

Las funciones son un componente clave de la informática sin servidor, pero también son una plataforma de proceso general para ejecutar cualquier tipo de código. En caso de que cambien las necesidades de la aplicación, puede usar el proyecto e implementarlo en un entorno que no sea sin servidor, una opción que le aporta la flexibilidad para administrar el escalado, realizar ejecuciones en redes virtuales e incluso aislar las funciones por completo.

## **Desventajas de una solución de informática sin servidor**

El proceso sin servidor no siempre será la solución adecuada para hospedar la lógica de negocios. Estas son algunas características de las funciones que pueden afectar a su decisión de hospedar los servicios en un proceso sin servidor.

### **Tiempo de ejecución**

De manera predeterminada, las funciones tienen un tiempo de espera de 5 minutos. Se puede configurar a un máximo de 10 minutos. Si la función necesita más de 10 minutos para ejecutarse, puede hospedarla en una máquina virtual. Además, si el servicio se inicia con una solicitud HTTP y espera ese valor como una respuesta HTTP, el tiempo de espera se restringe aún más a 2,5 minutos. Por último, también hay una opción denominada [Durable Functions](https://docs.microsoft.com/es-es/azure/azure-functions/durable) que le permite orquestar las ejecuciones de varias funciones sin tiempo de espera.

### **Frecuencia de ejecución**

La segunda característica es la frecuencia de ejecución. Si espera que varios clientes ejecuten la función de manera continua, sería recomendable calcular el uso y el costo de usar las funciones en consecuencia. Podría ser más barato hospedar el servicio en una máquina virtual.

Durante el escalado, solo se puede crear una instancia de aplicación de función cada 10 segundos, para hasta 200 instancias en total. Tenga en cuenta que cada instancia puede atender varias ejecuciones simultáneas, por lo que no hay ningún límite establecido en la cantidad de tráfico que una sola instancia puede controlar. Diferentes tipos de desencadenadores tienen diferentes requisitos de escalado, por lo que se recomienda investigar el desencadenador que elija y sus límites.

# **Ejercicio: Creación de una aplicación de funciones en Azure Portal**

600 XP

* 5 minutos

Espacio aislado activado. Tiempo restante: 3 h 57 min

Usó 3 de 10 espacios aislados por hoy. Mañana habrá disponibles más espacios aislados.

Elija el lenguaje de desarrollo

JavaScript

PowerShell

Ya está listo para empezar a implementar el servicio de temperatura. En la unidad anterior, determinó que una solución sin servidor sería la mejor opción para sus necesidades. Comencemos por crear una aplicación de función para que contenga la función de Azure.

## **¿Qué es una aplicación de función?**

Las funciones se hospedan en un contexto de ejecución que se conoce como una aplicación de función. Puede definir aplicaciones de función para agrupar y estructurar de manera lógica las funciones y un recurso informático en Azure. En nuestro ejemplo de las escaleras mecánicas, crearía una aplicación de función para hospedar el servicio de temperatura del engranaje impulsor de las escaleras. Es necesario tomar algunas decisiones con el fin de crear la aplicación de función: deberá elegir un plan de servicio y seleccionar una cuenta de almacenamiento compatible.

### **Selección de un plan de servicio**

Las aplicaciones de función pueden usar uno de dos tipos de planes de servicio. El primer plan de servicio es el plan de servicio de consumo. Este es el plan que elige cuando se usa la plataforma de aplicaciones sin servidor de Azure. El plan de servicio de consumo proporciona escalado automático y se factura cuando se ejecutan las funciones. El plan de consumo viene con un período de tiempo de espera configurable para la ejecución de una función. De manera predeterminada es de 5 minutos, pero se puede configurar para tener un tiempo de espera de hasta 10 minutos.

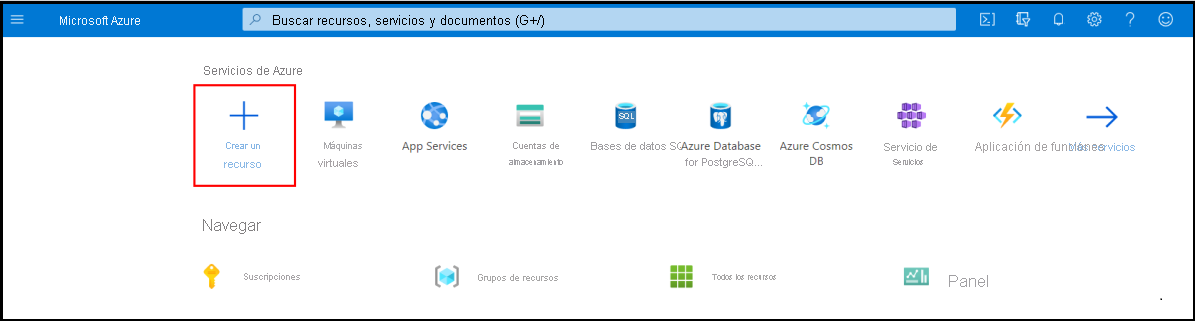
El segundo plan se denomina plan de Azure App Service. Este plan permite evitar los períodos de tiempo de espera al tener la función en ejecución continuamente en una máquina virtual definida. Cuando se usa un plan de App Service, usted es responsable de administrar los recursos de la aplicación en los que se ejecuta la función, por lo que técnicamente no se trata de un plan sin servidor. Sin embargo, puede resultar una opción más conveniente si las funciones se usan continuamente o si requieren más potencia de procesamiento o tiempo de ejecución que lo que puede proporcionar el plan de consumo.

### **Requisitos de la cuenta de almacenamiento**

Cuando se crea una aplicación de función, se debe vincular a una cuenta de almacenamiento. Puede seleccionar una cuenta existente o crear una. La aplicación de función usa esta cuenta de almacenamiento para las operaciones internas, como el registro de las ejecuciones de función y la administración de los desencadenadores de ejecución. En el plan de servicio de consumo también es donde se almacenan el código de la función y el archivo de configuración.

## **Creación de una aplicación de función**

Vamos a crear una aplicación de función en Azure Portal.

1. Inicie sesión en [Azure Portal](https://portal.azure.com/learn.docs.microsoft.com)  con la misma cuenta que activó para el espacio aislado.
2. Seleccione Crear un recurso.  
     
   Aparecerá el panel Crear un recurso.
3. En el panel de menú izquierdo, seleccione Proceso y, luego, en la lista Destacado, elija Aplicación de funciones. Aparece el panel Crear aplicación de funciones.
4. En la pestaña Aspectos básicos, escriba los valores siguientes para cada opción.

| **Configuración** | **Valor** |
| --- | --- |
| Detalles del proyecto |  |
| Suscripción | Suscripción de Concierge |
| Grupo de recursos | learn-944f7e49-84d0-4742-8976-e255996c502c |
| Detalles de instancia |  |
| Nombre de la aplicación de funciones | Escriba un nombre de aplicación globalmente único. El nombre de la aplicación de funciones servirá como la dirección URL base del servicio. Por ejemplo, puede asignarle el nombre escalator-functions-xxxxxxx, donde las siete (7) x pueden reemplazarse por sus iniciales y el año de nacimiento. Si no es único globalmente, pruebe cualquier otra combinación. Los caracteres válidos son a-z, 0-9 y -. |
| Publicar | Código |
| Pila en tiempo de ejecución | Node.js (lenguaje en el que implementaremos los ejemplos de la función de este ejercicio) |
| Versión | *default* |
| Region | Seleccione una ubicación geográfica próxima a usted. En un sistema de producción, se recomienda seleccionar una ubicación cercana a los clientes o consumidores de la función. |

1. Seleccione Revisar y crear y, luego, Crear. La implementación tardará unos minutos. Cuando finalice, recibirá una notificación.

## **Comprobación de la aplicación de función de Azure**

1. Una vez finalizada la implementación, seleccione Ir al recurso. Aparecerá el panel Aplicación de funciones.
2. En la sección Información esencial, seleccione el vínculo Dirección URL para abrirlo en un explorador. Aparecerá una página web predeterminada en la que se indique que la aplicación de funciones está en funcionamiento.

Vuelva a comprobar su trabajo

Use el portal para crear la aplicación de función y la función mediante las instrucciones anteriores.

# **Ejecución de código a petición con Azure Functions**

Completado

100 XP

* 10 minutos

Ahora que ya se ha creado una aplicación de funciones, veremos cómo compilar, configurar y ejecutar una función.

### **Desencadenadores**

Las funciones se basan en eventos, lo que significa que se ejecutan en respuesta a un evento.

El tipo de evento que inicia la función se denomina desencadenador. Debe configurar una función con un desencadenador exactamente.

Azure admite desencadenadores para los siguientes servicios.

| **Servicio** | **Descripción del desencadenador** |
| --- | --- |
| Blob Storage | Inicia una función cuando se detecta un blob nuevo o actualizado. |
| Azure Cosmos DB | Inicia una función cuando se detectan inserciones y actualizaciones. |
| Event Grid | Iniciar una función cuando se recibe un evento de Event Grid. |
| HTTP | Inicia una función con una solicitud HTTP. |
| Eventos de Microsoft Graph | Inicia una función como respuesta a un webhook entrante desde Microsoft Graph. Cada instancia de este desencadenador puede reaccionar a un tipo de recurso de Microsoft Graph. |
| Queue Storage | Inicia una función cuando se recibe un nuevo elemento en una cola. El mensaje de la cola se proporciona a modo de entrada para la función. |
| Service Bus | Inicia una función en respuesta a los mensajes de una cola de Service Bus. |
| Temporizador | Inicia una función según una programación. |

### **Enlaces**

Los enlaces son una manera declarativa de conectar datos y servicios a la función. Los enlaces saben cómo comunicarse con los diferentes servicios, lo que significa que no tiene que escribir código en la función para conectarse a orígenes de datos y administrar las conexiones. La plataforma se encarga de esta compleja tarea en su lugar como parte del código de enlace. Cada enlace tiene una dirección; el código lee datos desde enlaces de *entrada* y escribe datos en enlaces de *salida*. Cada función puede tener o no enlaces para administrar los datos de entrada y salida procesados por la función.

Un desencadenador es un tipo especial de enlace de entrada que tiene una capacidad adicional de iniciar la ejecución.

Azure proporciona un [gran número de enlaces](https://docs.microsoft.com/es-es/azure/azure-functions/functions-triggers-bindings#supported-bindings) para conectarse a servicios de mensajería y almacenamiento diferentes.

### **Definición de un enlace de ejemplo**

Veamos un ejemplo de configuración de una función con un enlace de entrada (desencadenador) y un enlace de salida. Supongamos que se quiere escribir una fila nueva en Azure Table Storage cada vez que aparezca un mensaje nuevo en Azure Queue Storage. Este escenario se puede implementar mediante un *desencadenador* de Azure Queue Storage y un *enlace de salida* de Azure Table Storage.

El fragmento de código siguiente es el archivo *function.json* para este escenario.

JSON

Copiar

{

"bindings": [

{

"name": "order",

"type": "queueTrigger",

"direction": "in",

"queueName": "myqueue-items",

"connection": "MY\_STORAGE\_ACCT\_APP\_SETTING"

},

{

"name": "$return",

"type": "table",

"direction": "out",

"tableName": "outTable",

"connection": "MY\_TABLE\_STORAGE\_ACCT\_APP\_SETTING"

}

]

}

La configuración de JSON especifica que la función se desencadenará cuando se agregue un mensaje a una cola denominada myqueue-items. Después, el valor devuelto de la función se escribe en la tabla outTable del almacenamiento de tablas de Azure. En el caso de las funciones de PowerShell, los enlaces de salida se escriben explícitamente con el cmdlet Push-OutputBinding.

Este ejemplo es una ilustración sencilla de cómo se configuran los enlaces para una función. Se podría cambiar la salida para que sea un correo electrónico con un enlace de SendGrid, colocar un evento en una instancia de Service Bus para notificar a otros componentes de la arquitectura o incluso tener varios enlaces de salida para insertar datos en varios servicios.

Sugerencia

Para ver y editar el contenido de *function.js* en Azure Portal, en el panel de menú izquierdo de la función, en Desarrollador, seleccione Code + Test (Código y prueba). En la ruta de acceso situada sobre el cuadro de código, seleccione *function.json* en la lista desplegable.

## **Creación Prueba de una función en Azure Portal**

Azure proporciona varias plantillas de función creadas previamente para escenarios comunes:

* Guía de inicio rápido
* Funciones personalizadas

### **Adición de plantillas de función**

Al agregar la primera función, se muestra el panel Agregar función, donde puede elegir el desencadenador de la función. Según las selecciones realizadas, Azure generará el código de función y la configuración con algún código de ejemplo proporcionado para mostrar los datos de entrada recibidos en el registro.

### **Plantillas de función personalizadas**

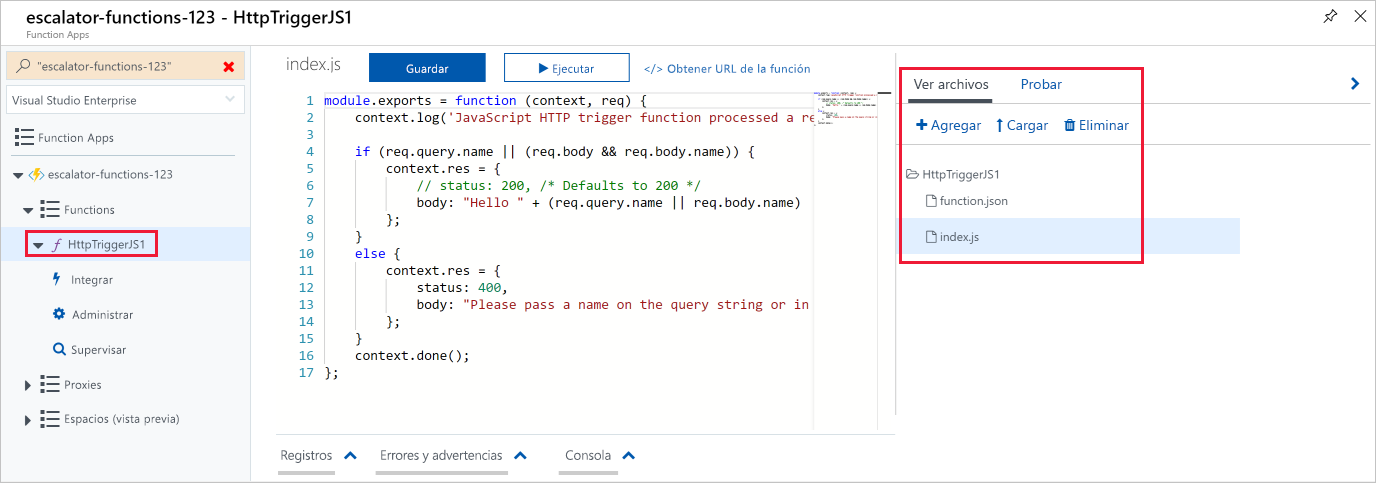
La selección de una plantilla en el panel Agregar función proporciona un acceso sencillo a las plantillas más comunes. Azure proporciona más de 30 plantillas adicionales para comenzar. Estas se pueden seleccionar en la lista de plantillas al crear las funciones subsiguientes.

## **Navegación a la función y los archivos**

Al crear una función a partir de una plantilla, se crean varios archivos. Por ejemplo, si optó por utilizar el inicio rápido Webhook + API mediante JavaScript, los archivos generados serían un archivo de configuración, function.json, y un archivo de código fuente, index.js.

Las funciones que cree en una aplicación de funciones aparecen al seleccionar Funciones en la categoría Funciones en el panel de menú izquierdo de la aplicación de funciones.

Al seleccionar una función en la aplicación de funciones, aparece el panel de la función. Para mostrar y editar el código de la función, en el panel de menú izquierdo, en Desarrollador, seleccione Code + Test (Código y prueba). En la ruta de acceso de la función que hay encima del cuadro de código, seleccione *function.json* en la lista desplegable y, luego, seleccione Test/Run (Probar o ejecutar) en la barra de menús superior para que aparezca el panel Entrada/salida. Vea la siguiente captura de pantalla.



Como puede ver, hay un panel a la derecha que incluye pestañas para Entrada y Salida. Al seleccionar la pestaña Entrada, se proporcionan parámetros para probar la solicitud HTTP de la función.

## **Prueba de la función de Azure**

Una vez creada la función, le interesa probarla. Estas son dos opciones:

* Ejecución manual
* Prueba desde el propio Azure Portal

### **Ejecución manual**

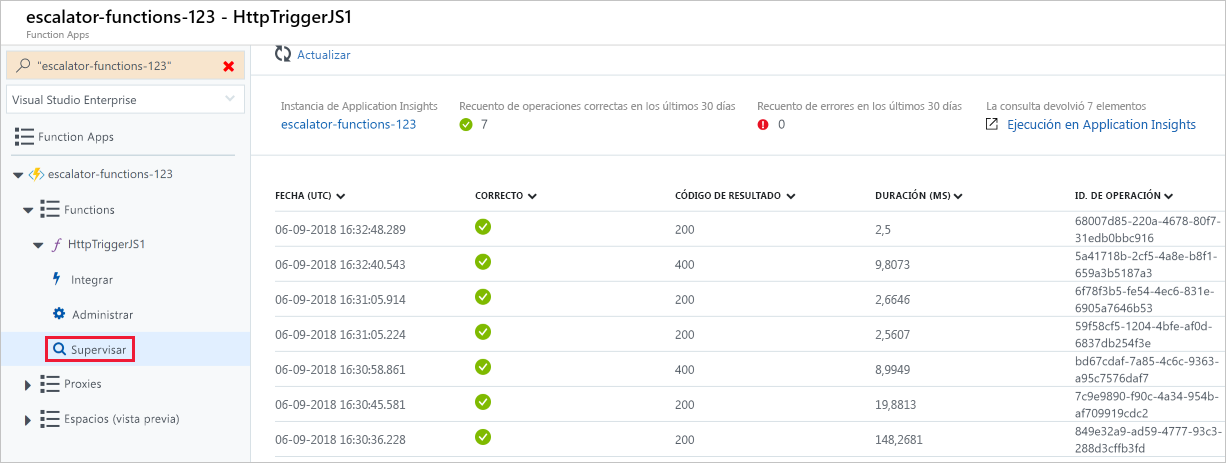
Puede iniciar una función si desencadena manualmente el desencadenador configurado. Por ejemplo, si usa un desencadenador de HTTP, puede usar una herramienta como Postman o cURL para iniciar una solicitud HTTP a la dirección URL del punto de conexión de función, que se encuentra disponible en la definición del desencadenador de HTTP (Obtener la dirección URL de la función).

### **Prueba en Azure Portal**

El portal también proporciona una manera cómoda de probar las funciones. Como se ha descrito anteriormente, después de seleccionar Test/Run (Probar y ejecutar) en la barra de menús superior, en el lado derecho del cuadro de código, aparece un panel Entrada/salida con pestañas. Aquí, puede proporcionar parámetros para probar la solicitud HTTP. Al seleccionar Ejecutar en este panel, los resultados aparecen en la pestaña Salida, junto con un código de estado en el panel Registros.

## **Panel de supervisión y de Application Insights**

La capacidad de registrar las funciones es fundamental durante el desarrollo y en producción. Azure Portal proporciona un panel de supervisión si activa la integración de Application Insights. En el panel de menú izquierdo de la aplicación de funciones, en Supervisión, seleccione Registros y abra Application Insights. Este panel de Application Insights ofrece una forma rápida de visualizar el historial de ejecuciones de funciones y muestra la marca de tiempo, el código de resultado, la duración y el identificador de la operación rellenados por Application Insights.



## **Panel de registros de streaming**

También se pueden agregar instrucciones de registro a la función para depuración en Azure Portal. A los métodos llamados en cada lenguaje se les pasa un objeto de "registro" que se puede usar para registrar información en el panel Registros que se encuentra en la parte inferior de la ventana de código en el panel Code + Test (Código y prueba) después de ejecutar una prueba.

El siguiente fragmento de código de JavaScript muestra cómo registrar un mensaje con el método context.log (el objeto context se pasa al controlador).

JavaScript

Copiar

context.log('Enter your logging statement here');

Se podría hacer lo mismo en C# con el método log.Info. En este caso, el objeto log se pasa al método de C# que procesa la función.

C#

Copiar

log.Info("Enter your logging statement here");

En PowerShell, use Write-Host para escribir en el registro:

PowerShell

Copiar

Write-Host "Enter your logging statement here"

### **Errores y advertencias del panel de registros**

Puede encontrar la pestaña de la ventana de errores y advertencias en el mismo menú flotante que la ventana de registro. En esta ventana se muestran los errores y las advertencias de compilación del código.

# **Ejercicio: Incorporación de lógica a la aplicación de funciones**

Completado

100 XP

* 10 minutos

Espacio aislado activado. Tiempo restante: 3 h 45 min

Usó 3 de 10 espacios aislados por hoy. Mañana habrá disponibles más espacios aislados.

Elija el lenguaje de desarrollo

JavaScript

PowerShell

Vamos a continuar con el ejemplo de transmisión por engranaje y a agregar la lógica para el servicio de temperatura. En concreto, vamos a recibir datos de una solicitud HTTP.

## **Requisitos de la función**

En primer lugar, es necesario definir algunos requisitos de la lógica:

* Las temperaturas entre 0 y 25 grados deberían marcarse como OK (Correcto).
* Las temperaturas entre 26 y 50 grados deberían marcarse como CAUTION (Precaución).
* Las temperaturas superiores a 50 grados deberían marcarse como DANGER (Peligro).

## **Incorporación de una función a la aplicación de función**

Como se ha explicado en la unidad anterior, Azure proporciona plantillas que ayudan a empezar a crear funciones. En esta unidad se usa la plantilla HttpTrigger para implementar el servicio de temperatura.

1. En el ejercicio anterior, después de que la aplicación de funciones se implementase correctamente en Azure Portal, seleccionó Ir al recurso. Se abre el panel escalator-functions-xxxxxxx de la aplicación de funciones.
2. En el panel de menús de la izquierda, en Funciones, seleccione Funciones. Aparece el panel Funciones de la aplicación de funciones.
3. En la barra de menús superior, seleccione Agregar. Aparece el panel Agregar función.
4. En el panel Agregar función, en Seleccionar una plantilla, haga clic en Desencadenador de HTTP y Agregar. Aparecerá el panel de la función HttpTrigger1.
5. En el panel de menús izquierdo, en Desarrollador, seleccione Código y prueba. Aparece el editor de código.
6. En el menú desplegable de origen, seleccione *index.js*. El código predeterminado que la plantilla ha generado de forma automática aparece en el siguiente fragmento de código.
7. JavaScript
8. Copiar

module.exports = async function (context, req) {

context.log('JavaScript HTTP trigger function processed a request.');

const name = (req.query.name || (req.body && req.body.name));

const responseMessage = name

? "Hello, " + name + ". This HTTP triggered function executed successfully."

: "This HTTP triggered function executed successfully. Pass a name on the query string or in the request body for a personalized response.";

context.res = {

// status: 200, /\* Defaults to 200 \*/

body: responseMessage

};

}

1. Nuestra función espera que se pase un nombre mediante una cadena de consulta de solicitud HTTP o como parte del cuerpo de la solicitud. La función responde con el mensaje Hola, {name}, donde devuelve el nombre que se ha enviado en la solicitud.  
   De nuevo, en el menú desplegable de origen, seleccione function.json para ver la configuración de la función, que debería tener un aspecto similar al siguiente código.
2. JavaScript
3. Copiar

{

"bindings": [

{

"authLevel": "function",

"type": "httpTrigger",

"direction": "in",

"name": "req",

"methods": [

"get",

"post"

]

},

{

"type": "http",

"direction": "out",

"name": "res"

}

]

}

1. Esta configuración declara que la función se ejecuta cuando recibe una solicitud HTTP. El enlace de salida declara que la respuesta se va a enviar como una respuesta HTTP.

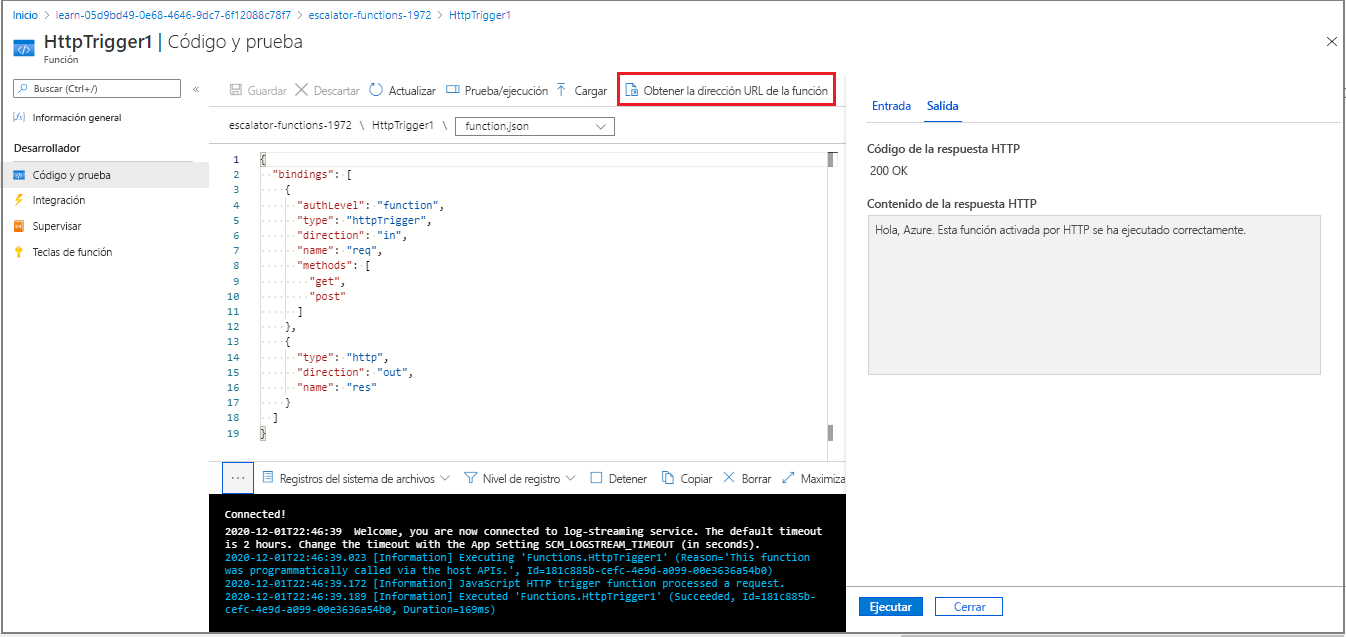
## **Probar la función**

Sugerencia

cURL es una herramienta de línea de comandos que se puede usar para enviar o recibir archivos. Se incluye con Linux, macOS y Windows 10 y puede descargarse para la mayoría de los demás sistemas operativos. cURL admite numerosos protocolos, como HTTP, HTTPS, FTP, FTPS, SFTP, LDAP, TELNET, SMTP, POP3, etc. Para obtener más información, vea los siguientes vínculos:

* <https://en.wikipedia.org/wiki/CURL>
* <https://curl.haxx.se/docs/>

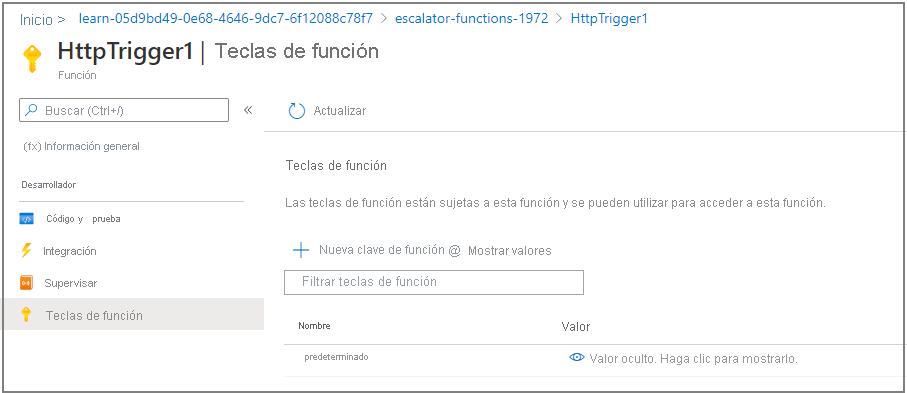
Para probar la función, puede enviar una solicitud HTTP a la dirección URL de la función con cURL en la línea de comandos.

1. Para buscar la dirección URL del punto de conexión de la función, expanda el marco Registro en la parte inferior del panel de la función de desencadenador.
2. En la barra de menús superior, seleccione Obtener la dirección URL de la función, como se muestra en la imagen siguiente. Para guardar este vínculo, seleccione el icono *Copiar al portapapeles* situado al final de la dirección URL. Almacene el vínculo en el Bloc de notas o en una aplicación similar para usarlo más adelante.  
   

### **Protección de desencadenadores de HTTP**

Los desencadenadores HTTP permiten usar claves de API para bloquear los llamadores desconocidos al requerir que la clave esté presente en cada solicitud. Cuando crea una función, se selecciona el *nivel de autorización*. De forma predeterminada, está establecido en *Función*, para lo que se requiere una clave de API específica para la función, pero también puede estar establecido en *Administrador* para usar una clave "maestra" global, o bien en *Anónimo* para indicar que no se requiere ninguna clave. También puede cambiar el nivel de autorización en las propiedades de la función tras su creación.

Como se ha especificado *Función* al crear esta función, será necesario proporcionar la clave al enviar la solicitud HTTP. Puede enviarla como un parámetro de cadena de consulta denominado code o como un encabezado HTTP (preferido) llamado x-functions-key.

1. Para buscar las claves maestras y la función, en el panel de menú de la izquierda, en Desarrollador, seleccione Claves de función. De forma predeterminada, están ocultas y necesita mostrarlas.
2. Muestre la tecla de función predeterminada y cópiela en el Portapapeles. Almacene esta clave en el Bloc de notas o en una aplicación similar para usarla más adelante.  
   
3. En la parte inferior de la pantalla, desplácese hacia la izquierda y seleccione la función. En la parte superior, en la sección Obtener la dirección URL de la función, copie la URL. Para ello, seleccione el icono *Copiar al Portapapeles* que encontrará al final de la dirección URL. Almacene el vínculo en el Bloc de notas o en una aplicación similar para usarlo más adelante.
4. Después, desplácese hacia la izquierda y, en el panel de menú de la izquierda, en Funciones, seleccione Funciones y luego HttpTrigger1.
5. En el panel de menús de la izquierda, en Desarrollador, seleccione Código y prueba y, en la barra de menús superior, seleccione Probar/ejecutar. Aparece el panel Pruebas.
6. En la pestaña Entrada, en el cuadro de texto Cuerpo, sobrescriba el código insertado; para ello, pegue el siguiente comando de cURL.
7. Bash
8. Copiar

curl --header "Content-Type: application/json" --header "x-functions-key: <your-function-key>" --request POST --data "{\"name\": \"Azure Function\"}" https://<your-url-here>

1. Reemplace los marcadores de posición del código por la dirección URL de la función y la clave de función que guardó anteriormente en el Bloc de notas o en una aplicación similar. Tome nota de estos aspectos del comando de cURL:
   * Ha usado una solicitud POST.
   * Ha agregado un valor de encabezado Content-Type de tipo application/json.
   * Se ha asegurado de reemplazar la dirección URL por la suya.
   * Ha pasado la tecla de función como el valor de encabezado x-functions-key.
2. Seleccione Run (Ejecutar).  
   En el panel Salida, para Código de respuesta HTTP, la función responde con el texto "200 OK".  
    Precaución  
   Si está en Windows, ejecute cURL en el símbolo del sistema. PowerShell tiene un comando *curl*, pero es un alias de Invoke-WebRequest y no es lo mismo que cURL.  
    Nota  
   También puede realizar las pruebas desde la sección de una función individual con el panel Pruebas del lateral de una función seleccionada. No podrá comprobar que el sistema de claves de función funciona, ya que no es necesario en este caso. Agregue el encabezado adecuado y los valores de parámetro en la interfaz Pruebas y seleccione Ejecutar para ver el resultado de la prueba.

## **Adición de lógica de negocios a la función**

Después, se agrega la lógica a la función que comprueba las lecturas de temperatura que recibe y establece un estado para cada una.

La función espera una matriz de lecturas de temperatura. El siguiente fragmento de código JSON es un ejemplo del cuerpo de la solicitud que enviaremos a nuestra función. Cada entrada reading tiene un identificador, la marca de tiempo y la temperatura.

JSON

Copiar

{

"readings": [

{

"driveGearId": 1,

"timestamp": 1534263995,

"temperature": 23

},

{

"driveGearId": 3,

"timestamp": 1534264048,

"temperature": 45

},

{

"driveGearId": 18,

"timestamp": 1534264050,

"temperature": 55

}

]

}

Vamos a reemplazar el código predeterminado de la función por el siguiente código que implementa la lógica de negocios.

En el panel de la función HttpTrigger1, abra el archivo index.js y reemplácelo por el código siguiente. Asegúrese de guardar el archivo después de actualizarlo.

JavaScript

Copiar

module.exports = function (context, req) {

context.log('Drive Gear Temperature Service triggered');

if (req.body && req.body.readings) {

req.body.readings.forEach(function(reading) {

if(reading.temperature<=25) {

reading.status = 'OK';

} else if (reading.temperature<=50) {

reading.status = 'CAUTION';

} else {

reading.status = 'DANGER'

}

context.log('Reading is ' + reading.status);

});

context.res = {

// status: 200, /\* Defaults to 200 \*/

body: {

"readings": req.body.readings

}

};

}

else {

context.res = {

status: 400,

body: "Please send an array of readings in the request body"

};

}

context.done();

};

La lógica que se agrega es sencilla. Recorremos la matriz de lecturas y comprobamos el campo de temperatura. Dependiendo del valor de ese campo, establecemos un estado de OK (Correcto), CAUTION (Precaución) o DANGER (Peligro). A continuación, devolvemos la matriz de lecturas con un campo de estado agregado a cada entrada.

Observe las instrucciones Log cuando expande Registros en la parte inferior del panel. Cuando se ejecute la función, estas instrucciones agregarán los mensajes en la ventana Registros.

## **Prueba de la lógica de negocios**

En este caso, vamos a usar el panel Probar del portal para probar la función.

1. En el panel Pruebas, pegue la solicitud de ejemplo en el cuadro de texto Cuerpo de la solicitud.
2. JSON
3. Copiar

{

"readings": [

{

"driveGearId": 1,

"timestamp": 1534263995,

"temperature": 23

},

{

"driveGearId": 3,

"timestamp": 1534264048,

"temperature": 45

},

{

"driveGearId": 18,

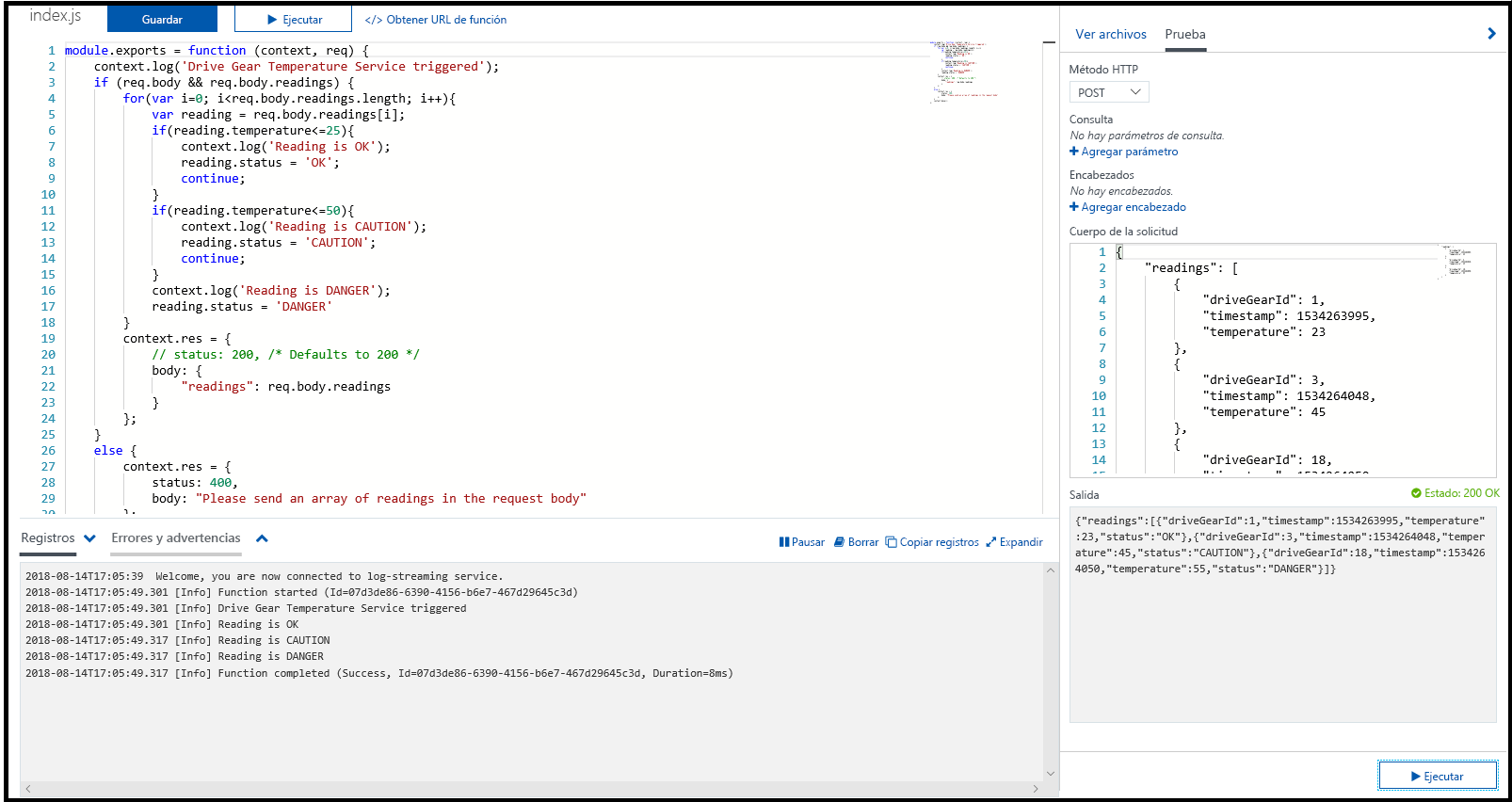
"timestamp": 1534264050,

"temperature": 55

}

]

}

1. Seleccione Ejecutar y vea la respuesta en el panel Salida. Para ver los mensajes de registro, abra la pestaña Registros en el control flotante de la parte inferior del panel. En la imagen siguiente se muestra un ejemplo de respuesta en el panel de salida y los mensajes en el panel Registros.  
     
   Puede ver en el panel Salida que nuestro campo de estado se ha agregado correctamente a cada una de las lecturas.  
   Para ver que la solicitud se ha registrado en Application Insights, en el panel de menú izquierdo, en Supervisión, seleccione Insights. Aparece el panel Insights correspondiente a la función de desencadenador.  
   