**UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE**

**SEDE SANTO DOMINGO DE LOS TSÁCHILAS**

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN - DCCO-SS**

**CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**

| | | |
|---|---|---|
| **PERIODO** | : | 2024 Octubre – Marzo 2025 |
| **ASIGNATURA** | : | Aplicación Distribuida |
| **TEMA** | : | Laboratorio 2 |
| **ESTUDIANTES** | : | Grupo 3 |
| **NIVEL-PARALELO - NRC** | : | Octavo Semestre |
| **DOCENTE** | : | Ing. Pablo Puente. |
| **FECHA DE ENTREGA** | : | 31 de Enero de 2025 |

**SANTO DOMINGO - ECUADOR**

**2025**

**Aplicación Distribuida**

## Índice de Contenido

**TEMA:** Funciones serverless.

## 1. Introducción

Las funciones serverless representan un paradigma en la computación en la nube que permite a los desarrolladores ejecutar código sin preocuparse por la gestión de servidores. Este modelo se basa en la ejecución de funciones en respuesta a eventos, lo que facilita la escalabilidad, optimización de costos y mantenimiento reducido. Al utilizar proveedores como AWS Lambda, Google Cloud Functions o Azure Functions, las empresas pueden desplegar aplicaciones sin necesidad de administrar infraestructura, pagando únicamente por el tiempo de ejecución del código. Esto ha permitido el desarrollo ágil de aplicaciones, integraciones y automatización de procesos en diversos entornos tecnológicos.

## 2. Objetivos

### 2.1. Objetivo General:

Desarrollar e implementar funciones serverless para optimizar la ejecución de procesos en una aplicación web, garantizando escalabilidad, eficiencia y reducción de costos operativos en la infraestructura de TI.

### 2.2. Objetivos Específicos:

- Comprender el modelo serverless y su impacto en el desarrollo de aplicaciones modernas, explorando ventajas y limitaciones.
- Diseñar e implementar funciones serverless utilizando plataformas como AWS Lambda, definiendo eventos y flujos de ejecución eficientes.
- Integrar las funciones con servicios en la nube, como bases de datos, colas de mensajes y sistemas de autenticación.

## 3. Desarrollo.

### Iniciar un nuevo proyecto Azure Functions

Inicializamos un proyecto en Azure Functions en JavaScript con el comando func init serverlesslab2 --javascript, lo que genera archivos clave como package.json, .gitignore, host.json y local.settings.json, además de una configuración para VS Code.

```
\Unidad 2>func init serverlesslab2 --javascript
```

```
Writing package.json
Writing .funcignore
Writing .gitignore
Writing host.json
Writing local.settings.json
Writing C:\Users\hp\Desktop\DEV\ESPE\
ons.json
Running 'npm install'...
```

Creamos una nueva función en Azure Functions con el comando func new, lo que nos permite seleccionar una plantilla de disparador (trigger). En este caso, elegimos "HTTP trigger", que ejecutará la función cuando reciba una solicitud HTTP. Esta opción es ideal para desarrollar APIs serverless o servicios web que respondan a peticiones GET, POST y otros métodos HTTP.

```
\serverlesslab2>func new
```

```
Use the up/down arrow keys to select a template:
Azure Blob Storage trigger
Azure Cosmos DB trigger
Durable Functions entity
Durable Functions orchestrator
Azure Blob Storage trigger (using Event Grid)
Azure Event Grid trigger
Azure Event Hub trigger
HTTP trigger
Azure Queue Storage trigger
Azure Service Bus Queue trigger
Azure Service Bus Topic trigger
Timer trigger
Dapr Publish Output Binding
Dapr Service Invocation Trigger
Dapr Topic Trigger
```

Creamos una nueva función en Azure Functions con el comando func new y seleccionamos la plantilla "HTTP trigger". Luego, asignamos el nombre "generarRegistros" a la función, lo

que generó automáticamente el archivo generarRegistros.js en la ruta del proyecto. Esta función responderá a solicitudes HTTP y fue creada exitosamente a partir de la plantilla seleccionada.

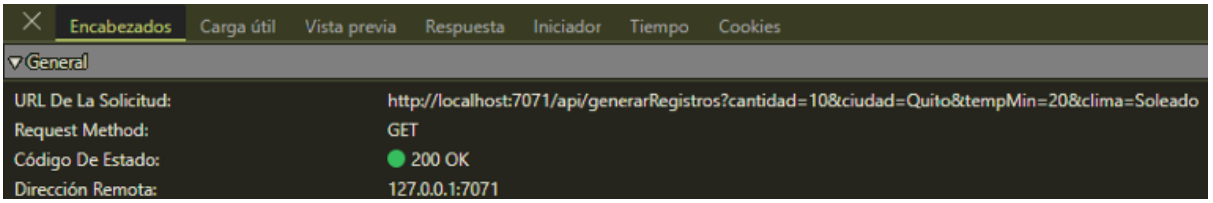
```
\serverlesslab2>func new
```

```
Use the up/down arrow keys to select a template:HTTP trigger
Function name: [httpTrigger] generarRegistros
```

Iniciamos la ejecución de nuestra función Azure Functions con el comando func start, lo que activa el servidor de desarrollo local. El proceso muestra que la función generarRegistros está disponible y puede ser invocada a través de una solicitud HTTP GET en la URL http://localhost:7071/api/generarRegistros. Esto permite probar la función de manera local antes de desplegarla en Azure.

```
> serverlesslab2@1.0.0 start
> func start


Azure Functions Core Tools
Core Tools Version:       4.0.6821 Commit hash: N/A +c09a2033faa7ecf51b377330
Function Runtime Version: 4.1036.1.23224

[2025-01-25T18:08:23.423Z] Worker process started and initialized.

Functions:

        generarRegistros: [GET] http://localhost:7071/api/generarRegistros
```

Realizamos una solicitud GET a la función generarRegistros con parámetros en la URL para filtrar los datos generados.



## 4. Conclusiones

- La implementación de funciones serverless permite un uso eficiente de los recursos computacionales, ya que solo se ejecutan cuando son necesarias, lo que reduce costos y optimiza el rendimiento.
- Al eliminar la necesidad de administrar servidores, los desarrolladores pueden enfocarse en la lógica de negocio y acelerar el proceso de desarrollo e implementación de nuevas funcionalidades.

## 5. Recomendaciones

- Antes de implementar funciones serverless, es fundamental analizar los distintos proveedores disponibles (AWS Lambda, Google Cloud Functions, Azure Functions) y elegir el que mejor se adapte a las necesidades del proyecto.
- Para evitar costos innecesarios y mejorar el rendimiento, se recomienda minimizar la latencia en la ejecución de las funciones mediante la optimización del código y la reducción de dependencias externas.

## 6. Bibliografía

Guia impartida por el docente