**Laboratorio 04:**

**Construyendo una Aplicación WebApi y un Cliente Web Estático**

****

Alumno: Marcelo José Medina López - 2020066917

Docente: Mag. Patrick José Cuadros Quiroga

Curso: Programación Web II

Código de curso: SI-982

Facultad: Facultad de Ingeniería

Escuela: Escuela Profesional de Ingeniería de Sistemas (EPIS)

TACNA – PERÚ

2025

**INDICE:**

[SESION DE LABORATORIO N° 01: Construyendo una Aplicación Web con ASP.NET MVC 3](#_Toc187691042)

[OBJETIVOS 3](#_Toc187691043)

[REQUERIMIENTOS 3](#_Toc187691044)

[CONSIDERACIONES INICIALES 3](#_Toc187691045)

[DESARROLLO 4](#_Toc187691046)

[PARTE I: PREPARACION DE LA INFRAESTRUCTURA 4](#_Toc187691047)

[PARTE II: CONSTRUCCION DE LA APLICACION 5](#_Toc187691048)

[PARTE III: DESPLIEGUE DE LA APLICACION 8](#_Toc187691049)

[ACTIVIDADES ENCARGADAS 10](#_Toc187691050)

# SESION DE LABORATORIO N° 04: Construyendo una Aplicación Web con ASP.NET MVC

## OBJETIVOS

* Comprender el desarrollo una Aplicación Web utilizando ASP.NET MVC

## REQUERIMIENTOS

* Conocimientos:
  + Conocimientos básicos de SQL.
  + Conocimientos shell y comandos en modo terminal.
* Hardware:
  + Virtualization activada en el BIOS.
  + CPU SLAT-capable feature.
  + Al menos 4GB de RAM.
* Software:
  + Windows 10 64bit: Pro, Enterprise o Education (1607 Anniversary Update, Build 14393 o Superior)
  + Docker Desktop
  + Powershell versión 7.x
  + .Net 8
  + Azure CLI
  + Azure Functions Core Tools (winget install Microsoft.Azure.FunctionsCoreTools)
  + Azure Static WebApps CLI (npm install -g @azure/static-web-apps-cli)

## CONSIDERACIONES INICIALES

* Tener una cuenta en Infracost (https://www.infracost.io/), sino utilizar su cuenta de github para generar su cuenta y generar un token.
* Tener una cuenta en SonarCloud (https://sonarcloud.io/), sino utilizar su cuenta de github para generar su cuenta y generar un token. El token debera estar registrado en su repositorio de Github con el nombre de SONAR\_TOKEN.
* Tener una cuenta con suscripción en Azure (https://portal.azure.com/). Tener el ID de la Suscripción, que se utilizará en el laboratorio
* Clonar el repositorio mediante git para tener los recursos necesarios en una ubicación que no sea restringida del sistema.

## DESARROLLO

### PARTE I: PREPARACION DE LA INFRAESTRUCTURA

1. Iniciar la aplicación Powershell o Windows Terminal en modo administrador, ubicarse en una ruta donde se ha realizado la clonación del repositorio.

*md Infra*

1. Abrir Visual Studio Code, seguidamente abrir la carpeta del repositorio clonado del laboratorio, en el folder Infra, crear el archivo main.tf con el siguiente contenido.

*terraform {*

*required\_providers {*

*azurerm = {*

*source = "hashicorp/azurerm"*

*version = "~> 4.0.0"*

*}*

*}*

*required\_version = ">= 0.14.9"*

*}*

*variable "suscription\_id" {*

*type = string*

*description = "Azure subscription id"*

*}*

*variable "sqladmin\_username" {*

*type = string*

*description = "Administrator username for server"*

*}*

*variable "sqladmin\_password" {*

*type = string*

*description = "Administrator password for server"*

*}*

*provider "azurerm" {*

*features {}*

*subscription\_id = var.suscription\_id*

*}*

*# Generate a random integer to create a globally unique name*

*resource "random\_integer" "ri" {*

*min = 100*

*max = 999*

*}*

*# Create the resource group*

*resource "azurerm\_resource\_group" "rg" {*

*name = "upt-arg-${random\_integer.ri.result}"*

*location = "eastus2"*

*}*

*resource "azurerm\_storage\_account" "storageaccount" {*

*name = "uptasa${random\_integer.ri.result}"*

*location = azurerm\_resource\_group.rg.location*

*resource\_group\_name = azurerm\_resource\_group.rg.name*

*account\_tier = "Standard"*

*account\_replication\_type = "LRS"*

*}*

*# Create the Linux App Service Plan*

*resource "azurerm\_service\_plan" "appserviceplan" {*

*name = "upt-asp-${random\_integer.ri.result}"*

*location = azurerm\_resource\_group.rg.location*

*resource\_group\_name = azurerm\_resource\_group.rg.name*

*os\_type = "Linux"*

*sku\_name = "Y1"*

*}*

*resource "azurerm\_linux\_function\_app" "azurefunction" {*

*name = "upt-afn-${random\_integer.ri.result}"*

*location = azurerm\_resource\_group.rg.location*

*resource\_group\_name = azurerm\_resource\_group.rg.name*

*storage\_account\_name = azurerm\_storage\_account.storageaccount.name*

*storage\_account\_access\_key = azurerm\_storage\_account.storageaccount.primary\_access\_key*

*service\_plan\_id = azurerm\_service\_plan.appserviceplan.id*

*site\_config {*

*minimum\_tls\_version = "1.2"*

*always\_on = false*

*application\_stack {*

*dotnet\_version = "8.0"*

*}*

*}*

*}*

*resource "azurerm\_static\_web\_app" "example" {*

*name = "upt-swa-${random\_integer.ri.result}"*

*location = azurerm\_resource\_group.rg.location*

*resource\_group\_name = azurerm\_resource\_group.rg.name*

*}*

*resource "azurerm\_mssql\_server" "sqlsrv" {*

*name = "upt-dbs-${random\_integer.ri.result}"*

*resource\_group\_name = azurerm\_resource\_group.rg.name*

*location = azurerm\_resource\_group.rg.location*

*version = "12.0"*

*administrator\_login = var.sqladmin\_username*

*administrator\_login\_password = var.sqladmin\_password*

*}*

*resource "azurerm\_mssql\_firewall\_rule" "sqlaccessrule" {*

*name = "PublicAccess"*

*server\_id = azurerm\_mssql\_server.sqlsrv.id*

*start\_ip\_address = "0.0.0.0"*

*end\_ip\_address = "255.255.255.255"*

*}*

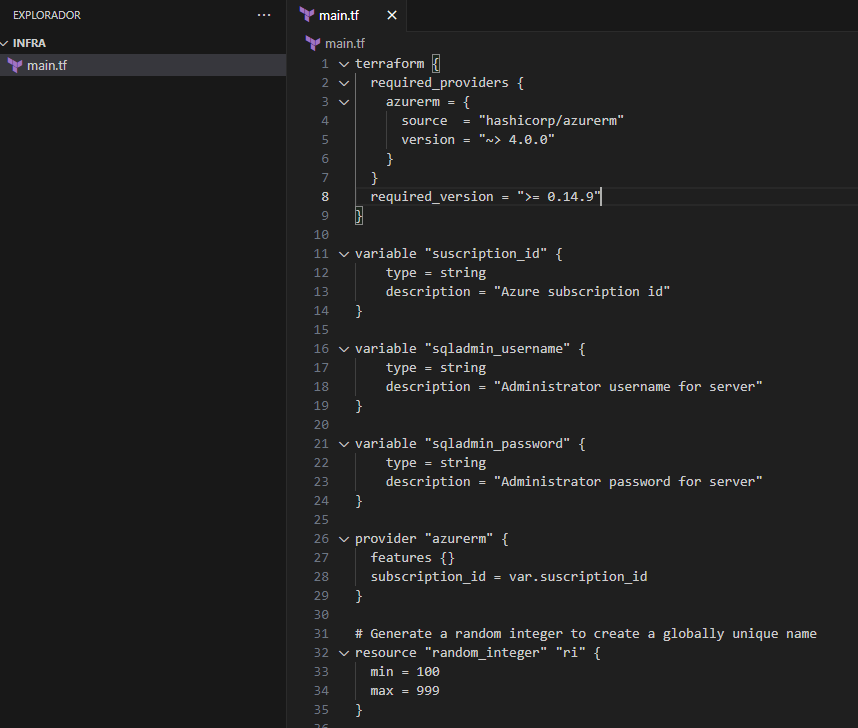
*resource "azurerm\_mssql\_database" "sqldb" {*

*name = "shorten"*

*server\_id = azurerm\_mssql\_server.sqlsrv.id*

*sku\_name = "Free"*

*}*

**

1. Abrir un navegador de internet y dirigirse a su repositorio en Github, en la sección Settings, buscar la opción Secrets and Variables y seleccionar la opción Actions. Dentro de esta crear los siguientes secretos

* AZURE\_USERNAME: Correo o usuario de cuenta de Azure
* AZURE\_PASSWORD: Password de cuenta de Azure
* SUSCRIPTION\_ID: ID de la Suscripción de cuenta de Azure
* SQL\_USER: Usuario administrador de la base de datos, ejm: adminsql
* SQL\_PASS: Password del usuario administrador de la base de datos, ejm: upt.2025



1. En el Visual Studio Code, crear la carpeta .github/workflows en la raiz del proyecto, seguidamente crear el archivo deploy.yml con el siguiente contenido

*name: Construcción infrastructura en Azure*

*on:*

*push:*

*branches: [ "main" ]*

*paths:*

*- 'infra/\*\*'*

*- '.github/workflows/infra.yml'*

*workflow\_dispatch:*

*jobs:*

*Deploy-infra:*

*runs-on: ubuntu-latest*

*steps:*

*- uses: actions/checkout@v4*

*- name: login azure*

*run: |*

*az login -u ${{ secrets.AZURE\_USERNAME }} -p ${{ secrets.AZURE\_PASSWORD }}*

*- name: Create terraform.tfvars*

*run: |*

*cd infra*

*echo "suscription\_id=\"${{ secrets.SUSCRIPTION\_ID }}\"" > terraform.tfvars*

*echo "sqladmin\_username=\"${{ secrets.SQL\_USER }}\"" >> terraform.tfvars*

*echo "sqladmin\_password=\"${{ secrets.SQL\_PASS }}\"" >> terraform.tfvars*

*# - name: Setup tfsec*

*# run: |*

*# curl -L -o /tmp/tfsec\_1.28.13\_linux\_amd64.tar.gz "https://github.com/aquasecurity/tfsec/releases/download/v1.28.13/tfsec\_1.28.13\_linux\_amd64.tar.gz"*

*# tar -xzvf /tmp/tfsec\_1.28.13\_linux\_amd64.tar.gz -C /tmp*

*# mv -v /tmp/tfsec /usr/local/bin/tfsec*

*# chmod +x /usr/local/bin/tfsec*

*# - name: tfsec*

*# run: |*

*# cd infra*

*# /usr/local/bin/tfsec --format=markdown --tfvars-file=terraform.tfvars --out=tfsec.md .*

*# echo "## TFSec Output" >> $GITHUB\_STEP\_SUMMARY*

*# cat tfsec.md >> $GITHUB\_STEP\_SUMMARY*

*- name: Setup Terraform*

*uses: hashicorp/setup-terraform@v3*

*- name: Terraform Init*

*id: init*

*run: cd infra && terraform init*

*# - name: Terraform Fmt*

*# id: fmt*

*# run: cd infra && terraform fmt -check*

*- name: Terraform Validate*

*id: validate*

*run: cd infra && terraform validate -no-color*

*- name: Terraform Plan*

*run: cd infra && terraform plan -var="suscription\_id=${{ secrets.SUSCRIPTION\_ID }}" -var="sqladmin\_username=${{ secrets.SQL\_USER }}" -var="sqladmin\_password=${{ secrets.SQL\_PASS }}" -no-color -out main.tfplan*

*- name: Create String Output*

*id: tf-plan-string*

*run: |*

*TERRAFORM\_PLAN=$(cd infra && terraform show -no-color main.tfplan)*

*delimiter="$(openssl rand -hex 8)"*

*echo "summary<<${delimiter}" >> $GITHUB\_OUTPUT*

*echo "## Terraform Plan Output" >> $GITHUB\_OUTPUT*

*echo "<details><summary>Click to expand</summary>" >> $GITHUB\_OUTPUT*

*echo "" >> $GITHUB\_OUTPUT*

*echo '```terraform' >> $GITHUB\_OUTPUT*

*echo "$TERRAFORM\_PLAN" >> $GITHUB\_OUTPUT*

*echo '```' >> $GITHUB\_OUTPUT*

*echo "</details>" >> $GITHUB\_OUTPUT*

*echo "${delimiter}" >> $GITHUB\_OUTPUT*

*- name: Publish Terraform Plan to Task Summary*

*env:*

*SUMMARY: ${{ steps.tf-plan-string.outputs.summary }}*

*run: |*

*echo "$SUMMARY" >> $GITHUB\_STEP\_SUMMARY*

*- name: Outputs*

*id: vars*

*run: |*

*echo "terramaid\_version=$(curl -s https://api.github.com/repos/RoseSecurity/Terramaid/releases/latest | grep tag\_name | cut -d '"' -f 4)" >> $GITHUB\_OUTPUT*

*case "${{ runner.arch }}" in*

*"X64" )*

*echo "arch=x86\_64" >> $GITHUB\_OUTPUT*

*;;*

*"ARM64" )*

*echo "arch=arm64" >> $GITHUB\_OUTPUT*

*;;*

*esac*

*- name: Setup Go*

*uses: actions/setup-go@v5*

*with:*

*go-version: 'stable'*

*- name: Setup Terramaid*

*run: |*

*curl -L -o /tmp/terramaid.tar.gz "https://github.com/RoseSecurity/Terramaid/releases/download/${{ steps.vars.outputs.terramaid\_version }}/Terramaid\_Linux\_${{ steps.vars.outputs.arch }}.tar.gz"*

*tar -xzvf /tmp/terramaid.tar.gz -C /tmp*

*mv -v /tmp/Terramaid /usr/local/bin/terramaid*

*chmod +x /usr/local/bin/terramaid*

*- name: Terramaid*

*id: terramaid*

*run: |*

*cd infra*

*/usr/local/bin/terramaid run*

*- name: Publish graph in step comment*

*run: |*

*echo "## Terramaid Graph" >> $GITHUB\_STEP\_SUMMARY*

*cat infra/Terramaid.md >> $GITHUB\_STEP\_SUMMARY*

*- name: Setup Graphviz*

*uses: ts-graphviz/setup-graphviz@v2*

*- name: Setup inframap*

*run: |*

*curl -L -o /tmp/inframap.tar.gz "https://github.com/cycloidio/inframap/releases/download/v0.7.0/inframap-linux-amd64.tar.gz"*

*tar -xzvf /tmp/inframap.tar.gz -C /tmp*

*mv -v /tmp/inframap-linux-amd64 /usr/local/bin/inframap*

*chmod +x /usr/local/bin/inframap*

*- name: inframap*

*run: |*

*cd infra*

*/usr/local/bin/inframap generate main.tf --raw | dot -Tsvg > inframap\_azure.svg*

*- name: Upload inframap*

*id: inframap-upload-step*

*uses: actions/upload-artifact@v4*

*with:*

*name: inframap\_azure.svg*

*path: infra/inframap\_azure.svg*

*- name: Setup infracost*

*uses: infracost/actions/setup@v3*

*with:*

*api-key: ${{ secrets.INFRACOST\_API\_KEY }}*

*- name: infracost*

*run: |*

*cd infra*

*infracost breakdown --path . --format html --out-file infracost-report.html*

*sed -i '19,137d' infracost-report.html*

*sed -i 's/$0/$ 0/g' infracost-report.html*

*- name: Convert HTML to Markdown*

*id: html2markdown*

*uses: rknj/html2markdown@v1.1.0*

*with:*

*html-file: "infra/infracost-report.html"*

*- name: Upload infracost report*

*run: |*

*echo "## infracost Report" >> $GITHUB\_STEP\_SUMMARY*

*echo "${{ steps.html2markdown.outputs.markdown-content }}" >> infracost.md*

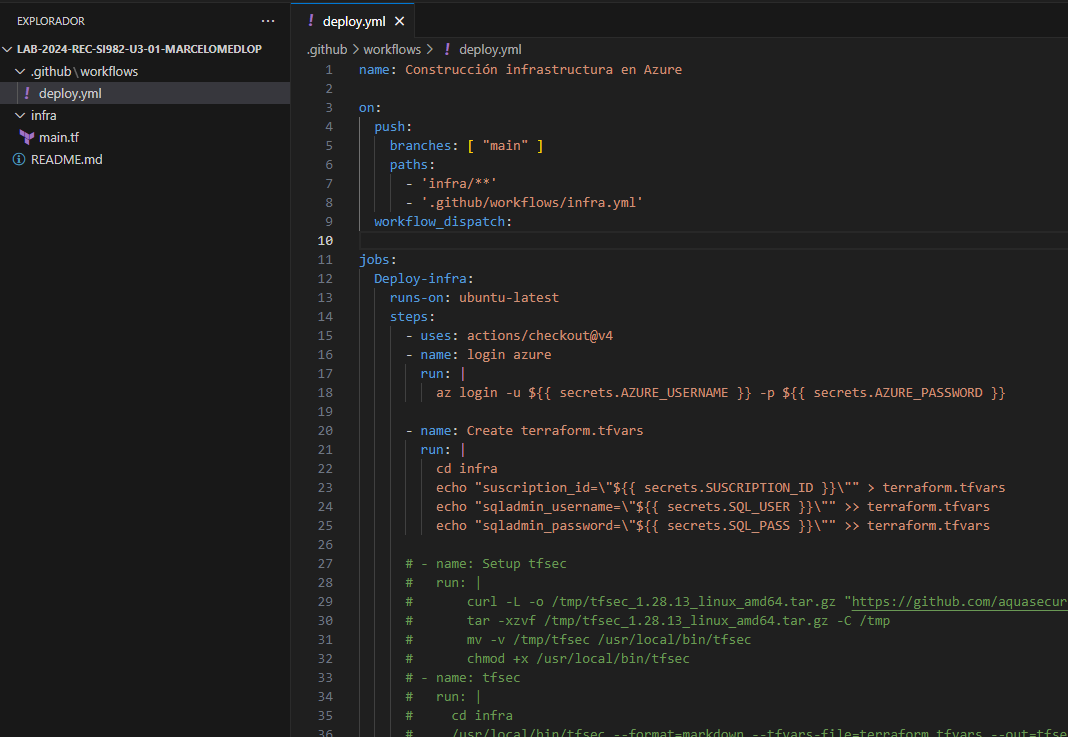
*cat infracost.md >> $GITHUB\_STEP\_SUMMARY*

*- name: Terraform Apply*

*run: |*

*cd infra*

*terraform apply -var="suscription\_id=${{ secrets.SUSCRIPTION\_ID }}" -var="sqladmin\_username=${{ secrets.SQL\_USER }}" -var="sqladmin\_password=${{ secrets.SQL\_PASS }}" -auto-approve main.tfplan*

**

1. En el Visual Studio Code, guardar los cambios y subir los cambios al repositorio. Revisar los logs de la ejeuciòn de automatizaciòn y anotar el número de identificación de Grupo de Recursos y Aplicación Web creados

*azurerm\_linux\_web\_app.webapp: Creation complete after 53s [id=/subscriptions/1f57de72-50fd-4271-8ab9-3fc129f02bc0/resourceGroups/upt-arg-XXX/providers/Microsoft.Web/sites/upt-awa-XXX]*

### PARTE II: CONSTRUCCION DE LA APLICACION

1. En el terminal, ubicarse en un ruta que no sea del sistema y ejecutar los siguientes comandos.

*dotnet new mvc -o src -n Shorten*

*cd src*

*dotnet tool install -g dotnet-aspnet-codegenerator --version 8.0.0*

*dotnet add package Microsoft.EntityFrameworkCore.InMemory --version=8.0.0*

*dotnet add package Microsoft.EntityFrameworkCore.SqlServer --version=8.0.0*

*dotnet add package Microsoft.EntityFrameworkCore.Tools --version=8.0.0*

*dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design --version=8.0.0*

*dotnet add package Microsoft.AspNetCore.Components.QuickGrid --version=8.0.0*

*dotnet add package Microsoft.AspNetCore.Components.QuickGrid.EntityFrameworkAdapter --version=8.0.0*

1. En el Visual Studio Code, en la carpeta src/Models, crear el archivo UrlMapping.cs con el siguiente contenido:

namespace Shorten.Models;

/// <summary>

/// Clase de dominio que representa una acortaciòn de url

/// </summary>

public class UrlMapping

{

/// <summary>

/// Identificador del mapeo de url

/// </summary>

/// <value>Entero</value>

public int Id { get; set; }

/// <summary>

/// Valor original de la url

/// </summary>

/// <value>Cadena</value>

public string OriginalUrl { get; set; } = string.Empty;

/// <summary>

/// Valor corto de la url

/// </summary>

/// <value>Cadena</value>

public string ShortenedUrl { get; set; } = string.Empty;

}

1. En el Visual Studio Code, en la carpeta src/Models, crear el archivo ShortContext.cs con el siguiente contenido:

*using Microsoft.EntityFrameworkCore;*

*namespace Shorten.Models;*

*/// <summary>*

*/// Clase de infraestructura que representa el contexto de la base de datos*

*/// </summary>*

*public class ShortContext : DbContext*

*{*

*/// <summary>*

*/// Constructor de la clase*

*/// </summary>*

*/// <param name="options">opciones de conexiòn de BD</param>*

*public ShortContext(DbContextOptions<ShortContext> options) : base(options)*

*{*

*}*

*/// <summary>*

*/// Propiedad que representa la tabla de mapeo de urls*

*/// </summary>*

*/// <value>Conjunto de UrlMapping</value>*

*public DbSet<UrlMapping> UrlMappings { get; set; }*

*}*

1. En el Visual Studio Code, en la carpeta src, modificar el archivo Program.cs con el siguiente contenido al inicio:

*using Shorten.Models;*

*using Microsoft.EntityFrameworkCore;*

*var builder = WebApplication.CreateBuilder(args);*

*// Add services to the container.*

*builder.Services.AddControllersWithViews();*

*builder.Services.AddDbContext<ShortContext>(options =>*

*options.UseInMemoryDatabase("ShortDB"));*

*builder.Services.AddQuickGridEntityFrameworkAdapter();*

*var app = builder.Build();*

*...*

1. En el terminal, ejecutar el siguiente comando para crear un nuevo controlador y sus vistas asociadas.

*dotnet aspnet-codegenerator Controller -name UrlMappingController -m UrlMapping -dc ShortContext -outDir Controllers -udl*

1. En el Visual Studio Code, en la carpeta src, modificar el archivo \_Layout.cshtml, Adicionando la siguiente opciòn dentro del navegador:

*...*

*<li class="nav-item">*

*<a class="nav-link text-dark" asp-area="" asp-controller="UrlMapping" asp-action="Index">Shortener</a>*

*</li>*

*...*

### PARTE III: DESPLIEGUE DE LA APLICACION

1. En el terminal, ejecutar el siguiente comando para obtener el perfil público (Publish Profile) de la aplicación. Anotarlo porque se utilizará posteriormente.

*az webapp deployment list-publishing-profiles --name upt-awa-XXX --resource-group upt-arg-XXX –xml*

Nota: Donde XXX; es el número de identificación de la Aplicación Web creada en la primera sección

1. Abrir un navegador de internet y dirigirse a su repositorio en Github, en la sección Settings, buscar la opción Secrets and Variables y seleccionar la opción Actions. Dentro de esta hacer click en el botón New Repository Secret
2. En el navegador, dentro de la ventana New Secret, colocar como nombre AZURE\_WEBAPP\_PUBLISH\_PROFILE y como valor el obtenido en el paso 5.
3. En el Visual Studio Code, dentro de la carpeta .github/workflows, crear el archivo ci-cd.yml con el siguiente contenido

*name: Construcción y despliegue de una aplicación ASP.NET MVC a Azure*

*env:*

*AZURE\_WEBAPP\_NAME: upt-awa-XXX # Aqui va el nombre de su aplicación*

*AZURE\_WEBAPP\_PACKAGE\_PATH: '.' # Es la ruta de destino*

*DOTNET\_VERSION: '8' # la versión de .NET*

*on:*

*push:*

*branches: [ "main" ]*

*workflow\_dispatch:*

*permissions:*

*contents: read*

*jobs:*

*build:*

*runs-on: ubuntu-latest*

*steps:*

*- uses: actions/checkout@v4*

*- name: Configurando .NET Core*

*uses: actions/setup-dotnet@v4*

*with:*

*dotnet-version: ${{ env.DOTNET\_VERSION }}*

*- uses: actions/setup-java@v4*

*with:*

*distribution: 'temurin'*

*java-version: '17'*

*- name: Restaurar los paquetes*

*run: cd src && dotnet restore*

*- name: Ejecutar pruebas*

*run: cd src && dotnet test --collect:"XPlat Code Coverage"*

*- name: ReportGenerator*

*uses: danielpalme/ReportGenerator-GitHub-Action@5.3.7*

*with:*

*reports: ./src/\*/\*/\*/coverage.cobertura.xml*

*targetdir: coveragereport*

*reporttypes: MarkdownSummary;MarkdownAssembliesSummary;MarkdownSummaryGithub*

*- name: Upload coverage report artifact*

*uses: actions/upload-artifact@v4*

*with:*

*name: CoverageReport*

*path: coveragereport*

*- name: Publish coverage in build summary #*

*run: cat coveragereport/SummaryGithub.md >> $GITHUB\_STEP\_SUMMARY*

*shell: bash*

*- name: Instalar Scanner*

*run: dotnet tool install -g dotnet-sonarscanner*

*- name: Ejecutar escaneo*

*run: |*

*cd src*

*dotnet-sonarscanner begin /k:"${{ env.SONAR\_PROJECT }}" /o:"${{ env.SONAR\_ORG }}" /d:sonar.login="${{ secrets.SONAR\_TOKEN }}" /d:sonar.host.url="https://sonarcloud.io"*

*dotnet build*

*dotnet-sonarscanner end /d:sonar.login="${{ secrets.SONAR\_TOKEN }}"*

*- name: Publicar la aplicación de manera local*

*run: cd src && dotnet publish -c Release -o ${{env.DOTNET\_ROOT}}/publish*

*- name: Subir el artefacto para el job de despliegue*

*uses: actions/upload-artifact@v4*

*with:*

*name: .net-app*

*path: ${{env.DOTNET\_ROOT}}/publish*

*deploy:*

*permissions:*

*contents: none*

*runs-on: ubuntu-latest*

*needs: build*

*environment:*

*name: 'Development'*

*url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}*

*steps:*

*- name: Descargar el artefacto desde el job de construccion*

*uses: actions/download-artifact@v4*

*with:*

*name: .net-app*

*- name: Desplegar a Azure Web App*

*id: deploy-to-webapp*

*uses: azure/webapps-deploy@v2*

*with:*

*app-name: ${{ env.AZURE\_WEBAPP\_NAME }}*

*publish-profile: ${{ secrets.AZURE\_WEBAPP\_PUBLISH\_PROFILE }}*

*package: ${{ env.AZURE\_WEBAPP\_PACKAGE\_PATH }}*

*restart: true*

1. En el Visual Studio Code o en el Terminal, confirmar los cambios con sistema de controlde versiones (git add ... git commit...) y luego subir esos cambios al repositorio remoto (git push ...).
2. En el Navegador de internet, dirigirse al repositorio de Github y revisar la seccion Actions, verificar que se está ejecutando correctamente el Workflow image
3. En el Navegador de internet, una vez finalizada la automatización, ingresar al sitio creado y navegar por el (https://upt-awa-XXX.azurewebsites.net).
4. En el Terminal, revisar las metricas de navegacion con el siguiente comando.

*az monitor metrics list --resource "/subscriptions/XXXXXXXXXXXXXXX/resourceGroups/upt-arg-XXX/providers/Microsoft.Web/sites/upt-awa-XXXX" --metric "Requests" --start-time 2025-01-07T18:00:00Z --end-time 2025-01-07T23:00:00Z --output table*

1. En el Terminal, ejecutar el siguiente comando para obtener la plantilla de los recursos creados de azure en el grupo de recursos UPT.

*az group export -n upt-arg-XXX > lab\_01.json*

1. En el Visual Studio Code, instalar la extensión ARM Template Viewer, abrir el archivo lab\_01.json y hacer click en el icono de previsualizar ARM.

### ACTIVIDADES ENCARGADAS

1. Subir el diagrama al repositorio como lab\_01.png y el reporte de métricas.
2. En el archivo main.tf, implementar el recurso azurerm\_app\_service\_source\_control, para el despliegue automatizado de la aplicación
3. Construir pruebas de interfaz para completar el 100% de cobertura de aplicación