


PASO 2 : Configurar Azure app services

En este paso configuramos el **backend** que llamara a los servicios de OpenAI.

Este backend es una app Python con un api rest que consume el **frontEnd** Whatsapp.

Microsoft Azure

 Microsoft

salamancamilsabores@gmail.com

Escribir contraseña

Contraseña

[¿Olvidó su contraseña?](#)

Enviar código por correo electrónico a
salamancamilsabores@gmail.com

Iniciar sesión

Microsoft Azure Actualización azure app

>

Services

io predeterminado (salamancamilsaboresgmail.o

rear ▾ Administrar aplicaciones elimin

por cualquier ca... Suscripción es igual

ndo de 0 a 0 de 0 registros.

ombre ↑↓

Todo Servicios (99+) Mark

Grupos de recursos (0)

Servicios

- Azure Active Directory
- Cognitive Search
- App Services**
- Azure Cosmos DB

Microsoft Azure Actualización

> Inicio >

App Services

Directorio predeterminado (salamancamilsaboresgmail.onmic

+ Crear ▾ Administrar aplicaciones eliminadas

- + Aplicación web**
- + Aplicación web estática
- + Aplicación web + base de datos

Nombre ↑↓

Servicios (99+) Mark

Grupos de recursos (0)

Active Directory

e Search

Services

osmos DB

Importante: crear un grupo de recursos y la pila de ejecución Python 3.10

Inicio > App Services >

Crear aplicación web ...

Detalles del proyecto

Seleccione una suscripción para administrar los recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción * ⓘ

Azure subscription 1

Grupo de recursos * ⓘ

(Nuevo) Grupo de recursos

Crear nuevo

Detalles de instancia

¿Necesita una base de datos? [Pruebe la nueva](#)

Nombre *

Publicar *

Pila del entorno en tiempo de ejecución *

Python 3.10

Sistema operativo *

☒ Linux ☐ Windows

Región *

West Europe

¿No encuentra su plan de App Service? Pruebe otra región o seleccione su App Service Environment.

Un grupo de recursos es un contenedor que tiene los recursos relacionados de una solución de Azure.

Nombre *

nuevo_grupo_recursos

Aceptar Cancelar

Planes de precios

Revisar y crear

< Anterior

Siguiente: Implementación >

Escoger plan de precios F1, capa “free” para un PoC es mas que suficiente

Planes de precios

El plan de tarifa de App Service determina la ubicación, las características, los costos y los recursos del proceso asociados a la aplicación. [Más información](#)

Plan de Linux (West Europe) * ⓘ

(Nuevo) ASP-nuevogrupo-recursos-8c4a

Crear nuevo

Plan de precios

Gratis F1 (Infraestructura compartida)

Explorar planes de precios

Redundancia de zona

Un plan de App Service se puede implementar como un servicio con redundancia de zona en las regiones que lo admiten. Esta es una decisión que solo se toma en el momento de la implementación. Después de la implementación, no se podrá hacer que un plan de App Service tenga redundancia de zona. [Más información](#)

- Redundancia de zona
- ☐

Habilitada: Su plan de App Service y las aplicaciones que contiene tendrán redundancia de zona. El número mínimo de instancias del plan de App Service será tres.
- ☒

Deshabilitado: El plan de App Service y las aplicaciones que contiene no tendrán redundancia de zona. El número mínimo de instancias del plan de App Service será uno.

Resumen



Aplicación web
de Microsoft

SKU Gratis

Precio estimado - Gratis

Detalles

Suscripción	465a68cd-97ea-48df-b3ee-19f297e5f095
Grupo de recursos	nuevo_grupo_recursos
Nombre	chatbotbackend1
Publicar	Código
Pila del entorno en tiempo de ejecución	Python 3.10

Plan de App Service (nuevo)

Nombre	ASP-nuevogrupo-recursos-8c4a
Sistema operativo	Linux
Región	West Europe
SKU	Gratis
ACU	Infraestructura compartida
Memoria	1 GB de memoria

Crear

< Anterior

Siguiente >

[Descargar una plantilla para la automatización](#)



Microsoft.Web-WebApp-Portal-f69114c3-ab67 | Información general

Implementación

Buscar

Eliminar Cancelar Volver a implementar Descargar Actualizar

Información general

Entradas

Salidas

Plantilla

La implementación está en curso



Nombre de implementación: Microsoft.Web-WebApp-Portal-f6911... Hora de inicio: 24/7/2023, 2
Suscripción: [Azure subscription 1](#) Id. de correlación: 16e5ae51
Grupo de recursos: [nuevo_grupo_recursos](#)

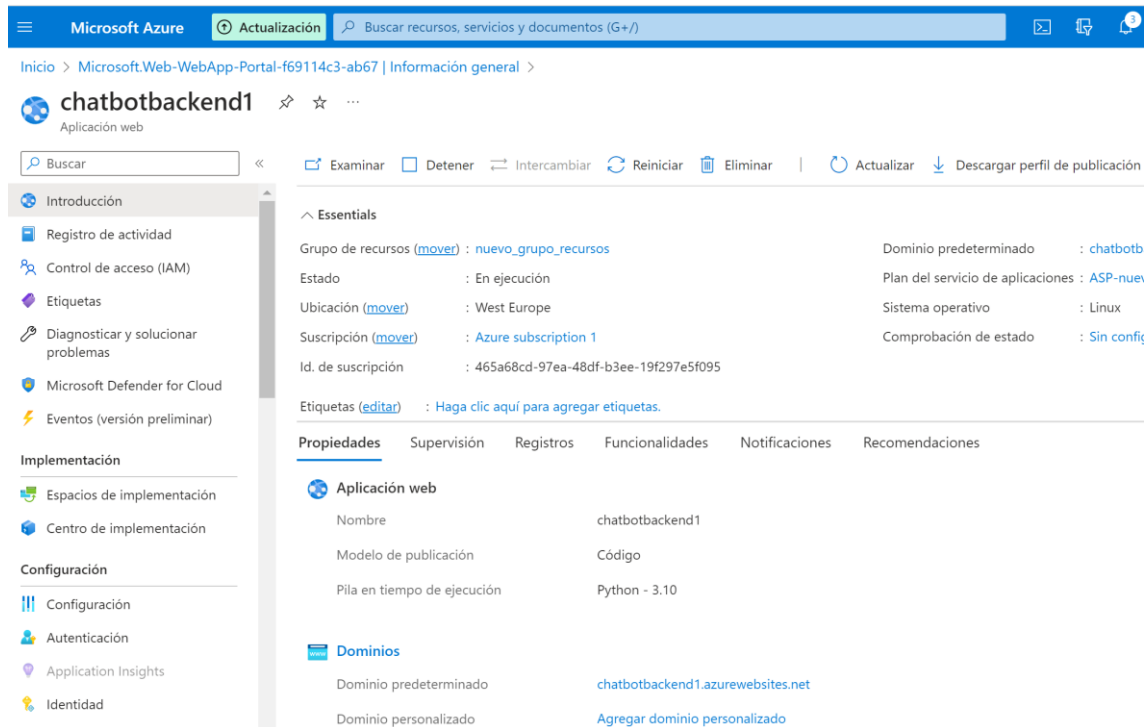
Detalles de implementación

Recurso	Tipo	Estado
No hay ningún resultado.		

Enviar comentarios

[Cuéntenos su experiencia con la implementación](#)

El App service Python en Azure esta creado!

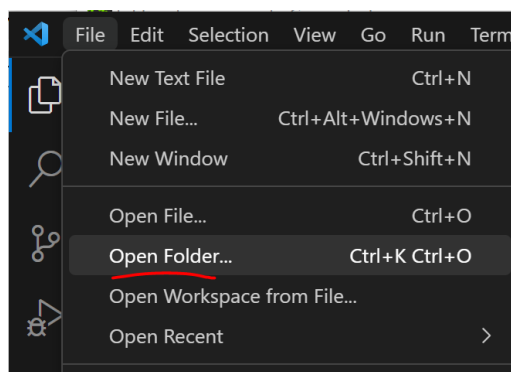


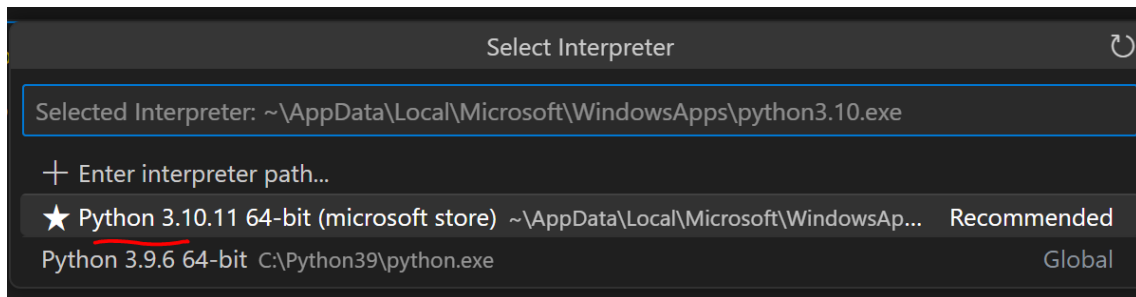
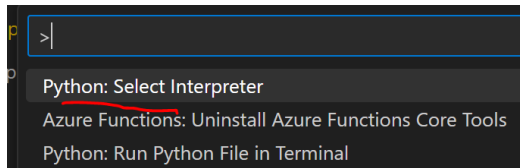
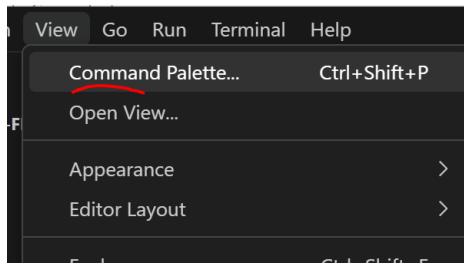
Para desplegar una app Python simple en Azure seguir la siguiente guía:

<https://learn.microsoft.com/en-us/azure/app-service/quickstart-python>

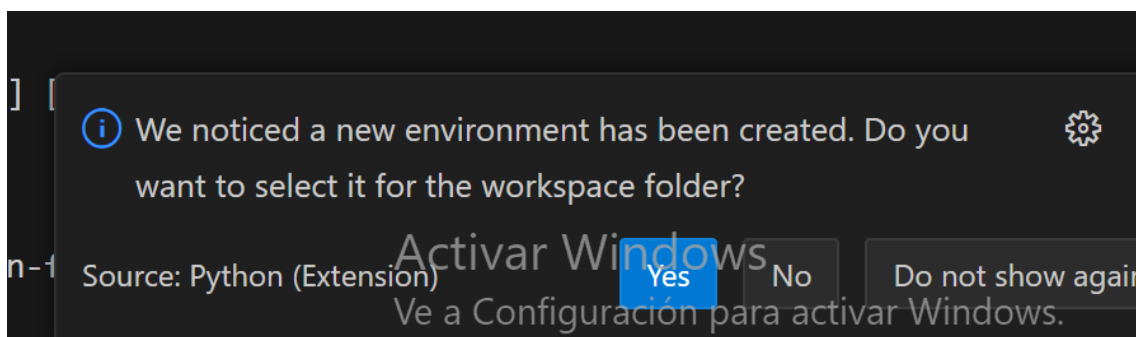
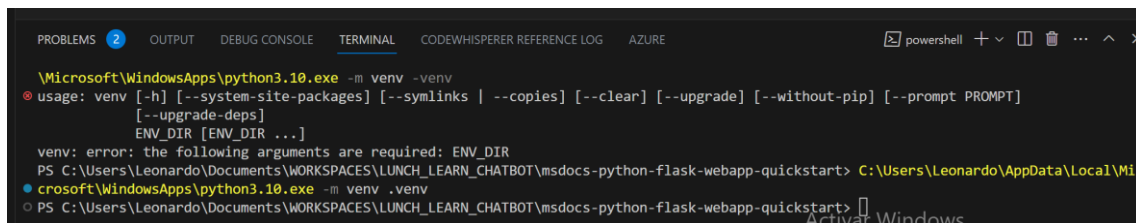
`git clone https://github.com/Azure-Samples/msdocs-python-flask-webapp-quickstart`

Una vez con el código (Template básico Flask) descargado abrir el proyecto con Visual Studio Code:



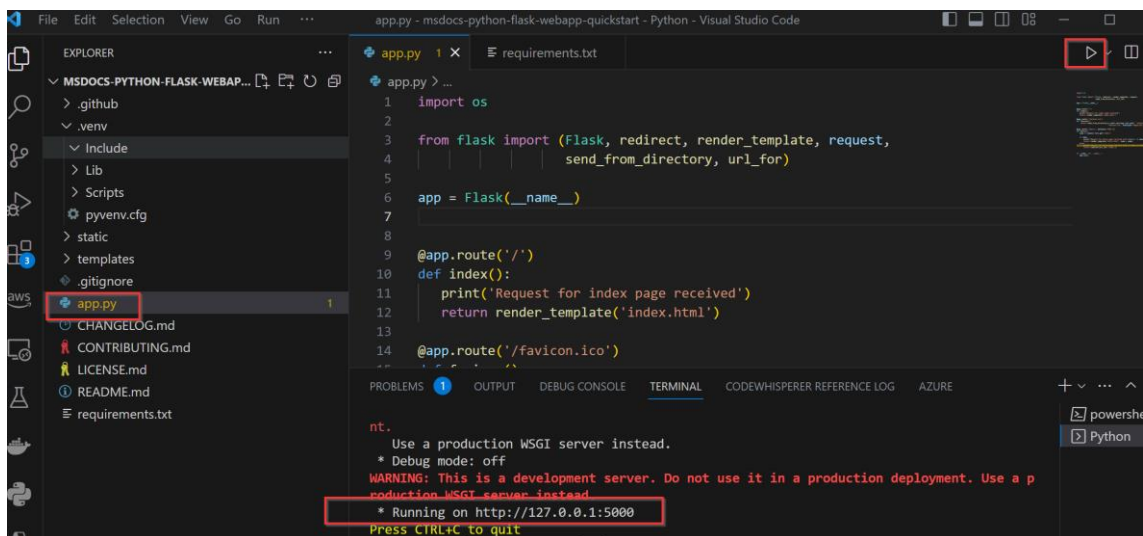


Importante crear el entorno virtual : `python3.10.exe -m venv .venv`



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG AZURE
PS C:\Users\Leonardo\Documents\WORKSPACES\LUNCH_LEARN_CHATBOT\msdocs-python-flask-webapp-quickstart> C:\Users\Leonardo\AppData\Local\Microsoft\WindowsApps\python3.10.exe -m venv .venv
PS C:\Users\Leonardo\Documents\WORKSPACES\LUNCH_LEARN_CHATBOT\msdocs-python-flask-webapp-quickstart> pip install -r requirements.txt
Collecting Flask==2.0.2
  Downloading Flask-2.0.2-py3-none-any.whl (95 kB)
    | 95 kB 546 kB/s
Collecting gunicorn
  Downloading gunicorn-21.2.0-py3-none-any.whl (80 kB)
    | 80 kB 2.2 MB/s
Collecting click>=7.1.2
  Downloading click-8.1.6-py3-none-any.whl (97 kB)
    | 97 kB 1.4 MB/s
Collecting Werkzeug>=2.0
  Downloading Werkzeug-2.3.6-py3-none-any.whl (242 kB)
    | 242 kB 6.8 MB/s
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Jinja2>=3.0
```

Para probar el local usar el botón de “play” sobre el fichero app.py



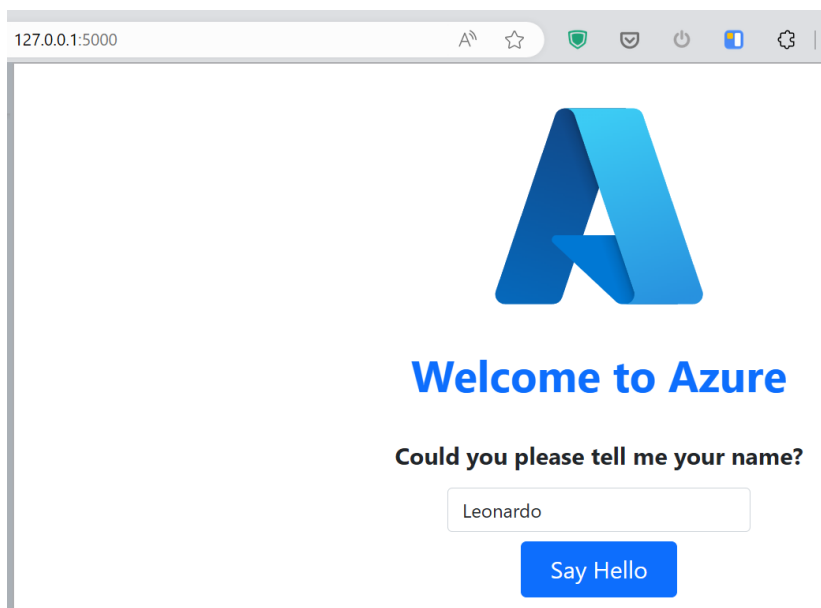
The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, the file `app.py` is highlighted. In the main editor, the `app.py` file is open, showing the following code:

```
1 import os
2
3 from flask import Flask, redirect, render_template, request,
4     send_from_directory, url_for
5
6 app = Flask(__name__)
7
8
9 @app.route('/')
10 def index():
11     print('Request for index page received')
12     return render_template('index.html')
13
14 @app.route('/favicon.ico')
```

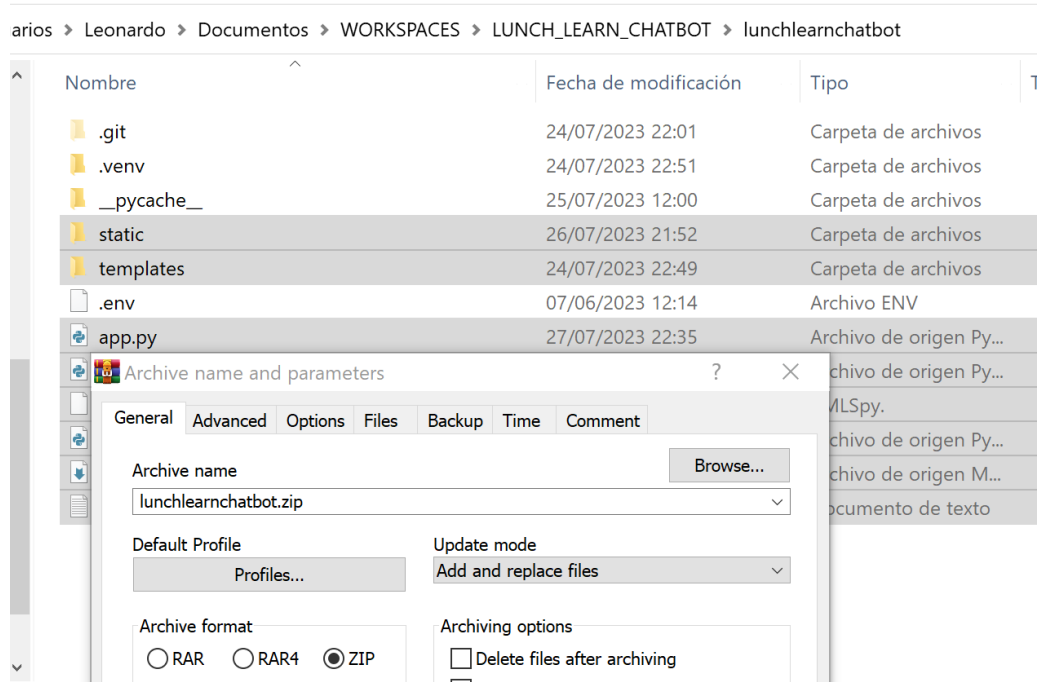
In the bottom right corner, the terminal output is visible:

```
nt.
Use a production WSGI server instead.
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a p
roduction WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

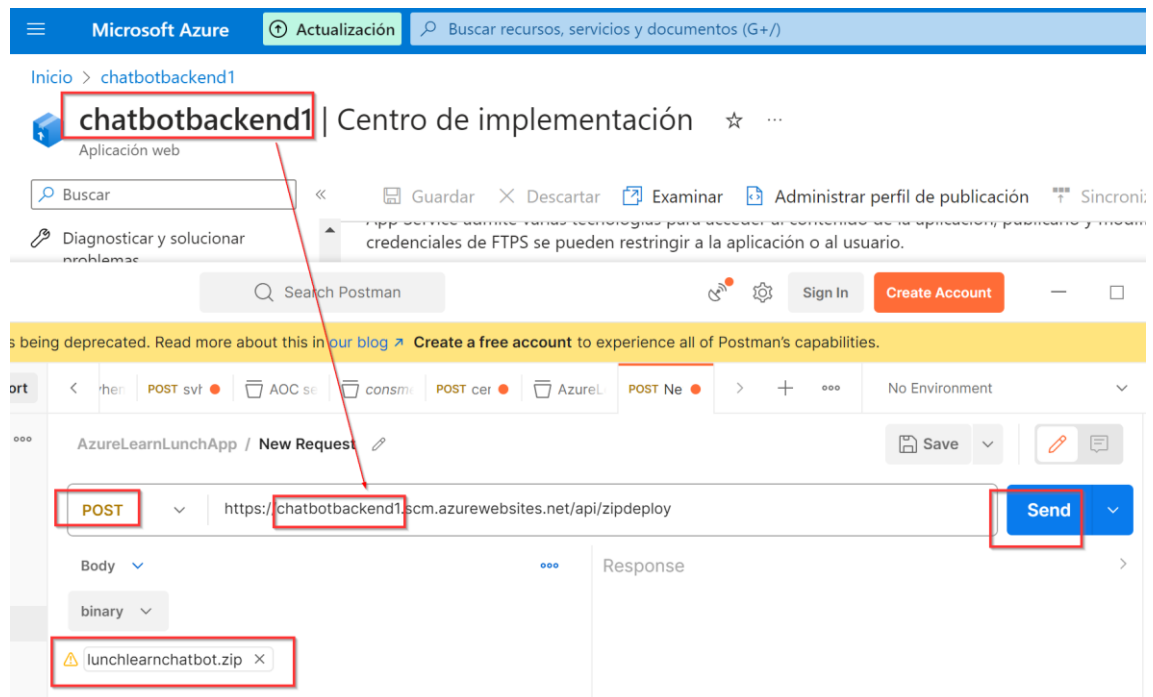
Prueba en local:



Para desplegar la app en Azure hay varias formas, una forma facil es hacer un zip (no incluir la carpeta .venv)



Enviar el zip en una llamada POST, usamos Postman



La autenticación es http basic y esta en la pestaña “centro de implementación”

The image shows two overlapping windows. The background window is the Microsoft Azure portal for the resource 'chatbotbackend1'. The left sidebar has a menu with 'Centro de implementación' highlighted. The main content area shows the 'Implementación' tab with fields for 'Punto de conexión de F...', 'Nombre de usuario de F...' (set to 'chatbotbackend1'), and 'Contraseña'. The foreground window is VS Code, showing the 'New Request' dialog for a POST request to 'https://chatbotbackend1.scm.azurewebsites.net/api/zipdeploy'. The 'Auth' tab is selected, and 'Basic Auth' is chosen. The 'Username' field is filled with 'chatbotbackend1' and the 'Password' field is filled with a long alphanumeric string. Red arrows point from the 'chatbotbackend1' text in the Azure portal to the corresponding fields in the VS Code dialog.

Y la tenemos desplegada!

