

Entregable – Taller Microservicios V4

Brayan Estiven Salcedo Suarez

David Alejandro Calderon Pineda

1. Despliegue del microservicio Places.

```
is deprecated. A suggested replacement is 'projects/ubuntu-os-cloud/global/images/ubuntu-2004-local-v20230508'

NAME: msd-places-db
TYPE: compute.v1.instance
STATE: COMPLETED
ERRORS: []
INTENT:

NAME: msd-places-ms
TYPE: compute.v1.instance
STATE: COMPLETED
ERRORS: []
INTENT:

NAME: msd-service-db
TYPE: compute.v1.firewall
STATE: COMPLETED
ERRORS: []
dav_calderonpi@cloudshell:~/ISIS2503-Microservices-AppDjango (isis2503-talleres-451901) $
```

Filter by name or value

<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect	
<input type="checkbox"/>	✓	msd-kong-instance	us-central1-a			10.128.0.81 (nic0)	34.44.164.128 (nic0)	SSH	⋮
<input type="checkbox"/>	✓	msd-measurements-db	us-central1-a			10.128.0.83 (nic0)		SSH	⋮
<input type="checkbox"/>	✓	msd-measurements-ms	us-central1-a			10.128.0.85 (nic0)	34.172.48.152 (nic0)	SSH	⋮
<input type="checkbox"/>	✓	msd-places-db	us-central1-a			10.128.0.86 (nic0)		SSH	⋮
<input type="checkbox"/>	✓	msd-places-ms	us-central1-a			10.128.0.87 (nic0)	104.198.233.225 (nic0)	SSH	⋮
<input type="checkbox"/>	✓	msd-variables-db	us-central1-a			10.128.0.82 (nic0)		SSH	⋮
<input type="checkbox"/>	✓	msd-variables-ms	us-central1-a			10.128.0.84 (nic0)	35.239.80.43 (nic0)	SSH	⋮

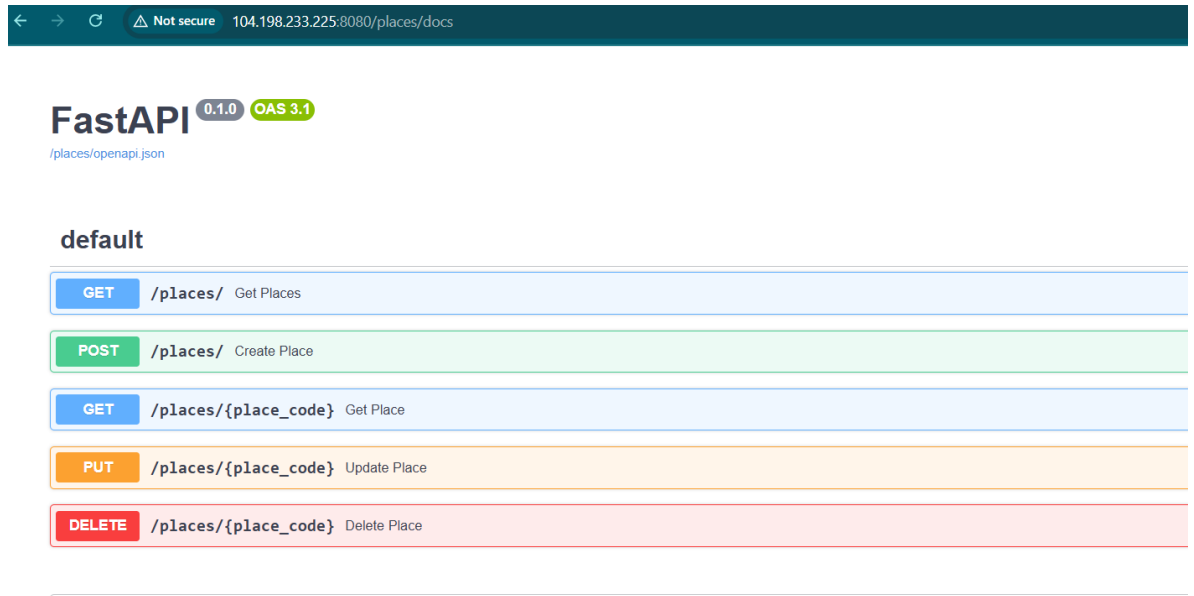
Related actions

Ejecución del microservicio:

```
dav_calderonpi@msd-places-ms:~$ cd /home/labs/ISIS2503-Microservices-AppDjango/places
dav_calderonpi@msd-places-ms:/home/labs/ISIS2503-Microservices-AppDjango/places$ sudo python3.12 main.py
/home/labs/ISIS2503-Microservices-AppDjango/places/main.py:14: DeprecationWarning:
    on_event is deprecated, use lifespan event handlers instead.

    Read more about it in the
    [FastAPI docs for Lifespan Events](https://fastapi.tiangolo.com/advanced/events/).

@app.on_event("startup")
INFO: Started server process [3343]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)
```



2. Modificación de configuración de Kong

```
- name: measurements_service
  host: measurements_upstream
  protocol: http
  routes:
    - name: measurements
      paths:
        - /measurements
      strip_path: false
```

```
- name: places_service
  host: places_upstream
  protocol: http
  routes:
    - name: places
      paths:
        - /places
      strip_path: true
```

```
upstreams:
  - name: variables_upstream
    targets:
      - target: 10.128.0.84:8080
        weight: 100

  - name: measurements_upstream
    targets:
      - target: 10.128.0.85:8080
        weight: 100

  - name: places_upstream
    targets:
      - target: 10.128.0.87:8080
        weight: 100
```

3. Modificación de microservicio Measurements

```
PLACES_HOST = os.environ.get('PLACES_HOST', '10.128.0.87')
PLACES_PORT = os.environ.get('PLACES_PORT', '8080')
PATH_PLACES = f"http://{PLACES_HOST}:{PLACES_PORT}/places/"

"monitoring/settings.py" 144L, 4079C
```

Se añade la función que verifica el lugar:

```
def check_place(data):
    r = requests.get(settings.PATH_PLACES, headers={"Accept": "application/json"})
    places = r.json()
    for place in places:
        if data["place"] == place["code"]:
            return True
    return False
```

se modifican las funciones de measurement para usar la función

```
def MeasurementCreate(request):
    if request.method == 'POST':
        data = request.body.decode('utf-8')
        data_json = json.loads(data)
        if check_variable(data_json) and check_place(data_json):
            measurement = Measurement()
            measurement.variable = data_json['variable']
            measurement.value = data_json['value']
            measurement.unit = data_json['unit']
            measurement.place = data_json['place']
            measurement.save()
            return HttpResponse("successfully created measurement")
        else:
            return HttpResponse("unsuccessfully created measurement. Variable or place does not exist")

def MeasurementsCreate(request):
    if request.method == 'POST':
        data = request.body.decode('utf-8')
        data_json = json.loads(data)
        measurement_list = []
        for measurement in data_json:
            if check_variable(measurement) and check_place(measurement):
                db_measurement = Measurement()
                db_measurement.variable = measurement['variable']
                db_measurement.value = measurement['value']
                db_measurement.unit = measurement['unit']
                db_measurement.place = measurement['place']
                measurement_list.append(db_measurement)
            else:
                return HttpResponse("unsuccessfully created measurement. Variable or place does not exist")

Measurement.objects.bulk_create(measurement_list)
```

4. Creación de Lugares de Prueba

INGRESO A : <http://104.198.233.225:8080/places/docs>

Parameters

Cancel

No parameters

Request body required

application/json

```
{
  "capacity": 80,
  "code": "ML213",
  "type": "classroom"
}
```

Execute

Clear

Parameters

No parameters

Request body required

```
{
  "capacity": 80,
  "code": "ML213",
  "type": "classroom"
}
```

Execute

Responses

Responses

Curl

```
curl -X 'POST' \
  'http://104.198.233.225:8080/places/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "capacity": 80,
    "code": "ML213",
    "type": "classroom"
  }'
```

Request URL

http://104.198.233.225:8080/places/

Server response

Code Details

201

Response body

```
{
  "code": "ML213",
  "capacity": 80,
  "type": "classroom",
  "id": "682c1864a8e964c8d5abb05"
}
```

Response headers

```
content-length: 81
content-type: application/json
date: Tue, 20 May 2025 05:51:31 GMT
server: uvicorn
```

Responses

5. Creación de *Cloud Function* para datos de oxígeno

Se desarrolló una función en Google Cloud Functions en Python.

Esta función se conecta a una API externa para obtener datos de oxígeno.

Luego transforma esos datos en el formato esperado por el microservicio measurements.

Deploying revision [Hide status](#)

Building source (see [logs](#)) ✓ Completed

Updating service ✓ Completed

Creating revision ✓ Completed

Routing traffic ✓ Completed

oxygen-consumption Region: us-central1 URL: <https://oxygen-consumption-1008673593258.us-central1.run.app> Scaling: Auto (Min: 0)

Metrics SLOs Logs Revisions **Source** Triggers Networking Security YAML

Source Base image: Python 3.9 (Ubuntu 18 Full) Function entry point: hello_http [Edit source](#)

main.py requirements.txt

```
1 import requests
2 import json
3 import os
4
5 def hello_http(request):
6     data = requests.get(os.environ.get('API_PATH'), headers={"Accept": "application/json"})
7     json_data = data.json()
8     response = requests.post(os.environ.get('MS_PATH'), json=json_data, headers={"Content-type": "application/json", "charset": "utf-8"})
9     return "The function was successfully executed - Oxygen data imported"
10
```

Creación del *Cloud Scheduler* para ejecutar la funcion:

Filter Filter jobs

<input type="checkbox"/>	Name ↑	Status of last execution	Region	State	Description	Frequency	Target	Last run	Next run	Last updated
<input type="checkbox"/>	api-consumption-scheduler	Success	us-central1	Enabled		*/* * * * *	URL: https://api-consumption-1008673593258.us-central1.run.app/	May 20, 2025, 1:25:00 AM	May 20, 2025, 1:30:00 AM	May 19, 2025, 10:14:58 PM
<input checked="" type="checkbox"/>	oxygen-consumption-scheduler	Has not run yet	us-central1	Enabled		*/* * * * *	URL: https://oxygen-consumption-1008673593258.us-central1.run.app/		May 20, 2025, 1:30:00 AM	May 20, 2025, 1:27:48 AM

Análisis del resultado del experimento

A partir del experimento realizado, se puede concluir que la arquitectura implementada benefició directamente al Atributo de Calidad (ASR) planteado, el cual exige que, una vez desplegado el sistema, la integración de un nuevo servicio no tome más de dos horas. Este objetivo se cumplió gracias al enfoque de microservicios adoptado, el cual permite desarrollar y desplegar servicios de manera independiente, reduciendo significativamente los tiempos de integración. La incorporación del microservicio *Places* y su vinculación con servicios existentes como *Measurements* y la *Cloud Function* se realizó sin requerir modificaciones sustanciales en el sistema base, lo que evidencia el nivel de desacoplamiento alcanzado. Asimismo, el uso de tecnologías como Docker para contenedorización, Kong como API Gateway, y servicios gestionados como Cloud Run y Cloud Scheduler facilitaron una orquestación eficiente y ágil de los componentes. Esta infraestructura estandarizada no solo permitió una integración rápida, sino también garantizó la estabilidad del sistema en su conjunto. En resumen, la arquitectura no solo respondió al requerimiento funcional del ASR, sino que además demostró su capacidad para escalar y adaptarse a nuevas necesidades sin comprometer tiempos ni calidad del servicio.

Créditos:

A continuación, se observan los créditos disponibles en la cuenta al finalizar el laboratorio:

Créditos

Todos los créditos

Consulta y descarga los detalles de crédito aquí. Los descuentos por compromiso de uso activos no están incluidos aquí y se pueden ver en la [página Compromisos](#).

Filtro Filtrar créditos								
Nombre del crédito	Estado ↑	Porcentaje restante	Valor restante	Valor original	Tipo	ID de crédito	Permiso ⓘ	
Free Trial	⚠ Vence en 8 días	<div><div></div></div> 92%	\$1,167,087.43	\$1,262,802.70	De un solo uso	FreeTrialUpgrad...	Uso específico; consulta la	▼
Free Trial	❌ Vencido	—	\$1,262,741.14	\$1,263,338.00	De un solo uso	FreeTrial:Credit-...	Uso específico; consulta la	▼