

REDES Y COMUNICACIONES

TRABAJO PRÁCTICO: MICROSERVICIO



Julián Perelda

Fecha: 10/11/2022

Para poder realizar una explicación del proyecto realizado primero tenemos que entender ciertos temas principales:

- **MICROSERVICIO:**

Los microservicios son un enfoque arquitectónico y organizativo para el desarrollo de software donde el software está compuesto por pequeños servicios independientes que se comunican a través de API bien definidas. Los propietarios de estos servicios son equipos pequeños e independientes.

Las arquitecturas de microservicios hacen que las aplicaciones sean más fáciles de escalar y más rápidas de desarrollar.

- **API GATEWAY:**

Una puerta de enlace de API es una herramienta de administración de API que existe entre un cliente y una colección de servicios de back-end. Una puerta de enlace API actúa como un proxy inverso para aceptar todas las llamadas de la interfaz de programación de aplicaciones (API), agregar los diversos servicios necesarios para cumplir con las solicitudes y luego devolver el resultado apropiado.

Herramientas que se utilizaron en el proyecto:

- **SMTPLIB** (Microservicio):

El Protocolo simple de transferencia de correo (SMTP en inglés) maneja el envío y el correo electrónico de enrutamiento entre los servidores de correo.

En Python, el módulo `smtplib` define un objeto de sesión de cliente SMTP que se puede usar para enviar correo a cualquier máquina de Internet con un dominio de escucha SMTP o ESMTP.

- **SYS** (Microservicio):

Este módulo provee acceso a algunas variables usadas o mantenidas por el intérprete y a funciones que interactúan fuertemente con el intérprete. Siempre está disponible.

- **OS** (Microservicio):

Este módulo provee una manera versátil de usar funcionalidades dependientes del sistema operativo.

- **WSGIREF** (Microservicio):

La Interfaz de Pasarela del Servidor Web, o *Web Server Gateway Interface* en inglés (WSGI), es una interfaz estándar entre el servidor web y aplicaciones web escritas en Python. Con una interfaz estándar es más sencillo usar una aplicación que soporte WSGI con diferentes servidores web.

- **Axios** (API):

Axios es una biblioteca de cliente HTTP que le permite realizar solicitudes a un endpoint determinado

- **Express** (API):

Express es un framework de backend Node.js minimalista y rápido, que proporciona

características y herramientas robustas para desarrollar aplicaciones de backend escalables. Ofrece el sistema de enrutamiento y características simplificadas para ampliar el framework con componentes y partes más potentes en función de los casos de uso de una aplicación.

- **Nodemon (API):**

Nodemon es una utilidad que monitorea los cambios en el código fuente que se está desarrollando y automáticamente re inicia el servidor. Es una herramienta muy útil para desarrollo de aplicaciones en nodejs.

- **MySQL2 (API):**

Permite realizar una conexión con las base de datos de mySQL.

- **Nodemailer (API):**

Nodemailer es un paquete de distribución de Node.js que podemos integrar a nuestro proyecto y nos permite enviar email a un servidor SMTP en formato texto o HTML.

Explicación del proyecto:

Se creo un microservicio en Python el cual esta conectada a la API que se realizo en el trabajo anterior, en la API se encuentra un endpoint con el metodo GET y en su interior se realiza un POST utilizando AXIOS hacia la URL del microservicio.

```
app.get('/microservicio', async (req, res) => {  
  const [rows] = await pool.query('SELECT * FROM users')  
  res.json(rows)  
  
  axios.post('http://localhost:8001/envioMail', {  
    mensaje: "Se obtuvieron con exito los registros de la tabla USERS",  
    data: rows  
  })  
})
```

En el microservicio conseguimos la información que se realizo dicho método hacia su URL. Allí recuperamos la ruta y validamos el nombre del endpoint utilizado. Si se cumplen las condiciones, se va a disparar una función en donde se realizara un envío de mail.

```

def hello_world_app(envIRON, start_response):
    # print(envIRON) # Obtengo entorno de variables que recupero de la request.
    res = os.path.join(path, envIRON["PATH_INFO"][1:]) # Se obtiene la ruta.
    status = '200 OK' # HTTP Status

    headers = [('Content-type', 'application/json; charset=utf-8')] # HTTP Headers
    start_response(status, headers) # Configuración de headers

    # Con la ruta recuperada validamos el nombre del endpoints que queremos que se ejecute.
    if ( res.rsplit('/', 1)[1] == "envioMail"):
        envioMail() # Función envío de mail

        response = {
            'mensaje': 'Envío de mail exitoso'
        }
    else:
        response = {
            'mensaje': 'Microservicio funcionando' # Valido que el microservicio funciona
        }

    return [ bytes(json.dumps(response), 'utf-8') ] # Devuelve respuesta del response.

with make_server('', 8001, hello_world_app) as httpd: # Server
    print("Serving on port 8001")
    path = sys.argv[1] if len(sys.argv) > 1 else os.getcwd() #Obtengo el path

    # Para que se mantenga corriendo.
    httpd.serve_forever()

```

Este correo se realiza desde la función “envioMail”, en la cual se ingresa una cuenta, se asigna el emisor, receptor y se ingresa el asunto y el cuerpo del mail. Esta función siempre se va a realizar cuando se dispare el método GET desde la API.

```

def envioMail(): # Función de envío de mail.
    mail_user= 'juliperelda@outlook.com'
    mail_pass = '*****'
    FROM = 'juliperelda@outlook.com'
    TO = "juliperelda@gmail.com" if type("juliperelda@gmail.com") is list else ["juliperelda@gmail.com"]
    SUBJECT = "Envío de mail - Prueba Microservicio"
    TEXT = "Esto es un mail de prueba, esperemos que haya llegado rey ;)"

    message = """\From: %s\nTo: %s\nSubject: %s\n\n%s
    """ % (FROM, "", ".join(TO), SUBJECT, TEXT)
    try:
        server = smtplib.SMTP("smtp.outlook.com", 587)
        server.ehlo()
        server.starttls()
        server.login(mail_user, mail_pass)
        server.sendmail(FROM, TO, message)
        server.close()
        print('!Correo enviado!')
    except:
        print('!Error al enviar correo!')

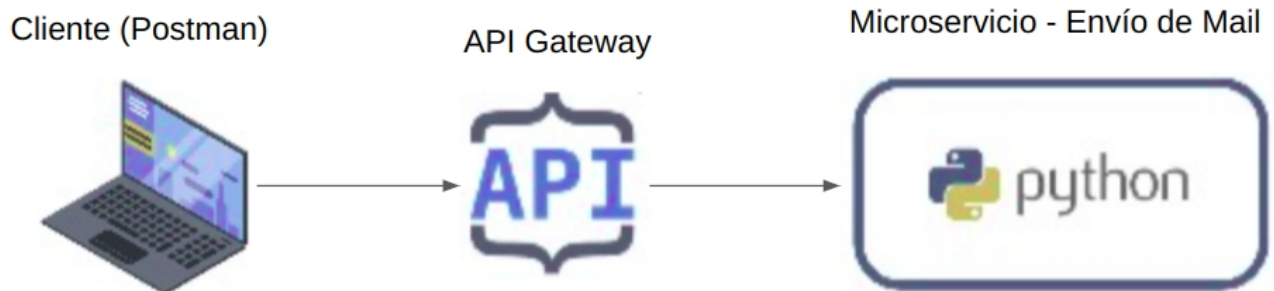
```

Problemas encontrados:

Los problemas que se plantearon fueron en gran parte en la utilización de un lenguaje en donde uno no tiene conocimientos. Ya desde ese punto se complica el querer realizar un código en donde cumpla su correcta funcionalidad.

Para solucionar este problema tuve que ir probando de a poco y obtener información de distintos lugares.

Diagrama de Red



Funcionamiento:

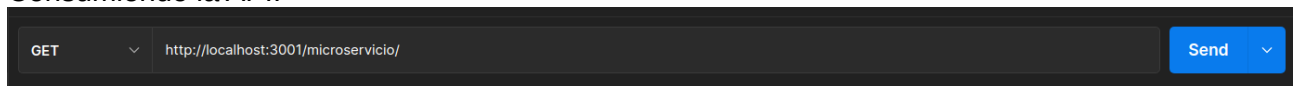
API en funcionamiento:

```
juliperelda@juliperelda-notebook:~/Escritorio/API - WEBHOOK/API - WEBHOOK/nodejs-mysql-railway$ yarn dev
yarn run v1.22.19
warning ../../../../package.json: No license field
$ nodemon src/app.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/app.js`
Server on port 3001
```

Microservicio en funcionamiento:

```
juliperelda@juliperelda-notebook:~/Escritorio/Microservicio-python$ python3 microservice.py
Serving on port 8001
```

Consumiendo la API:







Envío de correo:

```
juliperelda@juliperelda-notebook:~/Escritorio/Microservicio-python$ python3 microservice.py
Serving on port 8001
!Correo enviado!
127.0.0.1 - - [10/Nov/2022 20:33:15] "POST /envioMail HTTP/1.1" 200 36
```

Envio de mail - Prueba Microservicio ➤ Recibidos x



- **juliperelda@outlook.com**
Anda a full broderrrrr
- 18:40 (hace 1 hora) ☆
-
- **juliperelda@outlook.com**
Anda a full broderrrrr
- 18:55 (hace 1 hora) ☆
-
- **juliperelda@outlook.com**
Esto es un mail de prueba, esperemos que haya llegado rey :)
- 19:07 (hace 1 hora) ☆
-
- **juliperelda@outlook.com**
para mí ▾

- 20:33 (hace 1 minuto) ☆ ↶ ⋮
- Esto es un mail de prueba, esperemos que haya llegado rey :)

Repositorio GITHUB:
<https://github.com/juliperelda/Microservicio>