ESB

Xavier Jarro.

Febrero 2021.

Universidad Politécnica Salesiana
Ingeniería de Sistemas
Gerencia Informática

Abstract

En el desarrollo del presente trabajo de investigación se implementó una arquitectura SOA, donde se explicará cómo se implementaron varios servicios de distintos proveedores, también se detallará la forma en la cual se consumieron los servicios de los distintos proveedores con el fin dar una función global al proyecto.

Capítulo 1

Conceptos.

Concepto SOA

La Arquitectura Orientada a Servicios de cliente, conocida también como SOA por sus siglas en inglés, es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

La Arquitectura Orientada a Servicios, es la creación de sistemas de información ampliables, versátiles y flexibles que pueden ayudar a las organizaciones a impulsar el rendimiento y, al mismo tiempo, reducir costes de IT y mejorar la flexibilidad en los procesos del negocio. (krafzing, 2015)

Concepto ESB.

Un Enterprise Service Bus es una infraestructura de software que actúa como capa intermedia para comunicar los diferentes sistemas de una empresa, tanto sistemas internos de la misma como otros fuera del ámbito de ésta. Para llevar a cabo estas comunicaciones se basa principalmente en el uso de tecnologías web para enlazar los distintos puntos.

Capítulo 2

Elaboración de una arquitectura SOA.

Para la implementación de una arquitectura SOA me he planteado como problemática la venta online de peluches, partiendo de este problema se pudo implementar el servicio de búsqueda de usuarios por clave y la búsqueda de peluches con la obtención de la imagen del peluche. Se utilizó un cliente Python con el cual me permite conectarme de forma

directa con el ESB, el cual es MULESOFT, el ESB se conecta con los servicios de obtención de usuarios y la obtención de la imagen del peluche. A continuación, se mostrará la arquitectura que se empleó:

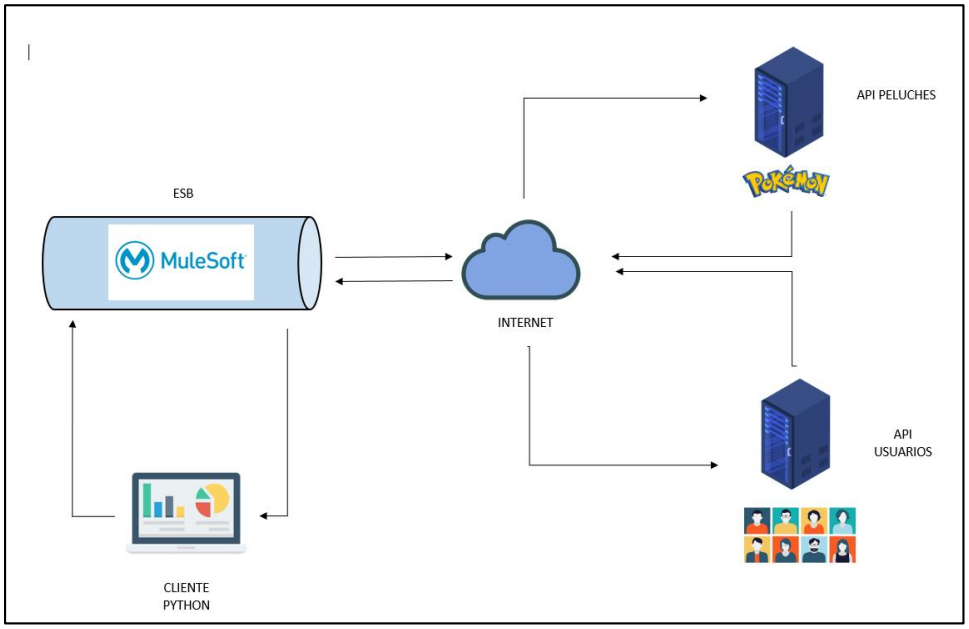

Ilustración 1. Arquitectura SOA

Funcionamiento de ESB.

El funcionamiento del ESB MuleSoft es intuitivo y fácil de poner en práctica, por lo que a continuación de describirá como funciona cada uno de los componentes de MuleSOft.

El siguiente diagrama muestra como usa un componente listener el cual sirve para poder escuchar las peticiones del cliente Python, el cual para como parámetros la operación en este caso el servicio que se quiere consumir.

Ilustración 2. Listener

El siguiente componente es un request el cual obtiene los parámetros que envía el cliente Python, luego estos parámetros son setedos en el componente llamado tipo el cual posterior nos ayudara a saber qué operación se quiere realizar.

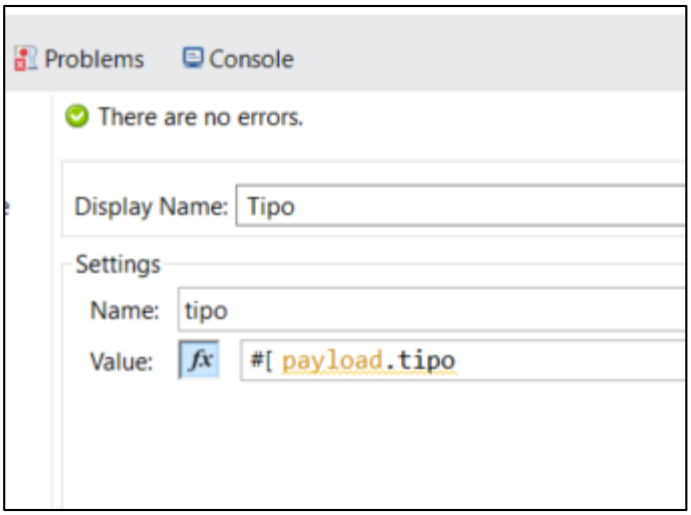

Ilustración 3. Variable tipo

El siguiente componente es un choice el cual actuaría como un if donde si se cumple la sentencia realiza una acción o caso contrario manda un error al cliente Python. En el primero componente al ser verdadero el caso del choice se llama a un componente flow reference, este lo que hace es llamar a otro método donde se realiza otras funciones en este caso lo que hace el obtener el usuario con sus datos y la imagen del peluche. En la siguiente ilustración se verá los métodos los cuales llama el componente flow reference.

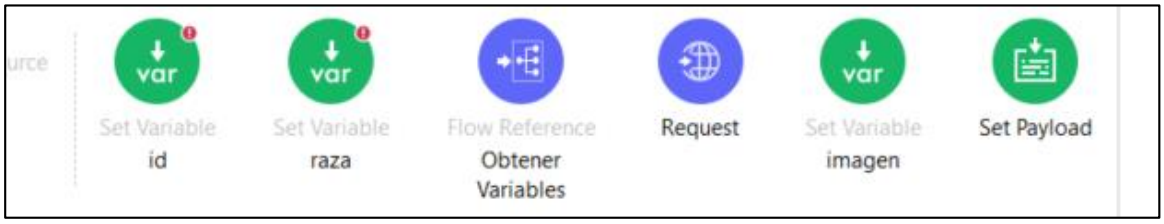

Ilustración 4.Obtencion de usuario e imagen

Como se ve en la imagen anterior se crean dos variables donde se setearan el id del usuario que queremos buscar y el nombre del peluche que se quiere obtener, una vez realizado dicho seteo lo siguiente será hacer un flow reference el cual llamara a otro método que retorna los datos del usuario. En la siguiente imagen se mostrará como está estructurado el método donde se llama a la API para buscar e obtener el usuario.

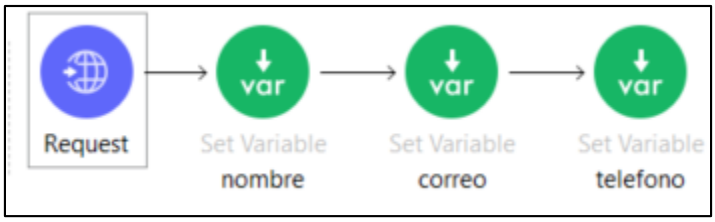

Ilustración 5.Método para obtener usuario

En el request se llama al api de obtención de usuarios donde se pasa como parámetro el id del usuario, se realizó la siguiente configuración en el request.

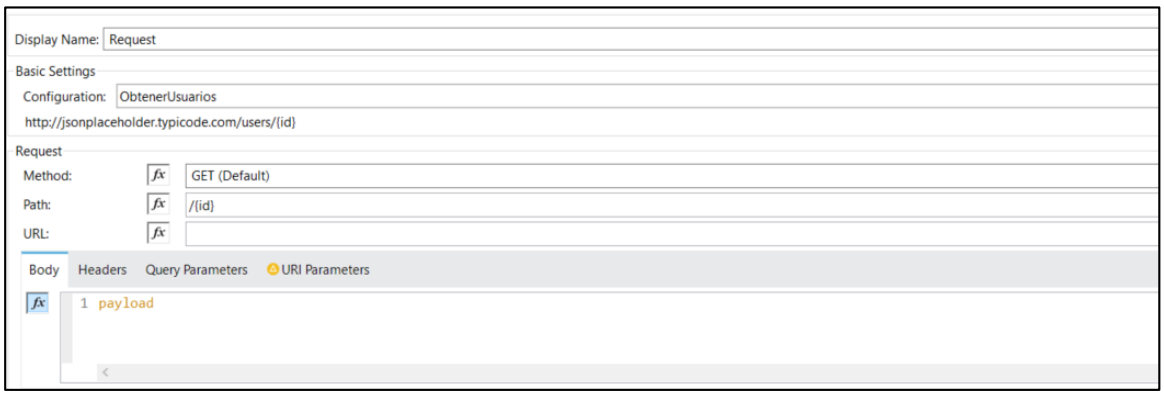

Ilustración 6Componente Request obtención de usuarios

Como se ve en la imagen anterior se obtiene un payload, el cual es la respuesta del api que se consumió para posterior almacenar los valores en las 3 variables nombre, correo y teléfono. Ya seteadas las variables se continua en el proceso del método anterior donde se hace un request al api de los peluches donde se obtendrá la imagen.

Ilustración 7Request Api imagen

Al igual que el proceso de los usuarios se setea la varibale imagen para almacenar el valor que tendrá el peluche, al final pasamos todos los valores de los usuarios e imágenes a un set payload el cual nos ayudara a retornar los valores que buscamos en las apis.

Ilustración 8. Payload

Capítulo 3

Resultados.

Ya realizado los procesos de consumo de servicios con el mediador ESB, lo siguiente será ver como se consume los servicios en la parte del cliente Python. El cliente Python tiene los siguientes métodos los cuales pasa como parámetros los valores del id del usuario y el nombre del peluche.

```python
@app.route('/obtener', methods=['POST'])
def sendDatos():
    if request.method == 'POST':
        id_usr = request.form['id_usr']
        raza = request.form['raza']
        dat = {'tipo': 1, 'id_usr': id_usr, "raza": raza}
        DAT.update(dat)
        url = "http://localhost:8081"
        r = requests.get(url=url, params=dat)
        js = r.json()
        print(js)
        if js['estado']:
            return render_template("index.html", datos=js)
    return render_template("error.html")
```

Ilustración 9.Consumo cliente Python

Como resultado del consumo se obtiene en la interfaz web hecha con flask el nombre, correo, teléfono del usuario y también la imagen del peluche.

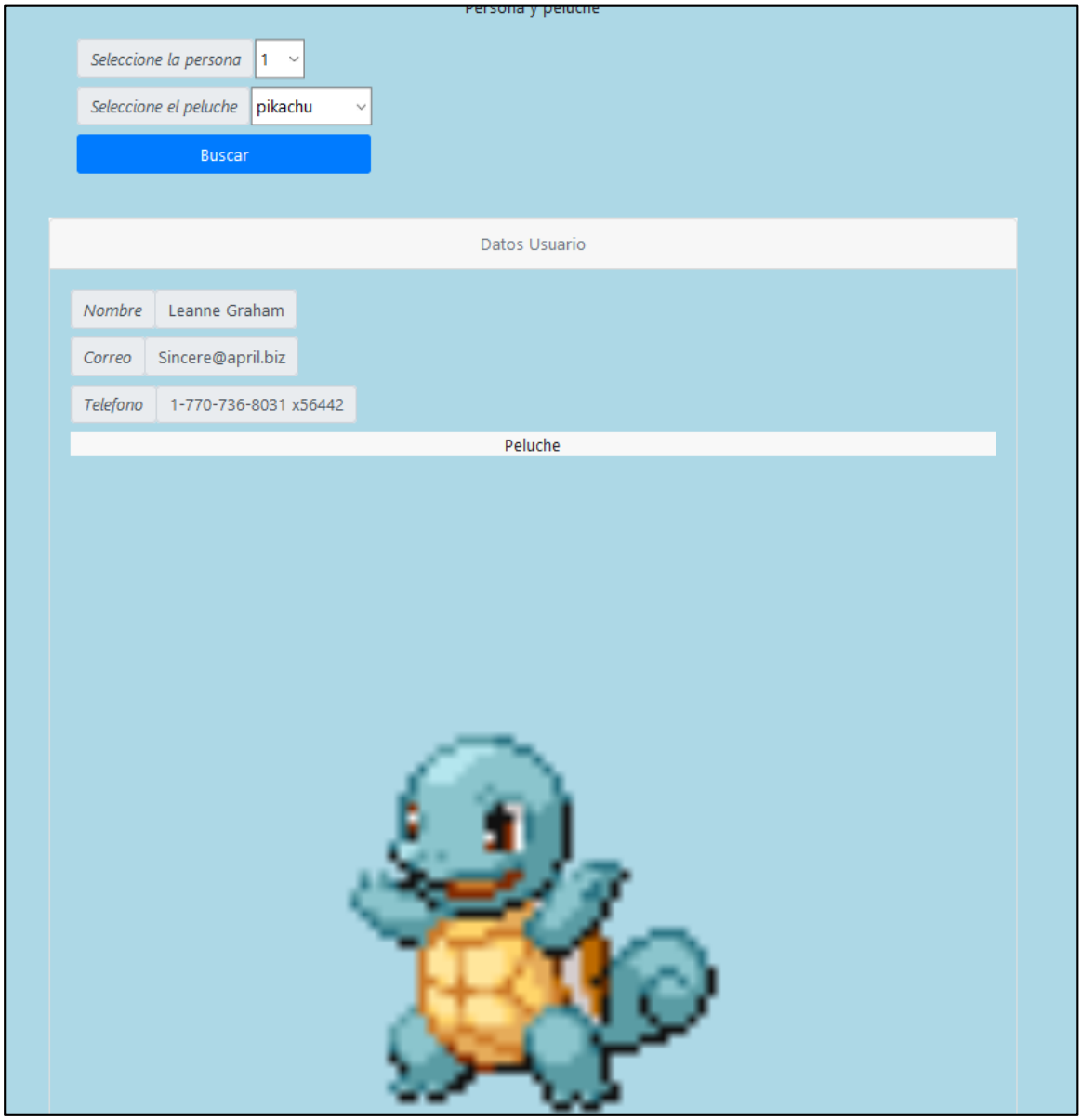

Ilustración 10.Resultado consumo

Lista de referencias

krafzing, D. (2015). *Enterprise SOA.* Maruland: PTR.