

Evaluación Conjunta

Universidad de las Fuerzas Armadas ESPE

Ingeniería en Tecnologías de la Información

Arquitectura de Software

Roberto Carlos Jacome Hidalgo

Objetivo

El objetivo principal fue crear tres microservicios independientes:

- **Central:** encargado de registrar las cosechas y enviar notificaciones.
- **Inventario:** ajusta las existencias de insumos de acuerdo a cada cosecha registrada.
- **Facturación:** calcula el valor de la factura basándose en precios predefinidos.

La interacción entre estos módulos debía realizarse mediante el intercambio de mensajes en colas de RabbitMQ.

Desarrollo

Microservicio Central

Registrar nuevas cosechas y enviar la información a las colas

```
JS index.js Central X package.json JS index.js facturacion JS index.js inventario
Central > JS index.js > connectRabbit
1 const express = require('express');
2 const amqp = require('amqplib');
3 const { v4: uuidv4 } = require('uuid');
4
5 const app = express();
6 app.use(express.json());
7
8 let cosechas = [];
9
10 async function connectRabbit() {
11   const conn = await amqp.connect('amqp://localhost');
12   const ch = await conn.createChannel();
13   await ch.assertQueue('cola_inventario');
14   await ch.assertQueue('cola_facturacion');
15   return ch;
16 }
17
18 let channel;
19 connectRabbit().then(ch => channel = ch);
20
21 app.post('/cosechas', (req, res) => {
22   const id = uuidv4();
23   const cosecha = { id, ...req.body, estado: 'REGISTRADA' };
24   cosechas.push(cosecha);
25
26   // Publicar eventos
27   channel.sendToQueue('cola_inventario', Buffer.from(JSON.stringify(cosecha)));
28   channel.sendToQueue('cola_facturacion', Buffer.from(JSON.stringify(cosecha)));
29
30   res.json({ mensaje: 'Cosecha registrada', cosecha });
31 });
32
33 app.get('/cosechas', (req, res) => {
34   res.json(cosechas);
35 });
36
37 app.listen(3000, () => console.log("Central corriendo en puerto 3000"));
```

Microservicio Inventario

Recibir el evento de una nueva cosecha y actualizar el stock de insumos.

```
JS indexjs Central  {} package.json  JS indexjs facturacion  JS indexjs inventario X
inventario > JS indexjs > ...
1  const amqp = require('amqplib');
2
3  let inventario = {
4    "Semilla Arroz L-23": 1000,
5    "Fertilizante N-PK": 500
6  };
7
8  async function start() {
9    const conn = await amqp.connect('amqp://localhost');
10   const ch = await conn.createChannel();
11   await ch.assertQueue('cola_inventario');
12
13   ch.consume('cola_inventario', msg => {
14     const cosecha = JSON.parse(msg.content.toString());
15     let semilla = cosecha.toneladas * 5;
16     let fertilizante = cosecha.toneladas * 2;
17
18     inventario["Semilla Arroz L-23"] -= semilla;
19     inventario["Fertilizante N-PK"] -= fertilizante;
20
21     console.log("Inventario actualizado:", inventario);
22     ch.ack(msg);
23   });
24 }
25
26 start();
27
```

3.3 Microservicio Facturación

Calcular el valor de la factura de una cosecha registrada.

```
JS indexjs Central  {} package.json  JS indexjs facturacion X  JS indexjs inventario
facturacion > JS indexjs > ...
1  const amqp = require('amqplib');
2
3  const precios = {
4    "Arroz Oro": 120,
5    "Café Premium": 300
6  };
7
8  async function start() {
9    const conn = await amqp.connect('amqp://localhost');
10   const ch = await conn.createChannel();
11   await ch.assertQueue('cola_facturacion');
12
13   ch.consume('cola_facturacion', msg => {
14     const cosecha = JSON.parse(msg.content.toString());
15     const precio = precios[cosecha.producto] || 100;
16     const total = cosecha.toneladas * precio;
17
18     console.log(`Factura generada para ${cosecha.producto}: ${total}`);
19     ch.ack(msg);
20   });
21 }
22
23 start();
24
```

Estado actual del proyecto

Código de los tres microservicios implementado.

Instalación de dependencias de Node.js realizada.

Ejecución de RabbitMQ fallida por errores en la instalación.

No se pudo hacer la prueba completa del flujo por la falta de conexión entre servicios.

Conclusiones

A pesar de no haber logrado la ejecución completa por problemas técnicos con RabbitMQ, el desarrollo permitió:

- Comprender la estructura básica de un sistema de microservicios.
- Implementar la lógica de cada módulo según su responsabilidad.
- Simular el flujo de datos usando memoria local, quedando pendiente la integración real con RabbitMQ.

Captura de las salidas de despliegues (simulada)

Microservicio Central (POST /cosechas)

```
Central > {} package.json > ...
1  {
2    "name": "central",
3    "version": "1.0.0",
4    "main": "index.js",
   ↳ Debug
5    "scripts": {
6      "test": "echo \"Error: no test specified\" && exit 1"
7    },
8    "keywords": [],
9    "author": "",
10   "license": "ISC",
11   "description": "",
12   "dependencies": {
13     "amqplib": "^0.10.8",
14     "express": "^5.1.0",
15     "uuid": "^11.1.0"
16   }
17 }
```

```
8  async function start() {
13    ch.consume('cola_facturacion', msg => {
15      const precio = precios[cosecha.producto] || 100;
16      const total = cosecha.toneladas * precio;
17
18      console.log("Factura generada para ${cosecha.producto}: ${total}");
19      ch.ack(msg);
20    });
21  }
22
23  start();
24
```

```
PS C:\Users\TUF GAMING\Desktop\proyecto_agrflow\facturacion> npm init -y
Wrote to C:\Users\TUF GAMING\Desktop\proyecto_agrflow\facturacion\package.json:

{
  "name": "facturacion",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

PS C:\Users\TUF GAMING\Desktop\proyecto_agrflow\facturacion> npm install amqp
added 5 packages, and audited 6 packages in 1s
```

```
8  async function start() {
13    ch.consume('cola_facturacion', msg => {
15      const precio = precios[cosecha.producto] || 100;
16      const total = cosecha.toneladas * precio;
17
18      console.log("Factura generada para ${cosecha.producto}: ${total}");
19      ch.ack(msg);
20    });
21  }
22
23  start();
24
```

```
PS C:\Users\TUF GAMING\Desktop\proyecto_agrflow\central> npm init -y
Wrote to C:\Users\TUF GAMING\Desktop\proyecto_agrflow\central\package.json:

{
  "name": "central",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

PS C:\Users\TUF GAMING\Desktop\proyecto_agrflow\central>
```

Conclusiones

A pesar de no haber logrado la ejecución completa por problemas técnicos con RabbitMQ, el desarrollo permitió:

- Comprender la estructura básica de un sistema de microservicios.
- Implementar la lógica de cada módulo según su responsabilidad.
- Simular el flujo de datos usando memoria local, quedando pendiente la integración real con RabbitMQ.