AA-004 – Documentación del Desarrollo de Microservicios e Integración mediante APIs

1. Resumen del Microservicio Creado

El proyecto Sistema-Reservas fue diseñado bajo una arquitectura modular, y como parte de su evolución hacia un modelo distribuido, se desarrolló el microservicio de notificaciones. Este componente tiene como propósito gestionar el envío de correos electrónicos automáticos a los usuarios cuando se genera, actualiza o confirma una reserva.

El microservicio se implementó en Node.js con Express.js y utiliza la librería Nodemailer para el envío de correos a través de servidores SMTP. Opera de manera independiente, pero se comunica con el backend principal mediante peticiones REST, recibiendo datos como el correo del usuario, asunto y mensaje.

Estructura General

```
microservice_notificaciones/
├── controllers/
│   └── mailController.js
├── services/
│   └── mailService.js
├── routes/
│   └── mailRoutes.js
├── .env
├── server.js
└── package.json
```

Relación con el Sistema Principal

El microservicio forma parte de la arquitectura general del sistema y se enlaza con el backend a través de una API. Cada vez que un usuario realiza una reserva en el sistema principal, este realiza una petición POST al endpoint /send del microservicio, el cual envía el correo de confirmación y devuelve una respuesta de éxito.

2. Integración mediante APIs

La integración entre el backend principal y el microservicio se basa en comunicación REST, lo que permite mantener un acoplamiento bajo y una interacción sencilla entre los servicios. El backend envía solicitudes HTTP al microservicio de notificaciones a través de un API Gateway interno o mediante la dirección local http://localhost:4000/send.

Ejemplo de Petición (Postman):
Método: POST
URL: http://localhost:4000/send
Body (JSON):

```
{
  'to': 'usuario@correo.com',
  'subject': 'Confirmación de reserva',
  'message': 'Su reserva ha sido registrada exitosamente.'
}
```

Respuesta Exitosa:

```
{
  'status': 'ok',
  'message': 'Correo enviado correctamente.'
}
```

3. Herramientas Colaborativas Utilizadas

Durante el desarrollo y documentación del microservicio se emplearon varias herramientas colaborativas para la coordinación del equipo y control del avance:

- **GitHub:** Control de versiones y repositorio principal del microservicio.
- **Discord:** Comunicación directa entre los integrantes del equipo.

4. Buenas Prácticas y Aprendizajes

Durante el desarrollo del microservicio se aplicaron varias buenas prácticas técnicas, entre las cuales destacan:

1. **Separación de responsabilidades:** Cada módulo cumple una función específica, lo que favorece la mantenibilidad y legibilidad del código.
2. **Uso de variables de entorno y manejo de errores:** Las credenciales SMTP se gestionaron mediante un archivo .env, y se implementó manejo de errores con respuestas uniformes en formato JSON.

Reflexión personal: La modularización del sistema a través de microservicios representa un avance significativo en escalabilidad y mantenimiento. En el caso del Sistema-Reservas, la creación del microservicio de notificaciones mejoró la organización del código y permitió una mayor eficiencia al gestionar el envío automático de correos.

5. Diagrama C4 (Container Nivel)

El Container Diagram representa cómo el microservicio de notificaciones se integra dentro del ecosistema general del Sistema-Reservas. El backend principal se comunica con el microservicio a través de una API REST, mientras que este interactúa con un servidor SMTP externo para enviar los correos.

Link del modelo C4:

https://drive.google.com/file/d/1PJyw28Fig74_Umka3ENZqVkuDDlWzEdm/view?usp=sharing

6. Conclusiones

El desarrollo del microservicio de notificaciones demostró cómo la arquitectura basada en microservicios permite aislar funciones críticas sin afectar el resto del sistema. La comunicación mediante API REST ofreció una integración simple y confiable, asegurando la interoperabilidad entre servicios. Las herramientas colaborativas utilizadas mejoraron la organización del equipo y el control del progreso técnico. Esta práctica reforzó la comprensión de conceptos como independencia de despliegue, desacoplamiento funcional y gestión modular de la lógica del sistema.

7. Bibliografía / Referencias

- Newman, S. (2021). Building Microservices. O'Reilly Media.
- Richardson, C. (2022). Microservices Patterns. Manning Publications.
- OWASP Foundation. (2023). OWASP Secure Microservices Design.
- Docker Inc. (2024). Docker Documentation.
- GitHub Docs. (2024). Collaborative Development Workflows.
- Node.js Foundation. (2024). Node.js API Documentation.