



# Guía de Actividades

## Práctico-Experimentales Nro. 004

### 1. Datos Generales

<b>Asignatura</b>	Desarrollo Basado en Plataformas
<b>Ciclo</b>	5 A
<b>Unidad</b>	1
<b>Resultado de aprendizaje de la unidad</b>	Discute cómo los estándares Web influyen en el desarrollo de software, bajo los principios de solidaridad, transparencia, responsabilidad y honestidad.
<b>Práctica Nro.</b>	004
<b>Título de la Práctica</b>	Despliegue de un microservicio independiente e integración básica con el servicio principal del proyecto.
<b>Nombre del Docente</b>	Edison Leonardo Coronel Romero
<b>Fecha</b>	Viernes 24 de octubre
<b>Horario</b>	07h30 – 10h30
<b>Lugar</b>	Aula 232
<b>Tiempo planificado en el Sílabo</b>	3 horas

### 2. Objetivo(s) de la Práctica:

Diseñar y desplegar un microservicio funcional que se comuniquen con el backend principal del proyecto.

Implementar comunicación REST entre servicios utilizando un API Gateway o endpoint compartido.

Documentar la arquitectura de microservicios en el modelo C4 y registrar evidencias del despliegue.

### 3. Materiales y reactivos:

- Computador con acceso a Internet.
- Docker / Docker Compose o entorno virtual local.
- Repositorio del proyecto (GitHub/GitLab).
- Postman / Swagger.
- Framework backend (Django REST Framework / Express / Spring Boot).

### 4. Equipos y herramientas

- Laboratorio de Desarrollo de Software o equipo personal.
- IDE: Visual Studio Code / IntelliJ IDEA.
- Git y GitKraken (flujo GitFlow).
- Terminal de comandos y contenedores Docker.

- Herramientas de modelado C4 (PlantUML, Mermaid, Draw.io).

## 5. Procedimiento / Metodología

### Inicio

- Contextualización de la práctica: introducción a microservicios, serverless y patrones de comunicación.
- Revisión del modelo C4 actual (backend monolítico) y planificación de su modularización.

### Desarrollo

#### 1. Diseño del microservicio

- Seleccionar una funcionalidad del proyecto que pueda desacoplarse (por ejemplo: módulo de notificaciones, gestión de equipos o usuarios).
- Definir su API interna y endpoints principales.

#### 2. Configuración del entorno

- Crear una nueva carpeta o repositorio `microservice_<nombre>` y configurarlo como módulo independiente.
- Implementar la estructura base con controladores, servicios y rutas.
- Configurar variables de entorno, dependencias y documentación básica (README).

#### 3. Comunicación entre servicios

- Configurar API Gateway o endpoint del backend principal para enlazar el nuevo servicio.
- Probar el intercambio de datos mediante peticiones HTTP.

#### 4. Despliegue y validación

- Ejecutar el microservicio en contenedor Docker o entorno local.
- Validar la conexión y registrar logs de comunicación.
- Documentar resultados en `/docs/architecture/container-diagram-v2.png`.

#### 5. Registro en modelo C4

- Actualizar los diagramas **Container** y **Component** incorporando el microservicio y el gateway.
- Registrar capturas o exportar los diagramas actualizados.

### Cierre (15 min)

- Socialización entre equipos: comparación de arquitecturas.
- Retroalimentación sobre ventajas y desafíos de microservicios.

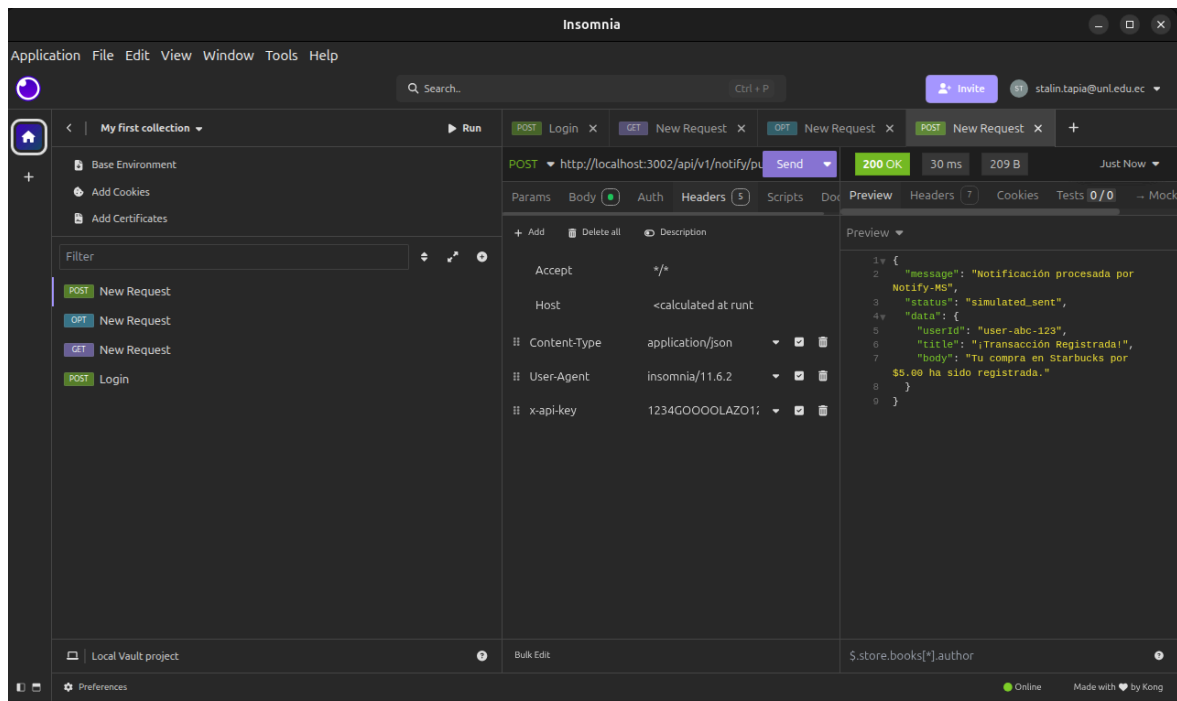
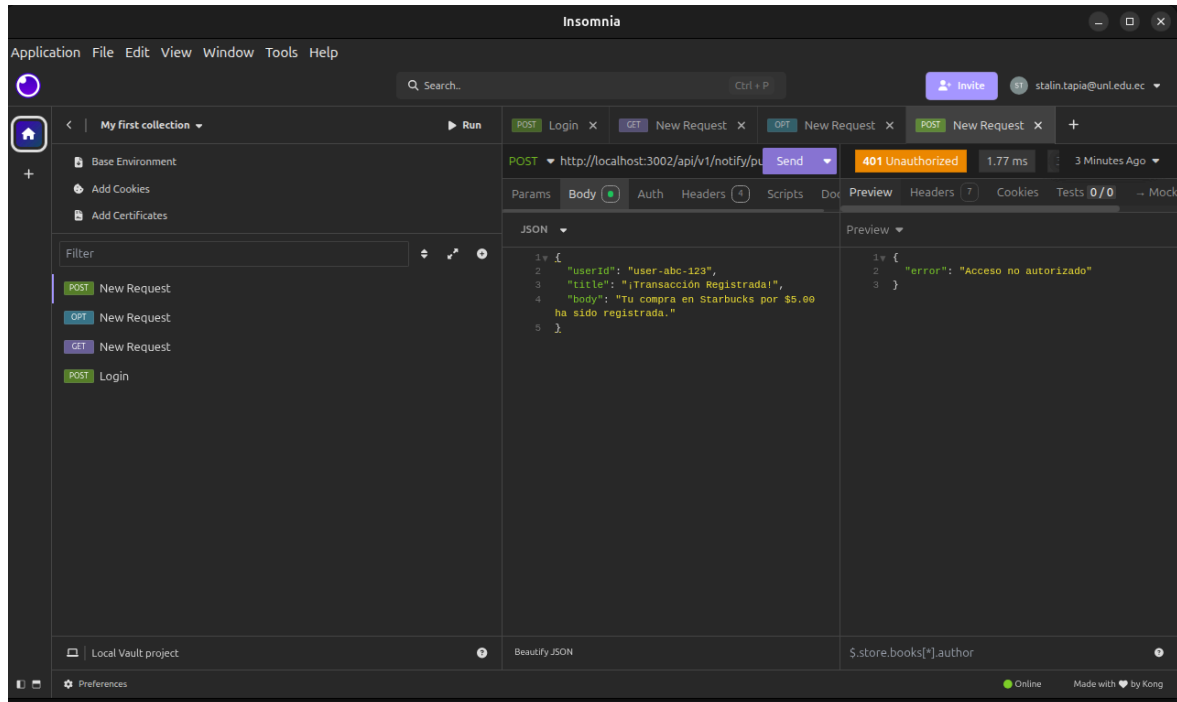


unl

Universidad  
Nacional  
de Loja

## 6. Resultados esperados:

- Microservicio desplegado y comunicándose correctamente con el backend principal.
- API Gateway o punto de integración funcional.
- Documentación técnica y diagramas C4 actualizados.
- Logs y capturas de comunicación.



## 7. Preguntas de Control:

- **¿Qué diferencia principal existe entre una arquitectura monolítica y una basada en microservicios?**

Diferencia principal: En una arquitectura monolítica todo el sistema está unido en una sola aplicación; en microservicios, se divide en servicios

- independientes que se comunican entre sí.
- **¿Qué beneficios aporta el uso de un API Gateway en la integración de servicios?**  
API Gateway: Centraliza el acceso, gestiona autenticación, balanceo de carga, enrutamiento y reduce la exposición directa de los servicios.
  - **¿Qué riesgos se presentan al no manejar correctamente la comunicación entre microservicios?**  
Riesgos: Fallos de comunicación, latencia, pérdida de mensajes o dependencias cíclicas que pueden causar caídas o inconsistencia de datos.
  - **¿Cómo se refleja la modularidad en los niveles Container y Component del modelo C4?**  
Modularidad en C4: En Container, cada microservicio es un contenedor independiente; en Component, cada contenedor se divide en módulos internos con responsabilidades claras.
  - **¿Qué consideraciones de seguridad se deben mantener al exponer endpoints entre servicios?**  
Seguridad: Usar autenticación y autorización (JWT/OAuth2), HTTPS, validación de datos, control de acceso por roles y políticas CORS seguras.

## Evaluación

Criterio	2 – Logro Alto	1 – Logro Medio	0 – Bajo / Sin Evidencia
<b>1. Diseño e implementación del microservicio</b>	Servicio funcional, correctamente desacoplado y con endpoints operativos.	Servicio funcional con errores menores o acoplamiento parcial.	Servicio no funcional o ausente.
<b>2. Integración con el backend principal</b>	Comunicación fluida entre servicios y validación en Postman / Swagger.	Comunicación parcial o errores en la respuesta.	No hay evidencia de integración.
<b>3. Documentación técnica y C4 actualizado</b>	Diagramas Container y Component actualizados y coherentes.	Diagramas incompletos o desactualizados.	Sin actualización del modelo C4.
<b>4. Despliegue y pruebas</b>	Despliegue funcional en Docker o entorno local con logs claros.	Despliegue parcial o sin validaciones suficientes.	No se logra el despliegue.

<b>5. Organización del repositorio y evidencias</b>	Estructura ordenada, README actualizado, capturas y registros claros.	Estructura parcial o documentación incompleta.	Sin documentación ni evidencias.
---	---	--	----------------------------------

## Bibliografía

- Newman, S. (2021). *Building Microservices*. O'Reilly Media.
- OWASP Foundation (2023). *OWASP Secure Microservices Design*.
- Docker Inc. *Docker Documentation*.
- Richardson, C. (2022). *Microservices Patterns: With examples in Java*. Manning Publications.


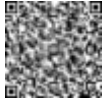


unl

Universidad  
Nacional  
de Loja

## Elaboración y Aprobación

- Elaborado por: nombre del Docente responsable de la práctica.
- Revisado por: nombre del Técnico de laboratorio (solo si aplica).
- Aprobado por: nombre del Director de Carrera.

<b>Elaborado por</b>	Edison L Coronel Romero <b>Docente</b>	 Firmado electrónicamente por: <b>EDISON LEONARDO CORONEL ROMERO</b> Validar únicamente con FirmaBC
<b>Aprobado por</b>	Edison L Coronel Romero <b>Director de Carrera</b>	 Firmado electrónicamente por: <b>EDISON LEONARDO CORONEL ROMERO</b> Validar únicamente con FirmaBC