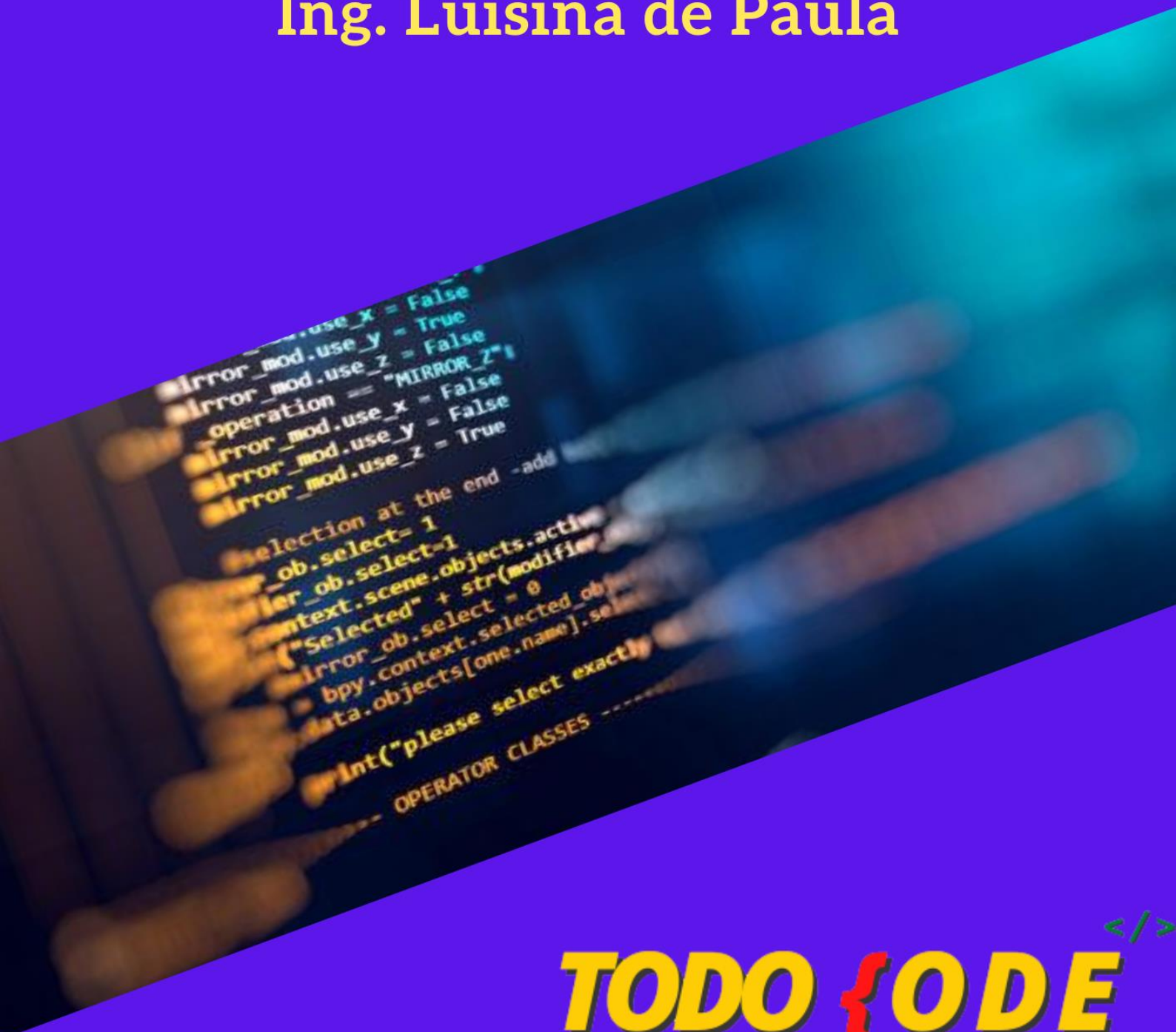
Microservicios con Spring Cloud

TP Integrador Final

Ing. Luisina de Paula



TODO CODE

www.todocodeacademy.com

TP Integrador Final

Objetivo

El objetivo de este proyecto integrador final es el de validar los **conocimientos prácticos y técnicos** referidos al desarrollo de **Microservicios** en el lenguaje de programación **Java** mediante **Spring Cloud** y otras herramientas, implementando diferentes patrones de diseño para el curso *"Microservicios con Spring Cloud"* de la TodoCode Academy.

Al mismo tiempo, este proyecto busca ser desarrollado para formar parte del **futuro portfolio del estudiante** para que sea una herramienta de presentación/demostración de conocimientos y habilidades ante la posibilidad de una **propuesta laboral**.

Escenario

Imaginate que estás trabajando en un proyecto de **una tienda de electrodomésticos online**. Para esta situación, el encargado de la tienda nos especificó como tarea desarrollar tres microservicios que se encargarán de diferentes aspectos del sistema; por supuesto, estos microservicios se comunicarán entre sí para proporcionar una experiencia de compra fluida a cada uno de los usuarios.

En base a las especificaciones técnicas relevadas por un Analista Funcional, los servicios a desarrollar son los siguientes:

- **Microservicio de Productos:** Se encargará de gestionar la información de los electrodomésticos disponibles en la plataforma. Debe ser capaz de listar los mismos y proporcionar detalles como ser código, nombre, marca y precio individual.

- **Microservicio de Carrito de Compras:** Este microservicio maneja, como dice su nombre, el carrito de compras de los usuarios. Éstos podrán, mediante este servicio, agregar y quitar artículos de la lista de electrodomésticos de su carrito de compras. Cada carrito tiene un número de identificación único (id) y un campo para almacenar el precio total de la sumatoria de todos los productos que se encuentren en el carrito.

- **Microservicio de Ventas:** Se encarga de registrar cada venta mediante un número de identificación y una fecha. Cada venta está asociada a un carrito de compras, por lo que al guardar una nueva venta se debe asignar, tal como se mencionó, un carrito a la misma. A través de esta asociación, la venta puede conocer el monto total de la misma (consultando al carrito de compras) y la lista de productos (consultando el servicio de carrito de compras que a su vez consume el servicio de productos).

Requerimientos

A partir del relevamiento realizado respecto, tener en cuenta los siguientes requerimientos:

1. Diagramar (dibujar/realizar un gráfico representativo de planteo) la **arquitectura de microservicios** presentada, considerando cómo se comunicarán estos servicios planteados entre sí.
2. Desarrollar cada uno de los servicios citados de manera individual para llevar a cabo los correspondientes CRUDS y operaciones necesarias. Tener en cuenta en cada servicio la correcta configuración de su archivo **application.properties**.
3. Configurar un **servidor de Eureka** para registrar cada uno de los microservicios. Asignar nombres representativos a cada uno de ellos
4. Implementar **balanceo de carga** mediante **Spring Cloud Load Balancer**.
    a. Crear 2 o más instancias de alguno de los microservicios a ser consultados en diferentes puertos (por ejemplo, producto que es consultado por carrito).
    b. Simular una serie de peticiones constantes mediante Postman para corroborar el correspondiente funcionamiento de Load Balancer.
5. Implementar **Circuit Breaker** mediante **Resilience4J** para asegurar de que, en caso que se produzca un error de comunicación entre los servicios, se ejecute un correspondiente fallbackmethod. Recordar de implementar de igual manera la posibilidad de REINTENTAR la comunicación entre servicios (**@Retry**).
6. Implementar una API Gateway que actúe como punto de entrada para clientes externos.
7. **Docker**
    a. Crear los correspondientes Dockerfiles de cada uno de los servicios.
    b. Llevar a cabo el correspondiente archivo docker-compose.yml
    c. Realizar el proceso de deploy de los servicios mencionados en Docker mediante docker-compose build y docker-compose up.
    d. Validar (mediante pruebas con Postman) que los servicios puedan comunicarse correctamente entre sí dentro del entorno creado con Docker.

8. **BONUS (OPCIONAL)**
    a. Llevar a cabo un repositorio local (Git) y remoto (GitHub) de los proyectos desarrollados.
    b. Implementar un Servidor Centralizado de Configuraciones (Config Server) a partir de dichos repositorios.
    c. Implementar cualquier funcionalidad o supuesto extra que consideres necesario o complementario.

Consideraciones

- Como se trata de un curso "**on demand**" (autogestionable), la idea de llevar a cabo esta actividad integradora es la de poner a prueba todas las nuevas habilidades que adquiriste a lo largo de este curso.
- Podés llevar a cabo esta actividad, sin embargo, la misma no será corregida ni tendrá asesoramiento o devolución personalizada por parte de ningún facilitador.
- Ante cualquier duda o consulta al respecto, recordá que contás con el Servidor de Discord de la comunidad de TodoCode, donde se encuentra el canal **#Ayuda-Consultas** destinado exclusivamente para esto.

Dicho todo esto... ¡Muchísimos éxitos! Y... ¡A CODEAR!