ANÁLISIS DE ARQUITECTURA

Sistema de Información de Cafetería

Se identifican las siguientes:

a) **Funcionalidades:**
   -Registrar Usuarios
   -Registrar Ventas
   -Controlar Inventarios
   -Administrar Proveedores
   -Administrar Costos, Gastos e Ingresos
b) **Componentes:**
   a) **MODULO USUARIOS**
   b) **MODULO VENTAS**
   c) **MODULO INVENTARIOS**
   d) **MODULO PROVEEDORES**
   e) **MODULO ADMINISTRATIVO (COSTOS, GASTOS E INGRESOS)**

1. MONOLITICA

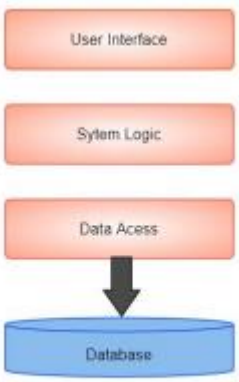

Figura 1- Arquitectura Monolítica

Se desarrolla todo en un mismo programa, alojándolo todo en un mismo servidor, sin separación entre módulos, con un mismo código y por tanto un mismo lenguaje, p.e. Java, en el que se programa una interfaz gráfica que será la que capture los datos de entrada en los diferentes módulos, se maneja una misma base de datos a la cual se accede mediante una cadena de conexión en el mismo programa, a través del cual también se procesa la información, y se dan las respectivas salidas.

2. MICROSERVICIOS

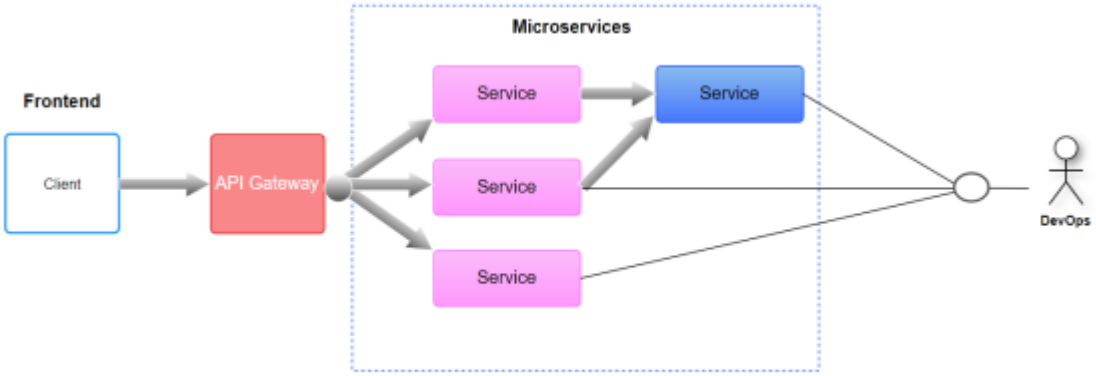

Figura 2- Arquitectura de Microservicios

Cada uno de los módulos se realiza por separado, en diferentes lenguajes incluso, y con diferentes bases de datos asociadas a diferentes servidores. Las funcionalidades se pueden realizar en diferentes aplicaciones.

Se utiliza un servicio de API REST para conectar los diferentes módulos o funcionalidades los cuales son independientes cada uno. El Front End se conecta con los diferentes servicios o funcionalidades y estos a su vez se conectan con las bases de datos o lógica del negocio.

La API REST administra las peticiones que se dan entre el cliente (usuario) y el servidor (lo que hay almacenado en el) las cuales pueden ser por acciones GET, POST, PUT, o DELETE.

Para una cafetería a gran escala, se usaría una metodología DevOps la cual permite la integración continua de todo lo relativo al aplicativo modularizado (Desarrollo + Operaciones). Esto permite automatizar tareas como Compilación de componentes, Pruebas unitarias, Pruebas de integración, Calidad de código, Escalado, Despliegue de software. El objetivo de la integración continua es permitir la integración de forma sencilla con distintos componentes de la aplicación y poder automatizar las pruebas. Para esta automatización de muchas aplicaciones, incluso en distintos sistemas operativos se puede utilizar Docker, este es un software de código abierto que permite virtualizar a nivel del sistema operativo usando contenedores; que, a diferencia de las máquinas virtuales, funciona con el sistema operativo del ordenador que lo ejecuta. Para controlar las aplicaciones con microservicios se utiliza un orquestador, el cual permite desplegar un gran número de contenedores dentro de un clúster, algunos de ellos son Docker Swarm, Apache Mesos, Kubernets.