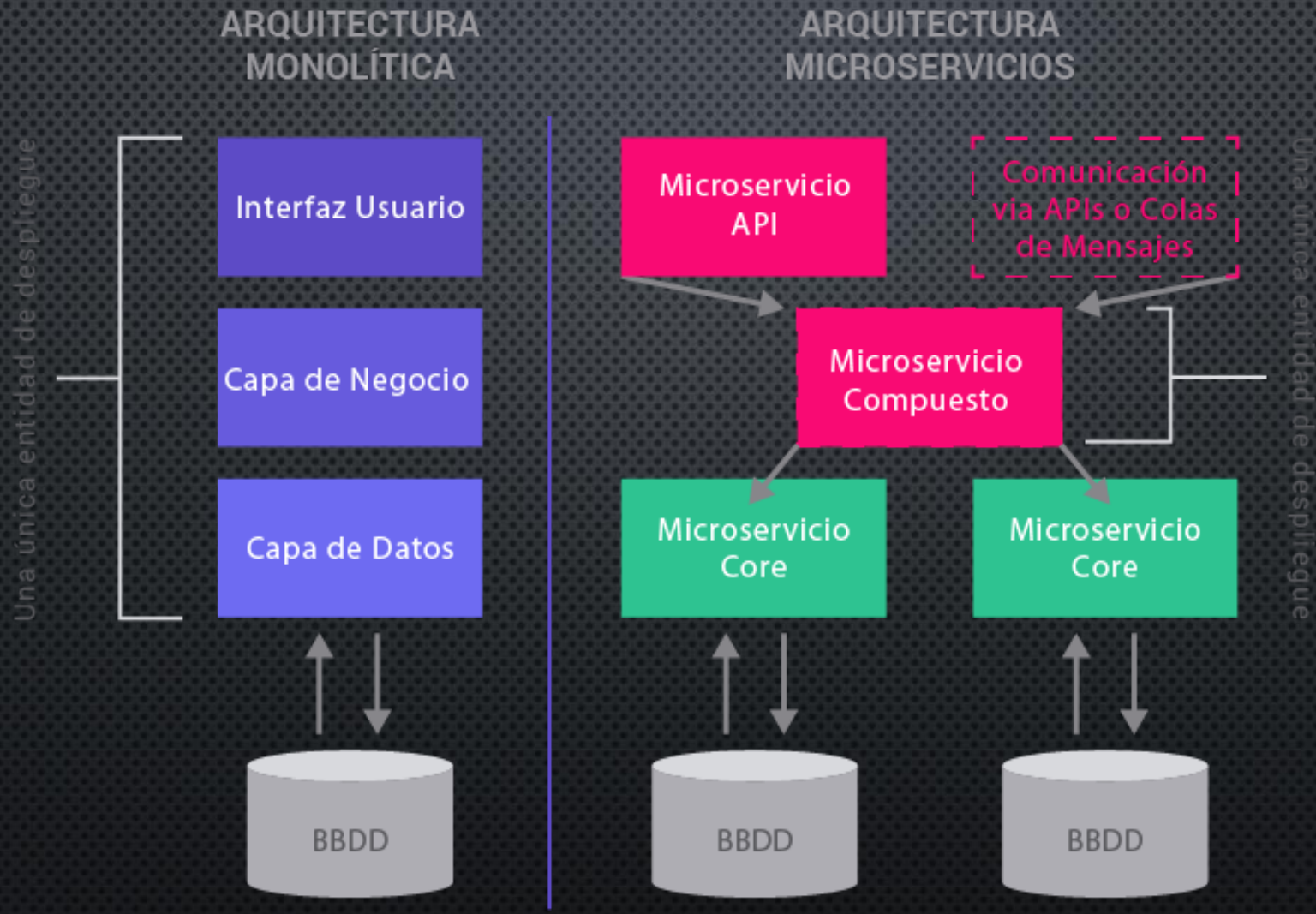
MICROSERVICIOS

MONOLÍTICA VS MICROSERVICIOS



QUE SON LOS MICROSERVICIOS

- Los Microservicios, son tan una arquitectura como un modo para desarrollar software que consiste en construir una aplicación de pequeños servicios independientes entre si, los cuales se ejecutan en su propio proceso y se comunican con mecanismos ligeros(normalmente una API REST).
- Pueden estar programados en distintos lenguajes y usar diferentes tecnologías de almacenamiento de datos

MICROSERVICIOS: VENTAJAS

- **Escalabilidad:** Estado eficiente, elástico y horizontal en función a la demanda . Brinda la alta escalabilidad.
- **Modularidad:** cada microservicio debe realizar una función acotada cada micro servicio hará una cosa y lo hará bien produciendo eficiencia y simplicidad.
- **Heterogéneas:** cada microservicio puede ser desarrollado en una tecnología diferente
- **Desacoplamiento:** cada microservicio se despliega independientemente, es responsable de una función acotada y maneja su propia base de datos, máximo desacople.
- **Rápido despliegue:** integración y despliegue continuo usando contenedores

DESVENTAJAS MICROSERVICIOS

- Alto consumo de memoria
- Necesidad de tiempo para poder fragmentar distintos microservicios
- Complejidad de gestión de un gran número de servicios
- Necesidad de desarrolladores para la solución de problemas como latencia en la red o balanceo de cargas
- Pruebas o testeos complicados al despliegue distribuido

TIPOS DE COMUNICACIÓN ENTRE MICROSERVICIOS

- La comunicación entre el cliente y los diferentes microservicios puede realizarse a través de diferentes tipos de comunicación, cada uno de ello enfocado a un escenario diferente. En general, **existen dos criterios para clasificar estos sistemas de comunicación**:
- **Por clase de protocolo:** sincrónico o asincrónico.
- **Por número de receptores:** uno o varios.

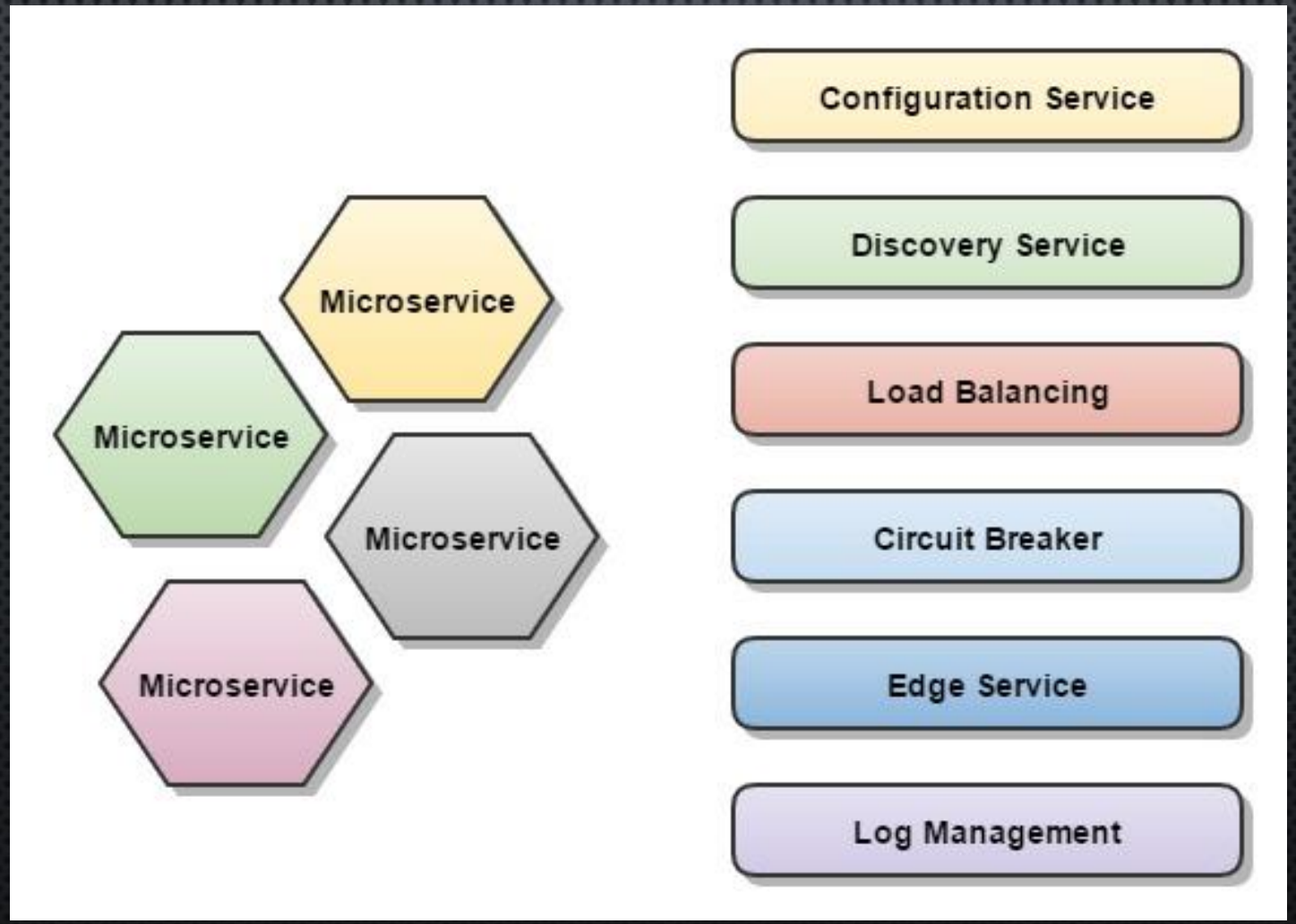
Protocolo sincrónico o protocolo asincrónico.

Protocolo sincrónico. Se caracteriza por ser un sistema que implica aislar lo máximo posible cada microservicio, puesto que **se produce un bloqueo de los subprocesos**. El **HTTP/HTPS** sería un ejemplo típico de protocolo de **comunicación entre microservicios** de tipo sincrónico, donde el cliente únicamente puede continuar su tarea en el momento que recibe una respuesta del servidor.

Protocolo asincrónico. En este caso, **los subprocesos no están bloqueados** y se utilizan protocolos compatibles con muchos sistemas operativos y entornos en la nube. Un ejemplo sería el **protocolo AMQP**, en el que normalmente el código del cliente o el remitente del mensaje no espera ninguna respuesta. Simplemente lo que hace es enviar un mensaje a una cola RabbitMQ o a cualquier otro agente de mensajes.

MODELO DE REFERENCIA MICROSERVICIOS

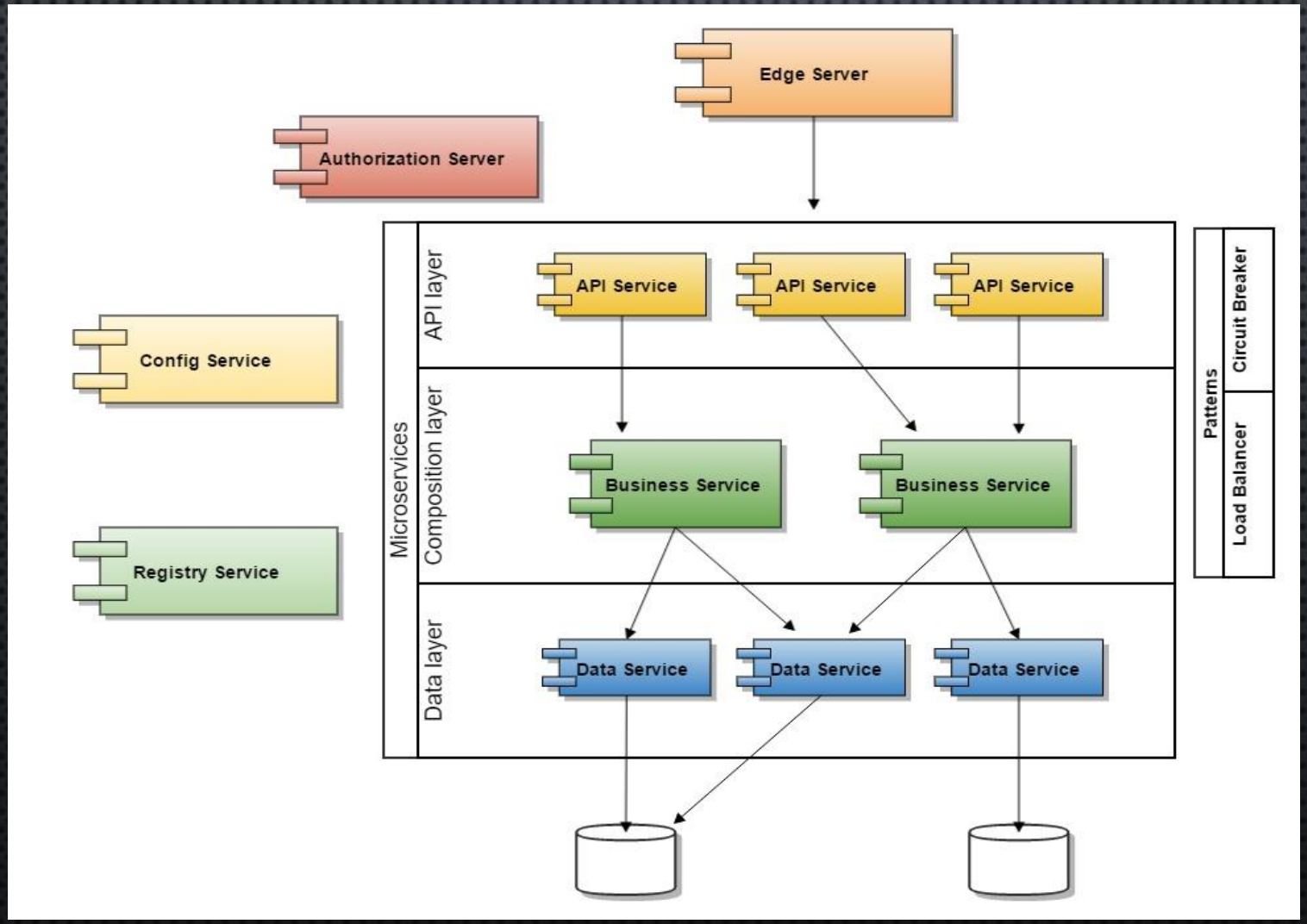
COMPONENTES QUE VAMOS A NECESITAR EN UNA ARQUITECTURA DE MICROSERVICIOS:



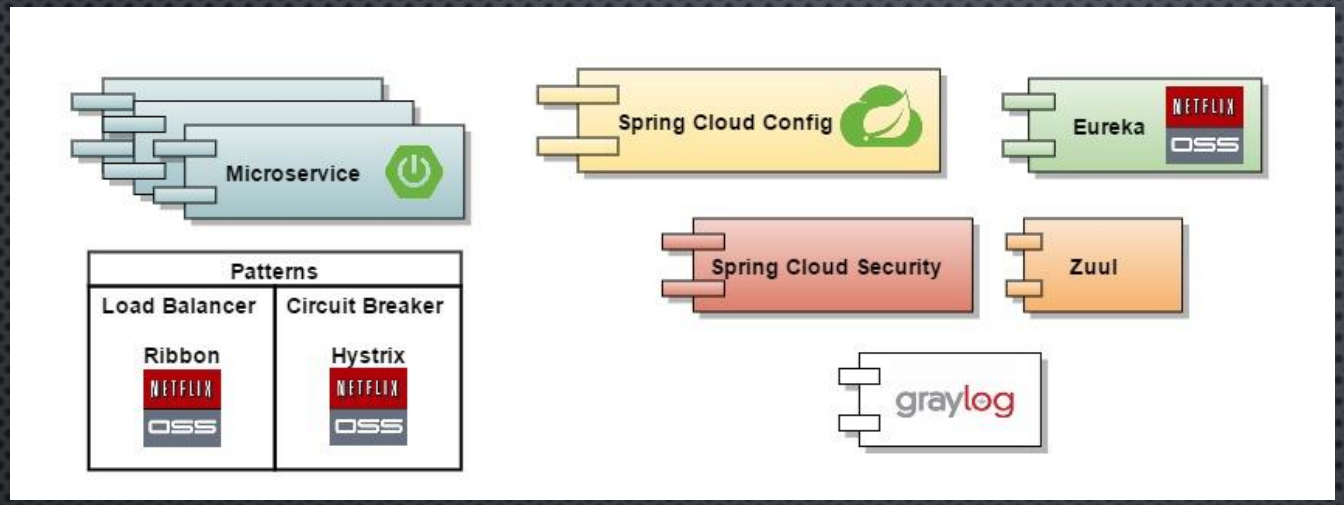
- **Servidor de configuración central**
- Este componente se encargará de centralizar y proveer remotamente la configuración a cada microservicio. Esta configuración se mantiene convencionalmente en un repositorio Git, lo que nos permitirá gestionar su propio ciclo de vida y versionado.
- **Servicio de registro / descubrimiento**
- Este servicio centralizado será el encargado de proveer los endpoints de los servicios para su consumo. Todo microservicio se registrará automáticamente en él en tiempo de bootstrap.
- **Balanceo de carga (Load balancer)**

  Este patrón de implementación permite el balanceo entre distintas instancias de forma transparente a la hora de consumir un servicio.
- **Tolerancia a fallos (Circuit breaker)**
  - Mediante este patrón conseguiremos que cuando se produzca un fallo, este no se propague en cascada por todo el pipe de llamadas, y poder gestionar el error de forma controlada a nivel local del servicio donde se produjo.
- **Servidor perimetral / exposición de servicios (Edge server)**
  - Será un Gateway en el que se expondrán los servicios a consumir.
- **Centralización de logs**
  - Se hace necesario un mecanismo para centralizar la gestión de logs. Pues sería inviable la consulta de cada log individual de cada uno de los microservicios.
- **Servidor de Autorización**
  - Para implementar la capa de seguridad (recomendable en la capa de servicios API)
- **Monitorización**

DIAGRAMA DE MICROSERVICIOS EN CAPAS

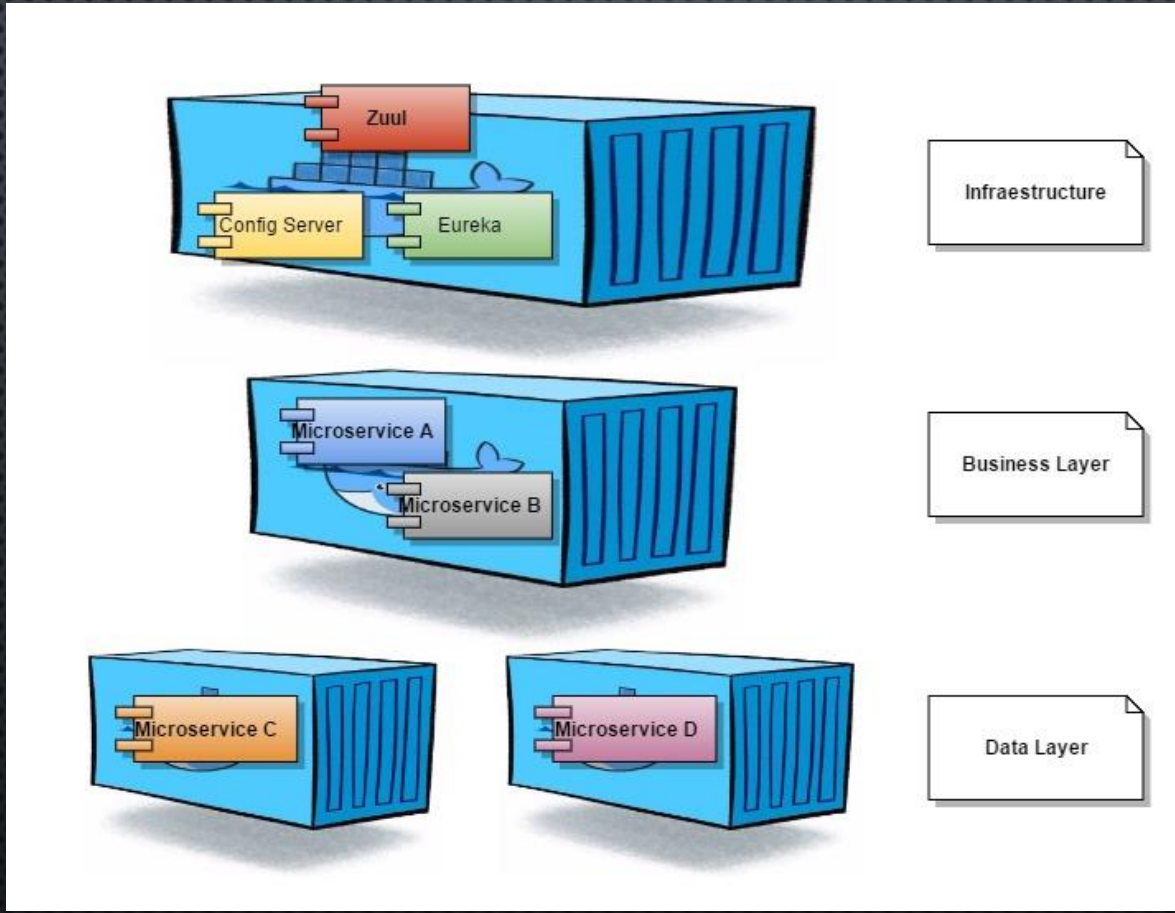


MODELO IMPLEMENTACIÓN



- **MICROSERVICIOS** PROPIAMENTE DICHOS: SERÁN APLICACIONES SPRING BOOT CON CONTROLADORES SPRING MVC. UTILIZAREMOS SWAGGER PARA DOCUMENTAR Y DEFINIR NUESTRO API.
- **CONFIG SERVER**: MICROSERVICIO BASADO EN SPRING CLOUD CONFIG. UTILIZAREMOS GIT COMO REPOSITORIO DE CONFIGURACIÓN.
- **REGISTRY / DISCOVERY SERVICE**: MICROSERVICIO BASADO EN **EUREKA** DE NETFLIX OSS.
- **LOAD BALANCER**: UTILIZAREMOS **RIBBON** DE NETFLIX OSS QUE YA VIENE INTEGRADO EN REST-TEMPLATE DE SPRING.
- **CIRCUIT BREAKER**: UTILIZAREMOS **HYSTRIX** DE NETFLIX OSS.
- **GESTIÓN DE LOGS**: UTILIZAREMOS **GRAYLOG**
- **SERVIDOR PERIMETRAL**: UTILIZAREMOS **ZUUL** DE NETFLIX OSS.
- **SERVIDOR DE AUTORIZACIÓN**: IMPLEMENTAREMOS EL SERVICIO CON **SPRING CLOUD SECURITY**.

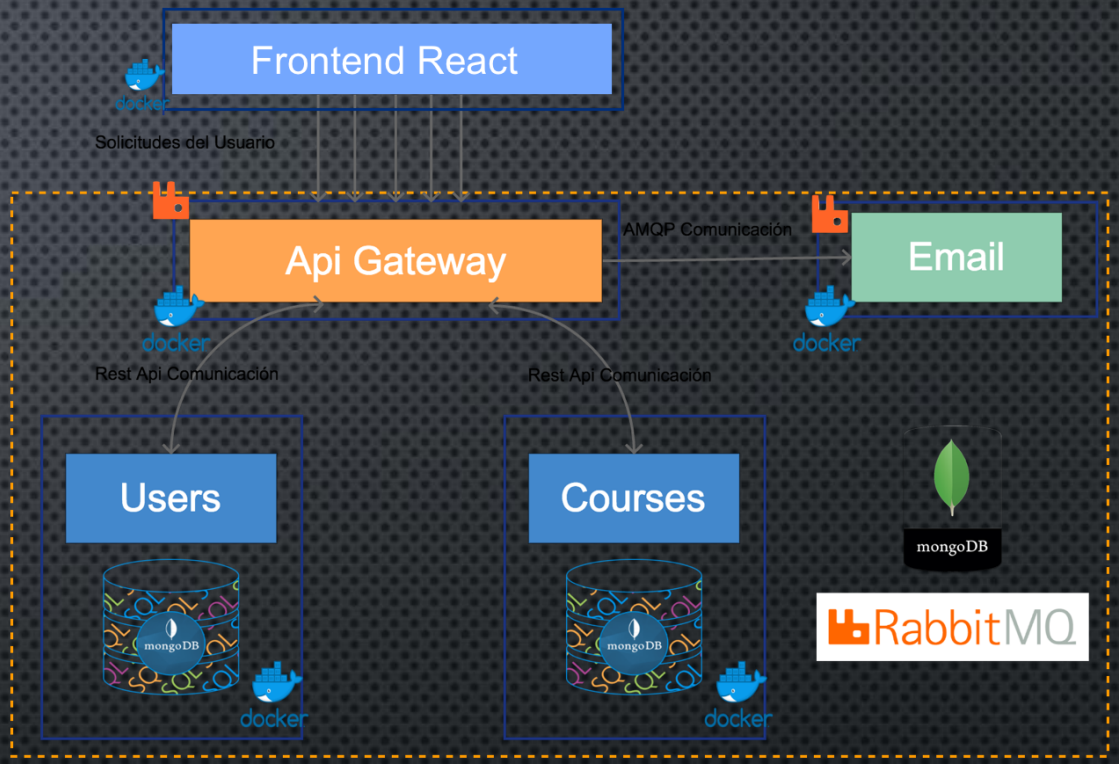
MODELO DE DESPLIEGUE



Diagrama

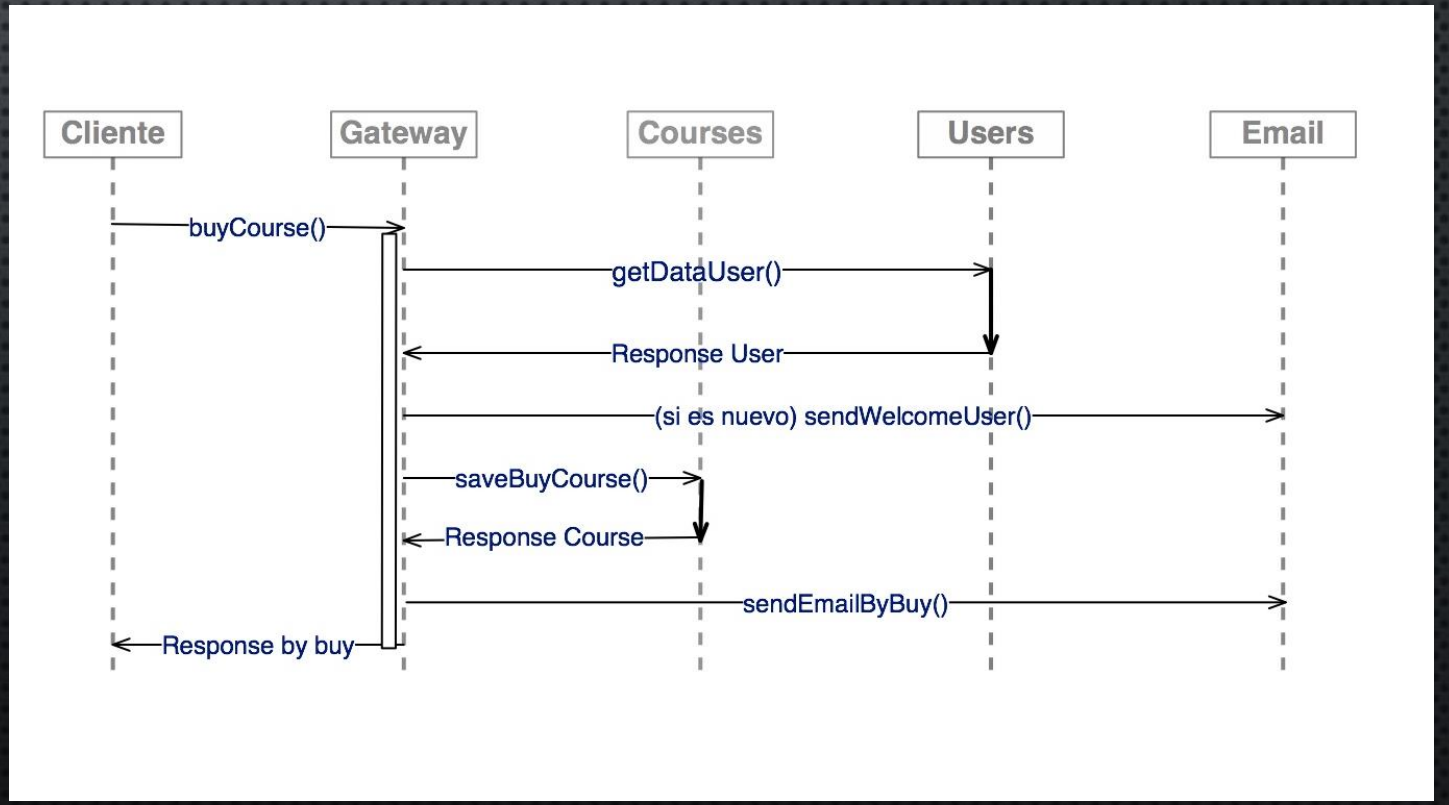
Para este pequeño ejemplo utilizaremos cuatro microservicios :

- **Api Gateway.-** Tiene la tarea de comunicarse con el frontend y según las peticiones de el usuario se comunica con los otros microservicios.
- **Users y Courses.-** Dos microservicios que están en un nivel atómico estos solo se encargan de ellos. Notemos que cada microservicio tienes su propia base de datos.
- **Email.-** Microservicio encargado solo de enviar correos a los usuarios, notemos que no tiene una comunicación REST API como en los anteriores microservicios, para este utilizaremos el protocolo AMQP ayudandonos con RabbitMQ, esto nos dará la capacidad de poner en cola las peticiones que le estén llegando.



Flujo

Se mostrará una variedad de libros o cursos de los cuales el usuario podrá seleccionar alguno, si desea obtener información de algún articulo entonces procedemos a pedirle su Correo para mandarle la información del artículo que selecciono ,si el Usuario es nuevo entonces se le enviara un correo de bienvenida a la aplicación, posteriormente se le enviaremos la información del artículo que selecciono.



- GRACIAS