Diseño de Sistemas

Agenda
de la clase

- Arquitectura: Repaso
- Caso de estudio (repaso): Rekomendashi, Fase 2
- Caso de estudio: She-Ra Enterprise
- Microservicios
- Caché
- Caso de estudio: Cyberfriday

1.
Arquitectura: Repaso

¿Donde estábamos?

Un poco de repaso

- La arquitectura trata
  - del diseño de lógico de alto nivel
  - de toma de decisiones sobre aspectos difíciles de cambiar
  - del diseño centrado en satisfacer requerimientos no funcionales
  - el diseño de los aspectos físicos del sistema
- Contamos con componentes y estilos / patrones

Un poco de repaso

Necesitamos definir:

- entre qué componentes lógicos y físicos separaremos nuestro sistema: arquitecturas monolítcas vs distribuidas
- en dónde desplegaremos: Bare Metal vs En la nube (*IaaS, PaaS, SaaS*)
- cómo escalaremos: escala Vertical, Horizontal y Selectiva

2.
Repaso: Caso de Estudio - Rekomendashi

Lluvia de nigiris

Escala horizontal

- La escalabilidad nos habla de la capacidad de crecer de un sistema sin interrumpir su servicio, en distintos ejes: de carga, geográfica y administrativa
- La escala horizontal nos permite aumentar la capacidad de de atención de un servicio en términos de carga, agregando dispositivos de cómputo en lugar de mejorando sus prestaciones.
- Necesitamos de un balanceador de carga (*load balancer*)

Escala horizontal

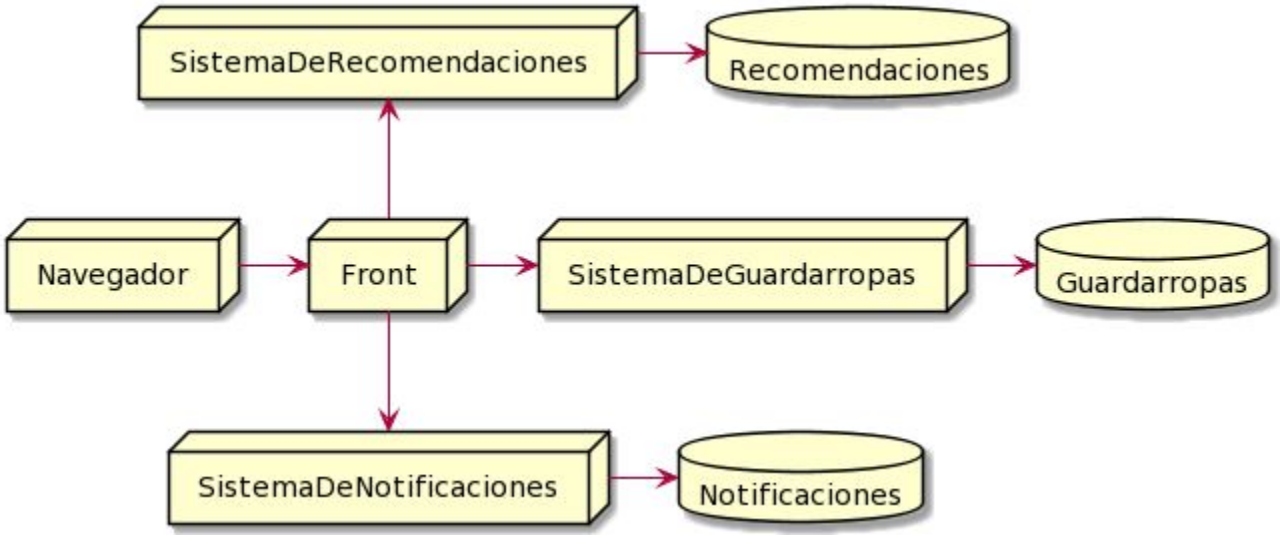
No todo es oro; surgen nuevos problemas:

- El balanceador se convierte en un punto único de fallo (*SPOF)*
- Tenemos que tener cuidado con el manejo de sesión:
  - Sesión basada en cookies (no es el *default* de Java)
  - Sesión sticky en memoria (es una configuración del balanceador)
  - Sesión en memoria compartida *(es un nuevo SPOF)*

3.
Caso de Estudio - SheRa

Por el (micro) poder de los servicios

Arquitectura
Microservicios



Arquitectura
Microservicios

- Particionar el sistema en servicios independientes según un criterio funcional e, idealmente, sin almacenamiento compartido
- Cada servicio puede estar escrito en tecnologías diferentes
- Estrategia de división organizacional: cada servicio es mantenido por un equipo diferente
- Para coordinar los servicios, pueden ser necesarios
  - Orquestadores / Pasarelas (*Gateways*)
  - *Message Brokers*

Escala
selectiva

Escalar cada servicio según cuánto y cómo necesite:

- Permite combinar escala vertical y horizontal en diferentes servicio
- Permite ajustar el nivel de escala de carga servicio

4.
Caso de Estudio - Ciberfriday

Gotta cache' all

Sin
Caché

```
1 const value = procesoCostoso()
```

Con
Caché

```
1 //
2 const value = cache.get(clave)
3 if (value == NULL) {
4   value = procesoCostoso()
5   cache.set(clave, procesoCosto)
6 }
7 // uso value
8
```

Caché or not Caché

¡Muchas nuevas decisiones!

- ¿Podemos cachear?
- Tamaño: ¿Qué y cuánto?
- Expiración: ¿Por cuánto?
- Invalidación: ¿Hasta cuándo?
- Tipo: ¿Dónde?
  - Memoria del proceso
  - Servidor de caché (en memoria)
  - Proxy HTTP Inverso

Muchas Gracias!!

Si tienen consultas

https://github.com/dds-jv/foro