

**PROJECTE DE XARXES**

**TREBALL PRÀCTIC: Load Balancing, High Availability i Monitorització de Microserveis**

**MEMÒRIA**

**Camilo Urán Pico, u1062049,** [**u1062049@campus.udg.edu**](mailto:u1062049@campus.udg.edu)

**Girona, Abril de 2024**

**Continguts**

[1 Introducció i objectius 1](#_Toc164896120)

[2 Conceptes bàsics 1](#_Toc164896121)

[6 Resum, conclusions i treball futur 2](#_Toc164896122)

[7 Treball en grup, dedicació i suggeriments 2](#_Toc164896123)

[Bibliografia 2](#_Toc164896124)

# 1 Introducció i objectius

Amb aquest treball volem fer servir eines relacionades amb els microserveis. Aquests son petits serveis el ús dels quals s’ha fet popular a les ultimes dècades, com a una alternativa a les aplicacions monolítiques1. Tot i que es fan servir per a fragmentar sistemes informàtics molt grans, i difícils de mantenir, els microserveis introdueixen una família nova de problemes, principalment els relacionats amb el que es coneix com a East-West traffic2.

Per exemple, trencar una aplicació gran en petits microserveis vol dir que tots els serveis han de operar a la mateixa vegada, amb el mínim de retards. Una petició a un monòlit es atesa per la mateixa aplicació, i això vol dir que el retard entre crides és mínim. En contrast, un microservei normalment ha de parlar amb altres microserveis per a completar la seva feina, i aquesta comunicació afegeix un cost que s’ha de considerar amb precaució3.

Al nostre cas, en particular, volem conèixer com:

* Controlar el tràfic que arriba als diferents microserveis (ingress). Per exemple, fer Load Balancing per a permitir que la carrega es distribueixi a tots els serveis disponibles.
* Fer servir memòries caus de dades, per a reduir el cost de peticions a la nostra aplicació que es repeteixen i que no depenen de dades volàtils – per exemple, demanar la llista de dies feiners al més de Maig. Aquesta petició segurament retornarà les mateixes dades cada cop que es faci.
* Monitoritzar el tràfic als nostres microserveis, per exemple el numero de peticions per hora, el numero de fallades 500, 404, etc.

El objectiu per tant es conèixer més les diferents eines que es fan servir al ecosistema dels microserveis.

# 2 Conceptes bàsics

**API**: Acrónim per Application Programming Interface. És el nom que se li sol donar a un sistema que per lo general s’encarrega de proporcionar una interfície a través de la qual es pot accedir a un altre servei. Per exemple, una API pot donar accés de forma estructurada a la informació a una base de dades. En lloc de accedir directament mitjançant una connexió mysql, es pot crear una aplicació que contesti a preguntes més concretes, com ara “quina és la llista dels usuaris registrats al mes del Gener“ o “crear un event al calendar per al mes de Setembre”. Les APIs es poden implementar mitjancant aplicacions monolítiques, o amb microserveis.

**Balancejament de carrega (Load Balancing)**: Un sistema de balancejament s’encarrega de distribuir el tràfic que arriba a un sistema. Per exemple, si tenim 2 ordinadors amb copies de la aplicació executant-se, el Load Balancer es encarrega de que el 50% de les peticions vagin a la primera màquina, i que el 50% restant vagin a l’altra. Nginx, tot i ser un servidor web, també és pot configurar com a Reverse Proxy i balancejador.

**Docker**: Eina de orquestració de sistemes complexos de software, que permet aïllar aplicacions, i executar múltiples copies de la mateixa aplicació, afegint una capa d’abstracció a la configuració normal d’un sistema distribuït, com ara els seus ports d’escolta, les seves variables d’entorn, etc. Farem servir Docker per a executar les aplicacions que volem fer servir, perquè ens permet controlar de millor manera la seva configuració i el seu estat.

**Memoria Cau (cache)**: Memoria d’emmagazetmage que es far servir per a mantenir un accés més ràpid a dades que es consulten sovint. Per exemple, el llistat de tots els cotxes a un concessionari de cotxes no canvia amb gran velocitat, i els cotxes que hi ha a les 2 de la tarde potser son els mateixos que hi ha a les 8 de la tarde. Durant aquest període de temps, no cal demanar a la base de dades cada vegada – les crides a la base de dades solen ser costoses, i si podem estalviar unes quantes, molt millor.

**Microservei**: Es tracta d’un sistema de software que per definició ha de ser petit, per lo general això es aconsegueix mitjançant la partició dels dominis i les responsabilitats – per exemple, en comptes de tenir una gran aplicació web que tingui una cesta de la compra, una passarel·la de pagament i un sistema per a enviar correus electrònics, podem tenir tres serveis que facin una part de tota aquesta feina. Dins dels avantatges de fer servir microserveis trobem la facilitat de actualitzar només una part petita del sistema, sense que es vegi afectar a la resta.

**Porta d’enllaç API (API Gateway)**: Es tracta d’una aplicació que controla el tràfic que arriba a una API, o a un conjunt d’aquestes. Es pot pensar com a una mena d’enrutador virtual, que permet tota classe de control i de configuració del tràfic que entra i surt. Per exemple, es podria dir a la porta d’enllaç que agafés les peticions que porten una cookie amb un determinat valor, i que les enviés a un servidor amb més amplada de banda i més capacitat de CPU.

**Traffic entrant (Ingress traffic)**: Tot el tràfic que entra a un sistema, pot ser a una API, o a una col·lecció de APIs.

**3 Preparació del entorn de desenvolupament**

Farem servir les següent eines:

* Go Lang: com a llenguatge de programació dels microserveis
* Docker: Per a executar els microserveis de manera aïllada
* Kong Api Gateway: Com a router dels nostres serveis
* Prometheus: Per a recollir dades dels serveis
* Grafana: per a visualitzar les dades recollides

# 6 Resum, conclusions i treball futur

Expliqueu el següent:

- El què heu fet, de manera resumida.

- Problemes que us heu trobat, coses que no han funcionat, i si s’han solucionat i com.

- El què heu descobert, deduït, lliçons apreses, recomanacions, etc.

- Com es podria continuar aquest treball, coses a explorar, a millorar, etc.

# 7 Treball en grup, dedicació i suggeriments

Expliqueu com heu treballat de manera conjunta per fer la pràctica.

Dieu les hores de classe (és a dir, 8.5h) i les hores fora de classe que heu necessitat (hores individuals per persona, és a dir, no sumeu les de la parella d’estudiants).

Suggeriments als professors a fi de millorar aquest treball.

# Bibliografia

[1] H, Chandler, *Microservices vs. monolithic architecture* [en línea]. Consulta: 24 de març de 2024. Disponible a: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith

[2] D, Daniels, *East-West Traffic: Everything You Need to Know* [en línea]. Consulta: 24 de març de 2024. Disponible a: <https://blog.gigamon.com/2023/11/06/east-west-traffic/>

[3] Thorben, *Communication Between Microservices: How to Avoid Common Problems*, [en línea]. Consulta: 24 de març de 2024. Disponible a <https://stackify.com/communication-microservices-avoid-common-problems/>