Mikroservisna Arhitektura

Dropwizard Vs Spring Boot

Nenad Pečanac
Serengeti

Sadržaj

1) Definicija
2) Monolitna arhitektura
3) Mikroservisna arhitektura
4) Prednosti i mane mikroservisa
5) Dropwizard Vs Spring Boot
6) Reference

1) Definicija

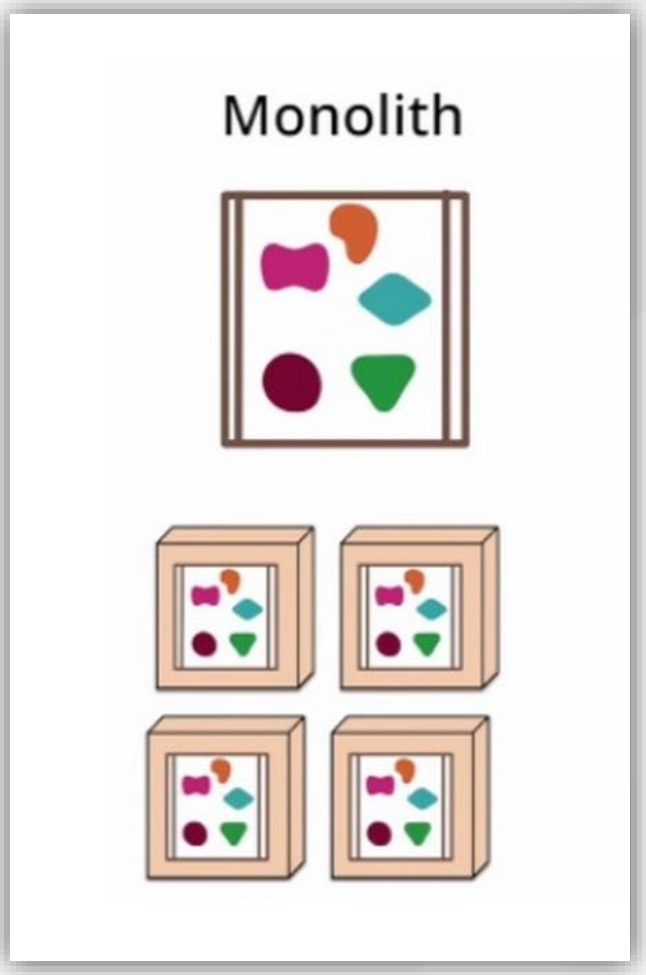
Mikroservisna Arhitektura

"Mikroservisna arhitektura je princip razvoja aplikacija u obliku malih, izdvojenih servisa, pri čemu svaki servis ima svoj proces i ostvaruje komunikaciju putem jednostavnih mehanizama kao što je HTTP API".

Martin Fowler, ThoughtWorks

2) Monolitna arhitektura

Monolitna arhihetktura



Monolitna arhihetktura (1)

Klasičan oblik arhitekture.

Aplikacija je izgrađena kao jedna cjelina.

Tipična troslojna Enteprise aplikacija:

- Klijentski UI (Browser, HTML + JS)
- Baza podataka (RDBMS, NoSql ..)
- Serverska aplikacija

(Java, .NET, Ruby, Python, PHP ..)

Monolitna arhihetktura (2)

Serverska aplikacija:

- obrađuje HTTP zahtjeve
- izvršava poslovnu logiku
- čita i ažurira podatke iz baze
- generira HTML i šalje ga klijentu.

Serverska aplikacija je Monolit -

jedinstvena logička izvršna cjelina.

Monolitna arhihetktura (3)

Monolitni server – prirodan način razvoja.

Sva logika obrade zahtjeva se izvršava u jednom procesu, razdijeljenom i organiziranom u klase, metode i pakete.

Aplikaciju je moguće razvijati i testirati na laptopu nakon čega slijedi postavljanje na testnu i produkcijsku okolinu.

Monolit se horizontalno skalira izvršavanjem više instanci aplikacije iza load-balancera.

Monolitna arhihetktura (4)

Promjene bilo kojeg dijela aplikacije zahtjevaju ponovno prevođenje i postavljanje nove verzije aplikacije na testnu i produkcijsku okolinu.

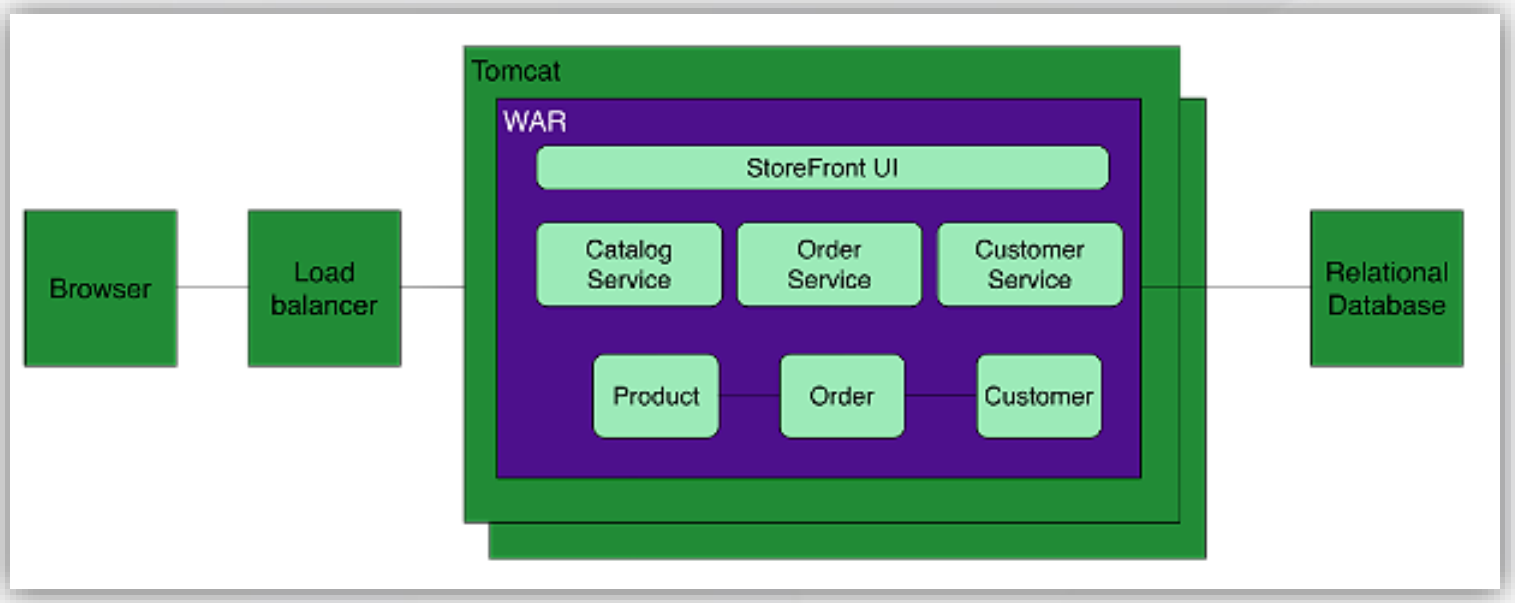
Promjene zahtjevaju dobro planiranje i koordinaciju.

Promjene su skupe.

Dobru modularnu strukturu je teško postići.

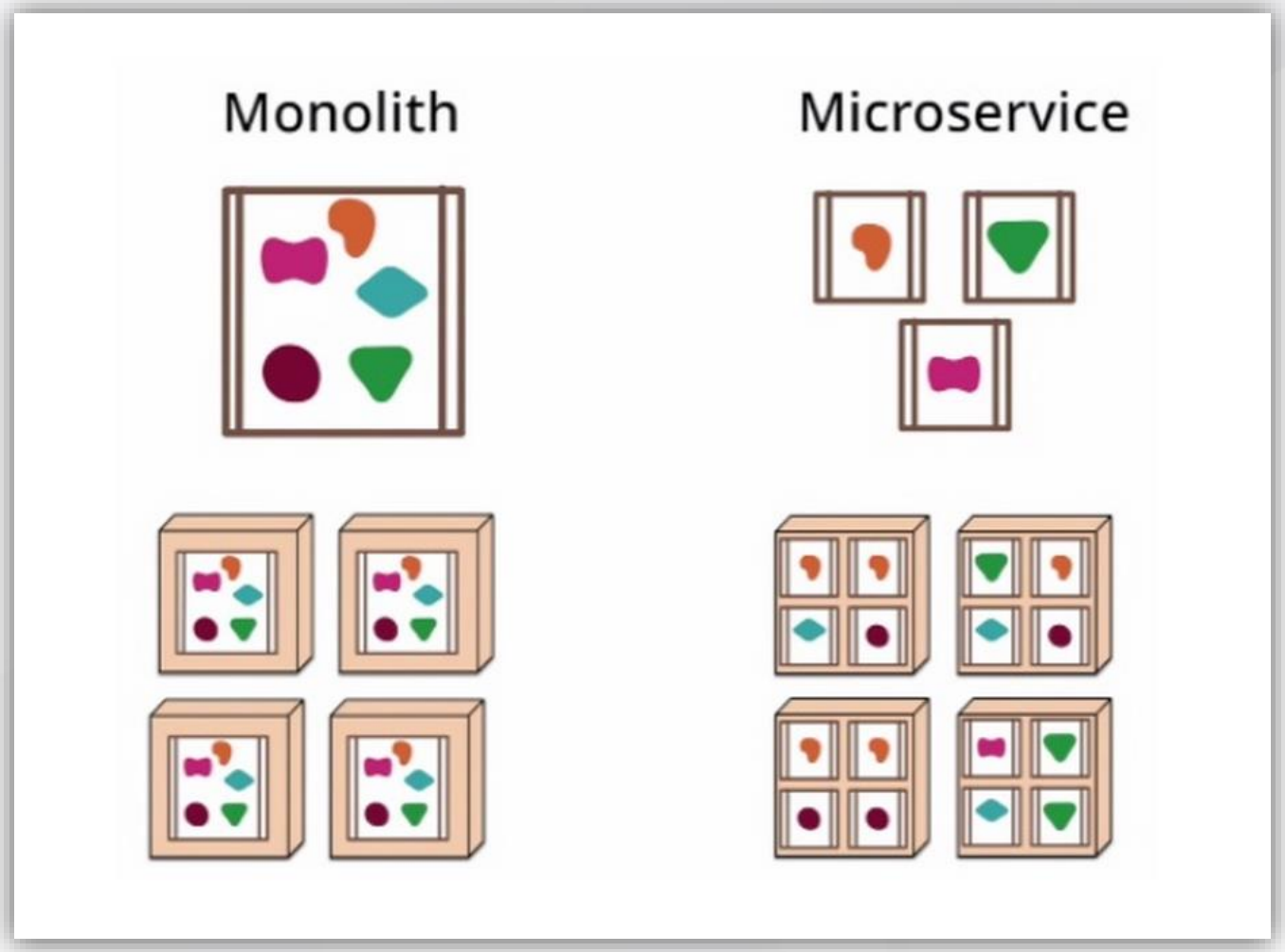
Skaliranje se provodi skaliranjem cijele aplikacije, umjesto skaliranjem dijelova koji zahtjevaju više resursa.

Monolitna arhihetktura
Primjer



3) Mikroservisna arhitektura

Mikroservisna arhitektura



Mikroservisna arhitektura (1)

Aplikacije se počinju razvijati kao Monoliti,
po potrebi se skaliraju i s vremenom prelaze
na Mikroservisnu arhitekturu.

Dijelovi aplikacije se izdvajanju iz Monolita zbog
boljeg upravljanja resursima.

Aplikacije se rastavljanju na Komponente
- manje, nezavisne servisne aplikacije.

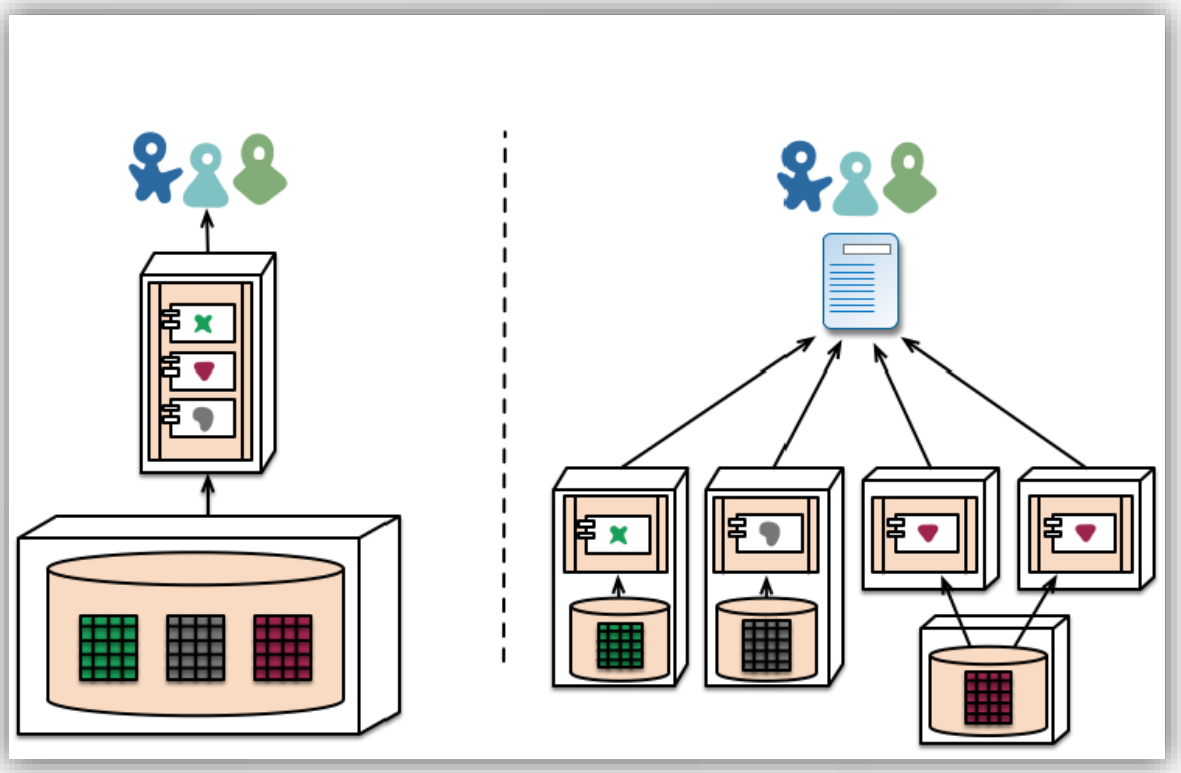
Mikroservisna arhitektura (2)

Komponente se grade nad poslovnim cjelinama.

Svaka komponenta se izvršava kao zasebna aplikacija u potrebnom broju instanci na klasteru.

Mikroservisna arhitektura (3)

Decentralizirano upravljanje podacima.



Mikroservisna arhitektura (4)

Komponente trebaju biti nezavisne i vrlo učinkovito povezane.

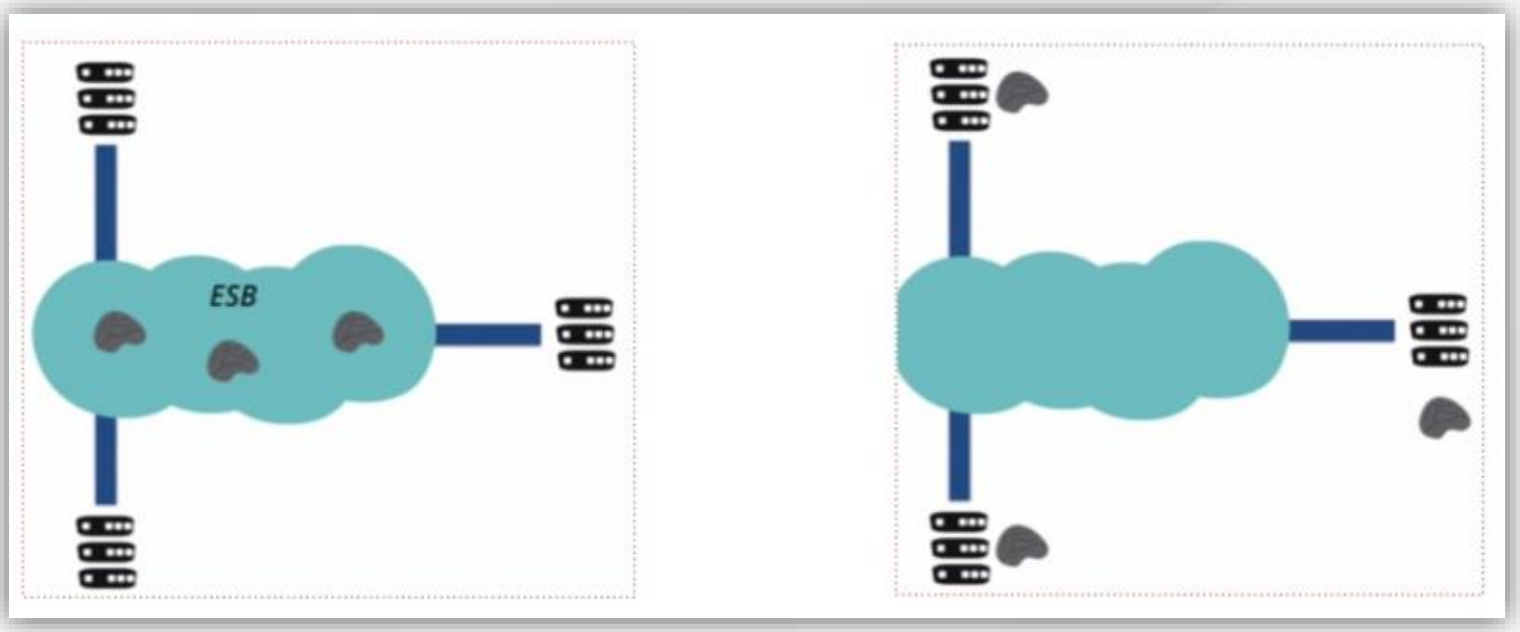
Komponente najčešće komuniciaju preko REST-a.

Ponekad se koristi jednostavna asinkrona komunikacija (RabbitMQ, Zero MQ).

Ne koristi se ESB ili neki drugi centralni komunikacijski mehanizam.

Mikroservisna arhitektura (5)

Decentralizirana komunikacija:



Mikroservisna arhitektura (6)

Komponente prema dizajnu moraju biti otporne na poteškoće i kvarove.

Svaka komponenta se može pokvariti ili postati nedostupna, u bilo kojem trenutnku.

Pravovremena detekcija grešaka i automatski oporavak, ukoliko je moguć, su vrlo važni.

Mikroservisne aplikacije pridaju puno važnosti nadgledanju rada komponenti i sustava u cijelosti.

Mikroservisna arhitektura (7)

Netflixov Chaos Monkey svakodnevno izaziva kvarove komponenti kako bi se testirala otpornost sustava na greške.

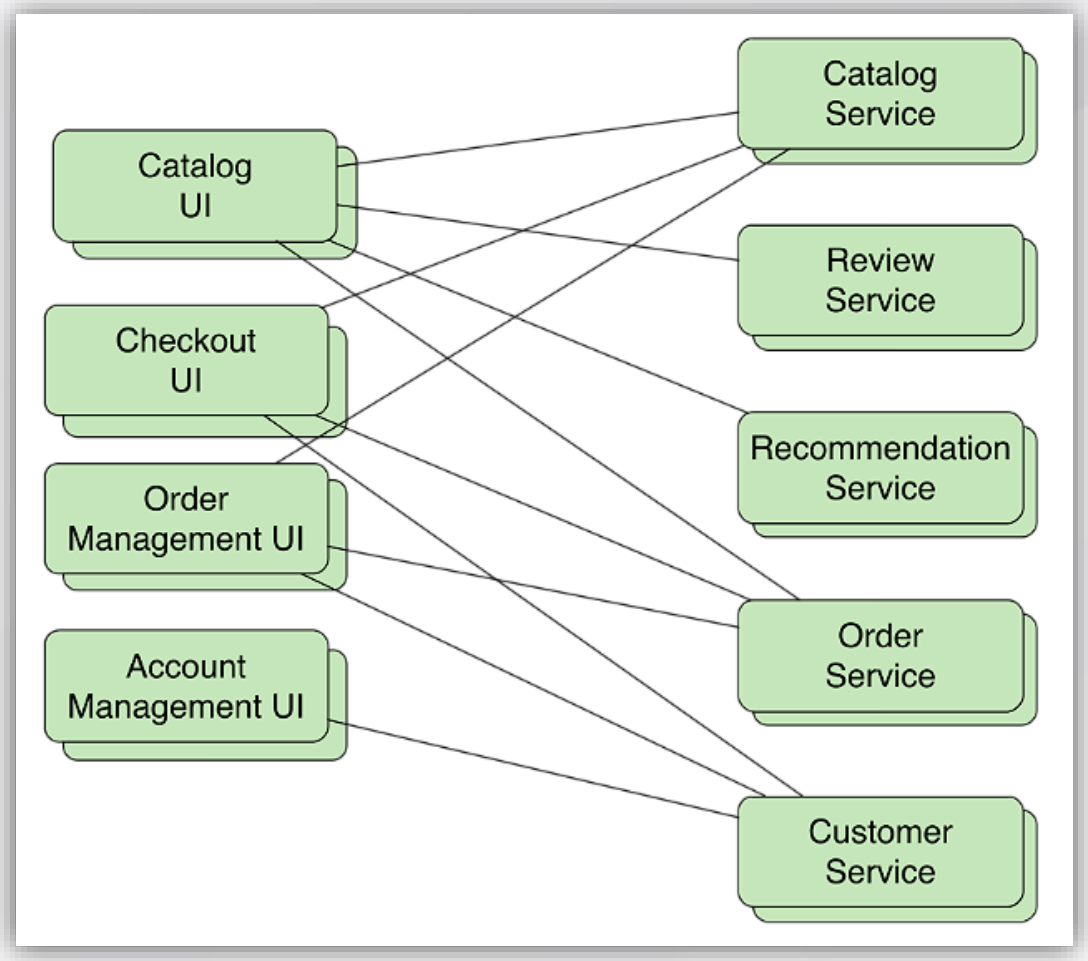
Mikroservisi i SOA

SOA	Mikroservisi
XML	JSON
Complex to integrate	Easy to integrate
Heavy	Lightweight
Requires tooling	Light tooling
HTTP/SOAP	HTTP/REST

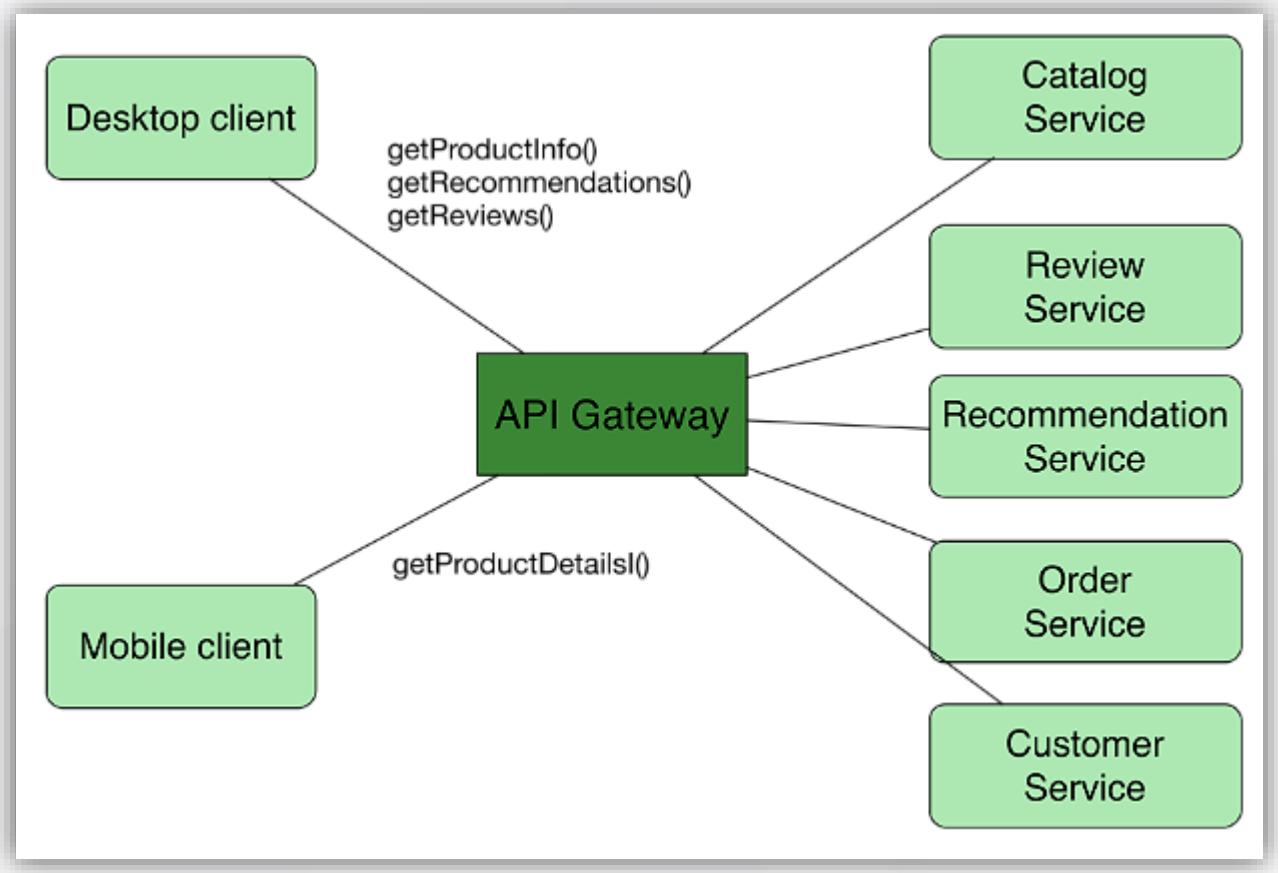
Mikroservisna arhitektura

Tko koristi Mikroservise u produkcijskim sustavima?

Mikroservisna arhitektura
Primjer



Mikroservisna arhitektura
API Gateway



4) Prednosti i mane mikroservisa

Prednosti (1)

Razvojni timovi mogu razvijati i isporučivati aplikacije *relativno neovisno* jedni o drugima.

Komponente mogu biti implementirane sa različitim tehnologijama i pisane u različitim programskim jezicima.

Različiti timovi mogu razvijati komponente.

Moguće je korištenje različitih baza podataka.

Centralizirano upravljanje je minimalno.

Prednosti (2)

Komponente se mogu nezvisno isporučivati i automatski postavljati na servere.

Mikroservisna arhitektura dobro funkcionira uz Continuous Integration i Continuous Delivery.

Moguće je često izdavanje i ažuriranje verzija komponenti uz održavanje stabilnosti ostatka sustava.

Mane (1)

Složenost distribuiranih sustava.

Složenost međuservisne komunikacije.

Nepostojanje distribuiranih transakcija.

Složenost operativnih procesa.

Otežano testiranje zbog interakcija komponenti.

Potreba za robusnim upravljanjem greškama i automatskim oporavkom.

Potreba za sofisticiranim nadgledanjem rada.

5) Dropwizard Vs Spring Boot

Dropwizard Vs Spring Boot (1)

Mikroservisni kontejneri i razvojni okviri za ubrzanu izradu mikroservisnih aplikacija.

Konvencija umjesto konfiguracije.

Aplikacije se pakiraju u izvšne JAR datoteke sa ugrađenim serverima (Jetty, Tomcat, Undertow).

Podrška za nadgledanje, logiranje i provjeravanje.

Podrška za korištenje standardnih biblioteka.

Dropwizard Vs Spring Boot (2)

	Dropwizard	Spring boot
HTTP	Jetty	Tomcat (default), Jetty or Undertow
REST	Jersey	Spring (default), JAX-RS
JSON	Jackson	Jackson, GSON, json-simple
Metrics	Dropwizard Metrics	Spring
Health Checks	Dropwizard	Spring
Logging	Logback, slf4j	Logback, Log4j, Log4j2, slf4j, Apache common-logging
Official integrations	Hibernate Validator, Guava, Apache HttpClient, Jersey client, JDBI, Liquibase, Mustache, Freemarker, Joda time	40+ Official Starter POMs for any purpose
Community integrations	Tens of available integrations, including Spring	4 Community led POMs

Dropwizard Vs Spring Boot (3)

Dropwizard podržava Jetty.

Spring Boot se fokusira na Spring aplikacije, a podržava Tomcat, Jetty i Undertow.

Dropwizard imlementira REST preko Jersey-a.

Spring Boot implementira REST kroz standardnu podršku iz Spring Framework-a.

Dropwizard koristi Logback, Spring podržava Logback, log4j I log4j2.

Dropwizard Vs Spring Boot (4)

Glavna razlika među kontejnerima je podrška za Dependency Injection.

Spring ima ugrađenu podršku dok Dropwizard ostavlja mogućnost integracije sa okvirom po izboru.

Oba kontejnera sadrže posebne module za testiranje koji uključuju JUnit i Mockito.

Razlike:

1) Spring Boot je dio Spring-ovog ekosustava i najprimjereniji je korišenju u Spring projektima.
2) Spring pruža široku listu podržanih tehnologija: REST, JMS, Messaging …
3) Dependency Injection – Dropwizard omogućava integraciju sa okvirima kao Guice, CDI, Pico …
4) Dropwizard sadrži popularnu Metrics biblioteku za nadgledanje rada aplikacija.

6) Reference

Reference

Martin Fowler: Microservices

Craig Richardson: Microservice Architecture Pattern

Craig Richardson: Decomposing Applications

Stefan Borsje: How we built microservcies at Karma

Benjamin Wootton: Microservices - Not a free lunch

Dropwizard: http://www.dropwizard.io/

Spring Boot: http://projects.spring.io/spring-boot/

Alex Zhitnitsky: Dropwizard Vs Spring Boot

Rizvan Ulah: Dropwizard Vs Spring Boot