Napredne arhitekture informacionih sistema

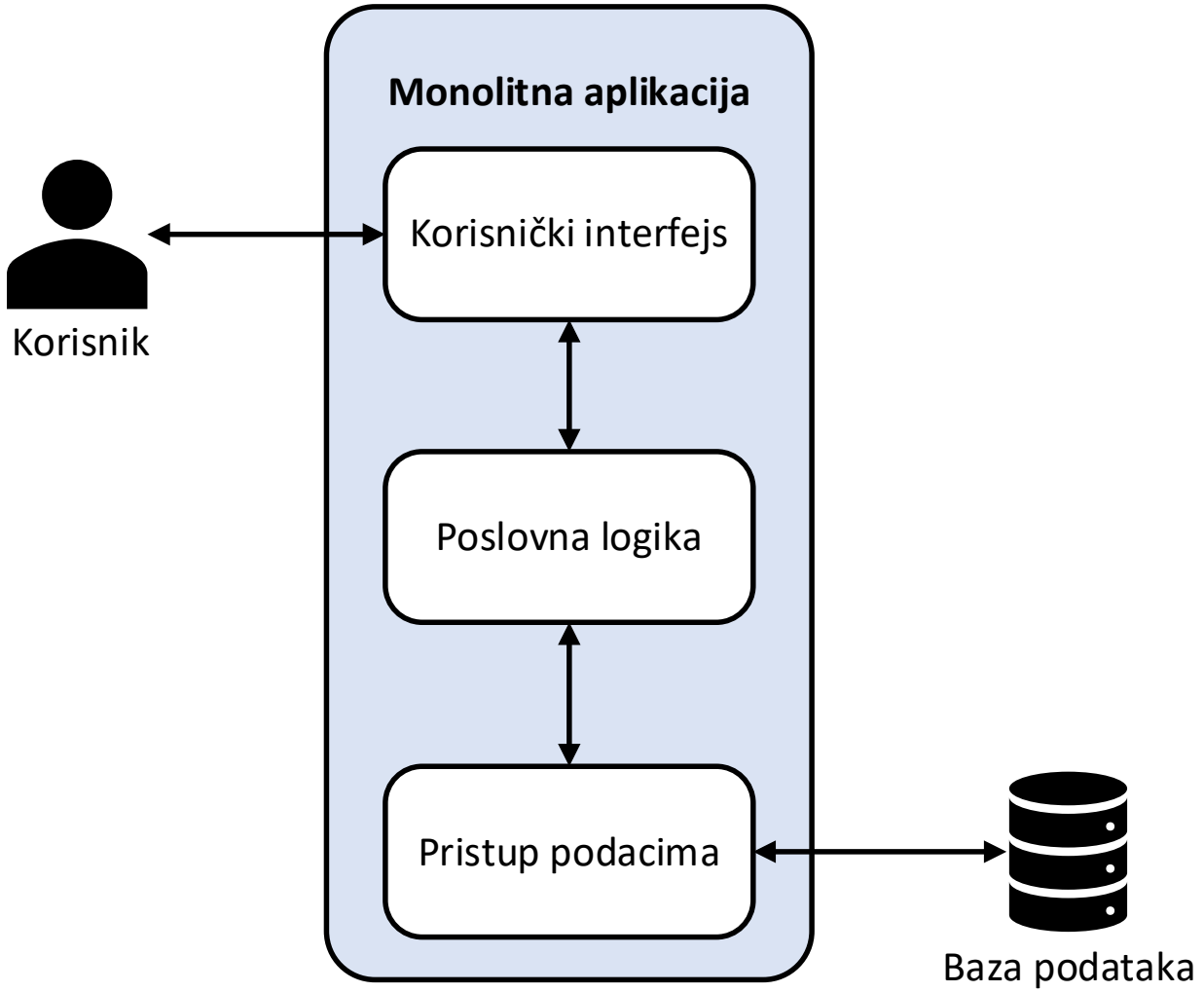
Mikroservisi i kontejnerizacija

Predmetni nastavnik:
dr Marko Vještica

DAIS
Data Science and Information Systems Group

Sadržaj

Aplikacija zasnovana na monolitnoj arhitekturi



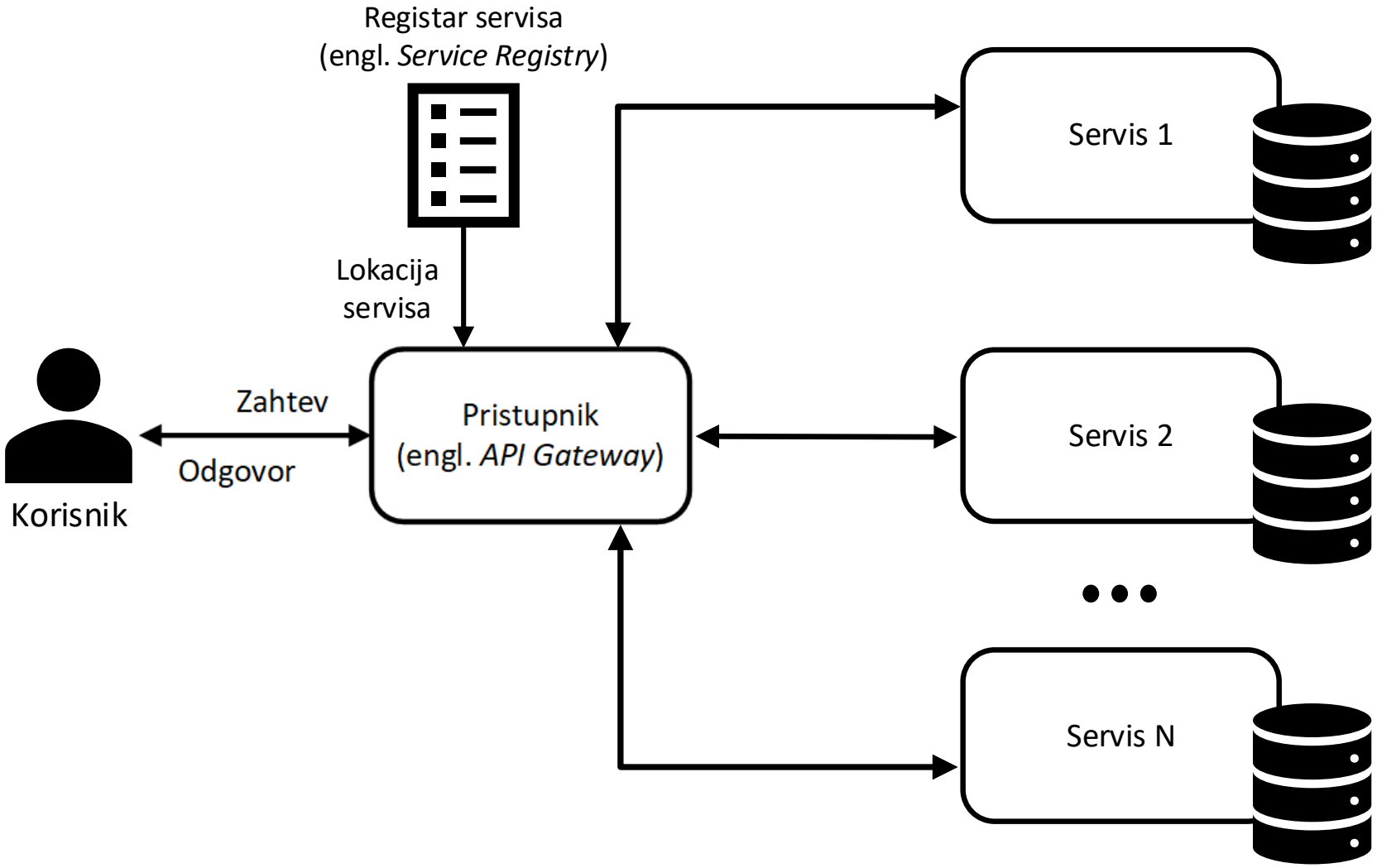
Aplikacija zasnovana na monolitnoj arhitekturi

- **Monolitna arhitektura:**
  - Projektovanje i razvoj softverskog sistema kao **jedne samostalne celine**
  - Mogu biti obimne i kompleksne aplikacije čiji razvoj i održavanje **vodi velik broj inženjera**
- **Monolitna web aplikacija:**
  - Sadrži **tri osnovna sloja:**
    - **Sloj korisničkog interfejsa** – deo softverskog rešenja kojem **korisnik ima direktan pristup**, najčešće pomoću Internet pretraživača
    - **Sloj poslovne logike** – deo softverskog rešenja za **izvršavanje poslovne logike**; pruža pristupne tačke (engl. *Application Programming Interface – API*) pomoću kojih se iniciraju odgovarajuće aktivnosti posredstvom sloja korisničkog interfejsa
    - **Sloj pristupa podacima** – deo softverskog rešenja za potrebe **ažuriranja baze podataka i postavljanja upita**

Aplikacija zasnovana na monolitnoj arhitekturi

- Prednosti:
  - **Razvoj** softverskog rešenja (uglavnom) pomoću jednog programskog jezika
  - **Testiranje** softverskog rešenja
  - **Pronalaženje i otklanjanje grešaka** (engl. *Debugging*) u programskom kôdu
  - **Isporuka** (engl. *Deployment*) softverskog rešenja
- Nedostaci:
  - Razvoj softverskog rešenja (uglavnom) pomoću **jednog programskog jezika**
  - Održavanje **obimnog i kompleksnog** softverskog sistema
    - Potencijalno **velik broj timova i njihovih članova** za razvoj i održavanje sistema
    - Bilo kakva izmena sistema zahteva **kompajliranje i isporuku kompletnog softverskog rešenja**
    - **Izmene u jednom modulu** mogu uticati na druge module
  - Pojava greške u jednom modulu može **uticati na druge module**
  - **Skalabilnost** za potrebe velikog broja korisnika i njihovih zahteva
    - Posebno ukoliko je potrebno samo **pojedine module** skalirati, neophodno je skalirati celo rešenje

Aplikacija zasnovana na mikroservisnoj arhitekturi



Aplikacija zasnovana na mikroservisnoj arhitekturi

- **Mikroservisna arhitektura:**
  - Projektovanje i razvoj softverskog sistema kao **skup nezavisnih servisa – mikroservisa**
  - Mikroservise razvijaju **timovi sa malim brojem članova**
  - **Dekomponovanje** jednog velikog sistema na male delove
- **Odlike mikroservisa:**
  - Relativno **malog obima**
  - Implementirani sa ciljem rešavanja **jednog zadatka ili skupa srodnih zadataka**
  - **Nezavisni** od ostalih mikroservisa
  - **Asinhrona komunikacija**
  - Mogu biti pokrenuti na **zasebnim serverima**

Aplikacija zasnovana na mikroservisnoj arhitekturi

- **Mikroservisno softversko rešenje** sadrži najmanje:
  - **Registar servisa** (engl. *Service Registry*)
    - Sadrži **informacije o instancama mikroservisa**, poput adrese, porta i statusa
    - Može periodično da **prima signal od mikroservisa**, ažurirajući status o živosti mikroservisa
    - Pokretanjem nove instance, **mikroservis mora najpre da se registruje**
  - **Pristupnik** (engl. *API Gateway*)
    - Obezbeđuje **jednu pristupnu tačku** za krajnjeg korisnika
    - **Dinamički rutira zahteve** korisnika odgovarajućim mikroservisima
    - Može da sadrži dodatne komponente i funkcionalnosti poput **ravnomernog raspoređivanja opterećenja** (engl. *Load Balancer*) **i postizanja tolerantnosti na kašnjenja i ispade** (engl. *Fault Tolerance*)
  - **Mikroservise organizacionog sistema**
    - Funkcionalne celine implementirane za potrebe **izvršavanja određenih zadataka u okviru domena**
    - Moguće je pokrenuti i održavati **više instanci pojedinačnih mikroservisa**
    - Moguće je implementirati mikroservise pomoću **različitih tehnologija i programskih jezika**

Aplikacija zasnovana na mikroservisnoj arhitekturi

- Prednosti:
  - Moguća **primena različitih tehnologija i programskih jezika** u različitim mikroservisima
  - **Izmena** jednog mikroservisa **ne utiče direktno** na druge mikroservise
    - Moguće je **potpuno zameniti ili ukloniti** neki od mikroservisa
    - Moguće je **promeniti tehnologiju ili programski jezik**
  - **Pojava greške** u jednom mikroservisu **ne utiče direktno** na druge mikroservise
    - Otkaz jednog mikroservisa **ne podrazumeva otkaz čitave aplikacije**
  - Mogućnost **skaliranja pojedinačnih mikroservisa**
  - Pojedinačni mikroservisi obuhvataju **relativno mali broj funkcionalnosti**
  - Timovi sa **malim brojem članova** razvijaju i održavaju pojedinačne mikroservise
- Nedostaci:
  - **Kompleksna arhitektura** sistema koja zahteva dodatne komponente
  - **Pronalaženje i uklanjanje grešaka** u softverskom rešenju nekada zahteva analizu više mikroservisa
  - **Očuvanje konzistentnosti podataka** u različitim mikroservisima
  - Uspešno **testiranje** jednog mikroservisa ne znači da je celo softversko rešenje korektno

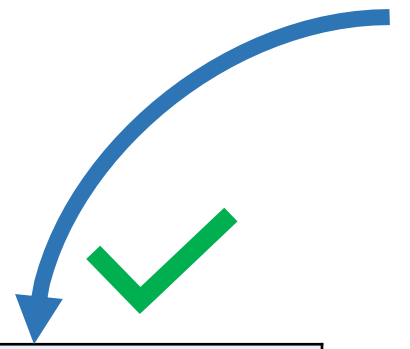
Monolitna ili mikroservisna arhitektura

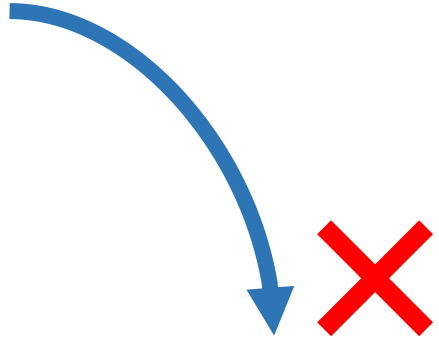
Monolitna arhitektura	Mikroservisna arhitektura
Jednostavnija arhitektura	**Kompleksnija** arhitektura
Jedna samostalna celina	**Više malih nezavisnih** servisa
Dekomponovanje po **modulima**	Dekomponovanje po **mikroservisima**
Jedan ili više timova sa **velikim brojem članova**	Više timova sa **malim brojem članova**
Potencijalno **obimne i kompleksne aplikacije**	**Više manjih i jednostavnijih** delova aplikacije
Upotreba **jedne tehnologije i programskog jezika**	Upotreba **različitih tehnologija i programskih jezika**
Skalabilnost **cele aplikacije**	Skalabilnost **pojedinačnih mikroservisa**

Sadržaj

Transakciona obrada podataka

```
UPDATE RadProj SET brc = brc * 0.5;
UPDATE Radnik SET plt = plt * 0.5;
```





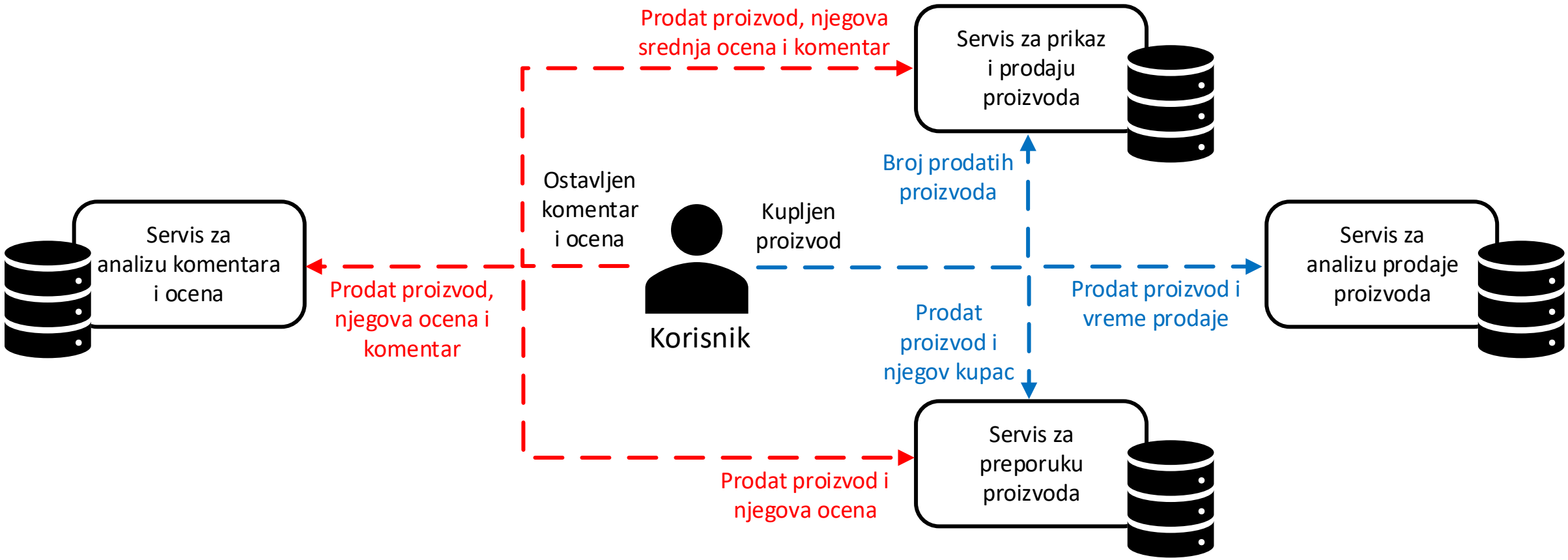
```
ALTER TABLE Radnik ADD CONSTRAINT ch_plt CHECK (plt > 5000);
```

RadProj		
Mbr	**Spr**	**Brc**
10	20	14
10	30	10
40	20	12
60	40	6
60	20	4
70	10	10

Nekonzistentni podaci

Radnik				
Mbr	**Ime**	**Prz**	**God**	**Plt**
10	Pera	Peric	01-01-1987	40000.00
40	Moma	Momic	01-06-1933	8000.00
60	Mira	Miric	01-07-1945	20000.00
70	Savo	Oroz	01-01-1933	35000.00

Transakciona obrada podataka u mikroservisima

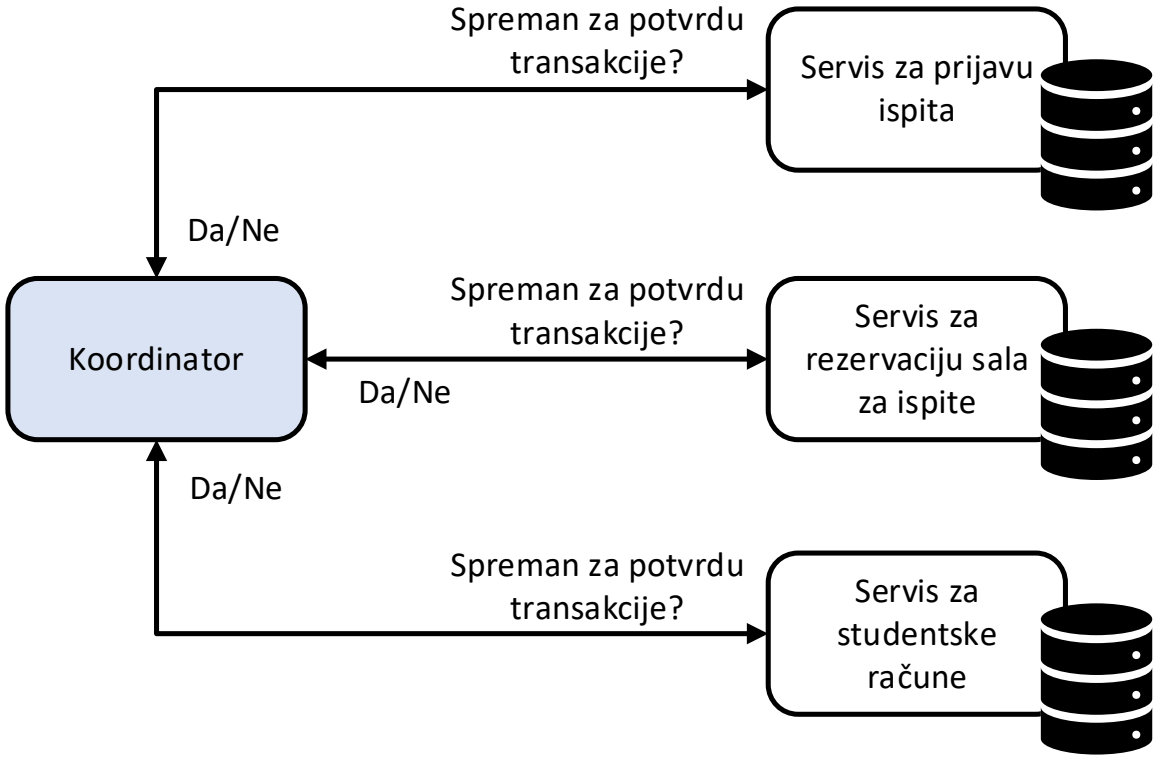


Transakciona obrada podataka u mikroservisima

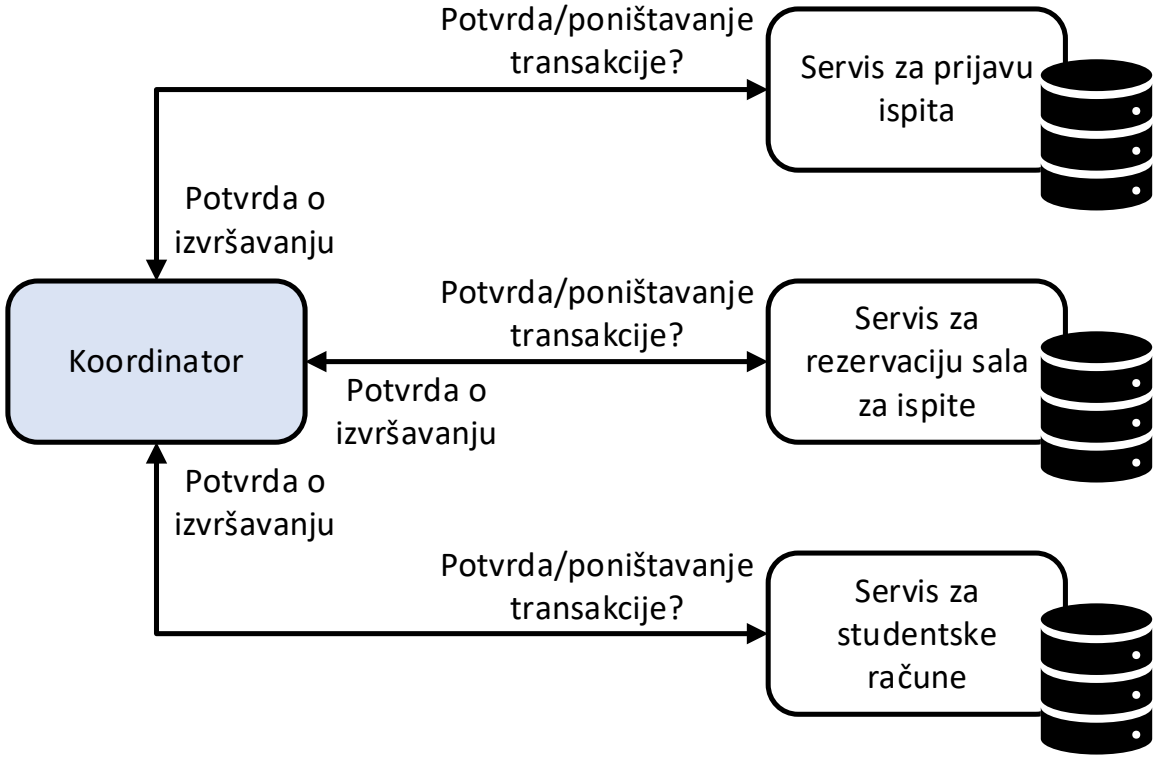
- Softversko rešenje zasnovano na mikroservisnoj arhitekturi predstavlja **distribuirani sistem**
- Jedan od izazova distribuiranih sistema predstavlja **očuvanje konzistentnosti podataka**
- Potrebno je **upravljati transakcijom** na nivou više različitih mikroservisa
  - Svaki mikroservis mora **uspešno da izvrši svoju lokalnu transakciju**
  - Ukoliko barem jedan od mikroservisa **ne izvrši transakciju**, prethodno izvršene lokalne transakcije **moraju biti poništene**
- Jedno od rešenja – **potvrda transakcije u dve faze** (engl. *Two-phase commit*)

Potvrda transakcije u dve faze

Faza 1: Priprema transakcije



Faza 2: Izvršavanje transakcije



Potvrda transakcije u dve faze

- **Koordinator** je softverska komponenta zadužena za **upravljanje transakcijama u distribuiranim sistemima**, što podrazumeva i mikroservisnu arhitekturu
- **Faza 1: Priprema transakcije**
  - Koordinator šalje svim učesnicima **upit da li mogu da garantuju potvrdu lokalnih transakcija**
  - Svi **učesnici šalju odgovor koordinatoru** da li mogu da potvrde svoju lokalnu transakciju
- **Faza 2: Izvršavanje transakcije**
  - **Opcija 1:** Ukoliko su **svi učesnici** odgovorili da mogu uspešno da izvrše transakcije, koordinator šalje **potvrdu (engl. *commit*) transakcije** svim učesnicima
  - **Opcija 2:** Ukoliko **barem jedan učesnik** odgovori da ne može uspešno da izvrši transakciju, koordinator šalje **poništavanje (engl. *rollback*) transakcije** svim učesnicima
  - Nakon što koordinator pošalje potvrdu ili poništavanje transakcije, učesnici vraćaju **potvrdu o izvršenom zadatku**

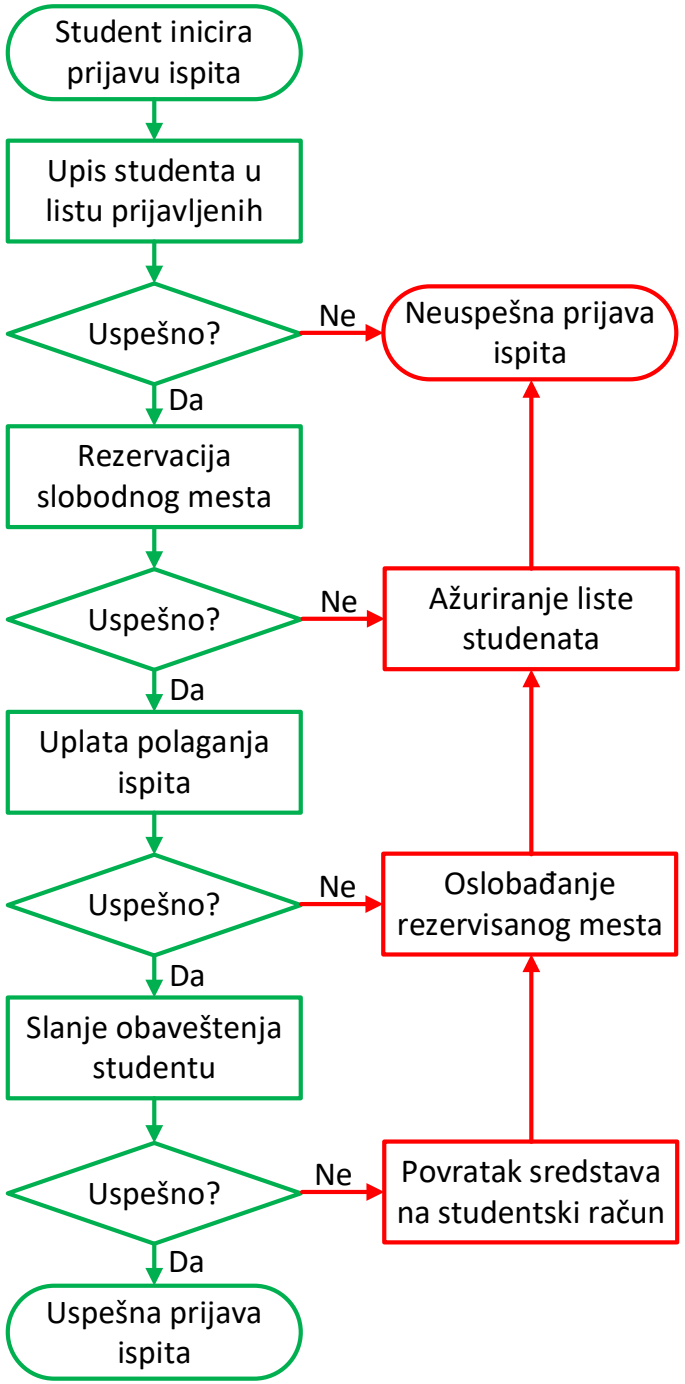
Potvrda transakcije u dve faze

- Nedostaci:
  - Potvrda transakcije u dve faze većinom **nije podržana u NoSQL** sistemima za upravljanje bazama podataka
  - Upravljanje transakcijama obavlja se **posredstvom koordinatora**
    - **Razmena poruka tokom izvršavanja transakcija** odvija se pomoću koordinatora
    - Potencijalno **lošije performanse** rada softverskog rešenja jer **koordinator postaje centralna tačka komunikacije**
    - Problem **skalabilnosti softverskog rešenja** jer je upravljanje transakcija centralizovano
  - **Brzina izvršavanja** transakcije zavisi od najsporijeg servisa
    - Svi servisi moraju da sačekaju da stigne potvrda **najsporijeg servisa**

Šablon arhitekture Saga

- **Šablon arhitekture Saga** omogućava **upravljanje transakcijama** koristeći sekvencu lokalnih transakcija i po potrebi sekvencu kompenzacionih transakcija
- **Lokalna transakcija** predstavlja aktivnost jednog učesnika arhitekture Saga (npr. mikroservisa)
- Svaka lokalna transakcija može biti poništena upotrebom **kompenzacione transakcije**
  - Kompenzaciona transakcija može da se **izvrši više puta i daje uvek isti rezultat**
- Arhitektura Saga obezbeđuje da se ili sve lokalne transakcije izvrše u potpunosti ili da se pokretanjem odgovarajućih kompenzacionih transakcija podaci vrate na **prethodno konzistentno stanje**


Servis za prijavu ispita
Servis za rezervaciju sala za ispite
Servis za studentske račune


Student inicira prijavu ispita
Upis studenta u listu prijavljenih
Uspešno?
Ne
Neuspešna prijava ispita
Da
Rezervacija slobodnog mesta
Uspešno?
Ne
Ažuriranje liste studenata
Da
Uplata polaganja ispita
Uspešno?
Ne
Oslobađanje rezervisanog mesta
Da
Slanje obaveštenja studentu
Uspešno?
Ne
Povratak sredstava na studentski račun
Da
Uspešna prijava ispita

Šablon arhitekture Saga

- **Koordinator izvršavanja Saga** predstavlja centralnu komponentu koja prikuplja **sekvencu događaja** nastalih tokom izvršavanja transakcije u okviru distribuiranog sistema
- Svi događaji čuvaju se u **dnevniku Saga**
  - **Događaj** predstavlja uspešnu ili neuspešnu lokalnu transakciju
  - Koordinator ima uvid u **izvršene transakcije, poništene transakcije i transakcije koje čekaju** izvršenje/poništenje
- **Ukoliko lokalna transakcija nije uspešno izvršena**, koordinator pristupa dnevniku i **identifikuje servis** u okviru kojeg transakcija nije izvršena, kao i **sekvencu kompenzacionih transakcija**
- U slučaju **otkazivanja koordinatora**, nakon što je ponovo pokrenut, može da **pročita dnevnik i nastavi sa prethodnim aktivnostima**

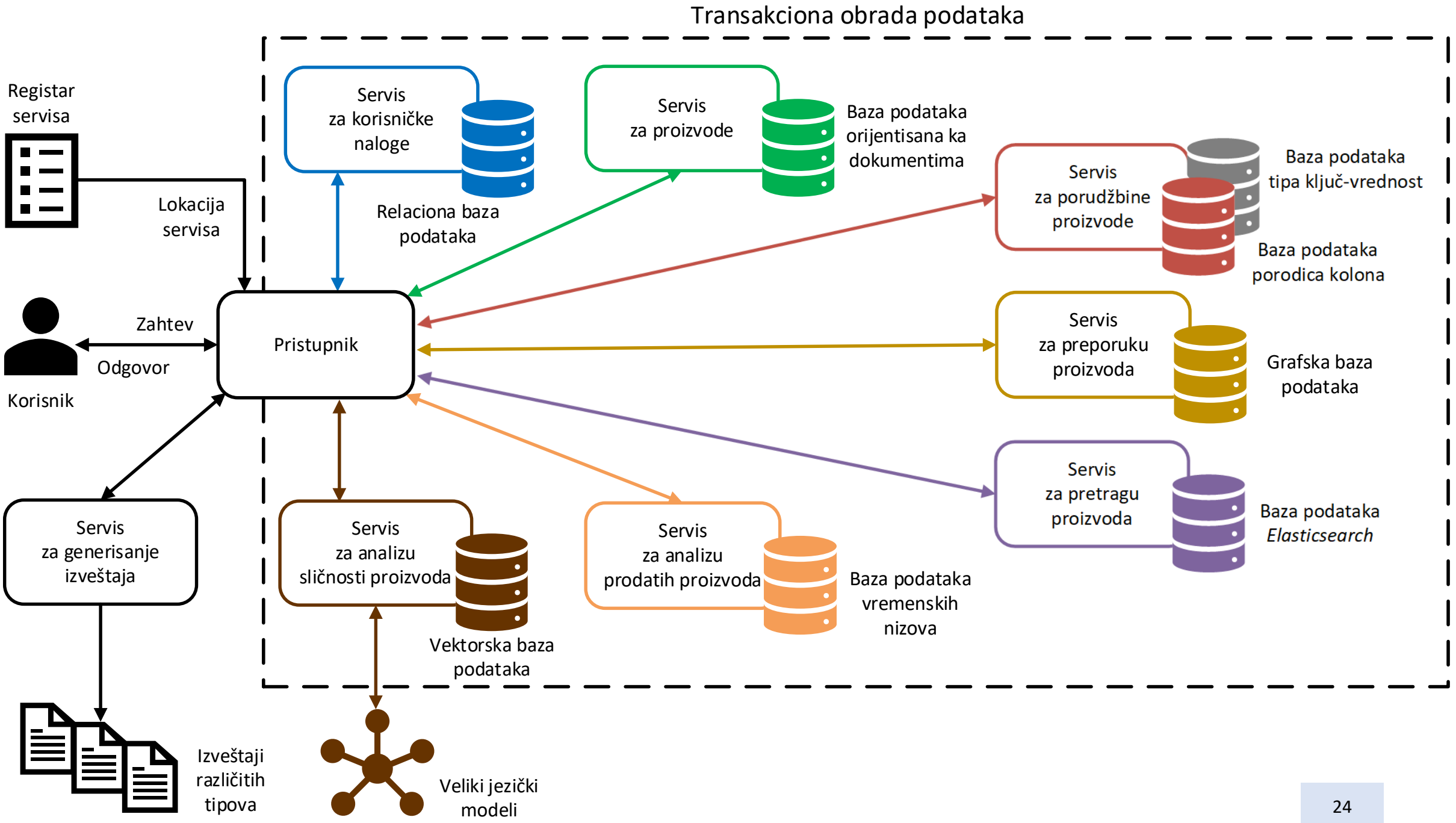
Šablon arhitekture Saga

- Dva pristupa implementacije šablona arhitekture Saga: **koreografija i orkestracija**
- Šablon Saga **orkestracije**:
  - **Centralni koordinator izvršavanja – orkestrator** (npr. *Camunda*, ili poseban servis)
    - Zadužen za **upravljanje redosledom izvršavanja lokalnih i kompenzacionih transakcija**
    - U slučaju narušavanja transakcije, **orkestrator je zadužen** za pokretanje odgovarajućih kompenzacionih aktivnosti
  - Pogodan kada postoje **postojeća mikroservisna rešenja**
    - **Nije potrebno menjati** postojeće mikroservise, već je poseban mikroservis zadužen za upravljanje transakcijama
  - Pogodan kada je potrebna **centralizovana kontrola** nad transakcijama

Šablon arhitekture Saga

- Dva pristupa implementacije šablona arhitekture Saga: **koreografija i orkestracija**
- Šablon Saga **koreografije**:
  - Ne postoji orkestrator, već **mikroservisi reaguju na događaje**
    - Mikroservis obuhvaćen transakcijom **objavljuje događaj** koji procesira naredni mikroservis u transakciji
    - Koordinator je **ugrađen u mikroservisima**
  - U slučaju narušavanja transakcije, **mikroservis prijavljuje grešku koordinatoru** koji poziva odgovarajuće kompenzacione transakcije sve dok se ne izvrše
  - Pogodan prilikom kreiranja **novih mikroservisnih rešenja**
  - Pogodan usled **jednostavnog skaliranja** mikroservisa, a samim tim i koordinatora

Sadržaj


Transakciona obrada podataka
Registar servisa
Lokacija servisa
Servis za korisničke naloge
Relaciona baza podataka
Servis za proizvode
Baza podataka orijentisana ka dokumentima
Servis za porudžbine proizvode
Baza podataka tipa ključ-vrednost
Baza podataka porodica kolona
Zahtev
Odgovor
Korisnik
Pristupnik
Servis za preporuku proizvoda
Grafska baza podataka
Servis za pretragu proizvoda
Baza podataka Elasticsearch
Servis za generisanje izveštaja
Servis za analizu sličnosti proizvoda
Vektorska baza podataka
Servis za analizu prodatih proizvoda
Baza podataka vremenskih nizova
Izveštaji različitih tipova
Veliki jezički modeli

Mikroservisi informacionog sistema *online* prodavnice

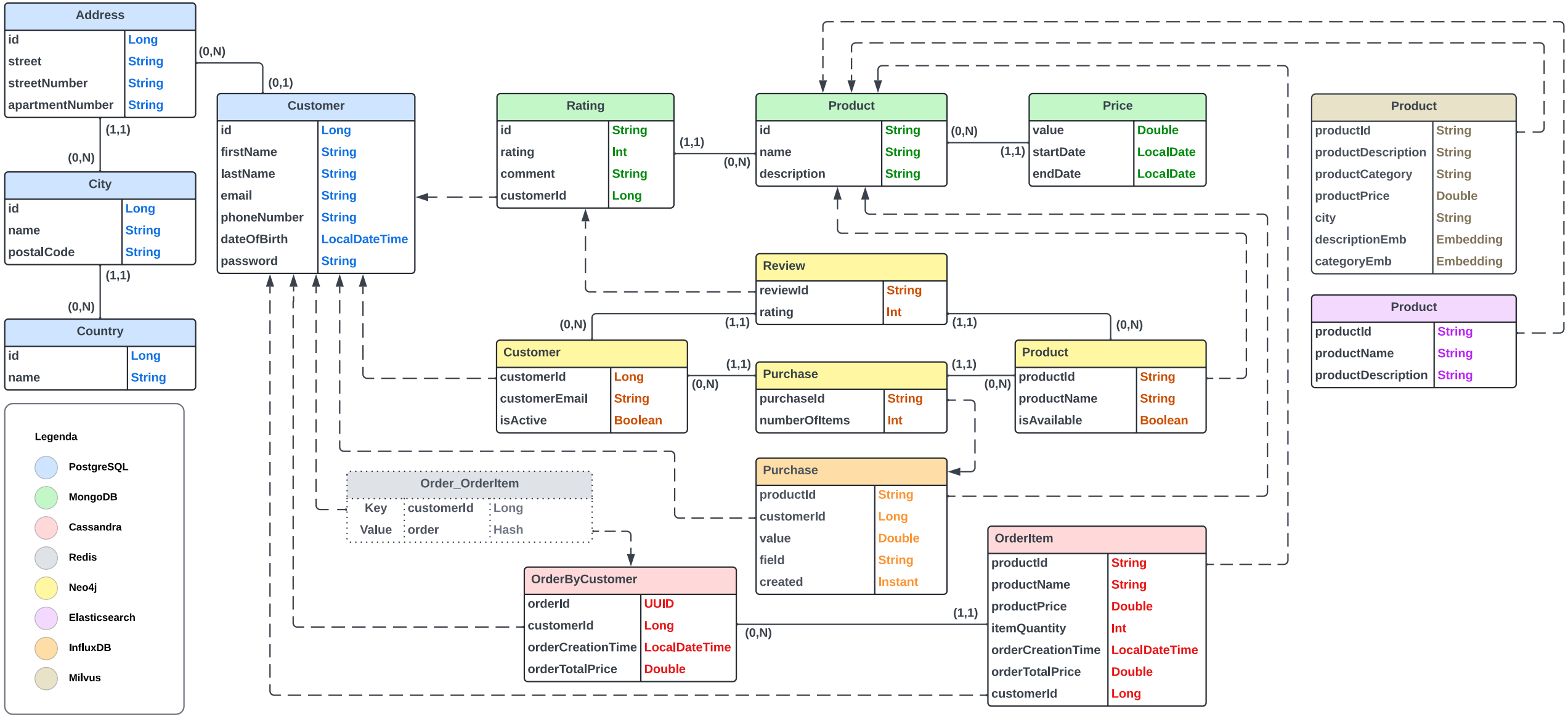
- Mikroservis za **korisničke naloge**
  - Koristi **relacionu bazu podataka** (*PostgreSQL*)
    - Čuva podatke o korisnicima i njihovim adresama
    - Postoji jasno definisana struktura podataka
- Mikroservis za **proizvode**
  - Koristi **bazu podataka orijentisanu ka dokumentima** (*MongoDB*)
    - Čuva podatke o proizvodima, njihovim cenama i recenzijama od strane korisnika
    - Ne moraju imati jasno definisanu strukturu podataka

Mikroservisi informacionog sistema *online* prodavnice

- Mikroservis za **porudžbine proizvoda**
  - Koristi **bazu podataka porodica kolona** (*Apache Cassandra*)
    - Čuva podatke o porudžbinama i stavkama porudžbina
    - Efikasno izvršavanje upita nad velikom količinom podataka, posebno analize podataka
  - Koristi **bazu podataka tipa ključ-vrednost** (*Redis*)
    - Čuva keširane podatke o porudžbinama ili korisničkim korpama
    - Dodatno poboljšanje performansi prilikom izvršavanja upita
- Mikroservis za **preporuku proizvoda**
  - Koristi **grafsku bazu podataka** (*Neo4j*)
    - Čuva podatke o korisnicima i kupljenim proizvodima
    - Služi da omogući naprednu preporuku proizvoda korisnicima na osnovu prethodno kupljenih proizvoda, ocena dodeljenih proizvodima i sličnosti između proizvoda

Mikroservisi informacionog sistema *online* prodavnice

- Mikroservis za **pretragu proizvoda**
  - Koristi **pogon za pretragu i analizu podataka** (*Elasticsearch*)
    - Čuva podatke o proizvodima (nazivima i opisima) i komentare na proizvode koje su korisnici ostavili
    - Mogućnost efikasne pretrage proizvoda i komentara na osnovu teksta
- Mikroservis za **analizu prodatih proizvoda**
  - Koristi **bazu podataka vremenskih nizova** (*InfluxDB*)
    - Čuva podatke o prodatim proizvodima i vremenskim odrednicama
    - Omogućava analizu događaja, odnosno logova
- Mikroservis za **analizu sličnosti proizvoda**
  - Koristi **vektorsku bazu podataka** (*Milvus*)
    - Čuva podatke o opisima proizvoda i njihovim fotografijama
    - Mogućnost poređenja fotografija, opisa i povezivanja servisa sa velikim jezičkim modelima (engl. *Large Language Models*), omogućavajući komunikaciju sa servisom pomoću prirodnog jezika


Address
id Long
street String
streetNumber String
apartmentNumber String
(0,N)
(0,1)
(1,1)
(0,N)
City
id Long
name String
postalCode String
(1,1)
(0,N)
Country
id Long
name String
Customer
id Long
firstName String
lastName String
email String
phoneNumber String
dateOfBirth LocalDateTime
password String
Rating
id String
rating Int
comment String
customerId Long
(1,1)
(0,N)
Product
id String
name String
description String
(0,N)
(1,1)
Price
value Double
startDate LocalDate
endDate LocalDate
Product
productId String
productDescription String
productCategory String
productPrice Double
city String
descriptionEmb Embedding
categoryEmb Embedding
Product
productId String
productName String
productDescription String
Review
reviewId String
rating Int
(0,N)
(1,1)
(1,1)
(0,N)
Customer
customerId Long
customerEmail String
isActive Boolean
(1,1)
(0,N)
Purchase
purchaseId String
numberOfItems Int
(1,1)
(0,N)
Product
productId String
productName String
isAvailable Boolean
Order_OrderItem
Key customerId Long
Value order Hash
Purchase
productId String
customerId Long
value Double
field String
created Instant
OrderByCustomer
orderId UUID
customerId Long
orderCreationTime LocalDateTime
orderTotalPrice Double
(1,1)
(0,N)
OrderItem
productId String
productName String
productPrice Double
itemQuantity Int
orderCreationTime LocalDateTime
orderTotalPrice Double
customerId Long
Legenda
PostgreSQL
MongoDB
Cassandra
Redis
Neo4j
Elasticsearch
InfluxDB
Milvus

Predmetni projekat

- Zašto domen ***online* prodavnice**?
- Kako su **kreirani podaci** za potrebe projekta?
  - Upotreba javno dostupnih podataka?
    - *Kaggle*: https://www.kaggle.com
    - *data.world*: https://data.world
    - *AWS registry*: https://registry.opendata.aws
    - *UCI ML Repository*: https://archive.ics.uci.edu
  - „Ručno“ kreiranje podataka?
  - Primena modela veštačke inteligencije?
- Kako **pokrenuti projekat** na različitim računarima i sistemima?

Sadržaj

Problem prenosivosti softverskog rešenja

- Razvoj softverskog rešenja na **jednom računaru** podrazumeva da programski kôd može da se **pokrene bez problema**
  - Prenos istog softverskog rešenja (istog programskog kôda) **na drugi računar:**
    - Potencijalna **nemogućnost pokretanja** rešenja ili nefunkcionisanje pojedinih delova rešenja
    - Predstavlja **problem prenosivosti** (engl. *Portability*) **softverskog rešenja**
  - **Primeri:**
    - Pokretanje softverskog rešenja na računaru kod novog člana razvojnog tima
    - Isporuka i pokretanje softverskog rešenja na serveru kompanije
- Potencijalni **uzroci** nastalog problema:
  - **Različiti operativni sistemi ili njihove verzije** na računarima
  - **Instaliran softver koji nije kompatibilan** sa softverskim rešenjem
  - **Različite verzije interpretera** programskog jezika na računarima

Problem prenosivosti softverskog rešenja

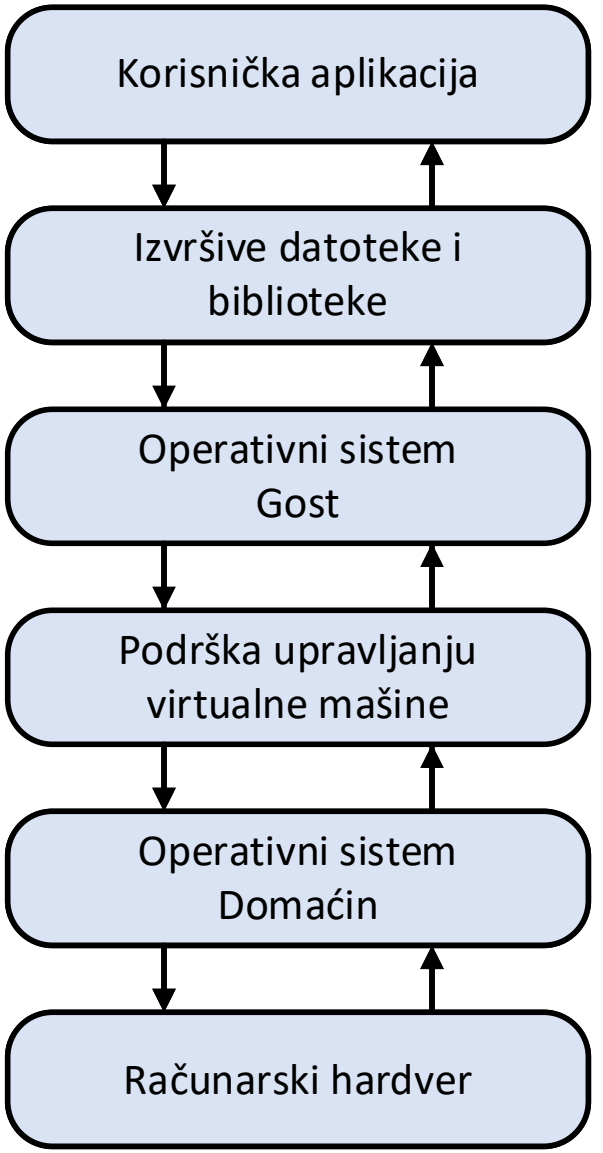
- **Problem prenosivosti** softverskog rešenja:
  - **Zahteva pronalaženje i uklanjanje grešaka** (engl. *Debugging*) specifičnih za radno okruženje ili operativni sistem
  - Može biti veoma **vremenski zahtevan zadatak**
- **Potencijalna rešenja** problema prenosivosti softverskog rešenja:
  - Primena **virtualnih mašina** (**virtualizacija na nivou hardvera**)
  - Primena **kontejnera** (**virtualizacija na nivou operativnog sistema**)

Virtualne mašine

- **Problem prenosivosti softverskog rešenja** bilo bi moguće razrešiti time što bi svaki računar sadržao **isti operativni sistem, instaliran softver, alate i biblioteke**
  - **Veoma teško** sprovesti u praksi
- Druga opcija: primena **virtualnih mašina**, odnosno **virtualizacija na nivou hardvera**
  - Mogućnost upotrebe **više operativnih sistema na jednom računaru**
- **Virtualna mašina** predstavlja **digitalnu kopiju računarskog hardvera** na kojem je pokrenut **operativni sistem gost**
- **Primeri** softverskih rešenja za podršku rada virtualnih mašina:
  - *Virtual box*
  - *VMWare*

Virtualne mašine

- **Izvršavanje** korisničke aplikacije na **računaru**:
  - Korisnička aplikacija upravlja hardverom posredstvom **jezgra operativnog sistema**, koristeći **izvršive datoteke i biblioteke**
- **Izvršavanje** korisničke aplikacije na **virtualnoj mašini**:
  - **Operativni sistem domaćin** (engl. *Host Operating System*) predstavlja operativni sistemi instaliran i pokrenut na računaru koji posredstvom jezgra upravlja **računarskim hardverom**
  - Softverska komponenta **podrške upravljanju virtualne mašine** (engl. *Virtual Machine Manager* ili *Hypervisor*) zadužena je za rad virtualne mašine na kojoj je pokrenut **operativni sistem gost** (engl. *Guest Operating System*), a koji upravlja virtualnim hardverom
  - **Korisnička aplikacija** upravlja računarskim hardverom posredstvom operativnog sistema gosta, podrške upravljanju virtualne mašine i operativnog sistema domaćina, koristeći **izvršive datoteke i biblioteke**



Virtualne mašine

- **Podrška upravljanju virtualne mašine** pokrenuta je na operativnom sistemu domaćinu
  - Zadužena je za kreiranje **virtualnih instanci hardvera**
  - Njenom upotrebom, korisnik može pokrenuti **operativne sisteme goste**
    - Prethodno mora definisati **potrebno zauzeće resursa računara** za svaki operativni sistem gost
- Za razliku od operativnog sistema domaćina, operativni sistem gost **ne upravlja direktno hardverom**
  - Već **upravlja virtualnim hardverom** koji je kreirala softverska komponenta podrške upravljanju virtualne mašine
  - Operativni sistem gost **ne prepoznaje virtualni hardver kao takav**, nego ga posmatra kao realni računarski hardver

Virtualne mašine

- **Prednosti** primene virtualne mašine:
  - Moguće je pokrenuti **više operativnih sistema** na jednom računaru
  - Svaki operativni sistem gost je **izolovan** od ostalih operativnih sistema
  - Moguće je na računaru pokrenuti **različite korisničke aplikacije** razvijane u **raznim okruženjima** i pokrenute na **raznim operativnim sistemima**
- **Nedostaci** primene virtualne mašine:
  - Svaka virtualna mašina ima svoj operativni sistem koji **zauzima prostor masovne memorije**
  - Svaka pokrenuta virtualna mašina značajno **opterećuje resurse računara**
    - Pored korisničke aplikacije koja je pokrenuta, takođe je **pokrenut i operativni sistem**
  - Povećavaju **kompleksnost** celokupnog sistema održavanjem više operativnih sistema

Kontejneri

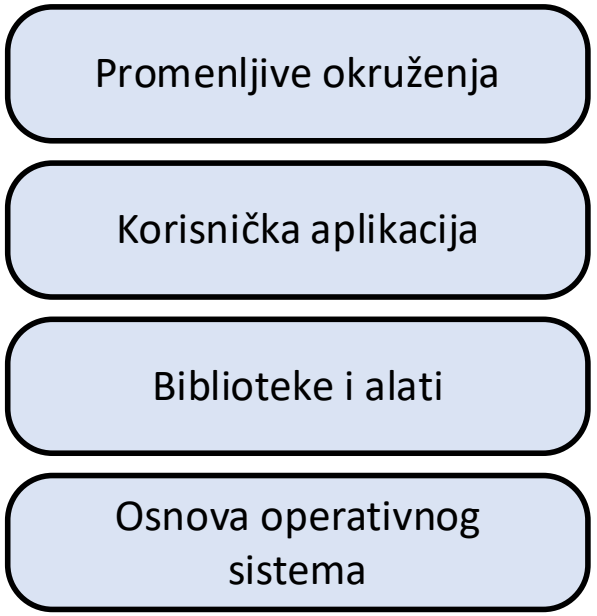
- Kako bi bili prevaziđeni nedostaci virtualizacije na nivou hardvera, uvedena je **virtualizacija na nivou operativnog sistema**, odnosno **primena kontejnera**
- **Kontejner** predstavlja **izolovan proces** koji sadrži aplikaciju koju je potrebno pokrenuti u okviru operativnog sistema
- Za razliku od virtualnih mašina, kontejneri **ne zahtevaju instalaciju operativnog sistema gosta**, već su pokrenuti na nivou operativnog sistema domaćina
  - Kontejneru su **dovoljne izvršive datoteke i biblioteke** koje aplikacija koristi, zbog čega zauzima znatno **manje računarskih resursa** od virtualne mašine
    - Npr. za pokretanje Java aplikacije neophodno je Java okruženje (engl. *Java Runtime Environment*) u okviru kontejnera
- **Primeri** softverskih rešenja za podršku kontejnera:
  - *Linux Container* – prva implementacija kontejnera
  - *Docker Container* – popularna (*de facto* standard) implementacija kontejnera

Kontejneri

- Za **kreiranje kontejnera** neophodna je slika kontejnera
- **Slika kontejnera** predstavlja **samostalni i izvršivi paket**
  - **Datoteka** koja u sebi sadrži **sistem datoteka** (engl. *File System*), **okruženje, biblioteke, alate, podešavanja i konfiguracije** neophodne za **pokretanje aplikacije** na odgovarajućoj platformi
  - Obezbeđuje **isporuku aplikacije nezavisno od operativnog sistema i računarskog hardvera**
- **Kontejner** predstavlja **jednu instancu slike kontejnera**
  - **Izolovano okruženje** za pokretanje aplikacije
  - Svaki kontejner dobija **fiksnu količinu resursa** na računaru
  - Kontejneri **rešavaju problem zavisnosti biblioteka** (engl. *Dependency Problem*)
    - Time što se potrebne **biblioteke**, čije su verzije međusobno **kompatibilne**, smeštaju unutar kontejnera koji predstavljaju jednu izolovanu celinu

Kontejneri

- Svaka **slika kontejnera** može da **sadrži više slojeva**, poput:
  - **Osnove operativnog sistema** – ne sadrži kompletan operativni sistem i jezgro, već samo sistem datoteka i potrebne izvršive datoteke
  - **Biblioteka, konfiguracija i alata** – neophodnih za pokretanje korisničke aplikacije (npr. interpreter za odabrani programski jezik)
  - **Korisničke aplikacije** – koju je potrebno pokrenuti na računaru
  - **Promenljivih okruženja** (engl. *Evnironment Variables*) – promenljive čije su vrednosti postavljene izvan okvira aplikacije, odnosno na nivou sistema
- U zavisnosti od **potreba korisničke aplikacije**, mogu postojati **različiti slojevi**
- Postoji **standard *Open Container Initiative* (*OCI*)** za specifikaciju slika kontejnera

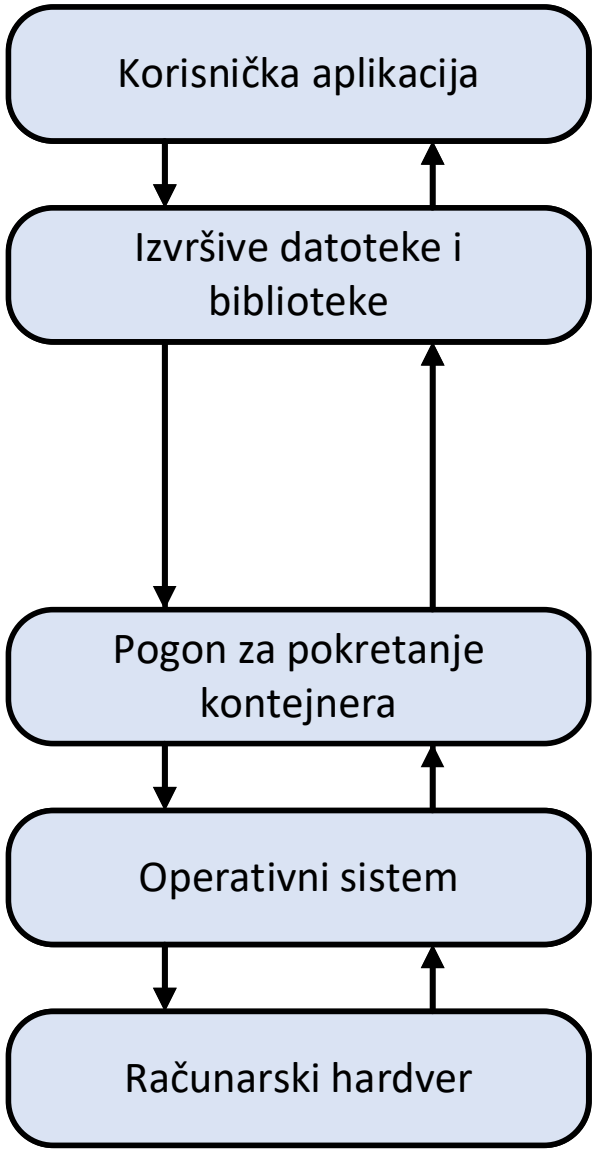


Kontejneri

- Ukoliko **isti slojevi** postoje u više kontejnera, oni mogu biti **deljeni** između kontejnera
- Pokretanjem kontejnera kreira se **virtualno okruženje**
- Unutar kontejnera vidljive su samo izvršive datoteke, biblioteke, konfiguracije, alati i promenljive koje se **nalaze u kontejneru**
- Kontejnere je moguće pokrenuti na različitim računarima **nezavisno od softverske podrške instalirane na računaru**
  - Npr. operativnog sistema, biblioteka i alata

Kontejneri

- **Izvršavanje** korisničke aplikacije u okviru **kontejnera**:
  - **Operativni sistem** instaliran i pokrenut na računaru upravlja **računarskim hardverom** posredstvom jezgra
  - **Pogon za pokretanje kontejnera** (engl. *Container Engine*) sadrži odgovarajuće pristupne tačke i zadužen je kao **veza između korisničke aplikacije i operativnog sistema**
    - **Upravlja resursima računara** neophodnih za izvršavanje aplikacije
    - **Zadužen za kreiranje kontejnera** na osnovu **slika** kontejnera
  - **Korisnička aplikacija**, uključujući **izvršive datoteke, biblioteke i konfiguracije**, koji su obuhvaćeni kontejnerom, upravlja računarskim hardverom posredstvom pogona za pokretanje kontejnera i operativnog sistema



Skaliranje softverskog rešenja

- Ukoliko je potrebno **skalirati** softversko rešenje:
  - U slučaju **virtualne mašine**, potrebno je pokrenuti **više instanci** virtualne mašine
    - **Zahtevno sa stanovišta zauzeća računarskih resursa i potrebnog vremena** prilikom pokretanja
  - U slučaju **kontejnera**, potrebno je pokrenuti **više instanci** kontejnera
    - **Ne zahteva veliko zauzeće računarskih resursa i potrebnog vremena** prilikom pokretanja
    - Čime se omogućava **jednostavnije skaliranje i poboljšanje odziva na korisničke zahteve**
    - Usled **fiksne količine računarskih resursa** po kontejneru, moguće je jednostavno povećati ili smanjiti broj instanci kontejnera i zauzeća resursa

Kontejnerizacija

- **Kontejnerizacija** predstavlja **metod isporuke i distribucije aplikacija**, pri čemu su aplikacije i njihove biblioteke **izolovane od operativnog sistema i računara** na kojem su pokrenute
  - Vrši se **pakovanje** programskog kôda aplikacije sa svim potrebnim datotekama i bibliotekama
- **Kontejnerizacija omogućava**:
  - **Isporuku** kompleksnih softverskih rešenja
  - Zauzeće **računarskih resursa** samo za potrebe korisničke aplikacije
  - **Prenosivost** softverskog rešenja time što aplikacije mogu da se pokrenu na bilo kojoj platformi ili oblaku jer su izolovane od operativnog sistema
  - **Izolaciju** pojedinačnih aplikacija ili servisa unutar softverskog rešenja, čime ih je moguće na jednostavan način zameniti ili unaprediti na noviju verziju bez narušavanja ostalih kontejnera
  - **Skalabilnost** aplikacija ili servisa pomoću kontejnera, čime je moguće razrešiti problem povećanja zahteva od strane korisnika

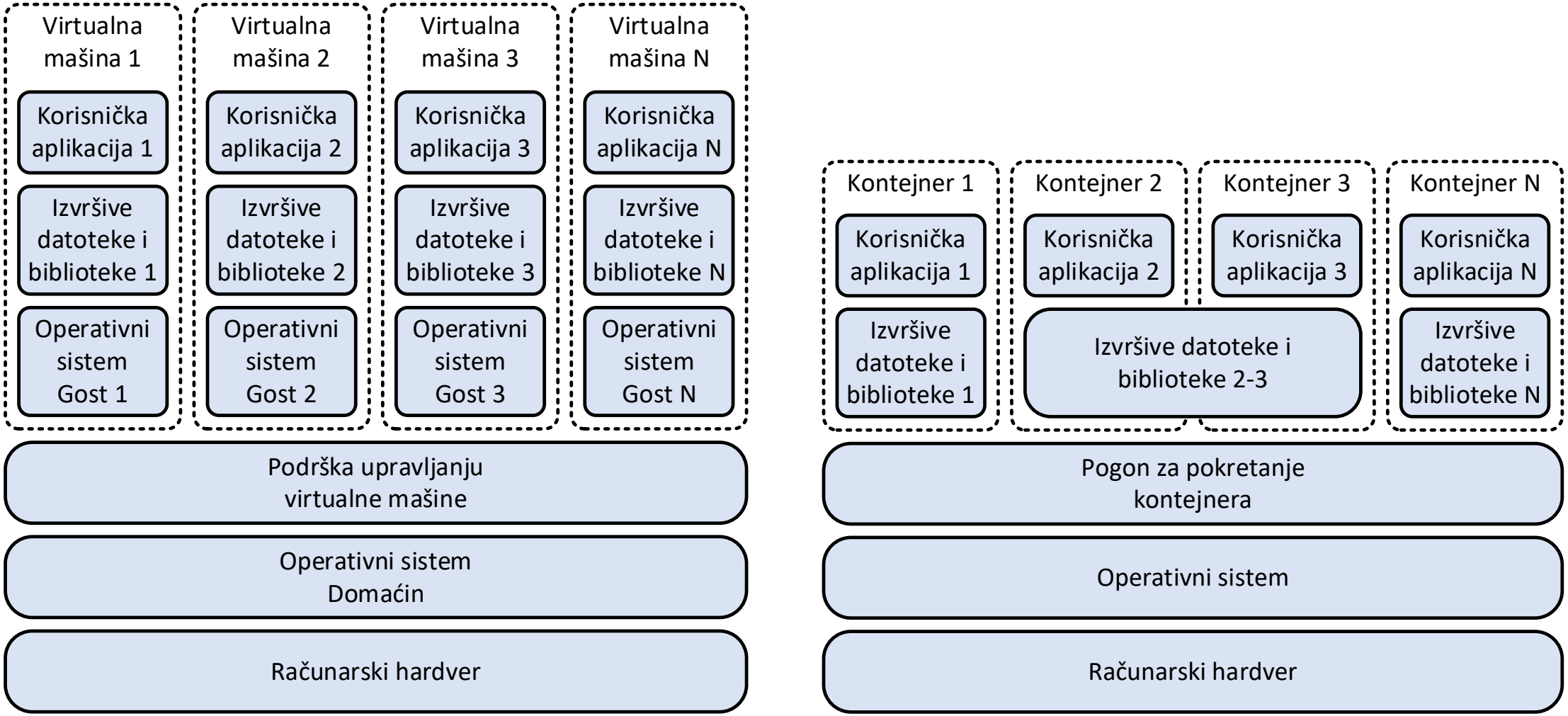
Kontejneri i mikroservisi

- Kontejneri su veoma **pogodni za mikroservisna softverska rešenja** usled:
  - **Osobine mikroservisa** da su **izolovane** celine koje mogu biti **zasebno isporučene**
  - Mogućnosti **prenosivosti** softverskog rešenja
  - Jednostavne **skalabilnosti** softverskog rešenja
  - **Razvoja** servisa u okviru jednog operativnog sistema i **pokretanje** servisa u okviru bilo kog drugog operativnog sistema
- Moguće je **mikroservis upakovati kao kontejner** i pokrenuti na drugom računaru

Orkestracija kontejnera

- Ukoliko postoji **velik broj kontejnera** koji se koriste, potrebno je odgovarajuće **upravljati** njima i povezati ih u jednu celinu, jer njihova organizacija postaje kompleksna
- **Orkestracija kontejnera** (engl. *Containter Orchestration*) zadužena je za **povezivanje kontejnera u jednu celinu**
  - **Pojednostavljuje i automatizuje** raspoređivanje, isporuku, povezivanje, skaliranje i upravljanje kontejnerima
  - Omogućuje **specifikaciju parametara i veza** između kontejnera
- Orkestracijom kontejnera moguće je obuhvatiti kontejnere koji predstavljaju **različite mikroservise u jednu celinu**
- **Primeri** orkestratora kontejnera:
  - *Kubernetes* – popularna (*de facto* standard) implementacija orkestratora kontejnera, podržana od strane *Docker*-a
  - *Docker Swarm* – implementacija orkestratora kontejnera za potrebe rada *Docker*-a

Virtualne mašine i kontejneri



Virtualne mašine i kontejneri

Virtualna mašina	Kontejner
Virtualizacija na **nivou hardvera**	Virtualizacija na **nivou operativnog sistema**
Sadrži sopstveni **operativni sistem gost**	**Ne sadrži operativni sistem**, već samo neophodne datoteke
Pokreće **razne procese operativnog sistema**	Pokreće jedino **procese neophodne za rad aplikacije**
Zauzima **više računarskih resursa** (usled potrebe rada operativnog sistema gosta)	Zauzima **manje računarskih resursa** (zauzeće resursa samo za aplikaciju koju je potrebno pokrenuti)
Sporije za pokretanje i izvršavanje	**Brže** za pokretanje i izvršavanje
Svaka virtualna mašina ima **svoje biblioteke**	Kontejneri mogu da **dele biblioteke** (retko u praksi, obično se posmatraju kao zasebni, izolovani procesi)
Relativno **teža integracija** virtualnih mašina	Relativno **jednostavnija integracija** kontejnera

Sadržaj

Literatura

- Sam Newman, Building Microservices: Designing Fine-Grained Systems, 2nd Edition, O'Reilly Media, Inc., 2021.
- Chris Richardson, Microservices Patterns: With examples in Java, 1st Edition, Manning Publications Co., 2019.

Napredne arhitekture informacionih sistema

Mikroservisi i kontejnerizacija

Pitanja?

Predmetni nastavnik:
dr Marko Vještica

DAIS
Data Science and Information Systems Group