Mikroservisi

- Koncpet
- Tehnologija mikroservisa

Tradicionalne monolitične app (1/3)

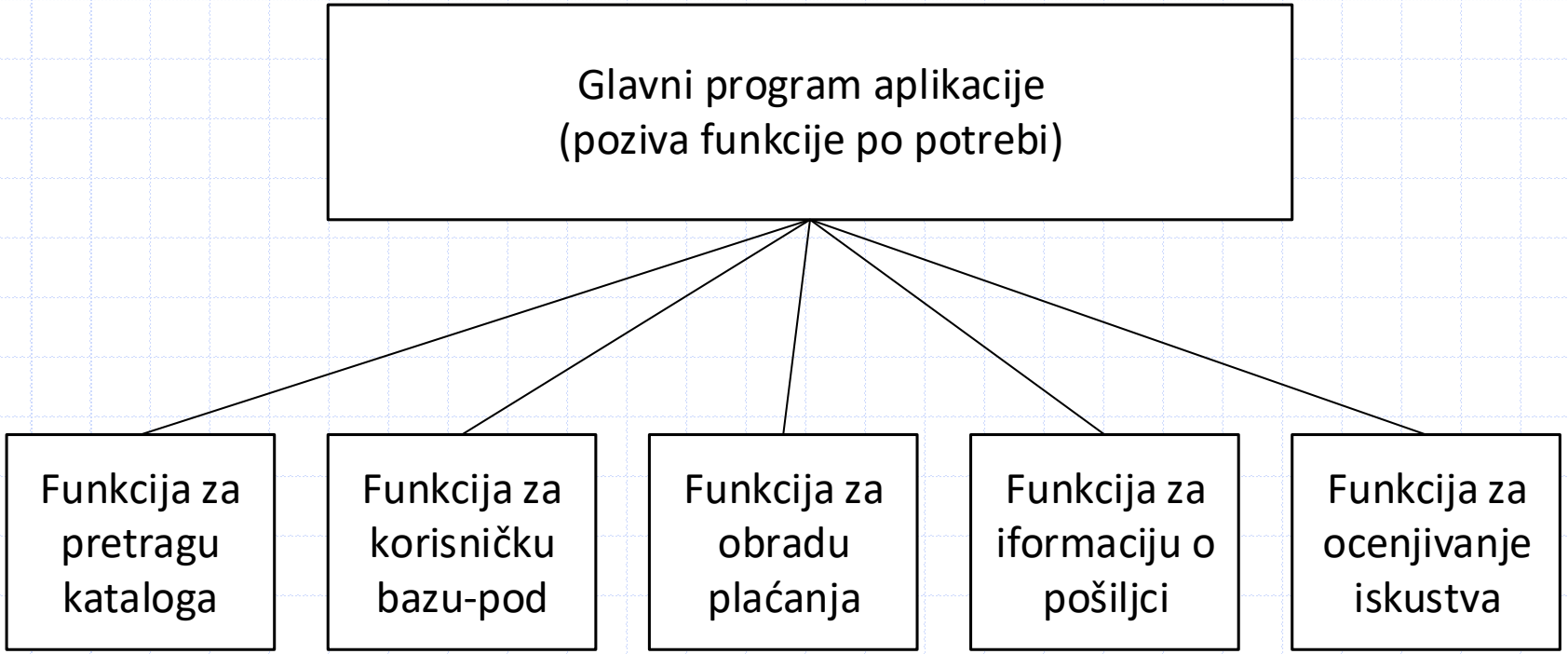
- Kako treba struktuirati app?
  - Kao jedan jedinstven prog. (monolitic), ili
  - Kao prog. podeljen u više delova koji rade zajedno
- Tradicionalno proizvođači SW su bili za I pristup
  - 1 izvršna jedinica, koju korisnik instalira i konfiguriše
- Kod SW iz više delova, postoje 2 vrste problema
  - Nepotpunost: ako korisnik ne uspe da instalra ceo SW
  - Nekopmaptibilnost: ako instalira nekomp. delove ili pređe na novije verzije samo nekih delova
  - Idealno, meni podešavanja bi trebao da obezbedi da ceo program prati korisničke izbore

Tradicionalne monolitične app (2/3)

- Def. Monolitične app su app iz jednog dela
  - Obično su velike i složene
  - Sadrže kod koji obavlja mnogo funkcija
- Primer mono. app: program za e-kupovine
  - Korisnik pristupa ovoj app iz svog web pretraživača
  - App omogućava izbor stavki, davanje info za pošiljku, plaćanje, i ocenjivanje iskustva
  - App sadrži delove za svaki od ovih koraka, plus kod za pretragu kataloga, pristup bazi-podataka, sistemu plaćanja, itd.
  - Glavni prog + skup funkcija/metoda, vidi sl. sliku

Tradicionalne monolitične app (3/3)

- Monolitična app za e-trgovinu:



Monolitične app u DC

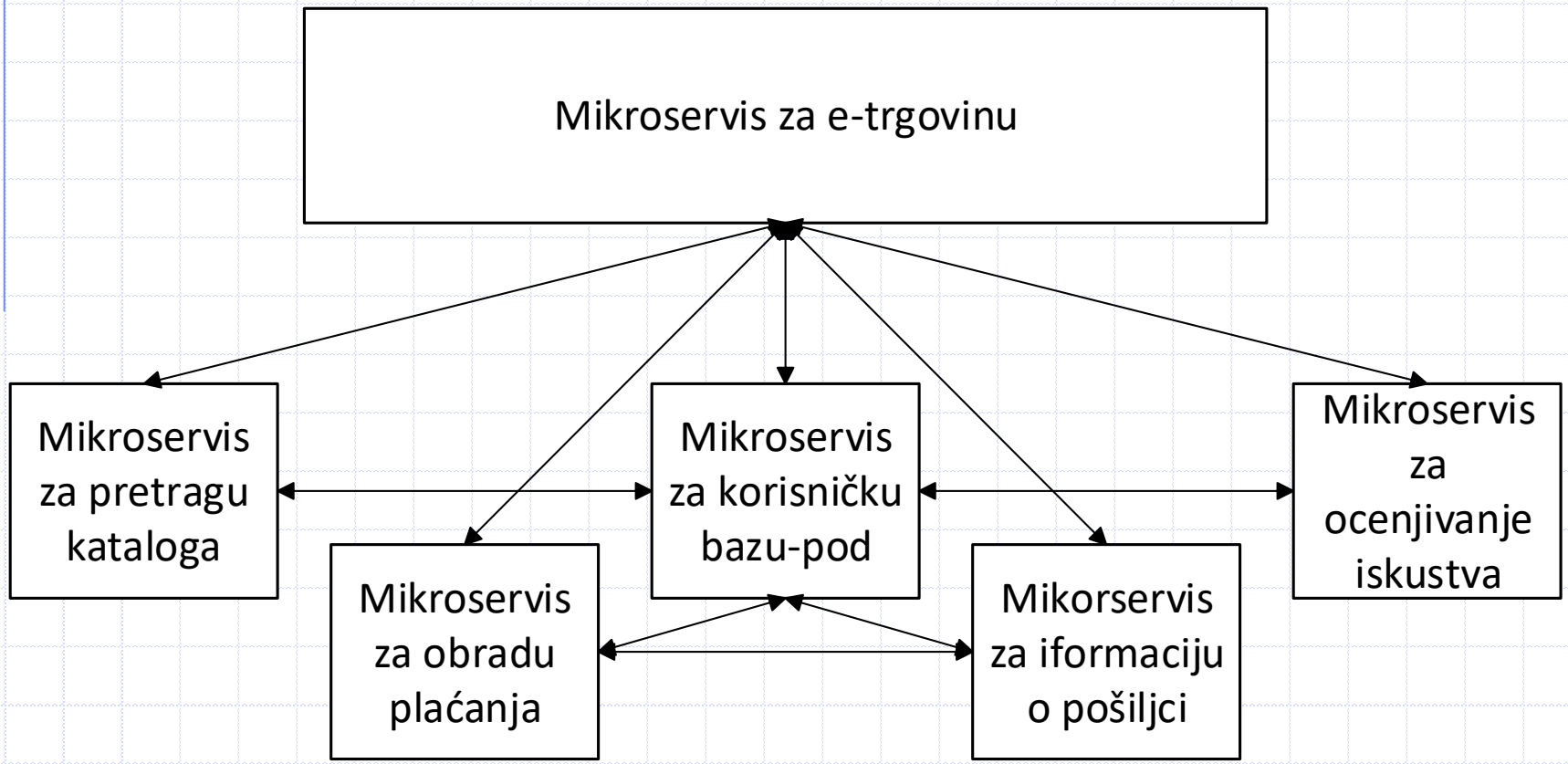
- Monolitična app može da radi u VM u DC
  - Ali je vreme replikacije duže nego za urođene app
- Vreme replikacije je duže iz 2 razloga:
  - VM ima veću režiju od kontejnera
  - Puni se cela app, iako se neki delovi ne koriste
- U dežurnom primeru:
  - Mnogi korisnici gledaju katalog bez prijave (log-in)
  - Bez kupovine, info za isporuku, i upitnika o iskustvu
  - Ti korisnici koriste samo 1 fun. a pune se sve fun.

Pristup na bazi mikroservisa (1/2)

- Arhitektura na bazi mikroservisa
  - Sis. nezavisnih apps koje komuniciraju preko mreže
  - Svaka app rukuje jednom funkcijom
- Ovaj pristup se može koristiti na 2 načina:
  - Za implementaciju novih app
  - Za podelu mono. app, tzv. disagregaciju
- Primena na dežurnom primeru
  - UI i funkcije postaju mikroservisi
  - Ilustracija na sledećem slajdu

Pristup na bazi mikroservisa (2/2)

- App za e-trgovinu na bazi mikroservisa:



Prednosti mikroservisa (1/3)

- Da li mikroservisi samo unose dodatnu režiju?
  - Ne, u okruženju oblaka donose prednosti, koje mogu da prevagnu dodatnu režiju
- Dve vrste prednosti:
  - Prednosti za razvoj SW
  - Prednosti za vođenje i održavanje sistema
- Prednosti za razvoj SW:
  - Manji vidokrug (scope) i bolja modularnost
  - Manji timovi
  - Manja složenost

Prednosti mikroservisa (2/3)

- Prednosti za razvoj SW – nastavak:
  - Izbor programskog jezika
  - Šire testiranje
- Šire testiranje:
  - Delovi mono. app interno interaguju
  - Potrebno je mnogo kombinacija ul. pod. da bi se proverile sve interakcije
  - Za najveće app moguće je samo randomizirano testiranje
  - Mikroservisi se mogu nezavisno testirati, čime je moguće šire i potpunije testiranje

Prednosti mikroservisa (3/3)

- Prednosti za vođenje i održavanje sistema:
  - Brzo raspoređivanje (deployment)
  - Poboljšana izolacija grešaka (faults)
  - Bolja kontrola skaliranja
  - Kompatibilnost sa kontejnerima i orkestratorima
  - Nezavisno unapređivanje svake usluge (servisa)

Mogući nedostaci mikroservisa

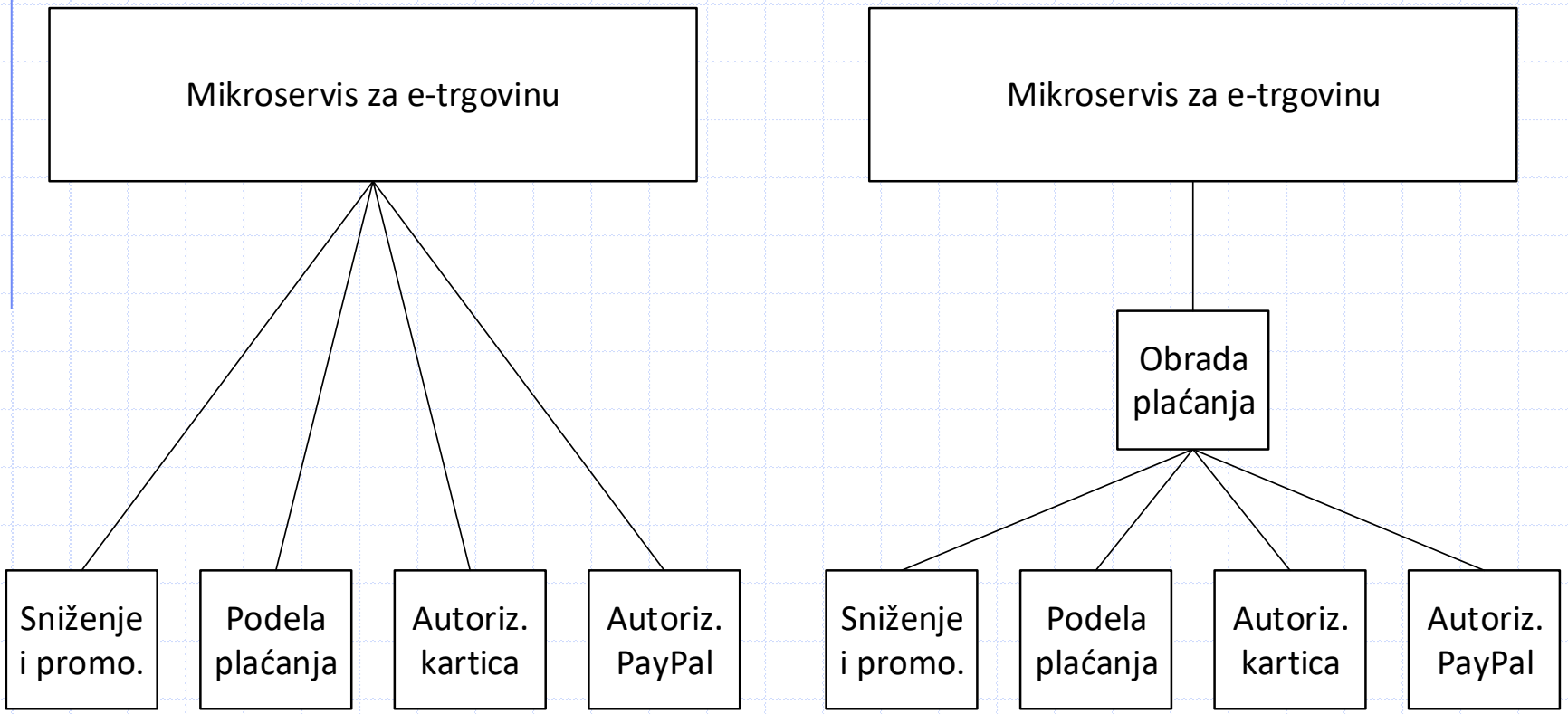
- Mogući nedostaci mikroservisa:
  - Kaskada grešaka
  - Dupliciranje i preklapanje funkcionalnosti
  - Složenost rukovanja (management)
  - Replikacija podataka i transmisiona režija
  - Povećana površ za napade na zaštitu sistema
  - Treniranje personala
- Replikacija podataka i transmisiona režija:
  - Funkcije u mono. app mogu da dele globalne pod
  - Mikroservisi moraju dobiti svoje kopije - režija

Granularnost mikroservisa (1/6)

- Ključan problem:
  - Izbor veličine mikroservisa
- U primeru: jedan mikroserv. za obradu plaćanja
  - Moguća su alternativna rešenja
- Obrada plaćanja uključuje:
  - Primena korisničkih sniženja i promotivnih kodova
  - Podela naplate (npr. između više kreditnih kartica)
  - Autorizacija korisničkih kreditnih kartica
  - Autorizacija korisničkog PayPal računa
- Moguće su 2 alternativna rešenja

Granularnost mikroservisa (2/6)

- Rešenja sa i bez mikroserv. za obradu plaćanja:



Granularnost mikroservisa (3/6)

- Tri heuristike za izbor granularnosti
  - Modeliranje poslovnih (biznis) procesa
  - Identifikacija zajedničkih funkcionalnosti
  - Adaptivna promena veličine i prestruktuiranje
- Modeliranje procesa poslovanja:
  - I pravilo: mikroservis po poslovnom procesu
  - Umesto slepe disagregacije app na mikroservise
  - Projektanti treba da uoče korake u radnom procesu (workflow)
  - Zatim te korake pretvaraju u mikroservise

Granularnost mikroservisa (4/6)

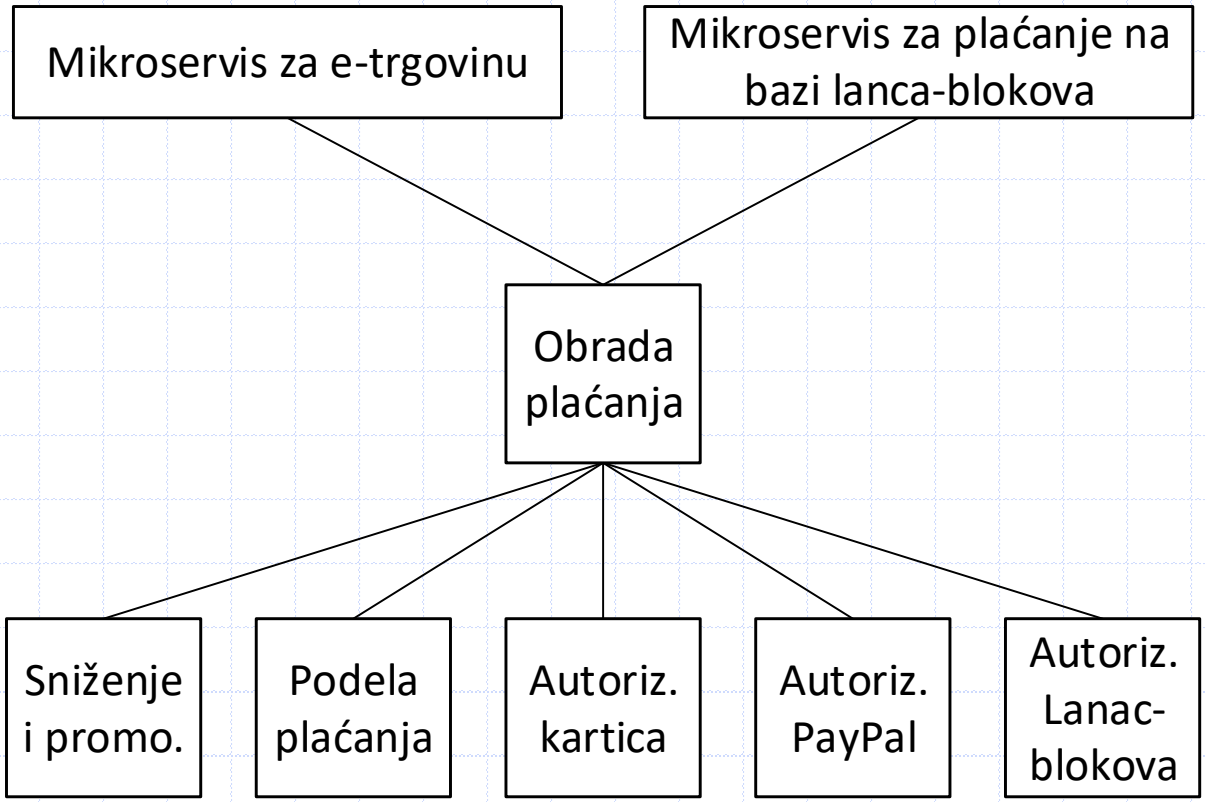
- Identifikacija zajedničkih funkcionalnosti:
  - II pravilo: iz SW inž. zajedničke funkcije, klase, itd.
  - Mikroserv. koji mogu da koriste više drugih mikroserv.
  - Npr. obrada plaćanja za kupce i na veliko i na malo
- Adaptivna promena veličine i prestruktuiranje:
  - III pravilo: iterativno odrediti najbolju veličinu i strukturu mikroservisa
  - Npr nova app koja ima preklapanje sa postojećim app
  - Potpuno novi skup mikrosev. bi doveo do dupliciranja i povećanja složenosti rukovanja
  - Bolje je prvo restruktuirati postojeće mikroservise

Granularnost mikroservisa (5/6)

- Jedan pristup restruktuiranju:
  - Razdvajanje API i unutrašnje funkcionalnosti
  - Onda se oba mogu nezavisno menjati
- Primer adaptivnog restruktuiranja:
  - Nova app koja koristi tehnologiju plaćanja na bazi lanca-blokova (blockchain)
  - Postojiće mikroservise restruktuiramo tako da novi mikroservis može da ih koristi – slika na sl. slajdu

Granularnost mikroservisa (6/6)

- Restruktuirani mikroservisi za novu vrstu plaćanja:



Komunikacioni protokoli za mikroservise

- Mikroservisi komuniciraju putem IP protokola:
  - Tako su dostupni i iz DC i van DC (zaštitćen pristup)
- U transportnom nivou većina koristi TCP
  - Dodatno koriste neki TP (transfer protocol)
  - TP definiše kako su bajti organizovani u poruke
  - Skup TP poruka definiše uslugu koja se nudi
- Moguće napraviti nov TP ili koristiti postojeći
- Dva najčešće korišćena TP:
  - HTTP: originalno razvijen za Web
  - gRPC: RPC (Remote Procedure Call) okruženje

HTTP

- Dvosmerna komunikacija entitet - mikroservis
  - Entitet može slati podatke mikroservisu ili
  - Zahtevati da mikroservis pošalje podatke
- Komunikacija:
  - Entitet šalje zahteve na koje mik.ser. odgovara
  - U zahtevu, pored oper. specificira stavku podataka putem njenog URI (Uniform Resource Identifier)
  - U nekim zahtevima, pošiljalac daje i podatke
  - Sledi kratak pregled 6 osnovnih operacija HTTP-a

Osnove operacije HTTP-a

Operacija	Opis
GET	Dobavi kopiju stavke podataka specificirane u zahtevu
HEAD	Dobavi metapodatke za zadatu stavku podataka (npr. vreme poslednje izmene)
PUT	Zameni specificiranu stavku podataka sa podacima poslatim u zahtevu
POST	Dodaj podatke poslate u zahtevu na kraj (append) specificrane stavke podataka
PATCH	Upotrebi podatke poslate u zahtevu za izmenu dela specificrane stavke podataka
DELETE	Ukloni stavku podataka specificiranu u zahtevu

gRPC (1/4)

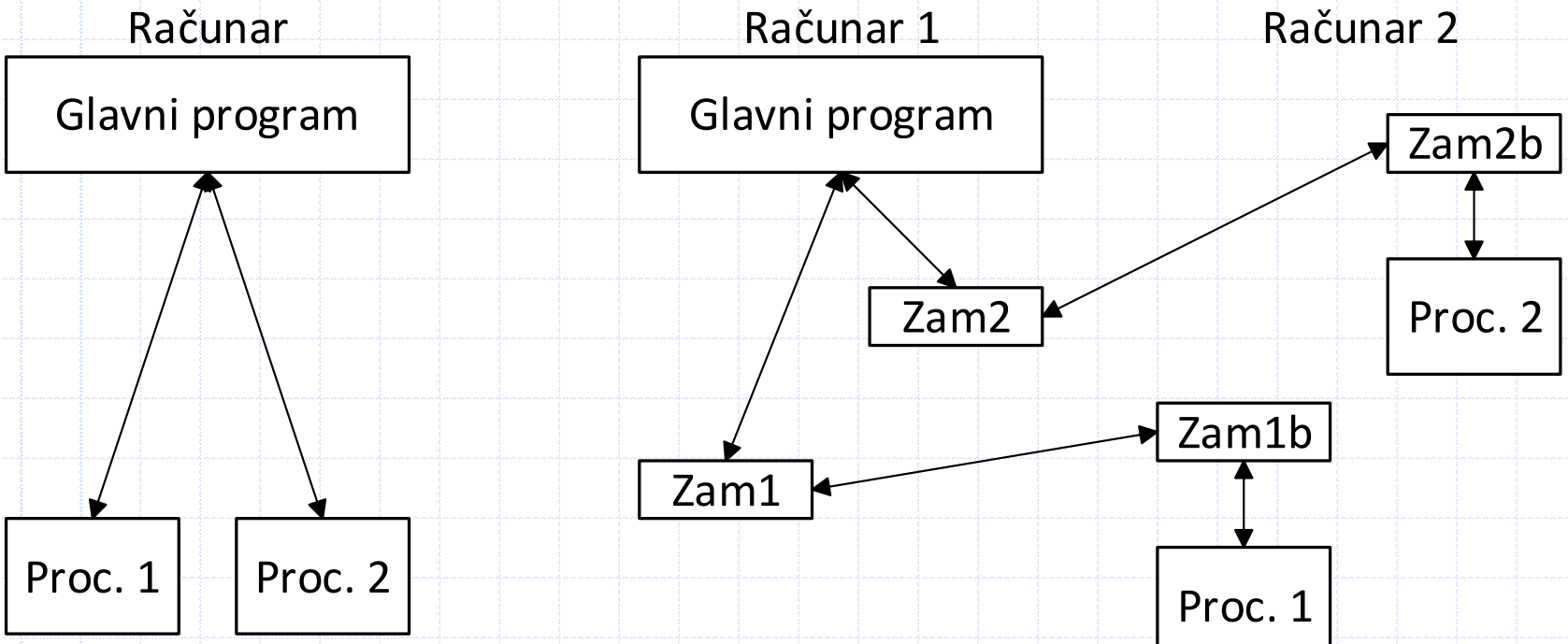
- gRPC obezbeđuje opštu osnovu za komunikaciju
  - Omogućava zadavanje specifičnog skupa operacija za svaku instnacu (tj. mikroservis)
- Ideja tradicionalnog RPC pristupa:
  - Program se izvršava na više računara, pri čemu su neke procedure lokalne a druge na udaljenim rač.
  - Da bi se pozvale udaljene procedure šalju se poruke
  - Poruke sadrže agrumente udaljene procedure
  - Povratna vrednost se šalje preko poruke odgovora

gRPC (2/4)

- RPC sistemi
  - Korisnik specificira tipove arg i povratne vrednosti
  - Sistem automatski generiše kod za razmenu poruka
  - Taj kod deluje transparentno, tj. originalni pozivi i procedure se ne menjaju
- Zamena udaljene procedure (stub/proxy)
  - U programu se umesto udaljene procedure postavlja njena zamena
  - Na udaljenom računaru korespodentna zamena poziva proceduru lokalnim pozivom (call instrukcijom)
  - Sledi primer koji ilustruje ovu ideju

gRPC (3/4)

- App sa 2 procedure i App sa 2 udaljene proc:



gRPC (4/4)

- gRPC proširuje tradic. RPC pristup na 2 načina:
  - Za razliku od većine RPC omogućava izbor prog. jezika, npr. C++, Java, Go, Node.js, Python, Ruby
  - Za razliku od većine RPC ne specificira format podataka između zamena udaljenih proc (npr. JSON)
  - Već koristi tehnologiju bafera protokola (protocol buffers) da napravi SW za serijalizaciju podataka
  - Za razliku od HTTP koji zahteva posebne zahteve za stavke pod, baferi protokola omogućavaju slanje toka (stream) više stavki podataka

Komunikacija između mikroservisa: REST

- 2 tipa komunikacije između mikroservisa:
  - Zahtev-odgovor (Request-response, REST interfejs)
  - Tok podataka (kontinualni interfejs)
- REST (ili RESTful) komunikacija/interfejs:
  - Pojavio se kao int. između pretraživača i web servera
  - Mikroservisi sa REST int. uzimaju ulogu servera
  - Podrazumevani TP je HTTP
- Nedostatak REST: 1 zahtev za svaku stavku pod
  - Za prenos velikog br stavki potreban veliki br zahteva
  - Komunikacija sa tokom podataka eliminiše taj problem

Komunikacija između mikroservisa: Tok podataka (1/2)

- Interfejs sa tokom podataka:
  - Klijent uspostavlja vezu sa mikroserv. i šalje 1 zahtev
  - Mikroserv. šalje sekvencu od 1 ili više stavki pod. kao odgovor na zahtev
- Glavna razlika između tradic. RPC i gRPC:
  - U gRPC, udaljena proc može da vrati tok stavki
- Kom. tipa objavi-pretplati (publish-subscribe):
  - Je varijanta komunikacije sa tokom podataka
  - Klijent pretplatnik (prijemnik) kontaktira mikroserv i specificira temu (topic)
  - Uspostavljena veza ostaje stalno otvorena

Komunikacija između mikroservisa: Tok podataka (2/2)

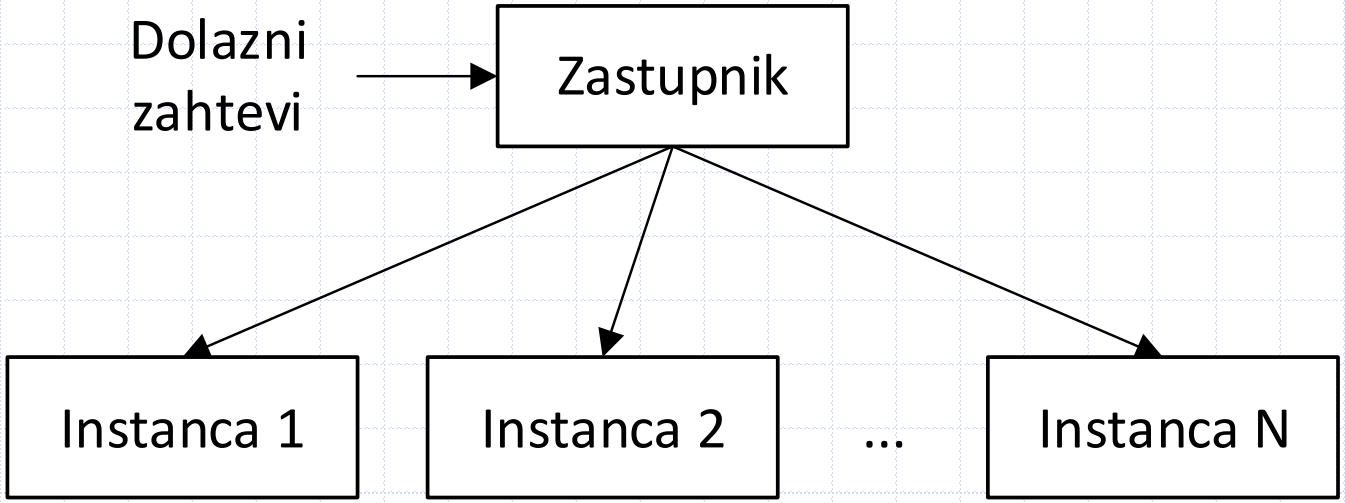
- Kom. tipa objavi-pretplati – nastavak:
  - Klijenti objavljivači (predajnici) kontaktiraju mikroserv radi slanja stavki pod
  - Svaka stavka pod je označena određenom temom
  - Kaže se da predajnici „objavljuju" info na razne teme, i da su prijemnici „pretplaćeni" na neke teme
  - Odatle potiče naziv „objavi-pretplati"

Mreža usluga (Service Mesh)

- Mikroserv može da koristi više tipova kom.
  - Npr. gRPC za komunikaciju sa drugim mikroserv. i REST za komunikaciju sa klijentima
  - Plus, svaki mikroserv može da se nezavisno skalira
- Mreža usluga je SW sistem koji obezbeđuje:
  - Pravljenje instanci mikroservisa
  - Usmeravanje zahteva ka instancama mikroservisa
  - Pretvararanje zahteva u interni format
  - Otkrivanje mikroservisa
- Rešenje za usmeravanje zahteva:
  - Uvodi se zastupnik mreže usluga

Zastupnik mreže usluga (Service Mesh Proxy)

- Rešenje za usmeravanje zahteva - nastavak:
  - Klijenti šalju zahteve zastupniku, on ih prosleđuje instancama mikorserv
  - Spoljni interfejs je npr. REST a unutrašnji gRPC



Mogućnost međusobnog blokiranja (1/4)

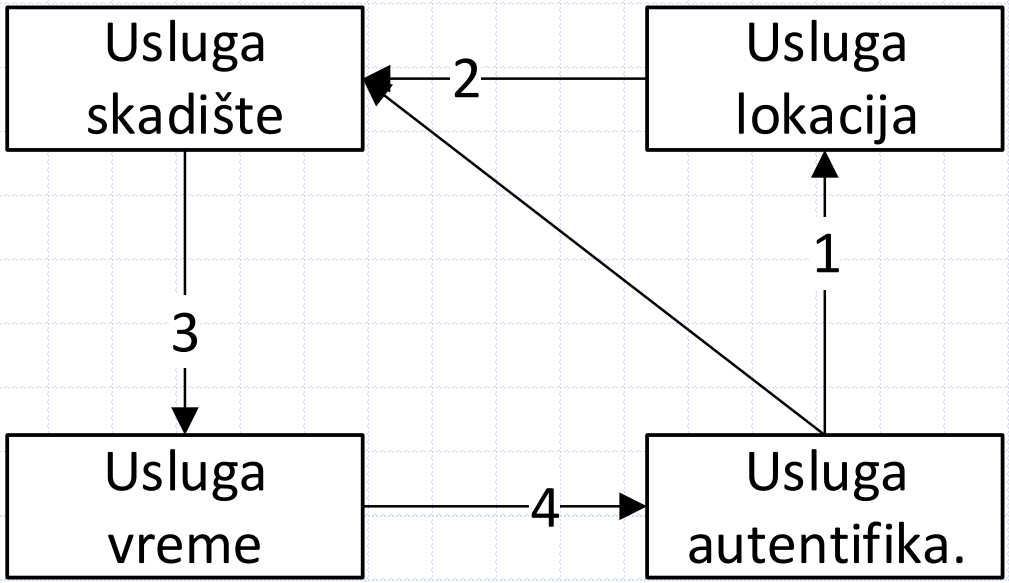
- Kružne međuzavisnosti u skupu mikroservisa
  - Ako svaki mikroserv čeka neki drugi mikroserv
  - Tada nastaje „međusobno blokiranje“ (deadlock)
- Možda auto skaliranje štiti sis. od ove patologije?
  - Ako mikroserv A zavisi od B, iako je instanca B blokirana, nova instanca B može biti napravljena
  - U slučaju na 1000-de mikroserv mogu se pojaviti skrivene međuzavisnosti
- Primer sa 4 mikroserv:
  - Vreme, lokacija, skladište, autentifikacija

Mogućnost međusobnog blokiranja (2/4)

- Primer sa 4 mikroserv - nastavak:
  - Skladište koristi vreme
  - Lokacija koristi skladište
  - Autentifikacija koristi lokaciju i skladište
  - Ako se svaki mikroserv završi i restartuje, sistem (nastavlja da) radi korektno
- Izmena u projektu:
  - Vreme autentifikuje svaki zahtev (čeka odgovor)
  - Ako se vreme uredno restartuje, sis. radi korektno
  - Međutim, skrivena kružna zavisnost u fazi pokretanja sistema dovodi do pojave međusobnog blokiranja

Mogućnost međusobnog blokiranja (3/4)

- Primer međusobnog blokiranja:
  - 1: autentifikacija zahteva lokaciju, 2: lokacija zahteva početnu konfig., 3: skladište zahteva vreme, 4: vreme zahteva autentifikaciju



Mogućnost međusobnog blokiranja (4/4)

- Interesantno, u predhodnom primeru:
  - Među. blokiranje se neće desiti ako sve usluge ne startuju istovremeno, i sistem će normalo raditi
  - Ali nakon velikog restarta, ovaj problem će se pojaviti
- Generalno:
  - Pošto se mikroservisi razvijaju nezavisno, kružne zavisnosti mogu ostati skrivene
  - A onda se otkrivaju u neobičnim situacijama, kao što je veliki restart sistema

Tehnologije mikroservisa

- Tehnologije mreže-servisa uključuju:
  - CNCF (Cloud Native Computing Foundation) projekat „Linkerd“
  - Zajednički projekat „Istio“ firmi Google, IBM, Lyft
- Okruženja za pravljenje i rukovanje mikroserv.:
  - Npr. „Spring Boot“