

Usluge i Interoperabilnost

Arhitekture · HTTP · Protokoli · Middleware · gRPC · REST · GraphQL

Što ćemo danas obraditi

01

Usluge

Definicije · Stanja · S stanjem u odnosu na stanje bez stanja

03

Arhitekture

Klijent-poslužitelj · Mikroservisi · SOA · Besserverski

05

Tehnologije

REST · gRPC · GraphQL · WebSockets · SOAP

02

HTTP protokol

Povijest · Verzije · Metode · Zaglavlja · Kodovi

04

Interoperabilnost

Komunikacija sloj po sloj · Slučajevi upotrebe u stvarnom svijetu

06

Middleware

Kafka · RabbitMQ · API pristupnik · Mreža usluga

01

Usluge

Definicije · Stanja · S stanjem u odnosu na stanje bez stanja

Što je Usluga?



U gospodarstvu

- Nematerijalni oblik dobra
- Ne može rezultirati vlasništvom
- Jedna stranka obavlja funkciju za drugu
- Primjeri: bankarstvo, zdravstvo, dostava, Uber, Spotify



U računarstvu

"Mehanizam za omogućavanje pristupa jednoj ili više mogućnosti, gdje se pristup omogućuje korištenjem propisanog sučelja i ostvaruje se u skladu s ograničenjima i pravilima kako je navedeno u opisu usluge."

— OASIS (Organizacija za unapređenje standarda strukturiranih informacija)



Ključni uvid: Usluga otkriva funkcionalnost putem dobro definiranog sučelja bez otkrivanja svoje interne implementacije - baš kao API koji pozivate bez poznavanja njegovog izvornog koda.

Komponenta vs. Servis: Komponenta je dio unutar programa koji se može ponovno koristiti (bez pristupa izvornom kodu). Servis je dostupan izvana, neovisno, putem sinkronog ili asinkronog pristupa.

Usluge s potvrdom stanja u odnosu na usluge bez potvrde stanja — izravno u usporedbi

Kriterij	 Stateless	 Stateful
Memorija sesije	Ništa — svaki zahtjev je samostalan	Poslužitelj prati sesiju kroz zahtjeve
Skalabilnost	Horizontalno skaliranje je jednostavno 	Skaliranje je složeno (ljepljive sesije) 
Performanse	Manja potrošnja memorije poslužitelja 	Veća potrošnja memorije 
Tolerancija grešaka	Visoko — bilo koji poslužitelj može obraditi zahtjev 	Neuspjeh može prekinuti aktivnu sesiju 
Složenost	Klijent mora svaki put ponovno poslati kontekst	Jednostavniji klijent, složenija logika poslužitelja
Primjeri upotrebe	REST API-ji, mikroservisi, CDN, DNS	Online igre, bankarske sesije, SSH
Primjer iz stvarnog svijeta	Instagram feed, Google Maps API	Igre za više igrača, bolnički kartoni pacijenata

► Statefull u odnosu na stateless <https://www.youtube.com/watch?v=oU6nEuLv-Z0>

Stateless u praksi — Kako funkcioniraju REST API-ji

Svaki zahtjev sadrži SVE potrebne informacije. Poslužitelj ne sadrži nikakav kontekst sesije.



"Stanje" (korisnički podaci, košarica, postavke) nalazi se u bazi podataka ili tokenu - NE u memoriji poslužitelja.

Instagram Reels API

Svaki zahtjev uključuje autorizacijski token. Bilo koji od tisuća poslužitelja može odgovoriti.

API za Google karte

Vaš zahtjev za kartu uključuje lokaciju, zumiranje, API ključ — potpuno samostalno.

Spotify /pjesme

Pristupni token + ID pjesme u zaglavlju = poslužitelj vraća MP3 stream.

DNS pretraga

Potpuno stateless: pošaljite upit, dobijte IP adresu. Gotovo. Nema sesije.

Stateful u praksi — Kada se stanje mora očuvati

Neke interakcije inherentno zahtijevaju očuvanje konteksta u više razmjena.

Igre za više igrača

Server igre mora u svakom trenutku znati poziciju, zdravlje i inventar svakog igrača. Novi server ne može samo 'preuzeti' pokrenutu igru - potrebno je puno stanje sesije.

Praćenje pacijenata

Sustavi za praćenje jedinice intenzivne njege moraju kontinuirano pratiti vitalne znakove, stanje lijekova i alarme. Sesija je život pacijenta - gubitak stanja nije prihvatljiv.

Blagajna e-trgovine

Košarica → dostava → plaćanje → potvrda. Koraci su ulančani i imaju status. Obraduje se putem sesijskih tokena ili ljepljivih sesija.

Bankarske transakcije

Višekoračni bankovni transfer mora pratiti: autentificiranog korisnika → odabir računa → potvrdu iznosa → provjeru OTP-a. Svaki korak ovisi o prethodnom.

VoIP / SSH sesija

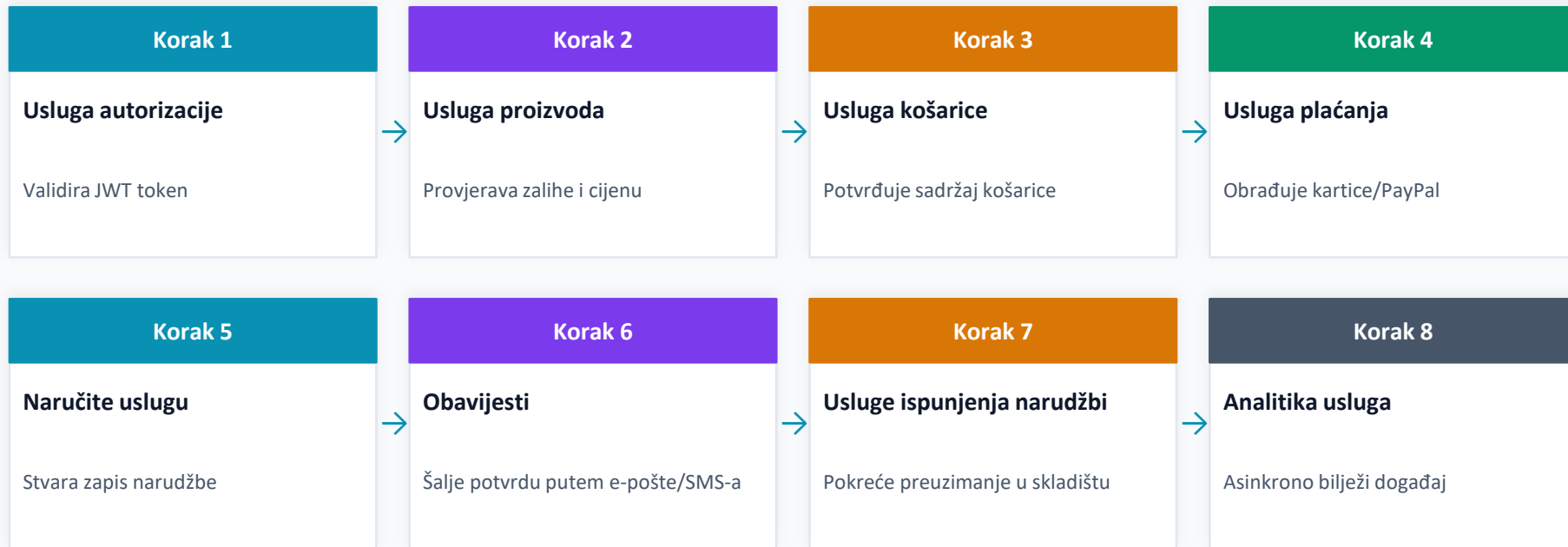
SSH sesija ili Zoom poziv održava živu vezu s ključevima za šifriranje, slijednim brojevima i dogovorenim kodekom - sve uz održavanje stanja.

LLM Chat (da, ovo!)

AI asistent čuva povijest razgovora kako bi pružio odgovore svjesne konteksta. Potpuna povijest razgovora JE stanje koje se prenosi sa svakom novom porukom.

Ulančavanje usluga — primjer online kupovine

Jedna radnja korisnika ("Kupi odmah") pokreće lanac mikroservisa:



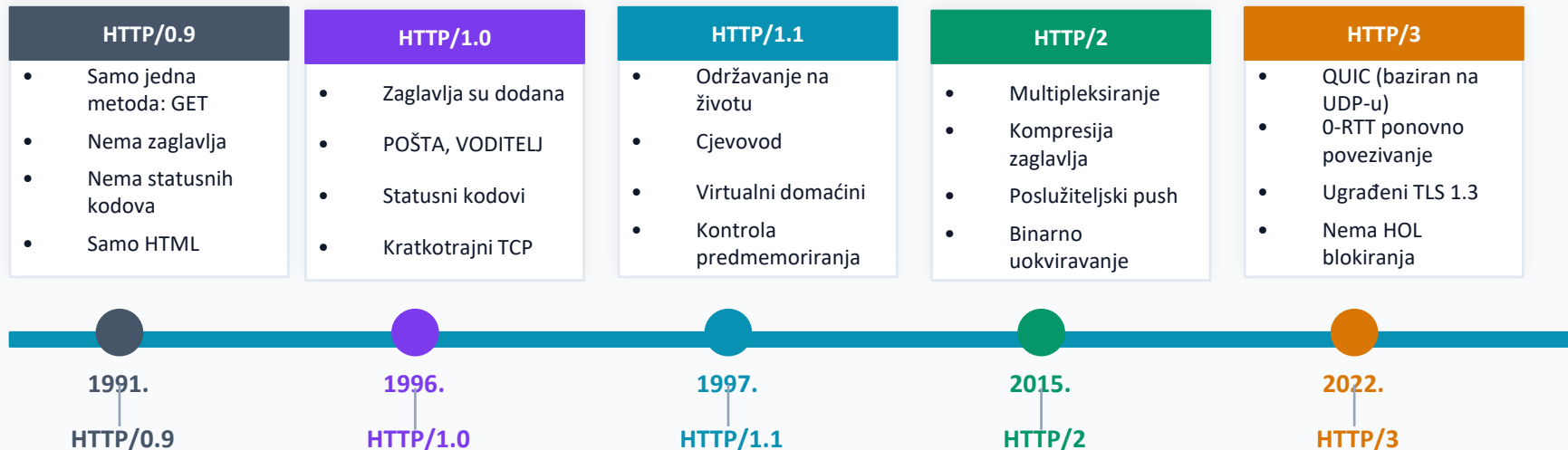
💡 *Koraci 1–6 su (uglavnom) bez stanja — svaka usluga je neovisna. Koraci se mogu izvoditi paralelno. Korak 8 je potpuno asinkroniran.*

02

HTTP protokol

Povijest · Verzije · Zahtjev/Odgovor · Metode · Statusni kodovi

HTTP — Kratka povijest web protokola



💡 HTTP/3 (temeljen na QUIC-u) sada koristi ~30% web stranica, uključujući Google, YouTube i web stranice koje poslužuje Cloudflare.

▶ HTTP/1 na HTTP/3 — Fireship <https://www.youtube.com/watch?v=a-sBfyiXysI>

HTTP verzije — Usporedba značajki

Značajka	HTTP/1.0	HTTP/1.1	HTTP/2	HTTP/3
Transportni sloj	TCP	TCP (održavanje veze)	TCP	QUIC (UDP)
Multipleksiranje	✗ Ne	⚠ Djelomično (cjevovod)	✓ Puno	✓ Puno
Kompresija zaglavlja	✗ Ne	✗ Ne	✓ HPACK	✓ QPACK
Push servera	✗ Ne	✗ Ne	✓ Da	✓ Da
TLS enkripcija	Izborno	Izborno	Praktično potrebno	✓ Ugrađeno (TLS 1.3)
Blokiranje na početku linije	⚠ Da	⚠ Da	⚠ TCP razina	✓ Eliminirano
Uspostavljanje veze (latencija)	1 RTT/zahtjev	1 RTT (ponovna upotreba)	1 RTT	0-RTT (ponovno povezivanje)
Trenutno usvajanje (2025.)	~Nasljeđe	~40% web poslužitelja	~65% najpopularnijih stranica	~30% najpopularnijih stranica

HTTP/3 i QUIC

HTTP/3 radi preko QUIC-a — transportnog protokola izgrađenog na UDP-u, koji je razvio Google, a standardizirao IETF (RFC 9000).

Zašto QUIC?

0-RTT veza:

Ponovno povezivanje bez kašnjenja u povratnom veznom toku (ključno na mobilnim uređajima)

Bez blokiranja HOL-a:

Izgubljeni paket zaustavlja samo svoj stream, ne SVE streamove (u odnosu na TCP)

Ugrađeni TLS 1.3:

Šifriranje je obavezno i integrirano, nije ugrađeno

Migracija veze:

Veza preživi mrežne prekidače (WiFi → 4G) pomoću ID-ova veze

Multipleksirani tokovi:

Više neovisnih tokova bajtova po vezi

Već u produkciji:

- Google pretraga, YouTube, Gmail
- Cloudflare CDN (pokreće više od 20% weba)
- Meta (Facebook, Instagram)
- Apple (Safari u potpunosti podržava HTTP/3)

Usporedba stogova:

HTTP/1.1 i 2: HTTP → TLS → TCP → IP

HTTP/3: HTTP → QUIC (TLS+UDP) → IP

Anatomija HTTP zahtjeva i odgovora

HTTP REQUEST

Start Line: GET /api/v1/feed HTTP/2

Host: api.instagram.com

Auth Header: Authorization: Bearer <JWT>

Accept: Accept: application/json

User-Agent: Mozilla/5.0 (iPhone; iOS 17)

Body: (empty for GET requests)



HTTP RESPONSE

Status Line: HTTP/2 200 OK

Content-Type: Content-Type: application/json

Date: Date: Tue, 10 Mar 2026 12:00:00

Cache-Control: Cache-Control: max-age=60

Body: { "posts": [...], "next": "..."}

Size: Content-Length: 48392

1xx Information

2xx Success

3xx Forwarding

4xx Client Error

5xx Server Error

HTTP metode — CRUD mapirane na stvarne API-je

Method	Safe?	Idempotent?	CRUD	Real API Example
GET	✓ Yes	✓ Yes	Read	GET /api/posts?user=alex — fetch Instagram posts
POST	✗ No	✗ No	Create	POST /api/tweets — create a new tweet
PUT	✗ No	✓ Yes	Replace	PUT /api/users/123 — replace full user profile
PATCH	✗ No	✓ Yes	Update	PATCH /api/users/123 — update only username field
DELETE	✗ No	✓ Yes	Delete	DELETE /api/posts/456 — delete a post
HEAD	✓ Yes	✓ Yes	Meta	HEAD /api/video/hd.mp4 — check size before downloading
OPTIONS	✓ Yes	✓ Yes	Info	OPTIONS /api/feed — CORS preflight check from browser

URI · URL · URN · MIME tipovi

Anatomija URI-ja:

`https://api.algebra.hr:443/v1/students?course=iis&year=2026#results`

► shema: **https**

► autoritet: **api.algebra.hr:443**

► put: **/v1/studenti**

► upit: **tečaj=iis&godina=2026**

► fragment: **rezultati**

URL — Jedinstveni lokator resursa

Identificira GDJE se resurs nalazi. Lokacija se može promijeniti!

`https://open.spotify.com/track/abc123`
`ftp://ftp.algebra.hr/iis/lecture03.pdf`

URN — Jedinstveni naziv resursa

Identificira ŠTO je resurs. Jedinstveno i trajno!

`urn:isbn:978-3-16-148410-0` (knjiga)
`urn:ietf:rfc:9000` (QUIC standard)

Uobičajeni MIME tipovi:

`audio/mpeg`

`video/mp4`

`višedijelni/obrazac-podaci`

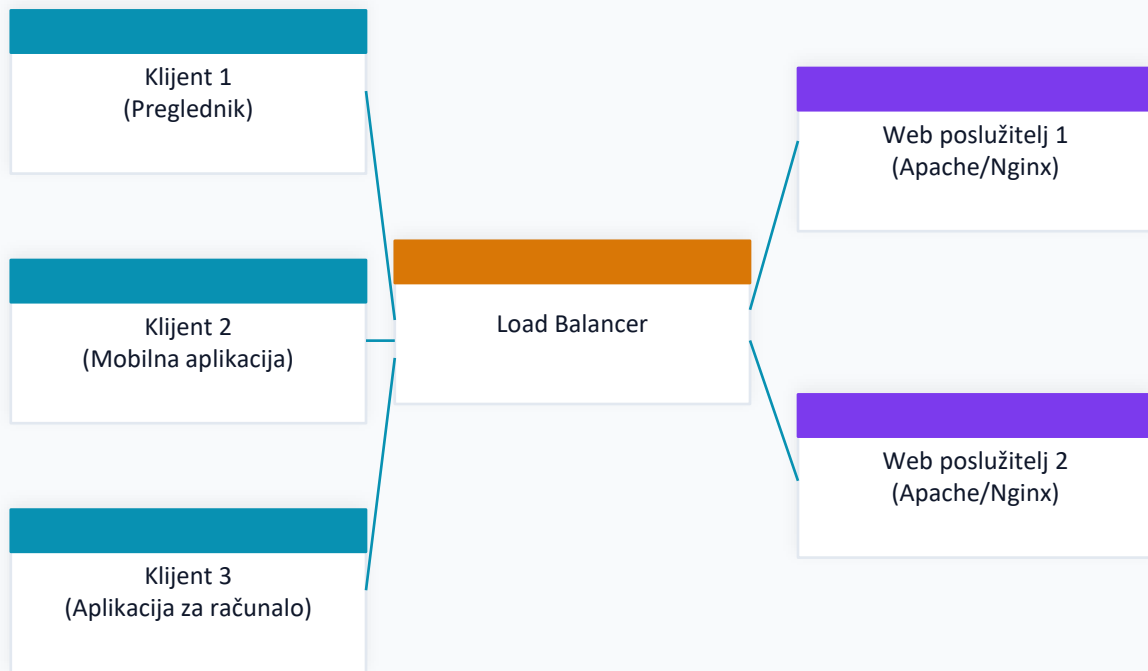
`aplikacija/tok-okteta`

03

Arhitekture distribuiranih sustava

Klijent-poslužitelj · N-slojni · Mikroservisi · SOA · Bez poslužitelja · Vođen događajima

Arhitektura klijent-poslužitelj



Ključne karakteristike

- Serveri uvijek osluškiju zahtjeve
- Klijenti mogu biti tanki (preglednik) ili *fat* (desktop aplikacija)
- Serveri mogu djelovati kao klijenti drugim serverima
- Uravnoteživač opterećenja distribuira promet
- LAMP paket: Linux + Apache + MySQL + PHP
- Većina web aplikacija danas slijedi ovaj model

N-slojna / višeslojna arhitektura

Prezentacijski sloj

Preglednik · Mobilna aplikacija · React/Angular · ASP.NET · JSP

Sloj prezentacije poslužitelja

Kontroleri · Prikazi · Predlošci · Upravljanje sesijama

Sloj poslovne logike

Usluge · Validatori · Pravila · Orkestracija · EJB · Spring

Sloj za pristup podacima

Repozitoriji · ORM (Hibernate/EF Core) · DAO · JDBC

Sloj podataka

PostgreSQL · MySQL · MongoDB · Redis · Oracle · NoSQL

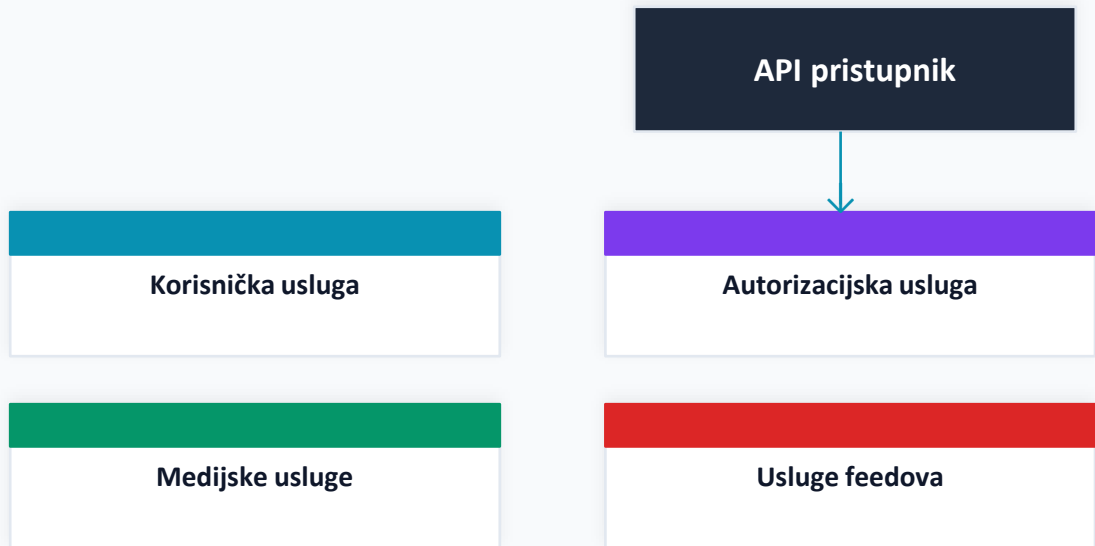
Zašto slojevi?

- Razdvajanje briga
- Svaki sloj se može zamijeniti neovisno
- Omogućuje specijalizaciju tima
- Lakše testiranje (probni donji slojevi)
- Klasični: 3-slojni (prezentacija + poslovanje + podaci)
- Poduzeće: Java EE, .NET, Spring Boot

 *Stvarni svijet: Facebook koristi prezentacijski sloj (React), poslovni sloj (Hack/PHP), sloj za pristup podacima (TAO graph) i sloj za podatke (MySQL + Cassette).*

Arhitektura mikroservisa

Razbijte monolit! Svaka usluga = jedna odgovornost, jedna kodna baza, jedna baza podataka, neovisno raspoređivanje.



Primjer Netflixa

- 700+ mikroservisa u produkciji
- Svaki tim je odgovoran za svoju uslugu od početka do kraja
- Neovisno implementirano (dnevna izdanja)
- Kvar jedne usluge ≠ kvar cijelog sustava
- Chaos Monkey namjerno ukida servise kako bi testirao otpornost

Više arhitektura — SOA · Event-Driven Architecture · Serverless

Servisno orijentirana arhitektura (SOA)

Arhitektura na razini poduzeća gdje usluge komuniciraju putem Enterprise Service Busa (ESB). Teža od mikroservisa, još uvijek uobičajena u bankarstvu i telekomunikacijama.

 Integracije sa SAP-om, IBM WebSphereom i Salesforceom

✓ Mogućnost ponovne upotrebe poslovnih usluga

⚠ ESB može biti usko grlo

✓ Snažno upravljanje i sigurnost

⚠ Teška, složena postavka

✓ Podrška za WS-* standarde

⚠ Sporije od mikroservisa

Event-Driven Architecture (EDA)

Usluge komuniciraju emitiranjem i reagiranjem na događaje. Proizvođači ne poznaju potrošače. Kafka/RabbitMQ su okosnica.

 Uberovo povećanje cijena, rezervacije na Airbnb-u, trgovanje dionicama

✓ Labava povezanost između usluga

⚠ Teže je otklanjati greške / pratiti

✓ Visoka skalabilnost i otpornost

⚠ Eventualni izazovi s dosljednošću

✓ Obrada u stvarnom vremenu

⚠ Složena orkestracija

Serverless / FaaS (Funkcija kao usluga)

Implementirajte pojedinačne funkcije — bez upravljanja poslužiteljem. Automatsko skaliranje u oblaku. Plaćajte samo za vrijeme izvršenja.

 AWS Lambda, Azure funkcije, Google Cloud Run

✓ Nulti menadžment infrastrukture

⚠ Latencija hladnog pokretanja

✓ Automatsko skaliranje od 0 do ∞

⚠ Ograničeno trajanje izvršenja

✓ Model plaćanja po korištenju

⚠ Rizik od vezanosti za dobavljača

Kada koristiti koju arhitekturu? — Vodič za odlučivanje

Arhitektura	Skala	Složenost	Veličina tima	Najbolje za	Primjeri iz stvarnog svijeta
Monolitni	Mali	Nisko	1–5	Prototipovi, startupi	Aplikacije u ranoj fazi, MVP-ovi
Klijent-poslužitelj	Srednji	Srednji	3–15	Web aplikacije, portali	WordPress, CMS stranice, LAMP aplikacije
3-slojni / N-slojni	Srednji	Srednji	5–20	Web aplikacije za tvrtke	Bankarski portali, ERP-ovi, Spring Boot aplikacije
SOA	Veliko	Visoko	20–100	Integracija poduzeća	SAP, IBM sustavi, B2B integracije
Mikroservisi	Vrlo veliko	Visoko	10–500+	Nativno u oblaku, velika brzina	Netflix, Amazon, Spotify, Uber
Event-Driven	Vrlo veliko	Visoko	5–100+	Asinkroni tijekovi rada u stvarnom vremenu	Kafkin stil: Uber, LinkedIn, Twitter
Serverless/FaaS	Neograničen	Nisko	1–50	Povremena opterećenja, API-ji	AWS Lambda funkcije, Firebase Cloud funkcije

04

Interoperabilnost slojeva

Klijent · Prezentacija · Poslovanje · Slojevi podataka — Primjeri upotrebe iz stvarnog svijeta

Interoperabilnost višeslojnih arhitektura

Različiti slojevi mogu se implementirati u različitim tehnologijama i dalje komunicirati - to je interoperabilnost.

Klijentski sloj

React Native · Angular · JavaFX · .NET WinForms

Prezentacijski sloj

Spring MVC · ASP.NET Core · JSP · Next.js

Sloj poslovne logike

Spring usluge · EJB · .NET domena · Python FastAPI

Sloj za pristup podacima

Hibernate · Entity Framework · JDBC · SQLAlchemy

Sloj podataka

MySQL · PostgreSQL · MongoDB · Oracle · Redis · Kafka

← Bilo koji sloj može interoperabilno djelovati →
s bilo kojim drugim slojem pomoću
standardni protokoli

HTTP/REST

gRPC

GraphQL

WebSocket

SOAP

JDBC/ODBC

Interoperabilnost slojeva — ključni scenariji

Klijent → Prezenciacija

Najčešći! React/Angular frontend pozivi Spring Boot ili ASP.NET Core REST API. JSON preko HTTP-a.

React Native aplikacija ↔ Spring Boot API
Angular SPA ↔ ASP.NET Core kontroler

Klijent → Sloj poslovne logike

Klijent zaobilazi prezentacijski sloj (spojen je s klijentom). AJAX poziva izravno iz preglednika prema pozadinskim uslugama.

TikTok web aplikacija koristi XMLHttpRequest za izravno pozivanje usluge preporuke feedova

Prezenciacija → Sloj poslovne logike

Najvažniji poslovni scenarij! JSP/ASP.NET poziva EJB / Spring Service / COM+. Preporučuje se s web servisima.

ASP.NET Core kontroler → Spring Boot @Service
JSP stranica → EJB sesijski bean

Sloj poslovne logike → Pristup podacima

Poslovne komponente pozivaju ORM ili repozitориjske obrasce preko tehnoloških granica.

Spring @Service → ADO.NET putem JCA konektora
Node.js servis → Hibernacija putem REST mosta

Pristup podacima → Sloj podataka

ORM ili DAO komunicira s relacijskom bazom podataka, NoSQL-om ili vanjskim izvorima podataka. JDBC, ODBC, JPA, EF Core.

Hibernacija (Java) ↔ Oracle baza podataka
Entity Framework (.NET) ↔ PostgreSQL

Sloj podataka ↔ Sloj podataka

Dijeljene baze podataka ili replikacija baze podataka. Jedna pohranjena procedura poziva podatke iz druge baze podataka. Tehnički moguće, ali koristite štedljivo!

PostgreSQL omotač stranih podataka → MySQL
Oracle DB link → MS SQL Server

Primjeri upotrebe interoperabilnosti u stvarnom svijetu

Spotify

React (JS) → Golang mikroservisi → Cassandra + GCS

Spotify web aplikacija (React) poziva više od 100 backend mikroservisa napisanih u Golangu, Javi i Pythonu. Svaki ima vlastitu bazu podataka - Cassandra za korisničke podatke, GCS za audio datoteke. Potpuna interoperabilnost više tehnologija.

Amazon

Klijentska aplikacija → API pristup → 500+ Java/.NET/Python usluga → DynamoDB/RDS

Amazonova naplata uključuje mikroservise u Javi, .NET-u i Pythonu, koji se međusobno pozivaju putem REST-a i gRPC-a. Učitavanje jedne stranice može pokrenuti više od 100 poziva servisa. Stanje se upravlja putem tokena sesije i DynamoDB-a.

Hrvatsko bankarstvo (višeplatformski ERP)

ASP.NET Core frontend → Spring Boot poslovne usluge → Oracle baza podataka

Hrvatski bankarski sustavi često koriste .NET za web portale okrenute klijentima, dok se osnovno bankarstvo izvodi na Javi (Spring/JBoss). Interoperabilnost se postiže putem SOAP web servisa ili REST API-ja preko interne mreže.

Hrvatski zdravstveni sustav (CEZIH)

Naslijeđene SOAP usluge + REST API Gateway → PostgreSQL + HL7 FHIR

CEZIH (Hrvatski zdravstveni informacijski sustav) izlaže podatke o pacijentima putem SOAP web servisa i novijih REST/FHIR API-ja. Bolnički sustavi (Java), softver za ljekarne (.NET) i aplikacije za liječnike opće prakse (razne) svi međusobno djeluju putem ovog sloja.

05

Tehnologije interoperabilnosti

REST · gRPC · GraphQL · WebSockets · SOAP

REST API-ji — Dominantni standard

Što je REST?

Representational State Transfer — arhitektonski stil (ne protokol) za dizajniranje mrežnih aplikacija. Definirao Roy Fielding (2000.). Koristi HTTP metode, URI-je i JSON/XML korisne terete.

Bez državljanstva

Nema sesije na poslužitelju. Svaki zahtjev je samostalan.

Klijent-poslužitelj

Frontend i backend su odvojeni

Slojeviti sustav

Proxyji i uravnoteživači opterećenja su transparentni

Jedinstveno sučelje

HTTP glagoli + URI-ji resursa (GET /users/123)

Može se predmemorirati

Odgovori se mogu predmemorirati radi performansi

Kod na zahtjev

Neobavezno: poslužitelj može poslati JS klijentu

REST API Example

```
GET /api/v1/users/42
Authorization: Bearer eyJ...
```

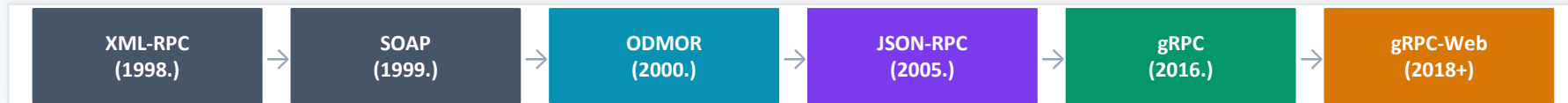
→ 200 OK

```
{
  "id": 42,
  "username": "alex",
  "courses": [
    "IIS", "Java", ".NET"
  ]
}
```

```
POST /api/v1/courses
{ "name": "IIS 2026" }
→ 201 Created
```

gRPC — Moderna zamjena za XML-RPC

gRPC (Google Remote Procedure Call) je visokoučinkoviti RPC okvir koji koristi HTTP/2 i Protocol Buffers. Razvio ga je Google, sada je CNCF projekt.



✨ Značajke gRPC-a

- HTTP/2 transport (multipleksiranje, streaming)
- Protokolni međuspremnici (binarni, strogo tipizirani) — 5-10× manji od JSON-a
- Automatski generirani klijentski/poslužiteljski kod iz .proto datoteka
- 4 vrste poziva: unarni, streaming poslužitelja, streaming klijenta, dvosmjerni
- Ugrađena TLS enkripcija
- Podržano u: Javi, Go-u, Pythonu, C#, Node.jsu, C++

Proto file (.proto)

```
syntax = "proto3";
service CourseService {
  rpc GetCourse (CourseRequest)
    returns (CourseResponse);
  rpc StreamLectures (CourseRequest)
    returns (stream Lecture);
}
message CourseRequest {
  int32 course_id = 1;
}
message CourseResponse {
  string name = 1;
  repeated string topics = 2;
}
```

 Koriste ga: Google (sve interne usluge), Netflix, Square, CoreOS, Docker, Cisco, Juniper, Dropbox, Lyft

GraphQL i WebSockets — Moderna komunikacija

GraphQL

Jezik upita za API-je — tražite točno ono što vam treba, ništa više, ništa manje. Razvio Facebook (2015.).

- Jedna krajnja točka (/graphql) za sve upite
- Strogo tipizirana shema
- Nema prekomjernog ili premalog preuzimanja
- Koristi se od strane: GitHub, Shopify, Twitter, Instagram
- Izvrsno za složene, ugniježdene podatke (društveni grafovi)

```
query {  
  user(id: 42) {  
    name  
    courses {  
      title  
      instructor  
    }  
  }  
} # Only fetches name + courses
```

WebSockets

Potpuno dupleksna, trajna veza preko jedne TCP veze. Idealno za aplikacije u stvarnom vremenu. Nadogradnje s HTTP-a.

- Dvosmjerno — poslužitelj može poslati podatke klijentu
- Niska latencija (bez HTTP opterećenja po poruci)
- Koristi se od strane: WhatsApp, Discord, Slack, Twitch chat uživo
- Platforme za trgovanje (cijene dionica u stvarnom vremenu)
- Igre za više igrača (ažuriranja pozicije u stvarnom vremenu)

```
// JavaScript WebSocket client  
const ws = new WebSocket(  
  'wss://chat.discord.com/ws'  
);  
ws.onmessage = (event) => {  
  console.log(event.data); // push!  
};  
ws.send('{"type":"message","text":"hi"}');
```

SOAP i XML-RPC — Zastarjeli, ali još uvijek u produkciji

XML-RPC (1998.) — Predak

- Udaljeni poziv procedure korištenjem XML-a preko HTTP-a
- Preteča SOAP-a — jednostavniji, ali ograničeniji
- Bez formalnog tipkanja, bez opisa usluge
- Još uvijek se nalazi u: WordPress admin API-ju, nekim starijim sustavima
- Zamijenjeno s gRPC-om, REST-om i GraphQL-om

SOAP web usluge (1999+)

- Protokol za razmjenu poruka temeljen na XML-u putem HTTP/SMTP-a
- WSDL — strojno čitljiv opis usluge
- WS-Sigurnost, WS-Atomska transakcija, WS-Pouzdana slanje poruka
- Zlatni standard poduzeća za razdoblje 2000.–2015.
- Još se koristi u: bankarstvu (SWIFT), zdravstvu (HL7), vladinim portalima

SOAP Envelope Example:

```
<soap:Envelope xmlns:soap="...">
  <soap:Header>
    <wsse:Security>...</wsse:Security>
  </soap:Header>
  <soap:Body>
    <m:GetCourse xmlns:m="...">
      <m:CourseId>42</m:CourseId>
    </m:GetCourse>
  </soap:Body>
</soap:Envelope>
```

Kada i dalje koristiti SOAP:

- Potrebni su formalni ugovori (WSDL)
- Ugrađena WS-Security za osjetljive podatke
- ACID transakcije (WS-AtomicTransaction)
- Integracija sa starijim poslovnim sustavima
- Zahtjevi za usklađenost s propisima

Tehnologije interoperabilnosti — Tablica usporedbe

Tehnologija	Protokol	Format	Performanse	Sigurnost tipova	Streaming	Najbolji slučaj upotrebe
ODMOR	HTTP/1.1–2	JSON/XML	Dobro	✗ Ništa	Ograničeno	Javni API-ji, CRUD, većina web servisa
gRPC	HTTP/2	Protobuf	⚡ Najbolje	✓ Snažan	✓ Puno	Interna komunikacija između mikroservisa
GraphQL	HTTP/1.1–2	JSON	Dobro	✓ Shema	⚠ (Titlovi)	Dohvaćanje složenih podataka, mobilne aplikacije, društveni grafovi
WebSocket	TCP/HTTP	Bilo koji	⚡ Najbolje	✗ Ništa	✓ Puno	U stvarnom vremenu: chat, igre, nadzorne ploče uživo
SOAP	HTTP/SMTP	XML	Usporiti	✓ WSDL	✗ Ne	Integracije s poduzećima, bankarstvom i naslijeđenim B2B sustavima
XML-RPC	HTTP	XML	Usporiti	✗ Ništa	✗ Ne	Naslijeđeni sustavi, WordPress API
JSON-RPC	HTTP/WS	JSON	Dobro	✗ Ništa	Ograničeno	Jednostavni RPC, Ethereum blockchain API
Događaji poslani s poslužitelja	HTTP	Tekst	Dobro	✗ Ništa	Poslužitelj→ Klijent	Feedovi uživo, nadzorne ploče (samo slanje putem poslužitelja)

06

Middleware

Kafka · RabbitMQ · Redis · API Gateway · Service Mesh

Middleware — ljepilo modernih sustava

„Softver koji omogućuje upravljanje podacima, komunikaciju i računalstvo između aplikacija na heterogenim računalnim platformama.“ — Omogućavanje interoperabilnosti od 1980-ih.

Posrednički softver za podatke

Povezuje baze podataka i aplikacije. JDBC, ODBC, JPA, Entity Framework, ADO.NET.

 *Hibernate ORM koji povezuje Spring Boot s PostgreSQL-om*

Objektno orijentirani middleware

RMI, CORBA, .NET Remoting. Prenos objekata između udaljenih sustava. Uglavnom zamijenjeno s REST/gRPC.

 *Java RMI: pozivanje metode na objektu koji se izvršava na drugom JVM-u*

RPC Middleware

Pozivi udaljenih procedura: XML-RPC, JSON-RPC, gRPC. Pozovite funkciju na udaljenom poslužitelju kao da je lokalni.

 *gRPC: Netflixovi pozivi između usluga s latencijom od mikrosekundi*

Transakcijski middleware

Osigurava ACID svojstva u distribuiranim sustavima. JTA (Java Transaction API), XA protokol.

 *Bankovni prijenos: tereti jedan račun i uplaćuje na drugi atomski preko dvije baze podataka*

Middleware orijentiran na poruke (MOM)

Asinkrono slanje poruka putem redova ili tema. Kafka, RabbitMQ, ActiveMQ, IBM MQ.

 *Narudžba poslana → događaj objavljen → zalihe, plaćanje, usluge e-pošte reagiraju neovisno*

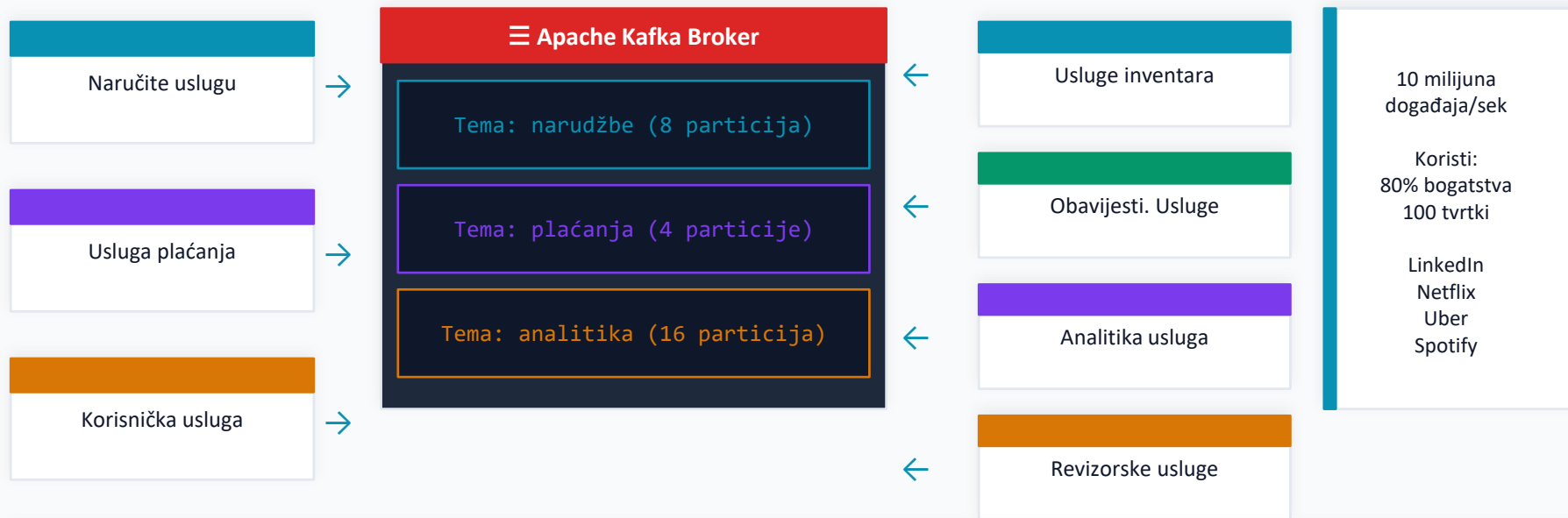
Aplikacijski / Portalni poslužitelji

JBoss/WildFly, WebLogic, WebSphere, IIS. Izvršavanje poslovnih aplikacija s web sučeljima.

 *Spring Boot ugrađeni Tomcat koji poslužuje REST API*

Apache Kafka — Streaming događaja u velikim razmjerima

Kafka je distribuirana platforma za strujanje događaja. Zamislite je kao zapisnik iz kojeg milijuni servisa istovremeno čitaju i u koji pišu.



💡 Ključne značajke: Distribuirano · Tolerantno na greške · Trajni zapisnik (ponovno reproducira događaje) · Grupe potrošača · Pravila zadržavanja · Semantika točno jednom

RabbitMQ

Tradicionalni posrednik poruka koji implementira AMQP. Podržava redove čekanja od točke do točke i teme objavljivanja/podnošenja.

- Podrška za AMQP protokol
- Razmjene + Redovi + Povezivanja
- Usmjeravanje, raspodjela, tematski obrasci
- Koristi se za: redove čekanja zadataka, e-poštu, webhookove

Kafka vs. Rabbit: Kafka = zapisnik streaminga (ponovno reproduciranje). RabbitMQ = red poruka (konzumiranje jednom).

API Gateway

Jedna ulazna točka za sve klijentske zahtjeve. Usmjerava prema mikroservisima, rješava probleme koji se odnose na više područja.

- Autentifikacija i autorizacija
- Ograničavanje i prigušivanje brzine
- Balansiranje opterećenja i usmjeravanje
- SSL prekid, zapisivanje zahtjeva
- Agregacija i keširanje odgovora

Primjeri: Kong, AWS API Gateway, NGINX, Traefik, Azure APIM

Service Mesh

Sloj infrastrukture za komunikaciju između usluga. Koristi bočne proxyje (poput Envoy-a) ubrizgane pored svake usluge.

- mTLS enkripcija između svih usluga
- Distribuirano praćenje i uočljivost
- Prekid strujnog kruga i ponovni pokušaji
- Podjela prometa (A/B, canary implementacije)
- Nisu potrebne promjene koda u uslugama

Primjeri: Istio, Linkerd, Consul Connect

Kako ovo funkcionira zajedno (Stak mikroservisa):



Resursi za učenje

► Statusno naspram bezstatnog

Jednostavno objašnjeno

<https://www.youtube.com/watch?v=oU6nEuLv-Z0>

► QUIC i HTTP/3

Husein Naser

<https://www.youtube.com/watch?v=xcrjamphIp4>

► gRPC u odnosu na REST

Vatrogasni brod (6 min)

<https://www.youtube.com/watch?v=gnchf0OjMk4>

► Usporedba API stilova

ByteByteGo

<https://www.youtube.com/watch?v=4vLxWqE9414>

► Apache Kafka za 6 minuta

ByteByteGo

<https://www.youtube.com/watch?v=R873B1NVUqQ>

► Distribuirani sustavi

Jednostavno objašnjeno

<https://www.youtube.com/watch?v=IJWwfMyPu1c>

► Objašnjenje HTTP/1 do HTTP/3

Vatrogasni brod (6 min)

<https://www.youtube.com/watch?v=a-sBfyiXysI>

► Kratki tečaj za REST API

Traversy Media

<https://www.youtube.com/watch?v=1sMQRaeKNDk>

► Objašnjenje GraphQL-a

Vatrogasni brod

<https://www.youtube.com/watch?v=yqpJiLyGUg8>

► Objašnjenje mikroservisa

TechWorld s Nanom

<https://www.youtube.com/watch?v=rvv4L1mLmVwk>

► Mreža usluga (Istio)

TechWorld s Nanom

<https://www.youtube.com/watch?v=QixK0B9Fh00>

► SOAP u odnosu na REST

IBM-ova tehnologija

<https://www.youtube.com/watch?v=1Xm16zqM0yg>

Usluge

Usluga pruža funkcionalnost putem definiranog sučelja. Bez stanja = skalabilno. Uz stanje = kada je kontekst važan.

Arhitekture

Odaberite pravu arhitekturu za svoju skalu. Mikroservisi ≠ uvijek bolji. Monolit za male timove. SOA za poduzeća.

Tehnologije

REST za jednostavnost. gRPC za performanse. GraphQL za fleksibilnost. WebSockets za rad u stvarnom vremenu. SOAP za naslijeđena poduzeća.

HTTP protokol

HTTP se razvio od jednostavnog tekstualnog protokola (0.9) do HTTP/3 protokola temeljenog na QUIC-u s multipleksiranjem, 0-RTT-om i ugrađenom sigurnošću.

Interoperabilnost

Bilo koji sloj može komunicirati s bilo kojim drugim slojem putem standardnih protokola. Java ↔ .NET ↔ Python — sve je moguće.

Middleware

Kafka = strujanje događaja u velikom obimu. RabbitMQ = tradicionalno čekanje. API Gateway = ulazna točka. Service Mesh = struktura za komunikaciju usluga.



Pitanja?

Interoperabilnost informacijskih sustava • Predavanje 03 • Usluge

Algebra Sveučilište Bernays