Otvoreno računarstvo

8. Web API, REST

- Web API
- REST
- RESTful API
- OpenAPI

Creative Commons

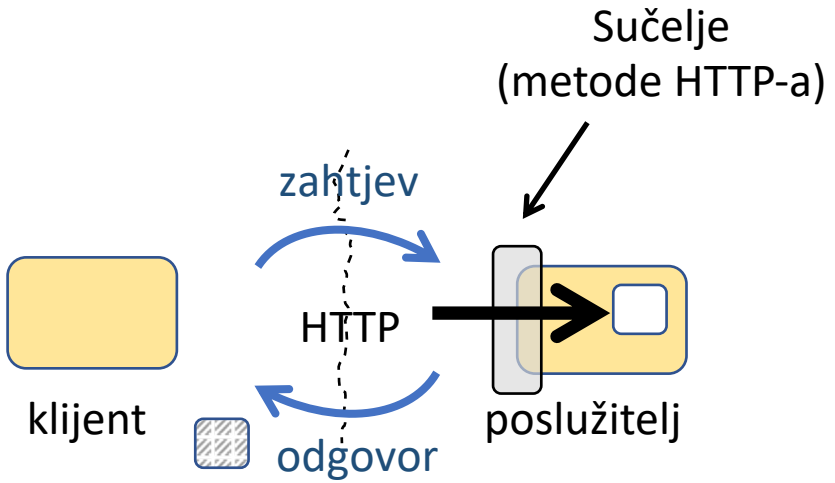
Otvoreno računarstvo

8. Web API, REST

- Web API
- REST
- RESTful API
- OpenAPI

Web i resursi

- **resurs**
  - sve što je dovoljno važno da se može referencirati
  - **adresirljivost -** svaki resurs mora imati svoj URI
  - **sučelje** definira moguće akcije nad resursom



- **reprezentacija resursa**
  - računalno čitljiv dokument koji sadrži informacije o resursu
  - može postojati **više od jedne** reprezentacije određenog resursa
  - klijent i poslužitelj **pregovaraju** o reprezentaciji
    (Accept: i Content-Type: polja zaglavlja, **media type**)

- HTTP zahtjev traži akciju nad resursom
- HTTP odgovor vraća (novu) **reprezentaciju** resursa

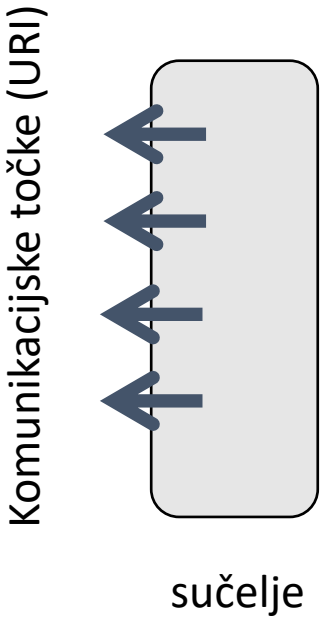
Resursi, akcije i reprezentacije

- Evolucija interneta
  - promjene u vrstama korisničkih agenata (mobilni uređaji, komunikacija m2m …)
  - promjene u vrstama i reprezentacijama posluživanih resursa
    - statički resursi -> dinamički resursi
    - *pripremljeni* resursi -> *sirovi* resursi
    - resursi -> usluge
    - html, css, png -> json, xml, …

- Ideja:
  - RMI, CORBA … suviše složeni za jednostavnije aplikacije (implementacija, komunikacija)
  - Potrebna slična platforma za dinamičku kompoziciju i komunikaciju između komponenti aplikacije
  - Iskoristiti postojeću jezično i plaformno-agnostičnu te skalabilnu infrastrukturu web-a (klijent-poslužitelj, zahtjev-odgovor, HTTP, tipovi reprezentacija, priručne kopije, zastupnici …) za pristup udaljenim resursima/uslugama
  - Podići razinu apstrakcije s razmjene HTTP poruka zahtjev-odgovor na paradigme korištene u programskim jezicima za razvoj komponenti aplikacija (slično zastupnicima i *stub/skeleton* u sustavima RPC, CORBA)

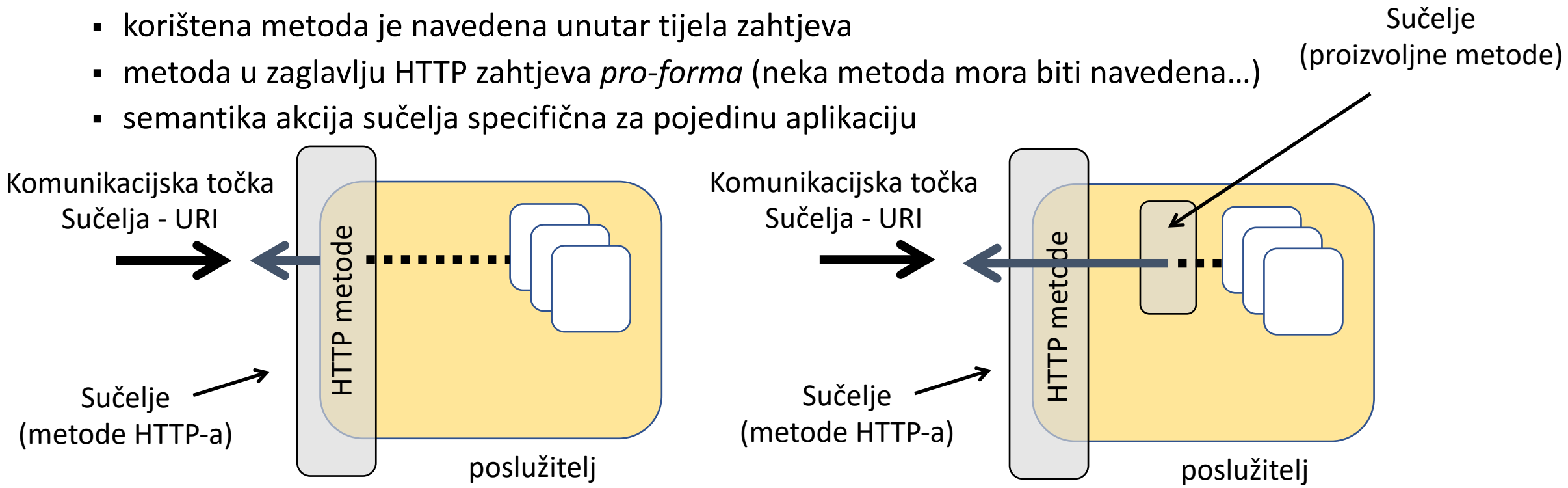
Web API

- Web API – skup logički povezanih komunikacijskih točaka koji pružaju uslugu
- Komunikacijska točka (engl. *communication endpoint*) definirana:
  - poslužiteljem – adresom/autoritetom
  - putom
  - dozvoljenim HTTP metodama
  - parametrima poziva
    - segment puta
    - upit
    - zaglavlje
    - tijelo upita (i očekivan format – *media type*)
  - odgovorom
    - status odgovora
    - zaglavlje odgovora
    - tijelo odgovora (u očekivanom formatu – *media type*)



Sučelje Web API-ja

- Sučelje (skup metoda/procedura):
  - Skup metoda ograničen na one definirane normom HTTP
    - metoda je navedena u zaglavlju HTTP zahtjeva (GET, POST, PUT, DELETE …)
    - približno očuvana semantika akcija za sva sučelja bez obzira na konkretnu aplikaciju
  - Proizvoljan skup metoda/procedura
    - korištena metoda je navedena unutar tijela zahtjeva
    - metoda u zaglavlju HTTP zahtjeva *pro-forma* (neka metoda mora biti navedena…)
    - semantika akcija sučelja specifična za pojedinu aplikaciju



Vrste implementacije sučelja API-ja

- Ad-hoc / aplikacijski specifični
  - ad-hoc definiran format poruka, akcije, formati parametara i odgovora
- RPC
  - Jednostavni *normirani* protokoli interakcije i formati parametara i odgovora
  - ad-hoc definirana sučelja
  - tipovi podataka vezani uz korišten jezik poruka (JSON, XML)
  - JSON-RPC i HTTP kao transportni protokol za JSON-RPC, XML-RPC
- *SOA*
  - SOAP, nastao na temelju XML-RPC, sada svijet za sebe
  - Usluge, valjane XML poruke, WSDL za opis usluga, dodatna infrastruktura (posrednici, repozitoriji …)

Implementacija API-ja: primjer ad-hoc implementacije

```
POST /knjige HTTP/1.0
Host: mojaknjiga.hr
Content-Type: text/xml
Content-length: …

<?xml version="1.0"?>
  <popisKnjiga poStranici=”20”>
    <kriteriji>
      <izdavač>Školska knjiga</izdavač>
    </kriteriji>
  </popisKnjiga>
```

Komunikacijska točka: /knjige

Pro-forma HTTP metoda

popisKnjiga – metoda sučelja komunikacijske točke aplikacijski specifična, ad-hoc definicija

Implementacija API-ja: primjer XML-RPC

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
  <methodCall>
    <methodName>examples.getStateName</methodName>
    <params>
      <param>
        <value>
          <i4>41</i4>
        </value>
      </param>
  </params>
</methodCall>
```

Komunikacijska točka: /RPC – za prihvat i izvođenje RPC poziva

Pro-forma HTTP metoda

Propisan format prijenosa informacija o pozivanoj metodi i parametrima

Izvor: http://xmlrpc.com/spec.md

Web API - klijenti

- tko su klijenti Web API-ja?
  - nisu (**samo**) preglednici Weba (za osobe)!
    - najčešće skripte, agenti...
  - nisu (**samo**) preglednici Weba (za osobna računala)!
    - mobiteli, tableti – *native aplikacije,* alternativni klijenti

- poslužitelj ne zna koji je cilj klijenta koji koristi API
  - alternativni alat (pokreće ga korisnik)
  - skripta – *crawler, monitor*
  - *agent*

- nema generičkih klijenata za API-je
  - krivi su poslužitelji, tj. dizajn API-ja
  - nema konzistentnosti – na desetke API-ja za sličnu svrhu

Životni vijek API-ja (I)

- problematično - API-ji su napravljeni za klijente!
  - mnoštvo implementacija klijenata
  - nepoštivanje koncepta hipermedija i samo-opisivanja
    - promjena načina uporabe API-ja se ne odražava u odgovorima poslužitelja
- kompatibilnost s prethodnim verzijama
  - ako nova verzija krši staru, stara treba ostati aktivna
- semantičko verzioniranje
  - *major.minor.patch*, npr. 1.0.0, 1.1.32, 2.0.5, 2.1.1
    - različite *major* verzije – nekompatibilna sučelja
    - različite *minor* verzije – viša verzija samo proširuje funkcionalnost nižeg sučelja
    - različite *patch* verzije – nema razlika u funkconalnosti i sučeljima, ispravke grešaka
- razdvajanje verzija
  - najčešće po URL-u pristupne točke
    - u imenu poslužitelja:
      - http://api-v1.mojsite.com http://api-v2.mojsite.com
    - u putu do početne točke:
      - http://api.mojsite.com/v1/ http://api.mojsite.com/v2/

Životni vijek API-ja (II)

- pristojno je dati obećanja
  - npr. *5 godina će raditi, krpanje rupa 2 godine…*
- primjeri koraka
  1. korak - proglasiti verziju zastarjelom (*deprecated*), ali još uvijek će raditi
  2. obavijestiti da prestaje krpanje rupa
  3. objaviti rok prestanka rada
  4. (možda) dodatni period & isključiti API
     - HTTP status code 410 (Gone) + objašnjenje + novi link

Otvoreno računarstvo

8. Web API, REST

- Web API
- REST
- RESTful API
- OpenAPI

REST

- REST ili **RE**presentational **S**tate **T**ransfer
  - Roy Fielding, 2000. g., doktorska disertacija
- stil programske arhitekture za izgradnju **raspodijeljenih** sustava
  - mogu se koristiti raznolike tehnologije i protokoli
- sustavi koji prate REST principe - "RESTful"
  - otvoreni
  - skalabilni
  - nadogradivi
  - jednostavni
  - *… (i sve ostale poželjne karakteristike)*

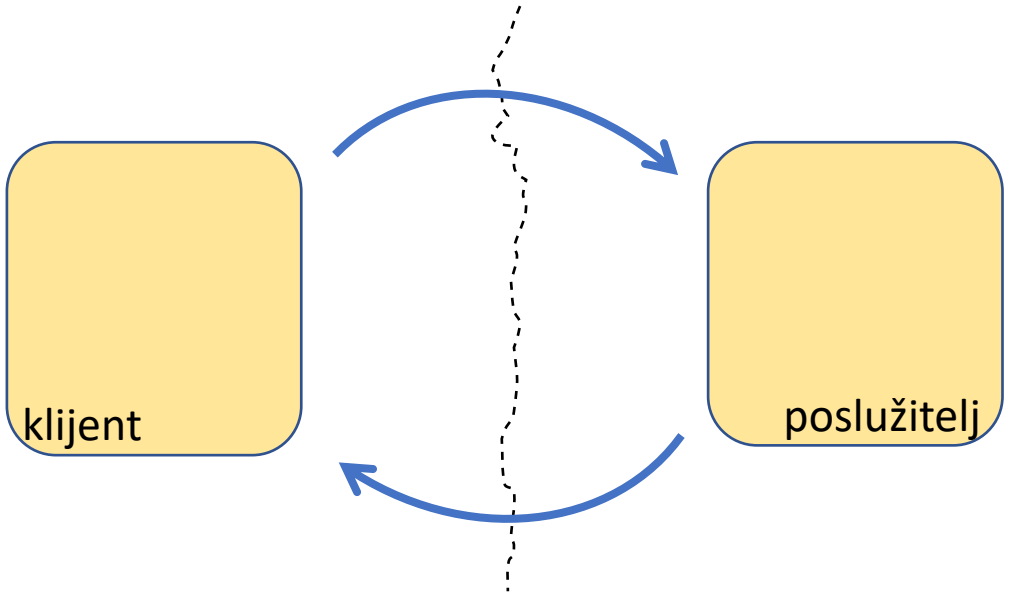
REST – ograničenja u oblikovanju API

- Samo pridržavanjem svih šest ograničenja omogućava se postizanje željenih nefunkcionalnih svojstava programskog sučelja aplikacije (jednostavnost, skalabilnost, prenosivost …)

1. Arhitektura klijent – poslužitelj
2. Jednoobrazno sučelje
3. Slojevitost sustava
4. Korištenje priručne memorije
5. Nepostojanje stanja u interakcijama
6. Proširenje funkcionalnosti klijenta programskim kodom

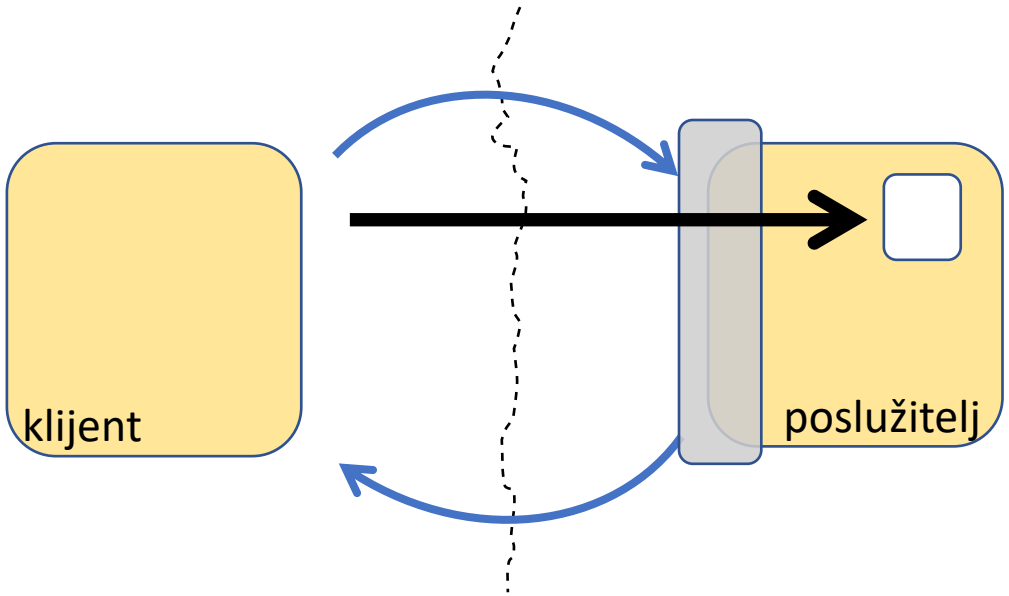
Arhitektura klijent - poslužitelj

- Razdvajanje odgovornosti između komponenata sustava
  - Prenosivost komponenti sustava na različite platforme
  - Skalabilnost komponenti sustava
  - Neovisan razvoj komponenti sustava (npr. preglednik razvijan neovisno o poslužitelju)



Jednoobrazno sučelje (I)

- Jednoobrazno sučelje:
  - Definira konačan broj istih operacija nad svim resursima u sustavu
  - Definira mehanizam identifikacije resursa koji je cilj tražene operacije
  - Predstavlja entitet *konektor* slojevitog modela
  - Omogućava interoperabilnost između komponenti (entiteta *slojeva*) sustava



sučelje – skup operacija nad resursima

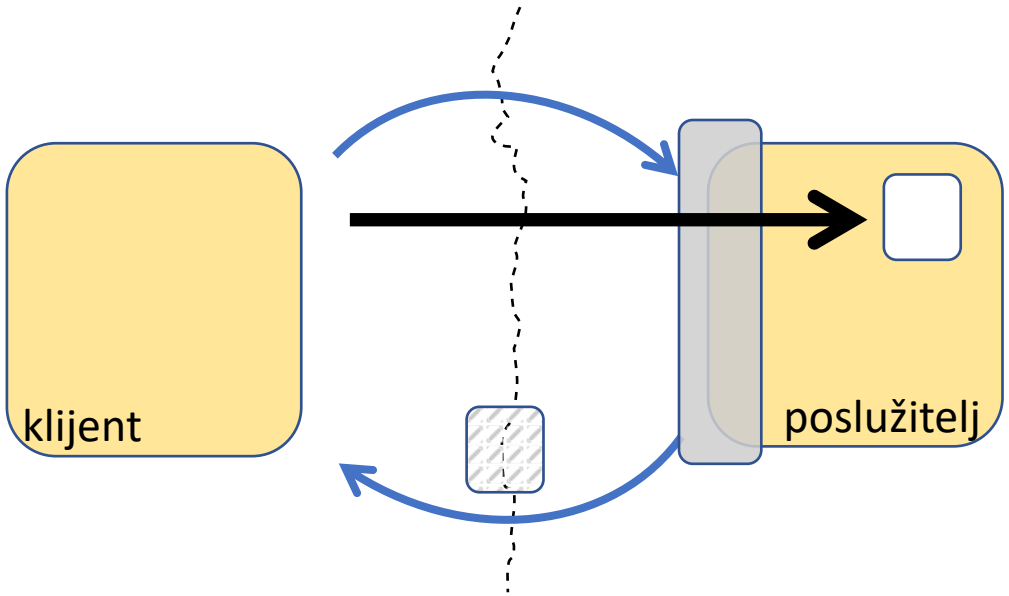
resurs

reprezentacija resursa

identifikator resursa

Jednoobrazno sučelje (II)

- Reprezentacija resursa:
  - Reprezentacija resursa na klijentu kao rezultat prethodne operacije nad resursom
  - Slika *stanja* resursa
  - Reprezentacija + meta podaci + operacija -> promjena stanja resursa



sučelje – skup operacija nad resursima

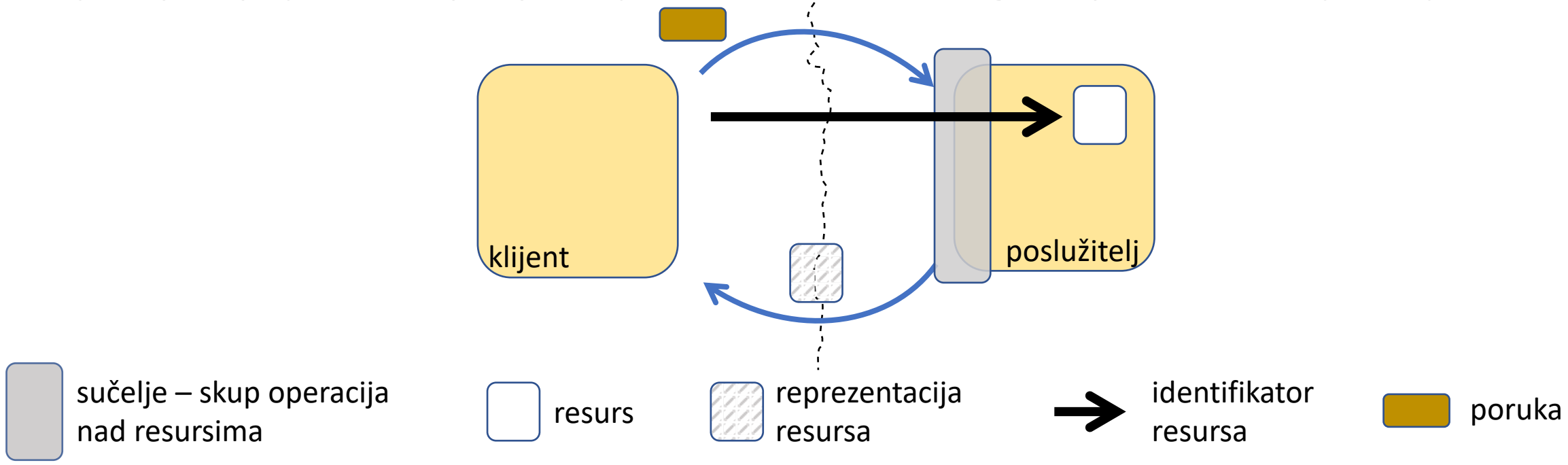
resurs

reprezentacija resursa

identifikator resursa

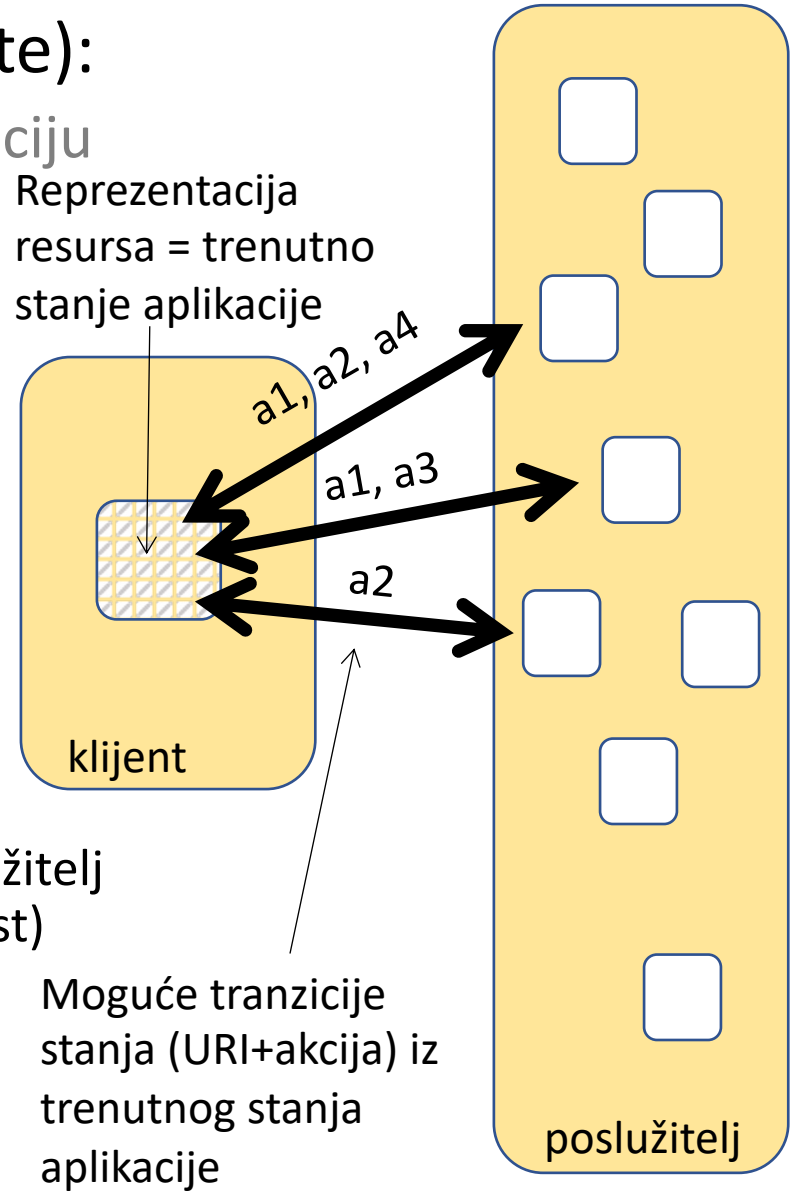
Jednoobrazno sučelje (III)

- Samo-opisne poruke:
  - Informacije sadržane u poruci dovoljne za potpunu obradu poruke
  - Posljedica: komunikacija između komponenti sustava je bez čuv*anja stanja (engl. stateless)*
  - Komponenta koja obrađuje zahtjev ne čuva stanje interakcije s klijentskom komponentom
  - Stanje interakcije klijentske s poslužiteljskom komponentom se čuva na strani klijenta i prosljeđuje poslužiteljskoj komponenti unutar svakog zahtjeva (samo-opisne poruke)



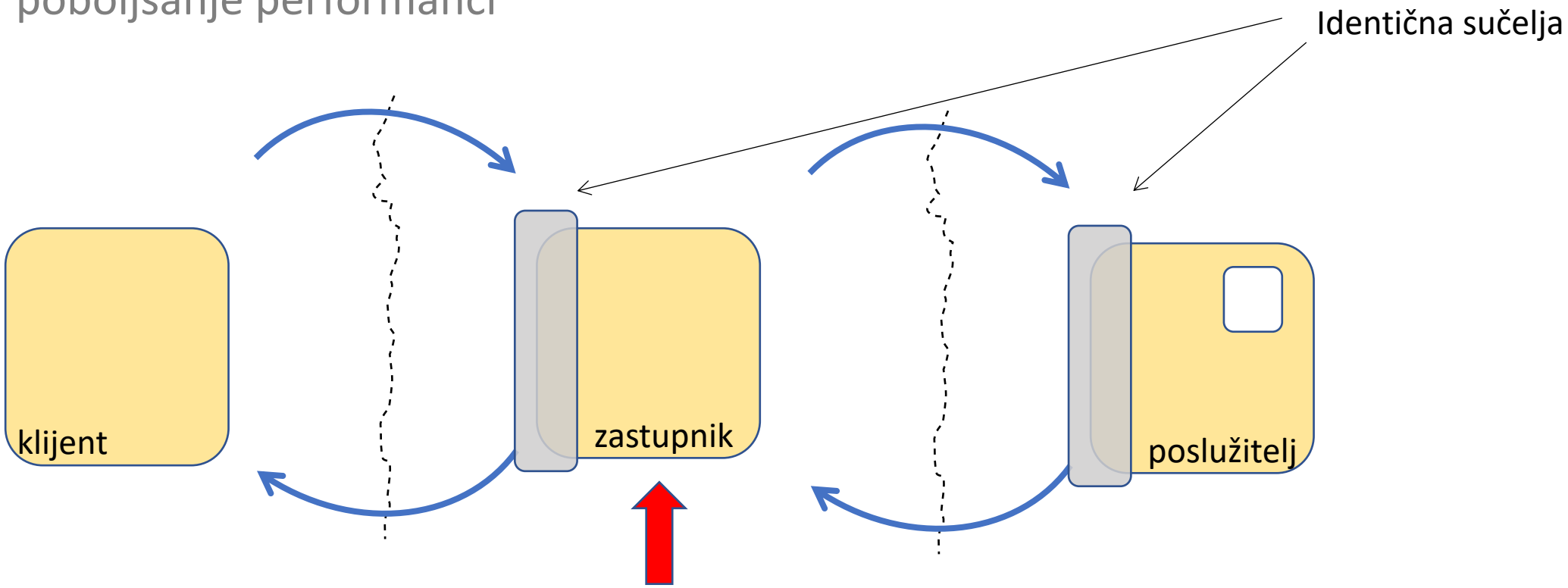
Jednoobrazno sučelje (IV)

- HATEOAS (Hypermedia as the engine of application state):
  - Svaka REST aplikacija izlaže samo inicijalni URI – *ulaz* u aplikaciju
  - Klijent ne podrazumijeva strukturu aplikacije (krajnje točke, akcije, parametre …) već se po strukturi aplikacije kreće dinamički – praćenjem ponuđenih veza i akcija definiranih unutar slike resursa kojom klijent raspolaže
  - Klijent upravlja promjenama stanja aplikacije s obzirom na:
    - reprezentaciju resursa kojom raspolaže (stanje aplikacije)
    - ponuđenim vezama prema drugim resursima i mogućim akcijama (tranzicije između stanja)
  - Posljedica:
    - Svaki klijent čuva stanje aplikacije za sebe (*hypermedia*) – a ne poslužitelj za sve aktivne klijente kao u primjeru korištenja sjednica (skalabilnost)
    - Klijent odlučuje (*engine*) o promjeni stanja aplikacije slijeđenjem mogućih tranzicija iz trenutnog stanja



Slojevitost sustava

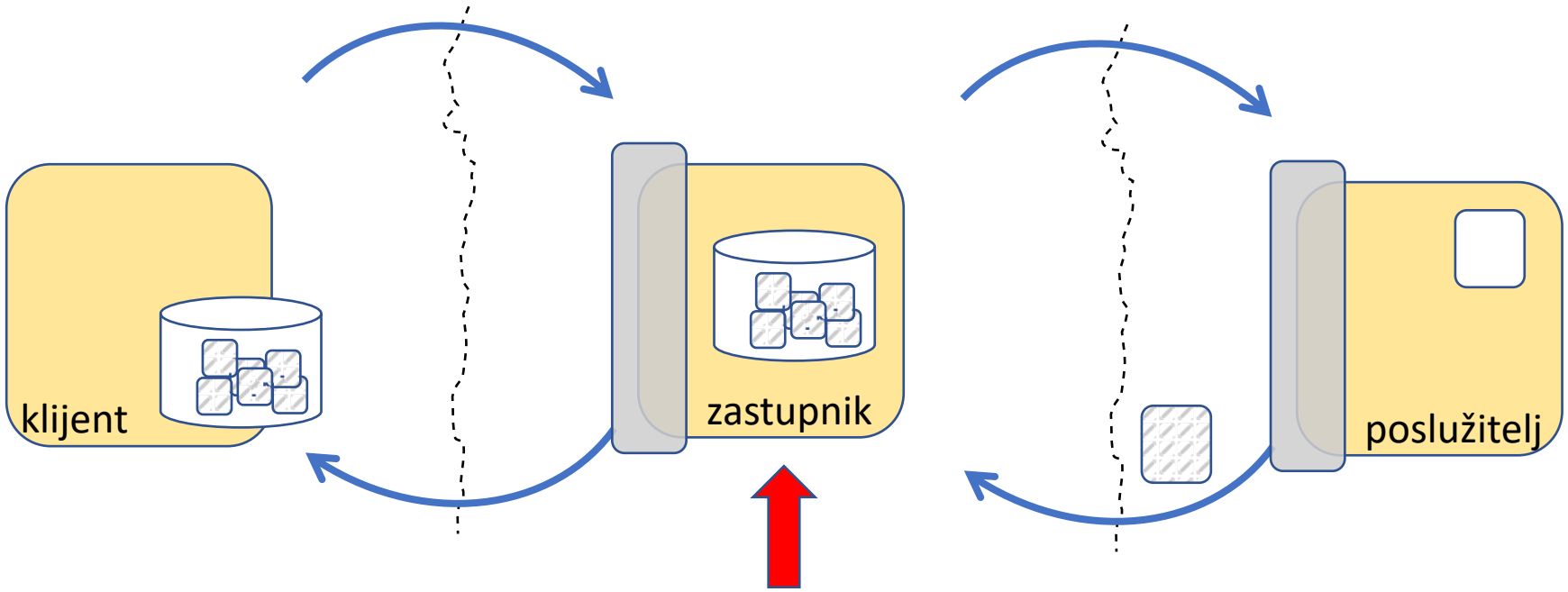
- Jednoobrazno sučelje omogućava jednostavno umetanje slojeva
  - Klijentska komponenta komunicira sa sljedećim slojem korištenjem istog protokola, akcija … transparentno za klijenta
  - Umetnuti slojevi - skalabilnost (*load balancing*, repliciranje komponenti, *cache*), poboljšanje performanci



Umetnut sloj između klijentske i poslužiteljske komponente

Priručne kopije

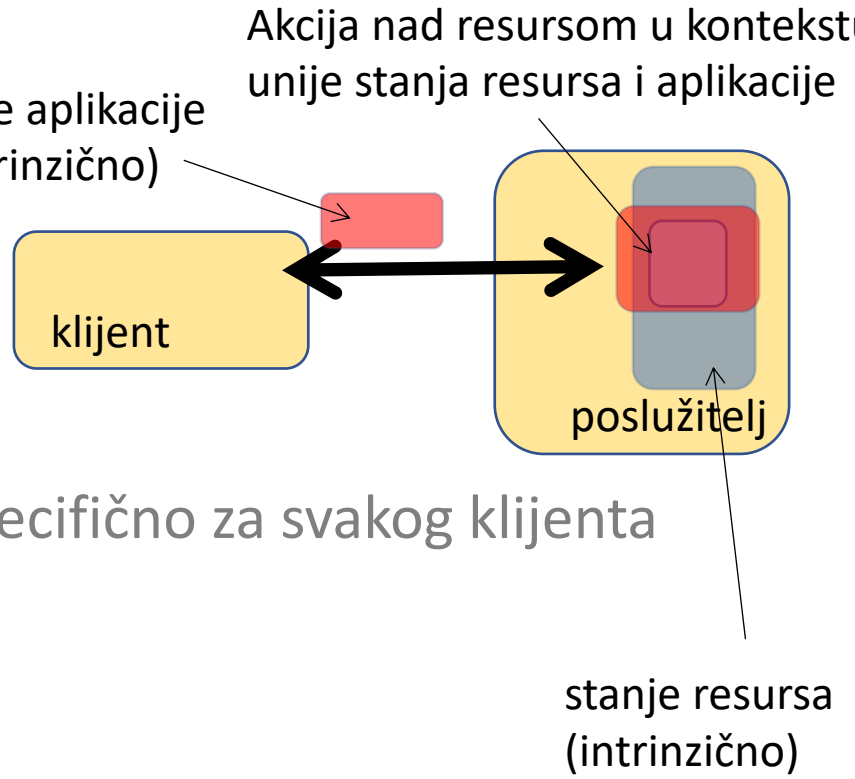
- Mogućnost čuvanja kopija reprezentacija resursa
  - Lokacije čuvanja: klijentska komponenta, komponente posrednici (zastupnici …)
  - Odgovori sadrže informaciju o *pohranjivosti* vraćene reprezentacije resursa
  - Povećanje performansi sustava izbjegavanjem komunikacije klijenta i poslužitelja



Umetnut sloj između klijentske i poslužiteljske komponente

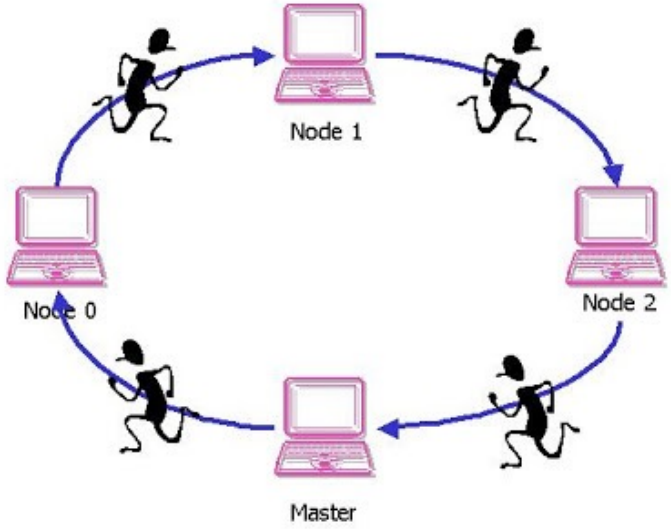
Nepostojanje čuvanja stanja aplikacije na poslužitelju

- Intrinzično stanje
  - stanje svih resursa na poslužitelju
  - pohranjeno na poslužitelju, dijele ga sve klijentske akcije
- Ekstrinzično stanje
  - stanje aplikacije (kako je vidi svaki klijent)
  - HATEOAS diktira pohranu stanja aplikacije na klijentu, specifično za svakog klijenta (nedjeljivo između klijenata)
- Akcija nad resursom na poslužitelju
  - Odvija se u kontekstu intrinzičnog i ekstrinzičnog stanja
  - Kod svakog zahtjeva (akcije) – klijent u sklopu zahtjeva prosljeđuje stanje aplikacije
- Posljedica:
  - Skalabilnost poslužitelja (nema čuvanja stanja N klijenata)
  - Jednostavno preusmjeravanje posluživanja zahtjeva na različite komponente sustava



Proširenje funkcionalnosti klijenta

- Privremeno proširenje funkcionalnosti korisničkog agenta (klijenta)
  - Prenošenjem i izvršavanjem programskog koda - Java Applets, JavaScript
  - Prenošenjem i koda i stanja - pokretni agenti
  - Značajan problem sigurnosti na strani korisničkog agenta
    - Pristup lokalnim resursima klijenta
    - Djelovanje prema drugim komponentama u ime korisničkog agenta



Otvoreno računarstvo

8. Web API, REST

- Web API
- REST
- RESTful Web API
- OpenAPI

Da li se Web temelji na principima REST? (I)

- **Klijent – poslužitelj arhitektura**
  - korisnički agent (preglednik weba, proxy …) i poslužitelj weba
  - MS Edge, Google Chrome, Apple Safari … – Apache, nginx …
- **Jednoobrazno sučelje**
  - **URI** – jedinstveni identifikator resursa
  - **HTTP (format + operacije) –** jedinstveno sučelje između komponenti sustava
  - Uniformne operacije nad resursima (**metode GET, POST, PUT, DELETE, PATCH, HEAD …**)
- **Reprezentacija resursa**
  - Niz okteta koji čine kodiranu reprezentaciju resursa + pripadni meta-podaci
  - Formati kodirane reprezentacije resursa: html, text, multipart, json, xml, png, mpeg …
  - Meta podaci:
    - Reprezentacije (*Content-type; encoding* …)
    - Resursa (*cacheable* …)

Da li se Web temelji na principima REST? (II)

- **Samo-opisne poruke**
  - protokol HTTP je bez pamćenja stanja (sjednice nisu originalni dio protokola)
- **HATEOAS**
  - *Ovisi o konkretnoj implementaciji aplikacije i sučelja (API)*
- **Slojevitost sustava**
  - Korisnički agent – posrednici (*proxy, firewall, load-balancer* …) – poslužitelj
- **Priručne kopije**
  - Na korisničkom agentu, na posredničkom entitetu, na poslužitelju …
  - Meta podaci resursa *Cache-Control* definiraju korištenje priručne kopije reprezentacije
- **Kod na zahtjev**
  - JavaScript

RESTful Web API

- HTTP (Web) API – koristi HTTP protokol
  - ne poštuje nužno API koncepte REST-a!

- **Konvencije za implementaciju REST principa u Web API**
  - **URI** su najčešći odabir za **imenice** (identificiraju resurse)
  - **metode HTTP-a** su najčešći odabir za **glagole** (konačan i uniforman broj operacija nad resursima - svedeno na metode HTTP-a - GET, HEAD, POST, PUT, DELETE, HEAD …)
  - Jedan URI kao „ulazna vrata” u API (agent ne smije pretpostavljati postojanje drugih osim osnovnog resursa i njegovog URI-ja)
  - Reprezentacije resursa sadrže informacije o mogućim daljnjim akcijama (dohvatljivim resursima i njihovom odnosu s reprezentacijom kojom korisnički agent trenutno raspolaže i raspoloživim akcijama nad tim resursima)
  - Korisnički agent koristi isključivo prethodno navedene informacije za kretanje po aplikaciji (pristupu resursima i obavljanju akcija nad njima)

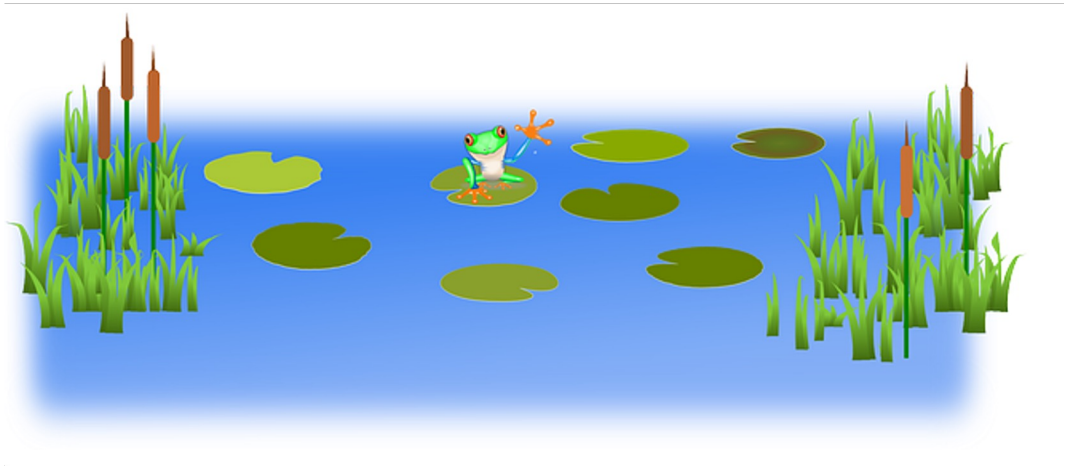
Web stranice / Web API / RESTful API?

- **Web API** *vs.* **svijet Web stranica**
  - koristite li pomoć za novoposjećenu Web stranicu?
  - prilagođavate li preglednik pojedinoj Web stranici?
  - hoće li promjena kôda Web stranice *skršiti* preglednik?

- Je li neki Web API ujedno i *RESTful* API?
  - modeli zrelosti
  - ***Richardson's maturity model*** *– 4 razine zrelosti*

Razine zrelosti API-ja

- **0. "The Swamp of POX" (*Plain Old XML)***
  - uporaba HTTP-zahtjeva kao tunela za poziv udaljenih procedura
  - metode GET i POST
  - upućivanje upita na jedan URI
  - proizvoljno generiranje URI-ova
  - poruke zapakirane u proizvoljni XML ili JSON



Razina 0 - primjer

Pozivi samo na jedan endpoint
URI ne identificira resurs
-ne koriste se imenice

Akcija (glagol) proizvoljno definirana, prenosi se unutar proizvoljno definiranog formata zapisa u tijelu zahtjeva

Reprezentacija resursa u proizvoljnom formatu, ne koriste se hiperveze za definiranje mogućih akcija (tranzicija stanja aplikacije)

http://api.mojsite.com

POST /narudzbe HTTP/1.1

```
<zahtjevProvjeri datum= "2014-06-01" restoran= "FinaHrana"/>
```

HTTP/1.1 200 OK

```
<listaDostupnihTermina>
  <termin početak="1800" />
  <termin početak="2000" />
</listaDostupnihTermina>
```

http://api.mojsite.com

POST /narudzbe HTTP/1.1

```
<zahtjevRezerviraj datum= "2014-06-01" restoran= "FinaHrana"
      početak="1800" korisnik="Pero" />
```

HTTP/1.1 200 OK

```
<rezervacija id="314" datum= "2014-06-01" restoran= "FinaHrana"
      početak="1800" korisnik="Pero" />
```

Razine zrelosti API-ja

- **1. Resursi**
  - imenice
  - više mjesta kojima se upućuju zahtjevi
    - razdvajanje jednog velikog servisa u više malih
    - URI označava **pojedini resurs**

Razina 1 - primjer

URI identificira resurs (korištenje imenica)

Akcija (glagol) proizvoljno definirana, prenosi se unutar proizvoljno definiranog formata zapisa u tijelu zahtjeva

Reprezentacija resursa u proizvoljnom formatu, ne koristi se URI moguće akcije agenta
Pozitivno: koriste se identifikatori uz pomoć kojih je moguće formirati URI resursa - imenice)

http://api.mojsite.com
POST **/restorani/FinaHrana** HTTP/1.1
<zahtjevProvjeri datum= "2014-06-01" />

HTTP/1.1 200 OK
<listaDostupnihTermina>
<termin id="12345678" početak="1800" />
<termin id="56781234" početak="2000" />
</listaDostupnihTermina>

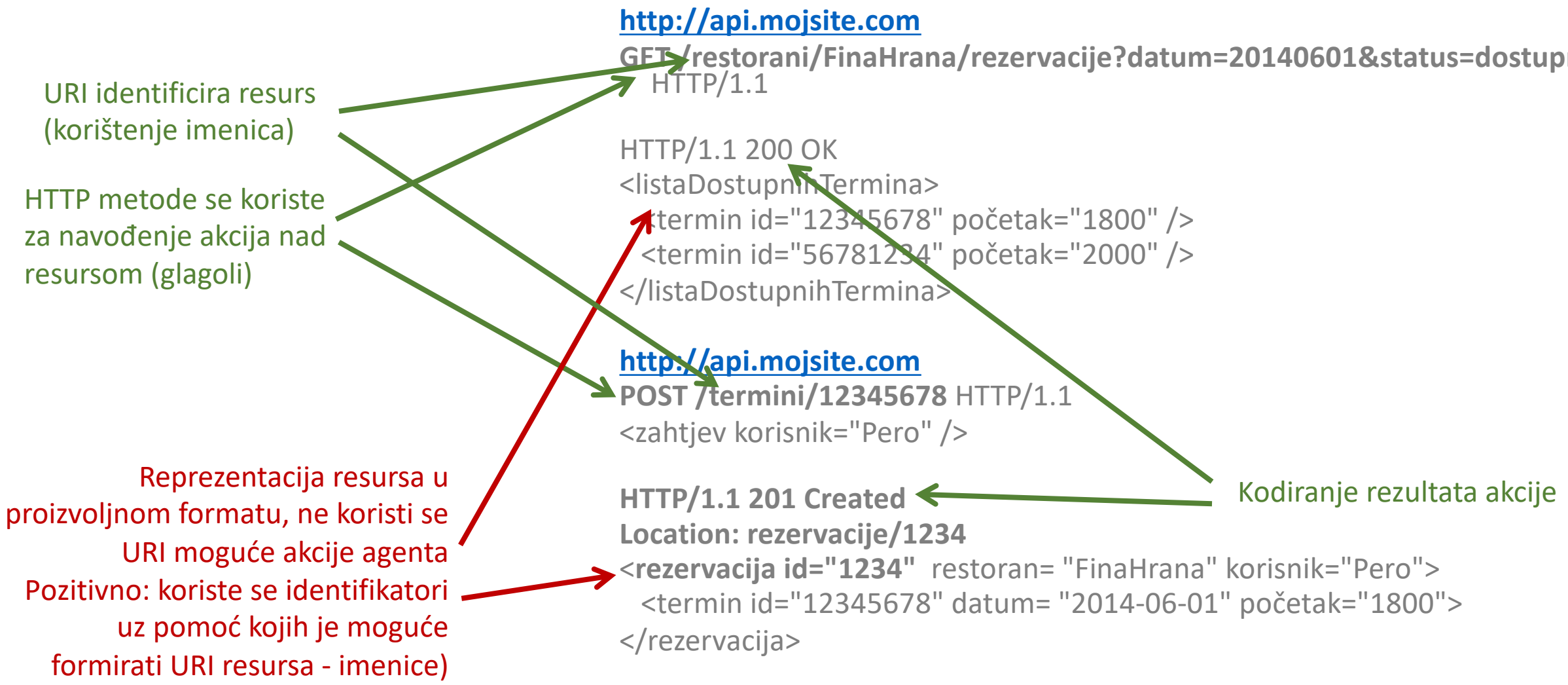
http://api.mojsite.com
POST **/termini/12345678** HTTP/1.1
<zahtjevRezerviraj korisnik="Pero" />

HTTP/1.1 200 OK
<**rezervacija id="1234"** restoran= "FinaHrana" korisnik="Pero">
<termin id="12345678" datum= "2014-06-01" početak="1800">
</rezervacija>

Razine zrelosti API-ja

- **2. Glagoli – metode HTTP-a**
  - metoda označava vrstu radnje koju želimo obaviti
  - GET, POST, PUT, DELETE, (PATCH)...
    - svaka metoda ima svoje specifičnosti
  - slične situacije obavljamo istim glagolom
    - npr. dodavanje komentara, dodavanje poruke – POST...
  - pravilna uporaba HTTP kôdova rezultata (*status codes*)

Razina 2 - primjer



Razine zrelosti API-ja

- **3. Hipermedijske kontrole –** upravljački elementi
  - konačno, "pravi" REST
    - točnije, uvjet za REST po originalnoj definiciji
  - HATEOAS :-)
    - poveznice koje se nalaze u dobivenom odgovoru pružaju upute o sljedećim koracima
    - klijent svojim odabirom mijenja stanje aplikacije
  - samo-opisne poruke, smanjuje se utjecaj dokumentacije

Razina 3 - primjer

http://api.mojsite.com

GET /restorani/FinaHrana/rezervacije?datum=20140601&status=dostupno HTTP/1.1

HTTP/1.1 200 OK

```
<listaDostupnihTermina>
  <termin id="12345678" početak="1800" >
    <link rel="rezerviraj" uri="/termini/12345678" />
  </termin>
  <termin id="56781234" početak="2000" />
    <link rel="rezerviraj" uri="/termini/56781234" />
  </termin>
</listaDostupnihTermina>
```

HATEOAS

rel – odnos akcije naspram reprezentacije resursa

url – resurs povezan s trenutnim resursom
(tj. njegovom reprezentacijom na agentu)

Razina 3 - primjer

http://api.mojsite.com

POST /termini/12345678 HTTP/1.1

<zahtjev korisnik="Pero" />

HTTP/1.1 201 Created

Location: rezervacije/1234

<**rezervacija id="1234"** restoran= "FinaHrana" korisnik="Pero">

<termin id="12345678" datum= "2014-06-01" početak="1800">

<link rel="self" uri="/rezervacije/1234" />

<link rel="otkaži" uri="/rezervacije/1234" />

<link rel="dodajNapomenu" uri="/rezervacije/1234/napomene" />

<link rel="promijeniTermin" uri="/restorani/FinaHrana/rezervacije?
datum=20140601&status=dostupno" />

<link rel="promijeniKontakt" uri="/korisnici/Pero" />

</rezervacija>

HATEOAS

Dostupne akcije ovisne su o stanju resursa - poslužitelj resursa odlučuje koje akcije su na raspolaganju agentu korisniku
(korisnik Marko možda ne bi dobio isti popis akcija)

Semantika metoda HTTP-a

- Nismo više samo u svijetu preglednika Weba
  - preglednici Weba (obrasci) – GET i POST
- **GET**
  - dohvaćanje reprezentacije resursa
  - lokacija zadana u URI-u
  - može zahtijevati određeni format reprezentacije resursa (XML, JSON …)
- **POST**
  - u užem smislu: **stvaranje novog resursa**
  - u širem smislu: različite promjene, zapisivanje na poslužitelj, služi "za sve"
  - primjer: *dodaj novi komentar na restoran="FinaHrana"*
    - još **nije poznata buduća lokacija** resursa, to rješava poslužitelj
  - HTTP odgovori:
    - 201 (Created), zaglavlje Location: URL novog resursa
    - 202 (Accepted)

Metode HTTP-a

- **PUT** – zamjenjuje stanje resursa novim stanjem opisanim u danoj reprezentaciji
  - može služiti i za dodavanje novog resursa
  - stavlja se na lokaciju **točno određenu URI-jem!**
    - za razliku od POST-a, gdje lokaciju određuje poslužitelj
  - HTTP-odgovori
    - 200 (OK) – *zamijenjeno, postavljeno*
    - 204 (No Content) – *tamo nije bilo tog resursa*
- **PATCH** – promjena dijela jednog podatka
  - tekstualni podaci
    - analogija: CVS/SVN/Git...
    - dodatak HTTP-u, (*još)* nije službena metoda, sve češća
- **DELETE** – brisanje resursa na definiranoj lokaciji

Svojstva metoda HTTP-a

- **GET, HEAD - nullpotentne** operacije
  - **sigurna** metoda (*safe method*), ne mijenja stanje resursa
- **PUT, DELETE - idempotentne** operacije
  - svaka sigurna metoda je i idempotentna
  - obrat ne vrijedi
- **POST nije** idempotentan
    - primjer: "dodaj novi komentar na restoran"

Veze

- odnosi među poveznicama (*link relations*)
- nema službene norme kako definirati veze :-(
- RFC8288, IANA Link Relation Types
- postoje definirane riječi
  - registracijsko tijelo: IANA
  - ako je potrebno, moguće je dodati (i opisati) vlastite

    - <**link rel="edit"** href="http://api.mojsite.com/rezervacije/214"

    - <**link rel="next”** href="http://api.mojsite.com/rezervacije/215">

Formati poruka API-ja

- **Internet media type** (bivši MIME type)
- Najzastupljeniji formati zapisa podataka:
  - JSON
  - XML
- Nerješeno pitanje podrške za **hipermedijske elemente**
  - kako implementirati HATEOAS, tj. kako u danom formatu zapisa navesti moguće tranzicije stanja aplikacije?
- XML
  - ima podršku za hipermedij: Xlink – XML Linking Language
  - Uporaba odumire – prevladava JSON

Formati poruka API-ja

- JSON
  - Prevladava kao format zapisa podataka kod korištenja API-ja
  - Problemi s osnovnim JSON-om:
    - **nema hipermedijsku podršku**
    - nema koncepta “veze” ili sličnog tipa podatka kao HTML (ili XML)
    - URI se tretira kao “obični” niz znakova
  - Implementirana semantika aplikacije **nije ponovno iskoristiva** (za drugi API)
    - ne postoji norma
    - parsiranje svakog novog API-ja ispočetka

JSON primjer


```
[
  {
    "place_id":"3677303151",
    "licence":"Data \u00a9 OpenStreetMap contributors, ODbL 1.0. http:\/\/www.openstreetmap.org\/copyright",
    "osm_type":"way",
    "osm_id":"201559182",
    "lat":"45.8007791",
    "lon":"15.9714059256088",
    "display_name":"Faculty of Electrical Engineering and Computing, 3, Unska ulica, Martinovka, Trnje, Zagreb, City of Zagreb, 10124, Hrvatska",
    "class":"amenity",
    "type":"university",
    "importance":0.51436424356534,
    "icon":"http:\/\/nominatim.openstreetmap.org\/images\/mapicons\/education_university.p.20.png"
  }
]
```

Zapis veza u JSON-u?

- Nije definirano

```
{
"_links" : [
        { "rel" : "self",
          "href" : "http://example.org/foo/bar" },
        { "rel" : "next",
          "href" : "http://example.org/foo/bar/2" }
        ]
}
{
  "hyperlink": [
      {"self": "http://example.org/foo/bar"},
      {"next": "http://example.org/foo/bar/2"}
   ]
}
```

Otvoreno računarstvo

8. Web API, REST

- Web API
- REST
- RESTful API
- OpenAPI

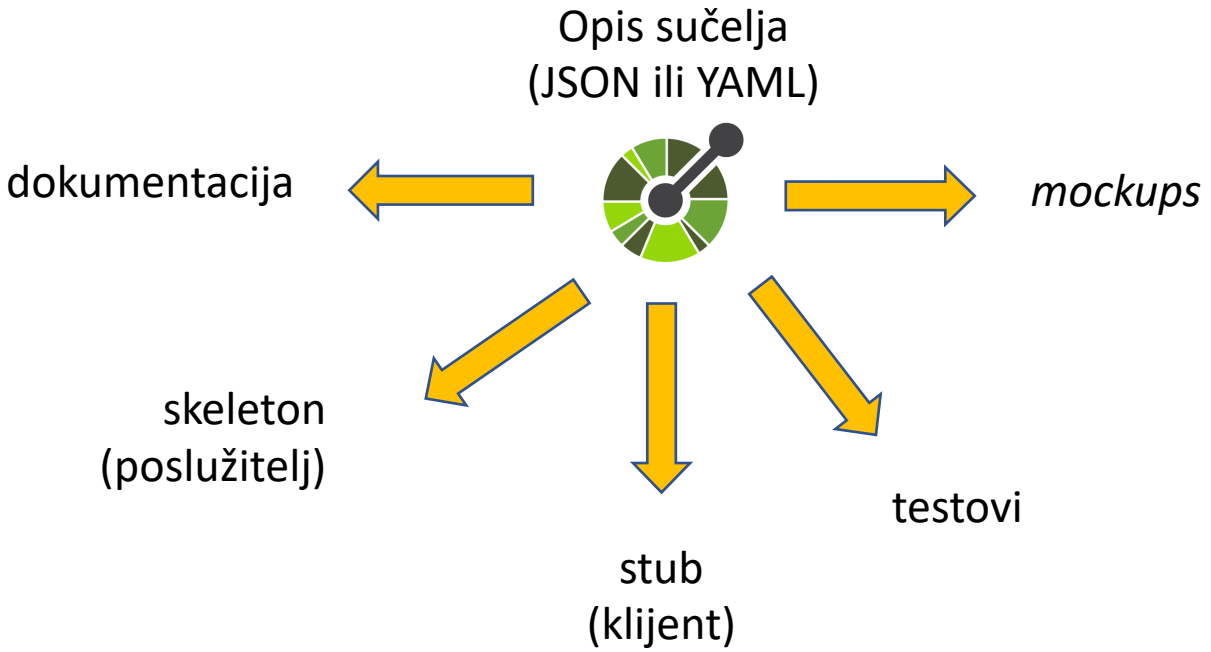
OpenAPI Inicijativa

- OpenAPI – format za opis REST sučelja
- OAI – OpenAPI Initiative (Linux Foundation)
- OpenAPI temeljen na Swagger Specification, donirano *open source* zajednici
- Članovi inicijative:

Google ebay IBM Microsoft

SAP ORACLE Atlassian ...

Korištenje OpenAPI-ja



- Skup alata više različitih dobavljača (besplatni, os, komercijalni alati)
  - generiranje programskog koda za različite jezike i platforme
  - generiranje automatiziranih testova za različite testne okvire
  - automatizirano stvaranje *dummy* (*mockup*) implementacija API za testiranje klijenata
  - generiranje ljudski i strojno čitljive dokumentacije

OpenAPI opis sučelja

- Opis sučelja sastoji se od:
  - skupa krajnjih točaka komunikacije
  - na krajnjim točkama podržanih akcija (HTTP metoda)
  - ulaznih i izlaznih parametara za pojedinu krajnju točku i akciju
  - metoda autentikacije
  - meta-podataka o sučelju (verzija norme, licence, opisi, …)
- Jezik opisa:
  - YAML i JSON
- Više detalja i upute:
  - https://swagger.io/docs/specification/about/