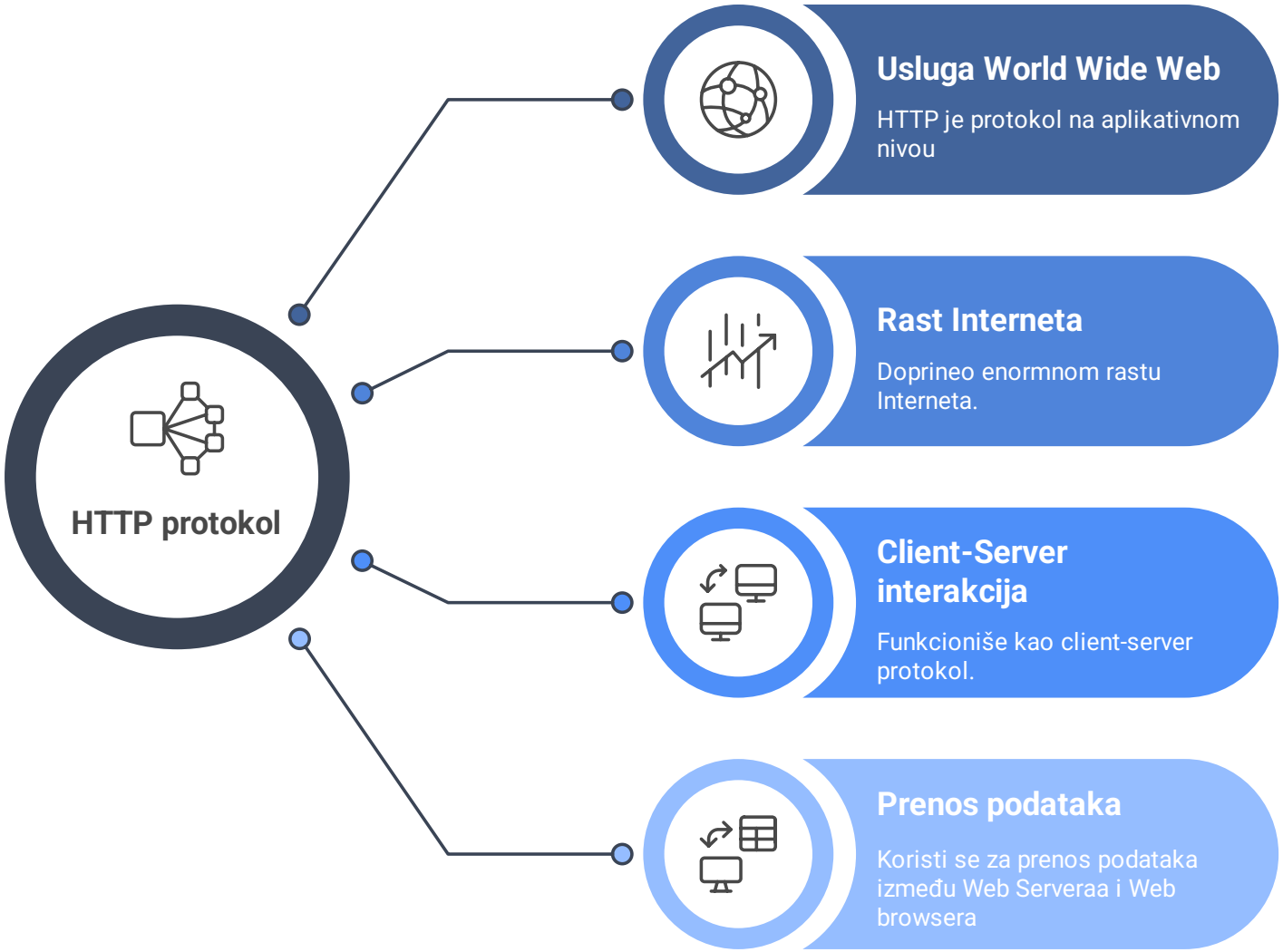
ARHITEKTURA WEB APLIKACIJA

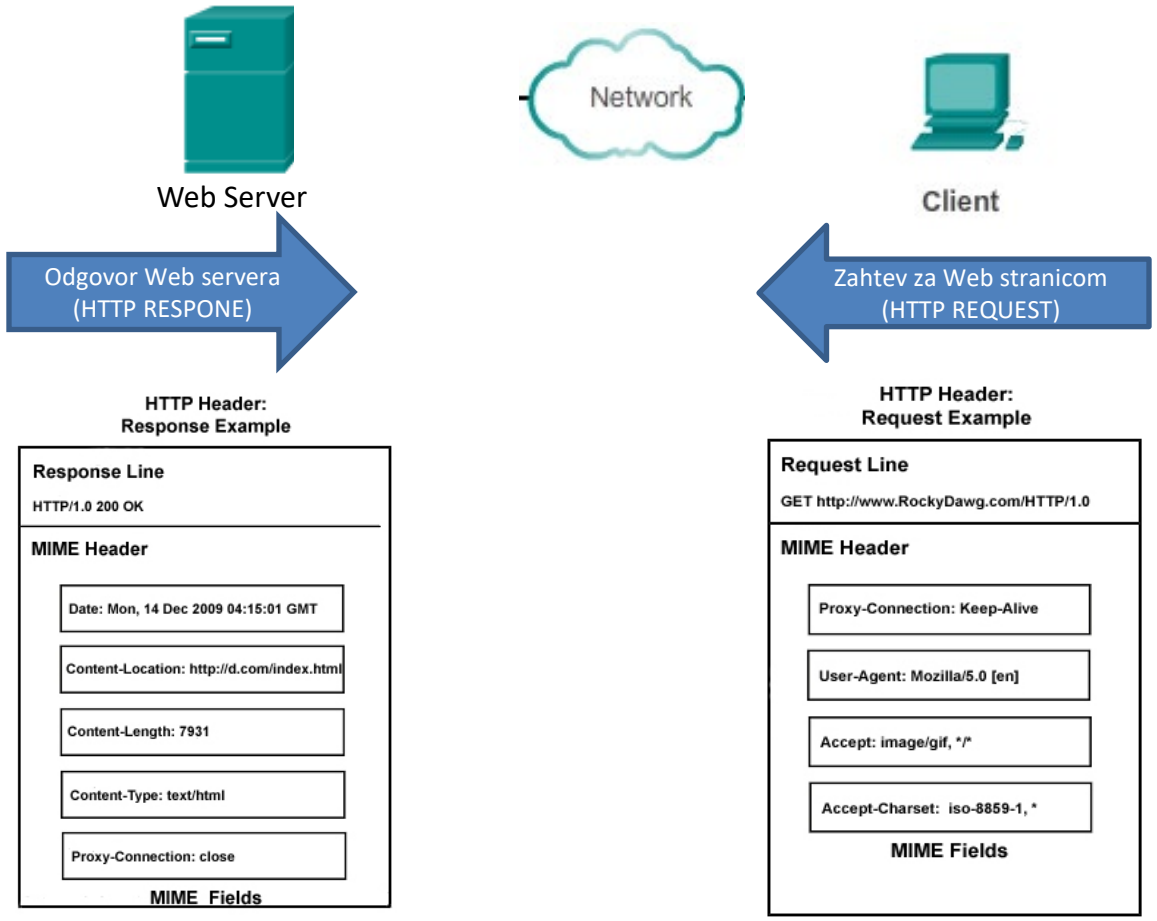
Predmet: Bezbednost aplikacija

Predavač: dr Dušan Stefanović

Hypertext Transfer Protocol (HTTP)



Hypertext Transfer Protocol (HTTP)



Web aplikacije dizajn i arhitektura

Unos podataka

Korisnik unosi podatke putem web browser-a kroz frontend modul aplikacije

Prikaz podataka

Frontend prikazuje sadržaj i omogućava interakciju

HTTP komunikacija

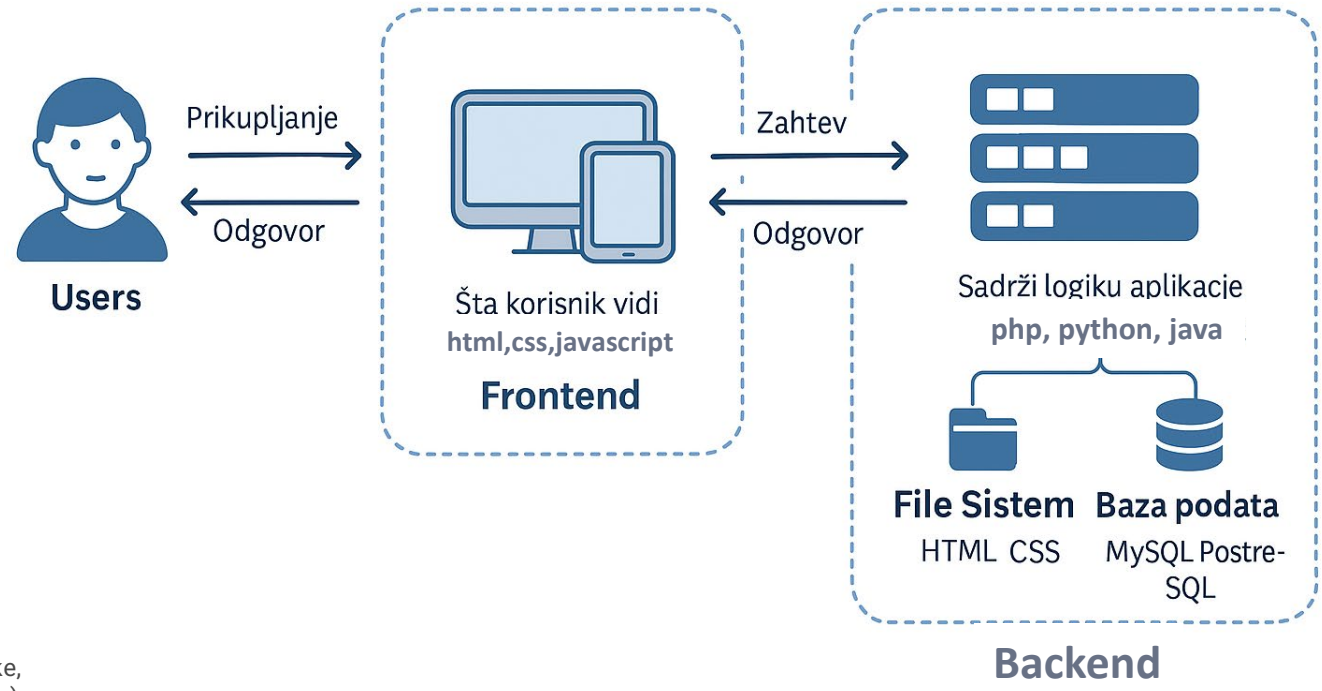
Klijent šalje zahtev, server odgovara

Poslovna logika

Backend obrađuje podatke i pristupa resursima kao što su file sistem (npr. slike, fajlovi) i bazi podataka (MySQL, MariaDB,..).

Arhitektura

Jasna separacija između prikaza (frontend) i obrade podataka (backend).
Omogućava modularnost, skalabilnost i bezbedniji rad aplikacija.



SADRŽAJ KOJI VRAĆA WEB SERVER

HTML stranica

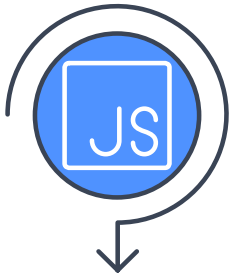
Standardni format web stranica.

Slike

Vizuelni sadržaj za web sajtove.

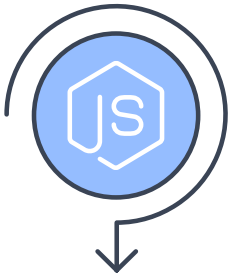
CSS

Pravila stilizovanja za web stranice.



JavaScript

Programski jezik za interaktivnost.



JSON

Format podataka za web aplikacije.

STANDARDNI I BEZBEDNI PROTOKOLI ZA WEB KOMUNIKACIJU

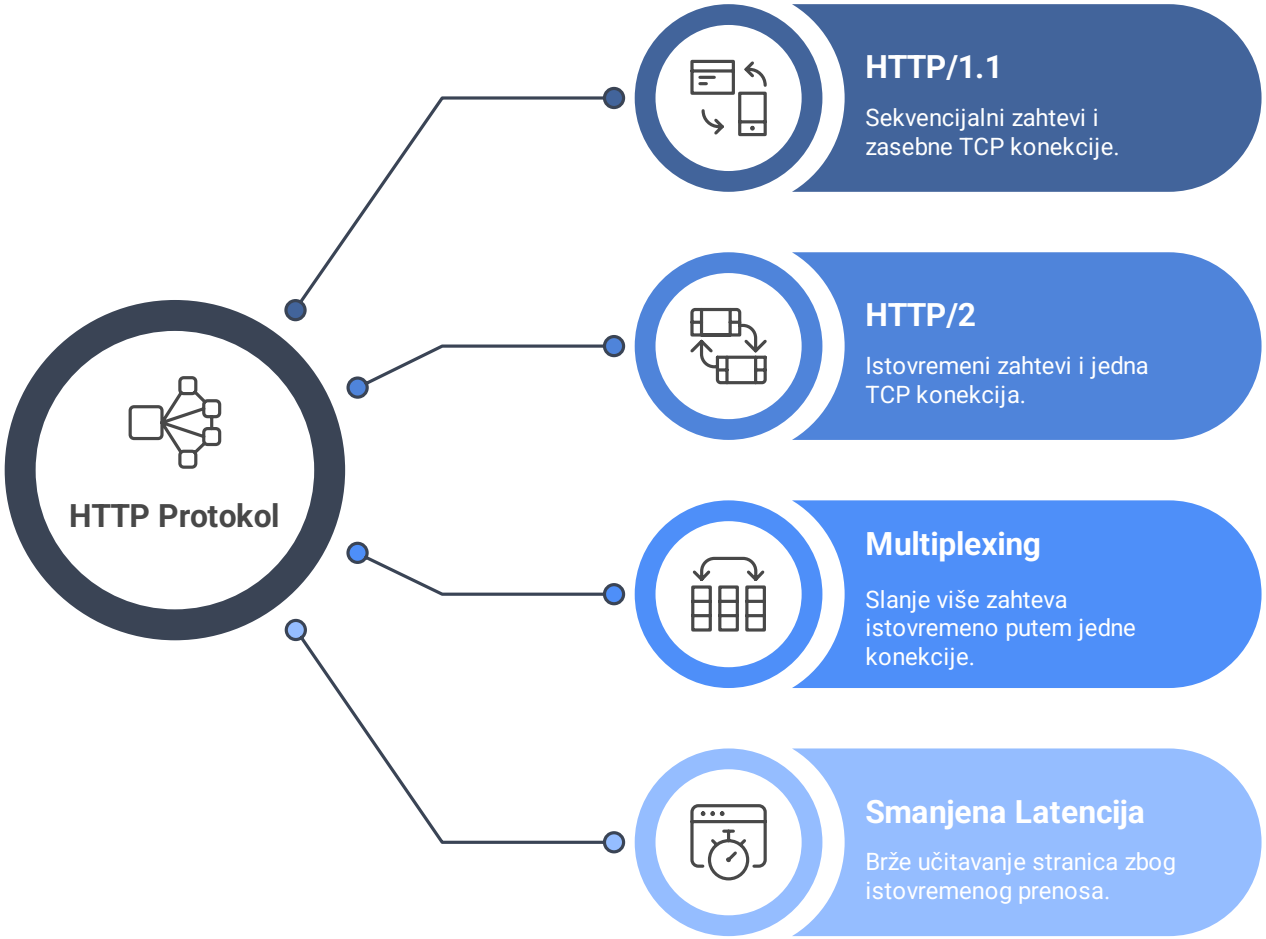
HTTP

Standardni TCP port 80 za web saobraćaj koji nije kriptovan

HTTPS

Bezbedan TCP port 443 sa ugrađenom TLS enkripcijom

RAZVOJ HTTP PROTOKOLA



HTTP/1.1	HTTP/2
Otvara konekciju i šalje jedan po jedan GET zahtev.	Otvara konekciju i šalje više GET zahteva paralelno.
Svaki resurs mora da sačeka prethodni da bi se poslao.	Server može da pošalje više odgovora paralelno.
Klijent čeka sve odgovore pre nego što renderuje stranicu.	Klijent brže dobija sve podatke i renderuje sadržaj.

HTTP REQUEST PORUKA (ZAHTEV WEB SERVERU)

```
GET /~srt/ HTTP/1.1
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; SLCC1; .NET
   CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.0.04506; InfoPath.1)
Host: www.vtsnis.edu.rs
Connection: Keep-Alive
```

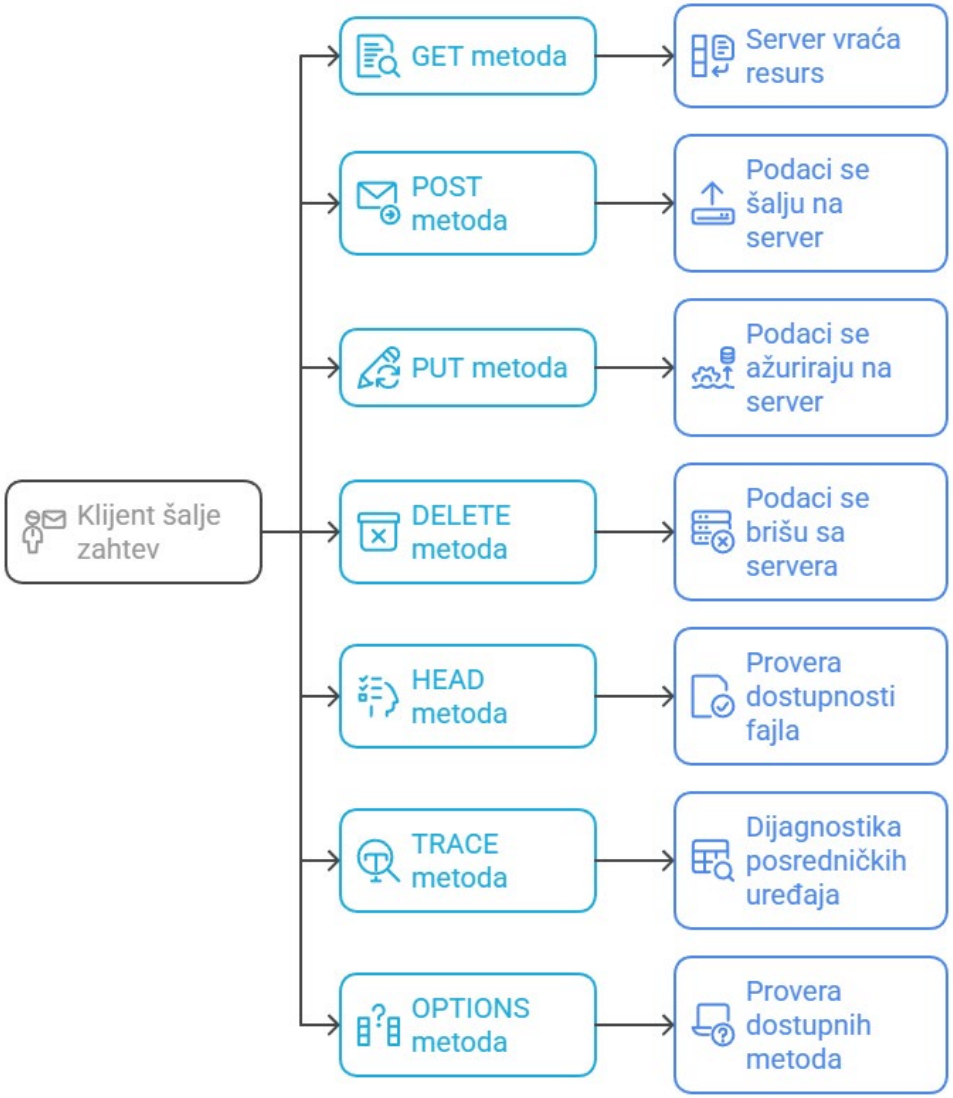
Request

GET - Browser/klijent traži Web stranu

/~srt/ - Browser traži Web stranu u direktorijumu (index.html)

HTTP/1.1 – Browser koristi verziju HTTP/1.1

HTTP REQUEST KOMANDE (METODE I NJIHOVA UPOTREBA)



HTTP REQUEST KOMANDE (GET METODA)

GET je HTTP metoda kojom se od web servera traži određeni resurs

Prikazivanje stranice

Koristi se za dostavljanje web stranice, koju web browser učitava

Dostavljanje liste podataka

Koristi se za dostavljanje liste podataka, kao što su proizvodi.

Učitavanje API podataka

Koristi se za učitavanje API podataka sa određenog API endpointa.

HTTP REQUEST KOMANDE (POST METODA)

POST je HTTP metoda koja se koristi za **slanje novih podataka** serveru (npr. registracija, unos komentara).

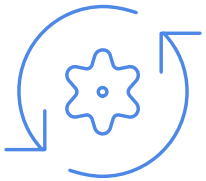
```
POST /korisnici HTTP/1.1
Host: sajt.com
Content-Type: application/json

{
  "ime": "Mina",
  "email": "mina@sajt.com"
}
```

Kreiranje naloga

Koristi se prilikom kreiranja novog korisničkog naloga.

Slanje podataka

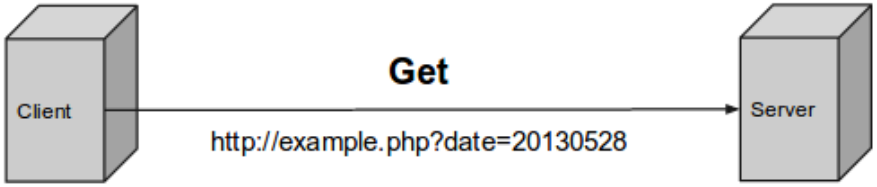
Koristi se prilikom slanja poruke, forme ili fajla.

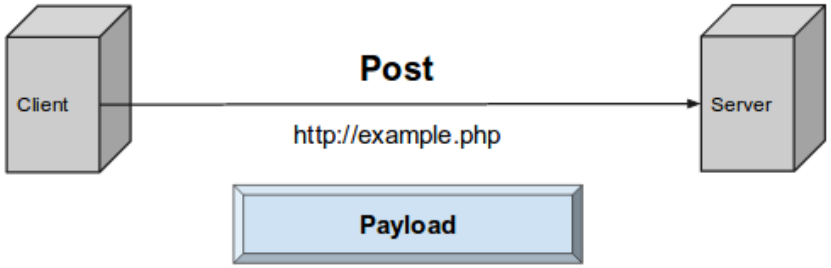


Kompleksna akcija

Koristi se prilikom inicijalizacije kompleksne akcije (Klijent šalje korpu sa više artikala, server kreira porudžbinu, proverava stanje, rezerviše robu, generiše račun.

HTTP REQUEST KOMANDE (GET vs POST)





GET metoda

može da se iskoristi i za prosleđivanje podataka na web server.

Ne preporučuje se jer se ti podaci nalaze u URL adresi tj. lako su vidljivi ako se npr radi o korisničkom imenu ili lozinki.

Prihvatljiva je za slanje vrednosti koje treba primeniti za filtriranje podataka iz baze podataka.

POST metoda

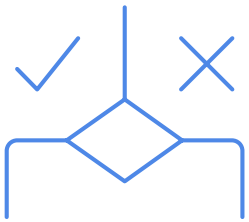
je način da se prosledi podatak preko forme na klijentskoj strani web serveru tj. bazi podataka radi provere autentifikacije ili upisa korisnika u bazu podataka.

Bezbedniji je način od get metode jer se vrednosti nalaze u zaglavlju HTTP poruke.

HTTP REQUEST KOMANDE (PUT METODA)

Metoda PUT se koristi radi kreiranja ili zamene resursa na tačno određenoj URI lokaciji.

Karakteristike PUT zahteva

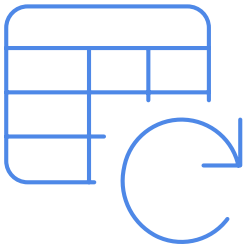


Idempotentnost

Višestruki identični PUT zahtevi imaju isti efekat.

Telo zahteva

Sadrži podatke koji se postavljaju na server (npr. JSON, XML, tekst).



Zamena resursa

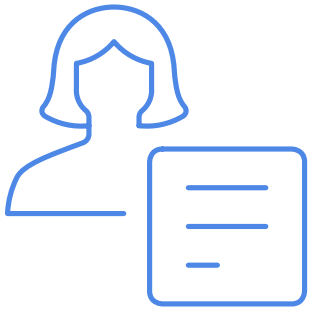
Ceo resurs se menja, za razliku od PATCH metode.

URI je obavezan

Klijent mora znati tačan URI resursa koji se ažurira ili kreira.

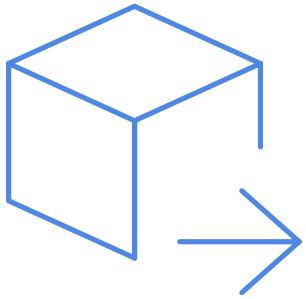
HTTP REQUEST KOMANDE (PUT METODA)

Tipična upotreba PUT metode:



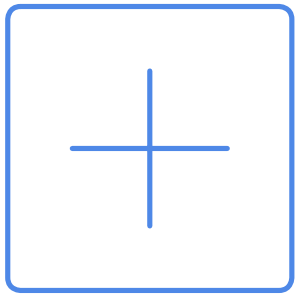
Ažuriranje korisničkog profila

Ažurira korisnički profil koristeći PUT /users/123



Zamena objekta proizvoda

Zamenjuje ceo objekat proizvoda koristeći PUT /products/456

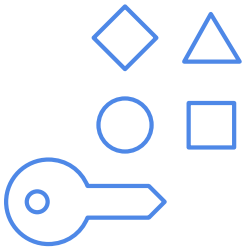


Kreiranje resursa

Kreira resurs ako ne postoji. Server može podržavati upsert logiku.

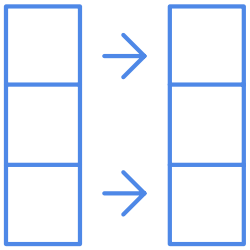
HTTP REQUEST KOMANDE (PUT METODA)

PUT naspram POST metode



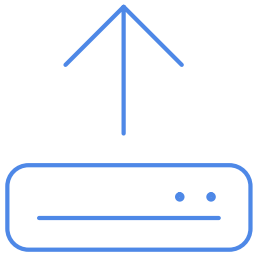
Funkcija

PUT ažurira ili kreira resurs.
POST kreira novi resurs.



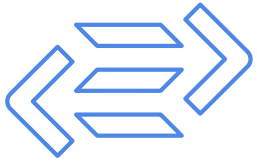
Idempotentnost

PUT je idempotentan,
POST nije.



URI

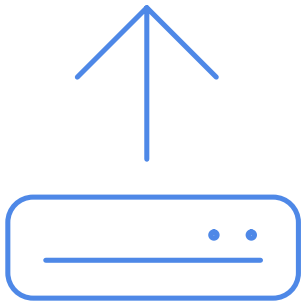
PUT zahteva poznat URI.
URI za POST određuje server.



Upotreba

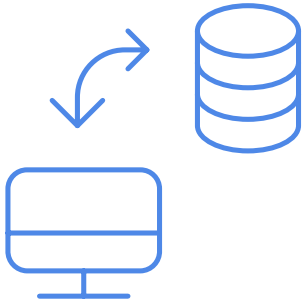
PUT se koristi kao `/users/123`.
POST se koristi kao `/users`.

HTTP REQUEST KOMANDE (PUT METODA - ZAKLJUČAK)



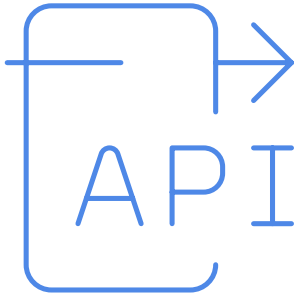
PUT metoda

PUT je strožiji, klijent zna URI i šalje resurs.



POST metoda

Server određuje lokaciju i način kreiranja resursa.

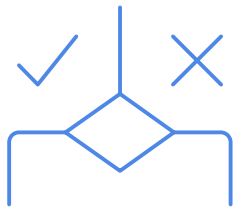


REST API dizajn

Nisu svi serveri podešeni da omogućavaju kreiranje resursa PUT metodom – to zavisi od REST API

HTTP REQUEST KOMANDE
(DELETE METODA - KARAKTERISTIKE)

Metoda DELETE se koristi za brisanje resursa na serveru koji je identifikovan zadatim URI-jem.



Idempotentnost

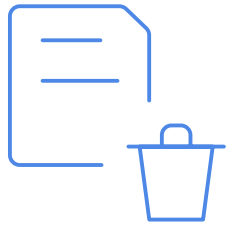
Brisanje istog resursa više puta ima isti rezultat.

Telo zahteva

Obično se ne koristi, mada standard dozvoljava.

Odgovor servera

`200 OK`, `202 Accepted`, ili `204 No Content` ako je uspešno.



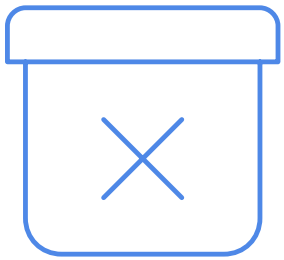
Garancija brisanja

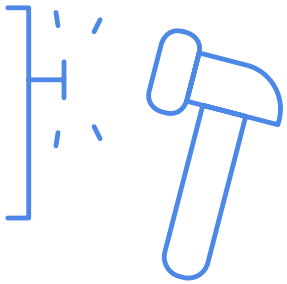
Ne garantuje trajno brisanje. Server može implementirati soft-delete (npr. obeležavanje kao obrisano).

HTTP REQUEST KOMANDE
(DELETE METODA - Zaključak)

Kada korisnik ili sistem treba da ukloni podatke (npr. korisnika, komentar, proizvod).

U REST API dizajnu za operacije brisanja entiteta prema resource/id.





Telo zahteva

DELETE metoda ne zahteva telo zahteva i ne očekuje telo u odgovoru.

Idempotentnost

DELETE zahtevi su idempotentni, ali mogu imati posledice (npr. gubitak podataka)..

Autorizacija

DELETE se može koristiti zajedno sa autorizacijom da se ograniči pristup.

HTTP REQUEST KOMANDE (HEAD METODA)

HEAD je HTTP metoda koja funkcioniše **isto kao GET**, ali **ne vraća telo odgovora**, već **samo zaglavlje (headers)**

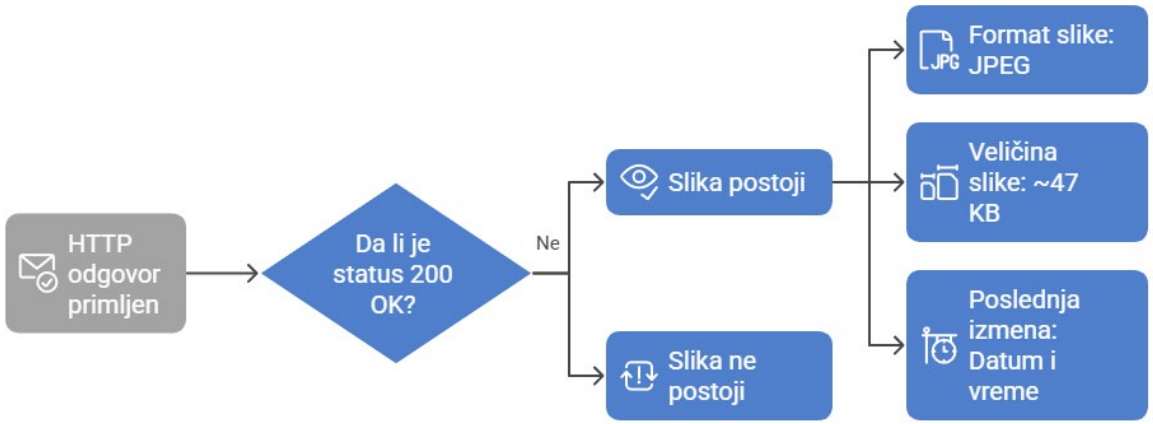
HTTP ZAHTEV:

```
HEAD /slika.jpg HTTP/1.1
Host: www.primersajt.com
```

HTTP ODGOVOR:

```
HTTP/1.1 200 OK
Date: Mon, 12 May 2025 08:00:00 GMT
Content-Type: image/jpeg
Content-Length: 47384
Last-Modified: Sat, 10 May 2025 19:20:00 GMT
```

ZAKLJUČAK NA OSNOVU HTTP ODGOVORA:



HTTP REQUEST KOMANDE (HEAD METODA)

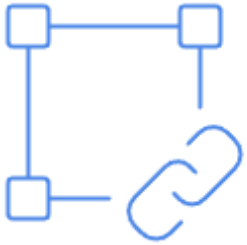
HEAD metoda se koristi za:

Postojanje resursa

Proverava postojanje resursa tj. da li fajl ili API endpoint postoji bez preuzimanja samog fajla

Dostupnost fajla

Proverava da li je fajl dostupan.



Testiranje linkova

Koristi se za testiranje linkova na stranici.

Provera keš verzije

Pomoću zaglavlja (ETag, Last-Modified) odlučuje se da li treba da se učita nova verzija.

HTTP REQUEST KOMANDE (TRACE METODA)

TRACE metoda se koristi za dijagnostiku komunikacije između klijenta i servera.

Kada klijent pošalje TRACE zahtev, server treba da vrati tačan zahtev nazad — onako kako ga je primio, uključujući zaglavlja.

Verifikacija zahteva

Da bi se proverilo da li je zahtev stigao do servera bez izmena.

Izmene na ruti

Da bi se otkrile izmene na ruti (proxy serveri).

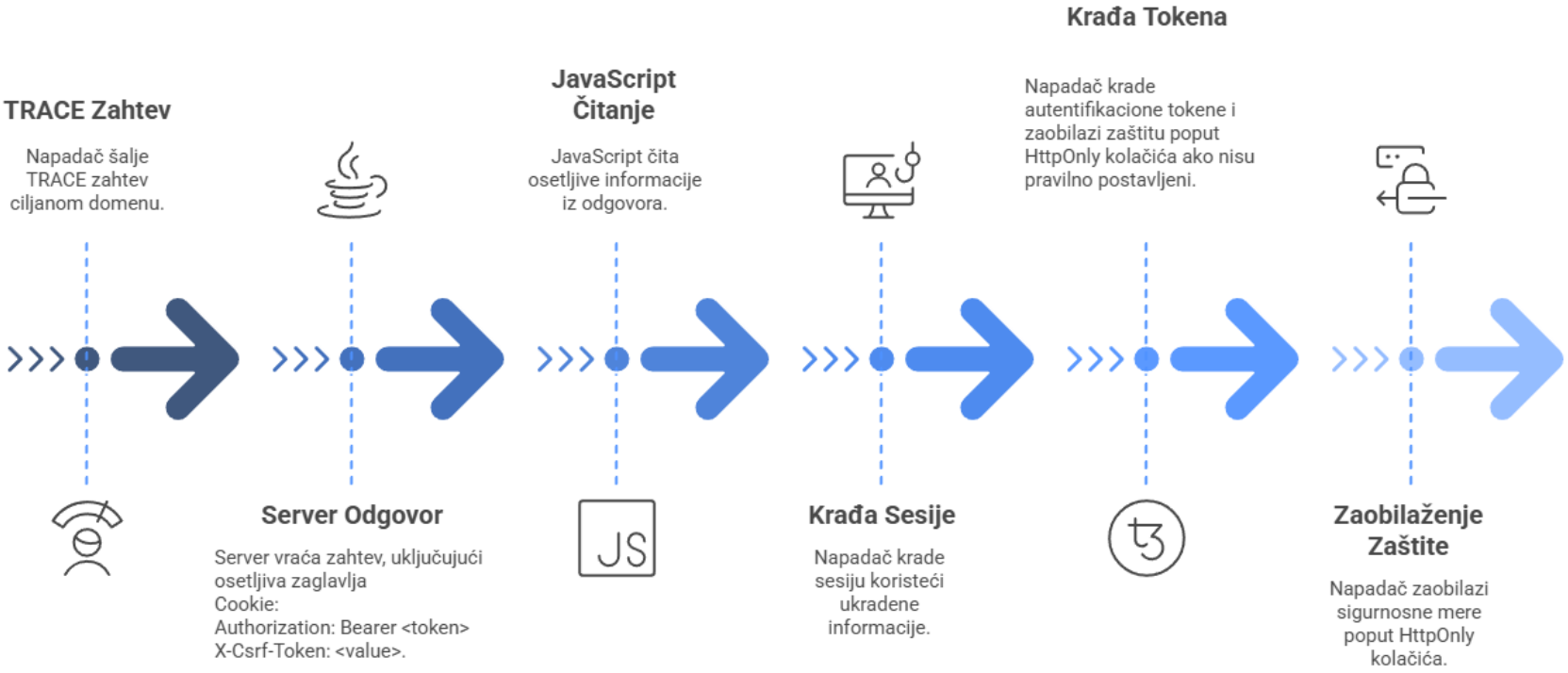
Identifikacija problema

Da bi se identifikovali problemi sa mrežom ili HTTP zaglavljima.

HTTP REQUEST KOMANDE
(TRACE METODA - Cross-Site Tracing Napad)

TRACE metoda može biti zloupotrebljena u XST (Cross-Site Tracing) napadima — kada se koristi za krađu kolačića (cookies) ili tokena preko JavaScript-a.

Zbog toga mnogi moderni serveri onemogućavaju TRACE metodu.



HTTP REQUEST KOMANDE

(TRACE METODA – Posledice Cross-Site Tracing napada)

HTTP REQUEST KOMANDE (TRACE METODA)

TRACE metoda nije zlonamerna sama po sebi, ali je rizična zato što "poslušno" vraća sve što primi – uključujući kolačiće.

Zato se najčešće onemogućava u realnim okruženjima.

Onemogućiti TRACE metodu

U većini slučajeva se ne koristi u produkciji.

HttpOnly flag za kolačiće

Sprečava pristup JavaScript-u,, čak i ako se vrati u telu odgovora.

Onemogućiti echo zaglavlja

Web serveri se mogu konfigurisati da ne vraćaju osetljive parametre iz zaglavlja.

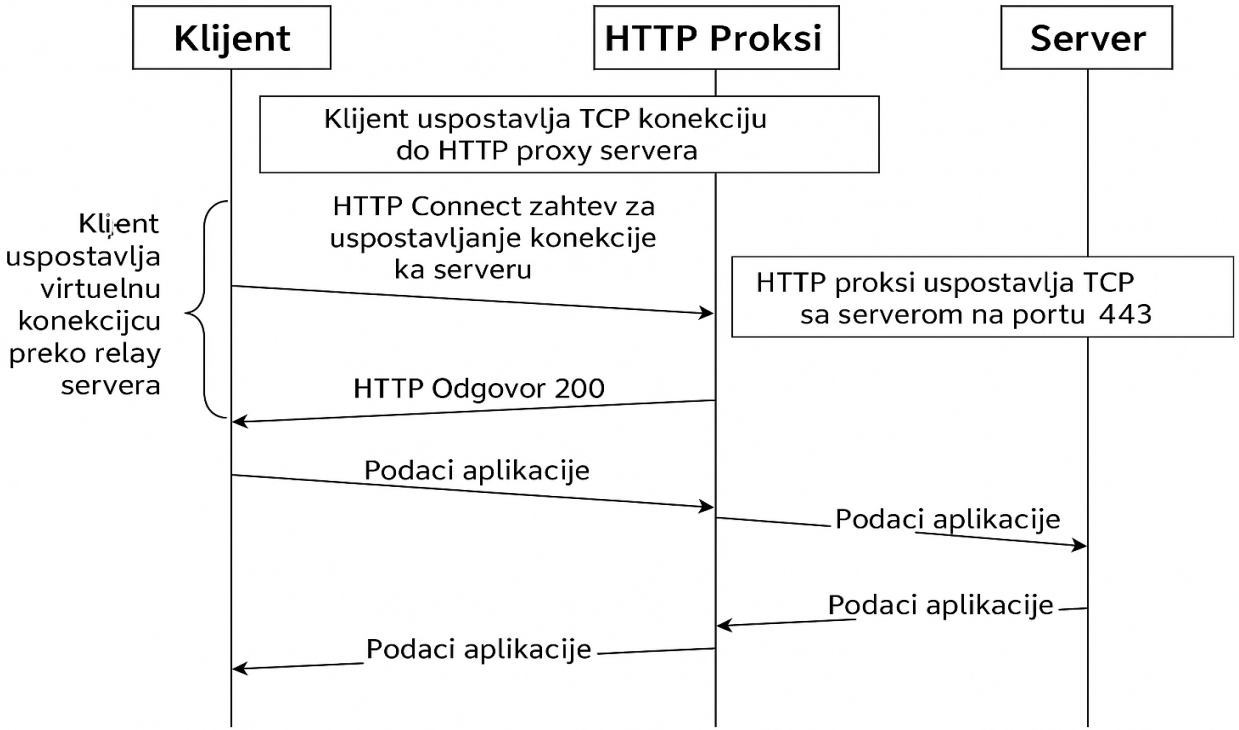
CSP/CORS politika

Ograničava međudomenske zahteve, sprečavajući izvršavanje zlonamernog JavaScript-a.

HTTP REQUEST KOMANDE (CONNECT METODA)

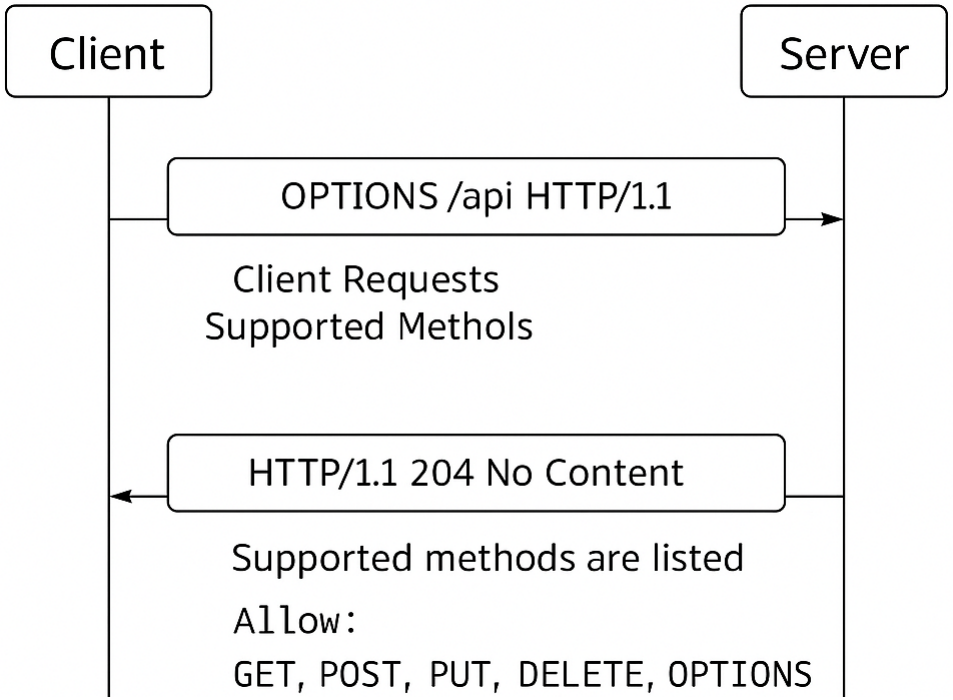
Metoda CONNECT se koristi za uspostavljanje tunela između klijenta i servera preko HTTP proxy-ja.

Najčešće se koristi za sigurnu HTTPS komunikaciju kroz proxy server.



HTTP REQUEST KOMANDE (OPTION METODA)

Metoda OPTIONS se koristi da bi klijent saznao koje su HTTP metode, zaglavlja i druge opcije dozvoljene za određeni resurs ili server




```
OPTIONS /api/users HTTP/1.1
Host: example.com
Origin: https://mojsajt.com
Access-Control-Request-Method: POST
```

```
HTTP/1.1 204 No Content
Allow: GET, POST, PUT, DELETE, OPTIONS
Access-Control-Allow-Origin: https://mojsajt.com
Access-Control-Allow-Methods: POST, GET
Access-Control-Allow-Headers: Content-Type
```

HTTP REQUEST KOMANDE (OPTION METODA)

Klijent šalje OPTIONS zahtev

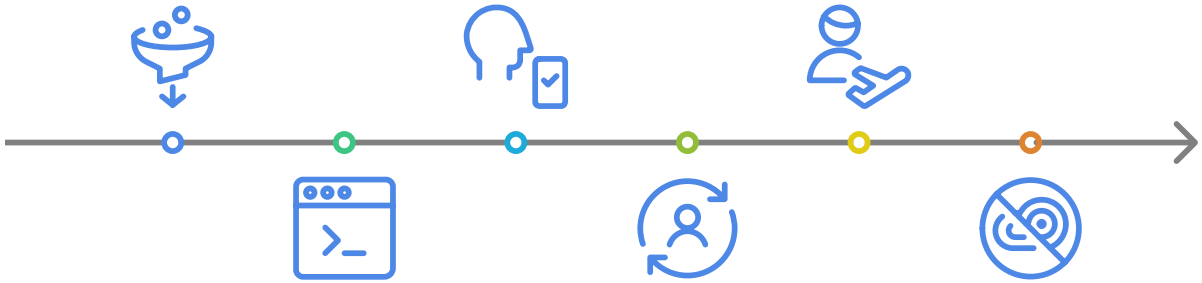
Klijent inicira zahtev za informacije o dozvoljenim metodama.

Klijent proverava dozvoljene metode

Klijent analizira odgovor servera kako bi potvrdio dozvoljene metode.

Klijent šalje CORS preflight zahtev

Klijent šalje kroz OPTION metodu preflight zahtev za proveru dozvola za pristup cross-origin resursu.



Server odgovara sa dozvoljenim metodama

Server pruža listu dozvoljenih HTTP metoda za resurs.

Klijent proverava pristup resursu

Klijent kroz option metodu proverava dozvolu za pristup cross-origin resursu..

Server odgovara sa CORS dozvolama

Server odgovara da li može da se pristupi traženom resursu

CORS (CROSS ORIGIN RESOURCE SHARING)

CORS i **CSP** su dva ključna bezbednosna mehanizma u modernom webu.

Šta je Cross-Origin Resource Sharing

CORS je bezbednosni mehanizam browsera koji kontroliše ko može da pristupi resursima na nekom domenu sa drugog domena

Problem koji rešava

Podrazumevano, web browseri blokiraju JavaScript aplikaciju sa jednog domena (npr. napadac.com) da pristupi API-ju na drugom domenu (npr. žrtva.com) – to je tzv. Same-Origin Policy. CORS zaglavlje se koristi za dozvolu pristupa

CSP (CONTENT SECURITY POLICY)

CSP je mehanizam zaštite od XSS napada

Omogućava serveru da definiše koji resursi smeju da se učitavaju i izvršavaju (JavaScript, slike, CSS, fontovi...).

```
Content-Security-Policy: default-src 'self'; script-src 'self’
https://cdn.trusted.com; object-src 'none';
```

Odakle sadržaj može da se učita

Učitava sadržaj samo sa istog domena "self".

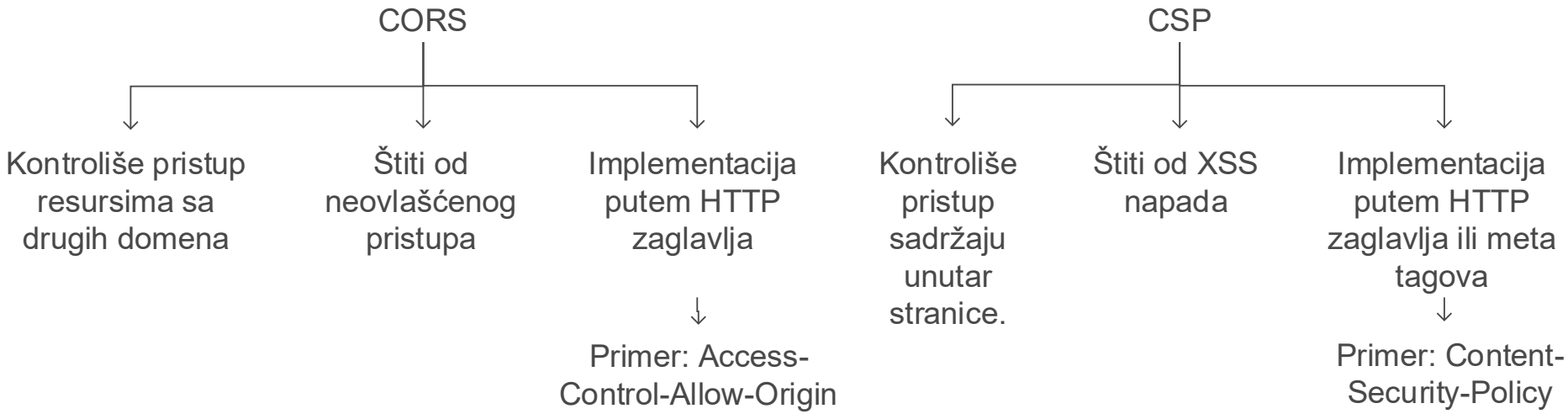
Dozvoljene skripte

Dozvoli skripte samo sa istog domena i CDN-a.

Šta se blokira

Blokiraj potencijalno ranjive tipove (object, embed, applet,...)

KONTROLA PRISTUPA I ZAŠTITA OD NAPADA



PODEŠAVANJA CORS PARAMETARA

Podešavaju se na strani servera, najčešće u okviru HTTP zaglavlja odgovora (response headers).
CORS se podešava na serveru (Apache, Nginx) ili u kodu aplikacije (Express JS, Flask,...)

Apache

```
Header set Access-Control-Allow-Origin "https://odseknis.edu.rs"
Header set Access-Control-Allow-Methods "GET, POST, OPTIONS"
Header set Access-Control-Allow-Headers "Content-Type, Authorization"
```

Node.js

```
app.use((req, res, next) => {
  res.setHeader('Access-Control-Allow-Origin', "https://odseknis.edu.rs");
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS');
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization');
  next();
});
```

PODEŠAVANJA CSP PARAMETARA

Podešavaju se na strani servera, u okviru HTTP odgovora putem Content-Security-Policy zaglavlja.
Može se definisati i unutar <meta> taga u HTML dokumentu, ali zaglavlje ima prednost.

Apache

```
Header set Content-Security-Policy "default-src 'self'; img-src *; script-src 'self' https://cdn.example.com"
```

HTTP zaglavlje

```
Content-Security-Policy: default-src 'self'; script-src 'self' https://trusted.cdn.com; object-src 'none';
```

✅ default-src 'self' — Dozvoljeno je učitavanje samo sa istog domena kao stranica (isti origin)
Sprečava učitavanje resursa sa drugih sajtova (npr. reklama, zlonamernih skripti).

✅ script-src 'self' https://trusted.cdn.com — Definiše izvore JavaScript fajlova koji su dozvoljeni
Blokira bilo koju skriptu sa nepoznatog domena

✅ object-src 'none' — Određuje izvore za <object>, <embed>, i <applet> tagove – koji se često koriste za pluginove. 'none' znači: nijedan objekat ne sme biti učitan tj. pluginova

HTTP REQUEST ZAGLAVLJE

```
GET /~srt/ HTTP/1.1
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; SLCC1; .NET
   CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.0.04506; InfoPath.1)
Host: www.vtsnis.edu.rs
Connection: Keep-Alive
```

Header parametri

Accept-Language : - Korisnik zahteva dati jezik za traženi objekat

User-Agent: - Opisuje tip browser-a koji je poslao zahtev

Host: – Server na kome se nalazi traženi objekat

Connection: - Client/browser saopštava serveru da TCP konekciju drži otvorenom.

HTTP RESPONSE PORUKA


```
HTTP/1.1 200 OK
Date: Fri, 22 Feb 2008 16:34:18 GMT
Server: Apache/2.0.52 (Red Hat)
Last-Modified: Thu, 15 Nov 2007 19:33:12 GMT
Content-Length: 15137
Connection: close
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

HTTP RESPONSE PORUKA

```
HTTP/1.1 200 OK
Date: Fri, 22 Feb 2008 16:34:18 GMT
Server: Apache/2.0.52 (Red Hat)
Last-Modified: Thu, 15 Nov 2007 19:33:12 GMT
Content-Length: 15137
Connection: close
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

Response poruka se sastoji:

- **Statusna linija**
- **Header linija**
- **Sadržaj**

HTTP RESPONSE PORUKA

```
HTTP/1.1 200 OK
Date: Fri, 22 Feb 2008 16:34:18 GMT
Server: Apache/2.0.52 (Red Hat)
Last-Modified: Thu, 15 Nov 2007 19:33:12 GMT
Content-Length: 15137
Connection: close
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

Statusna Linija

`HTTP/1.1` – Server koristi HTTP/1.1 verziju

`200 OK` - Statusni kod, zahtev je uspešno obrađen i tražena informacija se nalazi u telu poruke

Statusni kod se nalazi u svakoj response poruci koju šalje server a koja opisuje status interakcije između klijenta i servera

HTTP statusni kodovi su podeljeni u 5 klasa:

Informacione	1xx	Označava da je zahtev primljen ili da se obrađuje
Uspeh	2xx	Označava da je zahtev isporučen i obrađen od strane servera
Preusmeravanje	3xx	Preusmeravanje klijenta sa jednog URL-a na drugi
Greška klijenta	4xx	Greška na strani klijenta. Server nema sve potrebne podatke koje je dobio od klijenta
Greška servera	5xx	Greška na strani servera .

Opšti statusni kodovi

200 OK

- Zahtev je uspešno obrađen i tražena informacija je u response poruci.

301 Moved Permanently

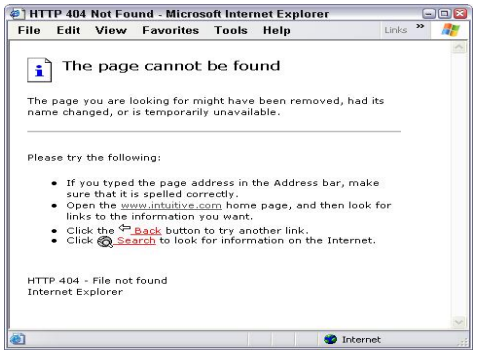
- Traženi objekat je trajno premešten.

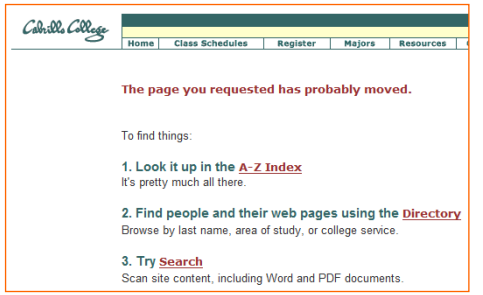
400 Bad Request

- Generička poruka o grešci, server nije razumeo request poruku.

404 Not Found

-Traženi dokument ne postoji na serveru.





INFORMACIONE PORUKE 1XX

100 Continue

Koristi se za optimizaciju slanja velikih zahteva proverom spremnosti servera. Koristi se kada klijent prvo pošalje zaglavlja, i želi da proveri da li je u redu da pošalje telo zahteva (npr. veliki fajl).

101 Switching Protocols

Koristi se npr. pri prelasku sa HTTP na WebSocket. Server prihvata zahtev iz zaglavlja Upgrade: i menja komunikacioni protokol.

102 Processing

Server je započeo obradu dugotrajnog zahteva i još uvek radi na tome. Informiše klijenta da server još uvek radi i da nije zaboravio zahtev.

USPEŠNA OBRADA 2XX

200 OK

Koristi se kada je zahtev uspešno obrađen i odgovor sadrži traženi sadržaj.

GET /api/users/123 → **200 OK**
(telo: { "id": 123, "ime": "Ana" })

201 Created

Koristi se kada je zahtev uspešno kreirao novi resurs.

POST /api/users → **201 Created**
Location: /api/users/124

202 Accepted

Koristi se kada je zahtev prihvaćen, ali nije odmah obrađen.

Klijent šalje zahtev za generisanje PDF fajla
→ server: „prihvaćeno, javićemo kad završimo”.

204 No Content

Koristi se kada je zahtev uspešno obrađen, ali nema sadržaja u odgovoru.

DELETE /api/posts/55 → **204 No Content**

205 Reset Content

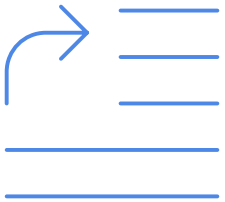
Koristi se za upućivanje klijenta da resetuje obrazac tj. unos.

206 Partial Content

Koristi se za preuzimanje u delovima (npr. streaming videa, nastavak prekinutog skidanja).

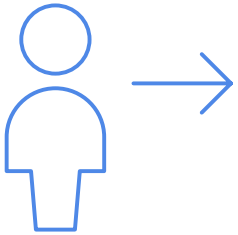
GET /video.mp4
Range: bytes=0-999 → **206 Partial Content**

REDIREKCIJA 3XX



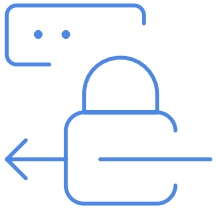
Stranica premestena

Kada je stranica premestena na novu lokaciju.



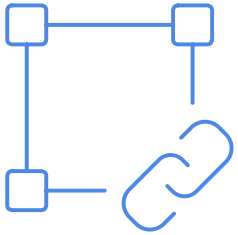
Preusmeravanje nakon logovanja

Kada korisnik treba da se preusmeri nakon logovanja.



Korišćenje HTTPS-a

Kada sajt koristi HTTPS umesto HTTP.



Skraćeni link

Kada se koristi skraćeni link (npr. bit.ly).

Status kod	Naziv	Opis
301	Moved Permanently	Resurs trajno premesten – browser treba da ažurira URL
302	Found (Moved Temporarily)	Privremeno preusmeravanje – koristi se npr. za logiku aplikacije
303	See Other	Koristi se posle POST zahteva da bi se klijent preusmerio GET metodom
307	Temporary Redirect	Kao 302, ali ne dozvoljava promenu HTTP metode
308	Permanent Redirect	Kao 301, ali zadržava HTTP metodu

GREŠKA NA STRANI KLIJENTA 4XX

Loše formatiran zahtev(400 Bad Request)

Klijent je poslao zahtev koji server ne može obraditi zbog grešaka u formatu.

POST /api/users
Content-Type: application/json
{ "name": "Ana", "age": } ← **nedostaje vrednost**

Pristup zabranjenom resursu (403 Forbidden)

Klijent pokušava pristupiti resursu za koji nema dozvolu.

Korisnik pokušava da pristupi /admin bez administratorskih prava.

Neautentifikovan klijent (401 Unauthorized)

Klijent nije autentifikovan ili nema potrebna prava pristupa.

GET /api/profile
→ 401 Unauthorized (**nema Authorization header**)

Resurs ne postoji (404 Not found)

Traženi resurs nije pronađen na serveru.

GET /api/products/999
→ 404 Not Found (**proizvod sa ID 999 ne postoji**)

Prekoračen broj zahteva (429 Too many requests)

Klijent je poslao više zahteva nego što je dozvoljeno.

Brute-force pokušaji logovanja ili spam zahtevi ka API-ju.

Konflikt sa stanjem resursa (409 Conflicts)

Dva korisnika pokušavaju da ažuriraju isti zapis istovremeno.

Klijent pokušava da kreira korisnika sa već postojećim emailom.

Nedozvoljena metoda (405 Method Not Allowed)

Klijent koristi metodu koja nije dozvoljena za taj resurs.

POST /api/users/123
→ 405 Method Not Allowed (dozvoljeni su samo GET i DELETE)

GREŠKA NA STRANI SERVERA 5XX

500 Internal Server Error

Kada se dogodi opšta greška servera bez specifičnog uzroka.

501 Not Implemented

Kada server ne podržava traženu HTTP metodu.

502 Bad Gateway

Kada server primi nevažeći odgovor od drugog servera.

503 Service Unavailable

Kada je server privremeno nedostupan zbog održavanja ili preopterećenja.

504 Gateway Timeout

Kada server ne primi odgovor na vreme od drugog servera.

505 HTTP Version Not Supported

Kada server ne podržava verziju HTTP protokola.

500 INTERNAL SERVER ERRORS

HTTP RESPONSE PORUKA

```
HTTP/1.1 200 OK
Date: Fri, 22 Feb 2008 16:34:18 GMT
Server: Apache/2.0.52 (Red Hat)
Last-Modified: Thu, 15 Nov 2007 19:33:12 GMT
Content-Length: 15137
Connection: close
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

Podaci u zaglavlju HTTP Response poruke

`Date:` – Vreme na serveru kada je kreirana respone poruka

`Server:` – Tip web servera koji je poslao poruku

`Last-Modified:` – Datum/vreme kada je objekat kreiran ili modifikovan

`Content-Length:` – Broj bajtova koji je poslat

`Connection:` – Server će zatvoriti TCP konekciju nakon što pošalje odgovor .

`Content-Type:` – Objekat u zaglavlju je HTML text

HTTP RESPONSE PORUKA

```
HTTP/1.1 200 OK
Date: Fri, 22 Feb 2008 16:34:18 GMT
Server: Apache/2.0.52 (Red Hat)
Last-Modified: Thu, 15 Nov 2007 19:33:12 GMT
Content-Length: 15137
Connection: close
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

Telo poruke

- HTML text i drugi objekti koje razume browser na klijentu
- JSON (JavaScript Object Notation)
- XML (eXtensible Markup Language)
- Binarni fajlovi (slike i video snimci)
- Skripta

HTTP REQUEST \ RESPONSE PORUKA

HTTP
Server

HTTP REQUEST(1)

HTTP RESPONSE(2)

HTTP
Klijent

HTTP RESPONSE PORUKA

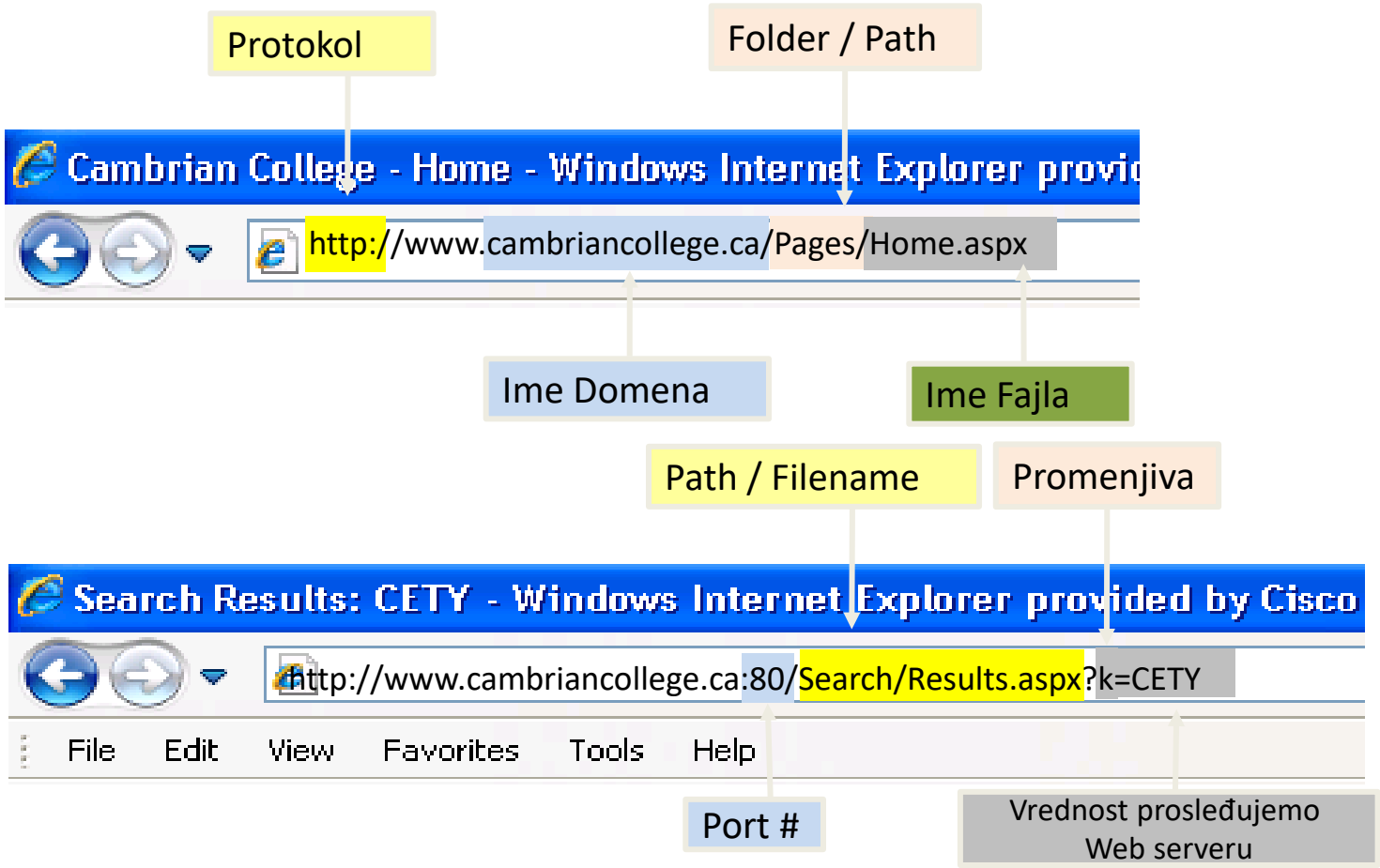
```
HTTP/1.1 200 OK
Date: Fri, 22 Feb 2008 16:34:18 GMT
Server: Apache/2.0.52 (Red Hat)
Last-Modified: Thu, 15 Nov 2007 19:33:12 GMT
Content-Length: 15137
Connection: close
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

HTTP REQUEST PORUKA

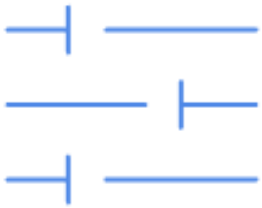
```
GET /~vtsnis/ HTTP/1.1
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE
    7.0; Windows NT 6.0; SLCC1; .NET CLR
    2.0.50727; Media Center PC 5.0; .NET
    CLR 3.0.04506; InfoPath.1)
Host: www.vtsnis.edu.rs
Connection: Keep-Alive
```

Struktura URL adrese

URL ili URI (Uniform Resource Identifier) je ime na osnovu koga se pristupa Web strani na Web serveru preko Web čitača



KLJUČNA HTTP OSOBINA



Stateless protokol

HTTP ne prati stanje sesije između dva uzastopna zahteva.

Nezavisna interakcija

Svaki zahtev je nova nezavisna interakcija sa serverom.

Dodatni mehanizmi

Ako korisnik uspešno izvrši logovanje putem jednog HTTP zahteva, naredni zahtev (npr. za pristup korisničkoj stranici) neće automatski sadržati informacije o toj prijavi – osim ako se dodatni mehanizmi ne koriste.

MEHANIZMI ZA POVEZIVANJE HTTP ZAHTEVA

HTTP kolačići

Tekstualne datoteke koje server šalje klijentu i koje se čuvaju u browseru. Oni omogućavaju čuvanje informacija između zahteva, kao što je ID sesije.

Tokeni za autentifikaciju

Tokeni se koriste za potvrdu identiteta korisnika i omogućavanje pristupa zaštićenim resursima. Mogu se slati putem kolačića ili HTTP zaglavlja, a najčešće se koriste JWT (JSON Web Token) i session ID.

Skrivena polja u formama

Nevidljivi podaci unutar web formi omogućavaju prenos podataka koji nisu vidljivi korisniku

URL parametri

Podaci se mogu slati putem URL-a korišćenjem query stringa (npr. ?id=123). Ova metoda je jednostavna i direktna, ali manje bezbedna jer su podaci vidljivi u adresnoj liniji.

Mehanizam	**Opis**	**Primer primene**
HTTP kolačići (cookies)	Čuvaju informacije o sesiji (npr. session ID) koje se automatski šalju uz svaki HTTP zahtev.	Set-Cookie: session_id=abc123; HttpOnly; Secure
Tokeni za autentifikaciju	Klijent dobija token (JWT, session ID) koji šalje u svakom narednom zahtevu za autentifikaciju.	Authorization: Bearer eyJhbGciOi... (JWT token u zaglavlju)
Hidden field-ovi u formama	Nevidljiva polja u HTML formama koja sadrže bitne informacije o sesiji.	<input type="hidden" name="token" value="abc123">
URL parametri	Podaci koji se prosleđuju kroz URL adresu, vidljivi korisniku.	https://example.com/profil?korisnik_id=123

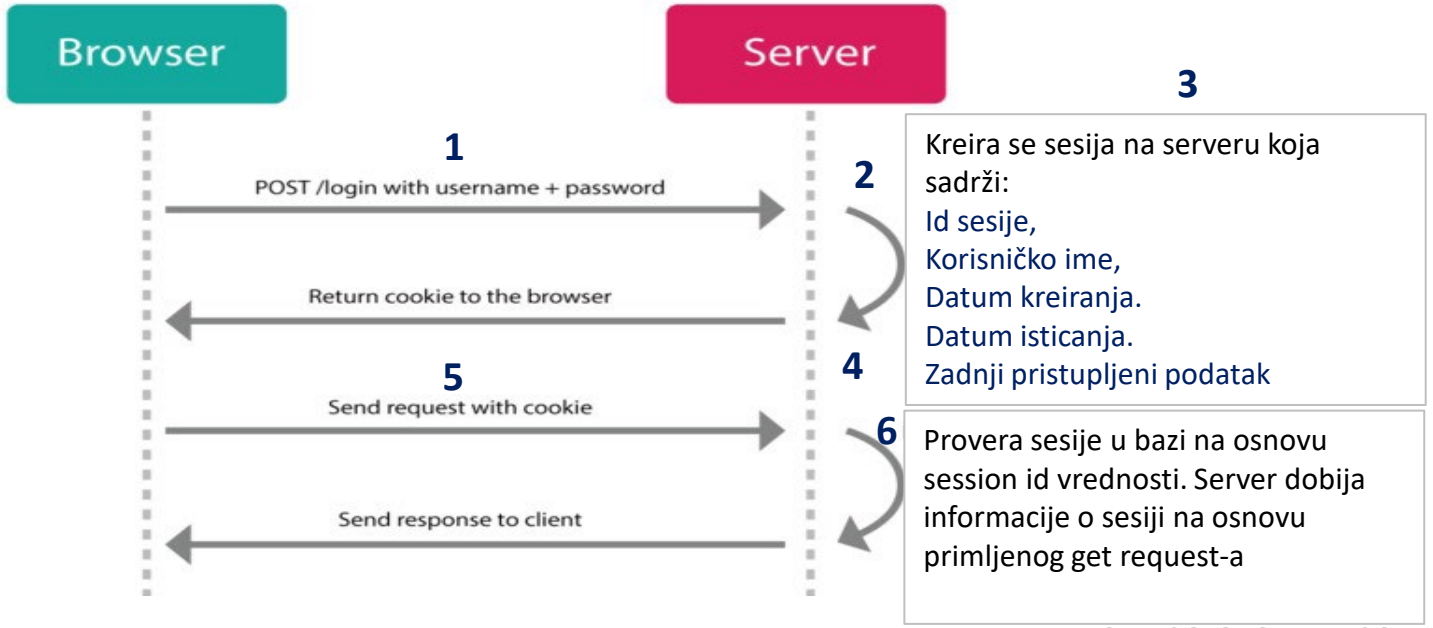
AUTORIZACIONE STRATEGIJE NA WEB-U

Session token - tradicionalna metoda

Cookies – Praćenje autentifikovanog korisnika

Stateful auth (odnosi se na sesije)

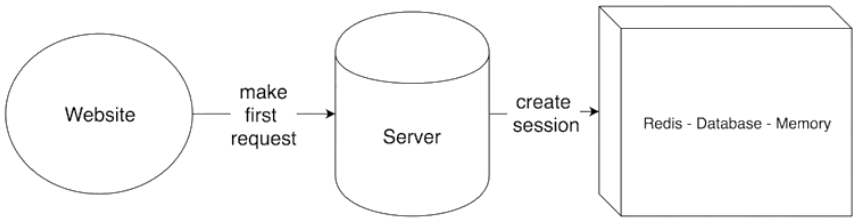
1. Korisnik šalje login kredencijale
2. Server proverava kredencijale u bazi
3. Server kreira privremenu tzv. korisničku sesiju
4. Server kreira **cookie za slanje informacije o kreiranoj sesiji** na osnovu Session ID
5. Korisnik šalje cookie koji se čuva na klijentskoj strani a sadrži Session id u svakoj request poruci
6. Server verifikuje na osnovu session id koji je prethodno kreirao i dozvoljava pristup
7. Kada se korisnik izloguje, server zatvara sesiju i briše kolačić.

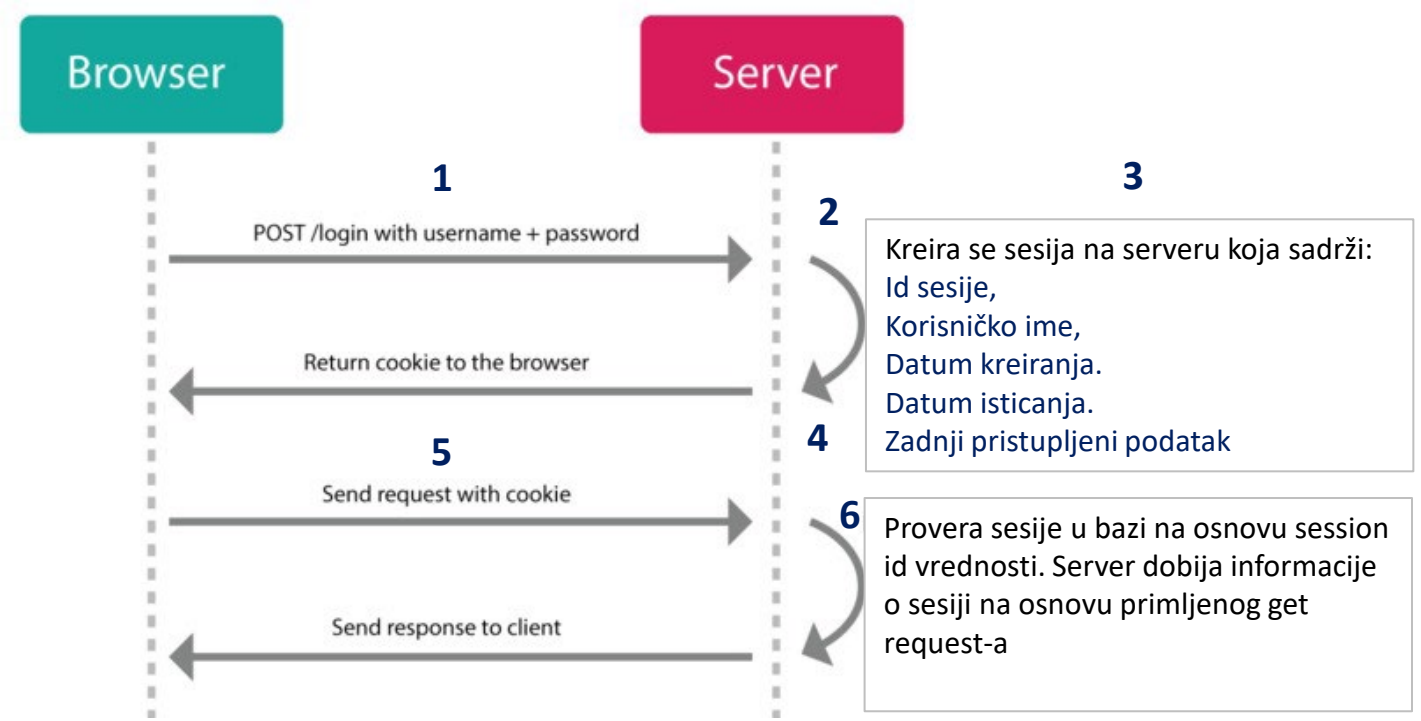


Cookies – Praćenje autentifikovanog korisnika

Svaka korisnička sesija se čuva **na serverskoj strani** (stateful)

1. Memoriji (fajl sistem) - retko
2. Keš (Redis ili Memcashed)-čest slučaj
3. Baza Podataka(Postgres ili MongoDb)



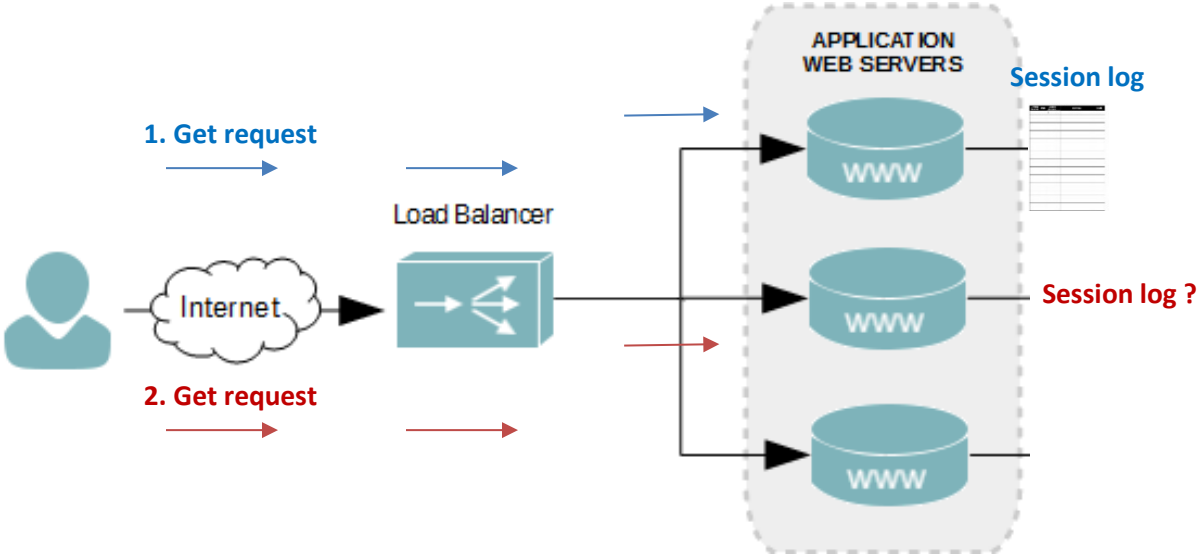


Session ID + Cookies – problem praćenja korisnika

Moderni web server zbog redudatnosti i bržeg opsluživanja klijenta sadrže više instanci tj. kopija.

Raspoređivači opterećenja (load balancer) kada primi zahtev od klijenta odlučuje kojoj www instanci će poslati zahtev

Problem se javlja kada se informacija o sesiji kreira na jednoj instanci a zatim neki od narednih zahteva load balanser prosledi drugoj instanci.



Moguće rešenje

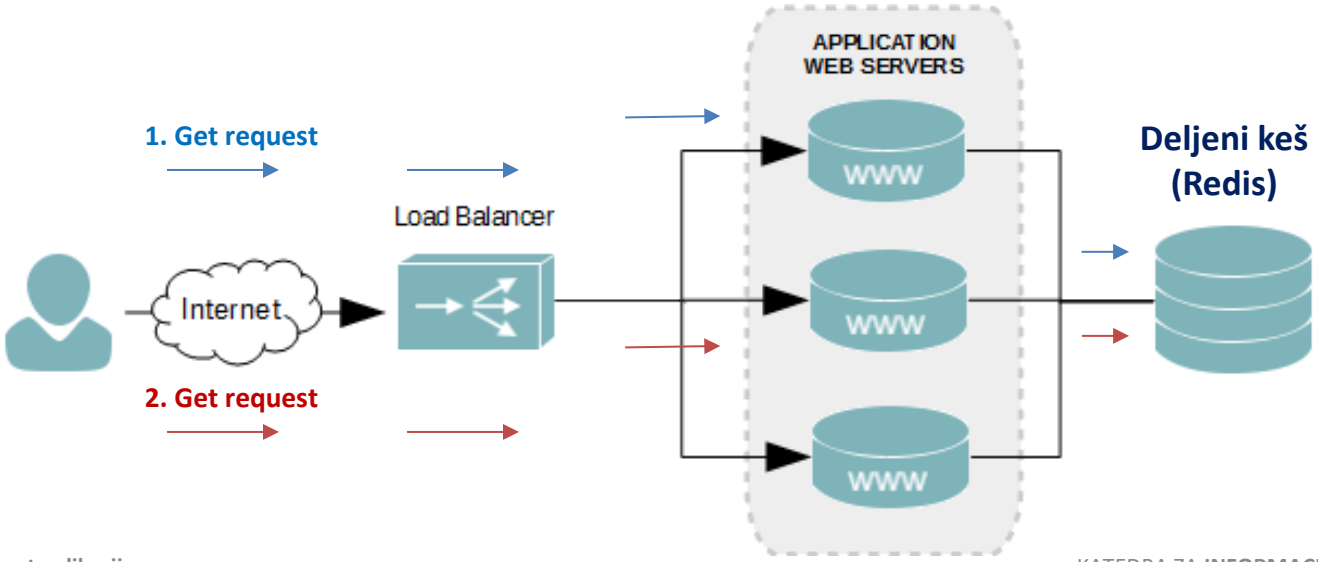
Kreira se in-memory baza podataka (Redis) koja čuva kreirane sesije svih instanci

Nedostatak je što je Redis *single point of failure* tj. ukoliko on otkaže sve informacije o sesijama se gube.

Alternativno rešenje je da se load balanser podesi da prati na kojoj instanci je kreirana koja sesija i da zahteve klijenata prosleđuje onoj instanci gde je kreirana njegova sesija.

Rešenje nije skalabilno

Drugi nedostatak je da onaj ko dobije pristup ID sesije može da se lažno predstavlja kao korisnik jer se od njega ne zahteva korisničko ime i lozinka.



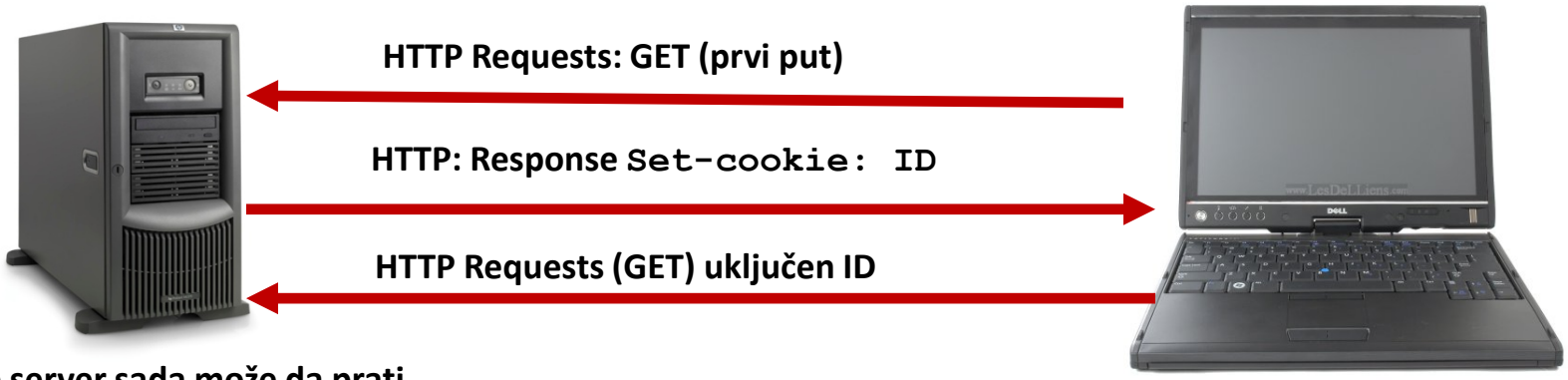
KLIJENT SERVER INTERAKCIJA PREKO KOLAČIĆA - COOKIES

Web serveri koriste kolačiće (**cookies)** za čuvanje informacija o sesiji.

Putem kolačića informacije o sesiji mogu da se prenosu do web servera.

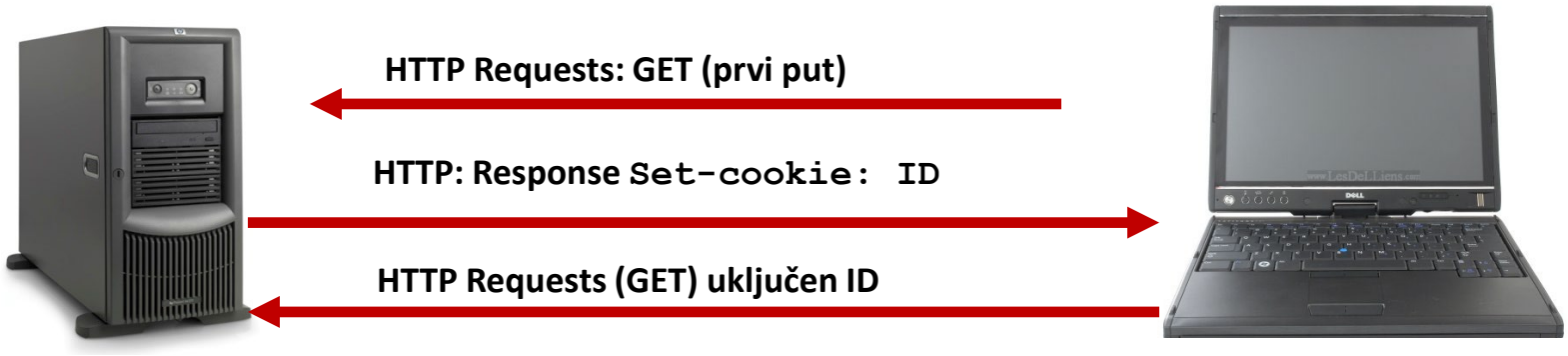
Najčešća metoda za prenos id sesije

Cookies su definisani RFC 2109



Web server sada može da prati aktivnost klijenta na web sajtu.

KOLAČIĆI (COOKIES)



Kolačić je deo informacije koji ima naziv, vrednost i parametre ponašanja

Kreira ga server a čuva se u fajl sistemu klijenta ili memoriji web pretraživača

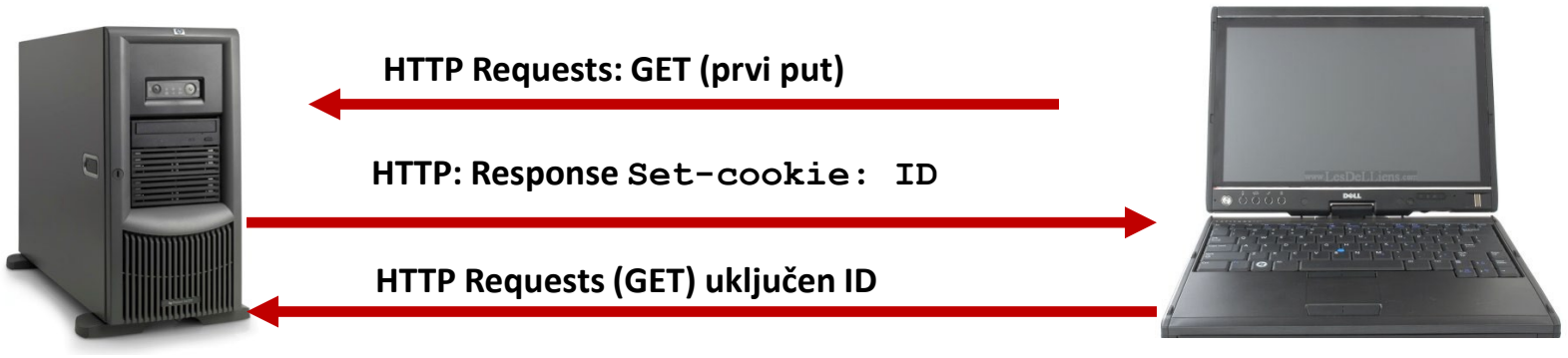
Vrednosti kolačića su povezane sa URL adresom.

Najčešći mehanizam prosleđivanja ID sesije između klijenta i servera je putem kolačića

Kada korisnik poseti originalni web sajt, pretraživač će poslati vrednosti kolaćića identifikujući korisnika.

Kolačići osim za praćenje sesija mogu da se upotrebe za čuvanje informacija za krajnjeg klijenta kao što su jezik i druge konfiguracione opcije.

KLIJENT SERVER INTERAKCIJA PREKO KOLAČIĆA - COOKIES



Web server instalira cookies na klijentu kada:

Pristupa web sajtu prvi put (Web server ne prepoznaje klijenta po imenu)

Korisnik saopšti informacije web serveru. (Web server prepoznaje klijenta po imenu)

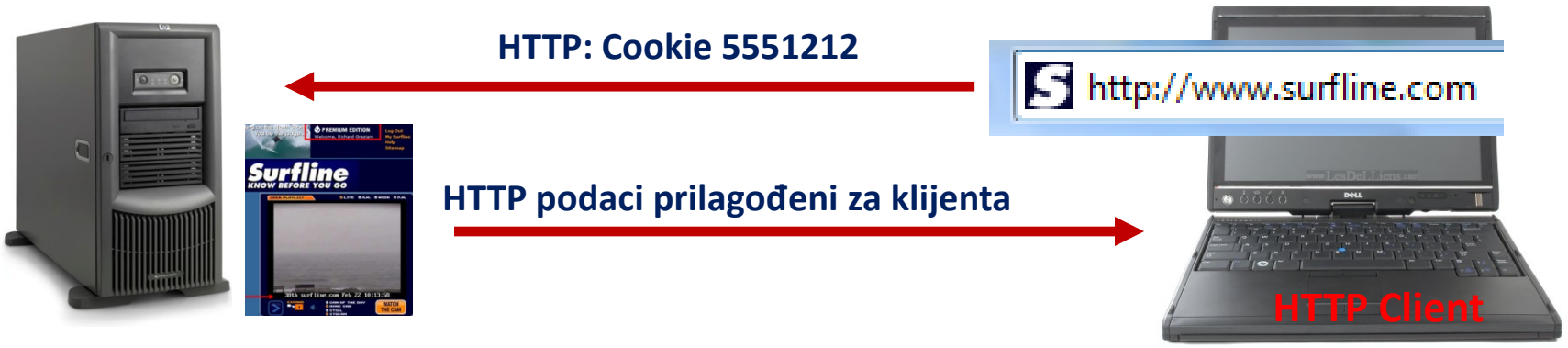
HTTP na Web serveru u zaglavlju odgovora uključuje **`Set-cookie:`** **ID**.

Ovaj **ID** se čuva na klijentskom računaru.

Svaki put kada client/browser pristupi web sajtu, **`GET`** metoda uključuje **`Cookie`** ili **`User_ID`**.

Kolačiće uvek kontroliše i postavlja server

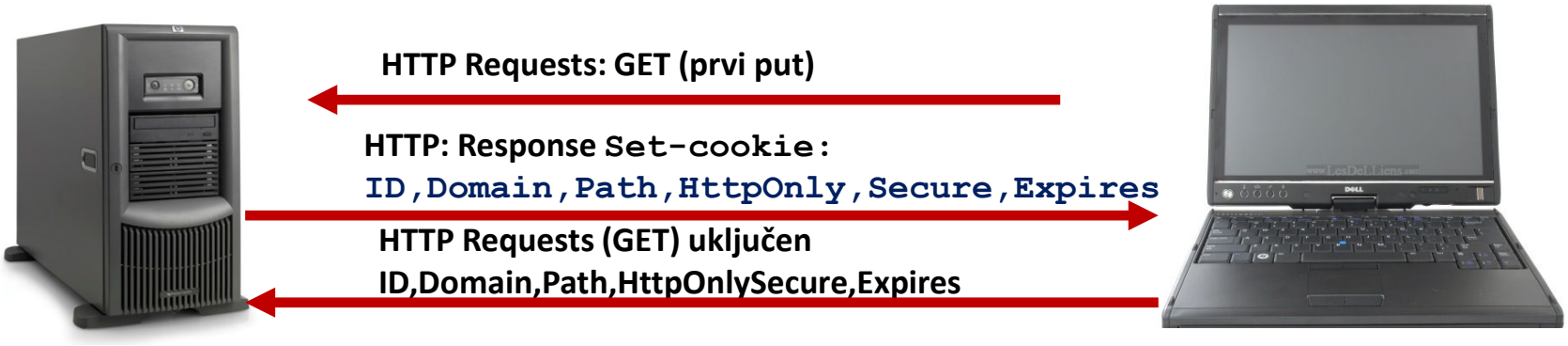
KLIJENT SERVER INTERAKCIJA PREKO KOLAČIĆA - COOKIES




```
HTTP/1.1 200 OK
Date: Fri, 22 Feb 2008 19:00:15 GMT
Server: Apache/1.3.34 (Unix)
Last-Modified: Fri, 22 Feb 2008
18:51:47 GMT
ETag: "760a31-18ce-47bf19c3"
Accept-Ranges: bytes
Content-Length: 6350
Keep-Alive: timeout=15, max=257
Connection: Keep-Alive
Content-Type: text/plain <......>
```

```
GET /jpeg/cap81/cam0.36705623.rgb888.enc HTTP/1.1
<.....................>
Cookie: SLSPOTNAME5=Cowells;
    SLSPOTNAME4=Waimea%20Bay;
    SLSPOTNAME3=Pipeline;
    SLSPOTNAME2=38th%20Ave%2E;
    SLSPOTNAME1=Cowells; SLSPOTID5=4189;
    SLSPOTID4=4755; SLSPOTID3=4750;
    SLSPOTID2=4191; SLSPOTID1=4189;
    OAX=R8bfwEbcU08ABCBu; USER_ID=5551212 <not my
    actual user-id>;........
```

Parametri kolačića



Web aplikacija može kolačiću da dodeli sledeće parametre:

Domain: sadrži domen kome se kolačić šalje

Path: putanja na serveru na kojoj će kolačić biti dostupan.

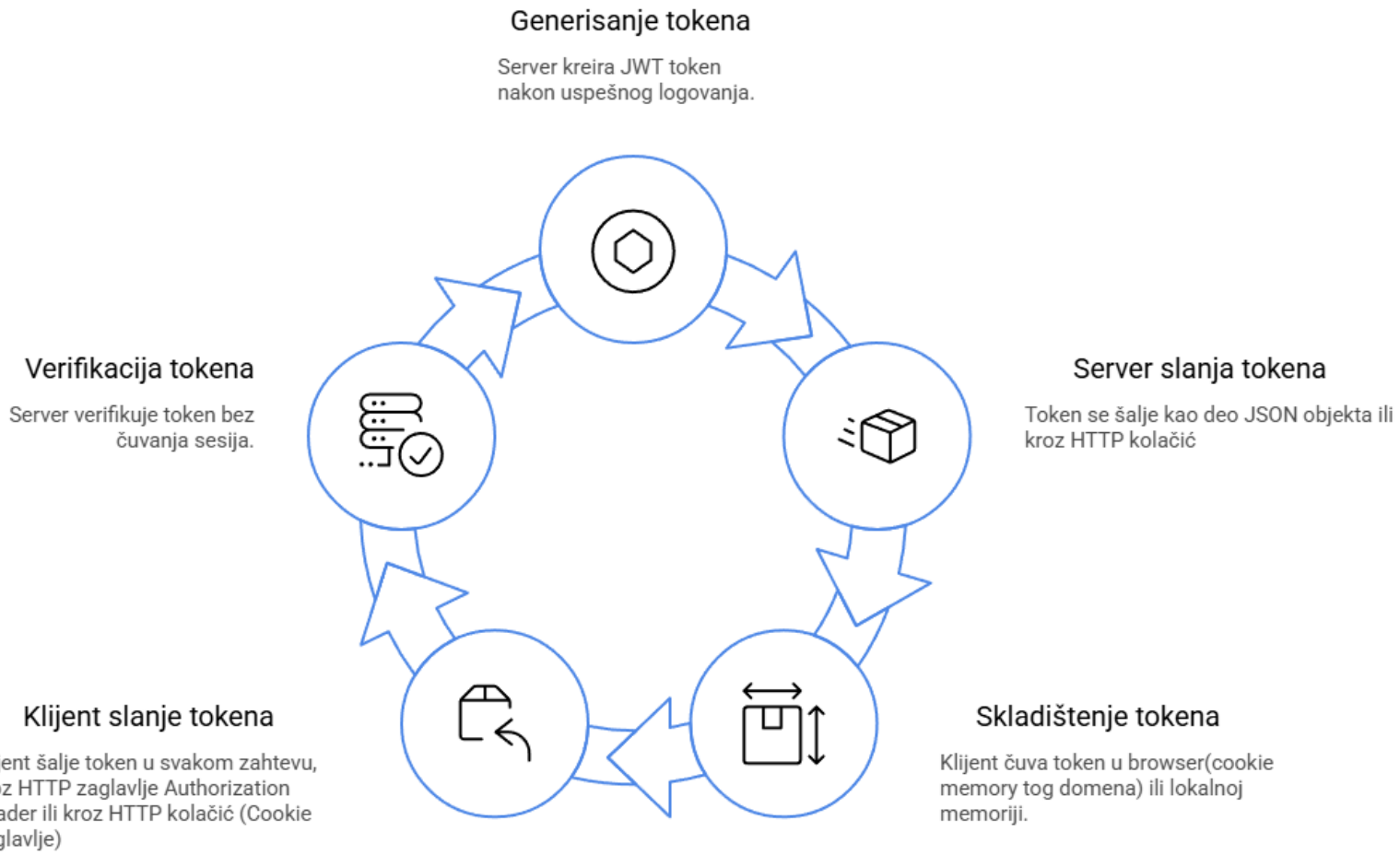
HttpOnly: sprečavamo da Java Script pristupi kolačiću a samim tim i potencijalni XSS (Cross site scripting) napad

Secure: zahteva slanje kolačića samo preko sigurnih prenosnih kanala (SSL/TLS)

Expires: definiše vreme brisanja kolačića

AUTORIZACIONE STRATEGIJE NA WEB-U

Jason Web token (JWT) – modernija metoda



JSON WEB TOKEN

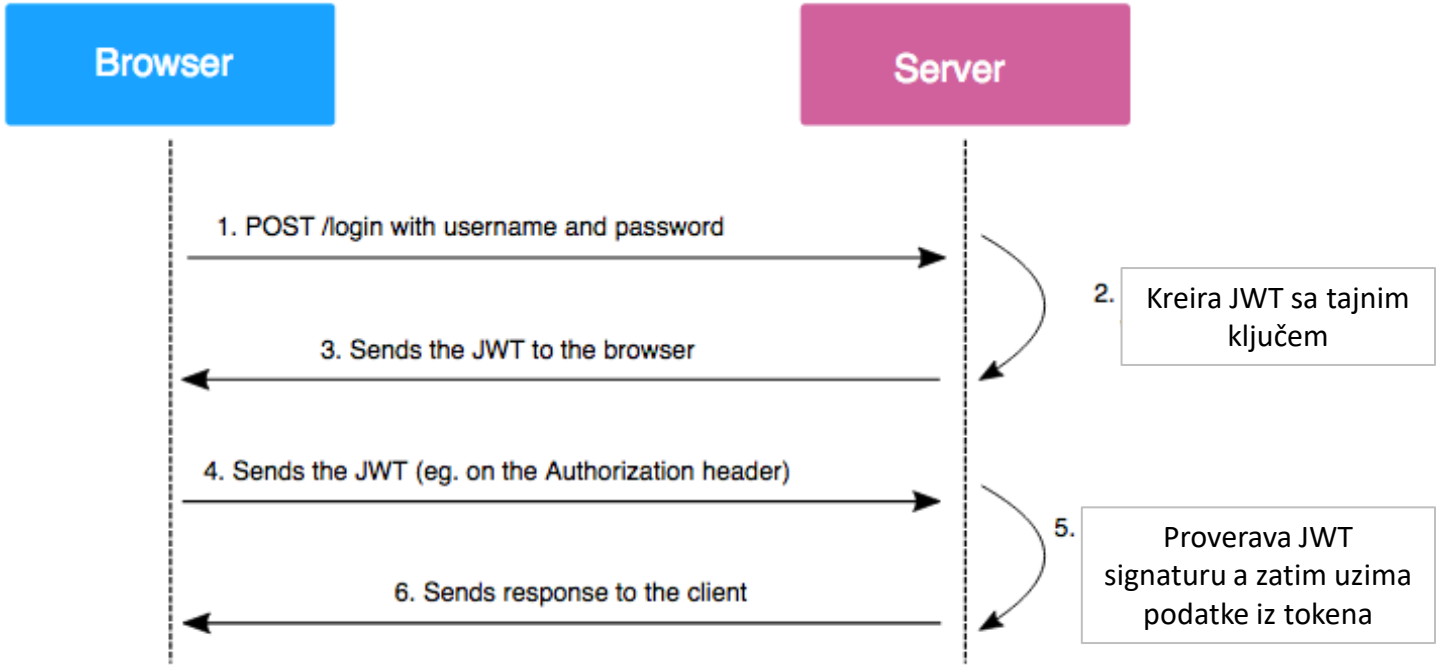
Koristi se za autorizacuju na web-u

Web aplikacija kreira token koji čuva klijent i koji sadrži digitalni potpis web aplikacije koja je kreirala token.

Jason token koji šalje klijent nije samo ID već sadrži sve informacije o samom klijentu (Jason objekat) jer je to način da server autentifikuje klijenta.

Problem bezbednosti je rešen jer je token potpisan.

JWT token se čuva na klijentu na disku (local storage ili u cookie) a šalje se kroz HTTP zaglavlje.



JSON WEB TOKEN

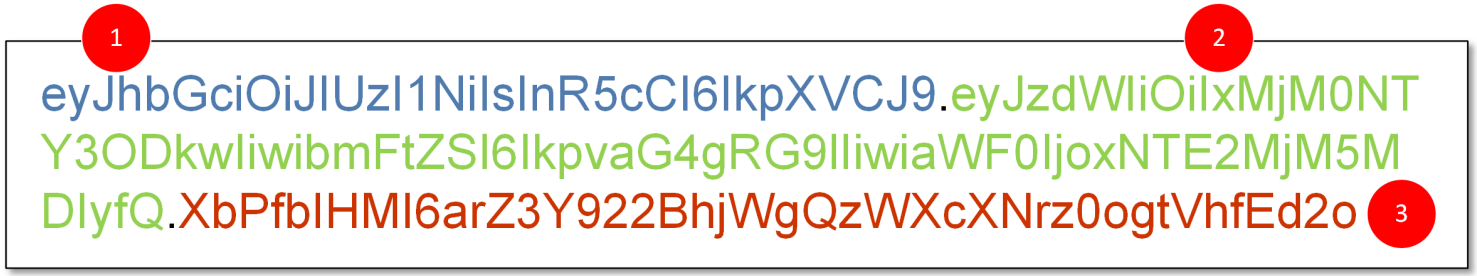
JWT token se najčešće sastoji iz tri dela koji su razdvojeni tačkom

Base64 enkoder se koristi da niz karaktera konvertuje u originalni niz

Payload su podaci u JSON formatu

Zaglavlje sadrži informaciju o algoritmu koj se koristi za kreiranje potpisa

Potpis služi da server verifikuje token tj. da su vrednosti tokena validne a ne da se token kriptuje. Samo server koji je kreirao token može proveri autentičnost tokena na osnovu kreirane signature i tajnog ključa koji samo on zna.



1 **Zaglavlje**
Algoritam i tip tokena

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

2 **Payload**
Podaci

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

3 **Signatura**
Digitalni potpis

```
HMACSHA256(
BASE64URL(header)
.
BASE64URL(payload),
secret)
```

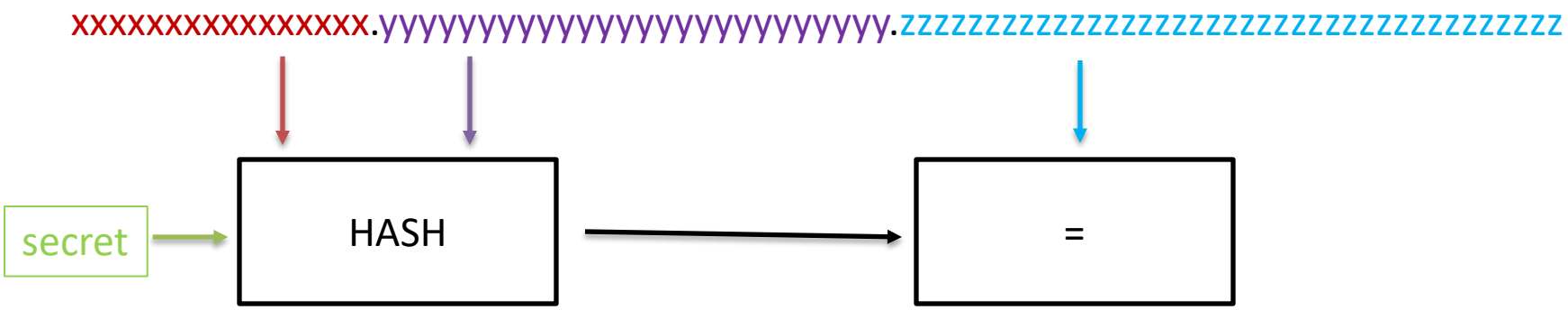
JSON WEB TOKEN – Verifikacija na serveru

JWT token ne mora da sadrži poverljive informacije kao što je lozinka već osnovne informacije da bi server znao o kome se radi npr. korisničko ime.

Ukoliko je JWT token ukraden, server neće znati da li je token stigao od prave osobe ili osobe koja je ukrala token. Zato se preporučuje da se vodi računa kako se token šalje preko mreže (HTTPS).

U odnosu na session token, ukoliko je došo do krađe session id-a čim se izlogujemo server briše sve informacije o sesiji.

JWT token se čuva na klijentu, server nema mogućnost da JWT token uništi tj. odbaci. Glavni nedostatak ukoliko dođe do zloupotrebe JWT tokena. Kao mogućnost postoji opcija da se na serveru čuva crna lista JWT tokena.

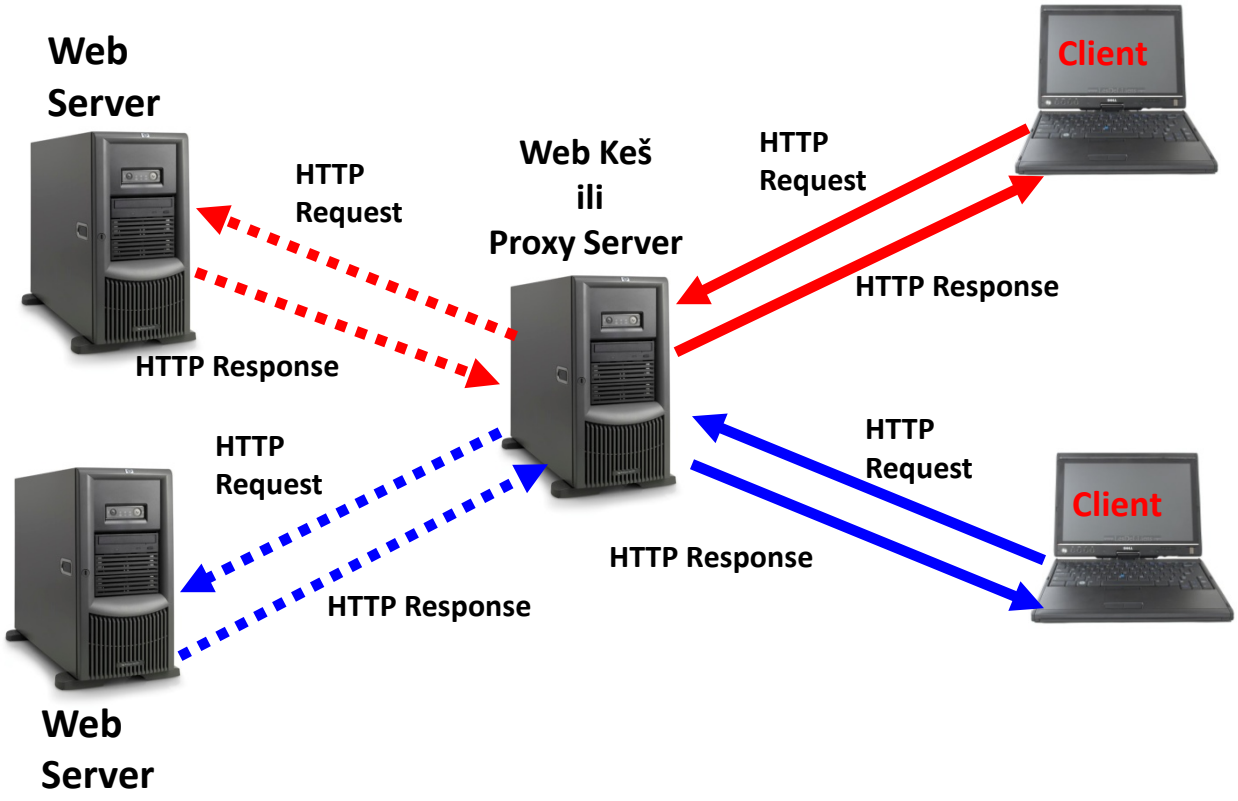


SLANJE JWT - HTTP REQUEST HEADER

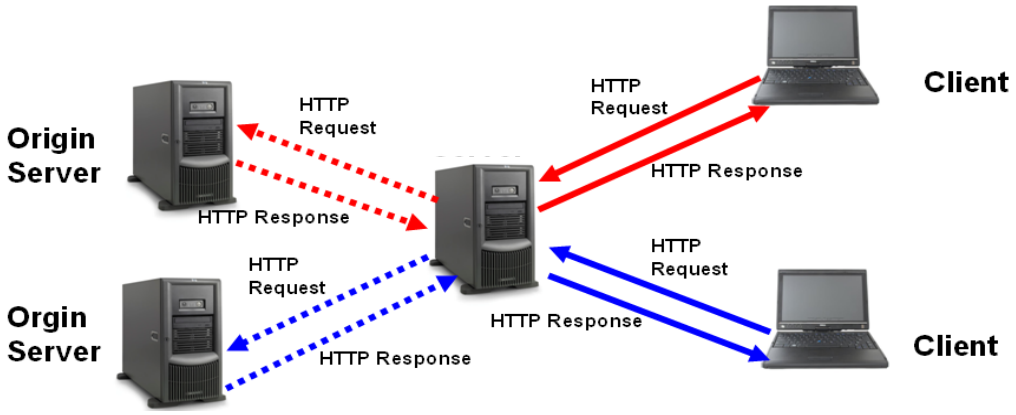
▼ **Request Headers** view source

Accept: application/json, text/plain, */*
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,fr;q=0.8
Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjp7Il9pZCI6IjU5ZjA3YjA3NzBmOWM0MDcxY2JiODA3MCIsInVwZGF0ZWRBdCI6IjIwMTctMTEtMTdUMDE6MTY6NDYuNDQyWiIsImNyZWF0ZWRBdCI6IjIwMTctMTAtMjVUMTE6NTI6MzkuMTMwWiIsImVtYWlsIjoiYWxhbjM0NUBnbWFpbC5jb20iLCJwYXNzd29yZCI6InNoYTEkMzY0ODc4YzUkMSQ1MDNhNjU0MTY4ZjQzM2QzZTNhZTUzNGE5MTE1OTNhMzkzMDQxOTgxIiwicm9sZSI6WyJjbGllbnQiXSwicGFpZW1lbnQiOnsic3RyaXBlIjpbXX0sIl9fdiI6NiwicHJvZmlsZSI6eyJhZGRyZXNzIjpbXSwiZGV0YWlsSG91c2UiOnsiYWNjZXNzVHlwZSI6IiIsImZsb29yIjoiIiwiYWNjZXNDb2RlIjoiIiwic3VyZmFjZSI6MCwidHlwZUhvdXNlIjoiIn0sIm90aGVyRGF0YSI6IiIsInN0YXR1c0hvdXNlIjoiIiwiY29sb3JDYWxlbmRhciI6IiNhZDIxMjEiLCJ0eXBlQ2xpZW50IjoiIiwicGhvbmVOdW1iZXIiOiIiLCJsYX

PROXY SERVER



PROXY SERVER



Browser klijenta šalje *HTTP Request* Web keš (*Proxy*) serveru.

Web keš proverava da li postoji lokalna kopija traženog objekta.

Ukoliko lokalna kopija postoji: Web keš šalje objekat nazad klijentu

Ukoliko lokalna kopija ne postoji: Web keš šalje *HTTP request* Web serveru koji sadrži traženu stranicu.

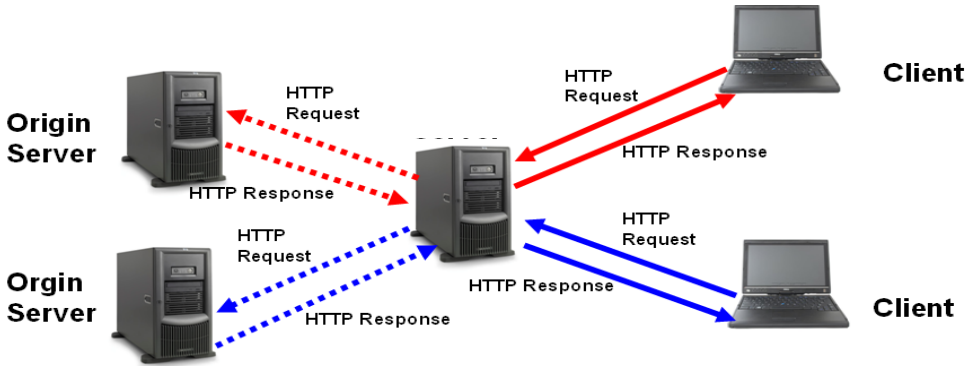
Web (Origin) server šalje traženi objekat Web keš serveru.

Web keš čuva lokalnu kopiju objekta.

Web keš prosleđuje kopiju objekta browseru na klijentu.

TCP konekcija se kreira između Klijenta i Web Keš servera, Web keš i Origin servera

PROXY SERVER



Problem – **Stale cache**

Objekat može da se modifikuje na Origin serveru nakon što je kopija objekta sačuvana na Web keš serveru.

Rešenje – **Uslovni GET**

Request metod: ***GET***

Uključuje heder: ***If-Modified-Since*:**

Web keš šalje Uslovni GET ka Origin serveru da bi proverio da li postoji novija verzija objekta.

Nema nove verzije: Šalji lokalni objekat

Ima nove verzije: Zameni loklani objekat i prosledi novu verziju