



## CS105 - OSNOVE VEB TEHNOLOGIJA

Osnove veb tehnologija

Lekcija 01

PRIRUČNIK ZA STUDENTE

# CS105 - OSNOVE VEB TEHNOLOGIJA

## Lekcija 01

### *OSNOVE VEB TEHNOLOGIJA*

- ✓ Osnove veb tehnologija
- ✓ Poglavlje 1: Internet i WWW
- ✓ Poglavlje 2: Veb standardi
- ✓ Poglavlje 3: HTTP protokol
- ✓ Poglavlje 4: Način funkcionisanja HTTP protokola
- ✓ Poglavlje 5: Programiranje sa klijentske i serverske strane
- ✓ Poglavlje 6: Responzivni veb dizajn
- ✓ Poglavlje 7: Pokazne vežbe: responzivni veb dizajn
- ✓ Poglavlje 8: Zadatak za samostalni rad: HTTP zahtev i odgovor
- ✓ Poglavlje 9: Domaći zadatak
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ✓ Uvod

### UVOD

*Cilj ovog predavanje je da da pregled osnovnih veb tehnologija i osnovnih termina koji se koriste na Internetu*

Cilj ovog predavanje je da da pregled osnovnih veb tehnologija i osnovnih termina koji se koriste na Internetu. Kroz ovo predavanje će biti objašnjen način funkcionisanja HTTP protokola, da se prikažu osnovni protokoli koji se koriste na vebu, kao i njihov način funkcionisanja. Na ovom predavanju biće reči i o opštim karakteristikama programiranja sa serverske i klijentske strane, kao i navedeni osnovni primeri za neke od često korišćenih skripting jezika.

Na ovom predavanju biće obrađene sledeće teme:

- Osnove veb standarda
- URI, URL, URN
- Struktura i način funkcionisanja HTTP protokola
- Programiranje sa klijentske strane
- Programiranje sa serverske strane.

## ▼ Poglavlje 1

# Internet i WWW

## INTERNET SERVISI

*Razvoj Interneta omogućio je razvoj novih tehnologija i upotrebu mnogih novih servisa koji koriste Internet kao infrastrukturu*

Internet, ili popularnije nazvan "net", između ostalog, pruža veliki izvor informacija, pruža olakšano poslovanje i unapređuje komunikaciju. Internet ustvari predstavlja milione mreža koje povezuju preduzeća, vladine organizacije, obrazovne institucije, pojedince i sl.

**Razvoj Interneta omogućio je razvoj novih tehnologija i upotrebu mnogih novih servisa koji koriste Internet kao infrastrukturu.** Najpopularniji Internet servisi su:

- E-mail
- Telnet
- Forum
- FTP
- Internet Chat
- World Wide Web
- Veb portali
- Veb Servisi

**World Wide Web i email su dva servisa koja se najčešće koriste na Internetu.** Iako se World Wide Web (WWW), ili jednostavno "veb", često meša sa Ineternetom, veb ustvari predstavlja samo jedan od servisa koji radi nad infrastrukturom Interneta.

WWW je Internet servis koji predstavlja skup elektronskih dokumenata. Ta dokumenta nazivamo veb stranicama. Veb strane obično sadrže sadržaj u formi teksta, zvuka, videa, grafike i animacije.

Za korišćenje veb servisa korisnicima služe **vеб читачи** (engl. **web browser**). Veb čitač je aplikacioni softver koji koristi korisnik da pristupi vebu. Veb čitač koriste i neki Internet servisi kao što su email ili FTP. Veb čitači obično imaju grafički korisnički interfejs koji je specijalno dizajniran kako bi se što efikasnije prikazale veb stranice. Veb čitači omogućavaju korisnicima da pregledaju hipertekstualne stranice. Najčešće korišćeni veb čitači su Internet Explorer, Firefox, Opera, Chrome, Safari.

**Hipertekst** je pojam kojim se opisuje dokument u kome se pojedini pojmovi iz tog dokumenta referenciraju na druge dokumente ili datoteke sa drugim sadržajima, na primer sa zvučnim ili video zapisima. Za referenciranje se koriste **hiperlinkovi** (engl. **hyperlink**), koji sadrže putokaz do resursa na koji izabrani pojmom iz hiperteksta referencira. Traženi resursi mogu biti na istom računaru ili na nekim udaljenim računarima koji su vezani na Internet. Za smeštaj i pristup

hipertekst stranica koriste se **veb serveri**. Oni imaju zadatak da veb čitačima pripreme i pošalju zahtevani sadržaj. Kolekcija hipertekst stranica na jednom veb serveru se naziva veb sajt (engl. **Web site**).

## ▼ Poglavlje 2

### Veb standardi

#### ŠTA SU VEB STANDARDI?

*Veb standardi su pravila i smernice koje je uspostavio World Wide Web Consortium*

**Veb standardi** su pravila i smernice koje je uspostavio **World Wide Web Consortium** (W3C) razvijen da promoviše konzistentnost u kodu koji čini veb stranicu.

Jednostavno rečeno, veb standardi predstavljaju smernice za mark-up jezike koji čine jednu veb stranicu.

Prednosti u pridržavanju standarda su mnoge:

- Veb stranice će se prikazivati u širokom spektru čitača i računara, uključujući nove tehnologije
- W3C standardi promovišu upotrebu "**Cascading Style Sheets**" (CSS) umesto da je ugrađuju u samu stranicu. Korišćenje CSS značajno smanjuje veličinu datoteke, što znači ne samo brže vreme učitavanja stranice, već i niže troškove hostinga za često posećene lokacije zbog korišćenja manjeg propusnog opsega.
- Mogućnosti dizajna, kao što su boje i fontovi, lako se menjaju samo izmenom jednog stila u CSS datoteci umesto da se ažurira svaka stranica pojedinačno, smanjujući troškove za modifikovanje sajta.
- Pretraživači mogu pristupiti i indeksirati stranice dizajnirane za veb standarde sa većom efikasnošću.

#### OSNOVNI WWW STANDARDI

*Osnovu World Wide Web-a čini nekoliko standarda, među kojima su najvažniji **URI**, **HTTP**, **HTML***

Osnovu World Wide Web-a čini nekoliko standarda, među kojima su najvažniji:

- **URI** - Uniform Resource Identifier- jedinstveni identifikator resursa koji je univerzalni sistem za referenciranje resursa na vebu, kao što su na primer veb strane
- **HTTP** - HyperText Transfer Protocol- protokol za transfer hiperteksta koji specificira kako komuniciraju veb čitač i veb server međusobno
- **HTML** - HyperText Markup Language- hipertekst markup jezik koji se koristi da se definije struktura i sadržaj hipertekstualnog dokumenta

## ✓ 2.1 URI, URL i URN

### RAZLIKA IZMEĐU URI, URL I URN

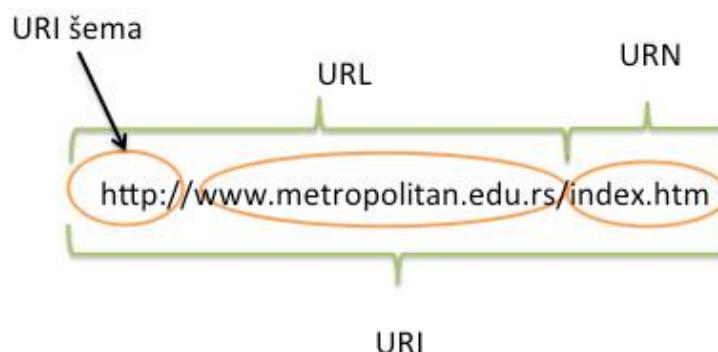
#### *URI se sastoji iz URL i URN-a*

**URI** (Uniform Resource Identifier) predstavlja jedinstveni identifikator resursa. URI je element Internet protokola koji se sastoji od stringa, odnosno od niza karaktera koji odgovaraju dogovorenoj sintaksi. String sadrži ime ili adresu koja se može upotrebiti da se referencira resurs. URI ima funkciju lokatora i imena. Tipičan URI se sastoji od tri elementa:

- naziva šeme mehanizma koji se koristi da se pristupi resursu
- ime domena gde se resurs nalazi
- ime samog resursa dato kao putokaz na hostu.

URI se može klasifikovati kao lokator **URL** (Uniform Resource Locator), kao **URN** (Uniform Resource Name) ili kao oba. URL sadrži informacije o tome kako da se pristupi resursu. U praksi, termin URL zamenjuje termin "veb adresa", iako to nije u potpunosti tačno s obzirom da URL adresa sadrži ime domena i URI šemu. URL predstavlja podskup URI-a. Kada se gleda čitav URI sintaksa, URI počinje sa URI šemom kao što su http, ftp, mailto i dr., zatim sledi dvotačka ":", a zatim sledi URL, pa URN.

URN je podskup URI i on identificuje resurs po imenu u definisanom domenskom prostoru. URN omogućava pozivanje resursa bez razmišljanja o njegovoj lokaciji. Na primer, neke knjige imaju svoj URN (na primer urn: ISBN : 0 - 395 - 36341 - 1) koji omogućava da se neka knjiga definiše po tom broju, istovremeno ne uzimajući u obzir gde se ona tačno nalazi. Primeri jedinstvenih identifikatora dati su na Slici 1.



Slika 2.1.1 Primer odnosa URI prema URL i URN [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### VIDEO

#### *Razlike između URI, URL i URN*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 3

# HTTP protokol

## ŠTA JE HTTP PROTOKOL?

*HTTP je protokol za prenos podataka koji definiše poruke između klijenta i servera na bazi zahteva i odgovora*

Protokol za prenos podataka koji se koristi širom Weba zove se [HTTP \(HyperText Transfer Protocol\)](#). Njime se definišu poruke koje se šalju između klijenta i servera, odnosno zahtevi i odgovori. Svaka interakcija sadrži jedan tekstualni (ASCII) zahtev i odgovor tipa MIME prema standardu RFC 822. Svi klijenti i svi serveri moraju da poštuju ovaj protokol.

Kada čitač želi da prikaže neku stranicu koja se nalazi na serveru, on šalje tom serveru zahtev za tom datotekom. Kao što smo prethodno opisali čitač uspostavlja TCP konekciju sa serverom na priključak 80, mada s formalnog stanovišta taj postupak nije obavezan. Razlog zašto se TCP koristi je taj kako bi se briga o poslatim porukama, njihovim potvrdama i eventualnom ponovom stanju prebacila na sam TCP i rasteretila čitač i server.

U verziji 1.0 protokola HTTP, pošto se veza uspostavi, šalje se samo jedan zahtev i dobija samo jedan odgovor. Tada se TCP veza raskida. Kada su se Web strane sastojale isključivo od HTML teksta, takav postupak je sasvim odgovarao. Međutim, Web strane danas sadrže mnogo vše od običnog teksta, zato verzija 1.1 protokola HTTP podržava trajne veze (engl. [persistent connections](#)).

Sada HTTP protokol omogućava da nakon uspostavljanja TCP veze i jednokratne razmene zahteva i odgovora, mogu da se šalju dodatni zahtevi i da se primaju odgovori na njih. Ukidanjem uspostavljanja i raskidanja TCP konekcije za svaki zahtev, smanjen je broj nepotrebnih operacija po zahtevu. Zahtevi se mogu slati i serijski, tj. može se poslati zahtev 2 pre nego što stigne odgovor na zahtev 1.

HTTP protokol je napravljen prvenstveno kako bi se koristio za rad na Webu, međutim definisane su i funkcionalnosti koje mogu da podržavaju i za buduće objektno orijentisane aplikacije. [Zbog toga HTTP ima operacije koje se odnose na slanje zahteva i odgovora, ali i druge operacije koje se nazivaju metode.](#)

## OSNOVNE DEFINICIJE

*Osnovni pojmovi koji su povezani sa razumevanjem HTTP protokola i rada web-a su resurs, veza, poruka, klijent, server, mrežni prolaz, korisnički agent i tunel*

Da bi se pravilno razumeo HTTP protokol, potrebno je definisati neke osnovne pojmove [1]:

**Resurs** (engl. *resource*) je mrežni objekt koji se sastoji od podataka ili servis koji može biti identifikovan URI-jem. Resurs je najčešće datoteka, ali resurs može takođe biti dinamički generisan rezultat upita, izlaz iz nekog CGI skripta ili nekog drugog skripta.

**Veza** (engl. *connection*) je virtualno kolo transportnog sloja koje se uspostavlja između dva programa u cilju komunikacije.

**Poruka** (engl. *message*) je osnovna jedinica u HTTP komunikaciji koja se sastoji od okteta i koja se prenosi preko ostvarene veze.

**Klijent** (engl. *client*) je program koji uspostavlja vezu da bi poslao neki zahtev.

**Server** je aplikacioni program koji prihvata konekciju da bi poslao odgovor na traženi zahtev. Jedan program može biti i klijent i server zavisno od uloge koju trenutno igra. Severi mogu da deluju kao izvorni server, proksi, mrežni prolaz ili tunel, menjajući ponašanje saglasno zahtevu.

**Izvorni server** (engl. *origin server*) je server na kome se dati resurs nalazi ili će biti kreiran.

**Proksi** (engl. *proxy*) je posredni program koji deluje kao server i klijent u svrhu pravljenja zahteva u ime drugih klijenata. Zahtevi se opslužuju interno ili se prosleđuju drugim serverima.

**Mrežni prolaz** (engl. *gateway*) je server koji deluje kao posrednik nekim drugim serverima. Za razliku od proksija, mrežni prolaz prima zahtev kao da je on izvorni server; klijent koji šalje zahtev ne mora da bude svestan da komunicira sa mrežnim prolazom.

**Korisnički agent** (engl. *user agent*) je klijent koji inicira zahtev. Ovo je najčešće čitač veb strana, ili neki robot program za prevlačenje veb strana.

**Tunel** (engl. *tunnel*) je posrednički program koji deluje kao slepi prenosnik između dva čvora na mreži. Kada se jednom aktivira, on se ne posmatra kao učesnik u HTTP komunikaciji, mada je možda bio inicijalizovan HTTP zahtevom. Tunel prestaje da postoji kada obe povezane strane zatvore vezu.

## ▼ Poglavlje 4

### Način funkcionisanja HTTP protokola

#### STRUKTURA ZAHTEVA I ODGOVORA HTTP PROTOKOLA

*Klijent šalje zahtev sa određenom strukturom poruke definisane standardom, dok klijent obrađuje taj zahtev i šalje ogovor koji je takođe u određenom formatu*

HTTP protokol je protokol tipa zahtev/odgovor. Klijent šalje serveru zahtev koji se sastoji od:

- metode zahteva,
- URI-ja,
- verzije protokola,
- poruke koja sadrži modifikatore zahteva,
- informacija o klijentu
- opcionog tela sa sadržajem

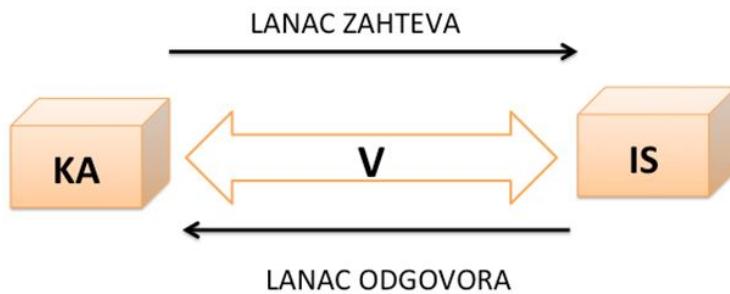
Odgovor servera se sastoji od:

- statusne linije koja uključuje verziju protokola poruke i poruku o uspešnosti odgovora ili kod greške u slučaju neuspešnog odgovora
- informacije o serveru
- meta informacije o entitetu koji se šalje
- tela entiteta

#### JEDNOSTAVNA HTTP VEZA

*Jednostavna HTTP veza se uspostavlja između klijenta, odnosno korisničkog agenta (na primer, veb čitač) koji upućuje serveru zahtev koji treba da se primeni na nekom resursu*

Većina HTTP komunikacija se inicijalizuje od strane korisničkog agenta (KA) i sastoji se od zahteva koji treba da se primeni na resurse nekog izvorишnog servera (IS). U najjednostavnijem slučaju, ovo se postiže jednom vezom (V) između korisničkog agenta i izvorишnog servera.



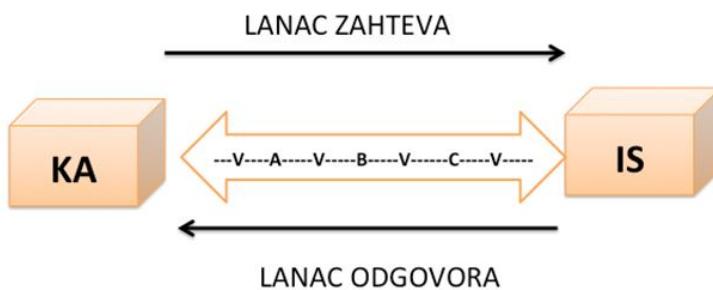
Slika 4.1.1 Jednostavna veza između korisničkog agenta (KA) i izvođačne strane (IS) [Izvor: Autor]

## HTTP VEZA SA TRI POSREDNIKA

*Kada između klijenta i servera postoje posrednici, zahtev će proći kroz odvojene veze (eng. link)*

Složeniji slučaj se javlja kada je jedan ili više posrednika uključeno u lanac zahtev/odgovor. Postoje tri česta oblika posrednika: proksi, mrežni prolaz i tunel. Proksi je prosleđujući agent koji prima zahtev za URI-jem u njegovoj apsolutnom obliku, prepisuje celu ili samo deo poruke i prosleđuje ga ka serveru koji je identifikovan URI-jem. Mrežni prolaz je prijemni agent, koji se ponaša kao sloj iznad nekog drugog servera, i ako je potrebno, prevodi zahtev na protokol servera koji je ispod njega. Tunel deluje kao veza između dva čvora koja ne menja poruku. Koristi se kada je potrebno komunikaciju izvršiti preko posrednika (kakav je na primer zaštitni zid – firewall) iako posrednik ne razume sadržaj poruke. Na narednoj slici je prikazan slučaj kada između korisničkog agenta i izvođačnog servera postoje tri posrednika. Zahtev i odziv će proći kroz četiri odvojene veze. Ovde se namerno pravi razlika zbog toga što neke HTTP komunikacione opcije mogu da se primene samo za povezivanje sa najbližim ne-tunel susedom, samo sa krajnjom tačkom lanca, ili sa svima u lancu

Učesnici u ovoj komunikaciji mogu istovremeno da budu angažovani u višestrukim komunikacijama. Na primer B može da prima zahteve i od drugih klijenata osim od A i/ili šalje zahteve i drugim serverima, a ne samo serveru C.

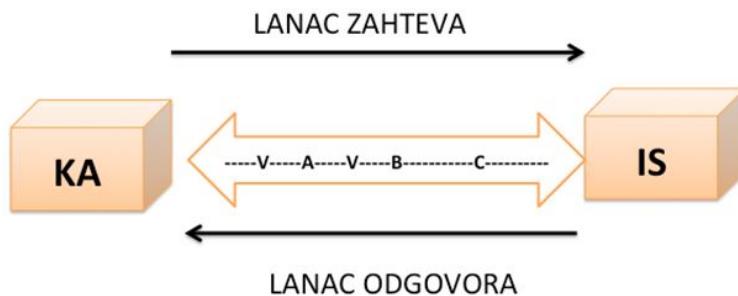


Slika 4.1.2 Veza sa tri posrednika [Izvor: Autor]

## VEZA SA NALAŽENJEM ODGOVORA U POSREDNIKU

*Kada neki član u lancu komunikacije ima keširani odgovor na zahtev korisničkog agenta, onda će se lanac zahtev/odgovor skratiti.*

Svaki član komunikacije koji ne deluje kao tunel može da ima interni keš za manipulaciju sa zahtevima. Ukoliko neki član u lancu komunikacije ima keširani odgovor na zahtev korisničkog agenta, onda će se lanac zahtev/odgovor skratiti. Na sledećoj šemi je prikazan primer u kome B ima keširanu kopiju ranijeg odziva izvođenog servera dobijenu preko C. U tom slučaju nije potrebno uspostavljati vezu između B i C i dalje prema IS, čime se lanac zahtev/odgovor skratio a proces ubrzao.



Slika 4.1.3 Veza sa nalaženjem odgovora u posredniku B [Izvor: Autor]

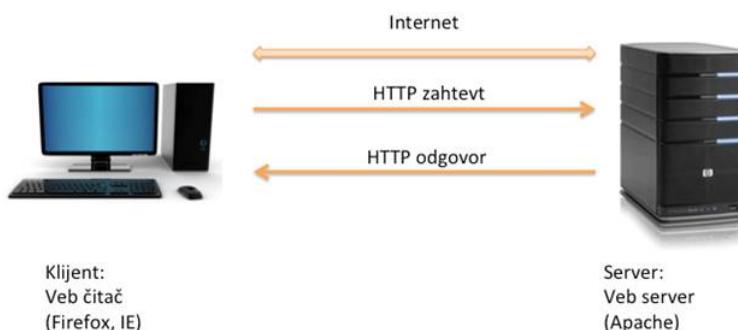
## ▼ 4.1 HTTP karakteristike

### KLIJENT/SERVER HTTP KOMUNIKACIONI MODEL

#### *Komunikacija između veb klijenta i veb servera je definisan HTTP-om*

Da bi se veb stranice i sadržaj koji oni sadrže i šalju (slike, rezultati upita i sl.) mogli isporučiti korisniku kada korisnik to zahteva, neophodno je da se prate određena pravila. **Hypertext Transfer Protocol** (HTTP) je protokol koji predstavlja skup pravila koja se koriste za prenos stranica na Internetu. Drugim rečima, komunikacija između veb klijenta i veb servera je definisan HTTP-om. U ovom slučaju, program koji šalje zahtev serveru (veb čitač) predstavlja klijent. Na osnovu zahteva server formuliše odgovor i šalje ga klijentu.

HTTP koristi klijent/server model koji omogućava klijentu da uspostavi vezu i da pošalje zahtev. Server šalje odgovor, obično zajedno sa traženim sadržajem. Kada se odgovor isporuči, server raskida uspostavljenu vezu.



Slika 4.2.1 Klijent/server HTTP komunikacioni model [Izvor: Autor]

HTTP ima nekoliko važnih karakteristika:

- HTTP/1.0 podrazumeva da kada se zahtev ispunii, da klijent raskida vezu sa serverom. Vezu je potrebno uspostaviti za svaki novi zahtev/odgovor. Server nema potrebu da održava otvorenu sesiju kako bi slao zahtevani sadržaj.
- HTTP/1.1 koristi perzistentne veze što podrazumeva da se koristi ista konekcija za slanje više zahteva i odgovora, a da pri tom nema potrebu da otvara novu konekciju za svaki zahtev i odgovor.
- HTTP je nezavistan od medija i može da prenosi bilo koji tip podataka dokle god i klijent i server mogu da ih obrade.
- HTTP ne zahteva ni od klijenta ni od servera da održava podatke o konekciji, tako da ni klijent ni server ne čuvaju informacije jedan o drugom kada se zahtev ispunii.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## HTTP ZAHTEV

*Najčešće korišćena HTTP metoda koja se koristi u zahtevu je metoda GET. HTTP metode su GET, POST, PUT, HEAD, DELETE, TRACE, CONNECT i OPTIONS*

Prvi red u HTTP zahtevu (engl. **request**) se razlikuje od prvog reda HTTP odgovora. Prvi red sadrži ime HTTP metode, HTTP verziju i putanju do traženog resursa. Najčešće korišćena HTTP metoda koja se koristi u zahtevu je metoda **GET**. Druge HTTP metode su **POST, PUT, HEAD, DELETE, TRACE, CONNECT i OPTIONS**. Verzija HTTP se uvek navodi u obliku “/HTTP/x.x” gde “x.x” predstavlja broj verzije.

Niže navedeni primer predstavlja primer GET metode i to zahtev za pregled resursa na stranici www.metropolitan.ac.rs . Može se primetiti da se vidljiva verzija HTTP, koja je 1.1. Zatim u zahtevu je navedeno ime hosta i “user-agent” koji obično predstavlja ime veb čitača koji šalje zahtev kao i njegovu verziju. “Accept” predstavlja metodu koja navodi šta može veb čitač da prihvati od jezika, metode šifrovanja i tip resursa.

GET / HTTP/1.1

Host: www.metropolitan.ac.rs

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:8.0) Gecko/20100101 Firefox/8.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: en-us,en;q=0.5

Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7

Connection: close

## STRUKTURA HTTP ZAHTEVA - VIDEO

### *Delovi HTTP zahteva*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## HTTP ODGOVOR

*Početni red odgovora sadrži HTTP verziju, statusni deo koda koji opisuje rezultat zahteva i reč na engleskom jeziku koja opisuje taj rezultat*

Početni red odgovora (engl. **response**) sadrži HTTP verziju, statusni deo koda koji opisuje rezultat zahteva i reč na engleskom jeziku koja opisuje taj rezultat. U primeru ta engleska reč je "OK". Primer prvog reda HTTP odgovora je:

HTTP/1.1 200 OK ili

HTTP/1.1 404 Not Found

U prethodnom primeru, zahtev je prikazan za hosta www.metropolitan.ac.rs. Odgovor na taj zahtev izgleda ovako:

HTTP /1.1 200 OK

Date: Tue,26 June 2020 11:37:00 GMT

Server: Apache-Coyote/1.1

Content-Type: text/html; charset=UTF-8

Connection: close

Transfer-Encoding: chunked

(Content)

Ukoliko je klijent tražio stranicu ili resurs koji ne postoji, na primer sa sledećih zahtevom:

GET /it101.html HTTP/1.1

Host: www.metropolitan.ac.rs

Tada će server odgovoriti sa sledećim odgovorom

HTTP /1.1 404 Not Found

Date: Tue, 26 June 2020 11:52:00 GMT

Server: Apache-Coyote/1.1

Content-Type: text/html

Connection: close

Transfer-Encoding: chunked

## METODE HTTP PROTOKOLA

*Token metode pokazuje koja će se metoda primeniti na resursu identifikovanom Request-URI-jem*

Startna linija poruke sa zahtevom od klijenta ka serveru se naziva linija zahteva. Ona se sastoji od tri elementa:

- Tokena metode koja će se primeniti na resurs (methods)
- Identifikator resursa (Request-URI)
- Verzije protokola koji se koristi (HTTP-Version)

koji su međusobno odvojeni karakterom **SP** (**space**). Na kraju linije zahteva je karakter **CRLF**. Opšti oblik linije zahteva je:

**Request-Line = Method SP Request-URI SP HTTP-Version CRLF**

Token metode pokazuje koja će se metoda primeniti na resursu identifikovanom Request-URI-jem. Moguće metode zahteva su:

- OPTIONS
- GET
- HEAD
- POST
- PUT
- DELETE
- TRACE

## OPISI HTTP METODA

*HTTP metode su GET, HEAD, PUT, POST, DELETE, TRACE, OPTIONS*

Metod OPTIONS predstavlja zahtev za informacijama o raspoloživim komunikacionim opcijama u lancu zahtev-odgovor identifikovanom pomoću URI-ja. Na taj način klijent opcije ili zahteve vezane za resurs, kao i mogućnosti servera bez angažovanja samog servera.

Metod **GET** omogućuje preuzimanje informacija identifikovanih URI-jem. To može da bude neka datoteka ili podaci koje je generisao neki proces.

Metod **HEAD** je isti kao GET s tom razlikom što server ne sme u odgovoru da vrati telo poruke. Ovaj metod se koristi kada je potrebno dobiti meta informacije o entitetu koji je zahtevan, a da se pri tom ne prenosi i samo telo entiteta. Na taj način je moguće lako testirati validnost, dostupnost i poslednju izmenu hiperlinkova.

Metod **DELETE** zahteva da odredišni server obriše resurs identifikovan URI-jem.

Metod **TRACE** se koristi za vraćanje eha poslatog zahteva odredišnom serveru. Drugim rečima, klijent može da vidi kakvu je poruku primio odredišni server. Na taj način klijent može da sazna koji su međuserveri menjali originalni zahtev, što je vrlo korisno za dijagnostiku.

Metod **POST** se koristi da se odredišnom serveru prosledi entitet koji se nalazi u zahtevu, pridruživanjem poslatog entiteta postojećem. POST metod omogućuje sledeće funkcije:

- Slanje napomena o postojećim resursima,
- Slanje poruka njuzgrupama, mejling listama, elektronskim oglasnim tablama
- Slanje bloka podataka, kao što su podaci iz popunjene forme, procesu koji obrađuje podatke,
- Upisivanje podataka u neku bazu podataka korišćenjem operacije append.

Metod **PUT** zahteva da se entitet primi i sačuva pod identifikacijom koja je definisana. Ako se URI odnosi na već postojeći resurs, entitet treba smatrati modifikovanom verzijom onog koji postoji na odredišnom serveru. Ako URI ne ukazuje na postojeći resurs, odredišni server može da kreira resurs sa tim URI-jem.

## VIDEO

### *HTTP metode*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 5

# Programiranje sa klijentske i serverske strane

## TEHNIKA PROGRAMIRANJA SA KLIJENTSKE STRANE

*Programiranje sa klijentske strane se može postići pisanjem skripta i ugnježdenim objektima*

Na klijentskoj strani veb se nalazi veb klijent, koji je uobičajeno neki čitač veb strana (engl. **web browser**). Zadatak veb čitača je da prikaže veb stranicu dobijenu od veb servera. Pri tom se korisniku ne pruža nikakva mogućnost za interakciju. Interakcija zahteva da se neke rutine izvršavaju i na klijentskoj strani. Mogućnost izvršenja koda na klijentskoj strani može da bude od velikog značaja.

Posmatrajmo primer izvršenja neke veb aplikacije u kojoj udaljeni klijent treba da popuni polja u nekoj formi. Validnost unetih podataka se može obaviti na serverskoj strani, ali u slučaju nevalidnih ili nepotpunih podataka korisnik mora više puta da pristupa serverskoj strani. Umesto toga, korišćenjem neke rutine koja može da se izvrši na klijentskoj strani verifikacija se može izvršiti lokalno, što je mnogo brže i smanjuje mrežni saobraćaj.

Postavlja se pitanje da li je i na klijentskoj strani moguće realizovati određenu interaktivnost, odnosno izvršiti neki program. Odgovor je potvrđan jer i klijentska strana koristi računarski sistem, pa može i da izvršava neki programski kod. Problem je, međutim, u tome što klijenti veb stranama pristupaju korišćenjem različitih hardverskih i softverskih platformi i različitih veb čitača, što otežava pisanje rutina koje će se izvršavati na klijentskoj strani.

Izvršavanje programa unutar veb strane na klijentskoj strani može da se postigne na dva načina:

- pisanjem skripta
- ugnezđavanjem programiranih objekata.

Skript jezici nisu tako kompleksni kao uobičajeni programski jezici, pa nisu podesni za pisanje kompleksnih programa. Uglavnom se koriste kada je potrebno napraviti jednostavne rutine kojim se proveravaju podaci na klijentskoj strani ili kada je potrebno manipulisati elementima veb stranice. Za veb programere je najlakši način dodavanja dinamičkog aspekta veb strani pisanje skripta koji se izvršava na klijentskoj strani (**client-side scripting**), uglavnom, korišćenjem JavaScript jezika.

Ako je reč o složenijim zahtevima onda se koristi tehnika ugnezđenih objekata. Ovi objekti se programiraju u računarskim jezicima kao što su Java ili C++.

## KADA SE KORISTI PROGRAMIRANJE SA KLIJENTSKE STRANE?

*Primeri skripting jezika koji se koriste na klijentskoj strani su JavaScript, VBScript, HTML, CSS, AJAX, jQuery i drugi*

Programiranje sa klijentske strane se koristi kada:

- želimo da postignemo interaktivnost veb stranice
- uopšteno kada želimo dinamičnost
- imamo interakciju sa privremenim skladištem
- pravimo interfejs između korisnika i servera
- šaljemo zahtev serveru
- imamo interakciju sa lokalnim skladištem
- želimo da pružimo udaljeni pristup serverskom delu programa
- želimo da obavimo verifikaciju lokalno, kako bi smanjili mrežni saobraćaj

Postoje mnogi skripting jezici koji se koriste na klijentskoj strani, od kojih su neki:

- JavaScript
- VBScript
- HTML (struktura)
- CSS (dizajn)
- AJAX
- jQuery itd.

Pored ovih mogu se koristiti i drugi jezici za modelovanje, grafiku, animaciju i dodatne funkcionalnosti.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## RAZLOZI ZA PROGRAMIRANJE SA SERVERSKE STRANE

*Kada je potrebno dodati interaktivnost veb strani ili kada je potrebna veća obrada podataka, onda je razumnije tu funkcionalnost ugraditi na serverskoj strani*

Kada je potrebno dodati interaktivnost veb strani ili kada je potrebna veća obrada podataka, onda je razumnije tu funkcionalnost ugraditi na serverskoj strani, osim ako interaktivnost nije čisto klijentska po prirodi, kao što je, na primer, u slučaju animacije ili provere podataka u formi. Postoje dva razloga zašto se radi programiranje serverske strane:

- kada je potrebno da se ima potpuna kontrola, s obzirom da se jedino serverska strana može kompletno upravljati, što je posebno važno sa aspekta bezbednosti

- klijentska strana generalno nedefinisana jer se ne zna ni hardverska ni softverska platforma, a i instalirani veb čitači mogu biti različiti. To ograničava mogućnosti za ozbiljniju obradu podataka na klijentskoj strani.

Neki od primera kada je poželjno programirati sa serverske strane su:

- kada se obrađuje korisnikov unos
- kada je potrebno obaviti interakciju sa serverom ili njegovim skladištem
- kada se šalju upiti bazi podataka ili kada se ona ažurira

Za programiranje serverske strane se koriste različite tehnologije, kao što su:

**1. CGI programi**

**2. Serverski moduli - Web server API programi**

- Apache moduli
- Java servleti

**3. Server-side scripting**

- Server-side Includes (.shtml)
- ASP ili ASP.NET (.asp/.aspx)
- ColdFusion (.cfm/.cfmx)
- PHP (.php)
- Java Server Pages (.jsp)

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 6

# Responzivni veb dizajn

## ŠTA JE RESPONZIVNI VEB DIZAJN?

*Responzivni veb dizajn je strategija koja omogućava da veb sajt prilagođava svoj izgled uređajima na osnovu veličine prozora njegovog pretraživača*

Većina pretraživača (engl. **web browser**) na malim uređajima kao što su pametni telefoni i tableti smanjuju veb stranicu da bi se uklopila u ekran i obezbedila mehanizme za zumiranje i navigaciju po veb stranici. Iako taj pristup funkcioniše, to ne predstavlja uvek dobro korisničko iskustvo. Tekst je u takvim situacijama često premali za čitanje, linkovi premali da se lako na njih klikne, a svo to zumiranje i pomeranje odvlači pažnju korisnika.

**Responzivni veb dizajn** (engl. **responsive web design**) je strategija koja omogućava da veb sajt prilagođava svoj izgled uređajima na osnovu veličine prozora za prikaz (prozora pretraživača). Trik za responzivni veb dizajn je korišćenje jednog HTML dokumenta na svim uređajima, ali primena različitih stilova na osnovu veličine ekrana kako bi se obezbedio najoptimizovaniji izgled za taj uređaj. Na primer, kada se stranica gleda na pametnom telefonu, pojavljuje se u jednoj koloni sa velikim linkovima za njihov lakši odabir. Međutim, kada se ta ista stranica pregleda na velikom ekranu koji koristi desktop računar, sadržaj se preuređuje u više kolona sa tradicionalnim elementima za navigaciju.

Možete videti mnoge inspirativne primere na sajtu galerije Media Queries ([www.mediaqueri.es](http://www.mediaqueri.es)). Pokušajte da otvorite dizajn u svom pretraživaču, a zatim promenite veličinu prozora veoma uskom i veoma širokom, i gledajte kako se izgled menja u zavisnosti od veličine prozora.

Prilagodljivi veb dizajn pomaže u vezi sa izgledom, ali nije rešenje za sve izazove koji postoje kod mobilnog veb dizajna. Činjenica je da pružanje najboljeg iskustva za korisnike i njihov izabrani uređaj može zahtevati optimizacije koje prevazilaze prilagođavanje izgleda. Neki problemi se bolje rešavaju korišćenjem servera za otkrivanje uređaja i njegovih mogućnosti, a zatim donošenje odluka o tome šta da se pošalje na klijentsku stranu. Koristeći progresivno poboljšanje (engl. **progressive enhancement**), možete pružiti osnovno iskustvo za pretraživače i uređaje koji nemaju velike mogućnosti, ali uređaji koji su tehnološki napredniji ka njima se mogu koristiti prikazi sa poboljšanim opcijama.

Za neke sajtove i usluge, možda bi bilo bolje dase napravi zasebna mobilna lokacija sa prilagođenim interfejsom i skupom funkcija koji koristi prednosti telefonskih mogućnosti kao što je geolokacija. Ipak, iako responzivni dizajn neće sve popraviti, on je važan deo rešenja za pružanje zadovoljavajućeg iskustva na širokom spektru pretraživača.

## DOSTUPNOST Veba

*W3C je pokrenuo Inicijativu za dostupnost veba (WAI) kako bi se pozabavio potrebom da se veb učini dostupnim za sve*

Važno je imati na umu da ljudi pristupaju vebu na mnogo različitih načina — pomoću čitača ekrana, Brajevog pisma, lupa, džoystika, nožnih pedala i tako dalje. Veb dizajneri moraju da prave stranice na način koji dostupan je za sve, bez obzira na mogućnosti korisnika i uređaj koji se koristi za pristup vebu.

Iako su namenjene korisnicima sa invaliditetom kao što su slab vid ili ograničena pokretljivost, tehnike i strategije razvijene za pristupačnost takođe koriste drugim korisnicima sa manje od optimalnog iskustva pregledanja, kao što su ručni uređaji ili tradicionalni pretraživači preko sporih internet veza ili sa slikama. Stranice koje zadovoljavaju široku dostupnost se efikasnije indeksiraju od strane pretraživača kao što je Google.

Postoje četiri široke kategorije invaliditeta koje utiču na to kako ljudi komuniciraju sa svojim računarima i informacijama o njima:

- **Oštećenje vida.** Ljudi sa slabim vidom ili bez vida mogu da koriste pomoćni uređaj kao što je čitač ekrana, ekran na Brajevom azbuku ili lupa za prikaz sadržaja sa ekrana. Oni takođe mogu jednostavno da koriste funkciju zumiranja teksta pretraživača da bi tekst učinili dovoljno velikim za čitanje.
- **Poremećaj pokretljivosti.** Korisnici sa ograničenom upotrebom ili bez upotrebe ruku mogu da koriste posebne uređaje kao što su modifikovani miševi i tastature, nožne pedale ili džoystici za navigaciju vebom i unos informacija.
- **Oštećenje sluha.** Korisnici sa ograničenim sluhom ili bez sluha će propustiti audio aspekte multimedije, pa je neophodno obezbediti alternative, kao što su transkripti za audio zapise ili titlovi za video.
- **Kognitivno oštećenje.** Korisnici koji imaju poteškoća sa pamćenjem, razumevanjem čitanja, rešavanjem problema i ograničenjima pažnje imaju koristi kada su sajtovi dizajnirani jednostavno i jasno.

W3C je pokrenuo Inicijativu za pristupačnost veba (WAI) kako bi se pozabavio potrebom da se veb učini dostupnim za sve. WAI sajt ([www.w3.org/WAI](http://www.w3.org/WAI)) je odlična polazna tačka da se nauči više o dostupnosti veba. Jedan od dokumenata koje je izradioW da bi pomogao programerima da kreiraju pristupačne sajtove su Smernice za pristupačnost veb sadržaja (WCAG i WCAG 2.0). Možete ih sve pročitati na <https://www.w3.org/WAI/standards-guidelines/wcag/>.

## ✓ Poglavlje 7

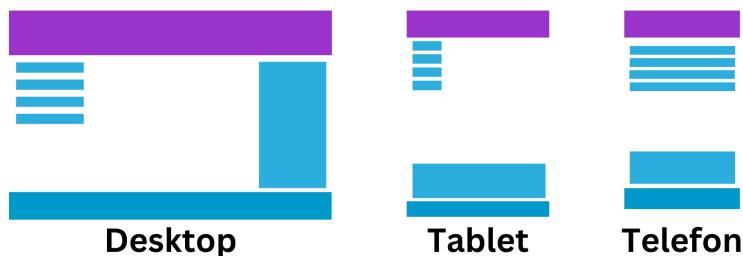
# Pokazne vežbe: responzivni veb dizajn

## RESPONZIVNI VEB DIZAJN

*Responzivni veb dizajn je kreiranje veb stranica koje izgledaju dobro na svim uređajima.*

Predviđeno vreme za pokazne vežbe je 20 minuta.

Responzivni veb dizajn je pristup koji sugeriše da dizajn i razvoj treba da odgovore na ponašanje i okruženje korisnika na osnovu veličine ekrana, platforme i orientacije. Praksa se sastoji od mešavine fleksibilnih mreža i rasporeda, slika i inteligentne upotrebe CSS medijskih upita (eng. media queries). Kako korisnik prelazi sa svog laptopa na tablet, veb lokacija bi trebalo automatski da se prebacuje da bi se prilagodila rezoluciji, veličini slike i sposobnostima skriptinga. Prilagodljiv veb dizajn će se automatski prilagoditi različitim veličinama ekrana i okvirima za prikaz, kao što je prikazano na slici.



Slika 7.1.1 Prilagođavanje rasporeda okviru za prikaz [Izvor: Autor]

## ZADATAK ZA DISKUSIJU

*Kroz diskusiju odgovorite na sledeća pitanja.*

Pogledajte i pažljivo analizirajte prikazane sajtove na sledećim linkovima:

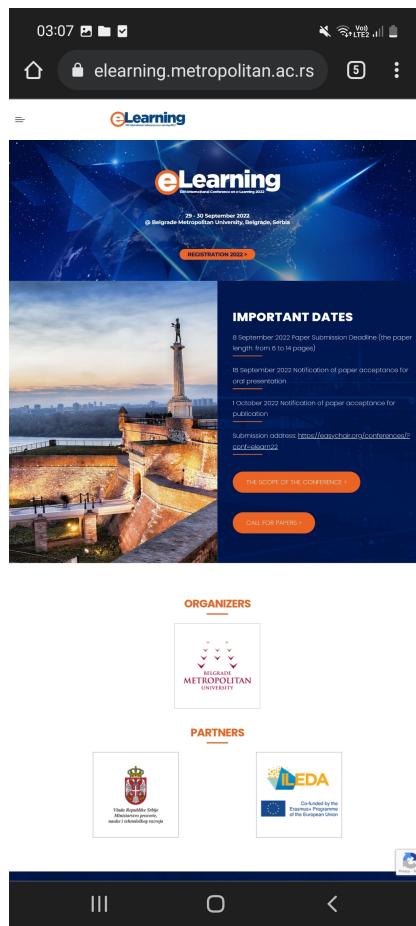
- <https://mediaqueri.es/cor/>
- <https://mediaqueri.es/ccy/>
- <https://mediaqueri.es/str/>

Kroz diskusiju analizirajte sledeće:

1. Da li prikazani veb sajtovi primenjuju responzivni dizajn?
2. Šta se sve menja prilikom promene veličine ekrana/uređaja koji se koristi?
3. Na slikama 2 i 3 prikazan je veb sajt kome je pristupljeno na mobilnom telefonu. Koja je razlika između prikazanog? Koji izgled sajta je prilagođen mobilnom telefonu? Šta je sve izmenjeno?
4. Pretpostavite da je potrebno da kreirate veb sajt za onlajn prodavnicu kozmetike. Napravite idejni plan kako biste primenili responzivni dizajn. Kako biste obezbedili da sajt bude prilagođen različitim uređajima?



Slika 7.1.2 Izgled veb sajta [Izvor: Autor]



Slika 7.1.3 Izgled veb sajta 2 [Izvor: Autor]

## ▼ 7.1 Kreiranje prototipa u alatu Marvel

### PRIPREMA ZADATKA - MATCH BEAUTY

*Definisanje scenarija upotrebe sajta.*

Predviđeno vreme pokazne vežbe je 50 minuta.

Prototip je simulacija stvarnog sistema koja se može brzo razviti i prototip demonstrira funkcionalnost vaše ideje pre nego što napravite konačnu verziju. Prototip pomaže da se shvati kako će proizvod izgledati, kako će funkcionisati i kakav će biti radni tok pri izvršavanju određenih zadataka.

Match Beauty je veb sajt namenjen kupovini kozmetičkih proizvoda. U našem primeru, posmatraćemo samo deo sajta namenjen kupcima. Kupci imaju mogućnost da pregledaju i naruče željene proizvode.

**Scenario upotrebe za koji ćemo izgraditi prototip:**

Korisnik pristupa sajtu. Može da odabere jednu od ponuđenih opcija: Login, Register ili Guest. Nakon uspešne registracije I/ili logovanja (ili odabira Guest opcije) korisnik pristupa početnoj strani sajta. Na početnoj strani su prikazane kategorije poput "Preporučeno za Vas" I "Najprodavaniji proizvodi" I opciju pretrage proizvoda. Nakon odabira željene kategorije, kupac dobija listu proizvoda. Nakon što odabere željeni proizvod prikazane su detaljnije informacije o proizvodu, kao I opcije za odabir količine, dodavanja proizvoda u kropu I povratka na početnu stranicu. Nakon što korisnik doda sve željene proizvode u korpu, korisnik ima mogućnost da pregleda I modificuje korpu I naruči proizvode.

### **Zadatak:**

Vreme potrebno za izradu zadatka: 30 minuta.

Studenti klasične nastave će biti podeljeni u dve grupe: jedna grupa studenata će kreirati prototip veb sajta Match Beatuy namenjenog računarama, dok će druga grupa studenta kreirati prototip veb sajta prilagođenog mobilnim telefonima. Studenti onlajn nastave mogu samostalno da odaberu tip uređaja za koji kreiraju prototip ili da kreiraju prototipe za oba uređaja.

## MARVEL ALAT ZA IZRADU PROTOTIPA

### *Uvod o MarvelApp alatu*

Alat Marvel (<https://marvelapp.com/>) pretvara statične modele, wireframes I dizajne u interaktivna mobilna I veb iskustva, sve bez potrebe za kodiranjem. Krajnji rezultat je prototip koji izgleda kao pravi sajt/aplikacija, ali za izgradnju je potrebno daleko manje vremena.

Prototipe možemo korisiti da bi:

- pronašli konkretne zahteve I ideje od ljudi koji se ne bave programiranjem
- testirali ideje za veb sajtove I aplikacije
- razumeli ponašanje korisnika
- demonstrirali ideje zainteresovanim stranama.

Sa Marvel-om možete kreirati prototip koji bi mogao da demonstrira vaše ideje na brojnim različitim platformama koje zatim možete podeliti sa programerima, dizajnerima, zainteresovanim stranama ili čak sopstvenim klijentima.

Na sledećem linku možete pročitati uputstvo za kreiranje prvog prototipa pomoću MarvelApp (članak sadrži I video uputstva): <https://help.marvelapp.com/hc/en-us/articles/360002536038>

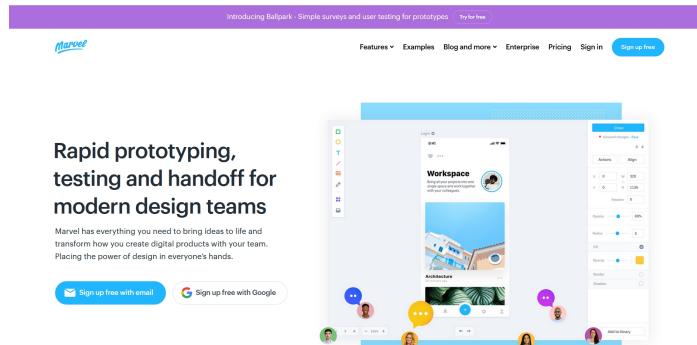
Primer prototipa: <https://marvelapp.com/examples/banking-app>

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

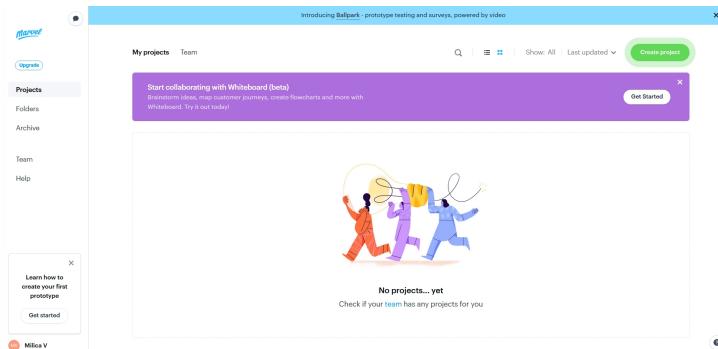
## KREIRANJE PROJEKTA U MARVEL-U

*Da bi smo kreirali prototip, idemo na početnu stranicu  
<https://marvelapp.com/>*

Da bi smo kreirali prototip, idemo na početnu stranicu (<https://marvelapp.com/>) i kreiramo analog. Besplatan nalog omogućava da imamo jedan projekat, međutim kada postojeći projekat uklonimo možemo kreirati novi. Nakon što se registrujemo, potrebno je da se prijavimo na sajt.



Slika 7.2.1 Početna stranica [Izvor: Autor]

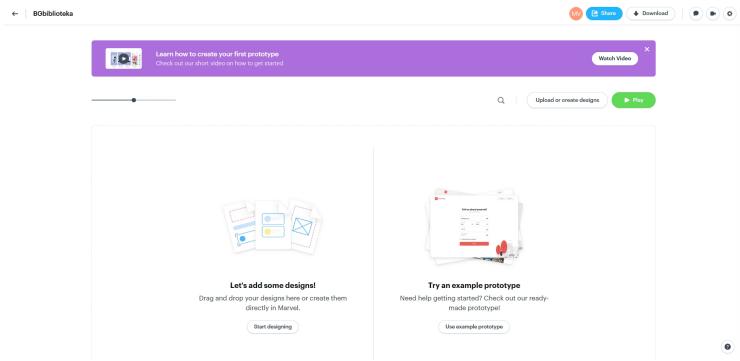


Slika 7.2.2 Dashboard [Izvor: Autor]

Nakon uspešne prijave, dobijamo prikaz Dashboard-a. Biramo opciju "Create project" i unosimo ime i **tip projekta** (obratite pažnju da odaberete pravi tip). Potom možemo uploadovati postojeći dizajn ili kreirati u Marvel-u odabirom opcije "Start designing" - što ćemo mi odabrati.

The screenshot shows the "Create prototype project" form. It has fields for "Project name" (filled with "BGbiblioteka") and "Project type" (set to "Website/TV Any resolution"). There's also an "Invite-only" toggle switch which is turned off. At the bottom, there's a green "Create project" button.

Slika 7.2.3 Kreiranje novog projekta [Izvor: Autor]



Slika 7.2.4 Nov projekat [Izvor: Autor]

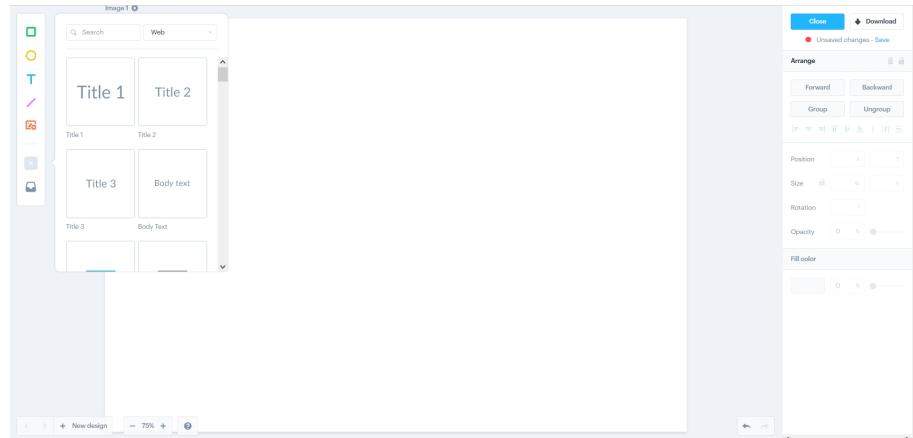
## DIZAJN EKRANA

### *Iz paleta sa leve strane biramo pojedinačne elemente*

Iz paleta sa leve strane biramo pojedinačne elemente - dodavanje slike (sa interneta ili računara), dodavanje oblika ili teksta, i dodavanje elemenata karakterističnih za izabran tip interfejsa (dugme, naslov, meni, dropdown, slajderi, grupe elemenata - login stranice i slično).

Elemente dodajemo na belu površinu i u meniju sa desne strane možemo da vršimo dodatna podešavanja.

Kada izaberemo element ili set elemenata - kao što je pripremljen dizajn za login stranicu, te elemente možemo da modifikujemo. Ako hoćemo da menjamo pojedinačan element grupu (nezavisno od grupe) potrebno je da ih razgrupišemo (ungroup).

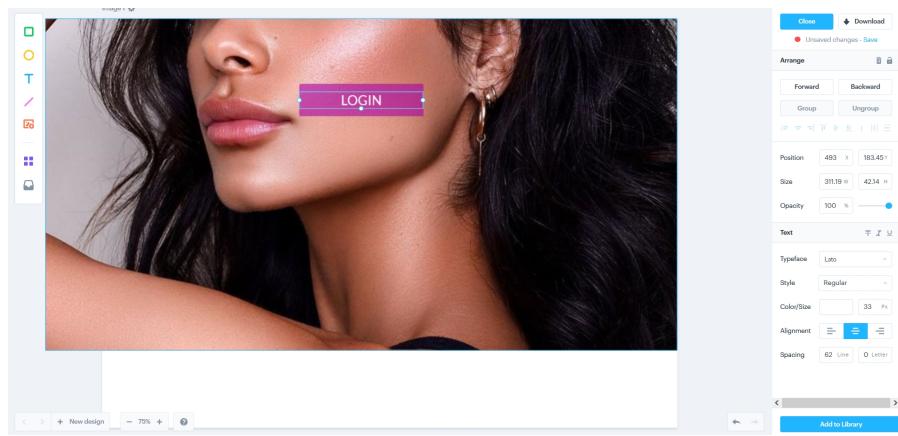


Slika 7.2.5 Početak kreiranja ekrana [Izvor: Autor]

Isto tako, ako hoćemo sa nekoliko elemenata da manipulišemo kao da su jedan potrebno je da ih grupišemo.

Važno je da dizajn sačuvamo pre nego što kliknemo na "Close".

Uvek možemo da se vratimo da doradimo dizajn odabiranjem opcije "Edit design".



Slika 7.2.6 Prikaz kreiranog elementa [Izvor: Autor]

## EKRANI KOJE DIZAJNIRAMO

*Za naš prototip kreiramo 7 ekrana - dizaj ne mora da bude identičan prikazanom*

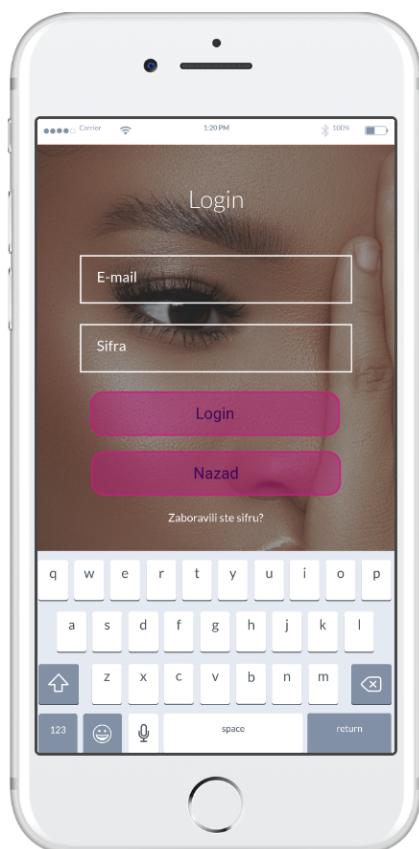
Za naš prototip kreiraćemo sledeće ekrane:

- Ekran 1: Landing page(slika 7)
- Ekran 2: Stranica za login (slika 8)
- Ekran 3: Početna stranica sa prikazom kategorija (slika 9)
- Ekran 4: Stranica sa prikazom proizvoda za odabranu kategoriju(slika 10)
- Ekran 5: Stranica sa detaljima o proizvodu (slika 11)
- Ekran 6: Stranica sa prikazom korpe (slika 12)
- Ekran 7: Stranica sa formom za unos podataka za naručivanje (slika 13)

Vaš dizajn ne mora da bude identičan prikazanom - potrebno je da sadrži tražene elemente. Ekrani 2, 3 i 4 moraju da sadrže dugme za pristup korpi. Stranice moraju da sadrže dugme za povratak na početni ekran (ekran 3).



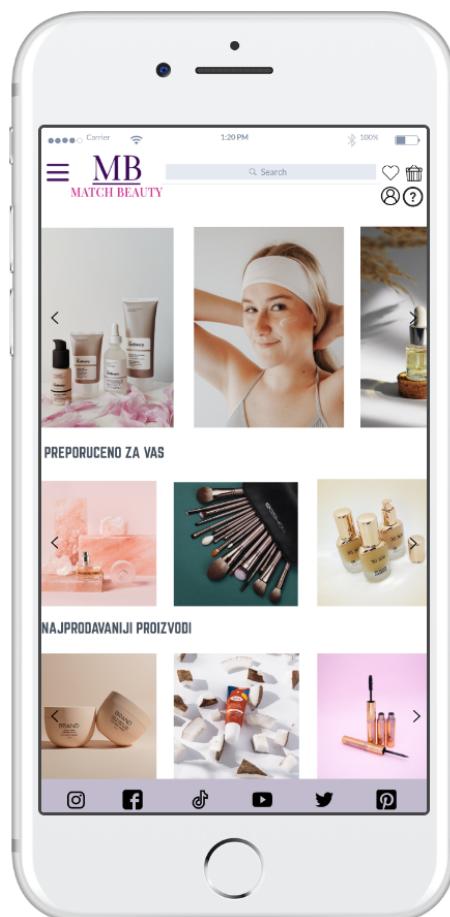
Slika 7.2.7 Ekran 1 [Izvor: Autor]



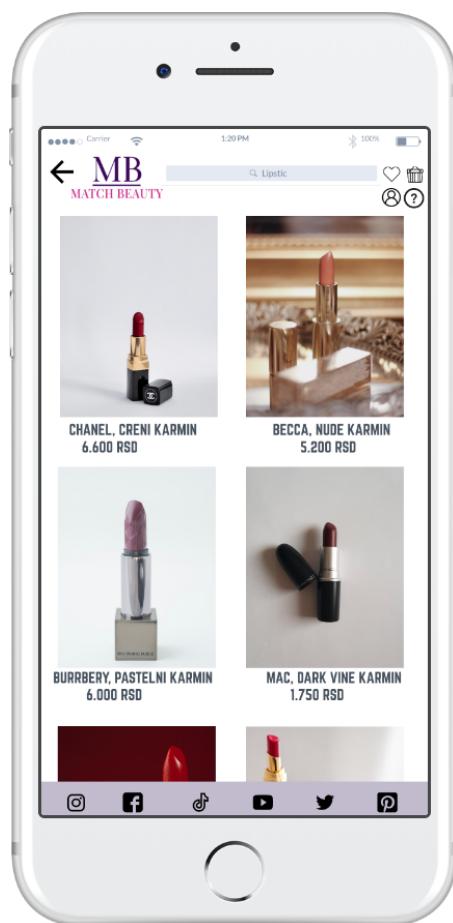
Slika 7.2.8 Ekran 2 [Izvor: Autor]

## IZGLED EKRANA 3-5

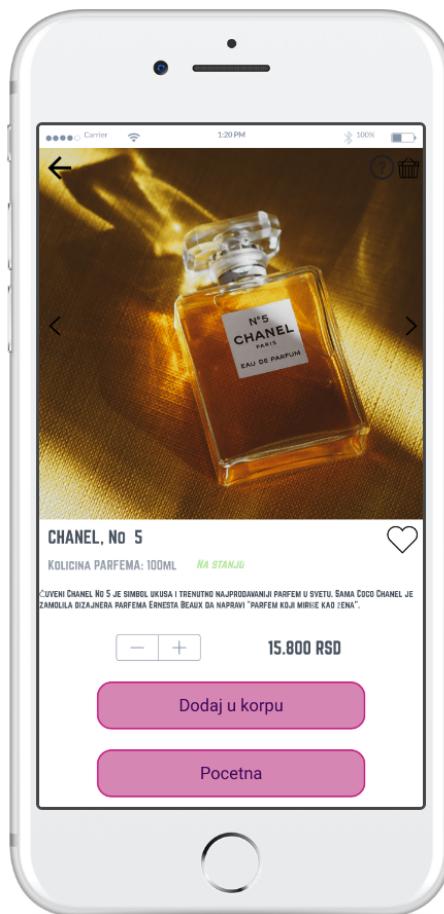
*Prikaz ekrana 3 - 5*



Slika 7.2.9 Ekran 3 [Izvor: Autor]



Slika 7.2.10 Ekran 4 [Izvor: Autor]



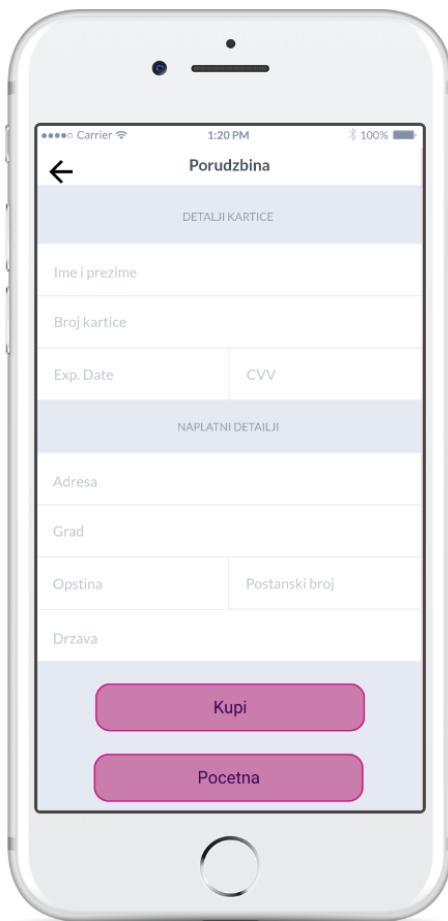
Slika 7.2.11 Ekran 5 [Izvor: Autor]

## IZGLED PREOSTALIH EKRANA

*Prikaz ekrana 6- 7*



Slika 7.2.12 Ekran 6 [Izvor: Autor]



Slika 7.2.13 Ekran 7 [Izvor: Autor]

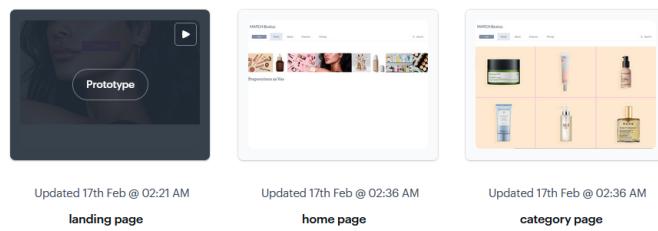
## INTERAKTIVNOST PROTOTIPA

*Potrebno je da označimo mišem deo ekrana kom dodajemo interaktivnost i ciljni ekran*

Nakon što smo završili sa dizajnom prototipa, potrebno je još da mu dodamo interaktivnost.

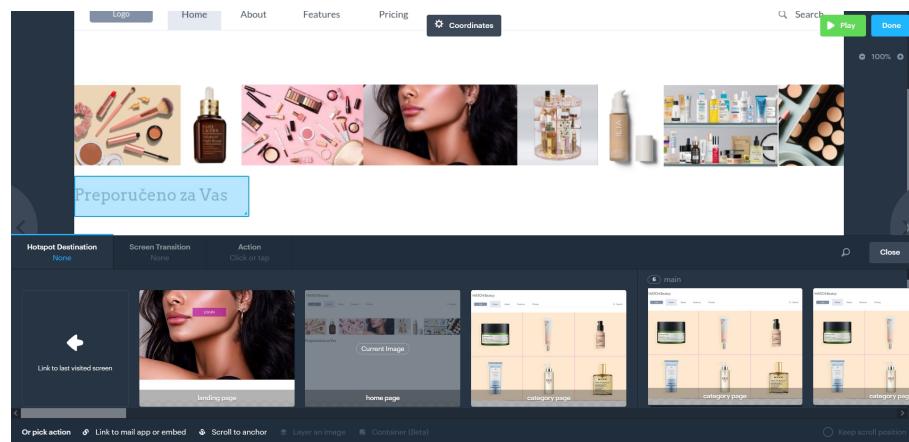
Sa stranice sa dizajnima za naš prototip potrebno je da na svakom ekranu kliknemo na "Prototype" (slika 14).

Potom potrebno je da označimo mišem šta je "dugme", tj. deo ekrana kom dodajemo interaktivnost i gde očekujemo da korisnik klikne. Na primer ekran koji prikazuje proizvode po kategoriji potrebno je da kliknemo na naziv kategorije. Nakon što označimo površinu, potrebno je da izaberemo ekran na koji želimo da predemo ti klikom (slika 15).

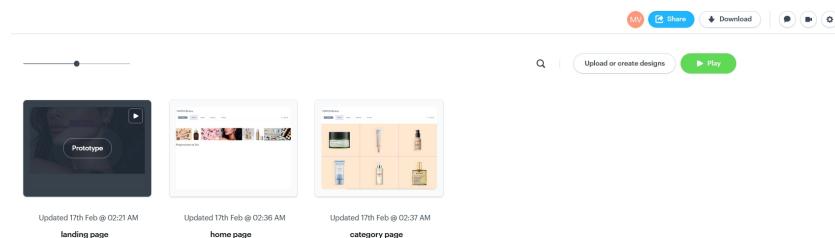


Slika 7.2.14 Stranica sa dizajnjima [Izvor: Autor]

Prateći scenario upotrebe, potrebno je da isto to uradimo i za preostale ekrane.



Slika 7.2.15 Definisanje interaktivnosti [Izvor: Autor]



Slika 7.2.16 Deljenje i korišćenje prototipa [Izvor: Autor]

Na kraju, odabirom opcije "Play" možemo sami da isprobamo naš prototip. Odabirom opcije "Share" možemo naš prototip da podelimo sa drugima.

## ▼ Poglavlje 8

# Zadatak za samostalni rad: HTTP zahtev i odgovor

## HTTP ZAHTEVI I ODGOVORI

### *Struktura zahteva i odgovora*

Predviđeno vreme pokazne vežbe je: 30 minuta.

HTTP poruke su način na koji se podaci razmenjuju između servera i klijenta. Postoje dve vrste poruka: zahtevi koje klijent šalje da pokrene akciju na serveru i odgovori - odgovor sa servera.

HTTP poruke se sastoje od tekstualnih informacija kodiranih u ASCII-u i obuhvataju više redova. U HTTP/1.1 i ranijim verzijama protokola, ove poruke su se otvoreno slale preko veze. U HTTP/2, nekada čitljiva poruka je sada podeljena na HTTP okvire, pružajući optimizaciju i poboljšanje performansi.

Veb programeri retko sami kreiraju ove tekstualne HTTP poruke: softver, veb pretraživač, proksi ili veb server, obavljaju ovu radnju. Oni obezbeđuju HTTP poruke preko konfiguracionih datoteka (za proksi servere ili servere), API-ja (za pretraživače) ili drugih interfejsa.

HTTP zahtevi i odgovori imaju sličnu strukturu i sastoje se od:

- Početni red koji opisuje zahteve koje treba implementirati, ili status uspešnog ili neuspešnog. Ova početna linija je uvek jedna linija.
- Opcioni skup HTTP zaglavlja koji specificira zahtev ili opisuje telo uključeno u poruku.
- Prazan red koji označava sve meta-informacije za zahtev je poslat.
- Opciono telo koje sadrži podatke povezane sa zahtevom (kao što je sadržaj HTML obrasca) ili dokument povezan sa odgovorom. Prisustvo tela i njegova veličina određuju startna linija i HTTP zaglavlja.

Početna linija i HTTP zaglavlja HTTP poruke su zajedno poznati kao glava (eng. *head*) zahteva, dok je *payload* poznat kao telo.

## PRIPREMA ZADATKA

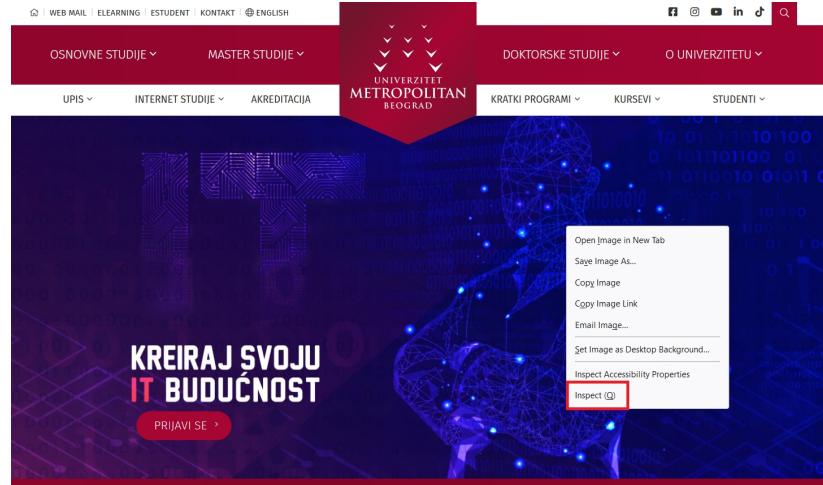
### *Slanje zahteva i dobijanje odgovora*

Otvorite veb čitač. Unesite sledeću veb adresu: <https://www.metropolitan.ac.rs/>

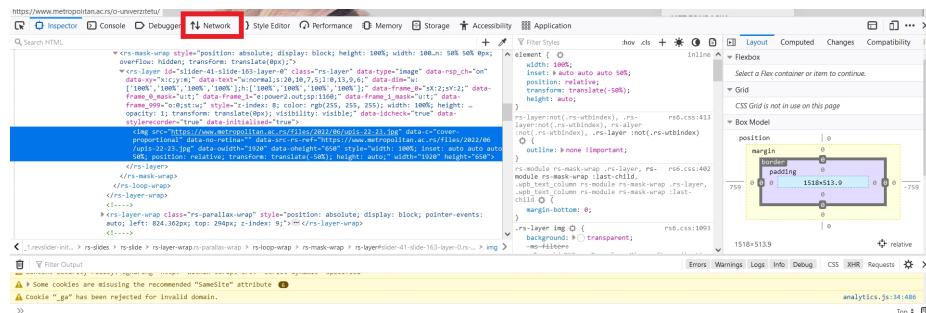
Veb čitač korišćen za pripremu vežbi je Mozilla Firefox. Ukoliko koristite drugi veb čitač dalji koraci mogu da se razlikuju.

Iz kontekstnog menija (desni klik) biramo opciju *Inspect* (slika 1). Potom biramo opciju *Network* (slika 2).

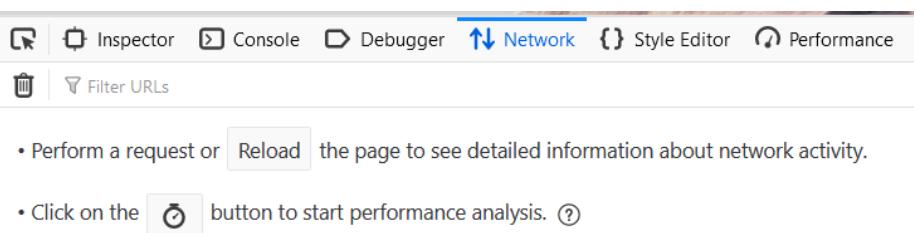
Da bismo prikupili informacije o aktivnosti na mreži potrebno je da ponovo učitamo stranicu odabirom opcije *Reload* (slika 3).



Slika 8.1 Korak 1 [Izvor: Autor]



Slika 8.2 Korak 2 [Izvor: Autor]



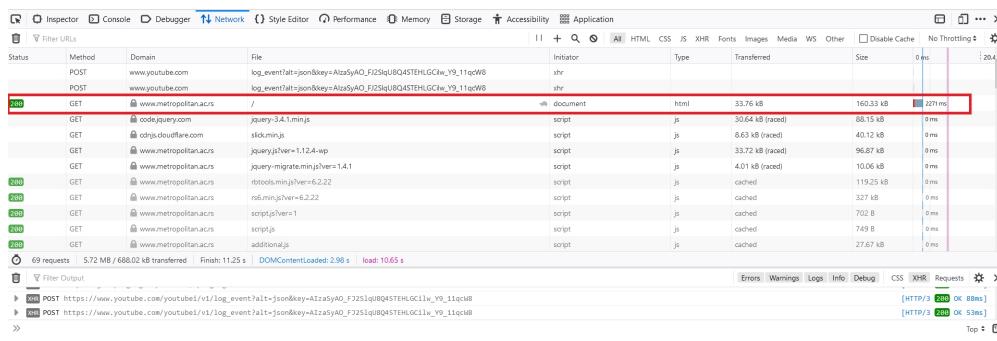
Slika 8.3 Korak 3 [Izvor: Autor]

Nakon ponovnog učitavanja stranice beleže se zahtevi i odgovori.

# PREGLED ZAHTEVA I ODGOVORA

## Odabir zahteva za analizu

Odabraćemo GET zahtev koji kao odgovor vraća HTML dokument kao što je prikazano na slici. Prelaskom mišem preko file dela vidimo da je u pitanju veb adresa <https://www.metropolitan.ac.rs/>. Klikom na navedeni red dobijamo detaljnije informacije, tj. glavu i telo zahteva i odgovora.



Slika 8.4 Pregled zahteva i odgovora [Izvor: Autor]

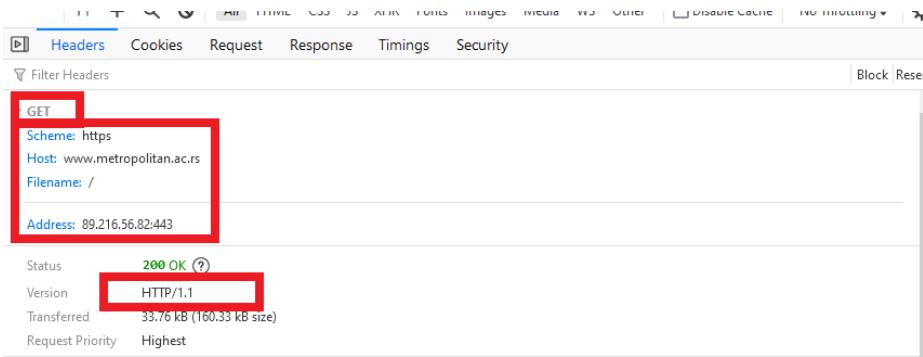
# ANALIZA REZULTATA

## Analiza početne linije

HTTP zahtevi su poruke koje klijent šalje da bi pokrenuo akciju na serveru. Njihova početna linija sadrži tri elementa:

- **HTTP metod**, glagol (poput GET, PUT ili POST) ili imenica (kao HEAD ili OPTIONS), koji opisuje radnju koju treba izvršiti. Na primer, GET označava da resurs treba da se preuzme ili POST znači da se podaci prosleđuju serveru (kreiranje ili modifikovanje resursa, ili generisanje privremenog dokumenta za slanje nazad).
  - U našem primeru metod je GET.
- **Cilj zahteva**, obično URL ili absolutna putanja protokola, porta i domena, obično karakteriše kontekst zahteva. Format ovog cilja zahteva varira između različitih HTTP metoda. To može biti:
  - Apsolutna putanja, koju na kraju prati '?' i string upita. Ovo je najčešći oblik, poznat kao izvorni oblik, i koristi se sa metodama GET, POST, HEAD i OPTIONS. Primer: /test.html?query=alibaba
  - Potpuna URL adresa, poznata kao absolutni obrazac, uglavnom se koristi sa GET-om kada je povezana sa proksijem. U našem primeru putanja je: <https://www.metropolitan.ac.rs/>
  - Komponenta ovlašćenja URL-a, koja se sastoji od imena domena i opciono porta (sa prefiksom ':'), naziva se obrazac ovlašćenja. Koristi se samo sa CONNECT kada se podešava HTTP tunel. Primer: developer.mozilla.org:80
  - Forma zvezdice, jednostavna zvezdica ('\*') se koristi sa OPTION, predstavljajući server kao celinu.

- **HTTP verzija**, koja definiše strukturu preostale poruke, koja deluje kao indikator očekivane verzije koja će se koristiti za odgovor.



Slika 8.5 Prikaz početne linije [Izvor: Autor]

Napomena: Na slici 5. prikazana je i IP adresa servera. IP adresa je jedinstvena adresa koja identificuje uređaj na internetu ili lokalnoj mreži. IP adresa je niz brojeva razdvojenih tačkama.

## ANALIZA ZAHTEVA

### *Analiza zaglavlja zahteva u zadatom primeru*

HTTP zaglavljia iz zahteva prate istu osnovnu strukturu HTTP zaglavljia: string neosetljiv na velika i mala slova praćen dvotačkom (':') i vrednost čija struktura zavisi od zaglavljia. Celo zaglavljje, uključujući vrednost, sastoji se od jednog reda, koji može biti prilično dugačak.

U zahtevima se može pojavit mnogo različitih zaglavljia. Mogu se podeliti u nekoliko grupa:

- Opšta zaglavljia, poput Via, primenjuju se na poruku kao celinu.
- Zaglavljia zahteva, kao što su User-Agent ili Accept, modifikuju zahtev tako što ga navedu dalje (kao Accept-Language), dajući kontekst (kao Referer) ili ga uslovno ograničavajući (kao If-None).
- Zaglavljia predstavljanja kao što je Content-Type koja opisuju originalni format podataka poruke i bilo koje primenjeno kodiranje (prisutno samo ako poruka ima telo).

Završni deo zahteva je njegovo telo. Nemaju svi zahtevi telo: zahtevima za preuzimanje resursa, kao što su GET, HEAD, DELETE ili OPTIONS, obično ne trebaju. Neki zahtevi šalju podatke serveru kako bi ih ažurirali: kao što je čest slučaj sa POST zahtevima (koji sadrže podatke HTML obrasca).

Request Headers (779 B)

- ⑦ **Accept:** text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,\*/\*;q=0.8
- ⑦ **Accept-Encoding:** gzip, deflate, br
- ⑦ **Accept-Language:** en-US,en;q=0.5
- ⑦ **Connection:** keep-alive
- ⑦ **Cookie:** \_ga=GA1.3.502811408.1656433890; \_fbp=fb.2.1656433890407.1196437135; prihvatan=; SMFCookie638=a%3A4%3A%7Bi%3A0%3Bs%3A4%3A%22335%22%3B%3A1%3Bs%3A40%3A%223ce83d9bd7c439e0f06c1dc4cb32d305fb24b93d%22%3Bi%3A2%3Bi%3A1694011754%3Bi%3A3%3Bi%3A2%3B%7D; PHPSESSID=jm3lfkueah7md09oq1j9do9li4; \_gid=GA1.3.1761484521.1667689491
- ⑦ **Host:** www.metropolitan.ac.rs
- ⑦ **Sec-Fetch-Dest:** document
- ⑦ **Sec-Fetch-Mode:** navigate
- ⑦ **Sec-Fetch-Site:** none
- ⑦ **Sec-Fetch-User:** ?1
- ⑦ **Upgrade-Insecure-Requests:** 1
- ⑦ **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:106.0) Gecko/20100101 Firefox/106.0

Slika 8.6 Prikaz HTTP zahteva [Izvor: Autor]

U našem primeru:

- **Accept** pokazuje koje tipove sadržaja, izražene kao medija tipove, klijent može da razume. Jasno je da očekujemo veb stranicu kao odgovor u vidu tekstualnog fajla, html, xhtml, xml fajlova.
- **Accept-Encoding** ukazuje na kodiranje sadržaja (obično algoritam kompresije) koje klijent može da razume.
- **Accept-Language** ukazuje na prirodni jezik i lokalizaciju koje klijent preferira. U našem slušaju to je engleski jezik (SAD). **Connection:** kontroliše da li mrežna veza ostaje otvorena nakon što se trenutna transakcija završi. Ako je poslata vrednost održavana, veza je trajna i nije zatvorena, što omogućava da se obave naredni zahtevi istom serveru.

## ZADATAK 1.

### *Analizirajte preostala zaglavila zahteva*

Predviđeno vreme za izradu zadatka je 10 minuta.

#### **Zadatak 1.**

Na slici 6. prikazana su zaglavila HTTP zahteva. Neka od tih zaglavila smo sada analizirali. Istražite šta predstavljaju preostala zaglavila. Na našem primeru, šta podrazumevaju vrednosti tih zaglavila?

Za rešavanje zadatka možete koristiti sledeći link: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

## ANALIZA ODGOVORA

### *Analiza odgovora u zadatom primeru*

Početna linija HTTP odgovora, koja se zove statusna linija, sadrži sledeće informacije:

- Verzija protokola, u našem primeru to je HTTP/1.1.

- Statusni kod, koji ukazuje na uspeh ili neuspeh zahteva. Uobičajeni statusni kodovi su 200, 404 ili 302. U našem primeru to je 202 OK., što znači da je zahtev uspešan i da, pošto je u pitanju GET metoda, resurs je preuzet i poslat u telu poruke. Značenje ostalih statusnih kodova možete videti na sledećem linku: <https://developer.mozilla.org/en-US/docs/Web/HTTP>Status>
- Tekst statusa. Kratak, čisto informativni, tekstualni opis statusnog koda koji pomaže čoveku da razume HTTP poruku.

HTTP zaglavla odgovora prate istu osnovnu strukturu HTTP zaglavla: string neosetljiv na velika i mala slova praćen dvotačkom (':') i vrednost čija struktura zavisi od zaglavla.

Celo zaglavje, uključujući njegovu vrednost, predstavlja se kao jedan red. U odgovorima se može pojaviti mnogo različitih zaglavla. Mogu se podeliti u nekoliko grupa:

- Opšta zaglavla, poput Via, primenjuju se na poruku kao celinu.
- Zaglavla odgovora, kao što su Vary ili Accept-Ranges, daju dodatne informacije o serveru (koje se ne uklapaju u statusnu liniju).
- Zaglavla predstavljanja kao što je Content-Type koja opisuju originalni format podataka poruke i bilo koje primenjeno kodiranje (prisutno samo ako poruka ima telo).

Status	200 OK <a href="#">?</a>
Version	HTTP/1.1
Transferred	33.76 kB (160.33 kB size)
Request Priority	Highest
<b>▼ Response Headers (549 B)</b>	
<a href="#">? Access-Control-Allow-Origin: *</a>	
<a href="#">? Cache-Control: no-store, no-cache, must-revalidate</a>	
<a href="#">? Connection: Keep-Alive</a>	
<a href="#">? Content-Encoding: gzip</a>	
<a href="#">? Content-Length: 33206</a>	
<a href="#">? Content-Type: text/html; charset=UTF-8</a>	
<a href="#">? Date: Sun, 06 Nov 2022 00:24:04 GMT</a>	
<a href="#">? Expires: Thu, 19 Nov 1981 08:52:00 GMT</a>	
<a href="#">? Keep-Alive: timeout=100, max=100</a>	
<a href="#">Link: &lt;https://www.metropolitan.ac.rs/wp-json/&gt;; rel="https://api.w.org/"</a>	
<a href="#">Link: &lt;https://www.metropolitan.ac.rs/&gt;; rel=shortlink</a>	
<a href="#">? Pragma: no-cache</a>	
<a href="#">? Server: Apache/2.2.26 (CentOS)</a>	
<a href="#">? Vary: Accept-Encoding</a>	
<a href="#">X-Powered-By: PHP/7.0.33</a>	
<b>▼ Request Headers (779 B)</b>	

Slika 8.7 Prikaz HTTP odgovora [Izvor: Autor]

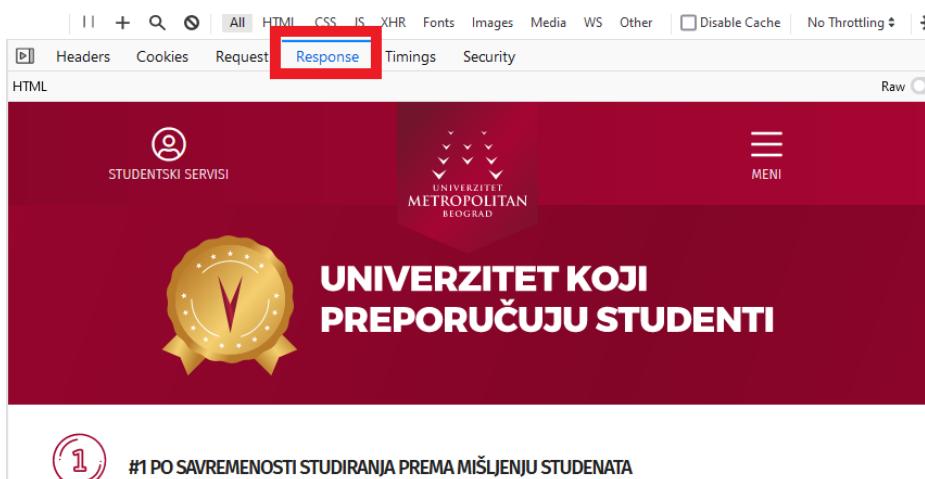
## ANALIZA ZAGLAVLJA I TELA ODGOVORA

### *Analiza konkretnih zaglavlja odgovora*

U našem primeru:

- **Content-Encoding** navodi sva kodiranja koja su primenjena na telo/teret poruke (payload) i kojim redosledom. Ovo omogućava primaocu da zna kako da dekodira telo poruke da bi dobio originalni format tela poruke. Kodiranje sadržaja se uglavnom koristi za komprimovanje podataka poruke bez gubitka informacija o izvornom tipu medija.
- **Content-Length** označava veličinu tela poruke, u bajtovima, koja se šalje primaocu.
- **Content-Type** se koristi za označavanje originalnog tipa medija resursa (pre bilo kakvog kodiranja sadržaja primjenjenog za slanje).
- **Server** opisuje softver koji koristi izvorni server koji je obradio zahtev — odnosno server koji je generisao odgovor.

Poslednji deo odgovora je telo. Nemaju svi odgovori jedan. Da bismo videli telo odgovora, u našem primeru, potrebno je da kliknemo na Response (slika 8).



Slika 8.8 Prikaz tela odgovora [Izvor: Autor]

## ZADATAK 2.

### *Analizirajte preostala zaglavљa odgovora*

Predviđeno vreme za izradu zadatka je 10 minuta.

#### **Zadatak 2.**

Na slici 7. prikazana su zaglavljia HTTP odgovora. Neka od tih zaglavljia smo sada analizirali. Istražite šta predstavljaju preostala zaglavljia. Na našem primeru, šta podrazumevaju vrednosti tih zaglavljia?

Za rešavanje zadatka možete koristiti sledeći link: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

## ✓ Poglavlje 9

### Domaći zadatak

#### OPIS DOMAĆEG ZADATKA - DZ01

*Za proizvoljni e-commerce sajt izradite prototip i testirajte ga*

Očekivano vreme izrade zadatka: 40min.

Za dodeljeni veb sajt dajte predlog kako bi isplanirali **responzivni dizajn** za dodeljeni sajt u **minimalno 200 reči** (idejni plan) i kreirajte izgled **početne stranice** za desktop i mobilni uređaj. Za dodeljeni veb sajt sami smislite tematiku, tj. šta ćete prikazati na početnoj stranici. Izgled početne stranice možete kreirati u Marvelu ili bilo kom drugom alatu. Slike/snimke ekrana dodajte u word dokument. Napomena: na sledećem linku možete videte šta se podrazumeva pod početnom stranom [www.mediaqueri.es](http://www.mediaqueri.es)

Zadatak dostaviti kao document pod nazivom **CS105-DZ01-Ime\_Prezime\_BrojIndeksa.docx**. Domaći zadatak pošaljite predmetnom asistentu na e-mail. Subject mejla mora biti **CS105-DZ01**.

Veb sajt koji dobijate određujete po sledećoj formuli: **Broj indeksa % 12 + 1** (Npr., 2378 % 15 + 1 = 3 – Student dobija veb sajt pod brojem 9).

1. Onlajn prodavnica (sami izaberite temu prodavnice)
2. Portfolio vefsajt (muzičari, umetnici, dizajneri...)
3. Blog (sami izaberite tematiku blog-a)
4. Veb sajt startup-a (predstavljanje kompanije i njihovog proizvoda)
5. Korporativni veb sajt (vebsajtovi poput Adobe, VMWare, PWC, Oracle...)
6. Vefsajt škole (škole, kursevi, radionice...)
7. Vefsajt namenjen događaju (koncert, sajam, konferencija...)
8. Vefsajt kluba/zajednice (sami izaberite temu kluba)
9. Veb sajt za pronalaženje posla
10. Veb sajt neprofite organizacije
11. Edukativni veb sajt (tema po izboru)
12. Informativni veb sajt (poput BBC, Buzzfeed, Motorsport, ESPN...)

## ✓ Poglavlje 10

### Zaključak

## ZAKLJUČAK

Na ovom predavanju bilo je reči o programiranju serverske i klijentske strane. Ono što je najosnovnije da treba shvatiti na ovom predavanju jesu prednosti i mane korišćenja skripting jezika sa serverske i klijentske strane. Potrebno je da svaki IT profesionalac zna gde je najefikasnije da se obave određene funkcionalnosti kako bi sistemi bili što efikasniji i kako se ne bi ugrozila njihova bezbednost. U predavanju je dat i pregled najčešće korišćenih skripting jezika, kao i njihovi primeri koje je potrebno znati razumeti.

### Literatura

1. G. Timothy, J. O'Leary, Linda I. Computing Essentials, complete edition. O'Leary ISBN 0-07-226110-2, Publisher: McGraw-Hill (2021).
2. B. Shelly, Misty E. Vermaat, Discovering Computers 2018-Introductory: Living in a Digital World, Cengage Learning 2018.