

Računarske mreže i serversko vob-programiranje

Jovana Brankov 108/2022

26.01.2026.

Sadržaj

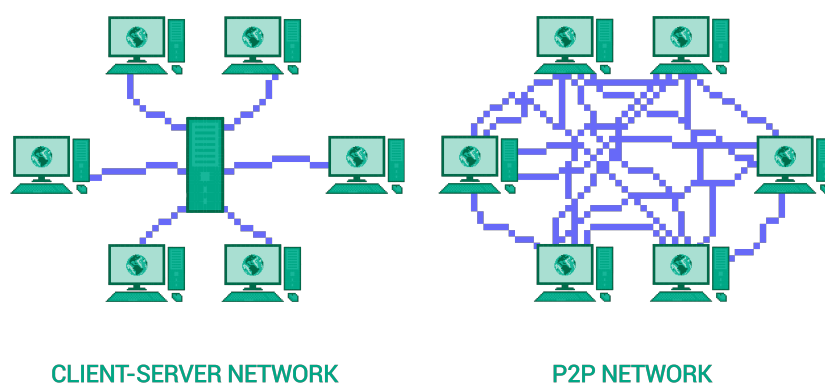
1	Računarske mreže	3
1.1	Računarske mreže i internet	3
1.2	Mrežni uređaji	3
1.3	Adrese	6
1.4	Mrežni slojevi i protokoli	7
1.5	Statičke veb-strane HTML/CSS	9
1.6	Dizajn veb-strane	11
1.7	Kreiranje veb-sajta	12
2	Serversko veb-programiranje	13
2.1	Veb-stranice	13
2.2	HTTP protokol	14
3	Flask biblioteka	16
3.1	Osnovni pojmovi	16
3.2	Prva Flask veb-aplikacija	20
3.3	Šabloni	23
3.4	Statičke datoteke i linkovi	26
4	Literatura	29

1. Računarske mreže

Računarska mreža je sistem koji se sastoji od skupa hardverskih uređaja međusobno povezanih komunikacionom opremom, snabdevenih odgovarajućim komunikacionim softverom, kojim se ostvaruje kontrola sistema tako da je omogućen prenos podataka između povezanih uređaja.

1.1. Računarske mreže i internet

Internet je globalna mreža koja povezuje milione manjih mreža. Mreže se najčešće organizuju u model **mreže ravnopravnih računata** ili model **klijent-server**.



Slika 1: Modeli računarskih mreža

U klijent-server modelu server je sistem koji pruža svoje resurse (podatke, softver, hardver), dok je klijent sistem koji inicira kontakt sa serverom da bi koristio resurse koje server pruža. Većina internet servisa uglavnom koristi model klijent-server. Na primer, digitalni uređaj na kojem pregledate sadržaj veba je klijent. Kada unesete adresu u pregledač, vaš uređaj se obraća nekom veb-serveru – udaljenom računaru na kome se nalaze veb-strane koje zahtevate (ili programi koji ih generišu). Server klijentu šalje dokumente (veb-stranice, slike itd.) i klijent ih prikazuje korisniku.

1.2. Mrežni uređaji

Da bi računar ili neki drugi digitalni uređaj mogao da učestvuje u računarskoj mreži i da se poveže na internet, neophodno je da poseduje mrežnu karticu, odnosno mrežni kontroler. Mrežni kontroleri mogu biti namenjeni za žično ili bežično povezivanje. Kod žičnog povezivanja najčešće se koristi UTP kabl sa RJ45 priključkom, dok se za bežično povezivanje koriste kontroleri zasnovani na radio-talasma. Većina stonih računara ima ugrađenu mrežnu karticu, dok prenosni računari, pored žične, poseduju i karticu za bežično povezivanje. Mobilni telefoni i tableti uglavnom koriste isključivo bežičnu mrežu, dok savremeni televizori često podržavaju obe vrste povezivanja.

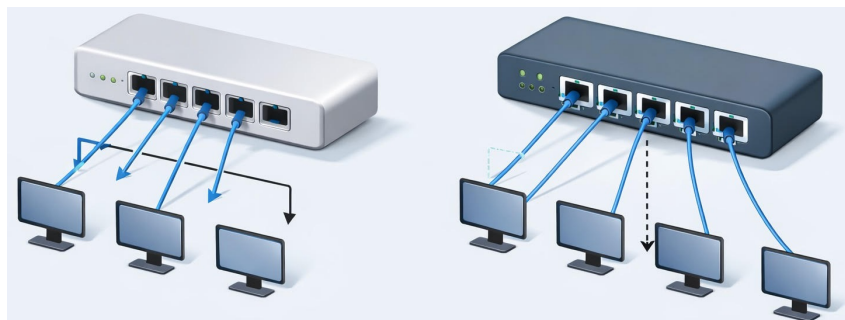


Slika 2: UTP kablovi sa konektorima RJ45, mrežna kartica za povezivanje UTP kablom, mrežna kartica za bežično povezivanje (sa antenom)

U računarskim mrežama važnu ulogu imaju i uređaji koji posreduju u komunikaciji.

Pojačivači (repetitori) služe za jačanje signala kako bi se omogućila komunikacija na većim udaljenostima. **Habovi** prosleđuju sve primljene podatke svim povezanim uređajima, zbog čega su danas gotovo u potpunosti zamenjeni savremenijim uređajima.

Svičevi povezuju više uređaja i podatke prosleđuju samo onom uređaju kome su namenjeni, čime se smanjuje opterećenje mreže i povećava efikasnost. Oni rade na nižim mrežnim slojevima i koriste fizičke (MAC) adrese.



Slika 3: Hab i svič

Ruteri su složeniji uređaji koji povezuju različite mreže i omogućavaju pristup internetu. Oni rade na mrežnom sloju, analiziraju IP adrese i određuju najbolji put kojim će podaci stići do odredišta. Zbog toga su veoma rasprostranjeni u domaćinstvima i malim mrežama.



Slika 4: Ruter

Pored mrežnih uređaja, važan deo mreže čine i komunikacioni kanali, odnosno medijumi za prenos podataka. Najčešće se koriste **upredene parice** (UTP/STP kablovi), **koaksijalni kablovi** i **optički kablovi**. UTP kablovi su najrasprostranjeniji u lokalnim mrežama, dok se optički kablovi koriste za prenos velikih količina podataka na velikim brzinama i na većim udaljenostima, naročito u okosnici interneta, a koaksijalni kablovi obično se koriste za prenos radio i televizijskog signala.

Kod bežične komunikacije koriste se radio-talasi, mikrotalasi ili infracrveni zraci. **Bluetooth** tehnologija namenjena je komunikaciji na malim udaljenostima, dok se **WiFi** mreže koriste za povezivanje uređaja u lokalnim mrežama. Za pokrivanje većih područja koriste se naprednije bežične tehnologije, kao i **satelitske veze** i **mobilne mreže**, koje omogućavaju komunikaciju na velikim razdaljinama.

1.3. Adrese

Postoji nekoliko različitih sistema adresa koji se koriste u mrežnoj komunikaciji. U današnjim mrežama se obično koriste:

- MAC adrese
- IP adrese
- DNS adrese i URL-ovi

Fizičke (MAC) adrese. MAC adresa (*Media Access Control*) je jedinstveni, trajni fizički identifikator mrežnog uređaja (kartice) koji dodeljuje proizvođač, zapisan u ROM memoriji. Obično su 48-bitne ili 64-bitne, prikazuju se kao 12 heksadecimalnih znakova (npr. 00:1B:44:11:3A:B7). Koristi se za komunikaciju na lokalnoj mreži (LAN).

IP adrese. IP adresa (*Internet Protocol Address*) je jedinstveni numerički identifikator (npr. 192.168.1.1) koji omogućava uređajima (računarima, telefonima) da komuniciraju na mreži, poput digitalne poštanske adrese koja usmerava podatke, a dele se na javne (vidljive na internetu) i privatne (za lokalnu mrežu), sa dve glavne verzije: stariji IPv4 (četiri broja) i noviji IPv6 (duži niz, 32 cifre).

Adrese mogu biti dodeljene statički i dinamički. **Statičke** dodeljene adrese su takve da uređaj ima fiksnu IP adresu kada god se priključi na internet. Primer: adresa servera nikada se ne menja kako bi klijenti mogli da mu uvek pristupaju na isti način, štampač koji se koristi u poslovnom prostoru i mora imati statičku IP adresu tako da svi u kancelariji mogu lako da se povezuju sa njim. **Dinamički** dodeljene adrese uređajima su adrese koje se uređaju dodeljuje neka slobodna IP adresa svaki put kada se priključuje na internet. Prednosti dinamičkih adresa su bezbednost i manja mogućnost da dodje do greške. Na primer, jednom laptopu koji se poveže na internet može se dodeliti određena IP adresa, a kada se odvoji, ta adresa se onda može slobodno koristiti za drugi uređaj koji se kasnije povezuje.

Domeni i URL. Krajnji korisnik najčešće ima dodira sa URL adresama (engl. *Uniform Resource Locator*). Nekada se javlja i pojam URI (engl. *Uniform Resource Identifier*). URL je specifična vrsta URI-ja koja pored identifikacije resursa daje i putanju kako do njega doći, dok URI generalno samo identifikuje resurs bez nužnog načina pristupa. U praksi URL-ovi su veb-adrese. Na primer: `http://info.cern.ch/hypertext/WWW/TheProject.html`, `https://www.wikipedia.org/`, `http://www.matf.bg.ac.rs/`. Šta sve sadrži prvi URL kod?

- **http** -Oznaka protokola koji se koristi za pristup resursu

- `info.cern.ch` -Naziv servera na kom se resurs nalazi
- `hypertext/WWW/TheProject.html` -Putanja do resursa unutar tog servera

Važan deo URL-a je adresa servera, to su na primer: `www.uns.ac.rs`, `raf.edu.rs` i `info.cern.ch`. Njih nazivamo **domenima**, oni se koriste u sklopu URL-ova i adresa elektronskih pošti (`petar.petrovic@uns.ac.rs`). Domeni su hijerarhijski organizovani, na primer: `matf.bg.ac.rs`

- `rs`→ nalazi se u Srbiji
- `ac.rs`→ deo je akademske mreže
- `bg.ac.rs`→ nalazi se na univerzitetu u Beogradu
- `matf.bg.ac.rs`→ nalazi se na Matematičkom fakultetu

Domeni nisu uvek vezani za zemlju, na primer:

- `.com` je komercijalni domen koji može svako da zakupi
- `.org` uglavnom koriste neprofitne organizacije
- `.edu` koriste obrazovne institucije
- `.aero` koriste avioprevoznici i aerodromi

Pošto su za interno funkcionisanje mrežne komunikacije neophodne numeričke IP adrese, a ne tekstualne adrese, svakom imenu domena pridružena je IP adresa odgovarajućeg uređaja registrovanog za taj domen (na primer, imenu servera `www.pmf.uns.ac.rs` pridružena je IP adresa 147.91.177.44). Ovo pridruživanje beleži se na posebnim serverima, koji se nazivaju serveri za imena domena (engl. *Domain Name Server*, DNS). **DNS** se često naziva telefonskim imenikom interneta. Pre započinjanja mrežne komunikacije, softver koji podržava imena domena (na primer, pregledač u koji korisnik unosi veb-adresu) obraća se DNS serveru i od njega traži IP adresu na osnovu imena domena koje mu pošalje.

1.4. Mrežni slojevi i protokoli

Računarske mreže su složeni sistemi jer uključuju veliki broj hardverskih i softverskih komponenti. Da bi se ta složenost pojednostavila, mreže su organizovane po principu slojevitosti, gde svaki sloj ima jasno definisanu ulogu i koristi određene protokole komunikacije. Protokoli predstavljaju skup pravila koja omogućavaju razmenu podataka između uređaja u mreži.

Internet je organizovan prema TCP/IP modelu koji se sastoji od četiri osnovna sloja: veznog (sloj mrežnog interfejsa), mrežnog (internet sloj), transportnog i aplikativnog. **Ve-
zni sloj** se bavi fizičkim prenosom podataka i pouzdanošću komunikacije između uređaja. Protokoli na ovom sloju su:

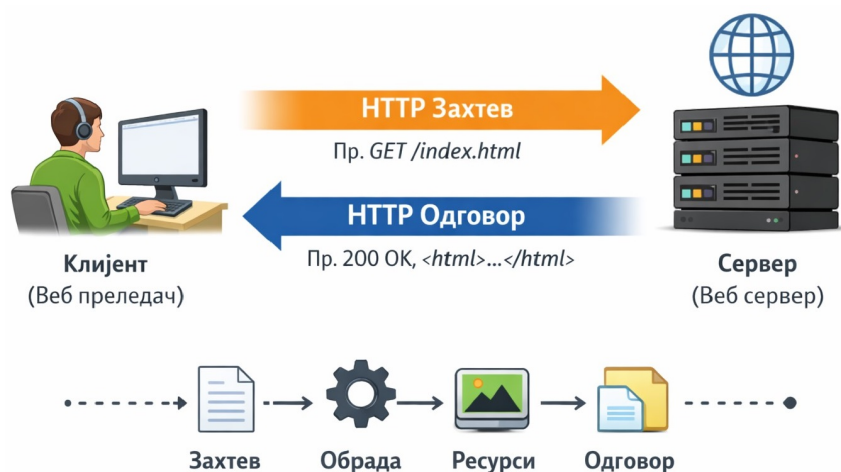
- protokol IP (engl. *Internet Protocol*) - za slanje poruka
- protokol ICMP (engl. *Internet Control Message Protocol*) - za slanje kontrolnih poruka
- protokol ARP (engl. *Address Resolution Protocol*) - za razrešenje adresa
- ARP – prevođenje IP adrese u MAC adresu
- RARP – prevođenje MAC adrese u IP adresu

Mrežni sloj omogućava adresiranje i rutiranje podataka kroz širu mrežu, odnosno određivanje putanje kojom paketi stižu do odredišta.

Transportni sloj obezbeđuje prenos podataka između aplikacija i koristi **portove** za njihovo razlikovanje. Najvažniji protokoli ovog sloja su TCP i UDP. **TCP** omogućava pouzdanu komunikaciju uz kontrolu grešaka, dok je **UDP** brži, ali manje pouzdan i koristi se u aplikacijama u realnom vremenu. **Aplikativni sloj** obuhvata protokole koji su namenjeni konkretnim internet servisima, neki od njih su:

- HTTP i HTTPS (za preuzimanje sa veb)
- POP3 i IMAP (za preuzimanje elektronske pošte)
- SMTP (za slanje elektronske pošte)
- ETP (za prenos fajlova)

Veb funkcioniše po modelu **klijent-server**, gde klijent šalje zahteve, a server na njih odgovara. Komunikacija se odvija putem HTTP ili HTTPS protokola, pri čemu HTTPS obezbeđuje veću sigurnost šifrovanjem podataka. Ovaj način rada predstavlja osnovu savremenog interneta i veb-aplikacija.



Slika 5: Klijent-server

1.5. Statičke veb-strane HTML/CSS

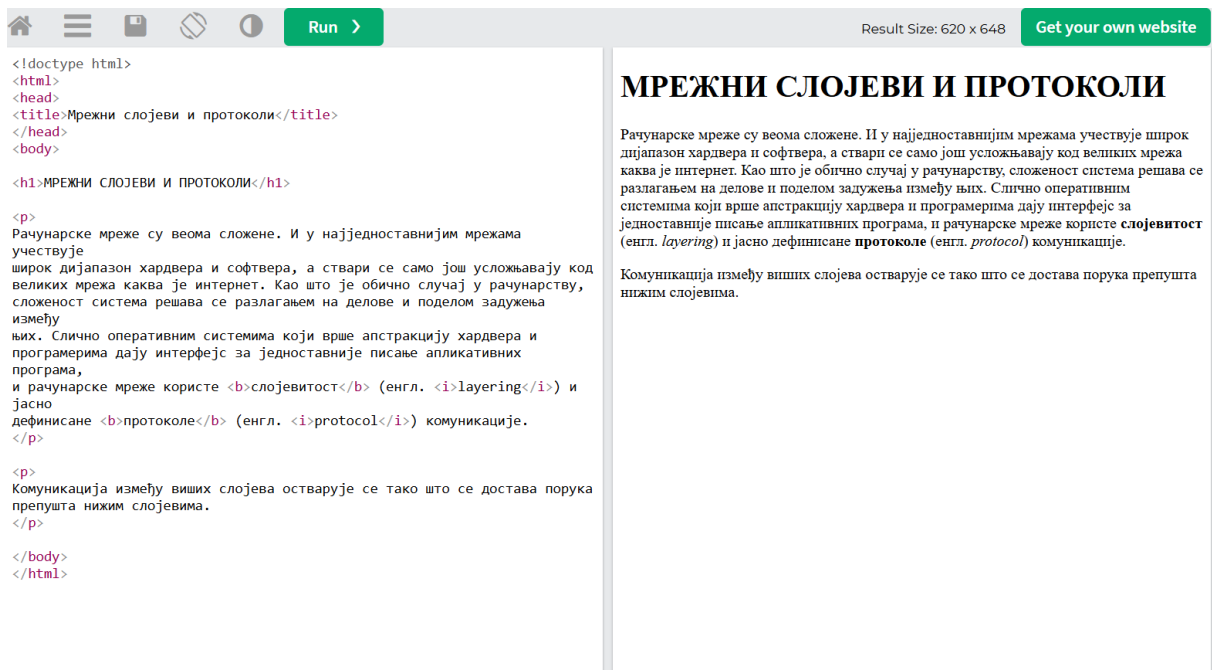
Na početku razvoja veba, većina veb-strana je bila **statička**, što znači da je njihov sadržaj bio unapred pripremljen i isti za sve korisnike. Takve stranice su služile isključivo za pregled informacija, bez mogućnosti značajnije interakcije. Uloge klijenta i servera u ovom slučaju su jednostavne: klijent šalje zahtev, a server pronalazi odgovarajuću HTML datoteku i šalje je nazad klijentu, bez ikakve dodatne obrade. Za izradu statičkih veb-strana koriste se osnovne veb-tehnologije – HTML i CSS. **HTML** je jezik za obeležavanje koji definiše strukturu i sadržaj veb-strane, dok se **CSS** koristi za njen vizuelni izgled. HTML dokumenti su tekstualne datoteke koje mogu da se pišu u bilo kom editoru teksta i koje svaki veb-pregledač može da prikaže.

HTML stranica se sastoji od elemenata koji su označeni tagovima. Tagovi se najčešće javljaju u parovima (otvoreni i zatvoreni tag), a zajedno sa sadržajem koji okružuju čine HTML element. Pomoću ovih elemenata mogu se definisati naslovi, pasusi teksta, liste, slike i drugi delovi stranice.

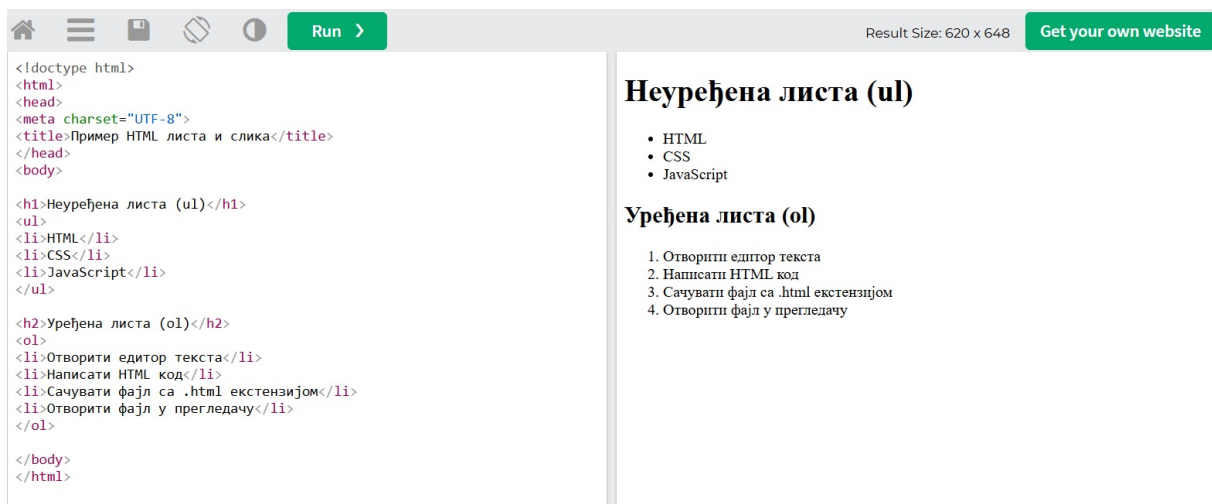


Slika 6: Osnovni tagovi

Za oblikovanje teksta koriste se različite oznake, kao što su tagovi za naslove različitih nivoa, pasuse, kao i tagovi za podebljan, iskošen ili podvučen tekst. Liste mogu biti uređene ili neuređene, u zavisnosti od načina prikaza njihovih elemenata.



Slika 7: Oblikovanje teksta



Slika 8: Liste

Slike su, u najvećem broju slučajeva, fajlovi koji se čuvaju odvojeno od HTML dokumenata (najčešće u nekom poddirektorijumu). Slike se postavljaju pomoću elementa ``, koji u svom atributu `src` sadrži ime slike koju treba prikazati.

``

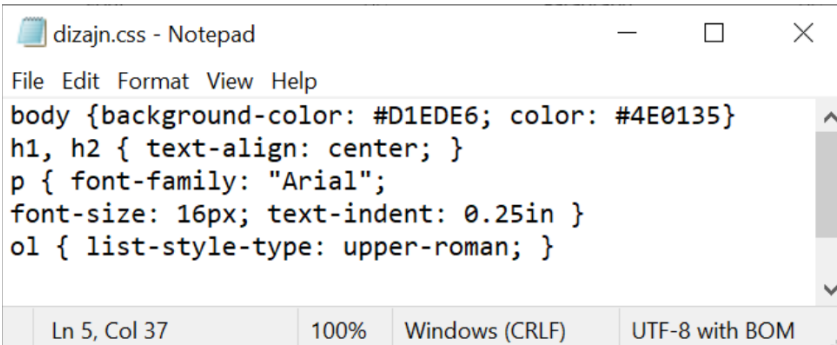
Nakon svake izmene HTML dokumenta, fajl je potrebno sačuvati i ponovo učitati u veb-pregledaču kako bi promene bile vidljive. Na ovaj način se kreiraju jednostavne statičke veb-strane koje predstavljaju osnovu daljeg razvoja veb-aplikacija.

1.6. Dizajn veb-strane

Prilikom izrade veb-stranica, HTML elementi se bez dodatnog oblikovanja prikazuju u pregledaču jedan ispod drugog, koristeći podrazumevane stilove kao što su boje, veličina teksta i razmak između elemenata. Međutim, da bi veb-stranica bila pregledna i funkcionalna, neophodno je prilagoditi izgled i raspored elemenata njenoj nameni.

Veb dizajn predstavlja proces oblikovanja vizuelnog izgleda veb-stranice i načina na koji su njeni elementi raspoređeni. On je tesno povezan sa organizacijom sadržaja i korisničkim iskustvom, koje ima za cilj da korišćenje sajta bude jednostavno i intuitivno za korisnike.

HTML jezik služi za definisanje strukture i sadržaja veb-stranice, dok se za uređivanje njenog izgleda koristi **CSS** (*Cascading Style Sheets*). Pomoću CSS-a se zadaju pravila koja određuju boje, fontove, poravnanje, razmake i druge vizuelne karakteristike elemenata na stranici. CSS pravila se zapisuju u posebnim fajlovima sa ekstenzijom `.css`, koji se mogu kreirati u bilo kom editoru teksta. U CSS dokumentu se navode pravila koja se odnose na pojedinačne HTML elemente ili na više njih istovremeno. Za svaki element mogu se definisati različita stilska svojstva, kao što su boja pozadine, boja teksta ili poravnanje. Na taj način se postiže jedinstven i usklađen izgled cele veb-stranice. Na slici ispod mozete videti jedan CSS kod koji sadrži sledeća pravila: `Background-color` zadaje boju pozadine, `color` boja teksta na stranici, `text-align: center` centrira tekst naslova, `font-family` određuje font, `font-size` određuje veličinu teksta, `text-indent` podešava uvlačenje prvog reda pasusa.



```
dizajn.css - Notepad
File Edit Format View Help
body {background-color: #D1EDE6; color: #4E0135}
h1, h2 { text-align: center; }
p { font-family: "Arial";
font-size: 16px; text-indent: 0.25in }
ol { list-style-type: upper-roman; }
```

Ln 5, Col 37 100% Windows (CRLF) UTF-8 with BOM

Da bi se stilovi primenili, potrebno je povezati CSS dokument sa HTML stranicom pomoću odgovarajuće oznake unutar dela dokumenta namenjenog za podešavanja. Nakon toga, prilikom učitavanja stranice u internet pregledaču, vidljive su promene u izgledu, kao što su drugačije boje, formatiranje teksta i način prikaza listi.

```
<!doctype html>
<html>
  <head>
    <title>Мрежни слојеви и протоколи</title>
    <link rel="stylesheet" href="dizajn.css">
  </head>
  <body>
    . . .
```

Slika 9: Povezivanje CSS dokumenta sa HTML stranicom

Na kraju, kreirana veb-stranica se proverava u internet pregledaču, a ukoliko se uoči da postoje određeni nedostaci, vrše se dodatne izmene u HTML i CSS dokumentima kako bi se postigao željeni i kvalitetan vizuelni izgled stranice.

1.7. Kreiranje veb-sajta

Veb-sajt se najčešće sastoji od više međusobno povezanih veb-stranica koje su smeštene u zajednički folder. Jedna od tih stranica ima posebnu ulogu i predstavlja početnu stranicu sajta. Radi lakšeg snalaženja korisnika, na svakoj veb-stranici je poželjno postaviti navigacione linkove koji omogućavaju prelazak na ostale stranice sajta.

```
<a href="podaci.html">Osnovni podaci</a>
<a href="više.html">Nešto više o meni</a>
<a href="zanimljivo.html"> Zanimljivo </a>
```

Slika 10: Element a omogućava da klikne na tekst

Povezivanje veb-stranica se ostvaruje pomoću HTML elementa za linkove, koji omogućava korisniku da jednostavnim klikom pređe sa jedne stranice na drugu. Na ovaj način se obezbeđuje funkcionalna i pregledna struktura veb-sajta.

Da bi sve stranice imale ujednačen izgled, koristi se jedan zajednički CSS dokument koji definiše dizajn celog sajta. Ovaj dokument se povezuje sa svakom veb-stranicom pojedinačno, čime se postiže vizuelna doslednost i jasno pokazuje da sve stranice pripadaju istom veb-sajtu.

Slike koje se koriste na veb-sajtu poželjno je organizovati u poseban podfolder, što olakšava preglednost i upravljanje fajlovima. Prilikom dodavanja slika u HTML dokumente, neophodno je navesti i putanju do foldera u kome se one nalaze, kako bi se pravilno prikazale u internet pregledaču.

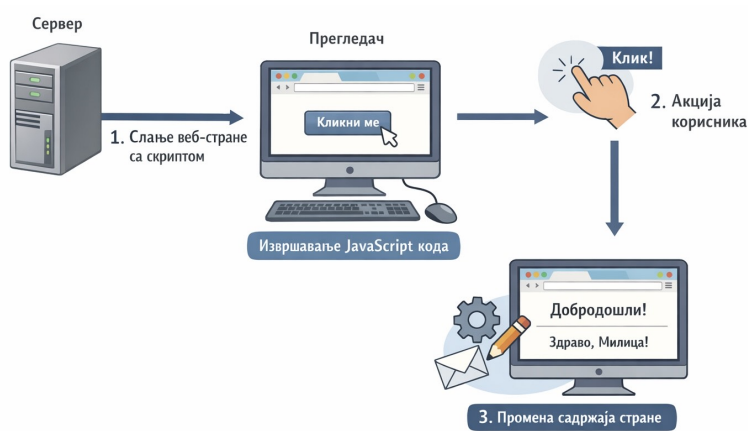
2. Serversko veb-programiranje

2.1. Veb-stranice

Na početku razvoja veba, veb-stranice su bile uglavnom statičke, što znači da su sadržale unapred pripremljen tekst i slike koje su svi korisnici videli na isti način. Korisnici su mogli samo da čitaju sadržaj i da se kreću između stranica pomoću linkova, bez ikakve naprednije interakcije.

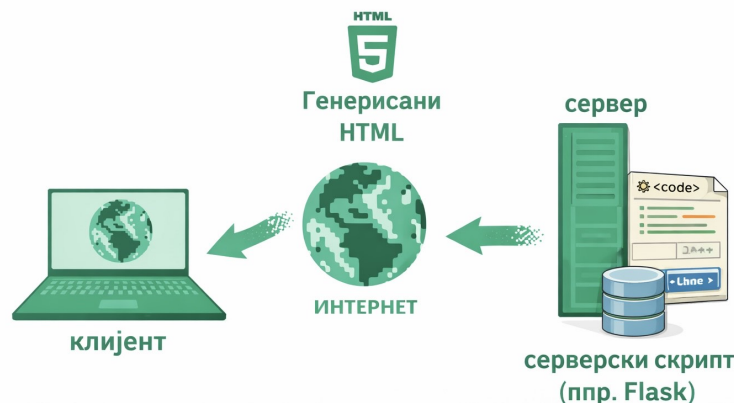
Kako je veb brzo napredovao, javila se potreba za većom interakcijom sa korisnicima. Zbog toga su, pored HTML-a i CSS-a, počeli da se koriste i programski jezici koji omogućavaju dinamičko ponašanje veb-stranica. U zavisnosti od toga gde se programski kod izvršava, razlikujemo klijentske i serverske veb-tehnologije.

Aktivne strane (klijentski skriptovi). Klijentske veb-tehnologije se izvršavaju u internet pregledaču korisnika. Najpoznatiji jezik za ovu namenu je **JavaScript**. On omogućava da veb-stranice postanu interaktivne, odnosno da reaguju na akcije korisnika, kao što su klikovi mišem, unos teksta ili izbor određenih opcija. Klijentski skriptovi se šalju sa servera zajedno sa veb-stranicom i izvršavaju se na računaru korisnika. Ove stranice se često nazivaju aktivne ili interaktivne veb-stranice.



Slika 11: Klijentski skriptovi

Dinamičke strane (serverski skriptovi). Serverske veb-tehnologije se izvršavaju na serveru. Serverski skriptovi, napisani na jezicima kao što su PHP, Python ili ASP.NET, obrađuju podatke, najčešće iz baza podataka, i na osnovu toga generišu HTML stranicu koja se šalje klijentu. Korisnik nikada nema uvid u sam programski kod serverskog skripta, već vidi samo rezultat njegovog izvršavanja. Ovakve stranice nazivaju se dinamičke veb-stranice.



Slika 12: Serverski skriptovi

Savremene veb-aplikacije najčešće kombinuju klijentske i serverske skriptove. Deo aplikacije koji se izvršava u pregledaču naziva se **prednja strana** (*fronted*), dok se deo koji radi na serveru naziva **zadnja strana** (*backed*). Postoje programeri koji se specijalizuju samo za jednu od ovih oblasti, ali i oni koji se bave obema, takozvani **full-stack** programeri.

Ajax. Posebno važan koncept u savremenim veb-aplikacijama je Ajax. On omogućava da veb-stranica komunicira sa serverom i razmenjuje podatke bez ponovnog učitavanja cele stranice. Na ovaj način je moguće, na primer, automatsko dopunjavanje pojmova pri pretrazi, ažuriranje poruka u čet aplikacijama ili učitavanje delova digitalnih mapa dok se korisnik kreće po njima.

Ajax koristi JavaScript za slanje zahteva serveru i obradu podataka koji stižu sa servera, najčešće u JSON (*JavaScriptObjectNotation*) formatu. Ovaj pristup omogućava brži i efikasniji rad veb-aplikacija i bolje korisničko iskustvo. Zahvaljujući Ajax-u razvijen je i koncept **jednostraničnih aplikacija**, kod kojih se ceo sadržaj aplikacije prikazuje na jednoj veb-stranici, a menja se dinamički.

2.2. HTTP protokol

Razmena podataka između klijenta (veb-pregledača) i servera na internetu obavlja se pomoću protokola HTTP (*HyperText Transfer Protocol*) ili njegove sigurnije varijante HTTPS (*HyperText Transfer Protocol Secure*). Osnovna razlika između ova dva protokola je u tome što HTTPS koristi šifrovanje podataka, čime se obezbeđuje zaštita informacija tokom njihovog prenosa kroz mrežu.

Kada korisnik u veb-pregledač unese URL adresu ili klikne na neku vezu, pregledač najpre analizira unetu adresu i razdvaja je na njene delove: protokol, naziv servera, putanju do resursa i eventualne parametre. Na osnovu ovih informacija, pregledač

zna kojim protokolom i sa kojim serverom treba da ostvari komunikaciju. Na primer:
`http://www.moja-prodavnica.com/site/products?page=3`

- `http` –oznaka protokola koji se koristi za komunikaciju
- `www.moja-prodavnica.com` –naziv veb-servera
- `/site/products` –putanja do veb-strane na serveru
- `page=3` -dodatni parametri

Sledeći korak je dobijanje IP adrese servera, što se postiže slanjem upita DNS serveru. Nakon toga, pregledač šalje serveru HTTP zahtev, u kome se navodi koja se veb-strana ili resurs traži, kao i dodatni parametri ako postoje. Server zatim obrađuje zahtev i, ukoliko traženi resurs postoji, šalje klijentu HTTP odgovor koji sadrži status zahteva i podatke koji se prikazuju u pregledaču.

Kada klijent primi HTML dokument, on ga analizira i prikazuje. Ukoliko se unutar HTML koda nalaze slike, stilovi ili multimedijalni sadržaji, pregledač šalje dodatne HTTP zahteve za te resurse. Zbog toga se na sporijim internet vezama najpre prikazuje tekst, a zatim se postepeno učitavaju slike i ostali elementi.

Svaki HTTP odgovor sadrži statusni kod koji obaveštava klijenta o ishodu zahteva. Najčešći status je **200 OK**, koji označava uspešno obrađen zahtev, dok kod **404 Not Found** označava da tražena stranica ne postoji. Pored statusne linije, odgovor sadrži i zaglavlja sa informacijama kao što su datum slanja odgovora, tip sadržaja i njegova veličina.

Praćenje HTTP komunikacije moguće je korišćenjem ugrađenih alata za programere u savremenim veb-pregledačima. U okviru alata za programere, kartica Network omogućava detaljan uvid u sve HTTP zahteve i odgovore, njihove statusne kodove, zaglavlja i sadržaj poruka, čime se olakšava analiza komunikacije između klijentske i serverske strane.

Iako se danas uglavnom koriste grafički alati u pregledačima, HTTP protokol se može koristiti i iz komandne linije, na primer uz pomoć alata **telnet** za HTTP ili **openssl** za HTTPS komunikaciju. Ovakvim pristupom jasno se vidi struktura HTTP zahteva i odgovora, kao i način na koji server reaguje na različite zahteve.

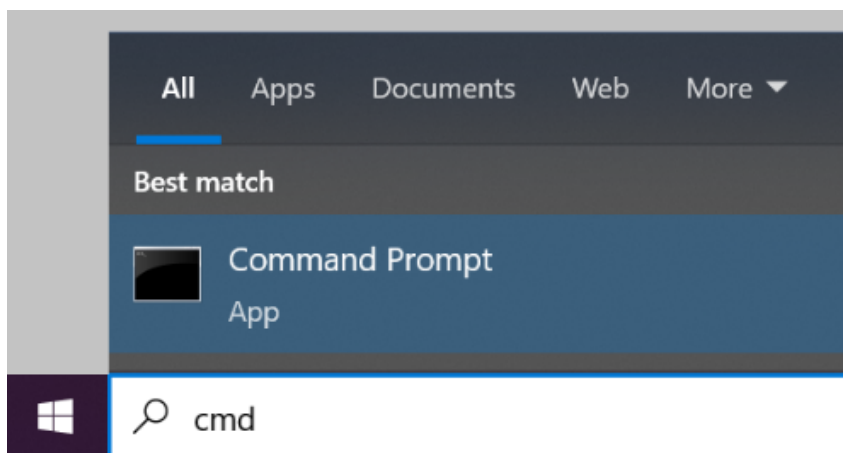
HTTPS protokol se sve više koristi jer obezbeđuje poverljivost i sigurnost podataka. On sprečava da treća lica čitaju ili menjaju podatke tokom njihovog prenosa i zato je danas standard za većinu savremenih veb-sajtova.

3. Flask biblioteka

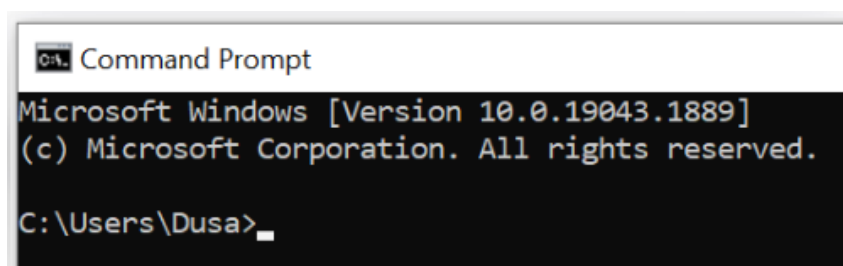
3.1. Osnovni pojmovi

Flask (<https://flask.palletsprojects.com/>) je biblioteka programskog jezika Python koja se koristi za kreiranje veb-aplikacija. Pored Flask-a, u Python-u postoje i druge biblioteke za razvoj veb-aplikacija, jedna od njih je **Django** (<https://www.djangoproject.com/>). Za razliku od Django-a, Flask je osmišljen kao lagana i jednostavna biblioteka, što ga čini pogodnim za početnike i manje projekte.

Kako se instalira biblioteka Flask? Instalacija biblioteke Flask je jednostavna i vrši se pomoću alatke `pip`, koja služi za upravljanje paketima u programskom jeziku Python. Potrebno je pokrenuti komandnu liniju (na primer, `unosomcmd` u pretragu operativnog sistema).

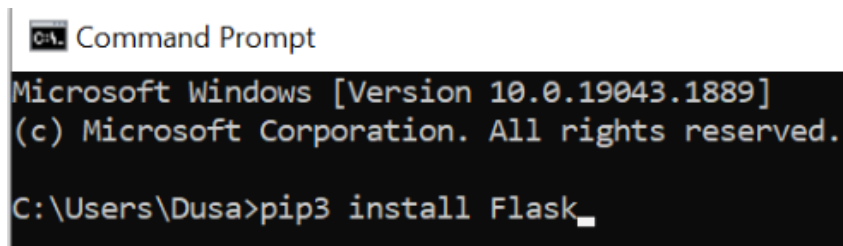


Slika 13: Unosimo cmd



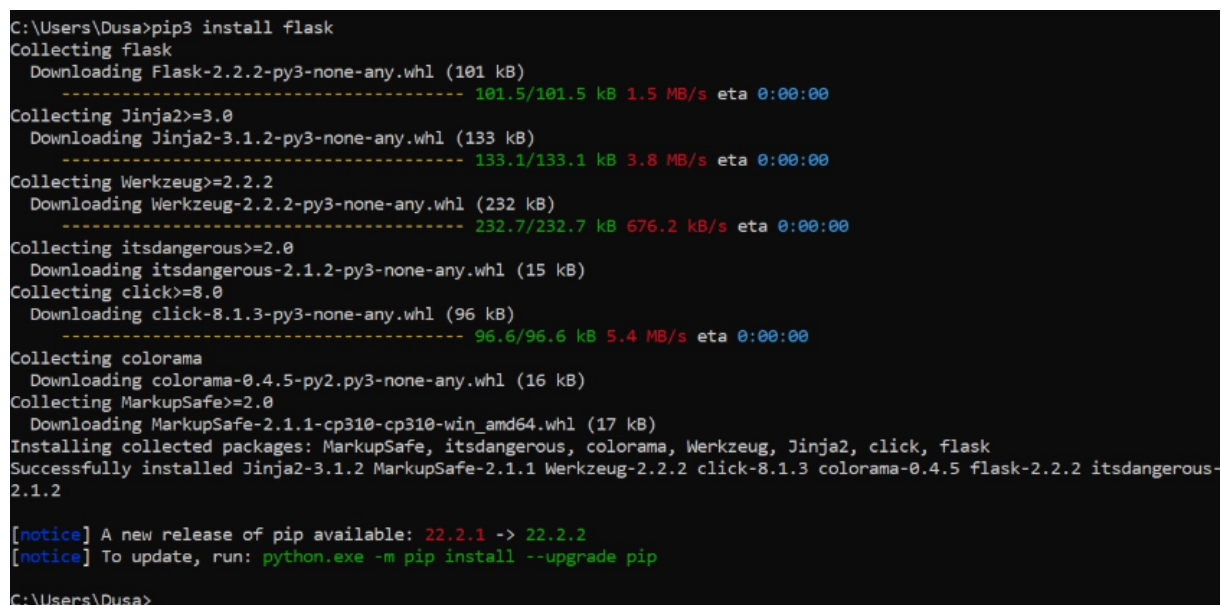
Slika 14: Otvoriće se komandni prozor

Pod uslovom da je na računaru već instaliran Python 3, u komandnu liniju je dovoljno uneti sledeću komandu:



```
C:\Users\Dusa>pip3 install Flask
```

Nakon izvršavanja komande i uspešne instalacije, biblioteka Flask je spremna za korišćenje. Ukoliko Python nije instaliran, neophodno je prvo instalirati programski jezik Python, a zatim nastaviti sa instalacijom Flask biblioteke.



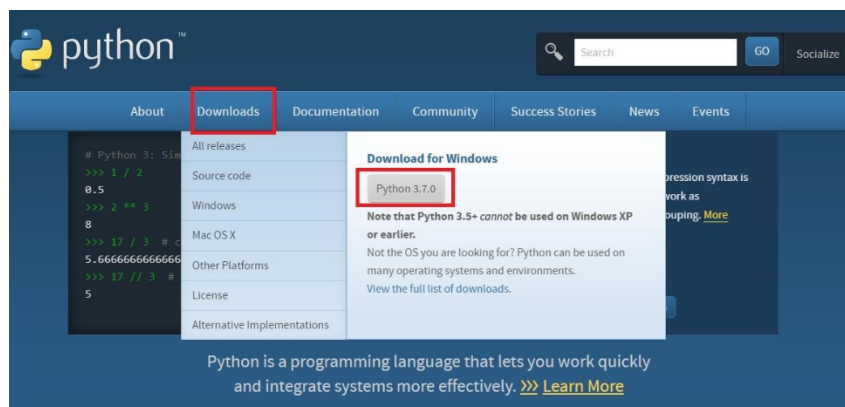
```
C:\Users\Dusa>pip3 install flask
Collecting flask
  Downloading Flask-2.2.2-py3-none-any.whl (101 kB)
----- 101.5/101.5 kB 1.5 MB/s eta 0:00:00
Collecting Jinja2>=3.0
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
----- 133.1/133.1 kB 3.8 MB/s eta 0:00:00
Collecting Werkzeug>=2.2.2
  Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
----- 232.7/232.7 kB 676.2 kB/s eta 0:00:00
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting click>=8.0
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
----- 96.6/96.6 kB 5.4 MB/s eta 0:00:00
Collecting colorama
  Downloading colorama-0.4.5-py2.py3-none-any.whl (16 kB)
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.1-cp310-cp310-win_amd64.whl (17 kB)
Installing collected packages: MarkupSafe, itsdangerous, colorama, Werkzeug, Jinja2, click, flask
Successfully installed Jinja2-3.1.2 MarkupSafe-2.1.1 Werkzeug-2.2.2 click-8.1.3 colorama-0.4.5 flask-2.2.2 itsdangerous-2.1.2

[notice] A new release of pip available: 22.2.1 -> 22.2.2
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Dusa>
```

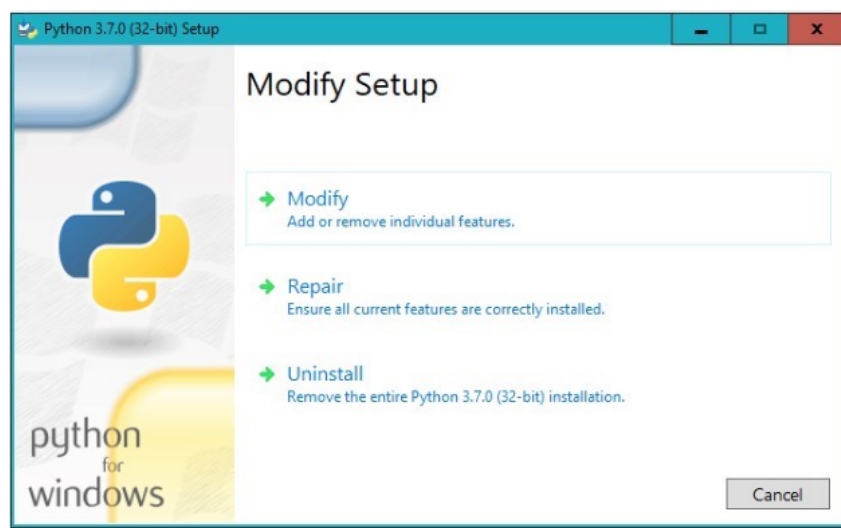
Slika 15: Pokrenut i uspešno sproveden

Kako da instaliramo jezik Python? Za instalaciju Python-a potrebno je posetiti zvanični sajt <https://www.python.org/> i u delu **Downloads** preuzeti najnoviju dostupnu verziju jezika Python. Preporučuje se instalacija aktuelne verzije koja je u tom trenutku ponuđena.



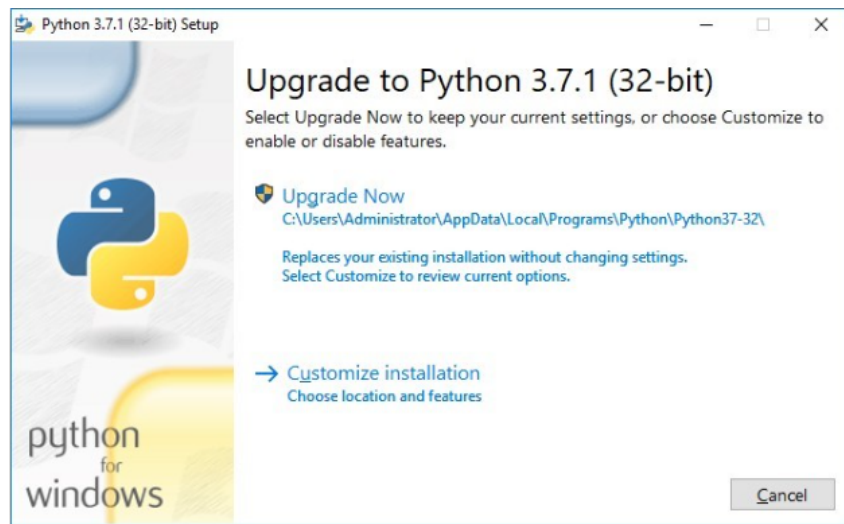
Slika 16: Idite na Downloads i kliknite na dugme za aktuelnu verziju kao na slici

Nakon preuzimanja, potrebno je pokrenuti instalacioni fajl.



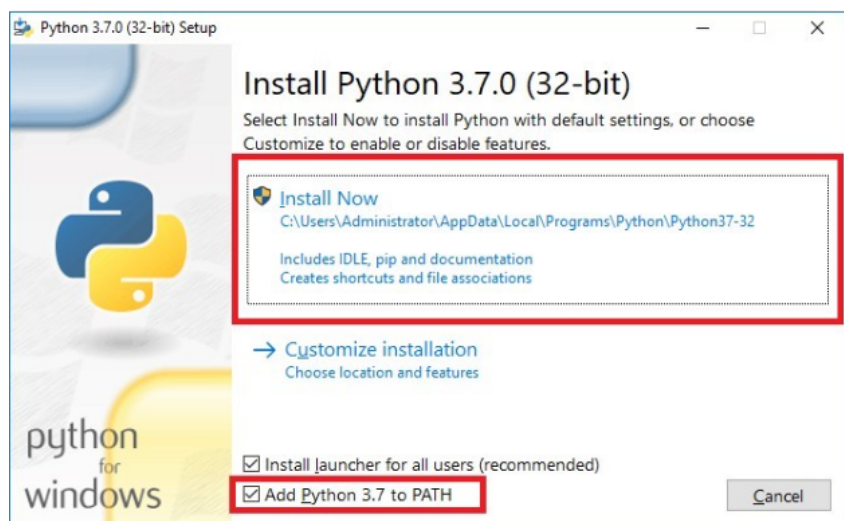
Slika 17: Ako se otvori ovaj prozor, kliknete samo na Close i koristite instaliran Python

Ukoliko je Python već instaliran, instalater će to prepoznati i ponuditi mogućnost nadogradnje ili prekida instalacije.



Slika 18: Kliknuti na dugme Upgrade Now

U slučaju da Python nije prethodno instaliran, potrebno je označiti opciju Add Python to PATH, a zatim izabrati Install Now kao na slici ispod.



Po završetku instalacije pojaviće se poruka koja potvrđuje da je instalacija uspešno obavljena, nakon čega se prozor može zatvoriti.

3.2. Prva Flask veb-aplikacija

Rad sa bibliotekom Flask započinje se kreiranjem najjednostavnije moguće veb-aplikacije, čiji je jedini zadatak da u veb-pregledaču prikaže tekst „Zdravo“. Ova vrsta aplikacije je poznata kao **Hello World** primer i tradicionalno se koristi pri učenju novih programskih jezika i tehnologija.

Svaka veb-aplikacija se sastoji od više fajlova, pa je uobičajena praksa da se oni organizuju u poseban folder. U ovom primeru kreira se folder pod nazivom `hello`, u kome se nalazi glavni programski fajl `app.py`. U njemu se definiše Flask aplikacija i jedna funkcija koja vraća tekst koji će biti prikazan u pregledaču.

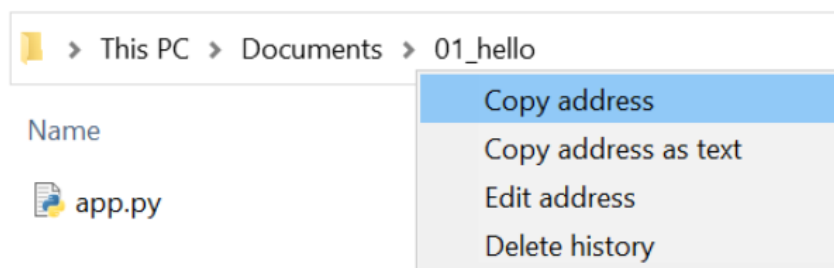
```
from flask import Flask

app = Flask(__name__)

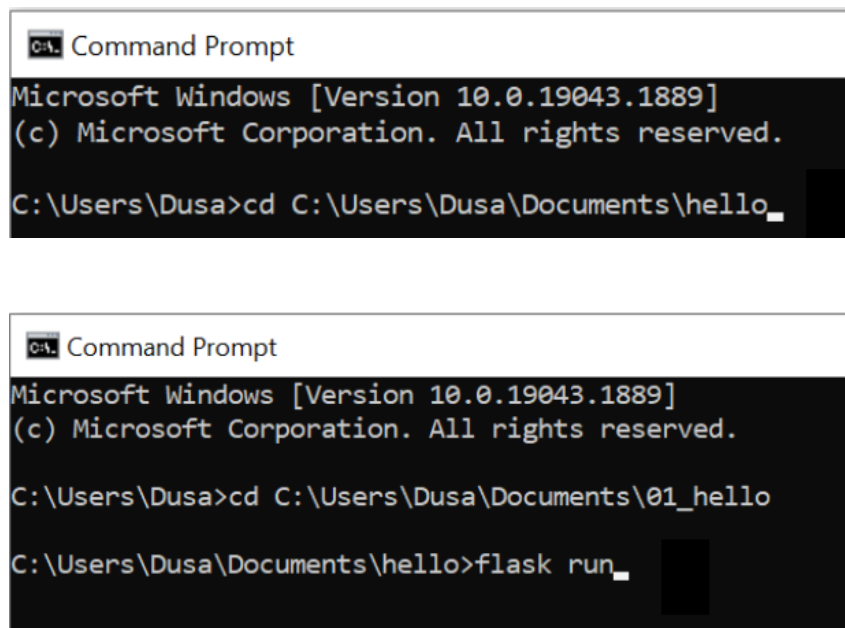
@app.route("/")
def index():
    return "Zdravo"
```

Programski fajlovi mogu se kreirati u bilo kom editoru teksta, kao što je Notepad, pri čemu je važno da se fajl sačuva sa ekstenzijom `.py`. Naziv `app.py` ima posebnu ulogu, jer Flask očekuje da se glavna aplikacija nalazi upravo u tom fajlu.

Da bismo imali tačnu putanju do fajla koja nam je neophodna da pokrenemo program, možemo da upotrebimo opciju **Copy address** kada uradimo desni klik mišem u prozoru **File Explorer**.



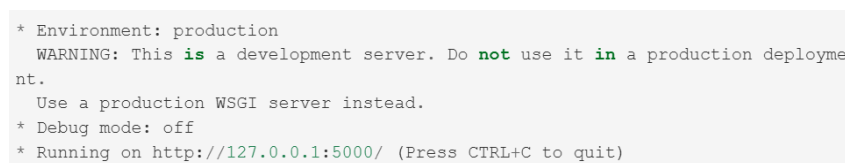
Nakon toga, u komandnoj liniji prelazimo u direktorijum u kome smo kreirali datoteku `app.py` tako što napišemo sledeće:



```
C:\Users\Dusa>cd C:\Users\Dusa\Documents\hello_

C:\Users\Dusa\Documents\hello>flask run_
```

Slika 19: Pokrećemo Flask aplikaciju kodom `flask run`



```
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Slika 20: Poruka koju dobijamo, ako je sve u redu.

Kada se aplikacija uspešno pokrene, Flask pokreće lokalni veb-server dostupan na adresi `http://127.0.0.1:5000/`. Unosom ove adrese u veb-pregledač prikazuje se strana sa tekстом „Zdravo“.



Zdravo

Veb-server radi sve dok se ne zaustavi kombinacijom tastera *Ctrl* + *C*. Ukoliko se naprave izmene u programskom kodu, potrebno je zaustaviti server, sačuvati izmene i ponovo pokrenuti aplikaciju kako bi se one prikazale u pregledaču.

Glavna Flask aplikacija predstavljena je objektom klase `Flask`, koji se najčešće smešta u promenljivu `app`. Prilikom kreiranja tog objekta koristi se specijalna promenljiva `__name__`, koja omogućava Flask-u da pravilno identifikuje aplikaciju.

```
from flask import Flask

app = Flask(__name__)
```

Veb-aplikacije obično imaju više različitih strana, a koja strana će biti prikazana zavisi od putanje u URL adresi. Kada korisnik unese određenu adresu u pregledač, šalje se HTTP zahtev veb-serveru, a aplikacija određuje koji će sadržaj biti vraćen kao odgovor.

U Flask-u se ovaj proces naziva **rutiranje**. Ruta predstavlja vezu između URL putanje i funkcije koja će obraditi zahtev. To se postiže upotrebom dekoratora `@app.route(...)`, koji se postavlja iznad definicije funkcije. **Dekorator** je mehanizam u programiranju koji omogućava da se funkciji ili metodi doda dodatna funkcionalnost bez izmene njenog izvornog koda. Funkcija vraća tekst ili drugi sadržaj koji se zatim prikazuje u pregledaču.

```
@app.route("/")
def index():
    return "Zdravo"
```

Na primer, ako je funkcija povezana sa putanjom `/`, ona će biti pozvana kada se pristupi osnovnoj adresi veb-aplikacije. Ako je povezana sa `/about`, ta funkcija će se izvršiti kada se u pregledaču unese adresa koja završava sa `/about`.

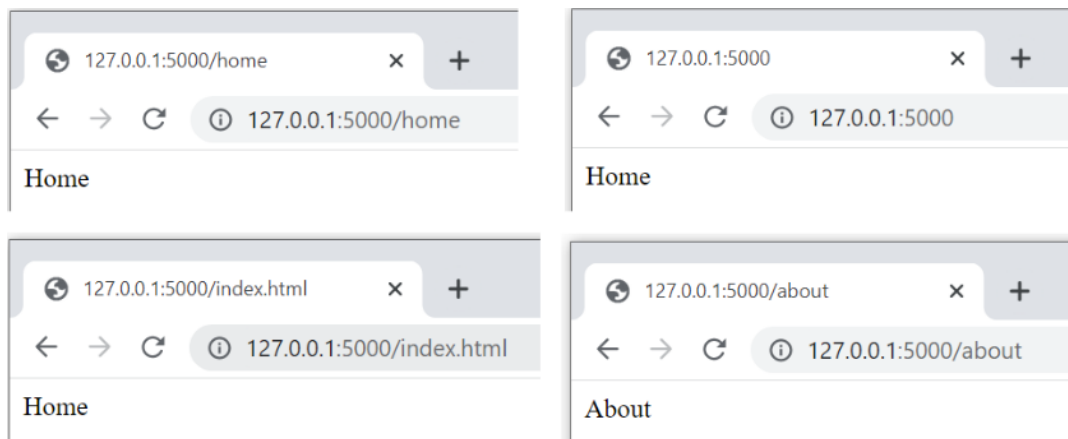
Jedna funkcija može biti povezana sa više različitih putanja, što se postiže navođenjem više dekoratora. Na taj način, isti sadržaj može biti dostupan preko više različitih URL adresa.

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
@app.route("/index.html")
@app.route("/home")
def home():
    return "Home"

@app.route("/about")
def about():
    return "About"
```



Funkcije koje obrađuju HTTP zahteve u Flask-u kao povratnu vrednost vraćaju nisku karaktera koja predstavlja sadržaj veb-strane, a taj sadržaj se šalje klijentu i prikazuje u pregledaču.

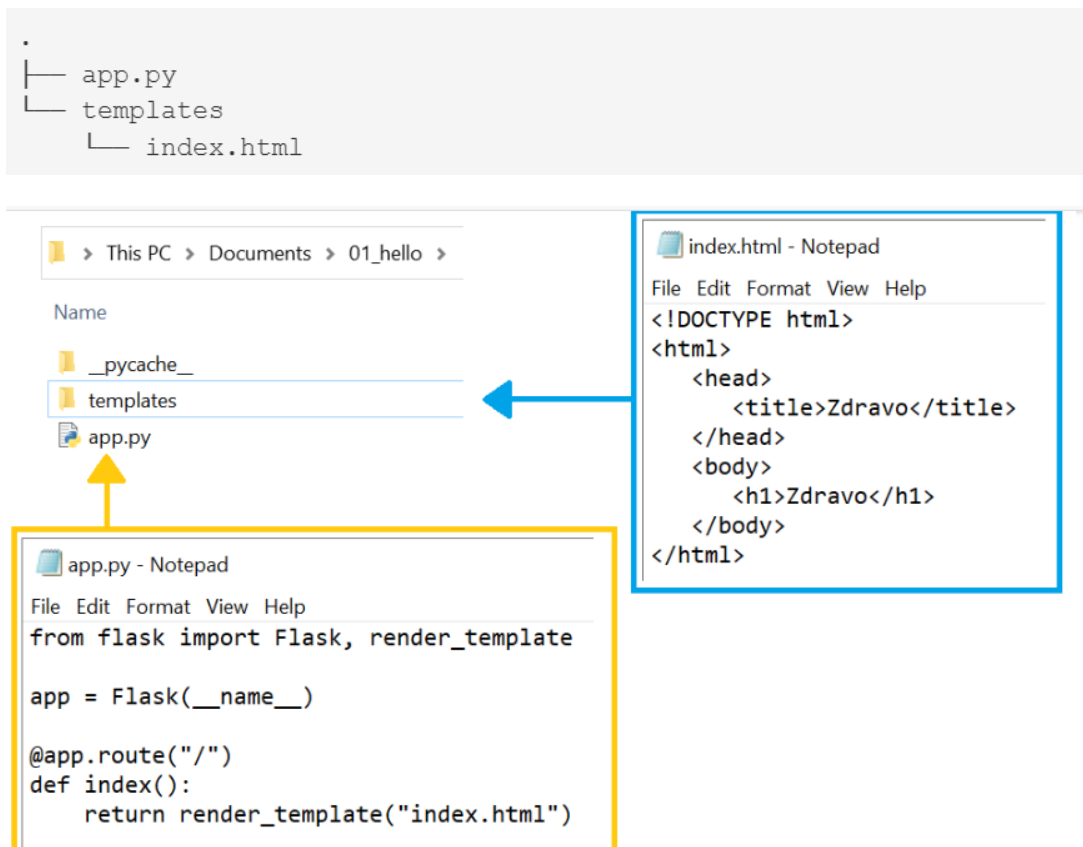
3.3. Šabloni

Veb-aplikacije kao odgovor klijentu najčešće šalju HTML dokumente. Direktno pisanje većih HTML sadržaja unutar programskog koda u vidu tekstualnih niski brzo postaje nepregledno i otežava održavanje aplikacije. Zbog toga savremeni veb-okviri, uključujući Flask, omogućavaju izdvajanje HTML koda u posebne datoteke koje se nazivaju **šabloni**.

U praksi se često koriste već postojeći primeri programskog koda koji se zatim prilagođavaju konkretnim potrebama aplikacije, umesto pisanja celokupnog rešenja od početka. U početnim primerima, odgovor koji aplikacija vraća klijentu predstavlja običan tekst, a ne ispravan HTML dokument. Da bi se generisala pravilna HTML stranica, potrebno je izmeniti funkciju koja obrađuje HTTP zahtev tako da vraća kompletan HTML sadržaj.

U najjednostavnijem slučaju, HTML se može vratiti kao višelinijaska tekstualna niska, pri čemu se koriste trostruki navodnici kako bi se omogućilo pisanje koda u više redova. Iako je ovakav pristup funkcionalan, mešanje Python i HTML koda u istoj datoteci nije preporučljivo zbog loše preglednosti i teškog održavanja.

Zbog toga Flask koristi biblioteku **Jinja** (<https://jinja.palletsprojects.com/>), koja omogućava rad sa šablonima. Jinja omogućava da se HTML kod čuva u posebnim datotekama, dok se u Python programu vrši njihova obrada i generisanje konačne HTML strane koja se prosleđuje klijentu. Flask podrazumeva da se svi šabloni nalaze u posebnom direktorijumu pod nazivom **templates**. U tom slučaju, aplikacija se sastoji od glavne Python datoteke i odgovarajućih HTML šablona koji se nalaze u pomenutom direktorijumu.



Za prikaz HTML stranice koristi se funkcija `render_template`, koja učitava šablon, obrađuje ga i vraća gotov HTML sadržaj kao odgovor klijentu. Pored samog naziva šablona, ovoj funkciji se mogu proslediti dodatni podaci koji se zatim koriste unutar šablona.

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html",
                           naslov="Predmeti",
                           spisak=["Matematika", "Srpski jezik", "Informatika", "Fizika"])
```

Slika 21: Datoteka `app.py`


```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Jinja šabloni</title>
  </head>
  <body>
    {% if naslov %}
    <h1>{{ naslov }}</h1>
    {% endif %}
    <ul>
      {% for stavka in spisak %}
      <li>{{ stavka }}</li>
      {% endfor %}
    </ul>
  </body>
</html>

```

Slika 22: Datoteka templates/index.html

Posebna prednost Jinja šablona jeste mogućnost korišćenja šablonskih direktiva, koje omogućavaju dinamičko generisanje sadržaja. U okviru HTML koda mogu se koristiti promenljive, uslovi i petlje, čija je sintaksa slična programskom jeziku Python. Izrazi zapisani između `{{ }}` zamenjuju se svojim vrednostima, dok se direktive poput `{% if %}` i `{% for %}` koriste za kontrolu toka izvršavanja unutar šablona.

Na ovaj način moguće je, na primer, prikazati liste podataka, tabele ili složenije strukture, pri čemu se konkretni podaci prosleđuju iz Python koda. Šabloni se često koriste za prikaz podataka koji potiču iz baza podataka, iako se u jednostavnijim primerima mogu koristiti i unapred definisane liste ili rečnici.

Šabloni omogućavaju i generisanje složenijih struktura, kao što su HTML tabele, pa čak i izvođenje jednostavnih proračuna pomoću ugnežđenih petlji. Time se značajan deo logike prikaza može premestiti iz programskog koda u šablon, čime se poboljšava čitljivost i organizacija aplikacije.

Vežbe radi, definišimo jednostavnu aplikaciju koja na osnovu liste parova (ime, prezime) gradi i prikazuje tabelu koja sadrži imena i prezimena.

```

from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    ucenici = [{"ime": "Петар", "prezime": "Петровић"},
               {"ime": "Јована", "prezime": "Јовановић"},
               {"ime": "Ана", "prezime": "Анић"}]
    return render_template("ucenici.html", ucenici=ucenici)

```

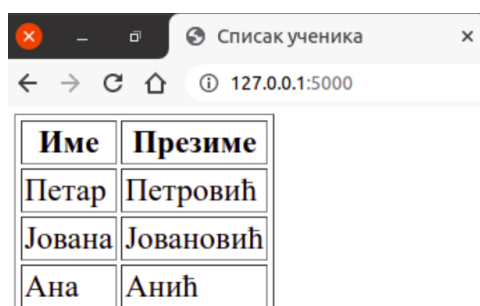
Slika 23: Datoteka app.py

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Списак ученика</title>
  </head>
  <body>
    <table border="1">
      <tr><th>Име</th><th>Презиме</th></tr>
      {% for ucenik in ucenici %}
      <tr><td>{{ ucenik.ime }}</td><td>{{ ucenik.prezime }}</td></tr>
      {% endfor %}
    </table>
  </body>
</html>

```

Slika 24: Šablon `templates/ucenici.html`



Име	Презиме
Петар	Петровић
Јована	Јовановић
Ана	Анић

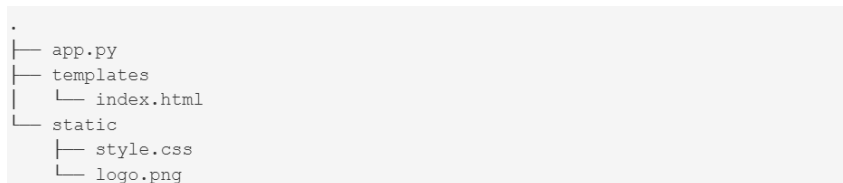
Slika 25: Izgled strane unutar pregledača veba

U aplikacijama koje se sastoje od više veb-strana često dolazi do ponavljanja istih delova HTML koda, poput zaglavlja, navigacije ili uključanja CSS i JavaScript biblioteka. Kako bi se ovo izbeglo, Jinja podržava **nasleđivanje** šablona. Zajednički delovi stranica izdvajaju se u osnovni šablon, dok ostali šabloni nasleđuju njegov sadržaj i definišu samo delove koji se razlikuju. Nasleđivanje šablona predstavlja važnu tehniku u razvoju veb-aplikacija, jer omogućava centralizovano održavanje zajedničkog sadržaja i značajno smanjuje ponavljanje koda. Time se aplikacija čini preglednijom, modularnijom i lakšom za dalji razvoj i održavanje.

3.4. Statičke datoteke i linkovi

Veb-stranica se ne sastoji samo od HTML koda, već i od različitih pratećih datoteka kao što su CSS stilovi, JavaScript datoteke, slike, video i audio-zapisi. Iako se kod dinamičkih veb-aplikacija HTML sadržaj često generiše na serveru, ove prateće datoteke su najčešće statičke i klijentu se šalju u nepromenjenom obliku.

U Flask veb-aplikacijama postoji jasno definisana struktura direktorijuma. HTML šabloni se smeštaju u direktorijum `templates`, dok se sve statičke datoteke (CSS, slike, JavaScript) nalaze u direktorijumu `static`. Na primer, u direktorijumu `static` mogu se nalaziti CSS datoteka za stil i slika koja se prikazuje na veb-strani.



Slika 26: Primer strukture i sadržaja direktorijuma veb-aplikacije

Prilikom korišćenja statičkih datoteka u HTML šablonima, nije preporučljivo ručno navoditi relativne ili apsolutne putanje, jer se putanja veb-stranice može menjati ili isti šablon može biti korišćen na više različitih URL-ova. Zbog toga Flask pruža funkciju `url_for`, koja automatski generiše ispravnu putanju u trenutku prikaza stranice.

Za pristup statičkim datotekama, funkciji `url_for` se kao prvi argument prosleđuje niska `static`, a kao imenovani argument navodi se naziv datoteke. Na taj način se, na primer, CSS stil uključuje u zaglavlju strane, a slika u telu dokumenta, bez potrebe da se unapred zna tačna putanja do datoteke. Flask će automatski generisati URL oblika `/static/naziv_datoteke`.

Na slikama ispod je ceo šablon `index.html` u kome se u zaglavlju uključuje CSS datoteka `style.css` (iz poddirektorijuma `static`), a u telu se uključuje slika `logo.png` (takode iz poddirektorijuma `static`).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Uključivanje statičkih datoteka</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
  }}" />
  </head>
  <body>
    <h1>Zdravo</h1>

    
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Uključivanje statičkih datoteka</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="/static/style.css" />
  </head>
  <body>
    <h1>Zdravo</h1>

    
  </body>
</html>
```



Slika 27: To bi u pregledaču izgledalo ovako

Pored statičkih datoteka, funkcija `url_for` se koristi i za pravljenje linkova ka drugim stranicama u okviru iste veb-aplikacije. Umesto da se u HTML-u ručno upisuju putanje, kao vrednost atributa `href` navodi se poziv funkcije `url_for` sa imenom Python funkcije koja generiše željenu stranicu. Ovak pristup olakšava održavanje aplikacije, jer se promene putanja automatski odražavaju na sve linkove.

Često se pravi zajednički osnovni šablon koji sadrži navigaciju, a koji se nasleđuje u ostalim stranicama. U takvom šablonu linkovi ka stranicama kao što su „Home“ i „About“ se definišu pomoću funkcije `url_for`, čime se obezbeđuje fleksibilnost i ispravno funkcionisanje navigacije u celoj veb-aplikaciji.

```
<!DOCTYPE html>
<html>
  <head>
    <title>{{ title }}</title>
  </head>
  <body>
    <ul>
      <li><a href="{{ url_for('home') }}">Home</a></li>
      <li><a href="{{ url_for('about') }}">About</a></li>
    </ul>
    {% block content %}
    {% endblock %}
  </body>
</html>
```

Slika 28: Primer zajedničkog šablona `osnovni.html` (koji se nasleđuje za sve veb-strane na sajtu) sa navigacionim linkovima na vrhu strane

4. Literatura

- <https://petlja.org/sr-Latn-RS/kurs/18657/20/6544>
- https://sr.wikipedia.org/sr-ec/Ra%C4%8Dunarska_mre%C5%BEa
- <https://www.w3schools.com/html/default.asp>