Univerzitet Singidunum

VEB PROGRAMIRANJE

Miloš Dobrojević
Nebojša Bačanin-Džakula

Beograd, 2021.

UNIVERZITET SINGIDUNUM

Miloš Dobrojević
Nebojša Bačanin-Džakula

VEB PROGRAMIRANJE

PRVO IZDANJE

Beograd, 2021.

VEB PROGRAMIRANJE

Autori:
dr Miloš Dobrojević
dr Nebojša Bačanin-Džakula

Recenzenti:
dr. Mladen Veinović
dr. Miodrag Živković
dr. Boško Nikolić

Izdavač:
UNIVERZITET SINGIDUNUM
Beograd, Danijelova 32
www.singidunum.ac.rs

Za izdavača:
dr Milovan Stanišić

Tehnička obrada:
dr Miloš Dobrojević
Miloš Višnjić

Dizajn korica:
Aleksandar Mihajlović, MA

Godina izdanja:
2021.

Tiraž:
1200 primeraka

Štampa:
Birograf, Beograd

ISBN: 978-86-7912-752-5

SADRŽAJ

PREDGOVOR **1**

1. PLATFORMA **7**
1.1 Osnovni pojmovi 7
1.1.1 HTML 7
1.1.2 Veb stranica 9
1.1.3 Veb server 9
1.1.4 Veb adresa 9
1.1.5 URL parametri 10
1.1.6 Veb sajt 10
1.1.7 Dinamički veb sajt 10
1.2 Osnove HTTP protokola 11
1.2.1 HTTP metode 11
1.2.2 HTTP zahtev 12
1.2.3 Odgovor veb servera 13
1.3 CSS 15
1.4 Javascript 15
1.5 MySQL / MariaDB 15
1.6 PHP programski jezik 16
1.7 XAMPP, instalacija i pokretanje komponenti 19
1.7.1 Instalacija 21
1.7.2 Konfiguracija platforme 24
1.8 Podešavanja Apache veb servera 25
1.8.1 Promena lokacije u kojoj se nalaze veb dokumenti 25
1.8.2 Izmena porta na koji se veb server odaziva 25
1.8.3 Bezbednost 26
1.8.4 Semantičke veb adrese 27
1.8.5 Ostala podešavanja veb servera 28
1.8.6 Podešavanje virtuelnog domena 29

2. PHP PROGRAMSKI JEZIK **35**
2.1 Uvod u sintaksu 35
2.2 Tipovi podataka 37
2.3 Promenljive 37
2.3.1 Vidljivost promenljivih 38
2.3.2 Superglobalne promenljive 40
2.4 Konstante 41
2.5 Operatori 41
2.5.1 Osnovne računske operacije, aritmetički operatori 41
2.5.2 Operatori dodele 42
2.5.3 Operatori umanjenja / uvećanja 42
2.5.4 Operatori poređenja 42
2.5.5 Logički operatori 43
2.5.6 Operatori uslovne dodele 43
2.6 Rad sa stringovima 44
2.7 Grananje 46
2.8 Petlje 46
2.9 Datum i vreme 48
2.10 Nizovi 51
2.10.1 Dužina niza 52
2.10.2 Ispis članova niza 52
2.10.3 Transformacija string u niz 53
2.10.4 Transformacija niza u string 53
2.10.5 Dinamičko generisanje niza 56
2.10.6 Rad sa nizovima 56
2.11 Funkcije 57
2.11.1 Anonimne funkcije 60
2.12 Serijalizacija podataka 61
2.13 HTTP Zaglavlja 63
2.14 Uključivanje PHP dokumenta iz drugih fajlova 64

3. VEB OBRASCI **69**
3.1 Označavanje namene elemenata 70
3.2 Imenovanje elemenata 71
3.3 Transfer podataka ka veb serveru 72
3.3.1 GET metoda 72
3.3.2 POST metoda 74
3.4 Otpremanje i prijem dokumenata 76
3.5 Provera podataka. Prijava greške. 79

4. RAD SA DOKUMENTIMA **85**
4.1 Čitanje dokumenata 85
4.1.1 Sekvencijalno čitanje dokumenta 87
4.2 Kreiranje novog dokumenta 88
4.3 Menadžment direktorijuma 89
4.3.1 Očitavanje sadržaja direktorijuma 89
4.3.2 Očitavanje i promena radnog direktorijuma 91
4.3.3 Kreiranje i brisanje direktorijuma 91
4.4 Menadžment dokumenata 92
4.4.1 Provera da li dokument postoji 92
4.4.2 Kopiranje dokumenta 93
4.4.3 Promena naziva i premeštanje dokumenta 93
4.4.4 Brisanje dokumenta 95
4.4.5 Manipulacija vremena 96

5. IDENTIFIKACIJA KORISNIKA **101**
5.1 HTTP kolačić 101
5.1.1 Struktura HTTP kolačića 102
5.1.2 Postavljanje i uklanjanje kolačića 103
5.2 Sesija 106
5.2.1 Upravljanje sesijama 106
5.3 Prepoznavanje korisnika 108

6. RAD SA OBJEKTIMA **115**
6.1 Klase i objekti 115
6.1.1 Modifikatori pristupa 116
6.1.2 Dodela vrednost atributa 117
6.1.3 Deklarisanje klase i instanciranje objekta 117
6.2 Nasleđivanje, intefejsi i statički atributi i metode 120
6.2.1 Sprečavanje nasleđivanja 123
6.2.2 Interfejsi 123
6.3 Statički atributi i metode 125

7. RAD SA BAZOM PODATAKA **129**
7.1 Privilegije i SQL DCL naredbe 129
7.2 PHP naredbe za rad sa bazom podataka 136
7.2.1 Proceduralna paradigma za rad sa bazom podataka 136
7.2.2 Objektno-orijentisana paradigma za rad sa bazom podataka 140
7.2.3 Izvoz baze 147
7.2.4 Kodiranje prikaza karaktera 149

8. TRANSFER PODATAKA **153**
8.1 XML format 153
8.2 JSON format 159
8.3 Poređenje XML i JSON formata 166
8.4 Prednosti i nedostaci XML i JSON formata 168

9. REGULARNI IZRAZI **173**
9.1 Razdelnici 173
9.2 Šablon 174
9.2.1 Metakarakteri 174
9.2.2 Pretraga bilo kog karaktera 174
9.2.3 Escaping 175
9.2.4 Klase karaktera 175
9.2.5 Kvantifikacija 177
9.2.6 Grupisanje 179
9.2.7 Ankeri 180
9.2.8 Modifikatori 180
9.3 Funkcije za rad sa regularnim izrazima 182

10. OTKLANJANJE GREŠAKA **189**
10.1 Greške u kodu 189
10.1.1 Obaveštenja 190
10.1.2 Upozorenja 190
10.1.3 Greške u sintaksi 191
10.1.4 Kritične greške 192
10.1.5 Kontrola prikaza grešaka 192
10.2 Greške u metodologiji 194
10.3 Izuzeci 197

11. FRONT-END TEHNOLOGIJE **201**
11.1 JavaScript HTML DOM 202
11.1.1 Pristup objektima HTML DOM-a pomoću JavaScript-a 203
11.1.2 HTML DOM događaji 213
11.1.3 DOM navigacija 226
11.2 JavaScript BOM 234
11.2.1 JavaScript BOM uvodne teme 234
11.2.2 JavaScript BOM naprednije teme 245
11.3 JavaScript AJAX 250
11.3.1 Osnovne teme 250
11.3.2 Naprednije teme 261

12. VEB APLIKACIJA **271**
12.1 Način rada 271
12.2 Prednosti i nedostaci veb aplikacija 271
12.3 Zajednički elementi veb aplikacija 273
12.3.1 Korisnički interfejs 274
12.3.2 Sistem za identifikaciju i autorizaciju korisnika 275
12.3.3 Taksonomija 278
12.3.4 Pretraga sadržaja 278
12.3.5 Višejezičnost 281
12.4 Arhitektura veb aplikacije 282
12.4.1 Slojevi aplikacije 284
12.4.2 Moduli aplikacije 286
12.4.3 Struktura dokumenata 287
12.4.4 MVC arhitektura 288
12.4.5 Ruter aplikacije 288

13. PRIMER PROJEKTNOG ZADATKA **295**
13.1 Sistem za upravljanje sadržajem 295
13.2 Osnovne postavke projekta 296
13.3 Grupisanje sadržaja 296
13.4 Struktura korisnika. Tipovi korisničkih naloga. 299
13.5 Struktura baze podataka 300
13.6 Korisnički interfejs 302
13.7 Arhitektura aplikacije 303
13.7.1 Jezgro aplikacije 307
13.7.2 Ruter aplikacije 310
13.7.3 Formiranje veb stranice 313
13.8 Moduli 314
13.8.1 Početna strana 315
13.8.2 Vesti 316
13.8.3 Kontakt stranica 331
13.8.4 Login 334
13.8.5 RSS 338
13.8.6 Greška 341
13.9 Javno dostupni dokumenti 342
13.9.1 JavaScript 342
13.9.2 CSS 343

14. LITERATURA **351**

15. INDEKS SLIKA **352**

16. INDEKS LISTINGA **355**

17. INDEKS TABELA **362**

PREDGOVOR

Veb aplikacije su nezaobilazni deo svakodnevnog, poslovnog i privatnog života. Koriste se za pristup elektronskoj pošti, za obavljanje kancelarijskih poslova, za upravljanje poslovnim procesima i izveštavanje, za pristup društvenim mrežama, oglasnicima, medijskim portalima i sl.

Ova knjiga je namenjena studentima koji prate predmet *Veb programiranje* na drugoj godini studija *Tehničkog fakulteta Univerziteta Singidunum*, na studijskom programu *Softversko i informaciono inženjerstvo*. Od studenata se očekuje da su prethodno odslušali predmete *Veb dizajn* i *Baze podataka*, odnosno da su upoznati sa *HTML* markirnim jezikom, *CSS*-om, osnovama *SQL* jezika, kao i sa osnovnim principima objektno orijentisanog programiranja.

Tehnologija za razvoj veb aplikacija sa kojom će se studenti upoznati na ovom predmetu je bazirana na *WAMP/LAMP/XAMPP*[1] platformi koju čine Apache veb server, programski jezici *PHP* (*backend*) i *JavaScript* (*frontend*), uz upotrebu *MySQL* ili *MariaDB* baze podataka.

PHP programski jezik je skriptni jezik koji se izvršava na strani servera, omogućava rad sa popularnim serverima baza podataka i otvorenog je koda. Prvenstveno je namenjen razvoju veb aplikacija, ali se može koristiti i u druge svrhe. U 2021. godini, *PHP* programski jezik koristi 8 od 10 (~79%) veb sajtova u svetu, među kojima su *Facebook, Wikipedia, Slack, Dailymotion*, i *Yahoo*!. Takođe, najpopularniji sistemi za upravljanje sadržajem (engl. *Content Management System, CMS*) kao što su *WordPress, Drupal, MediaWiki, Joomla*! i *Moodle* su napisani u *PHP* programskom jeziku.

Tekst je propraćen velikim brojem primera, listinga i slika, tako odabranih da čitaocu budu što jednostavniji za razumevanje, ali istovremeno i da utiču na formiranje inženjerskog načina razmišljanja u rešavanju problema u realnom okruženju. Iako je knjiga pisana kao udžbenik, osim studenata mogu je koristiti svi oni koji žele da se detaljnije upoznaju sa tehnologijom razvoja veb aplikacija.

1 *WAMP* - Windows OS, Apache web server, MySQL, PHP
LAMP - Linux OS, Apache web server, MySQL, PHP
XAMPP - Cross-platform, Apache web server, MariaDB, PHP i Perl

Knjiga je podeljena u 13 poglavlja, koja su napisali:

- **Miloš Dobrojević**

1. Platforma
2. PHP programski jezik
3. Veb obrasci
4. Rad sa dokumentima
5. Identifikacija korisnika
9. Regularni izrazi
10. Otklanjanje grešaka
12. Veb aplikacija
13. Primer projektnog zadatka

- **Nebojša Bačanin-Džakula**

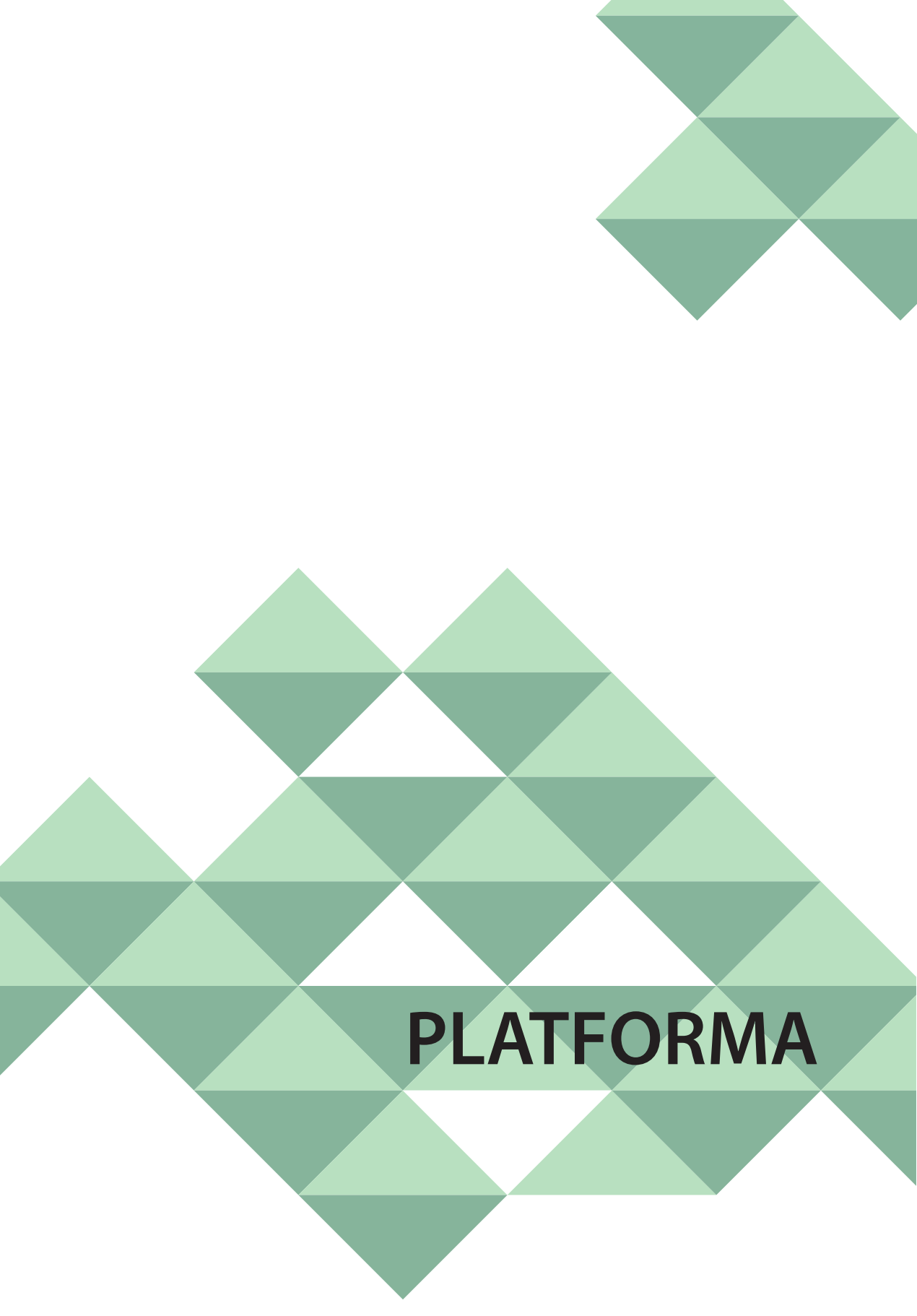
6. Rad sa objektima
7. Rad sa bazom podataka
8. Transfer podataka
11. Front-end tehnologije

Sve primedbe, komentari, preporuke, pohvale i eventualno informacije o uočenim greškama se mogu poslati na email adrese:

- mdobrojevic@singidunum.ac.rs ili
- nbacanin@singidunum.ac.rs.

Autori

Beograd, 2021.


PLATFORMA

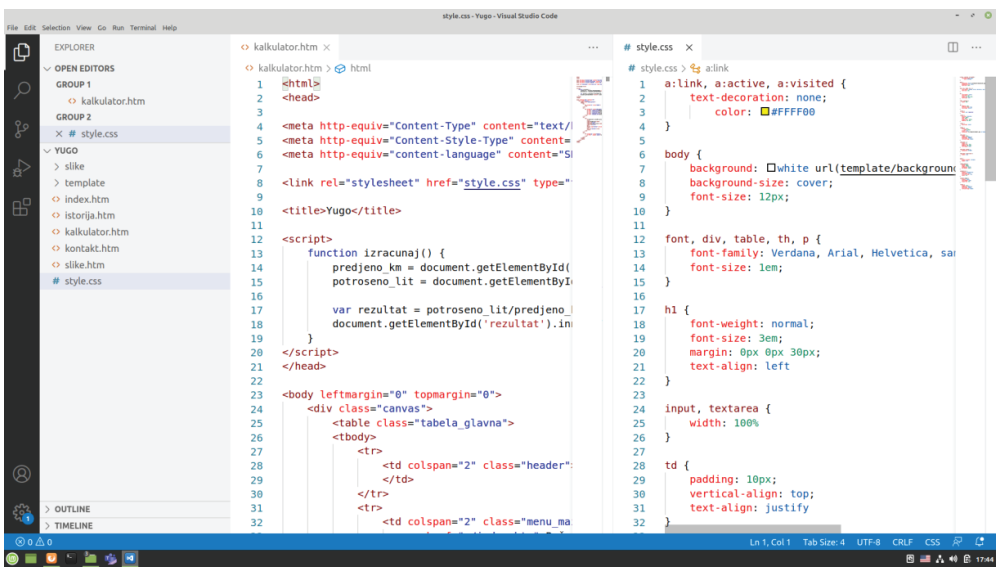
1. PLATFORMA

1.1 OSNOVNI POJMOVI

1.1.1 HTML

HTML je markirni jezik (engl. *HyperText Markup Langage*), koji se koristi za izradu veb stranica, a karakterističan je po tagovima koji razdvajaju instrukcije od sadržaja.

Slika 1 - Visual Studio Code, izgled radnog ekrana



1.1.2 Veb stranica

Veb stranica je tekstuelni dokument koji sadrži kod napisan u HTML markirnom jeziku, listing 1, i pregleda se pomoću veb brauzera. Iako je za njeno kreiranje dovoljan tekst editor, za ovu namenu se obično koriste namenski pravljene aplikacije, slika 1, tzv. editori koda. Popularni editori koda su prikazani u tabeli 1.

Tabela 1 - Popularni editori koda

1	Sublime text	Windows, Linux, MacOS
2	Atom	Windows, Linux, MacOS
3	Visual Studio Code	Windows, Linux, MacOS
4	Notepad2	Windows
5	Notepad++	Windows
6	Brackets	Windows, Linux, MacOS
7	Espresso	MacOs
8	Vim	Unix, Linux, Windows NT, MS-DOS, macOS, iOS, Android, Haiku, AmigaOS, MorphOS
9	BBedit	MacOS
10	Ultraedit	Windows, Linux, MacOS

Prilikom prikaza veb stranice, veb brauzer koordinira implementaciju teksta, veb linkova, slika i multimedije. Interakciju između korisnika i veb stranice je moguće kontrolisati pomoću JavaScript programskog jezika, a stilizovanje izgleda elemenata stranice se vrši putem CSS stilova.

Listing 1 - Osnovna struktura HTML dokumenta

```
<html>
<head>
    <!-- Zaglavlje stranice -->
    <meta charset="UTF-8">
    <link rel="stylesheet" href="style.css">
    <script src="prvi_primer.js"></script>
</head>
<body>
    <!-- Telo stranice -->
</body>
</html>
```

Veb stranice, na osnovu načina na koji su generisane i kontrolisane, mogu se podeliti na:

- *Statičke veb stranice*. Jednom formiran HTML kod se na menja duži vremenski period. Za prikaz ovakvih stranica je dovoljan samo veb brauzer.
- *Dinamičke veb stranice*. Sadržaj se menja u zavisnosti od stepena autorizacije korisnika. Npr. anoniman korisnik može da vidi samo prvi pasus teksta (tizer), registrovani korisnik može da vidi ceo tekst, a administrator još može i da izmeni tekst. Ovakve stranice se generišu na veb serveru pomoću odgovarajućeg programskog jezika, među koje ubrajamo i PHP programski jezik.
- *Interaktivne veb stranice*. Sadržaj stranice se delimično ili u potpunosti menja u skladu sa aktivnostima korisnika, što se postiže upotrebom JavaScript programskog jezika, HTML5 markirnog jezika i CSS3 stilova. Interaktivne mogu biti i statičke i dinamičke veb stranice.

1.1.3 Veb server

Pojam veb server se odnosi na softver, ili hardver koji pokreće istoimeni softver, koji je namenjen prijemu i ispunjavanju zahteva putem HTTP-a[2] primljenih preko interneta.

Kao odgovor na zahtev klijenta, veb server najčešće isporučuje HTML dokumente, iako može da isporuči i dokumente drugih tipova poput slika, CSS fajlova, JavaScript dokumenata i sl.

1.1.4 Veb adresa

Veb adresa, ili URL (engl. *Uniform Resource Locator*) predstavlja referencu ka veb resursu, odnosno njegovu lokaciju u računarskoj mreži. Formira se kao niz karaktera organizovanih po odgovarajućoj šemi:

```
https://www.singidunum.ac.rs/slike/logo.png
```

- http ili https sa pratećom dvotačkom - oznaka protokola
- www.singidunum.ac.rs - ime hosta
- /slike/logo.png - putanja do fajla i ime fajla koji se poziva

2 HTTP - Hypertext Transfer Protocol

1.1.5 URL parametri

Sastavni deo veb adrese mogu biti tzv. URL parametri, putem kojih se veb aplikaciji prenosi jedna ili više informacija:

```
https://www.singidunum.ac.rs/vesti.php?f=tf&order=desc
```

Segment veb adrese koji sadrži URL parametre je označen znakom pitanja (?). Parametri se navode u parovima ključ i vrednost, a međusobno su razdvojeni znakom *ampersand* (&). U navedenom primeru, značenje URL parametara je navedeno u tabeli 2.

Tabela 2 - Primer upotrebe URL parametara

Parametar	Opis
f=tf	*f* označava fakultet, vrednost *tf* označava tehnički fakultet
order=desc	ključ *order* označava redosled prikaza vesti, vrednost *desc* označava da će vesti biti prikazane opadajućim redosledom, npr. po datumu objave.

1.1.6 Veb sajt

Veb sajt predstavlja skup međusobno povezanih veb stranica i multimedijalnih dokumenata koji se nalaze na zajedničkom domenu. Veb sajtu se pristupa putem veb brauzer-a na određenoj URL ili IP adresi.

1.1.7 Dinamički veb sajt

Dinamički veb sajt prikazuje sadržaj u skladu sa ostvarenom interakcijom sa korisnikom, kao i sa ulogom i ovlašćenjima koja korisnik ima u aplikaciji. Veb aplikacije koje pokreću ovakve veb sajtove se nalaze na veb serveru i obično su pisane u skriptnom programskom jeziku kao što je PHP. Podaci se čuvaju u bazi podataka, odakle se po potrebi očitavaju i obrađuju.

1.2 OSNOVE HTTP PROTOKOLA

Protokoli su tehnička pravila i procedure na kojima se zasniva međusobna komunikacija umreženih računara. Da bi dva računara mogli međusobno da komuniciraju, moraju koristiti iste protokole.

HTTP (engl. *HyperText Transfer Protocol*) je mrežni protokol koji predstavlja najčešći način kojim se informacije prenose na Internetu. Koristi se za komunikaciju između servera i klijenta po principu zahtev/odgovor, a osnovna namena je isporuka HTML dokumenata.

Korisnik veb aplikacije inicira prenos podataka unosom željene adrese u adresnu liniju veb brauzera (HTTP klijent). Veb brauzer potom uspostavlja TCP/IP vezu sa veb serverom, koji predstavlja aplikacija koja osluškuje zahteve na određenom mrežnom komunikacijskom portu. Nakon što veb server odgovori na poslati zahtev, komunikacija se prekida sve do prijema sledećeg zahteva.

1.2.1 HTTP metode

HTTP podržava više metoda putem kojih je moguće inicirati zahtev, sve u skladu sa akcijom, odnosno efektom koji se želi postići. Najčešće se koriste sledeće metode:

- *GET* - server klijentu dostavlja resurse koji se nalaze na izabranoj URI adresi.
- *POST* - šalje podatke, smeštene u telo zahteva, ka serveru.
- *HEAD* - traži samo zaglavlje, bez tela zahteva.
- *PUT* - postavlja informaciju na navedenu lokaciju
- *DELETE* - briše označeni resurs na veb serveru.
- *CONNECT* - uspostavlja vezu, ali ne vrši interakciju.
- *OPTIONS* - vraća spisak podržanih metoda za traženu adresu.
- *TRACE* - provera putanje do cilja i identifikacija potencijalnih kritičnih tačaka.
- *PATCH* - modifikuje informaciju na navedenoj lokaciji.

1.2.2 HTTP zahtev

HTTP zahtev čine zaglavlje (engl. *header*), slika 2, prazna linija i telo zahteva. Zaglavlje sadrži tekst koji opisuje prirodu zahteva:

- Tip zahteva (GET ili POST)
- Naziv traženog dokumenta
- HTML verzija

Slika 2 - HTTP zaglavlje (engl. *header*)

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Host: www.citroen.rs
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
            (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36
```

Osim navedenih, u zaglavlju zahteva se mogu naći i opcioni podaci:

- Korisnički agent (engl. *User-agent*)
- Dužina sadržaja (engl. *Content-Length*)
- Tip dokumenta - MIME tip (tip medija) koji označava prirodu i format dokumenta, i karakter set.

Slika 3 - HTTP odgovor veb servera

```
http_code: HTTP/1.1 200 OK
Date: Fri, 21 Sep 2018 15:00:38 GMT
Last-Modified: Fri, 21 Sep 2018 13:28:40 GMT
Content-Length: 103932
Cache-Control: max-age=2592000, public
Content-Type: application/pdf
Expires: Sun, 21 Oct 2018 15:00:38 GMT
x-url: /wp-content/uploads/2018/06/2018-09-21-C4-SpaceTourer-Grand.pdf
Age: 262915
X-Cache: HIT
X-Cache-Hits: 33
Accept-Ranges: bytes
Strict-Transport-Security: max-age=0;
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
```

1.2.3 Odgovor veb servera

Odgovor veb servera sadrži status zahteva i zaglavlje, koje je iste strukture kao zaglavlje zahteva. Status zahteva možemo podeliti u sledeće grupe:

- *1XX* - Informacije.
- *2XX* - Uspešno.
- *3XX* - Redirekcija.
- *4XX* - Greška na klijentskoj strani.
- *5XX* - Greška na serveru.

Kodovi statusa zahteva koji se najčešće susreću u praksi su:

Grupa 200

- 200 - Uspešan zahtev.
- 201 - Kreirano.
- 202 - Prihvaćeno.
- 204 - Zahtev primljen, obrada još nije završena.

Grupa 300

- 301 - Adresa trajno promenjena.
- 302 - Privremena promena adrese. Zamenjen sa 303 i 307.
- 303 - Odgovor na zahtev se nalazi na drugoj URI adresi.
- 307 - Privremena redirekcija.
- 308 - Ponoviti zahtev i sve buduće zahteve koristeći novu adresu.

Grupa 400

- 400 - Loš zahtev.
- 401 - Neautorizovan.
- 403 - Zabranjen.
- 404 - Stranica nije pronađena.

Grupa 500

- 500 - Interna greška na serveru.
- 501 - Server ne prepoznaje zahtev ili ne može da ga izvrši.
- 503 - Server trenutno nije dostupan.

Listing 2 - Primer stilizovanja elementa na veb stranici primenom CSS koda

```
<html>
<head>
    <style>
        body {
            font-family: Arial;
        }
        .okvir {
            margin: 1em;
            padding: 1em;
            display: inline-block;
            border: 2px dotted red;
        }
    </style>
</head>
<body>
    <div class="okvir">
        Univerzitet Singidunum
    </div>
</body>
</html>
```

Slika 4 - Izgled elementa na ekranu

1.3 CSS

CSS (engl. *Cascading Style Sheets*) predstavlja jezik koji se koristi za stilizovanje prikaza jednog ili više elemenata koji se nalaze na veb stranici, ili veb stranice u celini. CSS sintaksa se sastoji iz dva elementa:

- selektor ciljanih elemenata koji može biti ime HTML taga, id ili ime elementa, ili naziv odgovarajuće klase
- opis izgleda, odnosno ponašanja elementa koji se navodi između vitičastih zagrade.

Opis izgleda odnosno ponašanja elementa se navodi kao niz svojstava i vrednosti, kao u primeru na listingu 2, rezultat je prikazan na slici 4.

1.4 JAVASCRIPT

JavaScript je skriptni jezik koji se koristi na klijentskoj strani radi kontrole interakcije korisnika aplikacije sa veb stranicom. Da bi funkcionisao, potrebni su veb brauzer i veb stranica sa odgovarajućim JavaScript kodom.

Za razvoj JavaScript aplikacija je dovoljan jednostavan editor koda. Ne treba ga mešati sa Java programskim jezikom, a sintaksa je slična sintaksi drugih programskih jezika poput PHP, C i sl.

1.5 MYSQL / MARIADB

SQL (engl. *Structured Query Language*) je standardni, veoma popularan deklarativni programski jezik koji se koristi za rad sa relacionim bazama podataka. Osnovne operacije za rad sa podacima, tzv. *CRUD*, su:

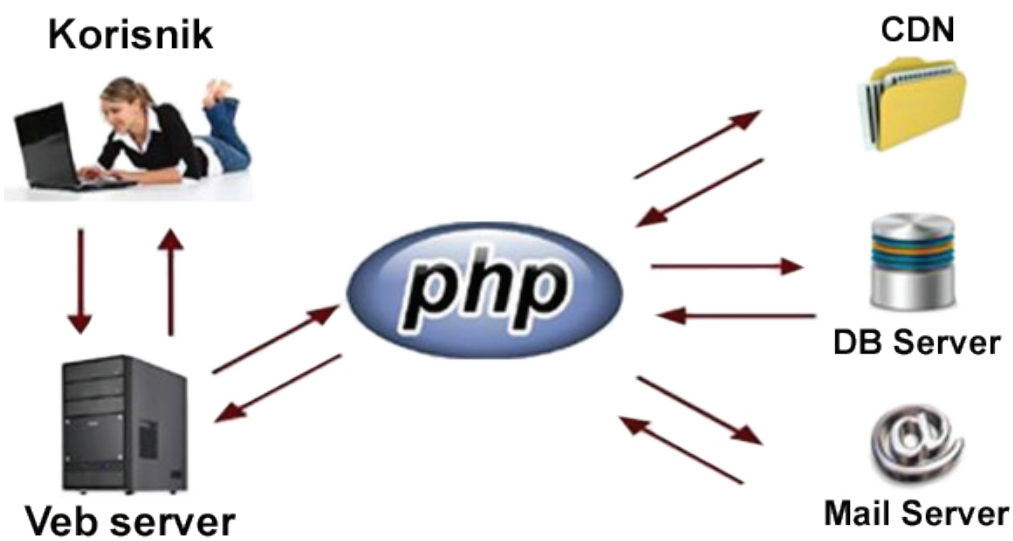
- Kreiranje (engl. *Create*)
- Čitanje (engl. *Read*)
- Izmena (engl. *Update*)
- Brisanje (engl. *Delete*)

SQL ekstenzije, ili dijalekti, kao što su MySQL, MariaDB, PostgreSQL, Oracle i sl. dodaju elemente proceduralnog programskog jezika. Osim toga, ekstenzije u mnogim programskim jezicima omogućavaju primenu tehnika objektno-orijentisanog programiranja u radu sa SQL serverom.

1.6 PHP PROGRAMSKI JEZIK

PHP[3] programski jezik je skriptni jezik koji se izvršava na strani servera. Napisan je u C programskom jeziku, omogućava rad sa popularnim serverima baza podataka i otvorenog je koda. Prvenstveno je namenjen razvoju veb aplikacija, ali se može koristiti i u druge svrhe. Rezultat izvršenog PHP koda može biti dokument bilo kog tipa, ali se najčešće koristi za generisanje dinamičkog veb sadržaja.

Slika 5 - PHP veb aplikacija: korelacija između komponenti sistema[4]



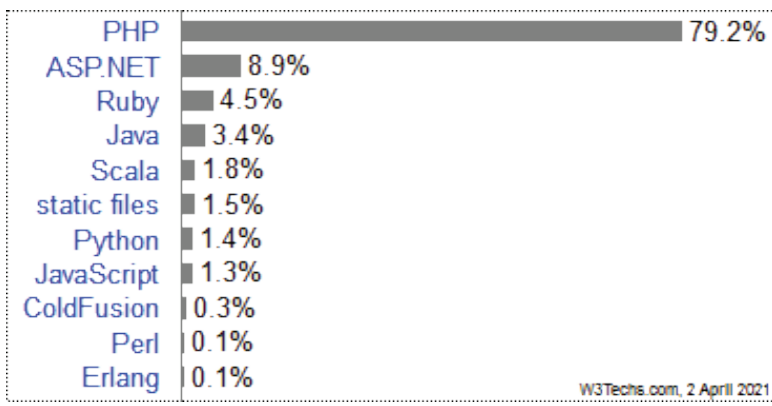
Odlikuje se relativno jednostavnom sintaksom koja je najsličnija sintaksi programskog jezika C. Omogućava i proceduralno i objektno orijentisano programiranje (OOP). Standardni PHP interpreter se distribuira kao besplatni softver pod PHP licencom, i dostupan je za različite operativne sisteme. PHP je modularan, a osnovni set funkcionalnosti se može proširivati dodavanjem odgovarajućih ekstenzija, slika 6.

Slika 6 - Konfigurisanje ekstenzija u php.ini konfiguracionom dokumentu

```
889 extension=gettext
890 ;extension=gmp
891 ;extension=intl
892 ;extension=imap
893 ;extension=interbase
894 ;extension=ldap
895 extension=mbstring
896 extension=exif      ; Must be after mbstring as it depends on it
897 extension=mysqli
898 ;extension=oci8_12c  ; Use with Oracle Database 12c Instant Client
899 ;extension=odbc
```

U 2021. godini, PHP programski jezik koristi ~79% veb sajtova u svetu, slika 7, među kojima su

Slika 7 - Upotreba programskih jezika na strani servera, april 2021.[5]



5 https://w3techs.com/technologies/overview/programming_language

- Facebook
- Wikipedia
- Yahoo!
- Flickr
- Tumblr
- Slack

PHP/FI[6] je inicijalno za lične potrebe 1994. godine napisao danski programer Rasmus Lerdorf u Perl programskom jeziku. Projekat je potom prebačen u C programski jezik, omogućena je komunikacija sa bazama podataka, a 1998. godine je objavljen PHP 3.0, verzija koju su razvili izraelski programeri Zeev Suraski i Andi Gutmans u saradnji sa Lerdorfom. PHP 4 je bio prva generacija ovog programskog jezika u čijoj je osnovi Zend Engine. 2004. godine je objavljen PHP 5, koji je doneo OOP i PDO[7] za komunikaciju sa bazom podataka. Sledeća verzija, PHP 6, je trebalo da reši dugogodišnje odsustvo podrške za Unicode, ali je zbog velikih kašnjenja i nerazumevanja zajednice ovaj poduhvat napušten, a verzija 6 nikada zvanično nije objavljena. Trenutno još uvek aktuelna, sedma generacija PHP programskog jezika je objavljena 2015. godine i donela je značajnu optimizaciju u performansama PHP aplikacije, dok su određene ekstenzije napuštene, uključujući i MySQL ekstenziju. Krajem 2020. godine objavljena je prva iteracija osme generacije PHP programskog jezika.

Prema statističkim procenama iz septembra 2020. godine, dve trećine veb sajtova koji koriste PHP su bazirani na starim verzijama ovog programskog jezika, prvenstveno 5.2-5.6. Ukoliko se uzme u obzir prestanak podrške za najpopularniju aktivnu verziju 7.2, 84% PHP veb sajtova u svetu u ovom trenutku koristi zastarele verzije PHP programskog jezika.

6 PHP - Personal Home Page Tools
7 PDO - PHP Data Objects

1.7 XAMPP, INSTALACIJA I POKRETANJE KOMPONENTI

XAMPP je besplatni softverski paket koji omogućava instalaciju i kontrolu veb servera na lokalnom računaru, a sadrži sledeće glavne komponente:

- Apache HTTP server
- MariaDB server baze podataka
- PHP programski jezik
- Perl programski jezik

Osim navedenih, u ovom paketu se nalaze i:

- Serveri
  - *FileZilla FTP* server koji se koristi za transfer dokumenata preko FTP protokola.
  - *Mercury Mail Server,* koji podržava glavne protokole vezane za sistem elektronske pošte, kao što su SMTP, POP3 i IMAP.
  - *Apache Tomcat Java Server* je Java HTTP veb server preko kojeg se može pokrenuti Java kod.
- Aplikacije
  - *phpMyAdmin* je veb aplikacija koja predstavlja online interfejs za menadžment baze podataka.
  - *Webalizer* je aplikacija pomoću koje može da se analizira ostvareni saobraćaj na veb sajtu.
  - *Fake SendMail* je aplikacija koja simulira slanje mail poruka.

Slika 8 - Preuzimanje XAMPP softverskog paketa, *apachefriends.org*

1.7.1 INSTALACIJA

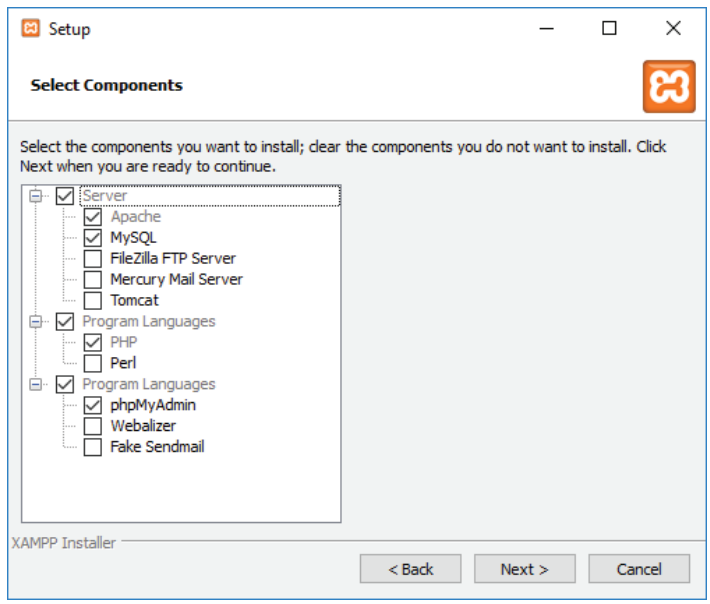
XAMPP se može preuzeti sa adrese *apachefriends.org*, slika 8, i dostupan je za *Windows, Linux* i *OS X* operativne sisteme, u izvornom obliku ili kao instaler aplikacija.

Za razvoj PHP veb aplikacija, neophodno je da se instaliraju Apache HTTP server, MySQL server baze podataka, PHP programski jezik i phpMyAdmin veb aplikacija za menadžment baze podataka. Sam proces instalacije je jednostavan, brz i završava se sa svega nekoliko klikova mišem.

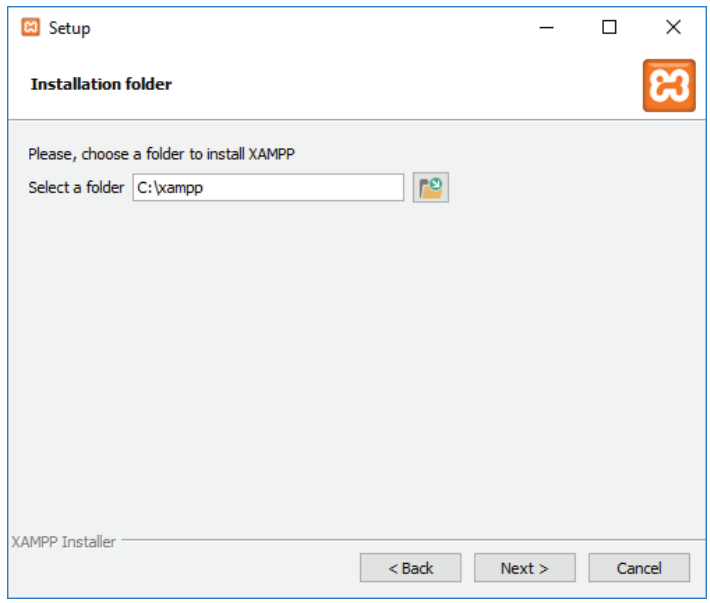
Prilikom izbora imena direktorijuma u koji će izabrane komponente XAMPP paketa biti instalirane, preporuka je da se ime izabere na takav način da ne sadrži razmake. U nastavku teksta, podrazumevaće se da su komponente instalirane u direktorijum c:\xampp.

- *c:\xampp* korektno ime direktorijuma
- *c:\xampp72* korektno ime direktorijuma
- *c:\xampp 7.2* izbegavati imena direktorijuma sa razmacima i karakterima koji nisu alfanumerički

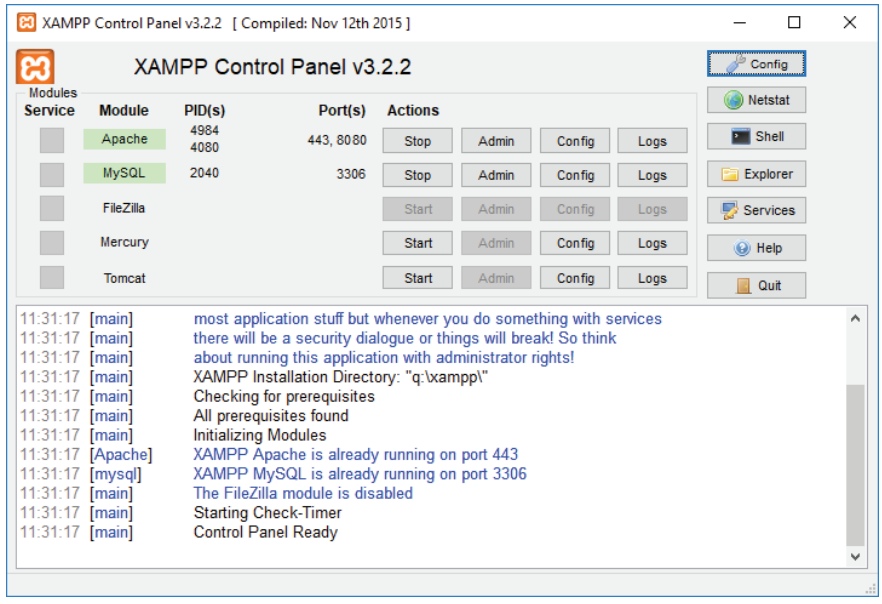
Slika 9 - Komponente XAMPP paketa koje su neophodne za razvoj PHP veb aplikacija



Slika 10 - Izbor imena direktorijuma u koji će komponente XAMPP-a biti instalirane



Slika 11 - XAMPP kontrolni panel za individualnu kontrolu instaliranih komponenti

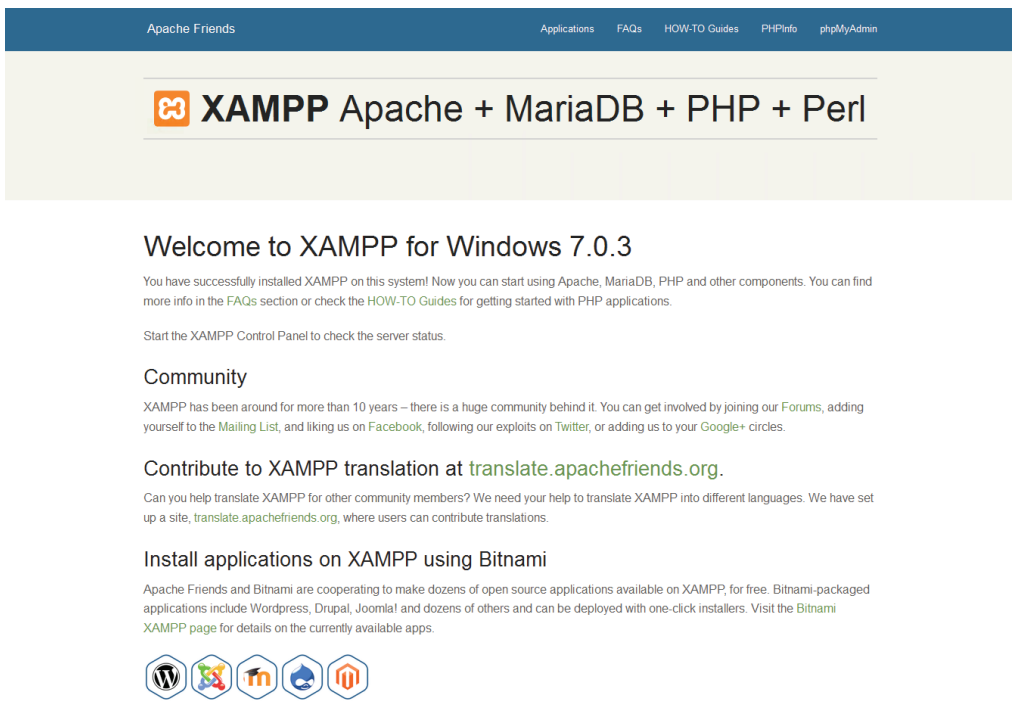


Nakon instalacije, komponentama XAMPP-a se može upravljati preko XAMPP kontrolnog panela, slika 11, ili direktnim pristupom odgovarajućim fajlovima u instalacionom folderu, tabela 3.

Tabela 3 - Kontrola komponenti XAMPP paketa putem komandne linije u Windows OS

c:\xampp\xampp_start.exe	pokreće Apache HTTP server i MySQL server
c:\xampp\xampp_stop.exe	zaustavlja Apache HTTP server i MySQL server
c:\xampp\xampp-control.exe	omogućava individualnu kontrolu instaliranih komponenti XAMPP paketa

Slika 12 - XAMPP dashboard



Po završetku procesa instalacije i pokretanja komponenti, inicijalni pristup veb serveru se može ostvariti putem veb brauzera na adresi *http://localhost*, pri čemu će biti prikazana stranica XAMPP dashboard kao znak da je instalacija uspešno izvršena i da HTTP server ispravno radi.

1.7.2 Konfiguracija platforme

Informacije o konfiguraciji komponenti XAMPP paketa se nalaze u odgovarajućim konfiguracionim fajlovima. Reč je o dokumentima čijem se sadržaju može pristupiti pomoću običnog tekst editora, i koji se nalaze na sledećim lokacijama:

Windows operativni sistem

Apache HTTP server

```
C:\xampp\apache\conf\httpd.conf
```

```
C:\xampp\apache\conf\extra\httpd-vhosts.conf
```

PHP

```
C:\xampp\php\php.ini
```

MySQL

```
C:\xampp\mysql\bin\my.ini
```

Linux operativni sistem

Apache HTTP server

```
/opt/lampp/etc/httpd.conf
```

```
/opt/lampp/etc/extra/httpd-vhosts.conf
```

PHP

```
/opt/lampp/etc/php.ini
```

MySQL

```
/opt/lampp/etc/my.cnf
```

1.8 PODEŠAVANJA APACHE VEB SERVERA

1.8.1 Promena lokacije u kojoj se nalaze veb dokumenti

Parametar *DocumentRoot* koji se nalazi u Apache konfiguracionom fajlu httpd.conf ukazuje na direktorijum u kojem se nalaze veb dokumenti. Predefinisana vrednost je

```
DocumentRoot "C:/xampp/htdocs"
```

Ukoliko želimo da promenimo ovaj direktorijum, dovoljno je da se samo izmeni odgovarajuća vrednost:

```
DocumentRoot "C:/www"
```

Napravljenu izmenu je potom potrebno sačuvati i restartovati Apache HTTP server.

1.8.2 Izmena porta na koji se veb server odaziva

Port na koji se Apache HTTP server inicijalno odaziva je port 80, koji je ujedno i predefinisani HTTP port i ne mora se pisati prilikom formiranja URL adrese.

Ukoliko se javi potreba da se ovaj port promeni, potrebno je izmeniti odgovarajuću vrednost koja se u *httpd.conf* konfiguracionom fajlu nalazi uz parametar Listen, npr.

```
Listen 8080
```

Izmenu je potrebno sačuvati, a Apache HTTP server restartovati. Nakon toga, server će osluškivati samo URL zahteve upućene na port 8080:

```
http://localhost:8080
```

U nastavku teksta, podrazumevaće se da je Apache veb server podešen da odgovara samo na zahteve upućene preko porta 8080.

Slika 13 - Set pravila za osnovni direktorijum (DocumentsRoot), httpd.conf

```
<Directory "c:/www">
    Options Indexes FollowSymLinks Includes ExecCGI
    AllowOverride All
    Require all granted
</Directory>
```

1.8.3 Bezbednost

Parametri pristupa za svaki direktorijum kome Apache veb server ima pristup mogu biti zasebno podešeni, uključujući servise i funkcije veb servera koje će u svakom od njih biti dozvoljene ili zabranjene. Prvo se navode osnovna pravila koja treba da budu što restriktivnija.

Slika 14 - Zabrana pristupa fajlovima.*htaccess* i *.htpasswd,* dokument *httpd.conf*

```
<Files ".ht*">
    Require all denied
</Files>
```

1.8.3.1 .htaccess

Dokument *.htaccess* definiše pravila pristupa za direktorijum u kome se nalazi i sve direktorijume višeg reda koji ne sadrže svoj dokument *.htaccess*. Tako deklarisana pravila zamenjuju pravila definisana u dokumentu *httpd.conf*. Koristan je za autorizaciju, izmenu veb adresa, blokiranje pojedinačnih IP adresa ili celih IP opsega, izmenu stranica koje se prikazuju u slučaju greške (npr. greške 403 i 404) i sl.

Prednost upotrebe .htaccess dokumenta se ogleda u tome što su unete izmene odmah aktivne, zato što se čita pri svakom HTTP zahtevu.

Slika 15 - Primer parova korisničko ime i hešovana lozinka (MD5)

```
jelena:n5mfKoHOIQkKg
ana:9fluR/2n84p4c
```

Slika 16 - Obavezna referenca ka *.htpasswd* u dokumentu *.htaccess*

```
AuthUserFile /usr/uj/jurbanek/.htpasswd
AuthType Basic
AuthName "My Files"
Require valid-user
```

1.8.3.2 .htpasswd

Dokument *.htpasswd* se koristi za kontrolu pristupa određenom direktorijumu. U njemu se nalaze parovi korisničko ime:lozinka, pri čemu su ovi parametri razdvojeni dvotačkom i u svakom redu sa nalazi samo jedan par. Lozinka je hešovana *MD5* algoritmom.

1.8.4 Semantičke veb adrese

Tehnika modifikacije veb adresa (engl. *User-friendly URLs*) se koristi za kontrolu ponašanja veb servera. Razmotrimo sledeće dve adrese, koje se zapravo odnose na istu veb stranicu:

```
https://www.singidunum.ac.rs/components/vest.php?nid=4325
https://www.singidunum.ac.rs/vesti/studenti/upis-2020-4325
```

U ovom primeru, prva adresa je karakteristična za aplikaciju u kojoj se moduli nalaze u različitim direktorijumima, a ulazna informacija za modul su URL parametri (parametar *nid*, vrednost *4325*).

Druga adresa je tzv. semantička adresa, u kojoj se željeni segment aplikacije poziva nizom ključnih reči. Važno je razumeti da struktura ovakvog URL-a ne označava realnu strukturu direktorijuma (*/vesti/studenti*), niti poziva konkretan dokument (*upis-2020-4325*). Ovakva tehnika je pogodna za upotrebu iz sledećih razloga:

- Omogućava plasiranje željenih ključnih reči u URL, čime pozitivno deluje na rangiranje veb stranice u rezultatima pretrage veb pretraživača.
- Sakriva realnu strukturu fajlova

Primenu semantičkih adresa omogućava Apache modul *mod_rewrite,* koji prethodno mora biti aktiviran u *httpd.conf* konfiguracionom fajlu:

```
LoadModule rewrite_module modules/mod_rewrite.so
```

1.8.5 Ostala podešavanja veb servera

Ime servera i port koje veb server koristi za samoidentifikaciju se podešavaju kao

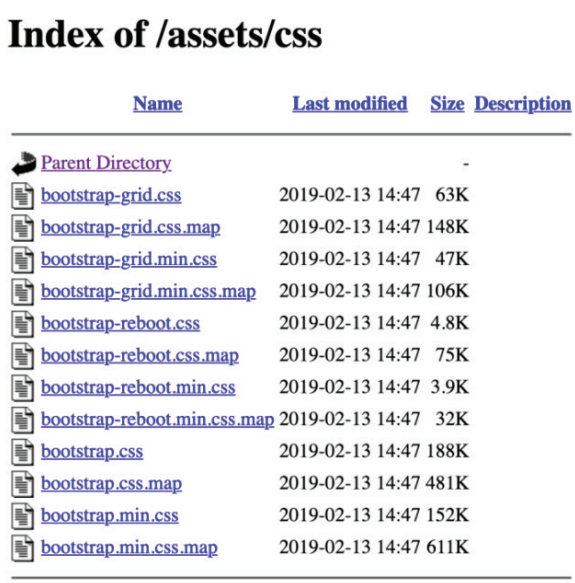
```
ServerName localhost:8080
```

Ime fajla koji će Apache veb server aktivirati u slučaju poziva u kome nije navedeno ime dokumenta se reguliše kao

```
DirectoryIndex index.php index.html index.htm
```

što znači da će u opisanoj situaciji veb server pokušati u datom direktorijumu da pronađe izlistane fajlove, datim redosledom. Time se sprečava prikaz listinga fajlova koji se nalaze u direktorijumu, slika 17.

Slika 17 - Apache veb server, listing direktorijuma kada nije navedeno ime dokumenta



Osnovni tip dokumenta (*mime tip*), ukoliko veb server nije u stanju da ga samostalno odredi, podešava se kao

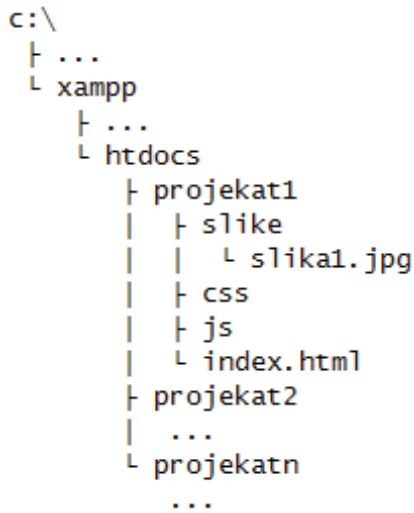
```
DefaultType text/plain
```

1.8.6 Podešavanje virtuelnog domena

Kada govorimo o razvoju veb aplikacija u realnim uslovima, velika je verovatnoća da će se na jednom računaru istovremeno nalaziti više projekata, a time se postavlja i pitanje organizacije pristupa pojedinačnim projektima.

Pristup različitim projektima preko localhost-a je moguć, ali u određenim situacijama može prouzrokovati probleme u radu aplikacija. Pretpostavimo sledeću strukturu direktorijuma sa projektima:

Slika 18 - Struktura veb projekata u htdocs direktorijumu



U veb brauzeru, dokument *slika1.jpg* koji se nalazi u okviru *projekta 1* se može pozvati preko sledeće adrese:

```
http://localhost:8080/projekat1/slike/slika1.jpg
```

Međutim, ukoliko se u dokumentu *index.html* napravi referenca ka istoj slici, a umesto apsolutne putanje stavi relativna putanja

```
<img src="/slike/slika1.jpg">
```

veb brauzer će ovu putanju protumačiti kao

```
http://localhost:8080/slike/slika1.jpg
```

i proizvesti grešku 404 koja označava da traženi dokument ne postoji. Osim toga, uvek postoji mogućnost da se struktura direktorijuma u produkcionoj verziji donekle razlikuje u odnosu na razvojnu verziju, tako da se upotreba apsolutnih putanja ka dokumentima ne preporučuje.

Kao koristan alat u izbegavanju ovakvih problema može da se iskoristi virtuelni hosting za koji Apache veb server ima podršku, i omogućava da jedan veb server na jednom računaru opslužuje više veb sajtova ili aplikacija. U tom smislu, svakom projektu, ili po potrebi direktorijumu se može dodeliti virtuelni domen, a da bi se to učinilo, dovoljno je modifikovati sledeća dva dokumenta:

- hosts fajl operativnog sistema, koji se nalazi na sledećoj putanji:

  Windows c:\windows\system32\drivers\etc

  Linux /etc/

- Apache *httpd-vhosts.conf* fajl

Informacija o virtuelnom domenu se u hosts fajl ubacuje na sledeći način:

```
127.0.0.1 veb-programiranje
```

gde IP adresa 127.0.0.1 označava localhost, dok veb-programiranje označava virtuelni domen koji se kreira na lokalnom računaru.

Apache konfiguracioni fajl httpd-vhosts.conf sadrži blokove za željene virtuelne domene, svaki sa osnovnim podešavanjima, slika 19.

Slika 19 - Definicija virtuelnog domena, Apache dokument *httpd-vhosts.conf*

```
<VirtualHost *:8080>
    DocumentRoot "C:/www/veb-programiranje"
    ServerName veb-programiranje
    ErrorLog "C:/www/veb-programiranje/error.log"
    CustomLog "C:/www/veb-programiranje/access.log" common
</VirtualHost>
```

- VirtualHost tag označava blok u kome se nalaze podešavanja za određeni virtuelni domen. Broj *8080* označava port na kome se osluškuju pozivi i u skladu je sa podešavanjem parametra *Listen* u Apache konfiguracionom fajlu *httpd.conf*.

- *ServerName*, naziv virtuelnog domena.
- *DocumentRoot*, direktorijum u kome se nalaze veb dokumenti koji pripadaju virtuelnom domenu.
- *ErrorLog*, lokacija i ime fajla u kojem će veb server beležiti greške.
- *CustomLog*, lokacija i ime fajla u kojem će veb server beležiti podatke o pristupu virtuelnom domenu.

Nakon što se završi sa modifikacijom konfiguracionog fajla *httpd-vhosts.conf,* unete izmene je potrebno sačuvati, a potom i restartovati veb server.

PHP PROGRAMSKI JEZIK

2. PHP PROGRAMSKI JEZIK

2.1 UVOD U SINTAKSU

Sintaksa PHP programskog jezika je u osnovi slična sintaksi programskog jezika C. PHP kod ne zahteva kompajliranje, već se nalazi i interpretira na veb serveru prilikom svakog poziva aplikacije.

Listing 3 - Primer PHP dokumenta

```
<?php
    print "Univerzitet Singidunum";
?>
```

Da bi pravilno bio interpretiran, mora biti označen početnim tagom *<?php* i završnim tagom *?>*, listing 3, ili u skraćenom pisanju kao *<?= i ?>*[8]. Ekstenzija imena fajla u kojem se nalazi mora biti ".php" U okviru PHP dokumenta se može nalaziti HTML kod, listing 4, kao i CSS i JavaScript.

Listing 4 - PHP i HTML kod u istom dokumentu

```
<html>
<head>
</head>
<body>
    <div>
        &copy;<?= date('Y', time()); ?>
        Univerzitet Singidunum
    </div>
</body>
</html>
```

8 Ekvivalent za <?php echo $a; ?>

Naredbe u PHP kodu se međusobno razdvajaju znakom ";" (tačka-zarez), dok se razmaci, tabovi i prelomi linija ignorišu, što programeru daje slobodu da stilizuje kod. Komentari mogu da se pišu u jednoj liniji kada se označavaju sa "//" (dupla kosa crta), ili u više linija kada se početak komentara označava sa "/*", a kraj sa "*/". Blokovi koda se označavaju vitičastim zagradama *{...}*.

PHP kod je moguće grupisati u funkcije i klase, a dalja segmentacija je moguća kroz sistem dokumenata aplikacije. Ukoliko se prilikom poziva aplikacije navede samo domen, bez imena fajla, npr.

```
https://www.singidunum.ac.rs
```

veb server će pokušati da pokrene dokument *index.php*, v. poglavlje 1.8.5.

Imena funkcija, klasa i ključnih reči se po želji mogu pisati velikim ili malim slovima (engl. *case-insensitive*), listing 5.

Listing 5 - Imena funkcija nisu osetljiva na velika i mala slova

```php
<?php
    ECHO "Univerzitet Singidunum<br>";
    echo "Univerzitet Singidunum<br>";
    EcHo "Univerzitet Singidunum<br>";
?>
```

2.2 TIPOVI PODATAKA

U PHP programskom jeziku podržan je rad sa podacima sledećih tipova:

- *Integer* je celobrojna vrednost koja može biti u opsegu od 2147483648 do 2147483647. Može se deklarisati u decimalnom, heksadecimalnom, oktalnom ili binarnom zapisu.
- *Float* je decimalni broj ili broj u eksponencijalnom zapisu.
- *Boolean* može imati vrednost TRUE (istinito) ili FALSE (nije istinito).
- *String* predstavlja niz karaktera, npr. "Zdravo!".
- *Array* je niz, odnosno matrica i omogućava skladištenje više vrednosti u jednoj promenljivoj.
- *Object* je objekat, odnosno instanca klase.
- *NULL* označava da promenljivoj nije dodeljena vrednost
- *Resource* ne predstavlja konkretan tip podataka, već je referenca u formi stringa ka funkcijama i drugim resursima koje PHP koristi u toku izvršavanja aplikacije.

2.3 PROMENLJIVE

Ime promenljive u PHP programskom jeziku počinje znakom "$" (dolar), nakon čega se navodi ime promenljive:

```
$boja
```

Ime promenljive može da sadrži samo alfanumeričke znakove (a-z, 0-9) i znak "_" (donja crta), pri čemu prvi karakter u imenu promenljive mora biti slovo (a-z) ili znak "_" (donja crta). Imena promenljivih su osetljiva na veličinu slova, tako da će PHP imena $boja i $BOJA tretirati kao dve nezavisne promenljive.

- *$1singidunum* - nepravilno, ime promenljive počinje brojem
- *$_1singidunum* - pravilno, ime promenljive počinje donjom crtom

PHP nema posebnu naredbu za deklarisanje promenljivih, tako da se nova promenljiva može uvesti u bilo kojoj tački programa. Takođe, promenljive nisu tipizirane, a PHP će automatski konvertovati tip u skladu sa sadržajem promenljive (celobrojna, sa pokretnim zarezom, string, matrica itd).

Iako PHP dozvoljava različite stilove pisanja koda, primer dobre prakse je da se imena promenljivih pišu malim slovima, a reči u imenu razdvajaju znakom "_".

- *$univerzitet_singidunum* - ispravno, lako za čitanje
- *$UniverzitetSingidunum* - ispravno, lako za čitanje, u slučaju da se izostave velika slova PHP tumači kao novu promenljivu.
- *$univerzitetsingidunum* - ispravno, teško za čitanje.
- *$Univerzitet-Singidunum* - neispravno, u imenu promenljive je upotrebljen znak "-" (srednja crta).

2.3.1 Vidljivost promenljivih

Područje validnosti promenljivih može biti

- Lokalno (*Local*)
- Globalno (*Global*)
- Statičko (*Static*)

Promenljiva deklarisana van funkcije, u glavnom toku programa, je globalna promenljiva (engl. *Global Scope*). Promenljiva deklarisana unutar funkcije je dostupna samo u toj funkciji (engl. *Local Scope*). Kada se funkcija izvrši, njene lokalne promenljive se brišu.

Listing 6 - Upotreba promenljivih u globalnom i lokalnom opsegu

```php
<?php
    $x=5;
    $y=10;

    function proba()
    {
        global $x, $y;
        $y=$x+$y;
    }

    proba();
    echo $y;
?>
```

U primeru prikazanom na listingu 6, promenljive *$x* i *$y* su deklarisane u glavnom toku programa, odnosno u globalnom opsegu. Da bi bile dostupne u funkciji *proba()*, neophodno je da se upotrebi ključna reč *global*. Kao krajnji rezultat rada ovog programa, promenljiva *$y* ima vrednost 15.

Listing 7 - Primena statičkih promenljivih

```
<?php

function proba()
{
    static $x=0;
    echo $x;
    $x++;
}

proba();
proba();
proba();

?>
```

Ukoliko je potrebno da lokalna promenljiva zadrži vrednost koju je imala prilikom poslednjeg poziva funkcije, potrebno je da se deklariše kao *Static*.

U primeru prikazanom na listingu 7, prilikom prvog poziva funkcije *proba()* inicijalizuje se statička promenljiva *$x* sa vrednošću 0. Potom se ispisuje njena vrednost i uvećava za 1. Prilikom svakog sledećeg poziva ove funkcije promenljiva *$x* ne dobija vrednost 0, već zadržava vrednost izračunatu u prethodnom prolazu.

2.3.2 Superglobalne promenljive

PHP poseduje predefinisane promenljive koje su dostupne u svim područjima validnosti, nazivaju se superglobalne promenljive, i ne zahtevaju upotrebu ključne reči *global* u funkciji ili metodu da bi im se pristupilo. Superglobalne promenljive su po svojoj prirodi asocijativni nizovi:

- *$GLOBALS* - sadrži informaciju o svim promenljivim koje su u datom trenutku deklarisane u globalnom opsegu.
- *$_GET* - sadrži informaciju o URL parametrima i njihovim vrednostima
- *$_POST* - sadrži informaciju o podacima poslatim HTTP POST metodom. Ključevi u nizu su imena elemenata u veb obrascu.
- *$_FILES* - sadrži informaciju o dokumentima prebačenim na veb server upotrebom HTTP POST metode.
- *$_SERVER* - sadrži informacije o zaglavljima zahteva, putanjama, i lokacijama skripte. Podatke u ovom nizu daje veb server, a koji će se sve podaci naći zavisi od tipa veb servera i njegove konfiguracije.
- *$_COOKIE* - sadrži informacije koje se nalaze u kukiju.
- *$_SESSION* - sadrži informacije o sesiji koje su dostupne pokrenutom programu.
- *$_REQUEST* - sadrži kumulativne informacije koje se nalaze u superglobalnim promenljivim *$_GET, $_POST i $_COOKIE*.
- *$_ENV* - sadrži podatke o promenljivim u okruženju pod kojim radi PHP interpreter.

2.4 KONSTANTE

Konstanta predstavlja ime ili identifikator za određenu vrednost. Nakon što je definisana, njena vrednost se više ne može promeniti tokom izvršavanja programa.

- define('BEOGRAD', 1, true);
- define('NOVI_SAD', 2);
- define(NIS', 3);

Iako treći parametar funkcije *define* ukazuje na to da li će ime konstante biti osetljivo na velika i mala slova, u cilju lakšeg tumačenja koda imena konstanti u PHP programskom jeziku se prema nepisanom pravilu pišu velikim slovima.

2.5 OPERATORI

Operatori u PHP programskom jeziku se koriste za izvođenje osnovnih računskih operacija, za dodelu, promenu ili poređenje vrednosti, kao i za logičke operacije.

2.5.1 Osnovne računske operacije, aritmetički operatori

U osnovne računske operacije se ubrajaju sabiranje, oduzimanje, množenje i deljenje. Osim toga, u ovu grupu se mogu uvrstiti ostatak deljenja dva broja, kao i operacija stepenovanja.

- Sabiranje: *$x + $y*
- Oduzimanje: *$x - $y*
- Množenje: *$x * $y*
- Deljenje: *$x / $y*
- Ostatak deljenja *$x* sa *$y: $x % $y*
- Podizanje *$x* na *$y* stepen: *$x ** $y*

2.5.2 Operatori dodele

Operatori dodele se koriste za dodelu vrednosti promenljivoj.

- Dodela vrednosti promenljivoj *$x, $x = $y*
- Dodavanje vrednosti, *$x = $x + $y* ili skraćeno *$x += $y*
- Oduzimanje vrednosti, *$x = $x - $y* ili *$x -= $y*
- Množenje, *$x = $x * $y* ili *$x *= $y*
- Deljenje, *$x = $x / $y* ili *$x /= $y*
- Ostatak deljenja, *$x = $x % $y*, ili *$x %= $y*

2.5.3 Operatori umanjenja / uvećanja

Operatori umanjenja, odnosno uvećanja se koriste kada je promenljivoj potrebno umanjiti, odnosno uvećati vrednost za jedan.

- ++$x Uvećaj $x za 1 pa vrati $x
- $x++ Vrati $x pa uvećaj $x za 1
- --$x Smanji $x za 1 pa vrati $x
- $x-- Vrati $x pa smanji $x za 1

2.5.4 Operatori poređenja

Operatori poređenja se koriste u situacijama kada je potrebno međusobno uporediti dve vrednosti. Novi operator nazvan s*vemirski brod* (engl. s*paceship*) uveden je u PHP 7 i koristi se za proveru dve veličine tako što vraća celobrojnu vrednost 1, 0 ili -1 u skladu sa tim da li je prva veličina veća, jednaka ili manja od druge veličine.

- $x == $y Jednako po vrednosti
- $x === $y Identično po tipu i vrednosti
- $x != $y Nije jednako
- $x <> $y Nije jednako
- $x !== $y Nije identično
- $x > $y $x veće od $y
- $x < $y $x manje od $y

- $x >= $y Veće ili jednako
- $x <= $y Manje ili jednako
- $x <=> $y Međusobni odnos vrednosti *$x* i *$y*:

2.5.5 Logički operatori

Logički operatori se najčešće koriste za kontrolu toka programa, proverom tačnosti jednog ili više uslova.

- $x and $y Tačno ako su tačni i *$x* i *$y*
- $x or $y Tačno ako su tačni *$x* ili *$y*
- $x xor $y Tačno ako su tačni *$x* ili *$y*, ali ne oba
- $x && $y I (*and*)
- $x || $y Ili (*or*)
- ! Negacija (*not*)

2.5.6 Operatori uslovne dodele

Operatori uslovne dodele se koriste kada je potrebno dodeliti vrednost u skladu sa postavljenim uslovima.

```
$x = $y ?? 'vrednost';
```

Operator sjedinjavanja (engl. *coalescing operator*, ??) uveden u PHP 7.4 se može koristiti u situacijama kada je potrebno da se promenljivoj *$x* dodeli vrednost promenljive *$y*, ali postoji verovatnoća da promenljiva *$y* nije deklarisana, što je realan scenario npr. prilikom očitavanja URL parametara. U tom slučaju će se dodeliti predefinisana vrednost koja se nalazi iza operatora, i izbeći da PHP generiše obaveštenje (*E_NOTICE*) o upotrebi nedeklarisane promenljive.

```
$x = $a == $b ? 'vrednost1' : 'vrednost2';
```

Ternarni operator (*?:*) se još naziva jednolinijski if uslov (engl. *inline if, iif*). U prikazanom primeru, proverava se ispunjenost uslova *$a == $b*. Ukoliko je uslov ispunjen, promenljivoj *$x* će se dodeliti vrednost *vrednost1*, u suprotnom će joj se dodeliti vrednost *vrednost2*.

2.6 RAD SA STRINGOVIMA

Prilikom rada sa stringovima, potrebno je obratiti pažnju na navodnike koji se koriste u konkretnom slučaju:

- 'Jednostruki navodnici'
- "Dvostruki navodnici"

Sadržaj koji se nalazi unutar jednostrukih navodnika se tretira kako je napisan, dok dvostruki navodnici omogućavaju ubacivanje dinamičkih vrednosti u string, kao što je npr. vrednost ranije deklarisane promenljive.

Listing 8 - Upotreba jednostrukih i dvostrukih navodnika

```php
<?php
    $a = 'Singidunum';
    $b = 'Beograd';
    $c = $a . ', ' . $b;
    $d = "$a, $b";
    echo $c . '<br>';
    echo "$d<br>";
?>
```

Sadržaj stringa se nastavlja pomoću operatora ".=" (tačka i znak jednakosti), dok se sadržaj dva stringa spaja primenom operatora "." (tačka).

U primeru prikazanon na listingu 8, sadržaj promenljive *$d* se formira unutar dvostrukih navodnika gde su smeštene promenljive *$a* i *$b*, a rezultat će biti *Singidunum, Beograd*. U slučaju da su umesto dvostrukih upotrebljeni jednostruki navodnici (*$d* = '*$a, $b*';), sadržaj promenljive *$d* bi bio tekst *$a*, $b. U istom primeru je pokazano da se kao sadržaj stringa mogu navesti i HTML tagovi, koje će veb brauzer parsirati na adekvatan način.

Listing 9 - Matematičke operacije nad stringovima

```php
<?php
$a = '1';
$b = "2";
$c = 3;
echo $a + $b + $c;
echo "<br>";
echo 10 + 'Univerzitet Singidunum';
?>
```

Ovde je korisno obratiti pažnju i na to da ukoliko su promenljive deklarisane kao stringovi, ali sadrže npr. numeričke vrednosti, listing 9, PHP će te vrednosti tretirati kao numeričke u slučaju izvršenja matematičkih operacija. Važi i obrnuto, ukoliko se u matematičkoj operaciji nađe string (tekst), biće tretiran kao

- nula, ukoliko počinje znacima ('*Univerzitet Singidunum*' → *0*)
- broj, ukoliko počinje brojevima ('*2020 Univerzitet Singidunum*' → *2020*)

Listing 10 - Kontrola toka programa - *if-else* struktura

```php
<?php

if (uslov) {
    // kod koji se izvršava ako je uslov ispunjen;
}
else if (uslov #1) {
    // kod koji se izvršava ako je uslov #1 ispunjen;
}
else {
    // kod koji se izvršava ako nijedan od gornjih uslova nije ispunjen;
}

?>
```

2.7 GRANANJE

PHP dozvoljava kontrolu ispunjenosti posmatranih uslova primenom *if (...) else* strukture, listing 10. Pojedinačni uslovi se međusobno razdvajaju logičkim operatorima, npr. *if ($a == 1 && $b < 10) { ... }*.

Listing 11 - Kontrola toka programa - *switch blok*

```
<?php
switch (n) // n je izraz ili promenljiva
{
    case vrednost1 :
            /* kod koji se izvršava ako je uslov ispunjen; */
            break;
    case vrednost2 :
            /* kod koji se izvršava ako je uslov ispunjen; */
            break;
    ...
    default :
            /* kod koji se izvršava u slučaju da za konkretnu vrednost
               ne postoji case provera; */
}
```

Ukoliko je poznat skup mogućih vrednosti promenljive ili rezultata logičke operacije, kontrola vrednosti se može izvršiti primenom *switch* bloka, listing 11.

2.8 PETLJE

Kada je jednu ili više naredbi potrebno izvršiti određeni broj puta, koriste se tzv. petlje. PHP podržava različite oblike petlji, koji se koriste u skladu sa konkretnom situacijom:

Listing 12 - PHP *for* petlja

```
<?php
for ($i = 0; $i <= 10; $i++)
{
    ...
}
?>
```

for petlja ponavlja blok koda navedeni broj puta, listing 12.

Listing 13 - PHP *foreach* petlja

```
<?php
foreach($niz AS $indeks => $vrednost)
{
    ...
}
?>
```

while petlja ponavlja blok koda sve dok se ne ispuni zadati uslov, listing 14.

Listing 14 - PHP *while petlja*

```
<?php
    $x=1;
    while($x<=5)
    {
        echo "Broj: $x <br>"; $x++;
    }
?>
```

Listing 15 - PHP *do-while* petlja

```
<?php
    $x=1;
    do
    {
        echo "Broj: $x <br>"; $x++;
    }
    while ($x<=5)
?>
```

do-while petlja će izvršiti blok koda, a potom ga iznova ponavljati sve dok se ne ispuni zadati uslov, listing 15.

Listing 16 - Očitavanje trenutnog vremena na veb serveru

```
<?php
$vreme = microtime(1);
print $vreme;
?>
```

2.9 DATUM I VREME

Za očitavanje trenutnog vremena na veb serveru se može upotrebiti funkcija *time()*, listing 16, koja kao rezultat vraća trenutni *UNIX timestamp*, odnosno broj sekundi proteklih od ponoći 1. januara 1970. godine. Ukoliko je potrebno preciznije očitavanje, funkcija *microtime()* kao dodatni podatak vraća i informaciju o mikrosekundama.

```
echo time();                ► 1609026827
echo microtime();           ► 0.96587900 1609026827
echo microtime(TRUE);       ► 1609026827.7003
```

Listing 17 - Formatiranje UNIX vremena u ljudima prepoznatljiv format

```
<?php
$vreme = time();
echo date('d.m.Y H:i:s', $vreme);
?>

03.01.2021 18:36:31
```

Rezultat funkcije *microtime* je string ili broj sa pokretnim zarezom, u skladu sa tim da li je opcioni parametar podešen na TRUE.

Tabela 4 - Najčešće korišćeni kodovi za formatiranje datuma pomoću funkcije *date()*

	Dani
d	Dan u mesecu (od 01 do 31)
j	Dan u mesecu bez vodeće nule (od 1 do 31)
D	Skraćeni tekstualni prikaz dana na engleskom, 3 slova (Mon-Sun)
l	Pun tekstualni prikaz dana na engleskom (Monday-Sunday)
N	Redni broj dana u sedmici (1=Ponedeljak, 7=Nedelja)
S	Sufiks na engleskom koji označava redni broj (1st, 2nd, itd)
w	Redni broj dana u sedmici (0=Nedelja, 6=Subota)
z	Redni broj dana u godini (od 0 do 365)
	Sedmice
W	Redni broj sedmice u godini prema ISO-8601, prvi dan je Ponedeljak
	Meseci
m	Redni broj meseca (od 01 do 12)
n	Redni broj meseca bez vodeće nule (od 1 do 12)
M	Skraćeni tekstuelni prikaz meseca na engleskom (Jan-Dec)
F	Pun tekstuelni prikaz meseca na engleskom (January-December)
t	Broj dana u datom mesecu
	Godine
Y	Godina, 4 cifre
y	Godina, 2 cifre
L	Da li je godina prestupna? (1=da, 0=ne)
o	Prikaz godine prema ISO-8601
	Vreme
H	Sati, sa vodećom nulom (00-23)
G	Sati, bez vodeće nule (0 to 23)
h	Sati, sa vodećom nulom (01-12)
g	Sati, bez vodeće nule (1-12)
i	Minuti, sa vodećom nulom (00-59)
s	Sekunde, sa vodećom nulom (00-59)
u	Mikrosekunde
	Ostalo
a	Oznaka za pre ili poslepodne na engleskom, mala slova (am/pm)
A	Oznaka za pre ili poslepodne na engleskom, velika slova (AM/PM)
e	Identifikator vremenske zone (npr. CET)
I	Letnje (1) ili zimsko (0) računanje vremena
O	Razlika u odnosu na vreme po Griniču, npr. +0100
c	ISO-8601 datum, primer: 2020-12-26T01:31:00+01:00)
U	Unix timestamp, broj sekundi proteklih od ponoći 1. januara 1970. GMT

Listing 18 - Konverzija stringa u UNIX vreme

```
<?php
$vreme_tekst = 'Sun 3. Jan 2021. 06:36PM';
$vreme_unix = strtotime($vreme_tekst);
echo date('d.m.Y H:i:s', $vreme_unix);
?>

03.01.2021 18:36:00
```

Vreme se može transformisati upotrebom funkcije *date(šablon, vreme)*, listing 17, u kojoj prvi parametar šablon definiše željeni format, dok se *vreme* navodi kao *UNIX timestamp*. Najčešće upotrebljavani parametri za formiranje šablona su prikazani u tabeli 4.

Ukoliko je podatak o trenutnom vremenu dat u obliku stringa na engleskom jeziku, potrebno je prvo izvršiti transformaciju ovog podatka u *UNIX timestamp* upotrebom funkcije *strtotime()*, pa tek onda formirati željeni oblik pomoću funkcije *date()*, listing 18.

2.10 NIZOVI

Niz, odnosno matrica, predstavlja skup podataka koji su smešteni u jednoj promenljivoj. U PHP programskom jeziku, niz se deklariše upotrebom *array()* funkcije:

```
$niz = array(1, 2, 3);
```

ili alternativno, uglastih zagrada:

```
$niz = [1, 2, 3];
```

Niz može biti, listing 19:

- *indeksiran*, indeksi su numerički
- *asocijativan*, indeksi su stringovi
- *višedimenzioni*, kada sadrži jedan ili više nizova.

Direktan pristup članu niza radi setovanja ili očitavanja vrednosti se vrši navođenjem imena promenljive i odgovarajućeg indeksa napisanog unutar uglastih zagrada:

```
echo $niz[1];
```

Listing 19 - Tipovi nizova: a) Indeksiran, b) asocijativan i c) višedimenzioni

```
$automobili = [
    1 => "BMW",
    2 => "Audi",
    3 => "Volkswagen"
];
```

a)

```
$automobili = [
    "BMW" => 32,
    "Audi" => 15,
    "Volkswagen" => 12
];
```

b)

```
$automobili = [
    "DE" => [
        "BMW",
        "Audi",
        "Volkswagen"
    ],
    "FR" => [
        "Renault",
        "Citroen",
        "Peugeot"
    ],
    "IT" => [
        "Fiat",
        "Alfa Romeo",
        "Ferrari"
    ]
];
```

c)

Slika 20 - Ispis sadržaja niza

```php
<?php
$automobili = ['BMW', 'Audi', 'Mercedes'];
print_r($automobili);
?>
```

Rezultat u veb brauzeru:

Array ([0] => BMW [1] => Audi [2] => Mercedes)

Izvorni kod stranice u veb brauzeru:

```
Array
(
    [0] => BMW
    [1] => Audi
    [2] => Mercedes
)
```

2.10.1Dužina niza

Podatak o broju članova u nizu se može dobiti primenom PHP funkcije *count()*. Za svaki od primera prikazanih na listingu 19, rezultat će biti 3.

2.10.2 Ispis članova niza

Kompletan sadržaj niza se može ispisati u okviru veb stranice upotrebom PHP funkcije *print_r,* slika 20. Ovaj metod se koristi pre svega prilikom testiranja aplikacije.

Listing 20 - Transformacija stringa u niz

```php
<?php
    $imena = 'Marko|Petar|Jovan|Saša';
    $imena_niz = explode('|', $imena);
    print_r($imena_niz);
?>

Array
(
    [0] => Marko
    [1] => Petar
    [2] => Jovan
    [3] => Saša
)
```

Listing 21 - Transformacija niza u string

```
<?php
    $imena_niz = [
        'Marko',
        'Petar',
        'Jovan',
        'Saša'
    ];
    $imena = implode('|', $imena_niz);
    print $imena;
?>

Marko|Petar|Jovan|Saša
```

2.10.3 Transformacija string u niz

String se može transformisati u niz primenom PHP funkcije *explode()*, ukoliko je u njemu moguće prepoznati jedan ili više karaktera koji se mogu iskoristiti kao razdelnik (engl. *separator*):

```
$imena = 'Marko|Petar|Jovan|Saša';
```

U prethodnom primeru, znak "|" (vertikalna crta, engl. *vertical bar*) se može iskoristiti kao razdelnik, listing 20. Separator ne može biti prazan string.

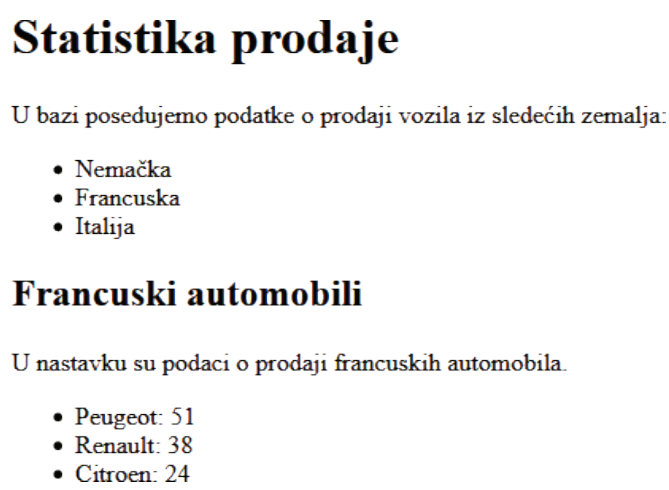
2.10.4 Transformacija niza u string

Niz se može transformisati u string primenom PHP funkcije *implode()*, pri čemu se kao argumenti funkcije navode string koji će biti iskorišćen kao razdelnik i niz koji se transformiše, listing 21.

Listing 22 - Kontrolisani ispis članova niza

```
<?php
$automobili = [
  "Nemačka" => [
    "BMW" => 32,
    "Audi" => 15,
    "Volkswagen" => 12
  ],
  "Francuska" => [
    "Renault" => 38,
    "Citroen" => 24,
    "Peugeot" => 51
  ],
  "Italija" => [
    "Fiat" => 125,
    "Alfa Romeo" => 12,
    "Ferrari" => 0
  ]
];
$francuski_automobili = $automobili['Francuska'];
arsort($francuski_automobili);
?>
<h1>Statistika prodaje</h1>
<div>
  U bazi posedujemo podatke o prodaji
  vozila iz sledećih zemalja:
</div>
<ul>
  <li>
    <?= implode('</li><li>', array_keys($automobili)) ?>
  </li>
</ul>
<h2>Francuski automobili</h2>
<div>
  U nastavku su podaci o prodaji francuskih automobila.
</div>
<ul>
<?php foreach(
        $francuski_automobili
          AS $marka => $prodato
      ): ?>
<li><?= "$marka: $prodato" ?></li>
<?php endforeach; ?>
</ul>
```

Slika 21 - Kontrolisani ispis članova niza, veb stranica



Statistika prodaje

U bazi posedujemo podatke o prodaji vozila iz sledećih zemalja:

- Nemačka
- Francuska
- Italija

Francuski automobili

U nastavku su podaci o prodaji francuskih automobila.

- Peugeot: 51
- Renault: 38
- Citroen: 24

Listing 23 - Dinamičko generisanje niza

```
<?php
$niz = [];
for($i = 0; $i < 10; $i++)
    $niz[] = rand(0, 100);
print_r($niz);
?>

Array
(
    [0] => 87
    [1] => 8
    [2] => 72
    [3] => 8
    [4] => 63
    [5] => 3
    [6] => 77
    [7] => 95
    [8] => 42
    [9] => 3
)
```

2.10.5Dinamičko generisanje niza

Osim predefinisane strukture prikazane u prethodnim primerima, sadržaj niza se može generisati i dinamičkim putem, listing 23. Ovde se promenljivoj *$niz* koja je deklarisana kao matrica, kroz petlju u svakoj iteraciji dodeljuje novi član sa slučajnom vrednošću između 0 i 100 primenom PHP funkcije *rand()*.

2.10.6 Rad sa nizovima

PHP poseduje veliki broj funkcija za rad sa nizovima. Ključevi i vrednosti članova niza se po potrebi mogu setovati, očitavati ili menjati, a sadržaj dva i više nizova se može međusobno kombinovati. Pogodni su za privremeno smeštanje podataka preuzetih iz baze podataka.

Model jednog takvog višedimenzionog skupa, listing 22, sadrži podatke o prodaji automobila prema marki, pri čemu su marke grupisane prema zemlji porekla. Kombinovanom upotrebom petlje i direktnog pristupa članovima niza se može ostvariti kontrolisani prikaz svih članova ili samo željenog podskupa. Ukoliko je potrebno da se generiše spisak sa zemljama porekla vozila, u ovom primeru se mogu izdvojiti ključevi prvog reda u nizu *$automobili* primenom PHP funkcije *array_keys()*. Rezultujući niz je potom moguće prikazati kao neuređenu HTML listu, linije 27-31.

Da bi se formirao izveštaj o prodaji francuskih automobila pri čemu su marke sortirane na osnovu broja prodatih vozila po opadajućem redosledu, formirana je nova promenljiva *$francuski_automobili* radi lakše manipulacije sa podacima, linija 19. Podskup je potom sortiran prema vrednostima u opadajućem redosledu primenom PHP funkcije *arsort()*, linija 20. Napokon, listing je formiran pomoću *foreach* petlje, linije 37-42. Rezultat rada ove skripte je veb stranica prikazana na slici 21.

2.11 Funkcije

Funkcija je blok koda koji se može pozvati, odnosno izvršiti po potrebi. Osim velikog broja ugrađenih funkcija, PHP programerima omogućava i kreiranje sopstvenih funkcija.

Listing 24 - Deklaracija funkcije

```
<?php
function ispisi_poruku($poruka) {
    if ($poruka == '')
        return;
    echo $poruka;
}

ispisi_poruku('Univerzitet Singidunum');
?>
```

Funkcija se deklariše pomoću ključne reči *function*, listing 24. Ime funkcije može da se sastoji od alfanumeričkih karaktera, dok se reči u imenu funkcije međusobno razdvajaju donjom crtom. Ukoliko funkcija zahteva ulazne parametre, oni joj se mogu proslediti prilikom poziva kao parametri. Imena funkcija nisu osetljiva na velika i mala slova, što je suprotno konvenciji koja se koristi prilikom formiranja imena promenljivih. PHP će u navedenom primeru ispravno izvršiti poziv funkcije sa:

```
ispisi_poruku('Univerzitet Singidunum')
```

ili

```
ispisi_PORUKU('Univerzitet Singidunum')
```

Funkcija se poziva ispisom njenog imena, nakon čega se u zagradi navode eventualni ulazni parametri. Parametri se međusobno razdvajaju zarezom. Ukoliko je potrebno da se izađe iz funkcije ili da se vrati rezultat rada funkcije, koristi se ključna reč *return*.

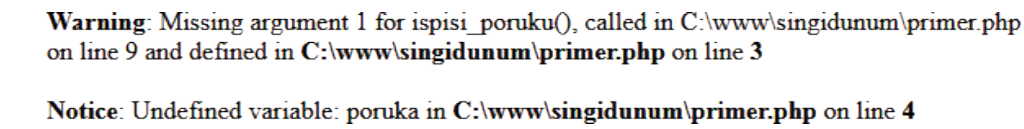
Počevši od verzije PHP 7, moguće je deklarisati tip podataka ulaznih parametara, kao i tip podatka koji se vraća kao rezultat rada funkcije.

U primeru na listingu 24, funkcija je deklarisana na takav način da se prilikom njenog poziva mora navesti i argument. Ukoliko bi poziv bio izvršen samo sa

```
ispisi_poruku();
```

PHP će generisati grešku, slika 22.

Slika 22 - Greška prouzrokovana pozivom funkcije bez argumenta



Da bi se izbegle ovakve situacije, dovoljno je prilikom deklaracije funkcije navesti pretpostavljenu (*default*) vrednost parametra, nekon čega isti više nije obavezno navoditi prilikom poziva funkcije:

```
function ispisi_poruku($poruka = "" ) {
          ...
      }
```

Ukoliko je priroda funkcije takva da je broj ulaznih parametara promenljiv, moguće je koristiti samo jedan parametar koji je matrica čiji su članovi zapravo ulazni podaci za funkciju.

Listing 25 - Prosleđivanje argumenata referencom

```
<?php
function obradi(&$rec) {
    $rec .= ' Singidunum';
}
$naslov = 'Univerzitet';
obradi($naslov);
echo $naslov;
?>
```

Kada se parametar funkcije zada kao vrednost, i ako se vrednost tog parametra u funkciji promeni, ova promena ostaje neprimećena van funkcije.
Ukoliko je potrebno da se dozvoli da funkcija može da izmeni vrednost ulaznog parametra, potrebno je da se prilikom pozivanja funkcije parametar unese isključivo kao referenca. To se postiže dodavanjem znaka "&" (engl. *ampersand*) odgovarajućem parametru prilikom deklaracije funkcije, listing 25.

Listing 26 - Anonimna funkcija, dodela vrednosti promenljivoj

```php
<?php
$kvadrat = function ($x) {
    return pow($x, 2);
};
echo "Dva na kvadrat je " . $kvadrat(2);
?>
```

Listing 27 - Obrada članova niza

```php
<?php
$niz = ['PHP', 'JavaScript', 'HTML', 'CSS'];
array_walk($niz, function(&$v, $k) {
  $v = $v . ' - Programski jezik';
});
print_r($niz);
?>
```

Slika 23 - Obrađeni članovi niza, rezultat rada listinga 27

```
Array
(
    [0] => PHP - Programski jezik
    [1] => JavaScript - Programski jezik
    [2] => HTML - Programski jezik
    [3] => CSS - Programski jezik
)
```

Listing 28 - Zamena dela stringa upotrebom regularnih izraza i anonimne funkcije

```php
<?php
echo preg_replace_callback('#-([a-z]+)#', function ($match) {
    return '-' . strtoupper($match[1]);
}, 'univerzitet-singidunum');
?>
```

2.11.1 Anonimne funkcije

Anonimne funkcije (engl. *anonymous functions*), ili zatvaranja/zatvorenja (engl. *closures*) su funkcije koje nemaju konkretno ime. Mogu se koristiti u različitim situacijama:

- Dodela vrednosti promenljivoj pomoću anonimne funkcije, listing 26.
- Obrada članova niza primenom anonimne funkcije, listing 27, koja se koristi kao parametar PHP funkcije *array_walk()*. Rezultat obrade niza je prikazan na slici 23.
- Zamena dela stringa upotrebom regularnih izraza, listing 28. Upotreba regularnih izraza je opisana u poglavlju 9.

Listing 29 - Vidljivost promenljive u anonimnoj funkciji

```
1  <?php
2  $rec  = 'Univerzitet';
3  function pozdrav() {
4      $rec = 'Singidunum';
5      $func  = function() use ($rec) {
6          echo "Dobrodošli na {$rec}";
7      };
8      $func();
9  }
10 pozdrav();
11 ?>
```

Prilikom upotrebe anonimnih funkcija treba obratiti pažnju na vidljivost promenljivih, odnosno kako da se u takvoj funkciji upotrebe promenljive koje se nalaze u opsegu vidljivosti kojem i sama funkcija pripada. To se može postići upotrebom ključne reči *use* prilikom deklaracije anonimne funkcije, listing 29, linija 5. U navedenom primeru, rezultat programa će biti

```
Dobrodošli na Singidunum
```

zato što je u anonimnoj funkciji dostupna promenljiva *$rec* deklarisana unutar funkcije *pozdrav()*, linija 4, dok globalna promenljiva sa istim imenom deklarisana u liniji 2 ostaje nedostupna.

2.12 SERIJALIZACIJA PODATAKA

Proces serijalizacije podataka predstavlja transformaciju izvornog podatka koji može biti promenljiva, niz, objekat i sl. u string koji je pogodan za skladištenje. Obrnuti proces se naziva deserijalizacija podataka, u kojem se serijalizovana sekvenca transformiše u izvorni oblik. U primeru na listingu 30, izvorni niz (linije 2-21) se serijalizuje i ispisuje kao string (linije 22-23, slika 24), a potom se string deserijalizacijom vraća u izvorni oblik (linije 24-25, slika 25).

Listing 30 - Serijalizacija niza i deserijalizacija podataka, kod

```
1  <?php
2  $singidunum = [
3      'centri' => [
4          'beograd' => [
5              'adresa' => 'Danijelova 32',
6              'telefon' => '011/3094 094'
7          ],
8          'niš' => [
9              'adresa' => 'Nikole Pašića 28',
10             'telefon' => '011/3094 094'
11         ],
12         'novi-sad' => [
13             'adresa' => 'Bulevar Mihajla Pupina 4a',
14             'telefon' => '021/6621-901'
15         ],
16         'valjevo' => [
17             'adresa' => 'Železnička 5',
18             'telefon' => '014/292 610'
19         ],
20     ]
21 ];
22 $serijalizovano = serialize($singidunum);
23 echo $serijalizovano;
24 $deserijalizovano = unserialize($serijalizovano);
25 print_r($deserijalizovano);
26 ?>
```

Slika 24 - Serijalizovani podaci, rezultat

```
Serijalizovani podaci

a:1:{s:6:"centri";a:4:{s:7:"beograd";a:2:{s:6:"adresa";s:1
3:"Danijelova 32";s:7:"telefon";s:12:"011/3094
094";}s:4:"niš";a:2:{s:6:"adresa";s:18:"Nikole Pašića
28";s:7:"telefon";s:12:"011/3094
094";}s:8:"novi-sad";a:2:{s:6:"adresa";s:25:"Bulevar
Mihajla Pupina
4a";s:7:"telefon";s:12:"021/6621-901";}s:7:"valjevo";a:2:{
s:6:"adresa";s:14:"Železnička
5";s:7:"telefon";s:11:"014/292 610";}}}
```

Slika 25 - Serijalizovani podaci vraćeni u izvorni oblik, rezultat

```
Serijalizovani podaci vraćeni u izvorni oblik

Array (
  [centri] => Array (
    [beograd] => Array (
      [adresa] => Danijelova 32
      [telefon] => 011/3094 094
    )
    [niš] => Array (
      [adresa] => Nikole Pašića 28
      [telefon] => 011/3094 094
    )
    [novi-sad] => Array (
      [adresa] => Bulevar Mihajla Pupina 4a
      [telefon] => 021/6621-901
    )
    [valjevo] => Array (
      [adresa] => Železnička 5
      [telefon] => 014/292 610
    )
  )
)
```

Serijalizacija se koristi u slučajevima kada je potrebno više podataka smestiti u jedno polje, npr. u bazi podataka.

Listing 31 - Redirekcija upotrebom HTTP zaglavlja, funkcija *header()*

```
<?php
header('Location: https://www.singidunum.ac.rs');
exit;
?>
```

2.13 HTTP ZAGLAVLJA

Kada je potrebno da aplikacija pošalje HTTP zaglavlje, opisano u poglavlju 1.2.2, koristi se PHP funkcija *header()*. U primeru na listingu 31, vrši se redirekcija veb brauzera na eksternu veb adresu (*singidunum.ac.rs*) uz slanje koda 302 (redirekcija). Skripta se potom obavezno zaustavlja, funkcija *exit*. Na listingu 32 se definiše kod statusa zahteva, što je u ovom slučaju greška 404, odnosno signal da tražena veb stranica nije pronađena.

Listing 32 - Slanje koda statusa zahteva, greška 404

```
<?php
header("HTTP/1.0 404 Not Found");
?>
```

Zaglavlje mora biti poslato pre ostalih izlaznih podataka (engl. output), bilo da su u pitanju HTML tagovi, prazne linije, razmak i sl. Na ovo mora posebno da se obrati pažnja kada se u tok aplikacije uključuje kod iz različitih fajlova, npr. upotrebom funkcije include(), ili kada se u istom dokumentu nalaze kombinovani PHP kod i HTML kod.

Listing 33 - Upotreba PHP funkcije include()

```
1 <html>
2 <head>
3 </head>
4 <body>
5     <?php include('header.php'); ?>
6     <?php include('modul.php'); ?>
7     <?php include('footer.php'); ?>
8 </body>
9 </html>
```

2.14 UKLJUČIVANJE PHP KODA IZ DRUGIH FAJLOVA

PHP omogućava uključivanje programskog koda koji se nalazi u PHP fajlu u drugi PHP fajl, pri čemu je rezultat isti kao da je deo koda kopiran. Mogućnost uključivanja PHP fajlova po potrebi omogućava segmentaciju koda i bitno pojednostavljuje proces razvoja veb aplikacije. U ovu svrhu se mogu koristiti sledeće PHP funkcije:

- *include($ime_fajla)* - uključi PHP kod iz fajla *$ime_fajla*. Jedan fajl po potrebi može biti više puta uključen. Ukoliko navedeni dokument ne postoji, funkcija će generisati grešku nivoa *E_WARNING*. Primer upotrebe funkcije *include()* je prikazan na listingu 33.
- *include_once($ime_fajla)* - isto kao funkcija *include(),* sa tom razlikom da fajl koji je već jedanput uključen, neće biti ponovo uključen u izvršavanje aplikacije.
- *require($ime_fajla)* - isto kao funkcija *include()*, osim u slučaju ako traženi dokument ne postoji kada funkcija *require()* generiše kritičnu grešku, a izvršavanje aplikacije se zaustavlja.
- *require_once($ime_fajla)* - isto kao *require()*, osim što će uključivanje dokumenta biti ignorisano ukoliko je dokument prethodno već bio uključen.

Funkcije *require_once()* i *include_once()* zahtevaju od sistema da vodi i proverava log o dokumentima koji su već uključeni u tok aplikacije. Iako to na prvi pogled ne predstavlja bitno opterećenje za server, u slučaju veb aplikacija koje generišu veliki broj stranica na dnevnom nivou *(n≥106)* to može predstavljati problem. Veliki broj *require_once()* i *include_once()* poziva u aplikaciji ukazuje na kod čiju arhitekturu je potrebno optimizovati.

VEB OBRASCI

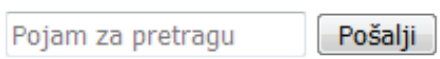
3. VEB OBRASCI

Jedan od važnih zadataka veb aplikacije jeste da prihvati, proveri i obradi podatke koje je korisnik uneo putem veb obrasca. Ti podaci potom mogu biti pohranjeni na veb serveru, sačuvani u bazi podataka, iskorišćeni za komunikaciju sa drugim aplikacijama putem API-ja[9], prosleđeni putem email poruke i sl. Nasuprot tome, ukoliko podaci koje je korisnik poslao nisu ispravni, potrebno je da aplikacija na veb stranici ispiše odgovarajuću poruku čime korisniku daje do znanja šta je potrebno ispraviti.

Slika 26 - Veb obrazac za pretragu sadržaja sajta - označavanje elemenata obrasca

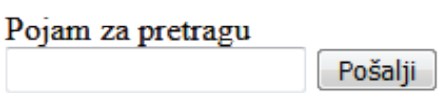
a)

```
<form>
  <input type="text" name="pojam" placeholder="Pojam">
  <button type="submit">Pošalji</button>
</form>
```



b)

```
<form>
  <label for="pojam">Pojam za pretragu</label>
  <input type="text" name="pojam">
  <button type="submit">Pošalji</button>
</form>
```



Veb obrazac je deo veb stranice koji korisniku omogućava da pošalje podatke ili dokument veb serveru. Sastoji se iz HTML elemenata za unos teksta (*input, textarea*), za izbor jedne ili više ponuđenih opcija *(select, input* tipa *radio* ili *checkbox*), dugmeta za pokretanje određene akcije (*submit, reset, cancel*) i sl.

9 Programski interfejs aplikacije (engl. *Application Programming Interface*)

3.1 OZNAČAVANJE NAMENE ELEMENATA

Namena svakog elementa u veb obrascu mora biti jasno naznačena, kako bi korisnik znao šta se od njega očekuje. Elementi se mogu vizuelno stilizovati upotrebom CSS-a, dok se po potrebi njihovo ponašanje može kontrolisati pomoću JavaScript-a. Na jednoj veb stranici se može se nalaziti više veb obrazaca. Veb obrasci međusobno ne smeju biti ugnežđeni.

Na slici 26 su prikazana dva primera kako može da se formira jednostavan obrazac za pretragu sadržaja veb sajta, koji se međusobno razlikuju po načinu označavanja elemenata.

Listing 34 - Pozicioniranje oznake elementa u okviru[10]

```
<style>
.group {
  position: relative;
  display: inline-block;
}
.group > label  {
  padding: 0 0.2em;
  position: absolute;
  top: -0.5em;
  left: 1em;
  background-color: white;
}
.group > input {
  padding: 0.5em;
  border-radius: 0.5em;
  border: 2px solid lightblue;
  outline: none;
}
</style>
<form>
  <div class="group">
    <label>Pojam za pretragu</label>
    <input type="text" name="pojam">
  </div>
    <button type="submit">Pošalji</button>
</form>
```

10 https://stackoverflow. com/questions/48260921/css-how-to-change-position-of-input-placeholder/48261244

Obrazac se sastoji iz HTML *input* elementa za unos teksta u jednoj liniji i dugmeta tipa *submit*. U prvom primeru, namena elementa za unos teksta je naznačena pomoću atributa *placeholder,* koji je vidljiv u elementu sve dok se u isti ne unesu podaci. U drugom primeru, namena elementa za unos teksta je prikazana primenom HTML taga *label*, koji zauzima dodatno mesto na veb stranici, ali ostaje vidljiv kada se u element unesu podaci. Kada se formiraju jednostavni veb obraci sačinjeni od minimalnog broja elemenata, može se koristiti atribut placeholder radi kompaktnijeg prikaza. U slučaju zahtevnijih obrazaca oznake elemenata treba formirati na takav način da su stalno vidljive, bez obzira na tok procesa unosa podataka. Alternativno, namena elementa veb obrasca može biti prikazana u njegovom okviru, listing 34.

3.2 IMENOVANJE ELEMENATA

Formiranjem odgovarajućih selektora, moguće je stilizovati željene elemente veb stranice pomoću CSS-a, ili kontrolisati njihovo ponašanje pomoću Java Script-a. Elemente je u ovu svrhu moguće ciljati preko njihovog HTML taga, imena CSS klase, identifikatora (*id*), imena (*name*) ili nekog drugog atributa.

Međutim, prilikom slanja podataka iz veb obrasca ka veb aplikaciji, neophodno je koristiti atribut *name*. Iz tog razloga, u sva tri prethodno prikazana primera (slika 26, listing 34), elementu za unos teksta je dodeljen atribut *name*.

Slika 27 - Veb obrazac za unos imena i prezimena

3.3 TRANSFER PODATAKA KA VEB SERVERU

Osnovna namena HTTP protokola je da omogući komunikaciju između klijenta i veb servera. Kada korisnik unese tražene podatke u veb obrazac i klikne na dugme za slanje, podaci dolaze do veb aplikacije koja mora da ih prihvati, proveri i obradi. Za ovu namenu se koriste HTTP metode GET i POST.

Listing 35 - Očitavanje vrednosti poslatih putem veb obrasca GET metodom

a)
```php
<?php
if ($_GET)
    print_r($_GET);
?>
<form method="get">
    <label>Ime</label>
    <input>
    <br/>
    <label>Prezime</label>
    <input>
    <br/>
    <button>Pošalji</label>
</form>
```

b)
```php
<?php
if ($_GET)
    print_r($_GET);
?>
<form method="get">
    <label>Ime</label>
    <input name="ime">
    <br/>
    <label>Prezime</label>
    <input name="prezime">
    <br/>
    <button>Pošalji</label>
</form>
```

3.3.1 GET metoda

GET metod se koristi za slanje podataka veb aplikaciji putem URL parametara, koji se integrišu u veb adresu. Ovo je podrazumevana metoda slanja podataka, tako da se atribut *method* u HTML tagu <*form*> može izostaviti.

Jednostavan veb obrazac za unos imena i prezimena, slika 27, je generisan pomoću koda prikazanog na listingu 35. Ukoliko *input* elementu nije naveden atribut *type*, dodeljuje mu se podrazumevana vrednost *text*. Na dugmetu, HTML tag <*button*>, takođe nije upotrebljen atribut type, a podrazumevana vrednost je *submit*.

Iako će obe verzije koda (a i b) prikazati u veb brauzeru obrazac na ispravan način, u prvoj varijanti (a) PHP neće moći da očita vrednosti URL parametara zato što *input* elementima nije dodeljen atribut *name*.

U drugoj varijanti koda (b), nakon popunjavanja veb obrasca i slanja podataka, URL-u stranice će automatski biti dodati URL parametri sa vrednostima koje su unete u odgovarajuće <*input*> elemente, npr.

```
index.php?ime=Jovan&prezime=Jovanović
```

PHP će vrednosti URL parametara smestiti u superglobalnu promenljivu *$_GET,* čiji sadržaj potom može biti prikazan na ekranu primenom funkcije *print_r()*:

```
Array ( [ime] => Jovan [prezime] => Jovanović )
```

Ove vrednosti je moguće dodeliti odgovarajućim promenljivim, npr. *$ime* i *$prezime*, i potom ih iskoristiti u aplikaciji prema potrebi:

```
$ime = $_GET['ime'] ?? '';

$prezime = $_GET['prezime'] ?? '';
```

U navedenom primeru je primenjen i operator sjedinjavanja (??), koji promenljivim dodeljuje vrednost praznog stringa u slučaju da odgovarajući podatak nije unet u veb obrazac, čime se izbegava PHP obaveštenje o upotrebi nepostojećeg ključa u matrici *$_GET*.

Teorijski, količina podataka koja se može poslati GET metodom nije ograničena[11]. Većina veb servera dozvoljava prenos do 8 KB podataka (8129 bajta)[12], a ukoliko se uzmu u obzir specifičnosti i ograničenja različitih veb brauzera i operativnih sistema, smatra se lošom praksom ukoliko se formiraju URL-ovi duži od 2000 karaktera.

Primenom GET metode, korisniku se omogućava da URL formiran na osnovu podataka unetih u veb obrazac podeli na drugim veb platformama, ili sačuva u veb brauzeru kao *bookmark*. Ova osobina je korisna npr. za filtriranje sadržaja. Nasuprot tome, kada se putem veb obrasca šalju podaci koje je potrebno smestiti na server ili ubaciti u bazu podataka, ili poverljivi podaci kao što je slučaj sa obrasacem za prijavljivanje (*login*), umesto GET metode se koristi POST metoda.

11 RFC 2616, sekcija 3.2.1
12 RFC 7230

3.3.2 POST metoda

Mehanizam formiranja veb obrasca i očitavanja poslatih podataka na strani veb servera je gotovo identičan kao u slučaju GET metode, sa tom razlikom da se metod slanja podataka (atribut *method*) mora striktno navesti uz HTML tag *<form>*, ovako poslati podaci se ne smeštaju u URL zahteva, a PHP preuzete podatke smešta u superglobalnu promenljivu *$_POST*.

Tabela 5 - Mime tip dokumenata

Office dokumenti		
.doc	Microsoft Word	application/msword
.docx	Microsoft Word	application/vnd.openxmlformats-officedocument.wordprocessingml.document
.ppt	Microsoft PowerPoint	application/vnd.ms-powerpoint
.pptx	Microsoft PowerPoint	application/vnd.openxmlformats-officedocument.presentationml.presentation
.xls	Microsoft Excel	application/vnd.ms-excel
.xlsx	Microsoft Excel	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
.odp	OpenDocument, presentation	application/vnd.oasis.opendocument.presentation
.ods	OpenDocument, spreadsheet	application/vnd.oasis.opendocument.spreadsheet
.odt	OpenDocument, text	application/vnd.oasis.opendocument.text
Slike		
.bmp	Windows OS/2 Bitmap Graphics	image/bmp
.gif	GIF image	image/gif
.ico	Icon format	image/vnd.microsoft.icon
.jpeg, .jpg	JPEG image	image/jpeg
.png	PNG image	image/png
.svg	Scalable Vector Graphics	image/svg+xml
.tiff	TIFF Image	image/tiff
Video		
.avi	Audio Video Interleave	video/x-msvideo
.mp3	MP3 audio	audio/mpeg
.mpeg	MPEG Video	video/mpeg
.ts	MPEG transport stream	video/mp2t
.wav	Waveform Audio Format	audio/wav
.3gp	3GPP audio/video container	audio/3gpp ili video/3gpp
.3g2	3GPP2 audio/video container	audio/3gpp2 ili video/3gpp2

Fontovi		
.eot	MS Embedded OpenType fonts	application/vnd.ms-fontobject
.otf	OpenType font	font/otf
.ttf	TrueType Font	font/ttf
.woff	Web Open Font Format	font/woff
.woff2	Web Open Font Format	font/woff2
Arhive		
.zip	ZIP archive	application/zip
.7z	7-zip archive	application/x-7z-compressed
.bz	BZip archive	application/x-bzip
.bz2	BZip2 archive	application/x-bzip2
.gz	GZip Compressed Archive	application/gzip
.rar	RAR archive	application/vnd.rar
.tar	Tape Archive (TAR)	application/x-tar

Ovom metodom je moguće poslati veću količinu podataka u odnosu na GET metodu, uključujući i fajlove.

Listing 36 - Slanje dokumenta putem veb obrasca

```
<?php
if ($_POST)
    print_r($_POST);
if ($_FILES)
    print_r($_FILES);
?>

<form method="post" enctype="multipart/form-data">
  <label for="ime">Ime</label>
  <input type="text" name="ime"><br/>
  <label for="prezime">Prezime</label>
  <input type="text" name="prezime"><br>
  <label for="slika">Slika</label>
  <input type="file" name="slika"><br>
  <button type="submit">Pošalji</button>
</form>
```

3.4 OTPREMANJE I PRIJEM DOKUMENATA

Ukoliko putem veb obrasca treba da se otpremi jedan ili više dokumenata sa računara korisnika, uz HTML tag <*form*> neophodno je navesti atribut *enctype* sa vrednošću *multipart/form-data,* dok sam veb obrazac mora da sadrži odgovarajući broj <*input*> elemenata tipa file.

Slika 28 - Prijem dokumenta putem veb obrasca

```
Array
(
    [ime] => Jovan
    [prezime] => Jovanović
)
Array
(
    [slika] => Array
    (
        [name] => avatar.jpg
        [type] => image/jpeg
        [tmp_name] => C:\xampp56\tmp\php95E.tmp
        [error] => 0
        [size] => 8970
    )
)
```

U primeru na listingu 36, putem veb obrasca osim imena i prezimena, korisnik može da pošalje i svoju fotografiju. Kao što je već opisano, PHP ovako poslate podatke iz veb obrasca smešta u superglobalnu promenljivu *$_POST*, dok podatke o primljenim fajlovima čuva u superglobalnoj promenljivoj *$_FILES*, slika 28:

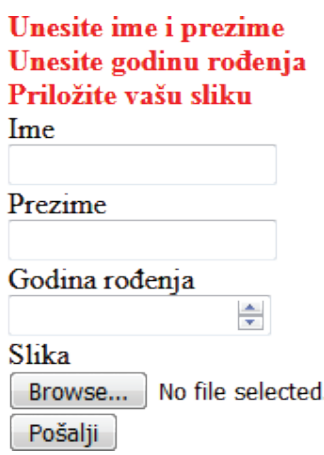
- *name* - ime originalnog dokumenta
- *type* - mime tip dokumenta, tabela 5
- *tmp_name* - privremena lokacija i ime dokumenta na veb serveru
- *error* - kod greške
- *size* - veličina prihvaćenog dokumenta

Prilikom otpremanja dokumenta, kod greške može imati neku od sledećih vrednosti:

- *UPLOAD_ERR_OK* (0) - otpremanje dokumenta je uspešno obavljeno, nema greške.
- *UPLOAD_ERR_INI_SIZE* (1) - veličina dokumenta je veća od maksimalno dozvoljene veličine u php.ini konfiguracionom fajlu.
- *UPLOAD_ERR_FORM_SIZE* (2) - veličina dokumenta je veća od maksimalno dozvoljene veličine navedene u HTML obrascu.
- *UPLOAD_ERR_PARTIAL* (3) - dokument je samo delimično otpremljen.
- *UPLOAD_ERR_NO_FILE* (4) - dokument nije otpremljen.
- *UPLOAD_ERR_NO_TMP_DIR* (6), privremeni direktorijum ne postoji.
- *UPLOAD_ERR_CANT_WRITE* (7), dokument ne može biti zapisan na disku na veb serveru.
- *UPLOAD_ERR_EXTENSION* (8), PHP ekstenzija je prestala sa radom prilikom otpremanja dokumenta na server. PHP ne pruža informaciju o tome koja je ekstenzija izazvala problem.

Nakon što je dokument u potpunosti otpremljen na server, treba ga premestiti iz privremenog direktorijuma u folder gde će trajno biti smešten, pod imenom koje će mu veb aplikacija dodeliti u skladu sa predefinisanim šablonom. Ime tako postavljenog dokumenta ne sme da bude identično originalnom nazivu dokumenta iz bezbednosnih razloga.

Slika 29 - Unos, prijem i provera podataka uz prijavu greške



Listing 37 - Unos, prijem i provera podataka uz prijavu greške, kod

```php
<?php
$greska = '';

function provera() {
    $greska = [];
    extract($_POST);
    extract($_FILES['slika']);
    $godina_rodjenja = (int)$godina_rodjenja;
    if ($ime == '' && $prezime == '')
            $greska[] = 'Unesite ime i prezime';
    else if ($ime == '') $greska[] = 'Unesite ime';
    else if ($prezime == '') $greska[] = 'Unesite prezime';
    else if (preg_match('#[^\p{L}]#u', "$ime $prezime", $m)) {
        $greska[] = "Ime i/ili prezime sadrže nedozvoljene karaktere";
    }
    if ($godina_rodjenja == 0)
            $greska[] = 'Unesite godinu rođenja';
    else if (
        $godina_rodjenja<1875
        || $godina_rodjenja > date('Y')
    ) $greska[] = 'Godina rođenja je van dozvoljenog opsega';
    if ($error == UPLOAD_ERR_NO_FILE)
        $greska[] = 'Priložite vašu sliku';
    else if ($error != UPLOAD_ERR_OK)
        $greska[] = 'Došlo je do greške prilikom preuzimanja slike';
    $greska = empty($greska) ? '' : implode('<br>', $greska);
    return $greska;
}
if ($_POST)
    $greska = provera();
?>

<style>
label, button { display: block; }
.greska { font-weight: bold; color: red; }
</style>

<?php if ($greska != ''): ?>
<div class="greska"><?php echo $greska ?></div>
<?php endif; ?>

<form method="post" enctype="multipart/form-data">
    <label for="ime">Ime</label>
    <input type="text" name="ime"
        value="<?php echo @$_POST['ime']; ?>"><br/>
    <label for="prezime">Prezime</label>
    <input type="text" name="prezime"
        value="<?php echo @$_POST['prezime']; ?>"><br>
    <label for="godina_rodjenja">Godina rođenja</label>
    <input type="number" name="godina_rodjenja"
        value="<?php echo @$_POST['godina_rodjenja']; ?>"><br>
    <label for="slika">Slika</label>
    <input type="file" name="slika"><br>
    <button type="submit">Pošalji</button>
</form>
```

3.5 PROVERA PODATAKA. PRIJAVA GREŠKE.

Prilikom popunjavanja veb obrasca, može doći do nenamernih grešaka prouzrokovanih nepažnjom ili nedovoljnim poznavanjem aplikacije od strane korisnika. Naravno, ne treba zanemariti ni mogućnost zloupotrebe aplikacije. Nakon prijema podataka a pre dalje obrade, podatke je neophodno proveriti:

- Da li je korisnik uneo sve potrebne podatke?
- Da li su uneti podaci po tipu i opsegu u skladu sa očekivanjima, npr. u skladu sa strukturom odgovarajuće tabele u bazi podataka?
- Da li je prilikom preuzimanja dokumenata došlo do greške?

Ukoliko postoje odstupanja koja mogu uticati na ispravan rad aplikacije, potrebno je da se na veb stranici ispiše kratka, ali jasna poruka na osnovu koje će korisnik moći da izvrši očekivanu korekciju.

Veb obrazac prikazan na slici 29, izvorni kod na listingu 37, omogućava unos sledećih podataka o korisniku:

- *Ime*, tip podatka *string*.
- *Prezime*, tip podatka *string*.
- *Godina rođenja*, tip *podatka broj.*
- *Slika*, dokument tipa *slika*, u formatu *jp(e)g* ili *png*

Celokupan kod je smešten u jedan fajl, čije ime može biti index.php. Struktura koda je sledeća:

- *Backend,* poslovna logika napisana u PHP programskom jeziku, linije 1-31.
- Ukoliko su poslati podaci POST metodom, linije 29-30, poziva se funkcija *provera()*, linije 4-28.
- *Frontend,* kod koji će korisniku biti prikazan u brauzeru, linije 33-56.
- CSS za stilizovanje input elemenata, dugmeta i poruke sa greškom, linije 33-36.
- Blok koji korisniku saopštava grešku, linije 38-41.
- Veb obrazac, linije 42-55.

U funkciji *provera()* se redom testiraju vrednosti podataka poslatih *POST* metodom, i ukoliko su otkrivena odstupanja u odnosu na očekivane vrednosti, odgovarajuća poruka se smešta u lokalnu matricu *$greska*.

Radi lakše manipulacije, poslati podaci su primenom PHP funkcije *extract()*, linija 6, transformisani u promenljive čija imena odgovaraju ključevima u matricama *$_POST* i *$_FILES['slika']*. Iako na prvi pogled ova funkcija deluje korisno, opasna je i ne treba da se koristi u glavnom toku programa zbog mogućnosti manipulacije vrednostima globalnih promenljivih. U ovom slučaju se nalazi u funkciji koja nema ulazne parametre i u kojoj se ne koriste globalne promenljive, tako da se njena upotreba može smatrati bezbednom. Ipak, potrebno je obratiti pažnju da se imena elemenata u veb obrascu ne poklapaju sa imenima ključeva u matrici *$_FILES['slika']*.

Ukoliko nisu otkrivene greške u poslatim podacima, matrica *$greska* je prazna (*empty*), i funkcija kao rezultat vraća prazan string. U suprotnom, funkcija će vratiti sadržaj matrice transformisan u string (*implode*), pri čemu su članovi matrice međusobno razdvojeni HTML tagom <*br*> (prelom linije). Ovde treba obratiti pažnju na dve stvari:

- U glavnom toku programa je inicijalizovana promenljiva *$greska*, linija 2, tip podatka string, u koju se smešta rezultat koji vraća funkcija *provera()*.
- U funkciji *provera()* se inicijalizuje matrica *$greska*, linija 5, u koju se smeštaju poruke o otkrivenim greškama koje treba prikazati korisniku aplikacije.

Iako nose isto ime, ove dve promenljive ne utiču jedna na drugu zato što se prva nalazi u glavnom toku programa, dok se druga nalazi u funkciji i lokalnog je karaktera.

Ova jednostavna veb aplikacija je namenjena našem govornom području, tako da se u imenu i prezimenu očekuju slova latinice. Iako se ovakvi podaci mogu proveriti primenom PHP funkcija za rad sa stringovima, neuporedivo efikasniji alat za ovu namenu su regularni izrazi, linija 13, o kojima će detaljnije biti reči u poglavlju 9. Ukoliko su podaci prilikom unosa izostavljeni, odgovarajuća poruka takođe treba da bude prikazana, linije 11-12.

Godina rođenja se tretira kao celobrojni podatak, linija 8. Osim toga, potrebno je proveriti da li ovaj podatak pripada očekivanom opsegu. Za najmanju dozvoljenu vrednost je određena 1875. godina, dok je najveća dozvoljena vrednost tekuća godina, linije 18-21.

Proces preuzimanja slike se ne proverava detaljno, osim što se vrednost promenljive *$error ($_FILES['slika']['error'])* poredi sa vrednostima konstanti UPLOAD_ERR_NO_FILE koja signalizira da dokument nije poslat, linije 22-23, i UPLOAD_ERR_OK koja označava da prilikom transfera dokumenta na veb

server nije došlo do greške, linije 24-25. U slučaju da je dokument otpremljen na veb server bez greške, potrebno je proveriti njegovu ispravnost:

- Da li ekstenzija u imenu dokumenta odgovara očekivanoj vrednosti - *jpg*, *jpeg* ili *png*? Provera može da se izvrši upotrebom regularnih izraza, npr.

```
if (preg_match('#\.(jpe?g|png)$#', $name))
```

- Da li je *mime* tip dokumenta u skladu sa očekivanom vrednošću, tabela 5?

```
if (!in_array($type, ['image/jpeg', 'image/png'])) {
  ...
}
```

- Da li je dokument zaista slika? Ova provera može da se izvrši pomoću funkcija *imagecreatefromjpeg()* ili *imagecreatefrompng()*, u kojima se kao parametar navodi ime preuzetog dokumenta.

Tek pošto dokument prođe navedene provere, može da se prebaci iz privremenog direktorijuma u direktorijum gde će biti trajno smešten.

RAD SA DOKUMENTIMA

4. RAD SA DOKUMENTIMA

Dokument, datoteka, odnosno fajl je skup podataka koji su zapisani na hard disku ili nekom drugom mediju za memorisanje. Svaki fajl ima svoje ime i odgovarajuću ekstenziju, koja se dodeljuje shodno tipu podataka koji se nalaze u dokumentu. Srodni dokumenti se obično grupišu u direktorijume, odnosno foldere. PHP programski jezik omogućava jednostavan rad sa dokumentima, bez obzira na njihov tip i lokaciju.

Prilikom rada sa dokumentima, potrebno je obratiti pažnju na način formiranja putanje do samog dokumenta, kao i na znakove koji se koriste za prelom linije:

- Na *UNIX* operativnim sistemima, znak "/" se koristi za razdvajanje direktorijuma, dok se prelom linije označava kao "\n".
- Na *Windows* platformama se za razdvajanje direktorijuma mogu ravnopravno koristiti znakovi "/" i "\", a prelom linije se označava sa "\r\n".

4.1 ČITANJE DOKUMENATA

Osnovne funkcije koje se koriste za čitanje sadržaja dokumenta u PHP programskom jeziku su

- *readfile(ime_dokumenta)* - čita sadržaj fajla i odmah ga smešta u izlazni bafer. Ukoliko je čitanje dokumenta uspešno okončano, funkcija vraća broj pročitanih bajtova, ili rezultat *false* ukoliko čitanje nije izvršeno.

```
$procitano = readfile('error.log');
print $procitano;
```

- *file_get_contents(ime_dokumenta)* - sadržaj celog dokumenta smešta u string, koji se potom može obrađivati. U slučaju čitanja velikih dokumenata, može prouzrokovati probleme sa memorijom.

```
$log = file_get_contents('error.log');
print $log;
```

- *file(ime_dokumenta)* - slično radi kao funkcija *file_get_contents*, sa tom razlikom što sadržaj pročitanog dokumenta smešta u matricu pri čemu svaki član niza predstavlja jednu liniju tog fajla.

```
$log = file('error.log');
print_r($log);
```

Tabela 6 - Funkcija fopen, režimi rada sa dokumentom[13]

Mod	Opis
r	Samo čitanje, počinje od početka dokumenta.
r+	Čitanje i/ili pisanje, počinje od početka dokumenta
w	Samo pisanje. Ukoliko dokument postoji, otvara ga i uklanja postojeći sadržaj. Ako dokument ne postoji, kreira ga. Pointer se postavlja na početak dokumenta.
w+	Čitanje i/ili pisanje. Ukoliko dokument postoji, otvara ga i uklanja postojeći sadržaj. Ako dokument ne postoji, kreira ga. Pointer se postavlja na početak dokumenta.
a	Samo pisanje. Otvara dokument i pointer postavlja na kraj. Ukoliko dokument ne postoji, kreira ga.
a+	Čitanje i/ili pisanje. Otvara dokument i pointer postavlja na kraj. Ukoliko dokument ne postoji, kreira ga.
x	Samo pisanje. Kreira novi dokument. Ukoliko dokument već postoji, vraća FALSE i grešku.
x+	Čitanje i/ili pisanje. Kreira novi dokument. Ukoliko dokument već postoji, vraća FALSE i grešku.
c	Samo pisanje. Otvara dokument, ili kreira novi ukoliko dokument ne postoji. Postavlja pointer na početak dokumenta.
c+	Čitanje i/ili pisanje Otvara dokument, ili kreira novi ukoliko dokument ne postoji. Postavlja pointer na početak dokumenta.

Navedene funkcije mogu čitati dokumenta koja se nalaze u lokalu na veb serveru, ali takođe i dokumenta koja se nalaze na drugim veb adresama tako što se kao parametar *ime_dokumenta* navede odgovarajući URL.

13 https://www.php.net/manual/en/function. fopen.php

Listing 38 - Sekvencijalno čitanje dokumenta

```
<?php
$fajl = fopen("error.log", "r");
while(!feof($fajl)) {
  $linija = fgets($fajl);
  echo "{$linija}<br/>";
}
fclose($fajl);
?>
```

4.1.1 Sekvencijalno čitanje dokumenta

Ukoliko je potrebno uspostaviti sekvencijalno čitanje dokumenta, odnosno strim (engl. *stream*), to se može postići kreiranjem deskriptora pomoću funkcije *fopen*, potom izvršiti očitavanje segmenata podataka upotrebom funkcija *fread* ili *fgets*, i na kraju zatvoriti uspostavljeni strim pomoću funkcije *fclose,* listing 38.

Obavezni parametri funkcije *fopen()* su ime dokumenta ili URL, i mod pristupa dokumentu, odnosno strimu, tabela 6.

Funkcija *fread* očitava *n* znakova iz otvorenog dokumenta, ili *x* znakova do kraja dokumenta ukoliko je *x*<*n*. Za razliku od nje, *fgets* čita celu liniju.

4.2 KREIRANJE NOVOG DOKUMENTA

Kao i čitanje dokumenta, kreiranje dokumenta i upis sadržaja se može obaviti na više načina:

- Funkcija *file_put_contents* će formirati dokument sa navedenim imenom, i u njega smestiti string naveden na mestu drugog parametra, listing 39. Ukoliko dokument sa istim imenom već postoji, biće prebrisan.

Listing 39 - Kreiranje novog dokumenta, *file_put_contents()*

```
<?php
$tekst = 'Univerzitet Singidunum';
file_put_contents('primer.txt', $tekst);
?>
```

- Funkcija *fopen* će kreirati novi dokument, ukoliko je uz naziv dokumenta naveden i odgovarajući mod, tabela 6. Upis podataka se potom vrši primenom funkcije *fwrite*, listing 40.

Listing 40 - Kreiranje novog dokumenta, *fwrite()*

```
<?php
$fajl = fopen('primer.txt', 'w');
$tekst = 'Univerzitet Singidunum';
fwrite($fajl, $tekst);
fclose($fajl);
?>
```

4.3 MENADŽMENT DIREKTORIJUMA

PHP programski jezik poseduje funkcije za rad sa direktorijumima, kao što su kreiranje, promena naziva i brisanje direktorijuma, očitavanje imena direktorijuma, radnog direktorijuma ili sadržaja direktorijuma i sl.

4.3.1 Očitavanje sadržaja direktorijuma

Za očitavanje spiska fajlova koji se nalaze u određenom direktorijumu, upotrebljava se PHP funkcija *glob()*. Šablon po kome se traže dokumenti može da sadrži sledeće džoker karaktere (engl. *wildcard characters*):

- * (asterisk) - nula ili više karaktera
- ? (upitnik) - tačno jedan karakter
- [...] - u uglastoj zagradi se navodi niz ciljanih karaktera
- \ - bukvalna interpretacija karaktera koji sledi (engl. *escaping*)

Listing 41 - Očitavanje spiska dokumenata u folderu, funkcija *glob()*

```
<?php
$spisak_fajlova = glob('*.txt');
print_r($spisak_fajlova);
?>
```

U skladu sa tim, funkcija *glob* će sa šablonom '*.txt', listing 41, generisati spisak fajlova sa ekstenzijom *txt* u tekućem direktorijumu. Rad ove funkcije može da se reguliše pomoću drugog, opcionog parametra, tabela 7.

Tabela 7 - Kontrola ponašanja funkcije *glob()*

Vrednost	Opis
GLOB_MARK	Dodaje obrnutu kosu crtu svakom rezultatu
GLOB_NOSORT	Vraća nesortiranu listu dokumenata
GLOB_NOCHECK	Vraća šablon ukoliko ništa nije pronađeno
GLOB_NOESCAPE	Obrnuta kosa crta se ne odnosi na metakaraktere
GLOB_BRACE	Modifikuj šablon {a, b, c} da prihvati 'a', 'b' ili 'c'
GLOB_ONLYDIR	Prihvati samo direktorijume koji zadovoljavaju zadati šablon
GLOB_ERR	Zaustavi izvršavanje u slučaju greške

Alternativno, u slučaju da se radi sa direktorijumima u kojima se nalazi veliki broj dokumenata (desetine ili stotine hiljada), mogu se upotrebiti PHP funkcije *readdir()*, listing 42 ili *scandir()*, koje su mnogo brže u odnosu na *glob()* funkciju. Ukoliko je iz nekog razloga potrebno da se prilikom rada sa ovim funkcijama pokazivač (promenljiva *$dir* u listingu 42) resetuje na početak direktorijuma, koristi se funkcija *rewinddir()*.

Listing 42 - Očitavanje sadržaja direktorijuma, funkcija *readdir()*

```
<?php
$dir_ime = "./";
$dokumenti = [];
if (is_dir($dir_ime)) {
    if ($dir = opendir($dir_ime)) {
        while (($fajl = readdir($dir)) !== false) {
            $dokumenti[] = $fajl;
        }
        closedir($dir);
    }
}
?>
<p>
    U direktorijumu <?php echo $dir_ime; ?> ima
    <?php echo count($dokumenti); ?> fajlova.
</p>
<?php if (!empty($dokumenti)): ?>
    <?php foreach($dokumenti AS $fajl): ?>
        <?php echo $fajl; ?><br>
    <?php endforeach; ?>
<?php endif; ?>
```

Listing 43 - Menadžment direktorijuma

```php
<?php
$dir_radni = getcwd();
echo "Trenutni radni direktorijum je {$dir_radni}<br/>";
mkdir('slike');
chdir('slike');
$dir_radni = getcwd();
echo "Radni direktorijum je sada {$dir_radni}<br/>";
chdir('..');
rmdir('slike');
?>
```

4.3.2 Očitavanje i promena radnog direktorijuma

Očitavanje direktorijuma u kome aplikacija trenutno radi (engl. *current working directory*) se vrši upotrebom funkcije *getcwd()*, listing 43. Po potrebi, ovaj direktorijum se može promeniti pomoću funkcije *chdir()* u kojoj se kao parametar navodi ime direktorijuma u koji se prelazi.

4.3.3 Kreiranje i brisanje direktorijuma

Novi direktorijum se kreira pomoću funkcije *mkdir()*, listing 43, u kojoj se osim imena direktorijuma kao opcioni parametri mogu navesti mod pristupa (npr. *0777*, parametar se ignoriše u *Windows OS*), kao i bulov tip (engl. *boolean*) koji predstavlja triger za rekurziju, odnosno kreiranje ugnežđenih direktorijuma navedenih u prvom parametru. Prazan direktorijum se uklanja primenom funkcije *rmdir()*.

4.4 MENADŽMENT DOKUMENATA

Osim kreiranja dokumenata i čitanja njihovog sadržaja, PHP omogućava i vršenje drugih operacija nad fajlovima, poput premeštanja dokumenta, kopiranja, brisanja, promene naziva i sl. U daljem tekstu će biti opisane operacije koje se najčešće koriste u praksi.

4.4.1 Provera da li dokument postoji

Da bi se proverilo da li dokument ili direktorijum sa navedenim imenom postoje, koristi se PHP funkcija *file_exists()*, listing 44. Kao parametar funkcije, navode se ime fajla i po potrebi putanja, kao jedinstveni string.

Listing 44 - Provera da li dokument postoji, funkcija *file_exists*

```
<?php
$slika_postoji = file_exists('slika.jpg');
if ($slika_postoji)
    echo 'Slika postoji';
else
    echo 'Slika ne postoji';
?>
```

Ukoliko je potrebna preciznija provera, da li je dokument čije se ime proverava direktorijum ili fajl, mogu se upotrebiti PHP funkcije *is_dir()*, odnosno *is_file()*.

Listing 45 - Kopiranje dokumenta, funkcija *copy*

```
1  <?php
2  $ime_fajla = 'singidunum.txt';
3  $novi_fajl = 'singidunum.bak01.txt';
4  if (!copy($ime_fajla, $novi_fajl)) {
5      echo "Dokument {$ime_fajla} nije kopiran!";
6  }
7  else {
8      echo "Dokument {$ime_fajla} je uspešno kopiran.";
9  }
10 ?>
```

4.4.2 Kopiranje dokumenta

Kopiranje dokumenata se može izvršiti pomoću PHP funkcije *copy()*, gde se kao parametri navode ime dokumenta koji se kopira i ime novog dokumenta, listing 45. Oba parametra osim imena mogu da sadrže apsolutne ili relativne putanje.

4.4.3 Promena naziva i premeštanje dokumenta

Kada je potrebno da se dokument preimenuje ili premesti sa jedne lokacije na drugu, u PHP programskom jeziku ovaj zadatak može da se izvrši upotrebom funkcije *rename()*. U slučaju da se premešta primljeni dokument (engl. *uploaded*) iz privremenog direktorijuma u direktorijum gde će trajno biti smešten, kako je to opisano u poglavlju 3.4, koristi se funkcija *move_uploaded_file()*. U obe funkcije, ulazni parametri su putanja i ime dokumenta koji se premešta, odnosno putanja i ime novog dokumenta.

Listing 46 - Premeštanje otpremljenog dokumenta, funkcija *move_uploaded_file()*

```
29 if ($_POST) {
30     $greska = provera();
31     if ($greska != '') {
32         $slika_ime = sprintf('slika_%d.png', time());
33         move_uploaded_file(
34             $_FILES['slika']['tmp_name'],
35             "primeri/{$slika_ime}"
36         );
37     }
38 }
```

Veb obrazac u primeru na listingu 37 dozvoljava korisniku aplikacije otpremanje slike. Slika se prilikom prihvata na veb server smešta u privremeni direktorijum u skladu sa podešavanjima u PHP konfiguracionom fajlu, direktiva *upload_tmp_dir*. Po završetku preuzimanja, dokument je potrebno premestiti u direktorijum gde će trajno biti smešten.

Blok linija 29-31 u listingu 37 može da se preuredi na način prikazan na listingu 46. Ukoliko funkcija *provera()* nije detektovala grešku, linija 30, aplikacija formira ime novog dokumenta prema šablonu *slika_broj.png*, linija 32, pri čemu je *broj* trenutni *UNIX timestamp*, a novi dokument se nalazi u direktorijumu *primeri*, linija 35:

- Privremena lokacija i ime otpremljenog dokumenta:

  `C:\xampp\tmp\phpCBEB.tmp`

- Trajna lokacija i ime dokumenta

  `primeri/slika_1614558207.png`

Napokon, upotrebom ovih parametara i funkcije *move_uploaded_file()*, otpremljeni dokument se premešta na trajnu lokaciju, gde će biti dostupan pod imenom koje je aplikacija generisala.

Listing 47 - Uklanjanje dokumenta, funkcija *unlink()*

```php
<?php
$dokument = 'c:/www/singidunum.txt';
if (file_exists($dokument)) {
    if (unlink($dokument)) {
        echo "Dokument {$dokument} je obrisan.";
    } else {
        echo "Dokument {$dokument} nije obrisan.";
    }
} else {
    echo "Dokument {$dokument} ne postoji."
}
?>
```

4.4.4 Brisanje dokumenta

Brisanje dokumenta podrazumeva proces uklanjanja fajla iz odgovarajućeg direktorijuma, nakon čega taj dokument više ne postoji. U PHP programskom jeziku dokument se može ukloniti primenom funkcije *unlink()*, pri čemu se kao parametri funkcije navode putanja i ime dokumenta koji se uklanja, listing 47. Kao rezultat, funkcija će vratiti *true* ukoliko je operacija uspešno izvršena, odnosno *false* u slučaju neuspeha.

Listing 48 - Očitavanje vremena kreiranja, izmene i pristupa dokumentu

```php
<?php
$dokument = 'php-listing.php';
$vreme_kreiran = filectime($dokument);
$vreme_modifikovan = filemtime($dokument);
$vreme_pristup = fileatime($dokument);

echo "Dokument {$dokument} je kreiran {$vreme_kreiran}.<br/>" .
     "Modifikovan je {$vreme_modifikovan}.<br/>" .
     "Vreme poslednjeg pristupa je {$vreme_pristup}.";
?>
```

4.4.5 Manipulacija vremena

Dokument uskladišten na disku nosi informaciju o vremenu

- kada je kreiran
- kada je poslednji put modifikovan
- kada je poslednji put pristupljeno dokumentu

Ovi podaci mogu da se očitaju pomoću funkcija *filectime()*, *filemtime()* i *fileatime()*, pri čemu se u svakoj od njih kao parametar navodi putanja i naziv fajla, listing 48.

Listing 49 - Izmena vremena izmene i pristupa dokumentu

```php
<?php
$dokument = 'php-listing.php';
$vreme = time() - 3600;
if (touch($dokument, $vreme)) {
    echo "Vremenski podaci na dokumentu {$dokument}" .
         "su uspešno izmenjeni";
} else {
    echo "Nešto nije u redu!<br/>" .
         "Vremenski podaci nisu izmenjeni.";
}
?>
```

U slučaju da je potrebno da se promeni vreme poslednje izmene dokumenta ili vreme poslednjeg pristupa dokumentu, koristi se PHP funkcija *touch()*, pri čemu se kao parametri navode ime fajla, novo vreme izmene i opciono novo vreme pristupa, listing 49. Ukoliko se poslednji parametar izostavi, iskoristiće se vrednost navedena u drugom parametru. Ako se izostave i drugi i treći parametar, dodeliće se trenutno sistemsko vreme.

IDENTIFIKACIJA KORISNIKA

5. IDENTIFIKACIJA KORISNIKA

5.1 HTTP KOLAČIĆ

HTTP kolačić (engl. *HTTP cookie*) je običan tekst fajl koji se nalazi na računaru korisnika, a koji postavlja, i kome pristupa veb brauzer prilikom pregleda određenog veb sajta. Vreme trajanja kolačića može biti vremenski ograničeno (npr. 1 sat, 1 dan, 30 dana i sl.) ili neograničeno. U kolačiću se čuvaju informacije vezane za posetu veb sajtu. Kolačići mogu biti:

- *Sesijski kolačić* (engl. *in-memory cookie*) se nalazi u privremenoj memoriji, validan je dok korisnik pregleda veb sajt, a po zatvaranju brauzera se briše. Od običnih HTTP kolačića se razlikuju po tome što nema dodeljen podatak o vremenu trajanja.
- *Trajni kolačić* (engl. *persistant cookie*) se ne briše prilikom zatvaranja brauzera, već ima vremenski ograničen rok trajanja.
- *Bezbedni kolačić* (engl. *secure cookie*) je moguće koristiti samo ukoliko se za pristup veb stranici koristi enkriptovana veza, odnosno HTTPS protokol.
- *Http-only kolačiću* ne mogu da pristupe aplikacije koje se nalaze na klijentskoj strani, kao što su JavaScript aplikacije. Upotreba ovakvih kolačića umanjuje, ali ne isključuje mogućnost krađe kolačića u slučaju XSS napada (engl. *cross-site scripting*).
- *Kolačići treće strane* (engl. *third-party cookie*) pripadaju drugačijem domenu u odnosu na domen sajta koji je trenutno posećen. Savremeni veb brauzeri korisniku ostavljaju mogućnost da dozvoli ili zabrani upotrebu ove vrste kolačića.
- *Zombi kolačići* (engl. *zombie cookie*) su kolačići koji se nakon uklanjanja automatski ponovo kreiraju. Ovo je moguće u slučajevima kada je kreirano više instanci izvornog kolačića na različitim lokacijama, kao što su *Flash Local shared object* ili *HTML5 Web storage*. Najšešće se koriste za analitiku i marketinška istraživanja.

5.1.1 Struktura HTTP kolačića

Kolačić se obično setuje u zaglavlju HTTP zahteva, slika 30, ali ga je moguće kreirati i direktno na klijentskoj strani pomoću JavaScript-a.

Slika 30 - Primer zaglavlja stranice sa setovanim kolačićem

```
HTTP/2.0 200 OK
server: Ring Publishing - Accelerator
date: Tue, 02 Mar 2021 17:21:13 GMT
content-type: text/html; charset=utf-8
content-length: 63389
vary: Accept-Encoding
content-encoding: gzip
link: <https://cdn.consentmanager.mgr.consensu.org/delivery
link: <https://fonts.googleapis.com/css?family=Merriweather
link: <https://ocdn.eu/ucs/static/blicperun/7aa14304b4aba6c
link: <https://cdn-images.mailchimp.com/embedcode/horizonta
link: <https://ocdn.eu>; rel=preconnect
link: <https://events.ocdn.eu>; rel=preconnect
cache-control: no-cache
etag: "63389-509496db68517722"
x-dns-prefetch-control: off
strict-transport-security: max-age=0
x-download-options: noopen
x-xss-protection: 1; mode=block
set-cookie: acc_segment=90; Path=/; Max-Age=604800; Secure
vary: user-agent
X-Firefox-Spdy: h2
```

Kao što se može videti u datom primeru, *set-cookie* u zaglavlju sadrži parove *ime=vrednost* međusobno razdvojene karakterom tačka-zapeta (;), pri čemu je vreme validnosti kolačića podešeno na 7 dana (604800 sekundi). PHP programski jezik podatke o kolačićima čuva u superglobalnoj promenljivoj *$_COOKIE*.

5.1.2 Postavljanje i uklanjanje kolačića

PHP programski jezik omogućava inicijalizaciju kolačića i postavljanje vrednosti pomoću funkcije

```
setcookie(ime, vrednost, ističe,
    putanja, domen, secure,
    httponly, opcije);
```

Listing 50 - Postavljanje kolačića

```php
<?php
$kolacic_ime = 'singidunum';
if (isset($_COOKIE[$kolacic_ime])) {
    echo "Kolačić {$kolacic_ime} je setovan<br/>" .
         " i ima vrednost {$_COOKIE[$kolacic_ime]}";
} else {
    echo "Kolačić {$kolacic_ime} nije setovan!";
    $kolacic_vrednost = 'Beograd';
    $kolacic_istice = time() + 30*30*24*30;
    $kolacic_putanja = '/';
    setcookie(
        $kolacic_ime,
        $kolacic_vrednost,
        $kolacic_istice,
        $kolacic_putanja
    );
}
?>
```

Listing 51 - Smeštanje niza u kolačić

```php
<?php
header('Content-Type: text/html; charset=utf-8');
$kolacic_ime = 'singidunum';
if (!empty($_COOKIE)) {
  $podaci = $_COOKIE[$kolacic_ime] ?? '';
  $podaci = unserialize($podaci);
  if (is_array($podaci) && !empty($podaci)) {
    foreach($podaci AS $k => $v) {
      echo "{$k} => {$v}<br/>";
    }
  }
} else {
  $podaci = [
    'Vežba' => 'Upotreba kolačića',
    'Zadatak' => 'Postavljanje i očitavanje podataka',
    'Vreme' => time()
  ];
  setcookie(
    $kolacic_ime,
    serialize($podaci),
    time()+3600,
    '/'
  );
  echo 'Ovo je prvi ulaz na stranicu.<br/>'.
       'Kolačić nije podešen.<br/>'.
       'Pritisnite taster F5 (Refresh)';
}
?>
```

pri čemu je značenje parametara sledeće:

- *ime* označava ime kolačića, odnosno ključ pod kojim će podaci biti dostupni u superglobalnoj promenljivoj *$_COOKIE*.
- *vrednost* je vrednost koja se dodeljuje kolačiću, čuva se u računaru korisnika i ne treba da sadrži poverljive podatke.
- *ističe* označava vreme (*UNIX timestamp*) kada kolačić ističe.
- *putanja* označava putanju na veb serveru na kojoj će kolačić biti dostupan.

Vrednost putanje '/' znači da će kolačić biti dostupan na celom domenu, dok vrednost '*/slike/*' ograničava dostupnost kolačića samo na direktorijum *slike*.

- *domen* predstavlja domen ili poddomen na kome je kolačić dostupan. Vrednost '*www.singidunum.ac.rs*' će ograničiti dostupnost kolačića na poddomen www i sve njegove poddomene, dok će vrednost '*singidunum.ac.rs*' kolačić učiniti dostupnim na celokupnom domenu i svim poddomenima.
- *secure* može imati vrednost *true* ili *false*, i označava da kolačić treba da se prenosi isključivo putem HTTPS protokola.
- *httponly* može imati vrednost *true* ili *false*. Kada mu je dodeljena vrednost *true*, kolačić neće biti dostupan scripting jezicima kao što je JavaScript.
- *opcije* se zadaju kao asocijativni niz sa ključevima *expires (ističe), path (putanja), domain (domen), secure, httponly* i *samesite*, u skladu sa prethodnim opisom. Ukoliko se u nizu nađe bilo koji drugi ključ osim navedenih, generisaće se greška nivoa *E_WARNING*. Ključ samesite može imati vrednost *None, Lax* ili *Strict*. Vrednost *Lax* će onemogućiti prenos kolačića u slučaju *cross-domain* zahteva, osim kada je u pitanju običan veb link. Zahtevi POST i PUT, kao i XHR zahtevi takođe neće sadržavati ovakav kolačić. Vrednost *Strict* će onemogućiti prenos kolačića u slučaju svih *cross-domain* zahteva, uključujući i veb linkove.

U primeru na listingu 50, prilikom prvog poziva aplikacije kolačić nije postavljen i izvršava se blok linija 7-17. Ime kolačića je *singidunum*, dodeljuje se vrednost *Beograd*, sa rokom trajanja od 30 dana. Prilikom svakog sledećeg poziva stranice aplikacija će prepoznati postavljeni kolačić i izvršavaće se blok linija 3-6, sve dok ne istekne postavljeni rok trajanja kolačića, ili dok se iz veb brauzera ne ukloni postavljeni kolačić.

U primeru na listingu 51, niz deklarisan u bloku linija 13-17 se smešta u kolačić sa imenom *singidunum*, linije 18-23. To je moguće zato što je sadržaj promenljive *$podaci* serijalizovan, linija 20. Prilikom osvežavanja veb stranice, aplikacija proverava da li u kolačiću postoje podaci, linija 4. Pre dalje obrade, ovi podaci moraju biti vraćeni u izvorni oblik deserijalizacijom, linija 6.

U situacijama kada je potrebno da se izmeni sadržaj kolačića, dovoljno je da se ponovo upotrebi fukcija *setcookie()*. Da bi se kolačić uklonio, takođe se koristi fukcija *setcookie()*, pri čemu se vreme isteka kolačića postavlja u prošlost, npr.

```
setcookie('singidunum', '', time() - 3600);
```

5.2 SESIJA

Sesija u PHP programskom jeziku predstavlja mehanizam čuvanja određenih podataka u superglobalnoj promenljivoj *$_SESSION* prilikom višestrukih, vezanih poseta veb aplikaciji. Posetiocu se dodeljuje jedinstveni identifikator sesije (engl. *session id*), koji se čuva u kukiju, ili mnogo ređe, kao URL parametar.

Za svaku sesiju, PHP generiše dokument u privremenom direktorijumu, u skladu sa podešavanjem u php.ini konfiguracionom fajlu, odnosno u skladu vrednošću dodeljenoj direktivi *session.save_path*.

5.2.1 Upravljanje sesijama

Prilikom pokretanja sesije, PHP izvodi sledeće operacije:

- Kreira jedinstveni identifikator koji dodeljuje toj sesiji. U pitanju je slučajno generisan string dužine 32 karaktera.
- Korisniku se automatski šalje kolačić sa imenom PHPSESSID, u kome se nalazi identifikacioni string sesije.
- Na veb serveru se u privremenom direktorijumu automatski kreira dokument, čije ime se sastoji od prefiksa sess_ i identifikacionog stringa sesije, npr. sess_3d4foj34c3qw973mnoop2fc937e44421.

Listing 52 - Praćenje broja poseta istog korisnika putem sesije

```php
<?php
session_start();
if (isset($_SESSION['brojac'])) {
    $_SESSION['brojac']++;
} else {
    $_SESSION['brojac'] = 1;
}
echo "Ovu stranicu ste posetili ".
     "{$_SESSION['brojac']} puta ".
     "u toku ove sesije.";
?>
```

Sesija se pokreće primenom PHP funkcije *session_start()*, pri čemu se opciono kao parametar može navesti asocijativni niz sa odgovarajućim ključevima i vrednostima[14]. PHP će prvo pokušati da u kolačiću PHPSESSID očita identifikacioni string sesije. Ukoliko taj podatak postoji, u privremenom direktorijumu će potražiti fajl sa odgovarajućim imenom. Nakon što je sesija uspešno pokrenuta, može se pristupiti podacima koji se nalaze u superglobalnoj promenljivoj *$_SESSION*.

U primeru na listingu 52, nakon pokretanja sesije, linija 2, proverava se da li u nizu *$_SESSION* postoji beleška sa ključem brojac, linije 3. Ukoliko postoji, vrednost će joj biti uvećana za 1, linija 4. U suprotnom, ako *$_SESSION['brojac']* ne postoji, biće inicijalizovan sa početnom vrednošću 1, linija 6. Na kraju se ispisuje trenutno stanje brojaca, linije 8-10.

14 https://www.php.net/manual/en/session. configuration.php

5.3 PREPOZNAVANJE KORISNIKA

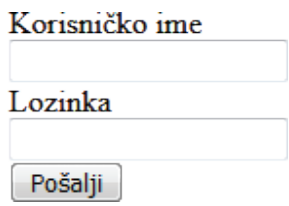
Primer aplikacije koja omogućа identifikaciju korisnika pomoću sesije je prikazan na listingu 53. Kompletan kod (*backend* i *frontend*) se nalazi u jednom fajlu čije ime može da bude *login.php*. Mehanizam prepoznavanja korisnika može da se modelira kroz sledeće korake:

Listing 53 - Prijava i odjava korisnika veb aplikacije

```php
<?php
$ime_skripte = $_SERVER['PHP_SELF'];
$poruka = '';
session_start();
if (@$_SESSION['ulogovan']) {
  if (@$_GET['izadji'] == 1) {
    $_SESSION['ulogovan'] = 0;
    header("location: $ime_skripte");
    exit;
  }
} else {
  if ($_POST) {
    if (
      $_POST['ime'] == 'admin' &&
      $_POST['lozinka'] == 'admin'
    ) {
      $_SESSION['ulogovan'] = 1;
    } else {
      $poruka = 'Podaci za pristup nisu dobri!';
    }
  }
}
?>
<html>
<head>
  <meta charset="utf8">
</head>
<body>
  <?php if (@$_SESSION['ulogovan'] == 1): ?>
    Ulogovani ste. <a href="?izadji=1">Odjavite se.</a>
  <?php else: ?>
    <?php if ($poruka): ?>
      <?php echo $poruka; ?>
    <?php endif; ?>
    <form method="post">
      Korisničko ime<br/>
      <input name="ime" value="<?php echo @$_POST['ime']; ?>"><br/>
      Lozinka<br/>
```

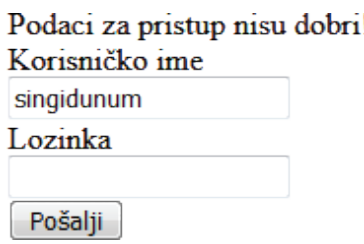
```
39        <input name="lozinka" type="password"><br/>
40        <button type="submit">Pošalji</button>
41      </form>
42    <?php endif; ?>
43 </body>
44 </html>
```

Slika 31 - Veb obrazac za prijavu korisnika



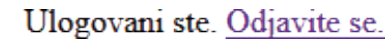
- Informacija o statusu korisnika se nalazi u superglobalnoj promenljivoj *$_SESSION*, pod ključem *ulogovan*. Da bi aplikacija mogla da pristupi ovom podatku, neophodno je pokrenuti sesiju, linija 4. Ukoliko korisnik nije prijavljen, na ekranu se prikazuje veb obrazac za unos korisničkog imena i lozinke, slika 31, linije 35-41.

Slika 32 - Uneti podaci nisu tačni. Prikaz greške iznad obrasca za prijavu.



- Nakon klika na dugme *Pošalji*, uneti podaci se šalju veb serveru, gde ih prihvata isti dokument. Aplikacija pokreće sesiju, i u slučaju da korisnik nije prijavljen, proverava poslate podatke, linije 12-21. U slučaju da su oba podatka tačna, status korisnika se menja na ulogovan, linija 17. U suprotnom, opet prikazuje veb obrazac za prijavu, uz koji prikazuje i odgovarajuću grešku, slika 32.

Slika 33 - Korisnik je ulogovan



- Kada je korisnik ulogovan, provera u liniji 29, na stranici su prikazani samo odgovarajuća poruka i link *Odjavite se,* slika 33. Link sadrži URL parametar *izadji* kome je dodeljena vrednost 1.
- Klikom na link *Odjavite se*, ponovo se poziva isti dokument. Ovoga puta, aplikacija proverava da li je korisnik ulogovan, linija 5, i da li je vrednost URL parametra *izadji* jednaka *1*, linija 6. Ukoliko su oba uslova ispunjena, vrednost ključa *ulogovan* u sesiji se vraća na *0*, i vrši se redirekcija na istu veb stranicu, ali ovoga puta bez URL parametra *izadji*.

RAD SA OBJEKTIMA

6. RAD SA OBJEKTIMA

U ovom poglavlju su date osnove objektno-orijentisanog programiranja (engl. *Object-Oriented Programming, OOP*) u PHP jeziku, sa primerima nasleđivanja i interfejsa. Pretpostavka je da su studenti tokom studija već upoznati sa objektno-orijentisanim programiranjem i zbog toga su udžbeniku prikazane samo osnove koje se odnose na objektno-orijentisano programiranje u PHP jeziku.

6.1 KLASE I OBJEKTI

Klasa je osnovna jedinica programiranja u objektno-orijentisanim programskim jezicima. Osnovni elementi klasa su:

- *atributi*, osobine, svojstva, ili promenljive koje opisuju objekat
- *metode,* operacije, funkcije, ili radnje koje objekat može da izvršava
- *konstruktori,* mehanizmi za kreiranje objekata na osnovu definicije

Sintaksa definisanja klase je prikazana na listingu 54

Listing 54 - Sintaksa definisanja klase

```
<?php
classtestKlasa{
[[deklaracijaOsobine1] osobina1;]
[[deklaracijaOsobine2] osobina2;
…]
[[deklaracijaMetode1] metoda1;]
[[deklaracijaMetode2] metoda2;
…]}
?>
```

Objekat je primerak, pojava ili instanca klase. PHP podržava objekte kao strukturu podataka. Objekti se definišu pomoću funkcija. Za svaki objekat su vezane osobine (atributi) i metode.

Osobine objekata su promenljive kojima se dodaju određene vrednosti. Metode su funkcije u okviru objekata pomoću kojih mogu da se menjaju njihove osobine. Objekti u se u PHP-u definišu pomoću operatora new.

```
testObjekat = new testKlasa;
```

Neki od osnovnih principa OOP su:

- *Polimorfizam*, svaki objekat izvedene klase izvršava operaciju onako kako je to definisano u njegovoj osnovnoj klasi.
- *Nasleđivanje*, jedna klasa plus jedna, ili više podklasa. Omogućava nadgradnju i proširenje postojećih klasa. Koncept nasleđivanja je detaljnije opisan u nastavku.

6.1.1 Modifikatori pristupa

Kao i u nekim drugim programskim jezicima treće generacije, PHP inkorporira modifikatore pristupa (engl. *access modifiers*). Modifikatori pristupa atributa (osobina) mogu da budu:

- *public (javni)* – vrednostatributamože direktno da se menja sa bilo kog mesta u kodu;
- *private (privatni)* - vrednostatributamože da se menja samo preko metode objekta;
- *protected (zaštićeni)* - atribut je nepoznat van objekta i
- *final (konačni)* - vrednost atributa ne može da se menja u klasama koje nasleđuju datu klasu.

Slično, modifikatori pristupa metoda mogu da budu:

- *public (javni)* - metod može da se koristi van svoje klase pozivanjem metode instance;
- *private (privatni)* - rezultati metoda se koriste samo u okviru objektal;
- *protected (zaštićeni)* - rezultati metoda se koriste samo u okviru objekta ili nekog od naslednika klase čiji je objekat definisan;
- *abstract (apstraktni)* - metoda je deklarisana samo u roditeljskojklasii
- *final (konačni)* - telo metode ne može da se menja u klasama naslednicima.

6.1.2 Dodela vrednost atributa

Vrednost atributa (osobine) objekta (instance) neke klase može da se dodeli promenljivoj na sledeći način:

```
promenljiva = testObjekat->Atribut1;
```

Vrednost atributa (osobine) objekta (instance) neke klase može da se promeni, ili naknadno da se definiše ukoliko nije bila ranije definisana na sledeći način:

```
testObjekat->Atribut1 = vrednost;
```

6.1.3 Deklarisanje klase i instanciranje objekta

Primer deklarisanja klase i instanciranje objekta prikazan je na listingu 55, gde je kreirana klasa Pravougaonik sa dva atributa, koji su redom *$strana1* i *$strana2*. Korišćena je metoda konstruktor, koja omogućava definisanje atributa objekta odmah pri njegovom kreiranju. U konstruktoru se koristi ključna reč *$this* koja upućuje na tekući objekat, a u kodu iz primera označava "vrednost atributa *strana1* objekta klase *Pravougaonik*, postavi na vrednost promenljive *$a*, koja se prosleđuje prilikom kreiranja objekta".

Klasa iz primera ima dve metode, *izracunajObim()* i i*zracunajPovrsinu()*. Ove metode nemaju argumente. Obe metode koriste modifikator pristupa *public*, što znači da su vidljive van klase.

Objekti (instance) se kreiraju van definicije klase,

```
$mojObjekat = new mojaKlasa();
```

Listing 55 - Deklarisanje klase i instanciranje objekta - I

```
1 <?php
2 class Pravougaonik {
3   // definisanje atributa
4   public $strana1;
5   public $strana2;
6   // kreiranje konstruktora
7   function __construct($a,$b) {
8     $this->strana1=$a;
9     $this->strana2=$b;
10  }
11  //metoda za izračunavanje obima
12  public function izracunajObim() {
13    return (($this->strana1 + $this->strana2)*2);
14  }
15  //metoda za izračunavanje povrsine
16  public function izracunajPovrsinu() {
17    return ($this->strana1*$this->strana2);
18  }
19 }
20 // kreiranje instance klase Pravouganik
21 $primerPravougaonika = new Pravougaonik(20, 20);
22 echo "Povrsina je: " . $primerPravougaonika->izracunajPovrsinu();
23 echo "<br/>";
24 echo "Obim je: " . $primerPravougaonika->izracunajObim();
25 echo "<br/>";
26 ?>
```

što je ekvivalentno liniji 21 u primeru na listingu 55. Prilikom poziva metode koristi se sledeća sintaksa:

```
$mojObjekat->mojaMetoda();
```

kako je prikazano u linijama 22 i 24 u priloženom primeru.

Na listingu 56 naveden je još jedan primer deklarisanja klase i kreiranja objekta.

Da bi se pozvala konstanta klase kojoj objekat pripada, objekat ne mora da se instancira, odnosno nije ga potrebno kreirati, listing 57.

Listing 56 - Deklarisanje klase i instanciranje objekta - II

```php
<?php
class Osoba{
  public $ime;
  // metoda za postavljanje vrednosti u atribut ime
  public function postaviIme($ime) {
    $this->ime = $ime;
  }
  public function prikaziIme() {
    return $this->ime;
  }
}
$ja = new Osoba();
$ja->postaviIme("Joe Doe");
echo "Moje ime je: " . $ja->prikaziIme();
?>
```

Listing 57 - Korišćenje konstante u deklarisanju klase

```php
<?php
class Konstanta {
  const PI=3.14159;
}
echo "Vrednost konstante PI je: " . Konstanta::PI;
?>
```

6.2 NASLEĐIVANJE, INTEFEJSI I STATIČKI ATRIBUTI I METODE

Nasleđivanje (engl. *inheritance*) je jedan od osnovnih koncepata OOP. Prema jednoj od raspoloživih definicija, nasleđivanje se definiše kao proces kreiranja novih klasa korišćenjem postojećih klasa. Kada jedna klasa nasleđuje drugu klasu, nova klasa (ona koja nasleđuje, klasa dete) nasleđuje i sve atribute i metode klase koju nasleđuje (roditeljska klasa). Klasa koja je nasleđena naziva se bazna klasa (klasa roditelj), dok se klasa koja nasleđuje naziva izvedena klasa (klasa dete).

U izvedenoj klasi mogu da se dodaju novi atributi i metode koje nisu bile definisane u baznoj klasi, a takođe mogu da se redefinišu (engl. *override*) metodi koji su već definisani u baznoj klasi. U literaturi se bazna klasa još naziva i *natklasa*, dok se izvedena klasa naziva *podklasa*.

Listing 58 - Nasleđivanje klasa i instanciranje objekata

```php
<?php
class A {
  // linije koda
}
class B extends A {
  // linije koda
}
class C extends B {
  // linije koda
}
// načini instanciranja objekta
$someObj = new A();
$someOtherObj = new B();
$lastObj = new C();
?>
```

PHP podržava nasleđivanje u više nivoa (engl. *multi-level inheritance*), što znači da klasa može da nasledi klasu koja je već nasledila neku drugu klasu. Međutim, u PHP-u jedna klasa ne može da nasledi dve ili više klasa, već samo jednu. Nasleđivanje se definiše pomoću rezervisane reči *extends*, listing 58.

Često se javlja potreba da se pozove metoda klase roditelj iz klase dete, za šta se koristi sledeća sintaksa:

```
parent::metoda();
```

Dva primera nasleđivanja data su u listinzima 59 i 60.

Listing 59 - Primer nasleđivanja - I

```php
<?php
class JednostavnaKlasa {
  public $promenljiva = "ovo je vrednost iz klase roditelj";
  public function prikazi() {
    // prikazuje vrednost atributa promenljiva
    echo $this->promenljiva;
  }
}
// klasa ProsirenaKlasa nasledjuje klasu JednostavnaKlasa
class ProsirenaKlasa extends JednostavnaKlasa {
  /* ova klasa redefinise metodu prikazi()
     u roditeljskoj klasi */
  public function prikazi() {
    echo "pozivamo metodu iz roditeljske klase" .
         "sa istim nazivom <br/>";
    parent::prikazi();
  }
}
// kreiramo objekat klase ProsirenaKlasa
$primer=new ProsirenaKlasa();
// pozivamo metodu prikazi() objekta $primer
$primer->prikazi();
?>
```

Listing 60 - Primer nasleđivanja - II

```php
<?php
class Vozilo {
  public $boja;
  public $tip;
}
class Automobil extends Vozilo {
  public function __construct() {
    $this->tip="automobil";
  }
  public function postaviBoju($boja) {
    $this->boja=$boja;
  }
  public function prikaziBoju() {
    return $this->boja;
  }
}
class Kamion extends Vozilo {
  public function __construct() {
    $this->tip="kamion";
  }
  public function prikaziTip() {
    echo "Ovo je kamion";
  }
}
$automobil=new Automobil();
$automobil->postaviBoju("teget");
echo "Boja automobila je: " . $automobil->prikaziBoju();
echo "<br/>";
$kamion=new Kamion();
$kamion->prikaziTip();
?>
```

6.2.1 Sprečavanje nasleđivanja

U PHP skripti često se može javiti potreba za sprečavanje nasleđivanja. Sprečavanje nasleđivanja vrši se pomoću ključne reči *final*. Kada se ključna reč *final* stavi ispred deklaracije klase, klasa ne može biti nasleđena.

Slično, kada se ključna reč *final* navede ispred deklaracije metode, ta metoda ne može da se redefiniše istoimenom metodom ni u jednoj potklasi koja nasleđuje klasu kojoj metoda pripada.

Pseudo-kod sprečavanja nasleđivanja klase i metode naveden je na listingu 61.

Listing 61 - Sprečavanje nasleđivanja klase i metode

```php
<?php
final function metoda1() {
  //naredbe
}
final class mojaKlasa {
  //atributi
  //metode
}
?>
```

6.2.2 Interfejsi

Kao što je već i napomenuto, PHP jezik ne podržava višestruko nasleđivanje, tj. jedna klasa ne može da nasledi više drugih klasa, već samo jednu. Interfejsi se koriste da bi se zaobišao "problem" nemogućnosti višestrukog nasleđivanja.

Interfejs deklariše određen broj metoda, koje moraju da budu realizovane (implementirane) u svim klasama koje realizuje (implementira) taj interfejs.

Interfejs sadrži samo deklaraciju metoda, bez njihove implementacije, tj. bez tela metoda.

```
interface MojInterfejs {
    function prikazi();
}
```

Listing 62 - Implementiranje interfejsa

```php
<?php
interface Car {
  public function setModel($name);
  public function getModel();
}
class myCar implements Car {
  private $model;
  // set metoda koja postavlja vrednost private atributa
  public function setModel($name) {
    $this -> model = $name;
  }
  // get metoda koja prikazuje vrednost private atributa
  public function getModel() {
    return $this -> model;
  }
}
$car=new myCar();
$car->setModel("Volvo");
echo "Ja vozim " . $car->getModel();
?>
```

Kaže se da klasa implementira interfejs. Klasa koja implementira interfejs "se obavezuje" da implementira sve metode koje su date u interfejsu. Za implementiranje koristi se rezervisana reč *implements*. Sintaksa je prikazana u nastavku.

```
class MojaKlasa implements MojInterfejs{
}
```

Primer koji ilustruje upotrebu interfejsa u PHP-u dat je u listingu 62.

Listing 63 - Statički atributi i metodi

```
<?php
class Poseta{
  private static $posetioci = 0;
  function __construct(){
    // pomocu naredbe self:: poziva se staticki atribut
    self::$posetioci++;}
    static function izbrojPosetioce(){
    return self::$posetioci;
  }
}
$posete = new Poseta();
echo Poseta::izbrojPosetioce() . "<br />";
$posete2 = new Poseta();
echo Poseta::izbrojPosetioce() . "<br />";
echo $posete->izbrojPosetioce();
?>
```

6.3 STATIČKI ATRIBUTI I METODE

Bez obzira što su studenti već upoznati sa statičkim atributima i metodama, na kraju dela udžbenika koji se bavi objektno-orijentisanim programiranjem u PHP programskom jeziku, korisno je napraviti kraći osvrt i na ove koncepte.

Statički atributi i metode mogu da se pozivaju bez kreiranja objekta, a deklarišu se korišćenjem rezervisane reči *static*. Sintaksa deklarisanja statičke metode prikazana je ispod.

```
class mojaKlasa {
static function izbroj(){
};
}
```

Statička metoda poziva se na sledeći način:

```
mojaKlasa::izbroji();
```

RAD SA BAZOM PODATAKA

7. RAD SA BAZOM PODATAKA

Obzirom na pretpostavku da su studenti na prethodnim kursevima na studijskim programima Univerziteta Singidunum izučavali materiju baza podataka i da su upoznati sa definicijama baza podataka, procesima logičkog i fizičkog modelovanja, kao i sa osnovnim DDL (engl. *Data Definition Language*), DML (engl. *Data Manipulation Langauge*) i DCL (engl. *Data Control Language*) SQL naredbama, u ovoj glavi navedeni koncepti nisu obuhvaćeni. Više informacija o osnovama baza podataka može se naći u udžbeniku za predmet *Baze podataka*[15].

U prvom delu ovog poglavlja prikazane su osnovne SQL naredbe iz familije DCL, koje omogućavaju kreiranje, brisanje i manipulisanje privilegijama i korisnicima sistema za upravljanje bazom podataka.

U drugom delu poglavlja prikazane su sekvencijalne (proceduralne) i objektno-orijentisane PHP naredbe za povezivanje sa bazom, kreiranje objekata u bazi, kao i za prikaz i manipulisanje podacima iz baze podataka. U svim primerima korišćen je MySQL sistem za upravljanje relacionim bazama podataka[16].

7.1 PRIVILEGIJE I SQL DCL NAREDBE

Razumevanje privilegija je neophodno za implementaciju bilo koje troslojne aplikacije (aplikacije koja koristi bazu podataka) zato što je, u skladu sa principom najmanjih privilegija (engl. *least privilege privilege*), za svaku aplikaciju koja se hostuje na veb serveru potrebno kreirati posebnu bazu podataka i korisnika koji ima sva ovlašćenja nad svim objektima date baze (tabelama, sekvencijama, indeksima i slično).

U scenarijima kada više Veb aplikacija koristi isti server baze podataka, u praksi se nažalost dešava da sve aplikacije pristupaju svojoj bazi podataka korišćenjem root naloga, koji ima privilegije nad svim objektima u svim bazama podataka koje se čuvaju na datom serveru baze podataka. Implementacija okruženja na ovaj način predstavlja ozbiljan sigurnosni propust. Tako na primer, ako jedna od aplikacija ima sigurnosni propust, koji može malicioznog napadača da dovede do informacija o root nalogu, onda će sve aplikacije čija se baza čuva na datom serveru baze podataka biti kompromitovane.

15 Veinović, M., Šimić, G., Jevremović, A., Tair, M., *Baze podataka,* Univerzitet Singidunum, Beograd, 2018.
16 Tehnička dokumentacija za korišćenje MySQL baze nalazi se na adresi https://dev.mysql.com/doc/

Zbog robusnosti, lakoće upotrebe i rasprostranjenosti, u svim primerima koji su dati u ovoj glavi udžbenika biće korišćen MySQL server. Kao što je već i naglašeno, MySQL je jedan od najpoznatijih i najviše korišćenih *open-source* sistema za upravljanje relacionim bazama podataka. Kao neke od značajnijih karakteristika MySQL-a navode se sledeće: upotreba i korišćenje ove baze podataka je besplatno, MySQL komande su neosetljive na veličinu slova (engl. *case insensitive*), ali je praksa da se pišu velikim slovima i promenljive su osetljive na veličinu slova.

Pozicija MySQL-a u rangiranju najšire korišćenih sistema za upravljanje bazama podataka prikazana je na slici 34.

Definisanje korisničkih privilegija vrši se pomoću SQL data control langauge (DCL) naredbi. Da bi korisnik mogao da izvrši bilo kakve funkcije nad bazom podataka (kreiranje tabela, sekvenci, pogleda, itd.), korisnik mora da ima određene privilegije. Postoje dve vrste privilegija:

- *sistemske* (engl. *system level privileges*). Ove privilegije obuhvataju dozvole (engl. *permissions*) za kreiranje tabela, sesija, i drugih sistemskih privilegija i
- *objektne* (end. *object privileges*). Ove privilegije obuhvataju dozvole za izvršavanje bilo koje komande ili upita nad tabelama, kao i nad drugim objektima baze podataka.

Slika 34 - Rangiranje sistema za upravljanje bazama podataka[17]

Rank			DBMS	Database Model	Score		
Mar 2021	Feb 2021	Mar 2020			Mar 2021	Feb 2021	Mar 2020
1.	1.	1.	Oracle	Relational, Multi-model	1321.73	+5.06	-18.91
2.	2.	2.	MySQL	Relational, Multi-model	1254.83	+11.46	-4.90
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1015.30	-7.63	-82.55
4.	4.	4.	PostgreSQL	Relational, Multi-model	549.29	-1.67	+35.37
5.	5.	5.	MongoDB	Document, Multi-model	462.39	+3.44	+24.78
6.	6.	6.	IBM Db2	Relational, Multi-model	156.01	-1.60	-6.55
7.	7.	↑ 8.	Redis	Key-value, Multi-model	154.15	+1.58	+6.57
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model	152.34	+1.34	+3.17
9.	9.	↑ 10.	SQLite	Relational	122.64	-0.53	+0.69
10.	↑ 11.	↓ 9.	Microsoft Access	Relational	118.14	+3.97	-7.00

17 https://db-engines.com/en/ranking

Najpoznatije *DCL* naredbe su *GRANT* i *REVOKE*. Naredba *GRANT* koristi se za dodeljivanje pristupnih i drugih privilegija, dok se naredba *REVOKE* korist i za oduzivanje pristupnih i drugih privilegija za određenu bazu podataka. Skup privilegija sysdba obuhvata sve privilegije.

Primer gde se korisinku dodeljuje privilegija za kreiranje bilo koje tabele u bazi određenoj bazi podataka dat je ispod.

```
GRANT CREATE ANY TABLE TO username
```

Primer gde se korisniku oduzima privilegija za kreiranje bilo koje tabele u bazi dat je u nastavku.

```
REVOKE CREATE TABLE FROM username
```

Kao što je poznato, baze podataka sastoje se iz tabela, koje se sastoje iz kolona i redova. Korisnici mogu pristupati ovim bazama, tabelama i kolonama u zavisnosti od svojih privilegija koje imaju nad tabelama i drugim objektima u bazi. Pristup može biti radi kreiranja baze, brisanja baze, dodavanja ili menjanja informacija u bazi, brisanja informacija iz baze, itd.

Za prisutp MySQL bazi i konfigurisanje korisničkih privilegija dostupni su razni klijenti, a neki od njih obuhvataju sledeće:

- *mysql klijent* podrazumeva šel komandni interfejs (engl. s*hell command interface*), pomoću kojeg može da se pristupa i konfiguriše baza podataka;
- *mysqladmin* klijent se prvenstveno fokusira na izvršavanje komandi koje zahtevaju administratorska ovlašćenja;
- *MySQL Administrator* je GUI klijent razvijen od strane mySQL tima;
- *phpMyAdmin* je GUI klijent prilagođen PHP-u i
- MySQL WorkBench.

Na nivou MySQL sistema privilegije se deifnišu u MySQL meta bazi podataka, u kojoj se čuvaju podaci o svim ostalim bazama koje se hostuju na MySQL serveru. Ova baza se sastoji iz sledećih tabela:

- *user* - određuje korisnike koji mogu da se loguju na server sa bilo kog hosta i njihove privilegije;
- *db* - određuje koji korisnici mogu da pristupe kojim bazama podataka;
- *tables_priv* - određuje koji korisnici mogu da pristupe kojim tabelama date baze;

- *columns_priv* - određuje koji korisnici mogu da pristupe kojim kolonama date tabele date baze
- *procs_priv* - čuva informacije o uskladištenim procedurama (engl. *stored procedures*) i funkcijama (engl. *stored functions*).

U tabelama 8-12 prikazane su strukture navedenih tabela MySQL meta baze podataka.

Administrator baze podataka kreira i briše korisnike, dodeljuje i oduzima privilegije, pomoću sledećih komandi:

- CREATE USER
- DROP USER
- GRANT i
- REVOKE

Privilegije su zapravo komande kojima se vrše operacije nad bazama podataka, poput modifikacije baze podataka, čitanje podataka iz baze, upisivanje podataka u bazu, ažuriranje, kao i brisanje podataka iz baze.

Sintaksa za naredbu kojom se kreira novi korisnik bez privilegija:

```
CREATE USER korisnik1 [IDENTIFIED BY [PASSWORD] 'lozinka1']
  [, korisnik2 [IDENTIFIED BY [PASSWORD] 'lozinka2']…]
```

Sintaksa kojom se uklanja postojeći korisnik:

```
DROP USER korisnik1 [, korisnik2…]
```

Tabela 8 - Struktura MySQL tabele *user*

Column	Datatype	Null	Default
Host	char(60) binary	No	No
User	char(16) binary	No	No
Password	char(41) binary	No	No
Select_priv	enum('N', 'Y')	No	N
Insert_priv	enum('N', 'Y')	No	N
Update_priv	enum('N', 'Y')	No	N
…	…	…	…

Tabela 9 - Struktura MySQL tabele *db*

Column	Datatype	Null	Default
Host	char(60) binary	No	No
Db	char(64) binary	No	No
User	char(16) binary	No	No
Select_priv	enum('N', 'Y')	No	N
Insert_priv	enum('N', 'Y')	No	N
Update_priv	enum('N', 'Y')	No	N
…	…	…	…

Tabela 10 - Struktura MySQL tabele *tables_priv*

Column	Datatype	Null	Default
Host	char(60)	No	No
Db	char(64)	No	No
User	Char(16)	No	No
Table_name	char(64)	No	No
Grantor	Char(77)	No	No
Timestamp	timestamp	Yes	Current timestamp
Table_priv	*skupPrivilegija*	No	No

Tabela 11 - Struktura MySQL tabele *columns_priv*

Column	Datatype	Null	Default
Host	char(60)	No	No
Db	char(64)	No	No
User	Char(16)	No	No
Table_name	char(64)	No	No
Column_name	Char(64)	No	No
Timestamp	timestamp	Yes	Null
Column_priv	*skupPrivilegija*	No	No

Tabela 12 - Struktura MySQL tabele *procs_priv*

Column	Datatype	Null	Default
Host	char(60)	No	No
Db	char(64)	No	No
User	Char(16)	No	No
Routine_name	char(64)	No	No
Routine_type	enum	No	No
Grantor	Char(77) binary	No	No
Proc_priv	*skupPrivilegija*	No	No

Primer kreiranja i brisanja korisnika dat je u listingu 64.

Listing 64 - Primer kreiranja i brisanja korisnika

```
-- CREATE I DROP USER
-- kreira se korisnik nebojsa na lokalnom računaru
CREATE USER 'nebojsa'@'localhost' IDENTIFIED BY 'sifra';
-- brise se korisnik nebojsa sa lokalnog računara
DROP USER 'nebojsa'@'localhost';
CREATE USER 'nebojsa'@'localhost' IDENTIFIED BY 'sifra';
```

Korisničke privilegije dodeljuju se naredbom *GRANT* koja ima sledeću sintaksu, listing 65:

Listing 65 - Dodela privilegija naredbom *GRANT*

```
GRANT tipPrivilegije1
      [(listaKolona1)]
      [, tipPrivilegije2 [(listaKolona2)] ...]
ON {imeTabele | * | *.* | imeBaze.*}
TO korisnik1 [IDENTIFIED BY 'lozinka1']
   [, korisnik2 [IDENTIFIED BY 'lozinka2'] ...]
[REQUIRE {NONE|SSL|X509|ISSUER sertifikacionoTelo}]
[WITH opcija]
```

U sledećem primeru korisniku *nebojsa@localhost* biće dodeljene *update* i *delete* privilegije za sve tabele u bazi podataka *sii*.

```
GRANT UPDATE, DELETE ON sii. * TO 'nebojsa'@''localhost';
```

Korisničke privilegije se oduzimaju naredbom *REVOKE*, čija je sintaksa prikazana ispod.

```
REVOKE tipPrivilegije1 [(listaKolona1)] [, tipPrivilegije2
[(listaKolona2)] ...]
ON {imeTabele | * | *. * | imeBaze. *}
FROM korisnik1 [, korisnik2 ...]
```

U sledećem primeru je demonstrirano oduzimanje privilegije *DELETE* od korisnika nebojsa@localhost nad svim objektima baze *sii*:

```
REVOKE DELETE ON sii. * FROM 'nebojsa'@'localhost';
```

Od mnogobrojnih dostupnih korisničkih privilegija[18], najčešće se koriste sledeće:

- `SELECT` dozvoljava korisnicima da selektuju (čitaju) zapise iz tabela;
- `INSERT` dozvoljava korisnicima da upisuju nove zapise (redove) u tabele;
- `UPDATE` omogućava korisnicima da menjaju postojeće zapise u tabelama;
- `DELETE` dozvoljava korisnicima da brišu zapise (redove) iz tabela;
- `INDEX` omogućava korisnicima da kreiraju i manipulišu indeksima;
- `CREATE` dozvoljava kreiranje baza, tabela i/ili drugih objekata;
- `ALTER` omogućava korisnicima da menjaju strukturu baze podataka:
- dodavanje novih kolona
- promena imena kolona ili tabela
- promena tipa podataka.
- `DROP` dozvoljava brisanje baza podataka ili tabela

Osim navedenih, korisno je da se spomene i *USAGE* privilegija, koja daje minimalne privilegije. Ova opcija je korisna ako je potrebno samo da se kreira novi korisnik. Među ostalim privilegijama navode se *CREATE USER, PROCESS* i *SHOW DATABASES*.

18 Kompletna lista korisničkih privilegija može da se vidi na sledećem URL-u: https://dev. mysql.com/doc/refman/8. 0/en/privileges-provided. html.

7.2 PHP NAREDBE ZA RAD SA BAZOM PODATAKA

U PHP bibliotekama dostupno je više proceduralnih funkcija, kao i više klasa za rad sa MySQL bazama podataka. Dakle, u opštem smislu, prilikom razvoja višeslojne PHP aplikacije, povezivanje PHP koda sa bazom podataka može da se implementira korišćenjem proceduralne i/ili objektno-orijentisane paradigme. Među najvažnijim izdvajaju se *mysqli* funkcije[19] i *mysqli* klase.

Takođe treba spomenuti da je od PHP verzije 5.1 dostupna i grupa naredbi iz *PDO* (engl. *PHP Data Objects*) familije.

U ovom delu udženika prikazan je rad sa MySQL komandama korišćenjem proceduralne i objektno-orijentisane paradigme.

7.2.1 Proceduralna paradigma za rad sa bazom podataka

Među najvažnijim funkcijama za rad sa MySQL bazama podataka koje su dostupne u PHP bibliotekama izdvajaju se sledeće:

- mysqli_connect(),
- mysqli_close(),
- mysqli_select_db(),
- mysqli_query(),
- mysqli_db_query(),
- mysqli_fetch_array(), itd.

Za povezivanje sa serverom baza podataka ili bazom podataka na serveru koristi se *mysqli_connect()* funkcija koja vraća referencu ka objektu baze podataka. Sintaksa:

```
link=mysqli_connect(host[, korisnik[, lozinka[, baza]])
mysqli_close(link)
```

Primer povezivanja sa serverom baze podataka, koji se nalazi na lokalnom računaru, gde se kao kredencijali koriste korisničko ime *nebojsa* i lozinka *sifra*, naveden na listingu 66.

19 Kompletna lista sa pratećom dokumentacijom mysqli funkcija može se naći na sledećem URL-u: http://www.php.net/manual/en/class. mysqli.php

Listing 66 - Povezivanje sa serverom baze podataka, funkcija *mysqli_connect()*

```php
<?php
/* konekcija i zatvaranje konekcije ka
   lokalnoj bazi pomoću mysqli_connect funkcije */
$link=mysqli_connect("localhost", "nebojsa", "sifra");
if(mysqli_close($link))
    echo "Konekcija je zatvorena" . "<br/>";
```

Za izvršavanje SQL upita, bilo da su u pitanju DDL ili DML SQL naredbe koristi se funkcija *mysqli_query()*. Sintaksa:

```
mysqli_query (link [, upit])
```

Primer sa kreiranjem i brisanjem baze podataka *sii* na lokalnom MySQL serveru dat je u listingu 67.

Listing 67 - Kreiranje i brisanje baze na lokalnom računaru, funkcija *mysqli_query()*

```php
<?php
/* kreiranje i brisanje baza: u nastavku se deklarišu
   varijable sa kredencijalima za povezivanje na lokalni
   server baze podataka */
$user='nebojsa';
$password='sifra';
$host='localhost';
$link=mysqli_connect($host,$user,$password) or die();
echo "konektovani smo <br/>";
$sql='CREATE DATABASE
      IF NOT EXISTS sii
      DEFAULT COLLATE utf8_general_ci';
mysqli_query($link,$sql) or die();
echo "baza sii kreirana<br/>";
// brisanje baze
$sql='DROP DATABASE sii';
mysqli_query($link,$sql) or die();
echo "baza sii je obrisana <br/>";
?>
```

Listing 68 - Kreiranje tabela, funkcija *mysqli_query()*

```php
<?php
$link = mysqli_connect("localhost", "nebojsa", "sifra", "sii")
  die('Greska! Konekcija na bazu nije moguća');
echo "Konekcija je uspostavljena <br/>";
// brisanje tabele professors i departments ako postoje
$sql='DROP TABLE IF EXISTS professors';
mysqli_query($link,$sql) or die();
$sql='DROP TABLE IF EXISTS departments';
mysqli_query($link,$sql) or die();
// Upiti za kreiranje tabela
$sql_d = 'CREATE TABLE departments
  (departmentID int AUTO_INCREMENT,
   departmentName varchar(255),
   departmentLocation varchar(255) NOT NULL,
   CONSTRAINT PK_departments PRIMARY KEY (departmentID)
  )';
$sql_p = 'CREATE TABLE professors
  (drofessorID int AUTO_INCREMENT,
   firstName varchar(255) NOT NULL,
   lastName varchar(255) NOT NULL,
   city varchar(255),
   department int,
   CONSTRAINT
    FK_professors_departments
    FOREIGN KEY(department)
   REFERENCES departments(departmentID),
   CONSTRAINT
    PK_professors
    PRIMARY KEY(professorID)
  )';
mysqli_query($link,$sql_d)
  or die('Tabela ne može da se kreira');
echo 'Tabela je uspešno kreirana <br/>';
mysqli_query($link,$sql_p)
  or die('Tabela ne može da se kreira');
echo 'Tabela je uspešno kreirana <br/>';
?>
```

Listing 69 - Upisivanje podataka u tabelu, funkcija *mysqli_query()*

```php
<?php
$link = mysqli_connect('localhost', 'nebojsa', 'sifra', 'sii')
  or die();
echo "Konekcija je uspostavljena <br/>";
$sql="INSERT INTO departments VALUES
      (3,'Electrical Engineering', 'Kumodraska 261/floor3')";
mysqli_query($link,$sql) or die();
echo "Podaci su uspešno unešeni <br/>";
?>
```

Kada je baza podataka kreirana, prilikom povezivanje sa serverom baze podataka može da se navede i četvrti argument *mysqli_connect()* funkcije, koji se odnosi na naziv baze na koju se vrši povezivanje. Nakon toga, *mysqli_query()* funkcija može da se koristi za izvršavanje *DDL* upita koji kreira tabelu u bazi. Primer sa kreiranjem tabela *professors* i *departments*, koje su spojene relacijom 1:više dat je na listingu 68.

Primer sa upisivanjem podataka u tabelu departments (*DML* naredba *INSERT*) pomoću *mysqli_query()* funkcije prikazan je na listingu 69.

U primeru 70 prikazano je čitanje podataka iz tabele deparments pomoću funkcije *mysqli_fetch_row()*.

Listing 70 - Prikaz rezultata izvršavanja upita, funkcija *mysqli_fetch_row()*

```php
<?php
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
$mysqli = mysqli_connect(
            "localhost", "my_user", "my_password", "world"
          );
$query = "SELECT Name, CountryCode
          FROM City
          ORDER BY ID DESC";
$result = mysqli_query($mysqli, $query);
/* fetch associative array */
while ($row = mysqli_fetch_row($result)) {
    printf("%s (%s)\n", $row[0], $row[1]);
}
?>
```

7.2.2 Objektno-orijentisana paradigma za rad sa bazom podataka

Dostupne PHP biblioteke obiluju klasama koje omogućavaju rad sa MySQL bazama podataka[20]. Kao neke od važnijih navode se *mysqli::query, mysqli_result::fetch_all* i *mysqli::close*.

Sintaksa za povezivanje sa serverom baze podataka:

```
$conn = new mysqli(host[, korisnik[, lozinka[, baza]]);
$conn->close();
```

Sintaksa za izvršavanje upita:

```
$conn->query ($upit);
```

Klasa *mysqli_result*[21] omogućava preuzimanje rezultata izvršavanja *SELECT* SQL upita. Sintaksa dostupnih metoda klase *mysqli_result*:

```
$res=$conn->mysqli($upit);
$res->fetch_all();
$res->fetch_assoc();
$res->fetch_array();
```

U listinzima prikazanim u nastavku navedeni su primeri izvršavanja *CRUD* (engl. *CREATE, RETRIEVE, UPDATE* i *DELETE*) operacija korišćenjem objektno -orientisane paradigme.

20 Kompletna lista objektno-orijentisanih naredbi može se naći na adresi http://www.php.net/manual/en/class. mysqli.php

21 Detaljna lista dostupnih metoda klase mysqli_result može se naći na adresi http://php.net/manual/en/class. mysqli-result.php

Listing 71 - Kreiranja i brisanje baze

```php
<?php
$conn=new mysqli('localhost', 'nebojsa', 'sifra');
if($conn->connect_error)
    die("konekcija nije moguća " . $conn->connect_error);
else
    echo "povezivanje je uspešno <br/>";
// definisanje i izvršavanje DROP i CREATE upita
$query='DROP DATABASE IF EXISTS myDB';
$query1='CREATE DATABASE myDB';
$conn->query($query);
if($conn->query($query1)==1)
    echo "baza je kreirana";
else
    echo "baza ne može da se kreira " . $conn->error;
$conn -> close();
?>
```

Listing 72 - Kreiranje tabele

```php
<?php
$conn = new mysqli('localhost', 'nebojsa', 'sifra', 'myDB');
if ($conn->connect_error)
    die("Konekcija nije uspela " . $conn->connect_error);
// ukoliko postoji, tabela test se briše
$sql="DROP TABLE IF EXISTS test";
$conn->query($sql);
// sql upit za krairanje tabele
$sql1 = "CREATE TABLE Test (
         id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
         firstname VARCHAR(30) NOT NULL,
         lastname VARCHAR(30) NOT NULL,
         email VARCHAR(50),
         reg_date TIMESTAMP
)";
if ($conn->query($sql1) === TRUE)
    echo "Tabela test je uspešno kreirana";
else
    echo "Greška u kreiranju tabele: " . $conn->error;
$conn->close();
?>
```

Listing 73 - Upis podataka u tabelu i prikaz trenutne *AUTO_INCREMENT* vrednosti

```php
<?php
$conn = new mysqli('localhost', 'nebojsa', 'sifra', 'myDB');
if ($conn->connect_error)
    die("Neuspela konekcija:" . $conn->connect_error);
$sql = "INSERT INTO test (firstname, lastname, email)
VALUES ('Jane', 'Doe', 'jane@singidunum.ac.rs')";
// ako tabela koristi autoincrement za primarni ključ,
// u ovom slučaju to je polje id,
// onda može da se vidi, odmah nakon izvršavanja upita
// vrednost primarnog ključa
if ($conn->query($sql) === TRUE) {
    $last_id = $conn->insert_id;
    echo "Novi zapis je uspešno upisan!<br/>" .
         "Poslednji ID je: {$last_id}";
} else
    echo "Error: " . $sql . "<br>" . $conn->error;
$conn->close();
?>
```

Primer sa brisanjem i kreiranjem baze podataka pomoću SQL DDL naredbi je dat na listingu 71, dok je kreiranje tabele pomoću SQL DDL naredbi prikazano na listingu 72.

Ako tabela za primarni ključ koristi atribut *AUTO_INCREMENT*, vrednost svojstva *insert_id* konekcionog objekta čuva vrednost primarnog ključa poslednje unetog zapisa u tabelu. Primer je naveden u listingu 73.

Listing 74 - Izvršavanje više upisa istovremeno

```php
<?php
$conn = new mysqli('localhost', 'nebojsa', 'sifra', 'myDB');
if ($conn->connect_error)
    die("Neuspela konekcija " . $conn->connect_error);
$sql = "INSERT INTO test (firstname, lastname, email)
        VALUES ('John', 'Doe', 'singidunum@example.com');";
$sql .= "INSERT INTO test (firstname, lastname, email)
        VALUES ('Marko', 'Markovic', 'marko@example.com');";
$sql .= "INSERT INTO test (firstname, lastname, email)
        VALUES ('Milena', 'Milenkovic', 'milena@example.com')";
// izvrsavamo multi query upita
if ($conn->multi_query($sql) === TRUE)
    echo "Više slogova je upisano!";
else
    echo "Greška: " . $sql . "<br>" . $conn->error;
$conn->close();
?>
```

Korišćenjem metode *multi_query()* više slogova može istovremeno da se upiše u tabelu SQL DML naredbom *INSERT*, kao što je dato u listingu 74.

U praksi se često koriste upiti sa pripremljenim naredbama (engl. *prepared statements*), koji se baziraju na tzv. vezanim parametrima (engl. *bound parameters*). Osim toga što su ovakvi tipovi upita brži i efikasniji, oni takođe obezbeđuju fleksibilnost prilikom unosa podataka u tabelu. Primer je dat u listingu 75.

Na listinzima 76 i 77 prikazano je izvršavanje SELECT upita pomoću *fetch_assoc()* metode.

Listing 75 - Upotreba pripremljenih upita i vezivanje parametara

```php
<?php
$conn = new mysqli('localhost', 'nebojsa', 'sifra', 'myDB');
if ($conn->connect_error)
    die("Greška u konekciji: " . $conn->connect_error);
// priprema upita i vezivanje parametara
$stmt = $conn->prepare(
          "INSERT INTO test (firstname, lastname, email)
           VALUES (?, ?, ?)"
        );
// "sss" pokazuje o kojim tipovima parametara se radi, konkretno
// u ovom slučaju upisuje se 3 puta string i zato se navodi "sss"
$stmt->bind_param("sss", $firstname, $lastname, $email);
// postavljanje vrednosti parametara i izvršavanje
$firstname = "John";
$lastname = "Doe";
$email = "singidunum@example.com";
$stmt->execute();
$firstname = "Milena";
$lastname = "Milenkovic";
$email = "milena@example.com";
$stmt->execute();
$firstname = "Marko";
$lastname = "Markovic";
$email = "marko@example.com";
$stmt->execute();
echo "Novi zapisi su uspešno upisani!";
$stmt->close();
$conn->close();
?>
```

Listing 76 - Izvršavanje *SELECT* upita i ispis rezultata

```php
<?php
$conn = new mysqli('localhost', 'nebojsa', 'sifra', 'myDB');
if ($conn->connect_error)
    die("Neuspela konekcija: " . $conn->connect_error);
$sql = "SELECT id, firstname, lastname, email FROM test";
$result = $conn->query($sql);
// prvo se proverava da li ima rezultata
if ($result->num_rows > 0) {
/* prikazuju se podaci za svaki red u formi asocijativnog niza,
   gde su indeksi nazivi kolona */
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"] .
    " " . $row["lastname"]. " Email: " . $row["email"] . "<br>";
}
// u suprotnom ispisuje se poruka da nema rezultata
else
  echo "nema rezultata";
$conn->close();
?>
```

Listing 77 - Izvršavanje *SELECT* upita i ubacivanje rezultata u HTML tabelu

```php
<?php
$conn = new mysqli('localhost', 'nebojsa', 'sifra', 'myDB');
if ($conn->connect_error)
  die("Connection failed: " . $conn->connect_error);
$sql = "SELECT id, firstname, lastname, email FROM Test";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    echo "<table><tr><th>ID</th><th>Name</th><th>Email</th></tr>";
    // prikaz rezultata za svaki red
    while($row = $result->fetch_assoc()) {
        echo "<tr>
                <td>{$row['id']}</td>
                <td>{$row['firstname']} {$row['lastname']}</td>
                <td>{$row['email']}</td>
              </tr>";
    }
    echo "</table>";
} else
    echo "nema rezultata";
$conn->close();
?>
```

Konačno na listinzima 78 i 79 prikazno je brisanje i ažuriranje slogova korišćenjem objektno-orijentisane paradigme, respektivno.

Listing 78 - Brisanje slogova

```php
<?php
$conn = new mysqli('localhost', 'nebojsa', 'sifra', 'myDB');
if ($conn->connect_error)
    die("Neuspela konekcija: " . $conn->connect_error);
// sql statement za  brisanje podataka
$sql = "DELETE FROM Test WHERE id=3";
if ($conn->query($sql) === TRUE)
    echo "Slog izbrisan uspesno";
else
    echo "Greska u brisanju sloga " . $conn->error;
$conn->close();
?>
```

Listing 79 - Ažuriranje slogova

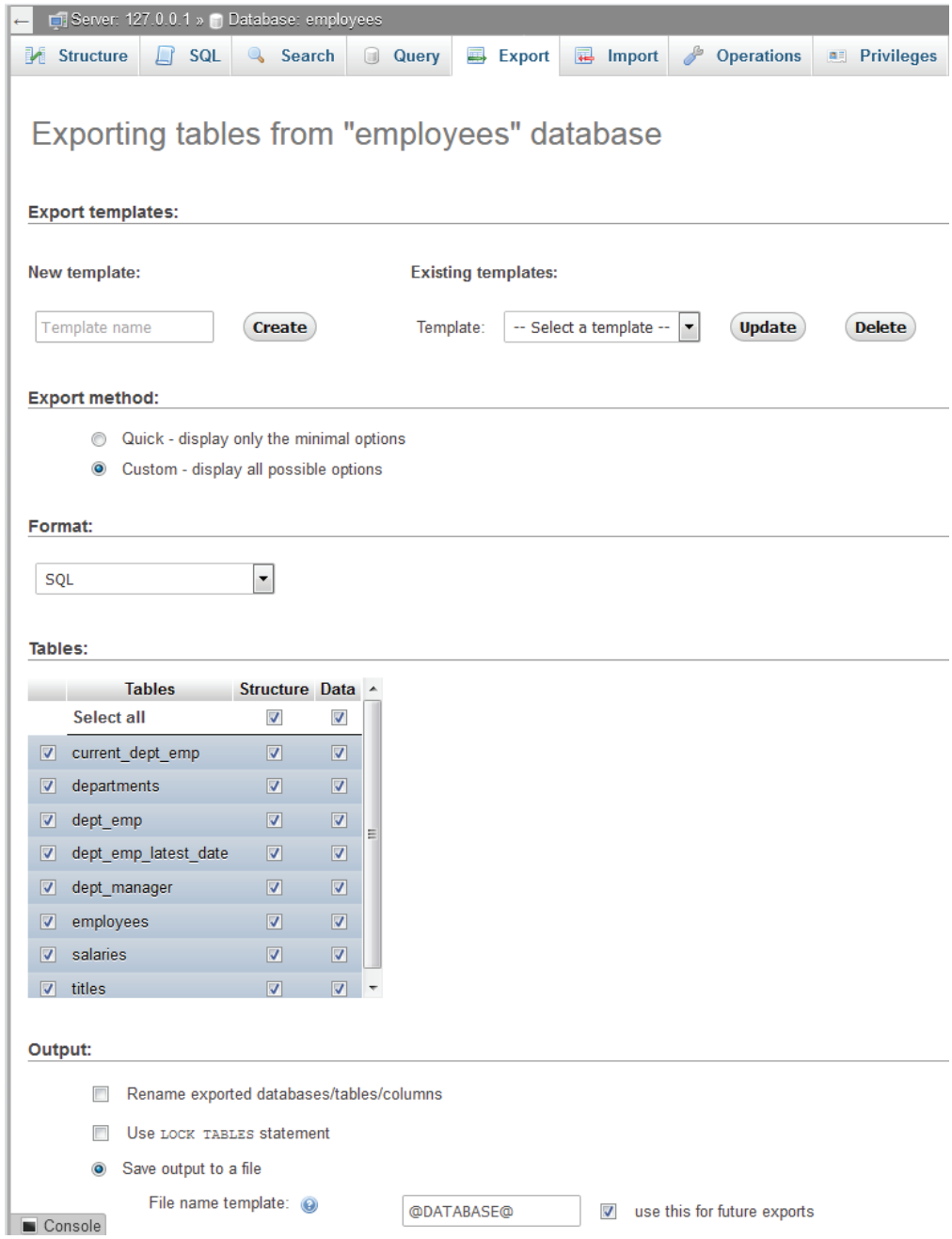
```php
<?php
$conn = new mysqli('localhost', 'nebojsa', 'sifra', 'myDB');
if ($conn->connect_error)
    die("Neuspela konekcija " . $conn->connect_error);
// sql update upit
$sql = "UPDATE test SET lastname='Doe' WHERE id=2";
if ($conn->query($sql) === TRUE)
    echo "Slog je uspešno ažuriran";
else
    echo "Greška prilikom ažuriranja" . $conn->error;
$conn->close();
?>
```

7.2.3 Izvoz baze

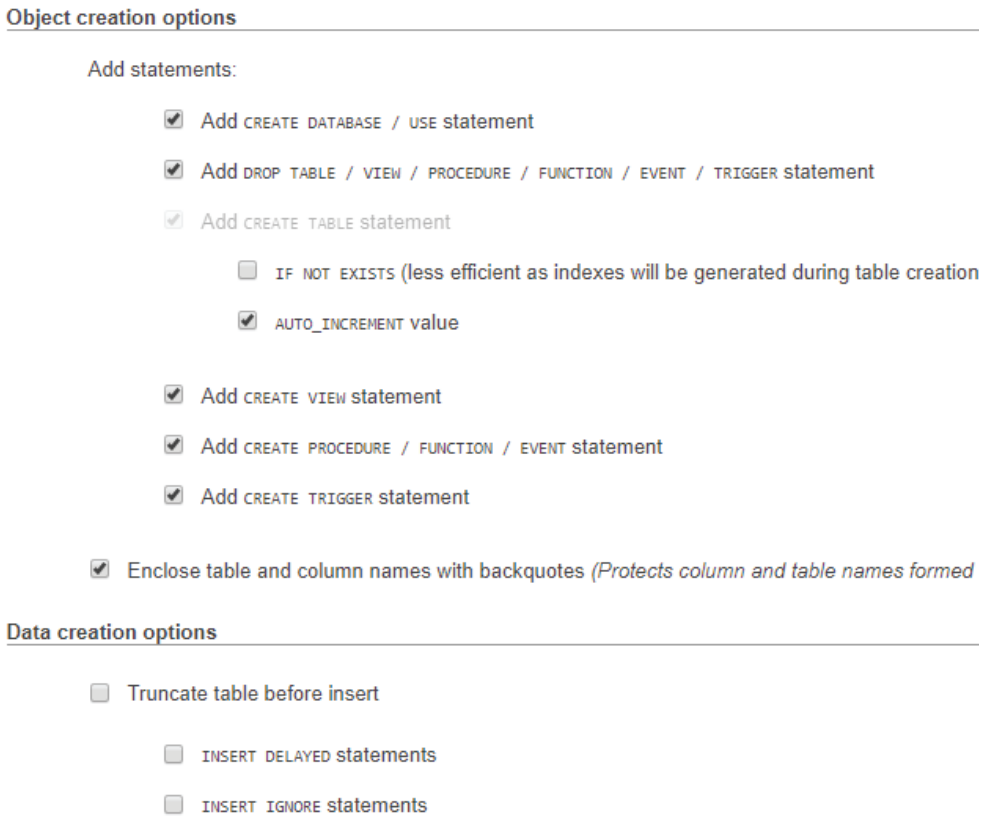
Dobra praksa je da se prvo kreira i testira MySQL baza u lokalnom (razvojnom) okruženju pre nego što se implementira u radno (produkciono) okruženje. Da bi se baza transferisala u produkciono okruženje, potrebno je da se baza izveze (engl. *export*). Jedan od najlakših načina za izvoz baze je korišćenjem *PHPMyAdmin* veb aplikacije. Izvoz baze vrši se putem četiri jednostavna koraka, slike 35 i 36.

- Ide se sa *localhost*-a i izabere se baza (bez ulaska u bazu), zatim se izabere opcija export, pa Custom.
- Izabere se baza koju je potrebno izvesti
- Štrikliraju se komande
- Add DROP DATABASE statement
- Add CREATE DATABASE / USE statement
- Klikne se na dugme *Go*

Slika 35 - *PHPMyAdmin*, izvoz baze - I



Slika 36 - *PHPMyAdmin,* izvoz baze - II



7.2.4 Kodiranje prikaza karaktera

Konačno, na kraju ovog dela valja spomenuti i kodiranje prikaza karaktera u bazi podataka. Prilikom kreiranja baze, tabela i polja definiše se njihov alfabet, odnosno *CHARACTER SET* i *COLLATION*. Da bi se u bazi na primer uspešno prikazala slova srpske azbuke, potrebno je podesiti UTF8 kodiranje.

```
mysqli_set_charset($veza, utf8");
```

TRANSFER PODATAKA

8. TRANSFER PODATAKA

U ovom poglavlju su prikazane osnove dva širko korišćena formata za transfer podataka - *XML* proširivi jezik za označavanje (engl. *eXtensible Markup Language*) i *JSON* (engl. *JavaScript Object Notation*). Oba formata se često koriste za potrebe transfera, odnosno razmene podataka, kako u slučaju klasičnih, tako i u slučaju veb baziranih aplikacija sa ili bez baza podataka.

Oba formata su standardizovana i zbog toga predstavljaju dobru opciju za razmenu podataka između aplikacija, tj. za uspostavljenje komunikacije na nivou mašina-mašina (engl. *machine to machine interaction*). U domenu veb servisa, *XML* i *JSON* se koriste za razmenu podataka između klijenta i servera u implementaciji *REST* (engl. *REpresentational State Transfer*), dok se u implementaciji *SOAP* (engl. *Simple Object Access Protocol*) veb servisa koristi isključivo *XML*. Na kraju glave dato je poređenje ova dva formata.

8.1 *XML* FORMAT

XML je alat koji je nezavistan od hardvera i softvera (engl. *software and hardvare independent*) za čuvanje i prenos podataka. Ovaj format spada u grupu jezika za označavanje, kakav je i *HTML*. Jedna od glavnih karakteristika *XML*-a jeste da je dizajniran kao samo-opisni (engl. *self-descriptive*) format, što znači da i ljudi analizom *XML* dokumenata mogu vrlo lako da opišu podatke koji su u njima sadržani.

Ovaj format je standardizovan od strane W3C (*World Wide Web Consortium*) organizacije, koja aktivno radi na njegovom održavanju i unapređenju.

Na listingu 80 je navedena relativno jednostavna XML sintaksa, koja prikazuje poruku koju je Marko ostavio Petru.

Listing 80 - Primer *XML* sintakse

```
<note>
  <to>Petar</to>
  <from>Marko</from>
  <heading>Podsetnik</heading>
  <body>Donesi mi knjigu iz Veb programiranja</body>
</note>
```

Prikazana *XML* sintaksa je samo-opisna i ona sadrži sledeće informacije:

- primaoc poruke;
- pošiljaoc poruke;
- zaglavlje poruke i
- poruka koja se šalje, tj. telo poruke

Međutim, potrebno je da se naglasi da *XML* dokumenti predstavljaju samo informacije koje su omotane u *XML* tagove. Međutim, *XML* ne podržava tagove koji su unapred definisani (predefinisani) i standardizovani. Na primer, tagovi poput <*to*> and <*from*> nisu definisani *XML* standardima. Naprotiv, tagovi se definišu isključivo od strane kratora *XML* dokumenata.

Za razliku od *XML*-a, *HTML* koristi predefinisane tagove, kao na primer <*div*>, <*table*>, <*h3*>, itd. i korisnik definiše samo sadržaj. U slučaju *XML*-a, korisnik definiše i tagove i strukturu samog dokumenta, listing 81.

Listing 81 - Primer *XML* sintakse, tagovi

```
<note>
  <date>28-3-2021</date>
  <hour>9:20am</hour>
  <to>Petar</to>
  <from>Marko</from>
  <heading>Podsetnik</heading>
  <body>Donesi mi knjigu iz Veb programiranja</body>
</note>
```

Kao što je navedeno u samom akronimu ove tehnologije, jedno od glavnih svojstva *XML*-a je proširivost. Većina *XML* aplkacije će funkcionisati nesmetano čak i kada se dodaju novi podaci, ili uklone postojeći. Tako na primer, ako se u primeru navedenom gore dodaju novi *XML* tagovi <*date*> i <*hour*>, starije verzije aplikacija koje ne koriste nove tagove će nastaviti normalno da funkcionišu.

Tehnologija *XML* značajno unapređuje kompatibilnost i proširivost aplikacija tako što pojednostavljuje:

- deljenje podataka;
- prenos podataka;
- promene platformi na kojima se aplikacije izvršavaju i
- unapređuje raspoloživost podataka.

Jedan od glavnih razloga za razvoj *XML* standardna je što mnogi tradicionalni računarski sistemi čuvaju podatke u nestandardizovanim formatima. Razmena podataka između takvih sistema je otežana i svaki put kada je potrebno omogućiti deljenje podataka između takvih sistema Veb programeri moraju da dograde kod, što predstavlja mukotrpan i često dugotrajan proces. Velike količine podataka moraju da budu konvertovani iz jednog u drugi format, dok se s druge strane često dolazi do gubitka nekompatibilnih podataka.

Tehnologija *XML* čuva podatke u formi običnog teksta (engl. *text plain format*). Tako je omogućeno je čuvanje, transfer i deljenje podataka na način koji ne zavisi od softvera, niti od hardvera. Format *XML* takođe omogućava proširenje i nadogradnju operativnih sistema, aplikacija i drugi alata bez gubitka podataka.

XML ne sadrži informacije o tome kako se podaci prikazuju. Implikacija ovoga jeste da isti *XML* podaci mogu da se koriste sa različitim tehnologijama za prezentovanje (prikaz) podataka. Drugim rečima, *XML* omogućava odvajanje samih podataka o načina njihovog prikaza.

U većini praktičnih implementacija, *XML* je komplement *HTML*-u, gde se *XML* koristi za čuvanje i transport podataka, dok je *HTML* zadužen za formatiranje i prikaz tih podataka.

XML podaci se obično čuvaju u formi posebnog fajla i zbog toga, kada se na primer promeni sadržaj *XML*-a (dodaju se novi tagovi, uklone se postojeći, itd.), *HTML* fajl koji se koristi za prikaz sadržaja *XML* dokumenta ne mora da se menja. Osim ovoga, pomoću samo nekoliko linija JavaScript koda, sadržaj *XML* fajla može da čita i prikaže pomoću *HTML* tehnologije.

Listing 82 - *XML* struktura, *root-child*

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

XML dokumenti se reprezentuju pomoću tzv. *XML* stabla (engl. *XML tree*). Stablo počinje korenim elementom (engl. *root element*) i grana se do elemenata dece (engl. *child elements*). Svaki *XML* element može da ima više elemenata dece, listing 82.

Listing 83 - Jednostavan *XML* dokument

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="Cloud Computing">
    <title lang="en">Introduction to Cloud Computing</title>
    <author>Nebojsa Bačanin</author>
    <author> Ivana Štrumberger</author>
    <year>2018</year>
    <price>50.00</price>
  </book>
  <book category="Cloud Computing">
    <title lang="en">Advanced Cloud Computing Concepts</title>
    <author>James Potter</author>
    <year>2020</year>
    <price>69.99</price>
  </book>
  <book category="Macine Learning">
    <title lang="en">Advances in Machine Learning</title>
    <author>Jeoffrey Hinton</author>
    <year>2013</year>
    <price>59.95</price>
  </book>
</bookstore>
```

Termini roditelj (engl. *parent*), dete i brat ili sestra (engl. *siblings*) koriste se za opis veza između elemenata. Roditelji imaju decu. Deca imaju roditelje. Brat ili sestra su deca na istom hijerahijskom nivou.

Da bi se opisala struktura *XML* dokumenata, korišćen je jednostavan demonstrativni primer prikazan na listingu 83. Kao što je već i navedeno, *XML* koristi samo-opisujuću sintaksu. Na početku *XML* dokumenta, u zaglavlju, definiše se *XML* verzija i shema kodiranja, linija 1. Odmah ispod zaglavlja *XML* dokumenta, navodi se koreni element <bookstore>, linija 2. Naredna linija 3 počinje elementom <*book*>. Kao što se vidi iz priloženog koda, opciono *XM*L element može da ima i atribut, što je ovom slučaju *category* koji ima vredost *Cloud Computing*. *Element* <*book*> ima 4 elemenata dece: <*title*>, <*author*>, <*year*>, <*price*>, linije 4-8. Konačno, za svaki element mora da se definiše i zatvarajući tag, što je u ovom slučaju </*book*>, linija 9.

Listing 84 - Primer *XML* dokumenta koji opisuje vesti[22]

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <nitf>
3    <head>
4      <title>Colombia Earthquake</title>
5    </head>
6    <body>
7      <headline>
8        <hl1>143 Dead in Colombia Earthquake</hl1>
9      </headline>
10     <byline>
11       <bytag>By Jared Kotler, Associated Press Writer</bytag>
12     </byline>
13     <dateline>
14       <location>Bogota, Colombia</location>
15       <date>Monday January 25 1999 7:28 ET</date>
16     </dateline>
17   </body>
18 </nitf>
```

22 https://www.w3schools.com/XML/xml_usedfor.asp

Pretragom veb-a može se videti da postoje različiti *XML* formati, koji se koriste za opis i reprezentaciju podataka u raznim industrijama, kao i za opis različitih tipova transakcija. Kao neki od primera navode se sledeći:

- akcije, obveznice, finansijske transakcije;
- medicinski podaci;
- matematički podaci, naučna izračunavanja;
- vesti, podaci o vremenu

Na listinzima 84 i 85 dati su primeri *XML* dokumenata koji prikazuju informacije o vestima i vremenskoj prognozi, respektivno.

Listing 85 - Primer *XML* dokumenta koji opisuje vremensku prognozu[23]

```
<?xml version="1.0" encoding="UTF-8"?>
<current_observation>
  <credit>NOAA's National Weather Service</credit>
  <credit_URL>http://weather.gov/</credit_URL>
  <image>
    <url>http://weather.gov/images/xml_logo.gif</url>
    <title>NOAA's National Weather Service</title>
    <link>http://weather.gov</link>
  </image>
  <location>New York/John F. Kennedy Intl Airport, NY</location>
  <station_id>KJFK</station_id>
  <latitude>40.66</latitude>
  <longitude>-73.78</longitude>
  <observation_time_rfc822>Mon, 11 Feb 2008 06:51:00 -0500 EST
  </observation_time_rfc822>
  <weather>A Few Clouds</weather>
  <temp_f>11</temp_f>
  <temp_c>-12</temp_c>
  <relative_humidity>36</relative_humidity>
  <wind_dir>West</wind_dir>
  <wind_degrees>280</wind_degrees>
  <wind_mph>18.4</wind_mph>
  <wind_gust_mph>29</wind_gust_mph>
  <pressure_mb>1023.6</pressure_mb>
  <pressure_in>30.23</pressure_in>
  <dewpoint_f>-11</dewpoint_f>
  <dewpoint_c>-24</dewpoint_c>
  <windchill_f>-7</windchill_f>
  <windchill_c>-22</windchill_c>
  <visibility_mi>10.00</visibility_mi>
  <icon_url_base>http://weather.gov/weather/images/fcicons/</icon_url_base>
  <icon_url_name>nfew.jpg</icon_url_name>
  <disclaimer_url>http://weather.gov/disclaimer.html</disclaimer_url>
  <copyright_url>http://weather.gov/disclaimer.html</copyright_url>
</current_observation>
```

23 https://www.w3schools.com/XML/xml_usedfor.asp

8.2 *JSON* FORMAT

JSON je lagni tekst-orijentisani (engl. *text-based*) standard za čitanje i razmenu podataka. Ovaj format se koristi za skladištenje podataka i informacija i omogućava lak pristup uskladištenim podacima i informacijama. Jedna od bitnijih karakteristika *JSON*-a jeste da ovaj format mogu da čitaju i ljudi (ne samo mašine). Konvencije korišćene od strane *JSON*-a koriste se u mnogim programskim jezicima poput C, C++, Java, Python, PHP, itd. Kao neke od osnovnih informacija o *JSON* formatu izdvajaju se sledeće:

- format je uspostavio Daglas Krokford (engl. *Douglas Crockford*);
- dizajniran je za čitljivu razmenu podataka;
- lako se čita i piše;
- nezavisan od programskih jezika;
- nasleđen je od JavaScript-a;
- ekstenzija koju koristi je .json;
- *JSON* Internet Media tip je aplication/json i
- Uniform Type identifikator je public.json.

Ubrzo nakon što je Daglas Krakford definisao osnovne specifikacije *JSON* formata ranih 2000-tih godina, zvanična veb prezentacija ovog, tada novog formata, je puštena u rad. Već decembra 2005. godine, neki od Yahoo! veb servisa su podržavali *JSON* format. Godine 2013., *JSON* je postao međunarodni ECMA standard (engl. *European Computer Manufacturers Association*), dok je najnovija verzija *JSON*-a objavljena 2017. godine.

Kao primeri upotrebe *JSON* formata mogu da se izdvoje sledeći:

- pisanje JavaScript aplikacija, kao i za Veb ekstenzije i sajtove;
- serijalizacija i prenos strukturiranih podataka preko Interneta;
- prenos podataka između servera i Veb aplikacija;
- prikaz javnih podataka od strane Veb servisa i API-ja i
- kao komplementarna tehnologija viših programskih jezika.

Sintaksa *JSON*-a je izvedena iz JavaScript notacije objekata i kao takva predstavlja podskup JavaScript sintakse. Osnovna *JSON* sintaksna pravila mogu da se sumiraju na sledeći način:

- podaci se predstavljaju u formi naziv/vrednost;
- nizovi objekata se nalaze unutar uglastih zagrada *[ ]*;
- vitičaste zagrade *{ }* u sebi sadrže objekte;
- u objektima se promenljive (podaci) razdvajaju zarezom;
- parovi naziv/vrednost se razdvajaju dvotačkom *(:)* i pišu se pod znacima navoda "..." i
- podržana su dva načina prikaza, kao kolekcija ili kao redni niz u zavisnosti od drugih tehnologija koje se uz *JSON* koriste.

Podaci se reprezentuju u formi naziv/vrednost (engl. *name/value*), tj. ključ/vrednost (engl. *key/value*). U primeru na listingu 86, vitičaste zagrade se koriste za reprezentaciju i čuvanje objekata, linije 1 i 11. name predstavlja naziv, dok je *Joeffrey* vrednost, linija 2. Podaci se razdvajaju zarezom, dok uglaste zagrade označavaju niz, linije 6 i 10.

Listing 86 - *JSON* sintaksa

```
1  var person = {
2    "name":"Joeffrey",
3    "surname":"Hinton",
4    "Occupation":
5    "Machine and Deep Learning,
6    "Profession": [
7      "University professor",
8      "Deep learning consultant",
9      "Scientific researcher"
10   ]
11 }
```

Primer *JSON* formata koji čuva podatke o raspoloživim knjigama dat je na listingu 87.

U sledećoj listi navedene su osnovne karakteristike tipova podataka koje *JSON* podržava:

Listing 87 - *JSON* format, podaci o raspoloživim knjigama

```
{
  "book": [
    {
      "id":"1",
      "title": "Parallel Models of Associative Memory",
      "edition": "first",
      "author": "Geoffrey E. Hinton",
      "publisher": "MIT Press"
    },
    {
      "id":"2",
      "title": "Deep Learning",
      "edition": "second",
      "author": "Ian Goodfellow",
      "publisher": "MIT Press"
    }
  ]
}
```

- Brojevi
- celobrojni pozitivni i negativni;
- decimalni;
- eksponencijalni;
- Stringovi
- niz od nula ili više *Unicode* karaktera unutar znakova navoda (" ");
- karakter je string dužine 1;
- specijalni karakteri su " \ / b f n r t u;
- Boolean

- čuva samo true ili false vrednosti;
- Nizovi
- redna kolekcija vrednosti;
- počinju i završavaju se sa uglastim zagradama;
- vrednosti su odvojene zarezom;
- indeksiranje može početi od 0 ili 1;
- Objekti
- neodređen set promenljivih i vrednosti;
- čuvaju su u vitičastim zagradama;
- ključevi (podaci) su odvojeni zarezom;
- ključevi moraju biti jedinstveni;
- objekti se koriste kada su ključevi neodređeni stringovi.
- Prazni prostori
- mogu se uneti da bi kod bio čitljiviji između dva tokena.
- Null
- označava praznu vrednost.

Primeri upotrebe *JSON* formata unutar *HTML*-a prikazani su na listinzima 88 i 89.

Listing 88 - Primer korišćenja *JSON*-a u *HTML* dokumentu

```
<html>
<head>
<title>Creating Object JSON with JavaScript</title>
<script language = "javascript" >
  var JSONObj = { "name" : "SINGIDUNUM UNIVERSITY", "year"  : 2005 };
  document.write("<h1>JSON with JavaScript example</h1>");
  document.write("<br>");
  document.write("<h3>University name = "+JSONObj.name+"</h3>");
  document.write("<h3>Year of establishment = "+JSONObj.year+"</h3>");
</script>
</head>
<body>
</body>
</html>
```

Kada se govori o *JSON*-u, potrebno je spomenuti i *JSON* shemu. Shema (engl. *schema*) je specifikacija *JSON* baziranog formata za definisanje strukture *JSON* podataka. Osnovne karakteristike *JSON* sheme sumirane su na sledeći način:

- definiše postojeći format podataka;
- omogućava kreiranje čitljivije dokumentacije;
- omogućva struktuiranu validaciju za automatsko testiranje, kao i validaciju podataka koji su unešeni od strane korisnika;

Listing 89 - Primer kreiranja niza *JSON* objekata u *HTML*-u

```
<html>
<head>
<title>Creation of array object in javascript using JSON</title>
<script language = "javascript" >
  document.writeln("<h2>JSON array object</h2>");
  var books = {
    "JAVA" : [
      { "Name"  : "Java for Beginners", "price" : 700 },
      { "Name"  : "Advanced Java Programming", "price" : 400 }
    ],
    "C"  : [
      { "Name"  : "C From Novice to Pro", "price" : 1000 },
      { "Name"  : "C in Depth", "price" : 1300 }
    ]
  };
  var i = 0;
  document.writeln("<table border = '2'><tr>");
  for(i = 0; i<books.JAVA.length; i++) {
    document.writeln("<td>");
    document.writeln("<table border = '1' width = 100 >");
    document.writeln("<tr><td><b>Name</b></td><td width = 50>" +
      books.JAVA[i].Name+"</td></tr>");
    document.writeln("<tr><td><b>Price</b></td><td width = 50>" +
      books.JAVA[i].price +"</td></tr>");
    document.writeln("</table>");
    document.writeln("</td>");
  }
  for(i = 0; i<books.C.length; i++) {
```

```
28   for(i = 0; i<books.C.length; i++) {
29     document.writeln("<td>");
30     document.writeln("<table border = '1' width = 100 >");
31     document.writeln("<tr><td><b>Name</b></td><td width=50>" +
32       books.C[i].Name+"</td></tr>");
33     document.writeln("<tr><td><b>Price</b></td><td width=50>" +
34       books.C[i].price+"</td></tr>");
35     document.writeln("</table>");
36     document.writeln("</td>");
37   }
38   document.writeln("</tr></table>");
39 </script>
40 </head>
41 <body>
42 </body>
43 </html>
```

- postoje različiti validatori za svaki programski jezik, a najkompletniji je JSV (engl. *JSON Validator*)
- *C* → VJElement (LGPLv3)
- *Java* → json-schema-validator (LGPLv3)
- *.NET* → Json.NET (MIT)
- *ActionScript 3* → Frigga (MIT)
- *Haskell* → aeson-schema (MIT)
- *Python* → Jsonschema
- *Ruby* → autoparse (ASL 2.0)
- *PHP* → php-json-schema (MIT)
- JavaScript → JSV

Listing 90 - JSON shema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Product",
  "description": "A product from Acme's catalog",
  "type": "object",
  "properties": {
    "id": {
      "description": "The unique identifier for a product",
      "type": "integer"
    },
    "name": {
    "description": "Name of the product",
    "type": "string"
    },
    "price": {
      "type": "number",
      "minimum": 0,
      "exclusiveMinimum": true
    }
  },
  "required": ["id", "name", "price"]
}
```

Primer *JSON* sheme je dat na listingu 90. Od *JSON* shema ključnih reči (engl. *keywords*) izdvajaju se sledeće:

- *$schema* → Znači da je shema napisana po v4 specifikaciji;
- *title* → Koristi se za nazivanje sheme;
- *description* → Opis sheme;
- *type* → Uvek JSON Objekt;
- *properties* → Opisuje vrednosti, njihove tipove i ograničenja;
- *required* → Lista potrebnih propertija;
- *minimum* → Ograničenje minimalne vrednosti;
- *exclusiveMinimum* → Boolean vrednost koja je true ako je instanca veća od minimuma;

- *maximum* → Ograničenje maksimalne vrednosti;
- *exclusiveMaximum* → Boolean vrednost koja je true ako je instanca manja od maksimuma;
- *maxLength* → Maksimalna dužina string instance;
- *minLength* → Minimalna dužina string instance i
- *pattern* → String instanca je validna ako se regularni iskaz poklapa sa instancom.

8.3 POREĐENJE *XML* I *JSON* FORMATA

Da bi se bolje uočila razlika između *XML* i *JSON* formata, sintaksa za definisanje objekta student za oba formata prikazana je na listingu 91, dok su ključne razlike između *XML* i *JSON* formata sumirane su u tabeli 13.

Listing 91 - Poređenje sintakse: *JSON* (levo) i *XML* (desno)

```
{
  "student": [{
      "id":"01",
      "name": "Marko",
      "lastname": "Markovic"
    },
    {
      "id":"02",
      "name": "Petar",
      "lastname": "Petrovic"
  }]
}
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <student>
    <id>01</id>
    <name>Marko</name>
    <lastname>Markovic</lastname>
  </student>
  <student>
    <id>02</id>
    <name>Petar</name>
    <lastname>Petrovic</lastname>
  </student>
</root>
```

Tabela 13 - Glavne razlike između *XML* i *JSON* formata

JSON	***XML***
JSON objekat ima tip podataka	*XML* podaci su netipizirani
JSON tipovi podataka: broj, string, niz, logički	Svi *XML* podaci su stringovi
Podaci iz *JSON*-a su raspoloživi korisnicima i aplikacijama bez obrade	*XML* podaci moraju da se parsiraju pre korišćenja
JSON je podržan od strane većine Internet pregledača	Kompatibilnosti pregledača u kontekstu *XML* parsiranja je problematična
JSON nema mogućnost prikaza podataka	*XML* omogućava prikaz podataka zato što spada u grupu jezika za označavanje
JSON podržava samo tekstualne i numeričke tipove podataka	Osim tekstualnih i numeričkih, *XML* podržava tipove podataka kao što su slike i grafikoni. Takođe omogućava transfer strukture i formata podataka zajedno sa pohranjenim podacima
Čitanje vrednosti podataka je lako	Čitanje vrednosti podataka je relativno teško
Podržan od strane AJAX seta alata	Nije podržan od strane AJAX seta alata
Serijalizacija/deserijalizacija podataka od strane JavaScript-a je automatizovana	Programeri moraju da napišu JavaScript kod koji će serijalizovati/deserijalizovati podatake iz *XML* formata
Podržava samo UTF-8 kodiranje	Podržava razne tipove kodiranja
Ne podržava komentare	Podržava komentare
JSON fajlovi se lako čitaju	*XML* dokumenti se relativno teško čitaju i interpretiraju
Ne podržava imenske prostore	Podržava imenske prostore
Manje je siguran nego *XML*	Sigurniji je od *JSON* formata

8.4 PREDNOSTI I NEDOSTACI *XML* I *JSON* FORMATA

Prilikom razvoja veb aplikacije, sagledavanjem prednosti i nedostataka *XML* i *JSON* formata moguće je izabrati format koji je pogodniji za primenu u određenoj situaciji.

Kao neke od prednosti JSON-a navode se sledeće:

- podržan od strane svih pregledača;
- lak za čitanje i pisanje;
- jednostavna i logična sintaksa;
- podrška parsiranja u osnovnim JavaScript bibliotekama pomoću *eval()* fukcije;
- lako se kreira i menja;
- podržan od strane većina glavnih JavaScript okvira;
- podržan od strane većine glavnih tehnologija za back-end;
- omogućava transfer i serijalizaciju struktuiranih podataka pomoću računarske mreže;
- može da se koristi u savremenim programskim jezicima i
- bilo koji JavaScript objekat može lako da se konvertuje u *JSON* tekstualni objekat i da se pošalje Veb serveru na obradu.

Glavni nedostaci *JSON* formata su:

- ne postoji podrška za prostore imenovanja i zbog toga *JSON* format ne pruža dovoljno mogućnosti za proširivanje funkcionalnosti;
- ograničena podrška razvojnih alata i
- lošija sigurnost.

Značajnije prednosti *XML* formata obuhvataju:

- omogućava da različiti tipovi dokumenata mogu relativno lako da se prenose između različitih sistema i aplikacija;
- omogućava laku razmenu podataka između različitih, heterogenih računarskih platformi;
- *XML* odvaja podatke od *HTML* koda i
- olakšava i pojednostavljuje migraciju aplikacije sa jedne na drugu platformu.

Glavni nedostaci *XML*-a odnose se na sledeće:

- korišćenje *XML*-a zahteva aplikaciju koja će format da parsira;
- sintaksa je slična kao kod alternativnih tehnologija za prenos podataka u formi teksta, što može da izazove konfuziju;
- ne postoji prava podrška za tipove podataka;
- *XML* sintaksa je redudantna i
- ne omogućava korisnicima da kreiraju personalizovane tagove.

REGULARNI IZRAZI

9. REGULARNI IZRAZI

Regularni izrazi (engl. *regular expressions*), skraćeno *regex* ili *regexp* predstavljaju niz karaktera koji formiraju šablon za pretragu stringa ili zamenu jednog ili više delova stringa. Rad sa regularnim izrazima je podržan u mnogim programskim jezicima, i predstavljaju neizostavan alat u veb aplikacijama i programima za obradu teksta. Najčešće oblasti primene su:

- Algoritmi za pretragu i/ili izmenu podataka
- Provera validnosti ulaznih podataka

Šablon regularnog izraza se može predstaviti kao string u sledećem formatu:

```
$sablon = '#(singidunum|univerzitet)#i';
```

9.1 RAZDELNICI

Razdelnik označava početak i kraj regularnog izraza, u prethodnom primeru je korišćen znak taraba (#, engl. *hashtag*). PHP programski jezik u radu sa regularnim izrazima podržava PCRE sintaksu, koja dozvoljava upotrebu sledećih karaktera kao razdelnika:

- # hashtag
- % oznaka za procenat
- + znak plus
- ~ tilda

Osim navedenih, kao razdelnici su podržane i zagrade - (), {}, [] i <>, ali se u praksi ne koriste zbog njihove česte upotrebe u okviru samog šablona.

9.2 ŠABLON

Između razdelnika se nalazi šablon, koji predstavlja šemu, odnosno pravilo ili skup pravila prema kojima se pretražuje string. Šablon čine atomi, koji mogu biti slova, klase karaktera ili metakarakteri.

9.2.1 Metakarakteri

Većina procesora regularnih izraza prepoznaje barem četrnaest metakaraktera, u koje ubrajamo zagrade - {}, [], i (), cirkumfleks znak (^), znak dolar ($), tačka (.), vertikalna crta (|), asterisk (*), znak plus (+), znak pitanja (?), i obrnuta kosa crta (\).

Svaki metakarakter ima tačno određenu funkciju, na osnovu koje se mogu podeliti u četiri grupe:

- Escaping
- Grupisanje
- Kvantifikacija
- Filtriranje karaktera

9.2.2 Pretraga bilo kog karaktera

Prilikom procesiranja stringova pomoću regularnih izraza, često se javlja potreba markiranja dela stringa čiji konkretan sadržaj zapravo ne utiče na krajnji rezultat operacije. U ovakvim situacijama se koristi, možda i najpoznatiji metakarakter regularnih izraza, tačka (.), koja traži bilo koji karakter, osim preloma linije. Ukoliko je u pretragu potrebno uključiti i prelom linije, to se može učiniti u šablonu navođenjem odgovarajućeg modifikatora iza drugog (zatvarajućeg) razdelnika.

Prilikom upotrebe ovog metakaraktera je potrebna doza opreza, zato što u pojedinim situacijama može da izazove nepredviđeno ponašanje regularnog izraza, odnosno da proizvede neželjene rezultate.

9.2.3 Escaping

Obrnuta kosa crta (\, engl. *backslash*) u šablonu regularnog izraza predstavlja tzv. *escape* karakter, koji označava da karakter koji se nalazi iza njega treba da se tretira u bukvalnom smislu, a ne kao metakarakter ili razdelnik. Važi i suprotno, obrnuta kosa crta se koristi kada je potrebno da se običan slovni karakter transformiše u klasu karaktera.

Ukoliko se kao razdelnik upotrebi kosa crta "/", a šablon proverava postojanje http protokola u stringu (http://), ispred kosih crta u šablonu koje predstavljaju deo oznake http protokola potrebno je staviti obrnutu kosu crtu:

```
$sablon = '/http:\/\//';
```

Međutim, ukoliko se kao razdelnik upotrebi *hashtag*, u šablonu se kose crte mogu ostaviti bez *escape* karaktera.

```
$sablon = '#http://#';
```

9.2.4 Klase karaktera

Klase, odnosno skupovi karaktera, predstavljaju alat koji omogućava jednostavno pronalaženje višestrukih karaktera koji pripadaju željenom skupu. Kao primer mogu da se navedu sledeća dva regularna izraza:

```
$sablon1 = "#w#";
$sablon2 = "#\w#";
```

U prvom slučaju, karakter "w" u šablonu označava slovo w. Međutim, ukoliko se ispred doda obrnuta kosa crta, "\w" predstavlja klasu karaktera, i odnosi se na sva slova i brojeve (alfanumerički znakovi), uključujući i donju crtu "_". Na sličan način može da se izvrši pretraga cifara, pomoću klase karaktera "\d".

U engleskom jeziku, shodno tome da li se koristi britanski ili američki engleski, ista reč može da se napiše na dva načina:

- demoralize
- demoralise

Regularni izrazi olakšavaju pretragu ovakvih reči, i u ovom slučaju šablon može da se formira na sledeći način:

```
$sablon = '#demorali[sz]e#';
```

Prikazani šablon funkcioniše tako da procesor pokušava da pronađe podstring *demorali,* nakon kojeg sledi jedan karakter iz skupa označenog uglastom zagradom, to su slova *s* ili *z*, i na kraju slovo *e*.

Klase karaktera takođe dozvoljavaju definisanje opsega u kojem se vrši pretraga, kao i kombinovanje različitih opsega:

- *[a-z]* - traži malo slovo abecede u opsegu od slova a do slova *z*.
- *[0-9]* - traži broj u opsegu od 0 do 9. Jednostavnije se piše kao *[\d]*.
- *[A-Za-z0-9_]* - traži veliko ili malo slovo u opsegu od a do z, broj u opsegu od 0 do 9 ili donju crtu. Jednostavnije se piše kao *[\w]*.

Klasa karaktera koja se odnosi na horizontalne i vertikalne razmake (engl. *whitespace character*) se označava sa "\s" i obuhvata:

- razmak (SPACE)
- \t, ASCII horizontalni tab (TAB)
- \v, ASCII vertikalni tab (VT)
- \r, ASCII Carriage Return (CR)
- \n, ASCII Linefeed (LF)
- \f, ASCII Formfeed (FF)

Moguće je vršiti pretragu karaktera koji se u regularnim izrazima tretiraju kao metakarakteri pod uslovom da se ispred stavi obrnuta kosa crta, npr. [\^] je klasa karaktera koja traži cirkumfleks znak.

Ukoliko je potrebno da se iz pretrage isključi određeni skup karaktera, to se može učiniti upotrebom tzv. negacije koja se obeležava cirkumfleks znakom.

Šablon

```
$sablon = '#[^0-9]#';
```

ili

```
$sablon = '#[^\d]#';
```

će u okviru stringa tražiti sve karaktere osim broja. U regularnim izrazima su podržane negativne klase karaktera, koje se označavaju velikim slovima:

- \W - traži sve, osim alfanumeričkih znakova i donje crte.
- \D - traži sve osim cifara.
- \S - traži sve osim razmaka.

9.2.5 Kvantifikacija

Kvantifikatori predstavljaju način na koji se procesoru regularnih izraza saopštava koliko puta treba da se u uzastopnom nizu pronađe karakter koji odgovara zadatoj klasi. Postoje tri glavna metakaraktera koji označavaju kvantifikaciju:

- Znak pitanja (?) daje signal procesoru regularnog izraza da je prethodni atom opcija, odnosno da ga treba pronaći 0 ili 1 puta:

  ```
  $sablon = '#behaviou?r#';
  ```

  Prikazani šablon će pronaći obe varijante tražene reči (*behaviour* i *behavior*).

- Asterisk, odnosno zvezdica (*) signalizira procesoru da se prethodni atom traži 0 ili više puta.
- Znak plus (+) funkcioniše na sličan način kao asterisk, tako što procesoru saopštava da pronađe prethodni atom 1 ili više puta.

9.2.5.1 Pohlepni izrazi

Važna osobina kvantifikatora je tzv. pohlepa, odnosno znak procesoru regularnog izraza da li nakon pronađenog atoma treba da nastavi da ga traži. Regularni izrazi su po prirodi uvek pohlepni, procesor pokušava da pronađe što više instanci zadatog atoma u neprekidnom nizu, što u određenim slučajevima može proizvesti neočekivane, ili neželjene rezultate.

Ovaj problem se rešava jednostavnim dodavanjem znaka pitanja (?), čime se procesoru daje signal da traženu sekvencu pronađe najmanji broj puta:

- . * - pronađi bilo koji karakter, pohlepno
- . *? - pronađi bilo koji karakter, nepohlepno
- *? - nepohlepna verzija * kvantifikatora
- +? - nepohlepna verzija + kvantifikatora

Razlika između pohlepnih i nepohlepnih izraza se najlakše može uočiti na primeru *HTML* linka nestovanog u *DIV* blok, listing 92.

Listing 92 - Parsiranje *HTML* koda regularnim izrazima

```
<div>
    <a href="https://www.singidunum.ac.rs">
        Univerzitet Singidunum
    </a>
</div>
```

Pohlepni izraz izražen kroz šablon '#<. *>#' će kao rezultat vratiti kompletan kod, zato što će procesor posle znaka manje (<) pokušati da pronađe bilo koji karakter (. *), sve do poslednjeg znaka veće (>), a to je karakter koji se nalazi na kraju stringa. Transformacijom pohlepnog šablona '#<. *>#' u nepohlepni '#<. *?>#', kao rezultat se vraćaju svi *HTML* tagovi koji se nalaze u kodu, slika 37.

Slika 37 - Rezultat nepohlepnog regularnog izraza

```
Array
(
  [0] => Array
    (
      [0] => <p>
      [1] => <a href="https://www.singidunum.ac.rs">
      [2] => </a>
      [3] => </p>
    )
)
```

9.2.5.2 Proizvoljna kvantifikacija

Ukoliko se javi potreba za pretragom stringa, pri čemu se očekuje pojavljivanje atoma tačno određeni broj puta u nizu, ili u određenom opsegu, koristi se kvantifikator označen vitičastim zagradama.

- [\d]{0, 1} - isto što i \d?
- [\d]{3} - pronađi 3 uzastopne cifre
- [\d]{3, 5} - pronađi najmanje 3, a najviše 5 uzastopnih cifara
- [\d]{0, } - isto što i \d*
- [\d]{1, } - isto što i \d+

9.2.6 Grupisanje

Provera da li se u stringu ponavljaju dva ili više atoma se može izvršiti grupisanjem atoma pomoću zagrada, metakarakteri (). Šablon

```
$sablon = '#sud(ovi)?#';
```

će kao pozitivan rezultat tretirati reč sud (jednina), ali i reč sudovi (množina). To je moguće zato što šablon traži podstring sud (prvi atom), a segment (ovi)? (drugi atom) daje signal procesoru regularnih izraza da prepozna kao rezultat podstring "ovi" koji se pojavljuje 0 ili 1 puta.

```
$sablon = '#sud(a|u|om|ovi)?#';
```

Izbor između više ponuđenih šablona se može realizovati grupisanjem u zagradu, dok se u zagradi različiti šabloni međusobno razdovjeni vertikalnom crtom "|". Prethodni primer će kao rezultat vratiti sledeće varijacije, ukoliko se nalaze u stringu koji se pretražuje:

- sud
- suda
- sudom
- sudovi

9.2.7 Ankeri

Ankeri definišu očekivanu poziciju atoma u stringu koji se pretražuje. Dva ankera koje podržava PHP programski jezik u radu sa regularnim izrazima su:

- Cirkumfleks znak (^) označava početak stringa ili početak linije, shodno tome da li se koristi odgovarajući modifikator.
- Znak dolar ($) označava kraj stringa, odnosno kraj linije uz odgovarajući modifikator.

```
$sablon1 = '#^(Univerzitet|Singidunum)#';
$sablon2 = '#(^Univerzitet|Singidunum$)#';
```

U ovom primeru, šablon dodeljen promenljivoj *$sablon1* proverava da li na početku stringa postoje reči *Univerzitet* ili *Singidunum*. Nasuprot tome, šablon dodeljen promenljivoj *$sablon2* će vratiti pozitivan rezultat ukoliko se na početku testiranog stringa nalazi reč *Univerzitet*, ili reč S*ingidunum* na njegovom kraju.

9.2.8 Modifikatori

Modifikatori se postavljaju na kraj regularnog izraza, iza drugog razdelnika. i koriste se za korekciju ponašanja procesora regularnog izraza. Ukoliko scenario predviđa proveru podatka poslatog putem veb obrasca, npr. imena univerziteta, a korisnik unese podatak *singidunum* malim slovima, šabloni dodeljeni promenljivim *$sablon1* i *$sablon2* u prethodnom primeru neće detektovati validan unos usled osetljivosti na velika i mala slova (engl. c*ase sensitive regex*).

```
$sablon1 = '#^(Univerzitet|Singidunum)#i';
```

Ovakvo ponašanje se može korigovati primenom modifikatora *i*, koji isključuje ovakvu osetljivost (engl. *case insensitive regex*).

U slučajevima kada je potrebno da se pomoću regularnih izraza proveri pozicija podstringa na početku linije u tekstu sa više linija, koristi se modifikator *m* (engl. *multiline mode*), slika 38.

Slika 38 - Primena modifikatora na tekstu sa više linija

```
slike/
slike/singidunum-logo.png
css/
css/slike/
css/slike/singidunum.png

#^slike/#m
```

U praksi se takođe često koriste modifikatori *s* i *u*. Prvi daje instrukciju da metakarakter tačka u pretragu uključi i prelom linije, što je korisno prilikom obrade velikih blokova teksta. Modifikator *u* označava da se karakteri u šablonu tretiraju kao UTF-8. Po potrebi, u šablonu se može koristiti više modifikatora.

9.3 FUNKCIJE ZA RAD SA REGULARNIM IZRAZIMA

PHP programski jezik omogućava rad sa regularnim izrazima kroz veći broj osnovnih funkcija čija imena većinom počinju prefiksom "*preg_*".

Listing 93 - Upotreba *preg_match* funkcije

```
<?php
$tekst = '<h1>Univerzitet Singidunum</h1>';
$rezultat = preg_match('#^<h1>(.*?)</h1>$#', $tekst, $m);
print_r($m);
?>
```

```
Array
(
    [0] => <h1>Univerzitet Singidunum</h1>
    [1] => Univerzitet Singidunum
)
```

Funkcija *preg_match*(*$sablon, $tekst, $m*) će izvršiti proveru da li se podstring koji zadovoljava regularni izraz zadat promenljivom *$sablon* nalazi u stringu *$tekst*. Ukoliko je podstring pronađen, biće smešten u niz *$m*, listing 93. Funkcija će vratiti vrednost *1* ukoliko je pronađen odgovarajući rezultati, *0* ako nema rezultata i *false* u slučaju da je došlo do greške.

Kada je potrebno da se u zadatom tekstu pronađu sva mesta koja odgovaraju zadatom šablonu, koristi se PHP funkcija *preg_match_all()*.

Listing 94 - Upotreba *preg_match_all* funkcije

```
<?php
$tekst = '
<h2>Univerzitet Singidunum</h2>
<div>Danijelova 32, Beograd</div>
';
$rezultat = preg_match_all('#^<(\w+)>(.*?)</\1>#m', $tekst, $m);
print_r($m);
?>


Array
(
    [0] => Array
        (
            [0] => <h2>Univerzitet Singidunum</h2>
            [1] => <div>Danijelova 32, Beograd</div>
        )

    [1] => Array
        (
            [0] => h2
            [1] => div
        )

    [2] => Array
        (
            [0] => Univerzitet Singidunum
            [1] => Danijelova 32, Beograd
        )
)
```

U primeru na listingu 94, regularni izrazi se koriste da bi se iz HTML koda veb stranice izdvojili upareni HTML tagovi i segmenti teksta koji im pripadaju. Atomi u šablonu mogu da se protumače na sledeći način:

- < - Otvoren HTML tag
- (\w+) - Prva grupa, alfanumerički karakteri, izdvoji ime HTML taga
- > - Zatvoren HTML tag
- (. *?) - Druga grupa, bilo koji karakter nepohlepno, izdvoji tekst unutar HTML taga
- </ - Otvoren zatvarajući HTML tag
- \1 - Referenca ka prvoj grupi, mora biti isti sadržaj
- > - Zatvoren zatvarajući HTML tag

Listing 95 - Upotreba *preg_split* funkcije

```
<?php
$tekst = '
1. Univerzitet Singidunum, Beograd
2. Univerzitet Singidunum, centar Niš
3. Univerzitet Singidunum, centar Novi Sad
';
$rezultat = preg_split('#[\n\r]^\d+\. #m', $tekst);
print_r($rezultat);
?>

Array
(
    [0] =>
    [1] => Univerzitet Singidunum, Beograd
    [2] => Univerzitet Singidunum, centar Niš
    [3] => Univerzitet Singidunum, centar Novi Sad
)
```

U poglavlju 2.10.3 je opisano kako string može da se transformiše u niz primenom PHP funkcije *explode()*, pod uslovom da je razdelnik jednoobrazan. U slučaju kada je razdelnik promenljiv, ali ga je moguće predstaviti regularnim izrazom, koristi se PHP funkcija *preg_split()*, listing 95.

U ovom primeru se obrađuje numerisana lista koju čine tri člana, pri čemu je svaki od njih napisan u novom redu. Transformacija ovakvog stringa u niz može da se realizuje primenom funkcije *explode*("\n", *$tekst*), pri čemu će uz svaki član ostati i redni broj. Ukoliko je potrebno izdvojiti samo nazive centara bez rednog broja, transformacija će biti efikasnija upotrebom regularnih izraza. Atomi šablona mogu da se protumače na sledeći način:

- *[\n\r]* - klasa koja sadrži jedan karakter \n ili \r. Većini savremenih procesora regularnih izraza je dovoljan samo znak "\n".
- ^ - početak linije
- \d+ - jedan ili više brojeva
- \. - karakter tačka praćen horizontalnim razmakom

Na sličan način se koriste i ostale PHP funkcije za rad sa regularnim izrazima, kao što su:

Listing 96 - Upotreba *preg_grep* funkcije

```
<?php
$centri = [
    '1. Univerzitet Singidunum, Beograd',
    '2. Univerzitet Singidunum, centar Niš',
    'Univerzitet Singidunum, centar Novi Sad'
];
$rezultat = preg_grep("#^\d+\. #", $centri);
print_r($rezultat);
?>
```

```
Array
(
    [0] => 1. Univerzitet Singidunum, Beograd
    [1] => 2. Univerzitet Singidunum, centar Niš
)
```

- *preg_replace* - zamena dela stringa.
- *preg_replace_callback* - zamena dela stringa upotrebom anonimne funkcije.
- *preg_grep* - pretraga niza, listing 96.

OTKLANJANJE GREŠAKA

10. OTKLANJANJE GREŠAKA

Otklanjanje grešaka (engl. *debugging*) predstavlja proces identifikacije i ispravljanja nedostataka odnosno bagova (engl. *bug*) u aplikaciji, koji ometaju njeno izvršavanje, troše resurse ili proizvode druge neželjene rezultate. Koliko će ovaj proces biti zahtevan zavisi od veličine i stepena složenosti aplikacije ili sistema u celini, kao i od tipa greške koja može biti

- Greška u kodu
- Greška u metodologiji

10.1 GREŠKE U KODU

PHP programski jezik poseduje dobro razrađen mehanizam prijavljivanja grešaka u radu aplikacije, bilo da su u pitanju jednostavni propusti u pisanju koda (engl. *syntax error*), pozivi nedeklarisanih promenljivih, problemi u komunikaciji sa serverom baze podataka i sl.

Da bi se problem rešio na odgovarajući način, prvi korak je da se utvrdi vrsta problema, ali i njegova lokacija koju čine dva podatka, ime fajla i linija koda. Zato je neophodno voditi računa o porukama koje PHP ispisuje na ekranu i beleži u log fajlovima.

Listing 97 - Upotreba nedeklarisane promenljive

```
1 <?php
2 $x = 1;
3 $z = $x + $y;
4 echo $z;
5 ?>
```

10.1.1 Obaveštenja

Obaveštenja su beleške o sitnim nepravilnostima koje su primećene prilikom izvršavanja PHP koda i ne prekidaju izvršavanje aplikacije. Najčešće se javljaju prilikom upotrebe nedeklarisanih promenljivih poput promenljive *$y* u primeru na listingu 97, linija 3:

```
Notice: Undefined variable: y in C:\www\index.php on line 3
```

Prikazano obaveštenje se može protumačiti na sledeći način:

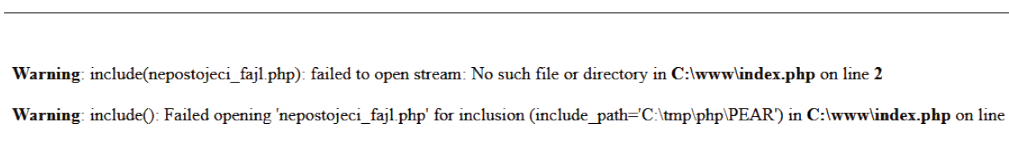
- *Notice* - tip greške
- Undefined variable: *y* - opis greške
- *C:\www\index.php* - fajl u kojem je greška otkrivena
- *line 3* - linija koda u kojoj je greška otkrivena

10.1.2 Upozorenja

Upozorenje u PHP programskom jeziku ne zaustavlja izvršavanje same aplikacije, već samo prijavljuje mesto na kome potencijalno može nastati problem prilikom daljeg razvoja softvera. Ovakav tip greške je najčešće izazvan zadavanjem neodgovarajućih parametara prilikom pozivanja funkcije, ili pokušajem uključivanja u izvršavanje programa nepostojećeg fajla. Kod

```
<?php
include('nepostojeci_fajl.php');
?>
```

Slika 39 - Upozorenje o grešci u kodu



će izazvati prikaz upozorenja na ekranu, slika 39. Beleška o ovom događaju će biti zabeležena i u dokumentu *error.log*.

Listing 98 - Greška u sintaksi

```
1 <?php
2 $x = 1;
3 $y = 2
4 $z = 3;
5 ?>
```

Parse error: syntax error, unexpected '$z' (T_VARIABLE) in **C:\www\index.php** on line **4**

10.1.3Greške u sintaksi

Greške u sintaksi nastaju kada su u kodu neuparene zagrade ili navodnici, ukoliko je između naredbi izostavljen znak ";", ukoliko su izostavljeni očekivani znakovi i sl. U tom slučaju kompajler zaustavlja izvršavanje koda i prijavljuje grešku, listing 98.

Listing 99 - Kritična greška

```
1 <?php
2 function obradi($a, $b) {
3     return $a + $b;
4 }
5 saberi(3, 2);
6 ?>
```

Fatal error: Call to undefined function saberi() in **C:\www\index.php** on line **5**

10.1.4 Kritične greške

Kritična greška (engl. *Fatal Error*) je greška koja takođe dovodi do prekida izvršavanja programa, a javlja se npr. prilikom poziva nedeklarisane funkcije ili klase, listing 99. Kritične greške se mogu podeliti u tri podgrupe:

- Kada sistem ne može da pokrene kod (engl. *Startup fatal error*)
- Greška koja je nastala u fazi kompajliranja (engl. *Compile time fatal error*)
- Greška nastala u toku izvršavanja aplikacije (engl. *Runtime fatal error*)

10.1.5 Kontrola prikaza grešaka

Podaci o otkrivenim greškama u kodu se beleže u *error.log*[24] dokumentu, koji se obično nalazi u osnovnom (*root*) direktorijumu projekta. Osim toga, poruke o greškama se prikazuju i u veb brauzeru. Proces beleženja grešaka se može kontrolisati:

- Na nivou veb servera
- U okviru same aplikacije

Na nivou veb servera, prikaz grešaka se podešava u PHP konfiguracionom fajlu *php.ini,* postavljanjem vrednosti za sledeće direktive:

- *display_errors* - kontroliše prikaz grešaka. Moguće vrednosti su *On* ili *Off.*
- *display_startup_errors* - kontroliše prikaz grešaka prilikom pokretanja samog PHP programskog jezika.
- *log_errors* - kontroliše proces beleženja grešaka u odgovarajućim log fajlovima.
- *log_errors_max_len* - maksimalni broj karaktera koji može da sadrži poruka.

Predefinisana vrednost direktiva za prikaz i logovanje grešaka je *On*, i preporučuje se za razvojno okruženje. Međutim, u produkcionom okruženju vrednost direktiva *display_errors* i *display_startup_errors* mora biti obavezno podešena na *Off*.

24 Ime dokumenta može varirati, u skladu sa podešavanjima veb servera i virtuelnog domena

Tokom izvršavanja aplikacije, za kontrolu prikaza grešaka se može koristiti PHP funkcija *error_reporting*(*$nivo*). Parametar funkcije *$nivo* određuje nivo grešaka koje mogu biti prikazane tokom izvršavanja programa, npr.[25,26]

- Prikazivanje grešaka isključeno

```
error_reporting(0);
```

- Prikaz svih grešaka, osim obaveštenja

```
error_reporting(E_ALL & ~E_NOTICE);
```

- Prikaz svih grešaka

```
error_reporting(E_ALL);
```

Listing 100 - Testiranje funkcije

```
1 <?php
2 function zbir($a, $b) {
3     return $a - $b;
4 }
5 $zbir = zbir(2, 3);
6 print $zbir;
7 exit;
8 ?>
```

25 https://www.php.net/manual/en/function. error-reporting.php
26 https://www.php.net/manual/en/errorfunc. configuration.php#ini. error-reporting

10.2 GREŠKE U METODOLOGIJI

Postoji više metoda za testiranje koda i otkrivanje metodoloških grešaka, u skladu sa stepenom složenosti aplikacije sa kojom se radi. Najjednostavniji način je direktan ispis trenutnih vrednosti na ekranu i zaustavljanje aplikacije primenom PHP funkcija *print, echo, print_r* ili *var_dump*, listing 100. U ovom primeru se testira ispravnost rezultata koji vraća funkcija *zbir()*, tako što će rezultat biti ispisan na ekranu (linija 6), a aplikacija zaustavljena (linija 7).

Pokretanjem aplikacije, za vrednosti parametara funkcije 2 i 3 se dobija rezultat -1. Aplikacija je izvršena, što znači da nema grešaka u sintaksi, ali rezultat očigledno nije tačan zato što je očekivana vrednost 5. Time je utvrđeno da kod u samoj funkciji (linija 3) nije dobro napisan, i ponovnim pregledom se otkriva da je u toj liniji primenjena pogrešna aritmetička operacija oduzimanja, umesto sabiranja.

Listing 101 - Praćenje poziva određene funkcije

```
1 <?php
2 function zbir($a, $b) {
3     print_r(debug_print_backtrace());
4     print "<br>";
5     return $a + $b;
6 }
7 $zbir = zbir(1, -1);
8 $zbir = zbir(2, 3);
9 print $zbir;
10 ?>
```

Slika 40 - Podaci o pozivima funkcije

```
#0 zbir(1, -1) called at [C:\www\index.php:7]
#0 zbir(2, 3) called at [C:\www\index.php:8]
5
```

Na sličan način se koriste i druge PHP funkcije:

- *get_defined_vars()* - vraća spisak svih trenutno deklarisanih promenljivih. Da bi se spisak prikazao na ekranu, mora se upotrebiti PHP funkcija *print_r()*.
- *var_dump($promenljiva)* - daje informaciju o tipu i trenutnoj vrednosti promenljive.
- *print_r($niz)* - ispisuje sadržaj matrice *$niz*.
- *debug_print_backtrace()* - pruža informaciju o tome iz kog fajla i iz koje linije koda je funkcija pozvana, listing 101 i slika 40.

Osim upotrebe osnovnih PHP funkcija, za potrebe testiranja aplikacije prilikom razvoja se mogu koristiti jedno ili više namenski razvijenih softverskih rešenja, kao što su *Xdebug*, *ZendDebugger* i sl.

Listing 102 - Aktiviranje izuzetka

```php
<?php
function podeli($deljenik, $delilac) {
    if ($delilac == 0) {
        throw new Exception("Deljenje sa nulom!");
    }
    return $deljenik / $delilac;
}
try {
    echo podeli(2, 0);
} catch (Exception $e) {
    $poruka = $e->getMessage();
    print $poruka;
    /* print_r($e); */
}
?>
```

```
Deljenje sa nulom!
```

Slika 41 - Struktura *Exception* objekta

```
Exception Object
(
    [message:protected] => Deljenje sa nulom!
    [string:Exception:private] =>
    [code:protected] => 0
    [file:protected] => C:\www\index.php
    [line:protected] => 4
    [trace:Exception:private] => Array
        (
            [0] => Array
                (
                    [file] => C:\www\index.php
                    [line] => 9
                    [function] => podeli
                    [args] => Array
                        (
                            [0] => 2
                            [1] => 0
                        )

                )

        )

    [previous:Exception:private] =>
)
```

10.3 IZUZECI

Izuzetak (engl. *exception*) je objekat koji opisuje grešku ili neočekivano ponašanje PHP skripte, mogu ga generisati PHP funkcije ili klase, i predstavlja način na koji mogu da se detektuju greške, i potom kontroliše ili obustavi izvršavanje koda.

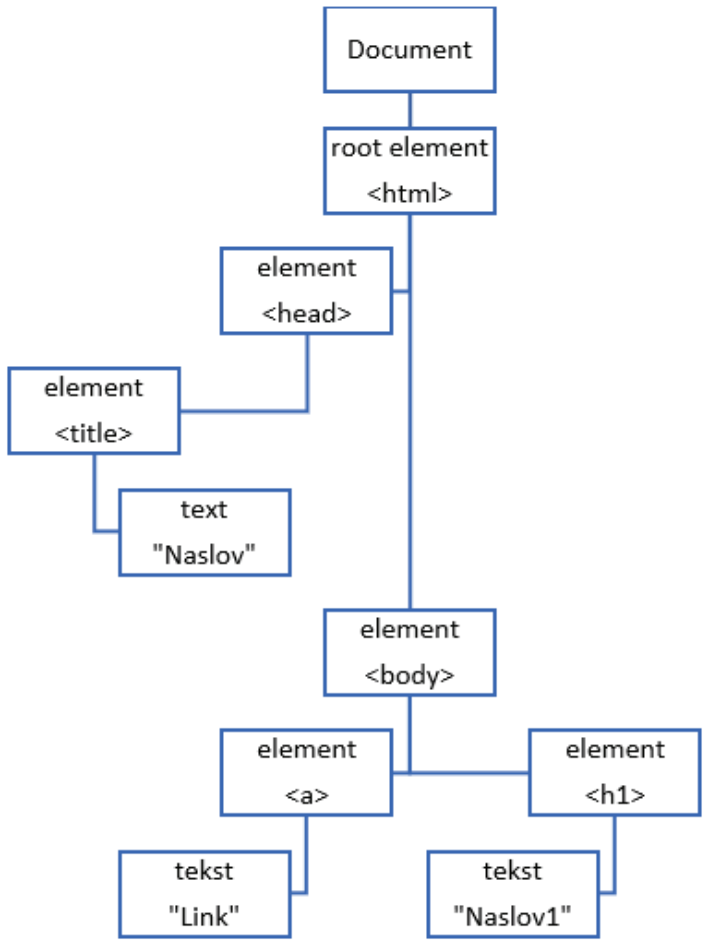
Na listingu 55 je prikazana jednostavna funkcija *podeli*, čiji je zadatak da izračuna i vrati količnik ulaznih parametara. Ukoliko se prilikom poziva funkcije kao delilac navede nula, predviđena računska operacija ne može da se sprovede. Zato se u funkciji prvo proverava da li je vrednost promenljive *$delilac* jednaka nuli (linija 3). Ukoliko to zaista jeste slučaj generisaće se objekat *Exception* (linija 4), koji se potom može obraditi u glavnom toku programa, linije 10-14. Uklanjanjem komentara sa linije 13, u okviru veb stranice će se ispisati struktura objekta *Exception*, slika 41.

FRONT-END TEHNOLOGIJE

11. FRONT-END TEHNOLOGIJE

U ovom poglavlju prikazane su odabrane teme iz domena JavaScript i Ajax tehnologija. Pretpostavka je da su studenti upoznati sa osnovama JavaScript-a, kao i HTML/CSS tehnologijama, koje su izučavali na kursevima koje su slušali na osnovnim studijama Univerziteta Singidunum.

Slika 42 - Prikaz *HTML DOM* stabla



11.1 JAVASCRIPT HTML DOM

U ovom podpoglavlju, kroz praktične primere, prikazana je interakcija sa HTML DOM (engl. *Document Object Model*) pomoću JavaScript tehnologije. HTML DOM je objektni model HTML-a, koji definiše:

- HTML elemente kao objekte;
- svojstva (engl. *properties*) za sve HTML elemente;
- metode (engl. *methods*) za sve HTML elemente i
- događaje (engl. *events*) za sve HTML elemente.

Pomoću HTML DOM-a, JavaScript može da pristupi i izmeni sve elemente u jednom HTML dokumentu. Odmah nakon učitavanja veb strane, pretraživač (engl. *browser*) kreira DOM veb strane. HTML DOM je predstavljen kao stablo objekata, čiji je prikaz dat na slici 46.

HTML DOM omogućava JavaScript jeziku da kreira dinamičku HTML stranu tako što:

- JavaScript može da promeni sve HTML elemente na stranici;
- JavaScript može da promeni sve HTML atribute na stranici;
- JavaScript može da promeni sve CSS stilove na stranici;
- JavaScript može da obriše HTML elemente i atribute;
- JavaScript može da doda nove HTML elemente i atribute;
- JavaScript može da reaguje na događaje HTML strane i
- JavaScript može da kreira nove HTML događaje na strani.

DOM je W3C (World Wide Web Consortium) standard, koji definiše standard za pristup dokumentima. Prema jednoj od najšire korišćenih definicija DOM-a, koju je dao W3C, DOM se definiše kao *platforma i interfejs koji nije vezan ni za jedan jezik i koji omogućava programima i skriptama da dinamički pristupaju i ažuriraju sadržaj, strukturu i stil dokumenta*.

- *W3C DOM* standard može da se podeli na tri celine:
- *CORE DOM* - standardni model za sve tipove dokumenata;
- *XML DOM* - standardni model za XML (eXtensible Markup Language) dokumente i
- *HTML DOM* - standardni model za HTML dokumente.

HTML DOM je standard za pristup, izmenu (ažiriranje), dodavanje i brisanje HTML elemenata.

Kao što je već i navedeno, gledanjem na HTML elemente kao objekte, HTML DOM definiše svojstva, metode i događaje svih HTML elemenata. HTML DOM metode predstavljaju akcije koje mogu da se izvrše sa HTML elementima. S druge strane, HTML DOM svojstva su vrednosti HTML elemenata koja mogu da se deifnišu (engl. *set*) i izmene (engl. *update*).

11.1.1 Pristup objektima HTML DOM-a pomoću JavaScript-a

Za pristup HTML DOM-u može da se koristi JavaScript, ili neki drugi programski ili skript jezik. Interfejs ka objektima su svojstva i metode svakog objekta. Svojstvo je vrednost koja može da se dodeli ili preuzme, kao na primer promena sadržaja HTML elementa. Metoda je, kao što je poznato, akcija koja može da se izvrši, kao što je na primer brisanje i dodavanje HTML elemenata na stranu.

Jedan od najčešće korišćenih metoda je *getElementById()*, pomoću koje se pristupa određenom elementu HTML strane na osnovu njegovog id atributa. Jedno od najšešće korišćenih svojstava je innerHTML svojstvo, pomoću koga se menja sadržaj HTML elementa. Na listingu 157 dat je primer korišćenja ove metode i svojstva.

Listing 103 - Primer sa *getElementById() metodom i innerHTML* svojstvom

```
<html>
<body>
<div id="test"></div>
<script>
  document.getElementById("test").innerHTML=
    "Today we learn <b> JS DOM </b>!";
</script>
</body>
</html>
```

Jedan od najlakših načina da se pristupi bilo kom HTML elementu sa HTML stranice je *getElementById()* metoda. Tako na primer, kod

```
Document.getElementById("test")
```

označava da se u celom HTML dokumentu traži element čiji je id (jedinstveni identifikator) test. Svojstvo *innerHTML* označava sadržaj HTML elementa, u prethodnom primeru, to je sadržaj između <*div*> i <*/div*> tagova. Svojstvo innerHTML može da se koristi i da se promeni sadržaj bilo kojih HTML elemenata, čak i <*html*> i <*body*> elemenata.

HTML DOM objekat document je "vlasnik" svih ostalih objekata na veb strani. Drugim rečima, objekat *document* predstavlja celu veb stranicu. Ako je potrebno da se pristupi nekom objektu u okviru HTML stranice, pretraga uvek počinje iz objekta *document*.

Tri najčešće korišćene metode za pretragu objekata su:

- *document.getElementById(id)* - pronalazi element na osnovu vrednosti njegovog id atributa;
- *document.getElementsByTagName(name)* - pronalazi elemente na osnovu naziva taga (na primer: *p, table, tr, td* itd.) i
- *document.getElementsByClassName(name)* - pronalazi elemente na osnovu naziva klase.

Najčešće korišćeni načini za izmenu HTML elemenata:

- *element.innerHTML = new html* content - menja vrednost unutrašnjeg HTML-a određenog elementa;
- *element.attribute = new value* - menja vrednost atributa određenog HTML elementa;
- *element.setAttribute(attribute, value)* - menja vrednost atributa određenog HTML elementa i
- *element.style.property = new style* - menja stil određenog HTML elementa.

Najčešći načini za brisanje i dodavanje HTML elemenata:

- *document.createElement(element)* - kreira novi HTML element;
- *document.removeChild(element)* - briše HTML element;
- *document.appendChild(element)* - dodaje HTML element;
- *document.replaceChild(element)* - vrši zamenu HTML elementa i
- *document.write(text)* - upisuje podatke u izlazni HTML tok.

Na listingu 104 dat je primer pronalaženja HTML elementa pomoću vrednosti id atributa, metoda *getElementById(id)*.

Listing 104 - Ciljanje elementa i izmena sadržaja, *getElementById()* i *innerHML*

```
<html>
<body>
  <p id="introduction">
    This is JS DOM introduction class!
  </p>
  <p id="test"> </p>
  <script>
    var p=document.getElementById("introduction");
    var text="Text from the " +
             "<i> Introduction </i> paragraph is: " +
             p.innerHTML;
    document.getElementById("test").innerHTML=text;
  </script>
</body>
</html>
```

Na listingu 105 je prikazan primer pronalaženja HTML elementa na osnovu naziva taga (tipa elemenata) primenom metode *getElementsByTagNames()*.

Listing 105 - Ciljanje elementa, metoda *getElementsByTagName()*

```
<html>
<body>
<p>This is JS DOM introduction class!</p>
<p>This class is very useful!</p>
<p>We will learn a lot about JS HTML DOM!</p>
<p id="test"> </p>
<script>
  // pronalazi sve <p> tagove i smesta u niz p
  var p=document.getElementsByTagName("p");
  // p elementima pristupamo preko index-a niza p
  var text=
    "First paragraph: <b> " +
    p[0].innerHTML +
    "</b><br/>";
  text+=
    "Second paragraph: <b> " +
    p[1].innerHTML +
    "</b><br/>";
  text+=
    "Third paragraph: <b> " +
    p[2].innerHTML +
    "</b><br/>";
  /* pronalazimo element sa id=test
     i postavljamo innerHTML svojstvo */
  document.getElementById("test").innerHTML=text;
</script>
</body>
</html>
```

Primer na listingu 106 prikazuje pronalaženje HTML elemenata na osnovu klasa kojoj pripadaju primenom metode *getElementsByClassName()*.

Listing 106 - Ciljanje elemenata, *getElementsByClassName()*

```
<html>
<body>
  <p class="test">This is JS DOM introduction class!</p>
  <p class="test">This class is very useful!</p>
  <p class="test">We will learn a lot about JS HTML DOM!</p>
  <p id="test"> </p>
  <script>
    /* pronalazi sve tagove sa
       class="test" i smesta u niz p */
    var p=document.getElementsByClassName("test");
    /* p elementima pristupamo preko index-a niza p */
    var text =
      "First paragraph: <b> " +
      p[0].innerHTML +
      "</b><br/>";
    text +=
      "Second paragraph: <b> " +
      p[1].innerHTML +
      "</b><br/>";
    text +=
      "Third paragraph: <b> " +
      p[2].innerHTML +
      "</b><br/>";
    /* pronalazimo element sa
       id=test i postavljamo innerHTML svojstvo */
    document.getElementById("test").innerHTML=text;
  </script>
</body>
</html>
```

HTML elementima može da se pristupi pomoću vrednosti CSS selektora, što je prikazano u listingu 107.

Listing 107 - Ciljanje HTML elemenata, *CSS selektor querySelectorAll()*

```
<html>
<body>
  <p>Hello everybody!</p>
  <p class="test">
    We learn DOM today.
  </p>
  <p class="test">
    Now we show <b>querySelectorAll</b>
    method for accessing HTML elements!
  </p>
  <p id="test"></p>
  <script>
    var p = document.querySelectorAll("p.test");
    document.getElementById("test").innerHTML =
      'The first paragraph (index 0) with class="test": ' +
      p[0].innerHTML +
      "<br/>" +
      'The second paragraph (index 1) with class="test": ' +
      p[1].innerHTML;
  </script>
</body>
</html>
```

Na listingu 108, dat je primer pronalaženja svih HTML elemenata koji pripadaju kolekciji HTML forme. HTML forma je kolekcija objekata, gde svaki objekat (element) ima svoju vrednost. Tako je na primer vrednost za input tekstualno polje tekst koji upiše korisnik, vrednost za select HTML element je opcija koju je korisnik izabrao, itd.

Listing 108 - Pronalaženje svih HTML elemenata koji pripadaju istoj HTML formi

```
<html>
<body>
  <form  id="form1">
    Enter name: <input type="text" name="name" value="John"><br/>
    Enter surname <input type="text" name="surname" value="Doe"><br/>
    <select name="age">
      <option value="21-30">21-30</option>
      <option value="31-40">31-40</option>
      <option value="41-50">41-50</option>
    </select>
  </form>
  <p id="test"></p>
  <button type="button" onclick="test()"> Click me </button>
  <script>
    function test(){
       var x = document.forms["form1"];
        var text = "";
        var i;
        for (i = 0; i < x.length ;i++) {
          text += x.elements[i].value + "<br>";
        }
        document.getElementById("test").innerHTML=text;
    }
  </script>
</body>
</html>
```

Primenom tehnologije JavaScript, dostupna je i metoda *document.write()* koja omogućava pisanje direktno u HTML izlazni tok podataka. Ova metoda bi trebla da se koristi nakon što je HTML dokument učitan, jer će u suprotnom biti prepisan (engl. *overwrite*) sadržaj originalnog dokumenta. Primer sa ovom metodom dat je u listingu 109.

Listing 109 - Primer sa *document.write() metodom*

```
<html>
<body>
  <script>
    document.write(
      "Hello there everybody, " +
      "I am in the output HTML stream!<br/>"
    );
    document.write(Date());
  </script>
</body>
</html>
```

Pomoću setera atributa moguće je promeniti vrednost HTML atributa. Sintaksa: *document.getElementById(id). attribute = new value,* gde je attribute naziv svojstva koje se menja. Primer gde se slika desktop računara zamenjuje slikom laptop računara dat je u listingu 110.

Listing 110 - Promena vrednosti svojstva HTML objekta

```
<html>
<body>
  <img
    src="desktop.jpg"
    width="180px"
    height="180px"
    style="border:2px solid black;display:block;"
    id="figure1"
  />
  <button type="button" onclick="test()">
    Change picture
  </button>
  <script>
    function test(){
      document.getElementById("figure1").src="laptop.jpg";
    }
  </script>
</body>
</html>
```

Slično kao u prethodnom primeru sa atributom HTML objekta, pomoću odgovarajućeg setera moguće je promeniti i vrednost svojstva CSS stila. Sintaksa: *document.getElementById(id).style.property = new style*, gde je property naziv CSS svojstva. Primer je dat u listingu ispod.

Listing 111 - Promena vrednosti svojstva CSS stila

```
<html>
<body>
  <p id="p1">Hello everyone</p>
  <p id="p2">Hello everyone!</p>
  <script>
    document.getElementById("p2").style.color = "red";
    document.getElementById("p2").style.fontFamily = "Arial";
    document.getElementById("p2").style.fontSize = "larger";
  </script>
  <p>The paragraph was changed by a script!</p>
</body>
</html>
```

11.1.2 HTML DOM događaji

HTML DOM omogućava izvršavanje predefinisanog koda kada se određeni događaj (engl. *event*) dogodi. Neki od događaja koji se generišu od strane pretraživača kada se "nešto dogodi" HTML elementima su:

- kada korisnik klikne na element;
- kada se strana učita i
- kada polja za unos (engl. *input fields*) promene vrednost

HTML DOM omogućava da JavaScript reaguje na HTML događaje. JavaScript kod može da bude izvršen kada se desi neki događaj, na primer kada korisnik mišem pređe preko određenog elementa (onmouseover događaj). Da bi se implementirala pomenuta funkcionalnost, kao vrednost HTML atributa treba da se doda JavaScript kod. Sintaksa:

```
onclick=JavaScript; event=JavaScript;
```

Kao još neki od primera HTML događaja, mogu da se navedu sledeći: kada se slika učita, kada je miš iznad elementa, kada se input polje promeni, kada se pošalje HTML forma, kada korisnik pritisne taster na tastaturi i slično.

Primeri sa jednostavnim događajima navedeni su ispod, listinzi 112-114.

Listing 112 - Primer jednostavnog događaja - I

```
<html>
<body>
  <h1 id="id1">Our heading H1</h1>
  <button
    type="button"
    onclick="document.getElementById('id1').style.color = 'red'"
  >
    Click Me!
  </button>
</body>
</html>
```

Listing 113 - Primer jednostavnog događaja - II

```
<html>
<body>
  <h1 onclick="this.innerHTML='Heading has changed'">
    Click on this text!
  </h1>
  <h1 onclick="changeIt(this)">
    This is heading 1
  </h1>
  <script>
  function changeIt(id) {
      id.innerHTML = "The heading has changed";
      id.style.color="Blue";
  }
  </script>
</body>
</html>
```

Listing 114 - Primer jednostavnog događaja - III

```
<html>
<body>
  <p>Example of onclick event</p>
  <button onclick="displayDate()">Current time is?</button>
  <script>
  function displayDate() {
    document.getElementById("demo").innerHTML = Date();
  }
  </script>
  <p id="demo"></p>
</body>
</html>
```

HTML DOM omogućava da JavaScript doda događaje HTML elementima, tako na primer elmentu paragraf (<*p*>) može da se dodeli događaj *onmouseover*, što je prikazano u listingu 115.

Listing 115 - Dodavanje događaja HTML elementima

```
<html>
<body>
  <p id="test">
    This is just a test paragraph.
    Mouse over this paragraph
  </p>
  <script>
  document.getElementById("test").onmouseover=test;
  function test(){
    document.getElementById("test").style.color="Blue";
    document.getElementById("test").style.fontSize="18pt";
  }
  </script>
</body>
</html>
```

Događaj *onchange* se u najvećem broju slučajeva koristi u kombinaciji sa validacijom input polja, listing 116. Događaji *onmousedown*, *onmouseup* i *onclick* pripadaju događajima koji se vezuju za radnje sa mišem. Kada korisnik klikne, pokreće se događaj *onmousedown*, a kada korisnik pusti dugme miša pokreće se događaj *onmouseup*. Konačno kada se ova sekvenca završi, pokreće se događaj *onclick*.

Listing 116 - Primer *onchange* događaja

```
<html>
<head>
<script>
function myFunction() {
    var x = document.getElementById("fname");
    x.value = x.value.toUpperCase();
}
</script>
</head>
<body>
  Enter your name:
  <input type="text" id="fname" onchange="myFunction()">
  <p>
    When you leave the input field, a function is triggered
    which transforms the input text to upper case.
  </p>
</body>
</html>
```

Listing 117 - Primer *onmouseup* i *onmousedown* događaja

```
<html>
<body>
  <div
    onmousedown="mDown(this)"
    onmouseup="mUp(this)"
    style="background-color:#D94A38;
      width:90px;height:20px;padding:40px;">
    Click Me
  </div>
  <script>
    function mDown(obj) {
        obj.style.backgroundColor = "#1ec5e5";
        obj.innerHTML = "Release Me";
    }
    function mUp(obj) {
        obj.style.backgroundColor="#D94A38";
        obj.innerHTML="Thank You";
    }
  </script>
</body>
</html>
```

Metoda *addEventListener()* vezuje određeni upravljač događaja (engl. *event handler*) za određeni element. Ova metoda ne prepisuje postojeće upravljače događaja. Za jedan element može da se veže više upravljača događaja. Takođe za jedan element može da se veže više upravljača događaja istog tipa, na primer *onmousedown* događaji.

Metoda *addEventListener()* može da se veže za bilo koji objekat, ne samo za HTML element (na primer za bilo koji BOM - *Browser Object Model* objekat). Kada se koristi metoda *addEventListener()*, JavaScript je odvojen od HTML-a, što čini kod preglednijim i čitljivijim. Za uklanjanje koristi se metoda *removeEventListener()*. Sintaksa:

```
element.addEventListener(event, function, useCapture);
```

Prvi parametar je tip događaja (na primer *mousedown, click, mouseup*, *mouseout, mousein*, itd). Drugi parametar je funkcija koja se poziva kada se određeni događaj dogodi. Treći parametar je boolean vrednost koja određuje da li događaj žubori (engl. *bubbling*) ili se hvata (engl. *capture*). Ovaj parametar je opcioni.

Listing 118 - Primer sa *addEventListener* metodom - I

```
<html>
<body>
  <p>
    Here we attache event listener
    when user clicks on a button
  </p>
  <button id="btn1">Test it</button>
  <script>
  document.getElementById("btn1").addEventListener(
    "click", function(){
      alert("We learn JS DOM!");
    }
  );
  </script>
</body>
</html>
```

Listing 119 - Primer sa *addEventListener metodom* - II

```
<html>
<body>
  <p>Here we add three eventListener to the same object button</p>
  <button id="btn1">Test it</button>
  <script>
    var x = document.getElementById("btn1");
    /* dogadjaj (eng. event) je click, a myClick je
      upravljac dogadjana (engl. event handler) */
    x.addEventListener("click", myClick);
    x.addEventListener("click", myClick1);
    x.addEventListener("click", myClick2);
    function myClick() {
      alert ("First click handler");
    }
    function myClick1() {
      alert ("Second click handler!");
    }
    function myClick2() {
      alert ("Third click handler");
    }
  </script>
</body>
</html>
```

Listing 120 - Primer sa *addEventListener* metodom - III

```
<html>
<body>
  <!--
    sada dodajemo eventListener za
    dogadjaj resize browser objekata
  -->
  <p>
    Here we add eventListener to the
    browser object's event resize
  </p>
  <p>
    When we resize the windows,
    the event is triggered
  </p>
  <p id="test"></p>
  <script>
    window.addEventListener("resize", function(){
      document.getElementById("test").innerHTML =
        Math.round(Math.random());
  });
  </script>
</body>
</html>
```

Listing 121 - Primer sa *addEventListener* metodom - IV

```
<html>
<body>
  <!--
    U ovom primeru prosledjujemo
    parametre funkcije eventHandler
  -->
  <p>
    Here we pass parameters to the
    eventHandler function
  </p>
  <p>Click on this button </p>
  <button id="btn1">Click it</button>
  <p id="test"></p>
  <script>
    var x=20;
    var y=30;
    /* parametre ne mozemo direktno da prosledjujemo,
       vec u okviru eventHandler funkcije pozivamo
       funkciju kojoj prosledjujemo parametre */
    document.getElementById("btn1").addEventListener(
      "click",
      function(){
        myFunction(x,y)
      }
    );
    function myFunction(a, b) {
      var result = a * b;
      document.getElementById("test").innerHTML = result;
    }
  </script>
</body>
</html>
```

Nekoliko primera dodavanja upravljača događaja sa iskomentarisanim linijama koda prikazano je u listinzima 118-121.

Postoje dva mehanizma koji određuju propagaciju događaja u HTML DOM-u: žuborenje (engl. *bubbling*) i hvatanje (engl. *capturing*). Popagacija događaja je način kojim se određuje redosled reakcije elemenata kada se događaj dogodi.

Ako na primer na HTML strani postoji <*h1*> element u okviru <*div*> elementa, gde oba elementa imaju događaj click, i kada korisnik klikne na sadržaj <*h1*> elementa, koji upravljač događaja će prvi da bude pozvan (za *h1* ili za *div* element)?

U slučaju žuborenja, prvo se upravlja događajem unutrašnjeg elementa (u ovom slučaju *h1*), pa tek onda spoljašnjeg (u ovom slučaju *div*).

U slučaju hvatanja, je obrnuto, prvo se upravlja događajem spoljašnjeg (u ovom slučaju *div*), pa tek onda unutrašnjeg elementa (u ovom slučaju *h1*).

Treći parametar *addEventListener()* metode određuje se tip propagacije događaja. Sintaksa:

```
element.addEventListener(event, function, useCapture);
```

Po default-u je treći parameter postavljena vrednost *false*, što znači da se koristi žuboreća propagacija. Kada je vrednost trećeg parametra postavljena na *true*, koristi se hvatajuća propagacija događaja.

U listingu 122 je dat primer sa propagacijom događaja.

Listing 122 - Primer sa propagacijom događaja

```
<html>
<head>
  <style>
  div {
    background-color: lightblue;
    border: 1px solid black;
    padding: 60px;
  }
  </style>
</head>
<body>
  <p>THe difference between bubbling and capturing event propagation</p>
  <div id="div1">
    <p id="p1">This parapgraph is bubbling</p>
  </div>
  <br>
  <div id="div2">
    <p id="p2">This paragraph is capturing</p>
  </div>
  <script>
    document.getElementById("p1").addEventListener(
      "click",
      function() {
        alert("You clicked the P element!");
      }, false
    );
    document.getElementById("div1").addEventListener(
      "click",
      function() {
        alert("You clicked the DIV element!");
      }, false
    );
    document.getElementById("p2").addEventListener(
      "click",
      function() {
        alert("You clicked the P element!");
      }, true
    );
    document.getElementById("div2").addEventListener(
      "click",
```

```
41        function() {
42          alert("You clicked the DIV element!");
43        }, true
44      );
45    </script>
46 </body>
47 </html>
```

Konačno, metoda *removeEventListener()* uklanja upravljače događajima koji su vezani za određeni događaj pomoću *addEventListener()* metode. Primer je dat u listingu 123.

Listing 123 - Primer sa *removeEventListener* metodom

```
 1 <html>
 2 <head>
 3  <style>
 4     div {
 5       width: 50%;
 6       padding: 30px;
 7       margin: auto;
 8       background-color: azure;
 9     }
10   </style>
11 </head>
12 <body>
13   <p>Example with removeAddListener() method</p>
14   <div id="div1">
15     <button type="button" id="btn1">Remove eventHandler</button>
16   </div>
17   <p id="test"> </p>
18   <script>
19     document.getElementById("div1").addEventListener(
20       "mousemove",
21       myFunction
22     );
```

```
23      function myFunction(){
24        document.getElementById("test").innerHTML=Math.random();
25      }
26      document.getElementById("btn1").addEventListener(
27        "click",
28        function(){
29          document.getElementById("div1").removeEventListener(
30            "mousemove",
31            myFunction
32          )
33        }
34      )
35    </script>
36  </body>
37  </html>
```

11.1.3 DOM navigacija

Pomoću HTML DOM-a, moguće je prolaziti kroz stablo čvorova (engl. *tree nodes*) korišćenjem veza između čvorova. Prema W3C HTML DOM standardu, svi HTML elementi u dokumentu se prikazuju kao čvor:

- ceo dokument je *document* čvor;
- svaki HTML element je element čvor;
- tekst u HTML elementima su predstavljeni kao *text* čvorovi;
- svaki HTML atribut je čvor tipa *attribute* i
- svi komentari su čvorovi tipa *comment*.

HTML DOM omogućava mehanizme kojima JavaScript pristupa celom stablu čvorova. Svakom čvoru može da se pristupi, čvorovi mogu da se dodaju, brišu i modifikuju.

Svi čvorovi u stablu čvorova su povezani hijerarhijskim vezama. Pojmovi roditelj (engl. *parent*), dete (engl. *child*) i brat/sestra (engl. *sibling*) koriste se za opise veza između čvorova. Glavni čvor se naziva korenski čvor (engl. *root node*).

Svaki čvor ima tačno jednog roditelja, osim korenskog čvora, koji nema roditelja. Svaki čvor može da ima više čvorova dece. Čvorovi braća ili sestre su čvorovi koji imaju istog roditelja. Jednostavni primer hijerarhije čvorova dat je ispod.

```
<html>
<head>
    <title>JavaScript DOM</title>
</head>
<body>
    <h1>Heading 1</h1>
    <p>Hello dear students!</p>
    </body>
</html>
```

Sa prikazanog stabla može da se utvrdi sledeće: <*html*> je korenski čvor, <*html*> nema roditelja, <*html*> je roditelj <*head*> i <*body*> čvorova, <*head*> je prvo dete <*html*> čvora, <*body*> je poslednje dete <*html*> čvora, <*head*> ima jedno dete pod nazivom <*title*>, <*title*> ima jedno dete (tekstualni čvor) pod nazivom "JavaScript DOM", <*h1*> i <*p*> su brat i sestra (ili braća, ili sestre), itd.

Listing 124 - Primer sa DOM navigacijom - I

```
<html>
<body>
  <h1 id="h1">Our Test Page</h1>
  <p id="p1"></p>
  <p id="p2"></p>
  <p id="p3"></p>
  <script>
    document.getElementById("p1").innerHTML =
      document.getElementById("h1").innerHTML;
    document.getElementById("p2").innerHTML =
      document.getElementById("h1").firstChild.nodeValue;
    document.getElementById("p3").innerHTML =
      document.getElementById("h1").childNodes[0].nodeValue;
    /* ne mozemo da pomocu firstChild.nodeValue,
       childNodes[0].nodeValue upisujemo vrednosti,
       jer su childNodes i firstChild.nodeValue
       read-only, za dodeljivanje koristimo innerHTML */
  </script>
</body>
</html>
```

Listing 125 - Primer sa DOM navigacijom - II

```
<!DOCTYPE html>
<html>
<body>
  <p>Hello World!</p>
  <div>
    <p>We learn JavaScript DOM</p>
    <p>
      This example demonstrates the
      <b>document.body</b> property.
    </p>
  </div>
  <script>
    // mozemo da procitamo sadrzaj body elementa
    alert(document.body.innerHTML);
  </script>
</body>
</html>
```

Listing 126 - Primer sa DOM navigacijom - III

```
<html>
<body>
  <p>Hello World!</p>
  <div>
    <p>We learn JavaScript today!</p>
    <p>
      This example demonstrates the
      <b>document.documentElement</b> property.
    </p>
  </div>
  <script>
    alert(document.documentElement.innerHTML);
  </script>
</body>
</html>
```

Za navigaciju kroz stablo čvorova JavaScript može da koristi sledeća svojstva:

- parentNode;
- childNodes[nodenumber];
- firstChild;
- lastChild;
- nextSibling i
- previousSibling

U navedenom primeru, element <*title*> ne sadrži tekst, već title ima čvor dete koje je tipa text koji ima vrednost "This is JavaScript HTML DOM".

```
<title id="test">This is JavaScript HTML DOM</title>
```

Vrednost tekstualnog čvora može da se pročita ili sa *innerHTML* ili sa *nodeValue* svojstvima. Ispod su navedeni primeri, gde je rezultat isti.

```
var title=document.getElementById("test").innerHTML;
var title=document.getElementById("test").firstChild.nodevalue;
var title=document.getElementById("test").childNode[0].nodeValue;
```

Jednostavni primeri sa DOM navigacijom su prikazani na listinzima 124-126.

Listing 127 - Primer sa n*odeName* svojstvom

```
<html>
<body>
  <p id="test"> This is just a test paragraph</p>
  <p id="p1"> </p>
  <p id="p2"> </p>
  <p id="p3"> </p>
  <p id="p4"> </p>
  <script>
    document.getElementById("p1").innerHTML =
      "The name of the node is: " +
      document.getElementById("test").nodeName;
    document.getElementById("p2").innerHTML =
      "The type of the node is: " +
      document.getElementById("test").nodeType;
    document.getElementById("p3").innerHTML =
      "The value of thenode is: " +
      document.getElementById("test").nodeValue;
    document.getElementById("p4").innerHTML =
      "The value of the child node is: " +
      document.getElementById("test").firstChild.nodeValue;
  </script>
</body>
</html>
```

Svojstvo nodeName definiše ime čvora i njegove karakteristike su:

- *nodeName* je read-only
- *nodeName* čvora je isto kao i naziv odgovarajućeg HTML taga
- *nodeName* atributa je naziv atributa odgovrajućeg HTML taga (elementa)
- *nodeName* tekstualnog čvora je uvek #text
- *nodeName* čvora document je uvek #document
- svojstvo *nodeValue* definiše vrednost čvora:
- *nodeValue* elementa (HTML taga) je nedodređeno (engl. *Undefined*)
- *nodeValue* tekstualnog čvora je tekst
- *nodeValue* atributa je vrednost atributa.
- svojstvo *nodeType* je read-only i ono definiše tip određenog čvora.
- tip čvora *HTML* elementa (taga) je 1, atributa je 2, teksta je 3.

Primer sa *nodeName* svojstvom naveden je prikazan na listingu 127.

Listing 128 - Primer sa *appendChild* metodom

```
<html>
<body>
  <div id="div1"></div>
  <script>
    var paragraph = document.createElement("p");
    var text = document.createTextNode(
                "This is added paragraph's text node"
              );
    paragraph.appendChild(text);
    document.getElementById("div1").appendChild(paragraph);
  </script>
</body>
</html>
```

Listing 129 - Primer sa *insertBefore* metodom

```
<html>
<body>
  <div id="div1">
    <p id="p1">This is a paragraph.</p>
    <p id="p2">This is another paragraph.</p>
  </div>
  <script>
    var para = document.createElement("p");
    var node = document.createTextNode("This is new.");
    para.appendChild(node);
    var element = document.getElementById("div1");
    var child = document.getElementById("p1");
    element.insertBefore(para,child);
  </script>
</body>
</html>
```

Da bi se dodao novi HTML element, prvo element mora da se kreira, a zatim mora da se doda na HTML DOM stablo u okviru postojećeg elementa. Metoda *appendChild()* dodaje element kao poslednje dete, kao što je prikazano u listingu 128.

Listing 130 - Brisanje HTML elementa pomoću imena roditelja

```
<html>
<body>
  <div id="div1">
    <p id="p1">This is paragraph to be removed</p>
    <p id="p2">This paragraph will not be removed</p>
  </div>
  <script>
    var parent=document.getElementById("div1");
    var childRemove=document.getElementById("p1");
    parent.removeChild(childRemove);
    // moze i ovako
    // var child=document.getElementById("p1);
    // child.parentNode.removeChild(child);
  </script>
</body>
</html>
```

Listing 131 - Primer *replaceChild()* metode

```
<html>
<body>
  <div id="div1">
    <p id="p1">This is a paragraph.</p>
    <p id="p2">This is another paragraph.</p>
  </div>
  <script>
    var parent = document.getElementById("div1");
    var child = document.getElementById("p1");
    var para = document.createElement("p");
    var node = document.createTextNode("This is new.");
    para.appendChild(node);
    parent.replaceChild(para,child);
  </script>
</body>
</html>
```

Metod *insertBefore()* ubacuje element pre određenog elementa u postojeći element, listing 129.

U slučaju da je potrebno da se obriše određeni HTML element, prvo mora da se sazna ime roditelja tog elementa, kao što je navedeno na listingu 130.

Za zamenu elementa koristi se metoda *replaceChild()*, listing 131.

11.2 JAVASCRIPT BOM

Objektni model pretraživača (engl. *browser object bodel - BOM*) omogućava JavaScript kodu da komunicira sa pretraživačem. U vreme pisanja ovog udžbenika nije definisan zvaničan BOM standard. Obzirom da savremeni veb pretraživači koriste skoro iste metoda i svojstva za komuniciranje sa JavaScript-om, obično se ove metode i svojstva nazivaju metode i svojsta BOM-a.

U nastavku su prvo prikazane uvodne teme iz domena JavaScript BOM, dok su na kraju ovog dela udžbenika prikazane naprednije teme iz istog domena.

11.2.1 JavaScript BOM uvodne teme

Objekat window podržavaju svi pretrežavači. Ovaj objekat označava prozor pretraživača. Svi globalni JavaScript objekti, funkcije i promenljive po default-u postaju članovi window objekta. Globalne promenljive su svojstva window objekta, dok su globalne funkcije metode ovog objekta. Čak je i document objekat HTML DOM-a svojstvo window objekta. Tako na primer, kod:

```
window.document.getElementById("header");
```

Ima istu funkciju kao i kod:

```
document.getElementById("header");
```

Da bi se prikazala vrednost veličine prozora pretraživača, na raspolaganju su dve metode, koje obe vraćaju vrednost veličine prozora koja je izražena i pikselima (engl. *pixels*), i to:

- *window.innerHeight* - prikazuje unutrašnju visinu prozora pretraživača i
- *window.innerWidth* - prikazuje unutrašnju širinu prozora pretraživača.

Napomena: veličina prozora pretraživača ne uzima u obzir dimenzije scrollbar-a i toolbar-a. Osim toga, dve navedene metode nisu podržane od strane svih pretraživača.

Listing 132 - Prikazivanje veličine prozora pretraživača

```
<html>
<body>
  <p id="test"></p>
  <script>
    /* koristimo univerzalno resenje
       za sve browser-e pomocu ili operatora */
    var w = window.innerWidth ||
            document.documentElement.clientWidth ||
            document.body.clientWidth;
    /* koristimo univerzalno resenje
       za sve browser-e pomocu ili operatora */
    var h = window.innerHeight ||
            document.documentElement.clientHeight ||
            document.body.clientHeight;
    var x = document.getElementById("test");
    x.innerHTML =
      "Browser inner window width: " +
      w + ", height: " + h + ".";
  </script>
</body>
</html>
```

Na listingu 132 naveden je praktičan primer JavaScript koda koji omogućava prikazivanje veličine prozora na svim pretraživačima.

Kao još neke metode window objekta navode se:

- *window.open()* - otvara novi prozor;
- *window.close()* - zatvara otvoreni prozor;
- *window.moveTo()* - pomera prozor i
- *window.resizeTo()* - menja veličinu prozora.

Preko objekta *window.screen* može da se pristupi informacijama o ekranu korisnika. Ovom objektu može da se pristupi i drektno (neposredno), bez referisanja *window*.

U neka od šire korišćenih svojstava ovog objekta ubrajaju se sledeća:

- *screen.width*;
- *screen.height*;
- *screen.availWidth*;
- *screen.availHeight*;
- *screen.colorDepth* i
- *screen.pixelDepth*.

Listing 133 - Upotreba svojstava *window.screen.width* i *window.screen.height*

```
<html>
<body>
  <p id="sirina"> </p>
  <p id="visina"> </p>
  <script>
    document.getElementById("sirina").innerHTML =
      "Sirina ekrana je: " + window.screen.width;
    document.getElementById("visina").innerHTML =
      "Visina ekrana je: " + window.screen.height;
  </script>
</body>
</html>
```

Preko svojstava *screen.width* i *screen.height* ispisuju se vrednosti za širinu i visinu ekrana korisnika u pikselima, kao što je navedeno u listingu 133.

Preko svojstava *screen.availWidth* i *screen.availHeight*, listing 134, čitaju se vrednosti za korisnu širinu i visinu ekrana korisnika u pikselima, koja se dobija kada se od ukupne širine i visine oduzme veličina interfejsa, kao što je na primer veličina Windows taskbar-a.

Listing 134 - Svojstava *window.screen.availWidth* i *window.screen.availHeight*

```
<html>
<body>
  <p id="sirina"> </p>
  <p id="visina"> </p>
  <script>
    document.getElementById("sirina").innerHTML =
      "Raspoloziva sirina ekrana je: " + window.screen.availWidth;
    document.getElementById("visina").innerHTML =
      "Raspoloziva visina ekrana je: " + window.screen.availHeight;
  </script>
</body>
</html>
```

Svojstvo *screen.colorDepth* prikazuje koliko se bitova koristi za prikazivanje jedne boje. Savremeniji računari koriste 24bit ili 32bit hardver za rezoluciju boje, gde je:

- 24 bitova = 16, 777, 216 (*True Colors*) i
- 32 bitova = 4,294,967,296 different (*Deep Colors*)

Stariji računari koristili su 16 bitova za prikazivanje boja, gde je postojalo 65, 536 različitih boja (*High Colors*). Jako stari računari su koristili samo 8bit-ova za prikazivanje boja, što je omogućavalo samo 256 različitih boja (*VGA Colors*).

Svojstvo *screen.pixelDepth* prikazuje dubinu ekrana u pikselima.

Listing 135 - Svojstva *window.screen.colorDepth* i *window.screen.pixel Depth*

```
<html>
<body>
  <p id="color"> </p>
  <p id="pixel"> </p>
  <script>
    document.getElementById("color").innerHTML =
      "Color depth ekrana " + window.screen.colorDepth;
    document.getElementById("pixel").innerHTML =
      "Pixel depth ekrana: " + window.screen.pixelDepth;
  </script>
</body>
</html>
```

Objekat *window.location* se koristi za dobijanje informacije o trenutnoj adresi veb strane (URL adresa) i za preusmeravanje pretraživača na novu stranu.

Objektu *window.location*, listinzi 136 i 137, može da se pristupa sa ili bez prefiksa window.Često korišćeni atributi (svojstva) su:

- *window.location.href* - vraća vrednost atributa href trenutne veb strane;
- *window.location.hostname* - vraća ime domena veb host-a;
- *window.location.pathname* - vraća putanju i naziv fajla trenutne strane;
- *window.location.protocol* - vraća korišćeni protokol za pristup, kao što je http ili https i
- *window.location.assign* - učitava novi dokument.

Primeri sa atributima window.location objekta dati su u listinzima.

Listing 136 - Primer sa atributima *window.location* objekta - I

```
<html>
<body>
  <p id = "href"> </p>
  <p id = "hostname"></p>
  <p id = "path"></p>
  <p id = "protocol"></p>
  <p id = "port"></p>
  <script>
    /* ako se skripta pokreće u localhost-u, vrednosti
      za protokol, port i hostname vrednosti biće null */
    document.getElementById("href").innerHTML =
      "Puna putanja do ove strane je " + window.location.href;
    document.getElementById("hostname").innerHTML =
      "Hostname ove stranice je: " + window.location.hostname;
    document.getElementById("path").innerHTML =
      "Pathname ove stranice je: " + window.location.pathname;
    document.getElementById("protocol").innerHTML =
      "Protocol ove stranice je: " + window.location.protocol;
    document.getElementById("port").innerHTML =
      "Port ove stranice je: " + window.location.port;
  </script>
</body>
</html>
```

Listing 137 - Primer sa atributima *window.location* objekta - II

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript</h2>
  <h3>Objekat window.location otvaranje nove strane</h3>
  <input
    type="button"
    value="Ucitaj novi dokument"
    onclick="newDoc()"
  >
  <script>
    function newDoc() {
        window.location.assign("https://www.singidunum.ac.rs")
    }
    </script>
</body>
</html>
```

Objekat *window.history* sadrži istoriju veb pretraživača. Ovaj objekat može da se referiše sa ili bez prefiksa window.Radi zaštite privatnosti korisnika, postoje izvesna ograničenja u prisupu JavaScript-a ovom objektu.

Listing 138 - Primer sa *window.history.back()* i *window.history.forward()* metodama

```
<html>
<body>
  <input type="button" value="Idi nazad" onclick="goBack()"/>
  <input type="button" value="Idi napred" onclick="goForward()"/>
  <script>
    function goBack(){
        window.history.back();
    }
    function goForward(){
        window.history.forward();
    }
  </script>
</body>
</html>
```

Dve osnovne metode *window.history* objekta su:

- *history.back()* - isti efekat kao i kada se pritisne dugme back na pretraživaču i
- *history.forward()* - isti efekat kao i kada se pritisne forward na pretraživaču.

Preko objekta *window.navigator* može da se pristupi informacijama o pretraživaču korisnika. Objektu *window.navigator* može da se pristupi sa ili bez referisanja prefiksa *window*.

Kao neki od primera osnovnih svojstava ovog objekta navode se sledeća:

- *navigator.cookieEnabled*;
- *navigator.appName*;
- *navigator.appCodeName* i
- *navigator.platform*

Atribut *cookieEnabled* vraća vrednost true ako korisnikom pretraživač podržava kolačiće, atribut *appName* prikazuje naziv aplikacije pretraživača korisnika, dok atribut *appCodeName* prikazuje naziv koda pretraživača korisnika, listing 139.

Listing 139 - Primer sa atributima *window.navigator* objekta - I

```
<html>
<body>
  <p id = "cookie"></p>
  <p id = "appName"></p>
  <p id = "appCodeName"></p>
  <script>
    document.getElementById("cookie").innerHTML =
      "Cookie su omoguceni: " + window.navigator.cookieEnabled;
    document.getElementById("appName").innerHTML =
      "App name pretrazivaca: " + window.navigator.appName;
    document.getElementById("appCodeName").innerHTML =
      "App Code Name pretrazivaca: " + window.navigator.appCodeName;
  </script>
</body>
</html>
```

Primeri još nekih svojstava *window.navigator* objekta navedena su ispod:

- atribut *product* prikazuje naziv proizvoda engine-a pretraživača.
- atribut *appVersion* prikazuje informaciju o verziji pretraživača.
- atribut *userAgent* prikazuje *user-agent* header koji pretraživač šalje serveru.
- atribut *platform* prikazuje platformu pretraživača (operativni sistem).
- atribut *language* prikazuje jezik pretraživača.
- atribut *online* prikazuje da li je pretraživač *online* i
- metoda *javaEnabled()* vraća *true* ako pretraživač podržava *Javu*.

Listing 140 - Primer sa atributima *window.navigator* objekta - II

```
<html>
<body>
  <p id="product"></p>
  <p id="appVersion"></p>
  <p id="userAgent"></p>
  <p id="platform"></p>
  <p id="language"></p>
  <p id="online"></p>
  <p id="JavaEnabled"></p>
  <script>
    document.getElementById("product").innerHTML =
      "Proizvod pretrazivaca: " + window.navigator.product;
    document.getElementById("appVersion").innerHTML =
      "App Version pretrazivaca: " + window.navigator.appVersion;
    document.getElementById("userAgent").innerHTML =
      "User agent pretrazivaca: " + window.navigator.userAgent;
    document.getElementById("platform").innerHTML =
      "Platforma pretrazivaca: " + window.navigator.platform;
    document.getElementById("language").innerHTML =
      "Jezik pretrazivaca: " + window.navigator.language;
    document.getElementById("online").innerHTML =
      "Da li je pretrazivac online: " + window.navigator.online;
    document.getElementById("JavaEnabled").innerHTML =
      "Da li pretrazivac podrzava Javu: " +
      window.navigator.javaEnabled();
  </script>
</body>
</html>
```

Kao napomena navodi se da informacije prikupljene preko *window.navigator* objekta često mogu biti pogrešne i kao takve ne treba da se koriste za detekciju verzije pretraživača zbog toga što:

- različiti pretraživači mogu da imaju isto ime;
- navigator podaci mogu da budu promenljeni od strane korisnika;
- neki pretraživači namerno sebe identifikuju pogrešno i
- pretraživači ne prepoznaju nove operativne sisteme koji su izašli na tržište nakon pretraživača.

Listing 141 - Primer sa *window.alert()*

```
<html>
<body>
  <h2>JavaScript Alert box</h2>
  <button onclick="myFunction()">Probaj alert box</button>
  <script>
    function myFunction() {
        alert("Dobrodošli. Ovo je alert box!");
    }
  </script>
</body>
</html>
```

JavaScript podržava tri vrste *popup* prozora: *Alert box, Confirm box* i *Prompt box*. *Alert box* koristi se kada je potrebno biti siguran da je korisnik pročitao neko obaveštenje, dobio neku informaciju, itd. Kada se *alert* prozor pojavi, korisnik mora da klikne na dugme *OK* kako bi nastavio sa radom, listing 141. *Alert* može da se piše sa ili bez prefiksa *window*. Sintaksa:

```
window.alert("tekst");
```

Confirm box se najčešće koristi kada korisnik treba nešto da proveri, ili da potvrdi. Kada se *confirm box* pojavi, korisnik treba da pritisne ili *OK* ili *Cancel* kako bi nastavio. Ako korisnik pritisne *OK, confirm box* vraća vrednost *true*. Ako korisnik pritisne dugme *Cancel, confirm box* vraća vrednost *false*, listing 142. Objektu *window.confirm* može da se pristupi sa ili bez prefiksa *window*. Sintaksa:

```
window.confirm("sometext");
```

Listing 142 - Primer sa *window.confirm()*

```
<html>
<body>
  <h2>JavaScript Confirm Box</h2>
  <button type="button" onclick="myFunction()">
    Pritisni dugme
  </button>
  <p id="test"></p>
  <script>
    function myFunction() {
      var txt;
      if (confirm("Pritisni dugme")) {
        txt = "Pritisnuli ste OK";
      } else {
        txt = "Pritisnuli ste Cancel";
      }
      document.getElementById("test").innerHTML = txt;
    }
  </script>
</body>
</html>
```

Listing 143 - Primer sa *window.prompt()*

```
<html>
<body>
  <h2>JavaScript Prompt</h2>
  <button onclick="myFunction()">Testiraj prompt box</button>
  <p id="test"></p>
  <script>
  function myFunction() {
    var txt;
    var person = prompt(
                    "Molim Vas, upisite Vase ime:",
                    "John Doe"
                  );
    if (person == null || person == "") {
      txt = "Korisnik je pritisnuo dugme Cancel";
     } else {
      txt = "Zdravo " + person + "! Sta Vi radite danas?";
    }
    document.getElementById("test").innerHTML = txt;
  }
  </script>
</body>
</html>
```

Prompt box se najčešće koristi ako hoćemo da korisnik upiše neku vrednost pre nego što pristupi veb strani. Kada se pojavi *prompt box*, korisnik mora ili da pritisne *OK* ili *Cancel* kako bi nastavio nakon što je upisao neku vrednost. Ako korisnik pritisne *OK, prompt box* vraća vrednost *true*. Ako korisnik pritisne *Cancel*, *prompt box* vraća vrednost *fals*e, listing 143. Sintaksa:

```
window.prompt("sometext", "defaultText");
```

Objektu *window.prompt* može da se pristupi sa ili bez prefiksa *window*.

11.2.2 JavaScript BOM naprednije teme

JavaScript kod može da se izvršava u određenim vremenskim intervalima. Ovakvo izvršavanje kontroliše se tzv. tajmining događajima (engl. *timing events)*. Objekat window omogućava ovakvo izvršavanje.

Dve metode koje JavaScript koristi za timing događaje su:

- *setTimeout(function, milliseconds*) - izvršava funkciju nakon što prođe određeni broj milisekundi i
- *setInterval(function, milliseconds)* - slično kao *setTimeout()*, ali se izvršavanje funkcije ponavlja neprekidno.

Metoda *window.setTimeout()* može da se koristi sa ili bez prefiksa window.Prvi parametar je funkcija koja se izvršava. Drugi parametar određuje broj milisekundi koji treba da prođe pre izvršavanja funkcije. Sintaksa:

```
window.setTimeout(function, milliseconds);
```

U listingu 144, kada korisnik klikne na dugme, nakon tri sekundi izlazi *alert box*.

Metoda *clearTimeout()* prekida izvršavanje funkcije koja je definisana u *setTimeout()* metodi. Sintaksa:

```
window.clearTimeout(timeoutVariable)
```

Metoda *window.clearTimeout()* može da se piše sa ili bez prefiksa *window*.

Listing 144 - Primer sa tajming događajima - I

```
<html>
<body>
  <p>Nakon 3 sekunde izaci ce alert box sa tekstom "Zdravo".</p>
  <button onclick="setTimeout(myFunction, 3000);">Klikni me</button>
  <script>
    function myFunction() {
        alert('Zdravo!!!!');
    }
  </script>
</body>
</html>
```

Listing 145 - Primer sa tajming događajima - II

```
<html>
<body>
  <p>Klikni na dugme probaj i nakon tri sekunde izlazi popup.</p>
  <p>
    Klikni dugme Stop pre nego sto proteknu tri seknude,
    kako bi sprecili izlazak popup prozora
  </p>
  <button onclick="myVar = setTimeout(myFunction, 3000)">
    Probaj
  </button>
  <button onclick="clearTimeout(myVar)">Stop</button>
  <script>
    function myFunction() {
        alert("Zdravo!");
    }
  </script>
</body>
</html>
```

Metoda *clearTimeout()* koristi promenljivu koju vraća metoda *setTimeout()*. Sitnaksa:

```
var myVar = setTimeout(function, milliseconds);
clearTimeout(myVar);
```

U primeru na listingu 145, nakon 3 sekunde izlazi alert box od trenutka kada kliknemo na dugme *Probaj*. Ako pre isteka tri sekunde kliknemo na dugme *Stop*, popup neće da izađe.

Metoda *setInterval()* ponavlja izvršavanje funkcije koja je prosleđena kao argument u određenim vremenskim intervalima. Sintaksa:

```
window.setInterval(function, milliseconds);
```

Metoda *window.setInterval()* može da se koristi sa ili bez prefiksa *window*.Prvi parametar ove metode je funkcija koju je potrebno izvršiti. Drugi parametar je dužina vremenskog intervala između svakog izvršavanja funkcije. Metoda *clearInterval()* može da se koristi kako bi se izvršavanje funkcije zaustavilo, slično kao i u slučaju metode *setTimeout()*.

U primeru na listingu 146 prikazuje se trenutno vreme u intervalu od jedne sekunde, slično kao na digitalnom satu. Prikazivanje vremena se prekida klikom na dugme *Stop*.

Listing 146 - Primer sa tajming događajima - III

```
<html>
<body>
  <p>
    Skripta koja prikazuje sat sa trenutnim
    vremenom sa preciznoscu od jedne sekunde
  </p>
  <p id="test"></p>
  <button onclick="clearInterval(myVar)">
    Zaustavi sat
  </button>
  <script>
    var myVar = setInterval(myTimer ,1000);
    function myTimer() {
      var d = new Date();
      document.getElementById("test").innerHTML =
        d.toLocaleTimeString();
    }
  </script>
</body>
</html>
```

Kao što je već i navedeno u udžbeniku, kolačići (engl. *cookies*) omogućavaju čuvanje informacija o korisniku na veb stranicama. Kolačići su podaci koji se čuvaju u malim tekstualnim fajlovima na računaru korisnika. Obzirom da je HTTP protokol protokol koji ne pampti stanja, kolačići se koriste za čuvanje informacija.

Kada na primer korisnik poseti neku veb stranu, informacije o njegovom korisničkom imenu se čuvaju u kolačić fajlu. Kada sledeći put korisnik poseti tu istu veb stranu, kolačić je zapamtio njegovo ime i na osnovu toga korisnik može da vidi personalizovan veb sajt. Kolačići se čuvaju u formi ime-vrednost, na primer *korisnicko_ime*="*SingiUser*".

Za kreiranje, ažuriranje i brisanje kolačića, JavaScript koristi *document. cookie metodu()*. Sintaksa:

```
document.cookie = "username=John Doe";
```

Uz kreiranje kolačića, može da se doda i vreme isteka kolačića (engl. *expiry date*). Vreme se zadaje u UTC (engl. *Universal Coordinated Time*) formatu. Sintaksa:

```
document.cookie = "username=John Doe; expires=Wed, 16 May 2018
12:00:00 UTC";
```

Takođe, u *document.cookie* može da se prosledi i treći parametar putanja (engl. *path*). Ovim parametrom može da se definiše kom delu veb sajta (kojoj stranici) kolačić pripada. Po default-u, kolačić pripada trenutnoj stranici. Sintaksa:

```
document.cookie = "username=John Doe; expires=Wed, 16 May
2018 12:00:00 UTC; path=/";
```

Vrednost kolačića se čita i smešta se u promenljivu. Sintaksa:

```
var cookie = document.cookie;
```

Na ovaj način se vrednosti svih kolačića smeštaju u string promenljivu.

Vrednost kolačića se ažurira tako što se zada isti name, ali nova vrednost. Sintaksa:

```
document.cookie = "username=Pera Peric; expires=Wed, 16 May
2018 12:00:00 UTC; path=/";
```

Na ovaj način kolačić username je prepisan novom vrednošću.

Atribut *document.cookie* izgleda kao normalni tekstualni string, ali on to nije. Čak iako se zapiše ceo cookie string u *document.cookie* atribut, kada se vrednost tog atributa pročita, moguće je videti samo *name-value* parove ovog atributa.

Kada se dodaju novi kolačići, stari kolačići se ne prepisuju. Nakon dodavanja novih kolačića, kada se pročita vrednost *document.cookie* atributa, dobijaju se rezultati slični ovome:

```
cookie1=vrednost;cookie2=vrednost;
```

Primer na listingu 147 poslužiće kao demostracija metoda za rad sa kolačićima. Primer pamti informacije o posetiocu veb sajta. Kada prvi put posetilac poseti veb stranu, posetilac treba da upiše svoje ime, koje se čuva u kolačiću. Sledeći put kada posetilac poseti istu veb stranu, posetilac dobija svoju pozdravnu poruku. Dakle, u primeru su korišćene tri JavaScript funkcije:

- funkcija za kreiranje vrednosti kolačića
- funkcija za čitanje vrednosti kolačića i
- funkcija za proveru vrednosti kolačića.

11.3 JAVASCRIPT AJAX

U prvom delu ove sekcije prikazani su osnovni funkcionalni principi AJAX tehnologije, dok su u drugom delu prikazane naprednije teme iz domena AJAX-a.

11.3.1 Osnovne teme

AJAX (**A**synchronous **J**avaScript **A**nd **X**ML) je tehnologija koja se često koristi u kreiranju dinamičkog sadržaja. AJAX omogućava sledeće:

Listing 147 - Primer sa kolačićima

```
<html>
<head>
<script>
  function setCookie(cname,cvalue,exdays) {
    var d = new Date();
    d.setTime(d.getTime() + (exdays*24*60*60*1000));
    var expires = "expires=" + d.toGMTString();
    document.cookie =
      cname + "=" + cvalue + ";" +
      expires + ";path=/";
  }
  function getCookie(cname) {
    var name = cname + "=";
    var decodedCookie = decodeURIComponent(document.cookie);
    var ca = decodedCookie.split(';');
    for(var i = 0; i < ca.length; i++) {
      var c = ca[i];
      while (c.charAt(0) == ' ') {
        c = c.substring(1);
      }
      if (c.indexOf(name) == 0) {
        return c.substring(name.length, c.length);
      }
    }
    return "";
  }
  function checkCookie() {
```

```
28      var user=getCookie("username");
29      if (user != "") {
30        alert("Dobrodosli ponovo " + user);
31      } else {
32        user = prompt("Molim Vas, unesite Vase ime:","");
33        if (user != "" && user != null) {
34          setCookie("username", user, 30);
35        }
36      }
37    }
38 </script>
39 </head>
40 <body onload="checkCookie()">
41 </body>
42 </html>
```

- ažuriranje veb strane bez ponovnog učitavanja;
- slanje HTTP request ka veb serveru nakon što je veb strana učitana;
- primanje HTTP response od servera nakon što je veb strana učitana
- slanje podataka veb serveru u pozadini.

Često se na forumima, ali i u stručnoj literaturi može naći konstantacija da je AJAX tehnologija programski jezik. Međutim, AJAX nije programski jezik.

AJAX predstavlja kombinaciju sledećih elemenata:

- ugrađenog XMLHttpRequest objekta u pretraživač klijenta koji služi za slanje HTTP zahteva (engl. *HTTP request*) za određenom veb stranom na veb serveru i
- JavaScript i HTML DOM koji omogućavaju prikaz i korišćenje podataka koje je server poslao.

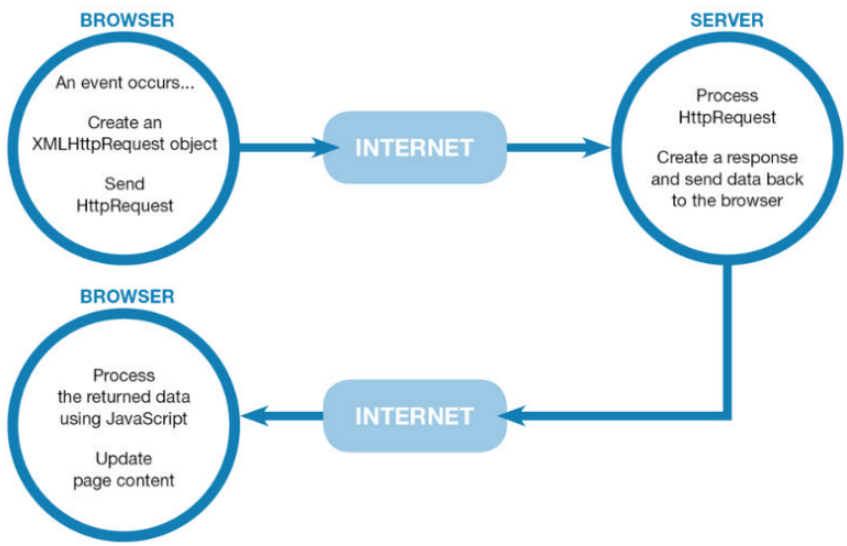
Sam akronim AJAX navodi na "pogrešan put". U primeni AJAX tehnologije može da se koristi XML (eXtensible Markup Language) za transfer podataka, ali se osim XML-a za transfer podataka koristi i slanje podatka u čistoj tekstualnoj formi (engl. *plain text*), kao i JSON (JavaScript Object Notation) format.

AJAX omogućava ažuriranje veb strana asinhrono razmenom podataka sa veb serverom u pozadini (engl. *background*), na način da korisnik to ne primećuje. Dakle, moguće je da se ažuriraju samo delovi veb strane bez učitavanja cele strane.

Izvršavanje AJAX-a se sastoji iz sedam koraka koji su prikazani na Slici. Ovi koraci su ukratko opisani u nastavku.

- Korak 1: događaj se dogodi na veb stranici (stranica je učitana, korisnik je pritisnuo dugme, itd.).
- Korak 2: kao reakciju na događaj, JavaScript kreira *XMLHttpRequest* objekat.
- Korak 3: *XMLHttpRequest* objekat šalje zahtev veb serveru.
- Korak 4: server obrađuje zahtev.
- Korak 5: server šalje odgovor veb strani.
- Korak 6: odgovor primljen od servera čita JavaScript kod.
- Korak 7: JavaScript izvršava odgovarajuću akciju, kao što je na primer ažuriranje veb strane novim podacima dobijenih od servera.

Slika 43 - Faze izvršavanja AJAX koda



"Srce" AJAX-a je *XMLHttpRequest* objekat. Svi savremeniji Internet pretraživači imaju podršku za *XMLHttpRequest* objekat. Ovaj objekat se koristi za razmenu podataka između klijenta i veb servera u pozadini, što znači da je moguće da se ažurira deo veb strane, bez ponovnog učitavanja cele strane.

Svi savremeni pretraživači poput *Microsoft Edge, Google Chrome, Firefox, Safari Opera, Internet Explorer 7+*, itd. imaju ugrađenu podršku za *XMLHttpRequest* objekat. Sintaksa za kreiranje *XMLHttpRequest* objekta:

```
variable = new XMLHttpRequest();
```

Iz bezbednosnih razloga, savremeni veb pretraživači ne dozvoljaju AJAX tehnologiji da ima pristup između različitih veb domena. Dakle, i veb strana i XML fajl koji treba da se učita moraju da budu na istom veb serveru.

Potrebno je da se napomene da starije verzije *Internet Explorer* pretraživača (IE5 i IE6) koriste *ActiveX* objekat umesto *XMLHttpRequest* objekta. Sintaksa za kriranje *ActiveX* objekta:

```
variable = new ActiveXObject("Microsoft. XMLHTTP");
```

Metode *XMLHttpRequest* objekta date su u tabeli 14, dok su atributi *XMLHttpRequest* metoda date su u tabeli 15.

Tabela 14 - Metode *XMLHttpRequest* objekta

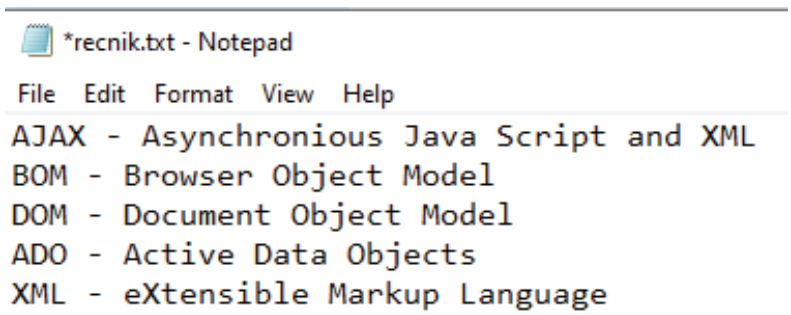
METODA	OPIS
new XMLHttpRequest()	Kreira novi XMLHttpRequest objekat
abort()	Prekida zahtev u toku
getAllResponseHeaders()	Vraća sve info iz response zaglavlja
getResponseHeader()	Vraća određenu info iz response zaglavlja
open(*method, url, async, user, psw*)	method: ili GET ili POST url: lokacija resursa async: true (asinhrono) or false (sinhrono) user: opcionalno korisničko ime psw: opcionalno lozinka
send()	Šalje zahtev serveru, koristi se uz GET
send(*string*)	Šalje zahtev serveru, koristi se uz POST
setRequestHeader()	Dodaje par label/vrednost u zaglavlje koje se šalje

Početni primer sa AJAX tehnologijom može da se demonstrira korišćenjem sledećeg hipotetičkog scenarija: Neka je u root-u veb servera dat fajl recnik.txt koji sadrži definicije nekih pojmova iz oblasti računarstva. Klikom na dugme pročitaj, fajl se čita i njegov sadržaj se prikazuje na veb strani. Sadržaj fajla recnik.txt dat je na 44. Da bi primer radio, potrebno je da fajl recnik.txt bude u istom folderu u okviru veb servera gde se nalazi i glavni kod.

Tabela 15 - Atributi *XMLHttpRequest* objekta

ATRIBUT	OPIS
onreadystatechange	Definiše funkciju koja će biti pozvana kada se readyState atribut promeni
readyState	Čuva statu XMLHttpRequest objekta 0: zahtev nije inicijalizovan 1: uspostavljena konekcija sa serverom 2: zahtev je primljen 3: zahtev se obrađuje 4: zahtev je obrađen i odgovor je spreman
responseText	Vraća odgovor kao string
responseXML	Vraća odgovor kao XML
status	Vraća status odgovora 200: OK 403: Zabranjeno 404: Strana nije pronađena
statusText	Vraća tekst za status (OK, Not found)

Slika 44 - Sadržaj fajla *rečnik.txt*



Kao što je već i navedeno, objekat *XMLHttpRequest* služi za razmenu podataka sa veb serverom. Za slanje zahteva veb serveru, koriste se *open()* i *send()* metode *XMLHttpRequest* objekta (sintaksu ovih metoda pogledati u tabeli):

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Listing 148 - Čitanje sadržaja fajla pomoću AJAX tehnologije

```
<!DOCTYPE html>
<html>
<body>
  <h2>Prvi AJAX primer</h2>
  <p id="test">
    AJAX ce da promeni tekst koji se nalazi u ovom paragrafu
  </p>
  <button type="button" onclick="loadDoc()">Procitaj</button>
  <script>
  function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("test").innerHTML =
          this.responseText;
      }
    };
    xhttp.open("GET", "recnik.txt", true);
    xhttp.send();
  }
  </script>
</body>
</html>
```

Listing 149 - Modifikovani primer sa čitanjem sadržaja fajla pomoću AJAX tehnologije

```
<!DOCTYPE html>
<html>
<body>
  <h2>Prvi AJAX primer</h2>
  <p id="test">
    AJAX ce da promeni tekst koji se nalazi u ovom paragrafu
  </p>
  <button type="button" onclick="loadDoc()">Procitaj</button>
  <script>
  function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("test").innerHTML =
          this.responseText;
      }
    };
    xhttp.open("GET", "recnik.txt?t=" + Math.random(), true);
    xhttp.send();
  }
  </script>
</body>
</html>
```

Listing 150 - Primer slanja podataka serveru pomoću GET metode

```
<!DOCTYPE html>
<html>
<body>
<h2>Prvi AJAX primer</h2>
<p id="test">
  AJAX ce da promeni tekst koji se nalazi u ovom paragrafu
</p>
<button type="button" onclick="loadDoc()">Procitaj</button>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("test").innerHTML =
        this.responseText;
    }
  };
  xhttp.open(
    "GET",
    "JSAJAX3.php?name=pera&surname=peric",
    true
  );
  xhttp.send();
}
</script>
</body>
</html>
```

Listing 151 - Kod fajla *JSAJAX3.php*

```
<?php
$name=$_GET['name'];
$surname=$_GET['surname'];
echo "Hello " . $name . " " . $surname;
?>
```

Često se postavlja pitanje da li da se koristi GET ili POST metoda. GET je jednostavnija i brža od POST i koristi se u većini slučajeva. Međutim, korišćenje POST metode je poželjno u sledećim situacijama:

- kada je potrebno da se izbegne keširanje fajlova;
- kada je potrebno da se pošalje velika količina podataka serveru, jer POST metoda nema ograničenja po pitanju veličine zahteva i
- kada se šalju podaci koje je uneo korisnik koji potencijalno sadrže nepoznate karaktere (POST je robusniji i sigurniji od GET).

Kao što je navedeno, korišćenjem GET metode mogu da se generišu keširani rezultati. Da bi se izbeli keširani rezultati korišćenjem GET metode, može da se dodamo jedinstveni ID u URL adresu. Modifikovani primer sa čitanjem podataka iz fajla *rečnik.txt* prikazan je u listingu 149.

Listing 152 - Primer slanja podataka serveru pomoću POST metode

```
<!DOCTYPE html>
<html>
<body>
<h2>Prvi AJAX primer</h2>
<p id="test">
  AJAX ce da promeni tekst koji se nalazi u ovom paragrafu
</p>
<button type="button" onclick="loadDoc()">Procitaj</button>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("test").innerHTML =
        this.responseText;
    }
  };
  xhttp.open("POST", "JSAJAX4.php", true);
  // bez naredne linije POST nece da radi!!!!
  xhttp.setRequestHeader(
    "Content-type",
    "application/x-www-form-urlencoded"
```

```
23    );
24    xhttp.send("name=pera&surname=peric");
25 }
26 </script>
27 </body>
28 </html>
```

Pomoću URL-a koji se naovid u GET metodi mogu da pošalju podaci serveru. Na primeru koji je prikazan ispod vrši se komunikacija sa PHP fajlom koji čita podatke koji su poslati pomoću GET metode. Kada se klikne na dugme koje je dato u kodu na listingu 150, poziva se PHP skripta JSAJAX3.php, koja prikazuje promenljive preko globalnog niza $_GET.

Da bi se koristila metoda POST za slanje podataka HTML forme, potrebno je dodati HTTP zaglavlje (engl. *header*) pomoću metode *setRequestHeader()*. Podaci koji se šalju POST-om dodaju se u *send()* metodu. Sintaksa:

```
setRequestHeader(header, value)
```

U navedenoj sintaksi, header predstavlja ime zaglavlja, dok vrednost predstavlja njegovu vrednost. U primeru koji je prikazan u listingu 152, prikazno je korišćenje metode POST za slanje informacija. Kod fajla JSAJAX4.php isti je kao i malopre prikazan kod fajla JSAJAX3.php.

U nastavku su ukratko opisani najznačajniji delovi koda prikazani u Listingu 154 i 155.

Parametar URL metode *open()* je adresa fajla na veb serveru:

```
xhttp.open("GET", "ajax_test. asp", true);
```

Ovaj fajl može da bude bilo kog tipa, kao što je *.txt, .php, .xml, . asp*. Važno je da fajl može da uradi određenu akciju, pre nego što server pošalje odgovor nazad klijentu.

Zahtevi ka serveru (engl. *requests*) bi trebalo da se asinhrono šalju, kako bi se iskoristile prednosti AJAX tehnologije. Parametar async metode *open()* treba da se postavi na vrednost true.

```
xhttp.open("GET", "ajax_test.asp", true);
```

Kada se zahtevi šalju asinhrono, JavaScript ne mora da čeka odgovor servera, već u međuvremenu može da radi nešto drugo, kao na primer da izvršava druge skripte i da obrađuje odgovor servera kada odgovor bude bio spreman.

Atribut *readyState* čuva status *XMLHttpRequest* objekta. Atribut *onreadystatechange* definiše funkciju koja će biti izvršena kada se vrednost *readyState* atributa promeni. Atributi *status* i *statusText* čuvaju informacije o statusu *XMLHttpRequest* objekta. U tabeli 16 opisani su navedeni atributi.

Tabela 16 - Atributi *XMLHttpRequest* objekta

METODA	OPIS
onreadystatechange	Definiše funkciju koja će da se izvrši kada se vrednost atributa readyState promeni
readyState	Čuva informaciju o statusu XMLHttpRequest objekta 0: zahtev nije inicijalizovan 1: konekcija sa serverom je uspostavljena 2: zahtev je primljen 3: zahtev se obrađuje 4: zahtev je obrađen i odgovor je spreman
Status	200: "OK" 403: "zabranjeno" 404: "stranica nije pronađena"
statusText	vraća opis statusa, kao na primer "OK" ili "Not Found"

11.3.2 Naprednije teme

Korišćenjem AJAX tehnologije, vrlo jednostavno mogu da se pročitaju informacije iz XML fajla. Da bi bilo demonstrirano čitanje informacija iz XML fajla, biće korišćen pojednostavljeni XML fajl *carCatalog.xml*, slika 45, gde se nalazi lista raspoloživih automobila za prodaju na imaginarnom placu novih vozila.

Primer AJAX koda prikazan je na listingu 153. Napominje se da je zbog velikog broja redova prikazan samo isečak XML fajla.

U primeru prikazanom na listingu 153, prvo se čita ceo sadržaj XML fajla *carCatalog.xml.* Zatim se pronalaze svi tagovi <MODEL>. Konačno, prikazuju se svi čvorovi deca ovog taga. Čvorovi deca su u ovom slučaju sam tekst koji stoji između <MODEL> i </MODEL> tagova.

Slika 45 - Struktura fajla *carCatalog.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG>
  <CAR>
    <MAKE>BMW</MAKE>
    <MODEL>320d</MODEL>
    <COUNTRY>GERMANY</COUNTRY>
    <PRICE>35.000</PRICE>
    <YEAR>2018</YEAR>
  </CAR>
<CAR>
    <MAKE>BMW</MAKE>
    <MODEL>520d</MODEL>
    <COUNTRY>GERMANY</COUNTRY>
    <PRICE>65.000</PRICE>
    <YEAR>2018</YEAR>
  </CAR>
<CAR>
    <MAKE>BMW</MAKE>
    <MODEL>760d</MODEL>
    <COUNTRY>GERMANY</COUNTRY>
    <PRICE>135.000</PRICE>
    <YEAR>2018</YEAR>
 </CAR>
```

```
<CAR>
    <MAKE>BMW</MAKE>
    <MODEL>X3 2.0D</MODEL>
    <COUNTRY>GERMANY</COUNTRY>
    <PRICE>55.000</PRICE>
    <YEAR>2018</YEAR>
 </CAR>
<CAR>
    <MAKE>BMW</MAKE>
    <MODEL>X5 3.0D</MODEL>
    <COUNTRY>GERMANY</COUNTRY>
    <PRICE>85.000</PRICE>
    <YEAR>2018</YEAR>
 </CAR>
</CATALOG>
```

Listing 153 - Očitavanje XML fajla pod nazivom *carCatalog.xml* pomoću AJAX-a - I

```
1  <html>
2  <body>
3    <p><b>A list of all available cars</b></p>
4    <button type="button" onclick="load()">
5      Click here to get all car models
6    </button>
7    <p id="demo"></p>
8    <script>
9      function load(){
10       var xhttp;
11       xhttp=new XMLHttpRequest();
12       xhttp.onreadystatechange=function(){
13         var text,xml,x,i;
14         xml=this.responseXML;
15         text="";
16         y=xml.getElementsByTagName("MAKE");
17         x=xml.getElementsByTagName("MODEL");
18         for(i=0;i<x.length;i++){
```

```
            text+=x[i].childNodes[0].nodeValue + "<br/>";
        }
        document.getElementById("demo").innerHTML=text;
      };
      xhttp.open("GET","carCatalog.xml",true);
      xhttp.send();
    }
  </script>
</body>
</html>
```

Naredni primer je malo komplikovaniji. U njemu su prikazani i nazivi proizvođača automobila, zajedno sa modelima u okviru HTML tabele. Ovaj primer dat je u listingu 154.

Korišćenjem AJAX tehnologije takođe može i da se pozove izvršavanje koda u PHP skripti, koji će da pristupi i da pročita određene podatke iz baze podataka.

Listing 154 - Očitavanje XML fajla pod nazivom *carCatalog.xml* pomoću AJAX-a - II

```
<html>
<head>
  <style>
    table,th,td{
        border:1px solid black;
        border-collapse: collapse;
    }
    td,th{
        padding: 5px;
        text-align: center;
    }
  </style>
</head>
<body>
  <p>
    Click on the button to show all available
    cars with vendors and models
```

```
    cars with vendors and models
  </p>
  <button type="button" onclick="load()">
    Show cars
  </button>
  <table id="demo"> </table>
</body>
<script>
  function load(){
    var xhttp=new XMLHttpRequest();
    xhttp.onreadystatechange=function(){
      if(this.readyState==4 & this.status==200){
        myFunction(this);
      }
    };
    xhttp.open("GET","carCatalog.xml",true);
    xhttp.send();
  }
  function myFunction(xhttp){
    var text="";
    text="<tr><th>Make</th><th>Model</th></tr>";
    var xml=xhttp.responseXML;
    var car=xml.getElementsByTagName("CAR");
    var i,value;
    for(i=0;i<car.length;i++) {
      text +=
        "<tr><td>" +
        car[i].getElementsByTagName("MAKE")[0].childNodes[0].nodeValue
        "</td><td>" +
        car[i].getElementsByTagName("MODEL")[0].childNodes[0].nodeValue
        "</td></tr>";
    }
    document.getElementById("demo").innerHTML=text;
  }
</script>
</html>
```

Listing 155 - Primer pozivanja PHP fajla pomoću AJAX-a

```
<html>
<body>
  <form>
    Izaberi zaposlenog:
    <select name="database" onchange="hint(this.value)">
      <option value="">Select employee</option>
      <option value="10001">Prvi</option>
      <option value="10002">Drugi</option>
      <option value="10003">Treci</option>
    </select>
  </form>
  <span id="hint"></span>
  <script>
    function hint(str){
      if(str==""){
        document.getElementById("hint").innerHTML="";
        return;
      }
      var xhttp=new XMLHttpRequest();
      xhttp.onreadystatechange=function(){
      if(this.status==200 && this.readyState==4){
        document.getElementById("hint").innerHTML =
          this.responseText;
      }
      };
      xhttp.open("GET","JSAJAX7.php?q=" + str,true);
      xhttp.send();
    }
  </script>
</body>
</html>
```

U listingu 155 dat je primer takvog koda. Kod php fajla JSAJAX7.php koji se poziva iz JavaScript koda dat je na Listingu.

Korisnik iz dropdown box-a (html select) bira redni zaposlene. Nakon izbora, poziva se PHP skripta koja iz baze *employees* prikazuje podatke za izabranog zaposlenog.

Listing 156 - Kod fajla *JSAJAX7.php*

```
<?php
$servername="localhost";
$username="nebojsa";
$password="sifra";
$db="employees";
$conn=new mysqli($servername,$username,$password,$db);
if($conn->connect_error){
    die("Greska");
}
$q=$_REQUEST['q'];
$hint="";
if($q!="") {
  $sql="SELECT
      first_name,
      last_name,
      gender
    FROM employees
    WHERE
      emp_no='$q'";
  $result=$conn->query($sql);
  if($result->num_rows>0){
    while($row=$result->fetch_assoc()){
      $hint .=
        "Ime: " . $row['first_name'] .
        " Prezime " . $row['last_name'] .
        " Pol: " . $row['gender'] .
        "<br/>";
    }
  } else {
    $hint="nema rezultata";
  }
  $conn->close();
}
echo $hint;
?>
```

VEB APLIKACIJA

12. VEB APLIKACIJA

Veb aplikacija je aplikativni softver koji se nalazi na veb serveru. Za razliku od izvornih (engl. *native*) aplikacija koje mogu da funkcionišu samo na određenom operativnom sistemu ili tipu hardvera za koji su pravljene, veb aplikacije se pokreću iz veb brauzera i ne zavise od operativnog sistema ili tipa uređaja preko kojeg krajnji korisnik pristupa aplikaciji.

Veb aplikacije su postale nezaobilazni deo svakodnevnog, poslovnog i privatnog života. Koristimo ih za pristup elektronskoj pošti, za obavljanje kancelarijskih poslova, za upravljanje poslovnim procesima i izveštavanje, za pristup društvenim mrežama, oglasnicima, medijskim portalima i sl.

12.1 NAČIN RADA

Veb aplikaciji se obično pristupa preko mreže, najčešće interneta ili intraneta, tako što korisnik u veb brauzeru unosi željenu adresu. Računar korisnika šalje HTTP ili HTTPS zahtev preko mreže do veb servera na kojem se nalaze HTTP server, skripting jezik i veb aplikacija.

Veb aplikacija analizira primljeni zahtev, po potrebi preuzima potrebne podatke i dokumenta iz baze podataka ili sa diskova, komunicira sa drugim serverima ili veb servisima. Rezultat svog rada potom isporučuje u željenom obliku, a to mogu biti veb stranica, ili dokumenti različitih tipova (CSS, XML, JSON, slike, video, audio itd.).

12.2 PREDNOSTI I NEDOSTACI VEB APLIKACIJA

U kontekstu razvoja softvera, velika prednost veb aplikacija u odnosu na izvorne aplikacije je kompatibilnost sa različitim operativnim sistemima i hardverskim platformama. Istoj veb aplikaciji korisnik može pristupiti putem desktop ili laptop računara, tablet uređaja ili pametnog telefona, bez obzira da li koristi Windows, Linux, MacOS ili neki drugi operativni sistem.

Implementacija i održavanje veb aplikacija se odvija na veb serveru, a ne na računarima korisnika, što može biti velika prednost u organizacijama i sistemima sa velikim brojem korisnika. Svaka izmena koda veb aplikacije na produkcionom serveru, ili podataka u bazi podataka, odmah postaje dostupna svim korisnicima. Krajnji korisnik nema pristup delu izvornog koda aplikacije koji se izvršava na veb serveru, čime je softversko piratstvo u velikoj meri otežano.

Osim očiglednih prednosti, tehnologija veb aplikacija ima i određene nedostatke. U slučaju prekida Internet veze, aplikacija postaje delimično ili u potpunosti neupotrebljiva, a količina podataka koji se mogu sačuvati na računaru korisnika do ponovnog uspostavljanja veze je minimalna u odnosu na tehničke mogućnosti savremenih računara.

Iako su veb aplikacije inicijalno svoju popularnost stekle u prvoj deceniji XXI veka modelom slobodne distribucije kao softvera otvorenog koda, u poslednjih nekoliko godina sve prisutniji postaje model softvera kao usluge (engl. *Software as a Service, SaaS*) u kome krajnji korisnik može samo da se pretplati na korišćenje određene usluge, ali ne dobija pristup izvornom kodu aplikacije. U situacijama kada se kod aplikacije isporučuje klijentu, sve više se pribegava tehnikama zaštite izvornog koda poput obfuskacije (engl. *obfuscation*), koja podrazumeva transformaciju koda sa ciljem da se u što većoj meri oteža mogućnost njegovog čitanja i razumevanja.

Vlasnik ili operater aplikacije (engl. *Application Service Provider, ASP*) određuju uslove korišćenja, cikluse održavanja i sam životni vek aplikacije. Samim tim, u slučaju delimične ili potpune obustave određene usluge (npr. usled implementacije nove verzije softvera, bankrota kompanije i sl.), krajnji korisnik nema uticaj na ta dešavanja. Krajnji korisnik mora biti svestan i pitanja bezbednosti pre svega poslovnih podataka, zato što vlasnik aplikacije tehničko-tehnološki ima pristup podacima korisnika i može da prati njihove aktivnosti.

12.3 ZAJEDNIČKI ELEMENTI VEB APLIKACIJA

Bez obzira na namenu aplikacije, većina veb aplikacija koristi određene zajedničke elemente koje možemo sistematizovati kao:

- Korisnički interfejs
- Sistem za identifikaciju i autorizaciju korisnika
- Taksonomija
- Pretraga
- Višejezičnost
- Menadžment sadržaja

Slika 46 - Osnovni elementi korisničkog interfejsa veb aplikacije

12.3.1 Korisnički interfejs

Korisnički interfejs (engl. *User Interface, UI*) se sveobuhvatno definiše kao okruženje koje omogućava interakciju između aplikacije i čoveka. Mašina čoveku prikazuje informacije putem ekrana, brojčanika, lampica, na osnovu kojih se od korisnika očekuje dalja akcija. Interakcija se može ostvariti putem dodira (tastatura, taktilni ekran, poluge i sl), glasa (prepoznavanje govora), gestikulacije (prepoznavanje pokreta) i sl.

Korisnički interfejs veb aplikacije je veb stranica koja ima zadatak da prikaže tekst, dokumenta, multimedijalni sadržaj i sl, ali i da primi informaciju o tome šta korisnik želi da učini - prelazak na drugu stranicu, preuzimanje dokumenata, unos podatka itd.

Korisnički interfejs mora biti jednoobrazan u celoj aplikaciji, što podrazumeva istu strukturu i položaj osnovnih elemenata na svim stranicama. Postavljanjem osnovnih gradivnih elemenata veb stranice na pozicije koje su prihvaćene kao industrijski standard, korisnik može da koristi veb aplikaciju bez prethodne obuke.

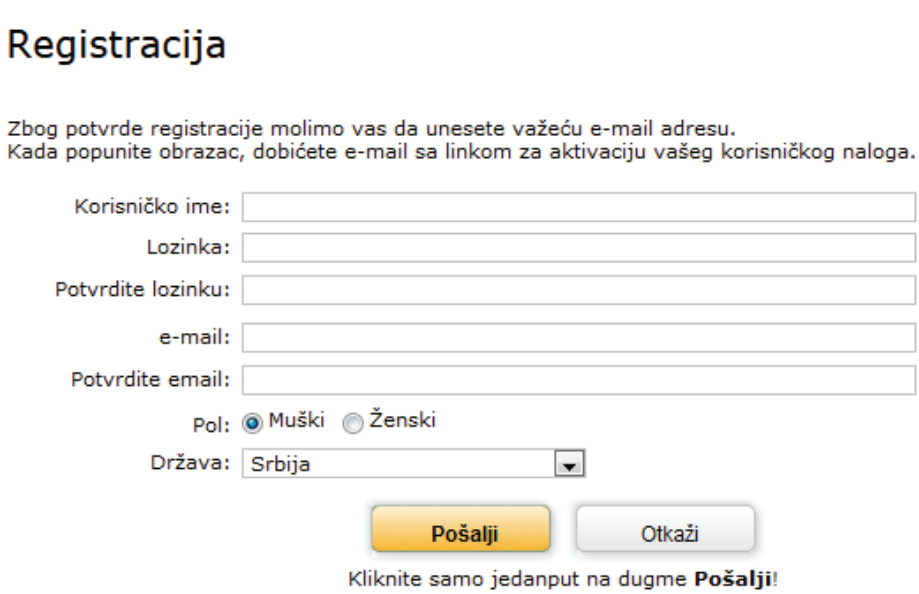
Osnovni gradivni elementi veb stranice su prikazani na slici 46:

- U *zaglavlju* se obično nalaze logo, polje za pretragu, link za prijavu odnosno odjavu korisnika, podmeni za izbor jezika ukoliko je ugrađena višejezička podrška, korpa u veb prodavnicama.
- *Glavni navigacioni meni* se formira kao lenta ispod zaglavlja, sadrži veb linkove, i u svim sekcijama aplikacije treba da ima istu strukturu. Linkovi ka glavnim sekcijama aplikacije su vidljivi, dok se linkovi drugog i viših redova nalaze u odgovarajućim padajućim menijima.
- *Glavni sadržaj* veb stranice sadrži sledeće elemente:
- *Navigacione mrvice* (engl. *breadcrumbs*) se postavljaju iznad naslova stranice, prikazuju putanju kojom je korisnik došao do konkretne stranice.
- Naslov stranice
- *Članak* mogu činiti tekst, jedna ili više tabela sa podacima, grafikoni, izveštaji, slika, video klip, ili bilo koji drugi dokument čiji se sadržaj prikazuje ili preuzima u okviru veb stranice
- *U desnoj koloni* se prikazuje sporedni sadržaj. To mogu biti blokovi sa linkovima ka povezanim člancima, tematskim sekcijama, i sl.
- *U futeru* se prikazuju podaci o vlasniku veb sajta, obaveštenje o autorskim pravima, linkovi ka osnovnim sekcijama veb sajta ili linkovi ka drugim domenima u vlasništvu istog pojedinca ili kompanije.

12.3.2 Sistem za identifikaciju i autorizaciju korisnika

Autentifikacija korisnika u veb aplikaciji predstavlja bezbednosni proces prilikom interakcije na relaciji čovek-mašina, koji se koristi u situacijama kada se od korisnika zahteva da poseduje korisnički nalog i da je ulogovan.

Slika 47 - Obrazac za registraciju korisničkog naloga



12.3.2.1 Kreiranje korisničkog naloga

Prilikom kreiranja korisničkog naloga, unose se jedinstveni podaci koji pripadaju samo tom nalogu, i koji će se potom koristiti u procesu prijave za dokazivanje identiteta korisnika. Za ovu svrhu se obično koriste korisničko ime ili email adresa, i lozinka. Podaci vezani za korisnički nalog se čuvaju u odgovarajućoj tabeli u bazi podataka, pri čemu se šifra iz bezbednosnih razloga obično enkriptuje.

Radi sprečavanja eventualnih zloupotreba, uobičajena je praksa da prilikom kreiranja korisničkog naloga aplikacija na priloženu email adresu pošalje automatski generisanu poruku sa linkom za aktivaciju naloga. Ukoliko je korisnik naveo nepostojeću email adresu ili adresu kojoj nema pristup, neće moći da aktivira korisnički nalog.

12.3.2.2 Korisnički nivoi

Korisnički nivo predstavlja skup ovlašćenja koja korisnik ima u određenoj veb aplikaciji. Veb aplikacije koje funkcionišu u javnom domenu, obično imaju barem tri nivoa korisničkih naloga:

- *Anoniman korisnik* može da čita javno dostupni sadržaj, postavlja komentare, i da vrši osnovnu pretragu sadržaja.
- *Registrovani korisnik* može da unosi članke, npr. postavlja oglase, slike, video klipove i sl. Ovom korisničkom nivou se često ostavlja i mogućnost menadžmenta takvog sadržaja, npr. izmena i brisanje.
- *Administrator sistema* ima pristup svim podacima i funkcionalnostima aplikacije, uključujući i menadžment korisničkih naloga.

Slika 48 - Obrazac za prijavljivanje korisnika (login)

12.3.2.3 Prijavljivanje korisnika - log in

Prijavljivanje korisnika se vrši preko odgovarajućeg obrasca za prijavu, slika 48[27], koji se sastoji iz sledećih elemenata :

- Kratko objašnjenje o prirodi obrasca
- Elementi za unos teksta u jednoj liniji, tip *tekst* za korisničko ime, i tip *password* za lozinku.
- *Captcha* slika sa nasumično generisanim kodom i elementom za unos koda tipa *tekst,* radi prevencije zlonamernih automatizovanih pokušaja prijave.
- Dugme za slanje podataka
- Pomoćni linkovi za kreiranje ili aktivaciju korisničkog naloga.

12.3.2.4 Resetovanje korisničkog naloga

U slučaju da je korisnik zaboravio svoje korisničko ime, ali zna email adresu vezanu uz korisnički nalog, aplikacija može da pošalje mail sa korisničkim imenom.

Ukoliko je korisnik zaboravio lozinku, ali zna korisničko ime ili mail adresu koji su registrovani uz korisnički nalog, aplikacija će generisati novu šifru za korisnički nalog i korisniku poslati email poruku sa šifrom i kratkim uputstvom. Pri prvom sledećem uspešnom pokušaju prijave, dobra je praksa da se od korisnika tražiti da promeni šifru.

27 Magma CMS, https://www.magmacms.org

12.3.3 Taksonomija

Taksonomija predstavlja proces grupisanja srodnih podataka u veb aplikaciji u sekcije, kategorije, stranice, prema ključnim rečima i sl. Na slici 49 je prikazana početna strana poslovnog adresara. Profili kompanija su grupisani u kategorije prema delatnosti koju obavljaju. Svaka kategorija je predstavljena svojim nazivom, kratkim opisom i odgovarajućom sličicom ili ikonom.

Slika 49 - Grupisanje u kategorije profila kompanija prema delatnosti

12.3.4 Pretraga sadržaja

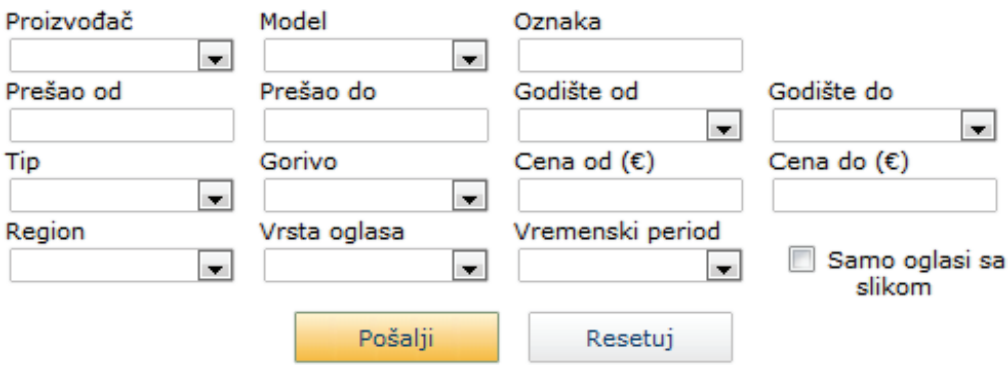
Podrazumeva se da savremene veb aplikacije poseduju funkcionalnost pretrage sadržaja, a u skladu sa tipom aplikacije primenjuju se različite metode pretrage:

- Direktna pretraga podataka u bazi podataka
- Upotreba specijalizovanih pretraživača (engl. *search engine*) na strani servera, kao što su *Elastisearch* ili *Solr*.
- Upotreba komercijalnih alata za pretragu, poput Algolia ili AWS Cloud-search

Slika 50 - Veb obrazac za jednostavnu pretragu sadržaja veb sajta

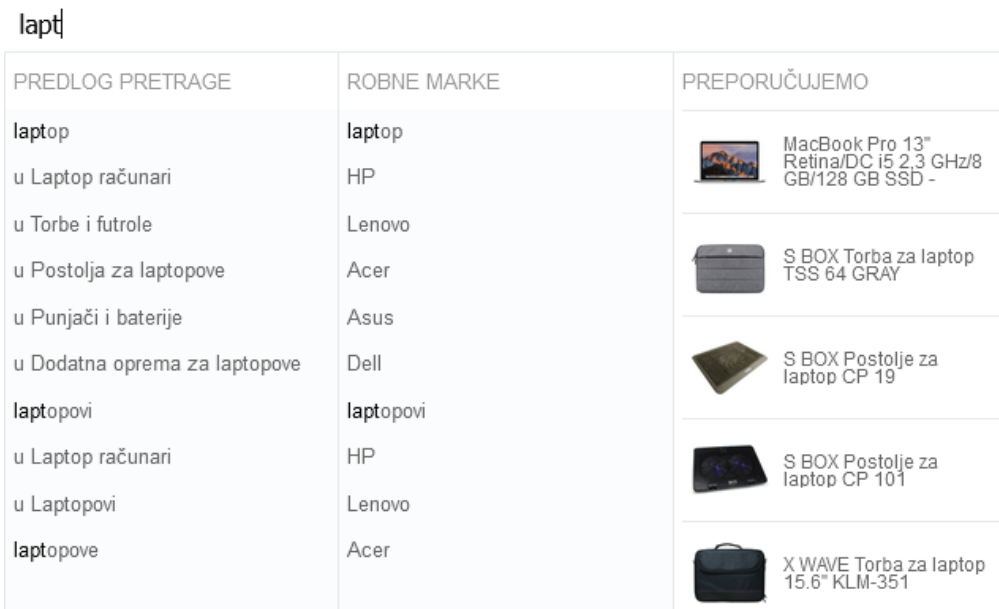
Koji će sistem pretrage biti primenjen zavisi od više faktora, kao što su frekvencija ubacivanja novih podataka u bazu podataka, vrsta podataka koji se pretražuju - npr. pretraga log fajlova i pretraga sadržaja članaka se vrše na različite načine, na koji način se tretiraju reči koje nisu bitne za rezultat pretrage (engl. *stop words*) i sl.

Slika 51 - Veb obrazac za pretragu oglasnika po osnovu više kriterijuma



Korisnik obično definiše kriterijum pretrage kao string, niz ključnih reči ili niz uslova, preko odgovarajućeg veb obrasca koji se sastoji iz jednog (slika 50) ili više input elemenata (slika 51).

Slika 52 - Veb obrazac za pretragu veb prodavnice



Napredni sistemi pretrage, slika 52, nude dodatne mogućnosti poput:

- Automatsko dovršavanje započetog stringa za pretragu (engl. *autocomplete*). Kucanjem reči *Univerzi*...aplikacija će predložiti konkretan pojam - *Univerzitet Singidunum*.
- Provera pravopisa i sugestija ključnih reči. Proverom pravopisa se filtriraju relevantne ključne reči za pretragu. Na e-Commerce veb sajtovima primenom ove tehnike se može postići da se među rezultatima pretrage nađu relevantni proizvodi za koje potrošači i ne znaju.
- Pretraga po sinonimima je ključna u online prodavnicama gde potrošači isti proizvod mogu da traže pod različitim nazivima.
- Pretraga po relevantnosti rezultata pretrage. U ovom slučaju se koristi matematički aparat koji kvantifikuje relevantnost svakog rezultata pretrage u odnosu na traženi string.

12.3.5 Višejezičnost

Pojam višejezičnosti ili lokalizacije, podrazumeva sposobnost veb aplikacije da delove korisničkog interfejsa kao što su navigacija, oznake elemenata za unos podataka u veb obrascima, instrukcije, poruke o greškama i sl. prikaže na različitim jezicima, slika 53.

Slika 53 - Višejezičnost veb aplikacije na primeru obrasca za prijavu (login) korisnika

Višejezičnost u ovom kontekstu ne podrazumeva automatsko prevođenje sadržaja na druge jezike, ali je poželjno da aplikacija poseduje ugrađenu sposobnost grupisanja srodnog sadržaja koja se potom može iskoristiti za postavljanje materijala na različitim jezicima.

Metodologija implementacije višejezičnosti zavisi od namene veb aplikacije i vrste podataka koje je potrebno sačuvati ili prikazati:

- Striktna struktura podataka, koja podrazumeva da je svaki uneseni podatak dostupan na svakom jeziku.
- Slobodna struktura podataka dozvoljava slobodno formiranje sadržaja, koji se može razlikovati na različitim jezicima.

Višejezičnost predstavlja važan alat u procesu optimizacije sadržaja za veb pretraživače (engl. *Search Engine Optimization, SEO*), zbog čega se često kombinuje sa poddomenima i semantičkim veb adresama:

- *https://sr.wikipedia.org/wiki/Универзитет_Сингидунум*
- *https://en.wikipedia.org/wiki/Singidunum_University*

U navedenom primeru, poddomeni *sr* i *en* označavaju srpski i engleski jezik, a za formiranje semantičke adrese se koristi naslov članka.

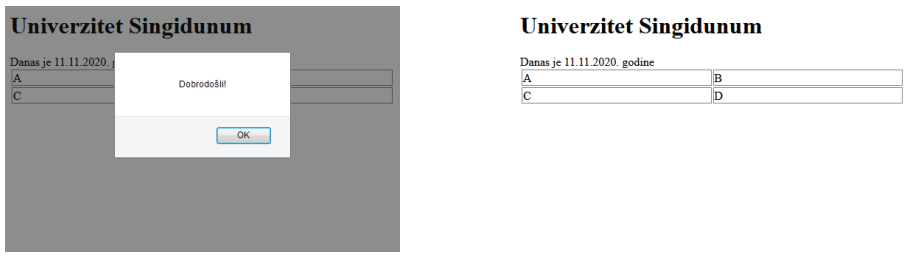
12.4 ARHITEKTURA VEB APLIKACIJE

PHP programski jezik dozvoljava da se u jednom dokumentu nađe kod pisan u različitim jezicima, listing 157, koji će proizvesti očekivani rezultat - veb stranicu, slika 54.

Listing 157 - PHP, HTML, CSS i JavaScript kod u jednom dokumentu

```
<html>
<head>
    <style>
        table { width: 100%; }
        table td { border: 1px solid gray; }
    </style>
</head>
<body>
    <h1>Univerzitet Singidunum</h1>
    <?php
        echo 'Danas je ' . date('d.m.Y.', time()) . ' godine';
    ?>
    <table>
        <tr>
            <td>A</td>
            <td>B</td>
        </tr>
        <tr>
            <td>C</td>
            <td>D</td>
        </tr>
    </table>
    <script>
        alert('Dobrodošli!');
    </script>
</body>
</html>
```

Slika 54 - Veb stranica kao rezultat koda prikazanog na listingu 157



U navedenom primeru, prilikom poziva stranice prvo se formira DOM stablo, tabela je stilizovana odgovarajućim CSS kodom i aktivira se JavaScript kod koji u prozoru ispisuje poruku dobrodošlice. Tek nakon interakcije sa ovim elementom - klik na dugme *OK*, korisnik dobija slobodan pristup stranici.

Ovakav stil pisanja koda nije optimalan. Tumačenje koda veb stranice sa složenijom strukturom će biti otežano, dok je razvoj veb aplikacija na ovaj način otežan, ili nemoguć.

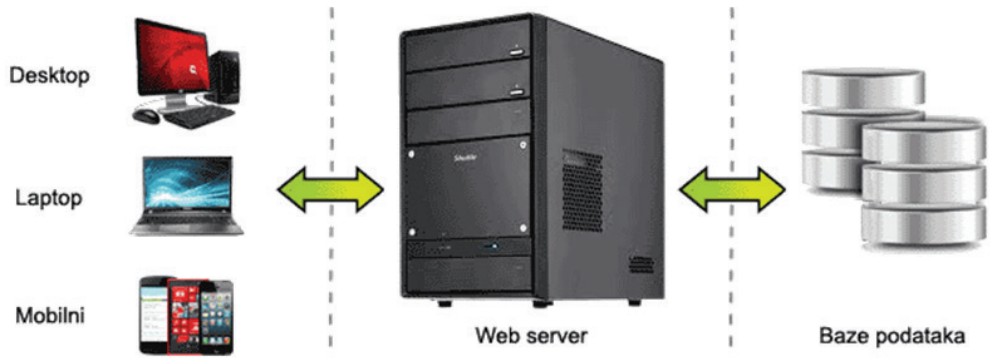
Razmotrimo sada još jedan mogući scenario. Veb aplikacija pokreće popularni veb sajt za prikaz video sadržaja, pri čemu postoji verovatnoća da će jedan video istovremeno gledati veći broj posetilaca. Video zapisi zavisno od dužine i kompresije mogu biti izuzetno veliki dokumenti, a istovremeni pristup više posetilaca takvom dokumentu može prouzrokovati otežano funkcionisanje veb servera.

Kako bi se i u ovakvim situacijama omogućilo nesmetano funkcionisanje veb sajta, isti video dokument se može postaviti na više veb servera. Veb aplikacija će pratiti opterećenje servera, i na osnovu toga svaki novi zahtev upućivati ka serveru koji je u tom trenutku najmanje opterećen.

12.4.1 Slojevi aplikacije

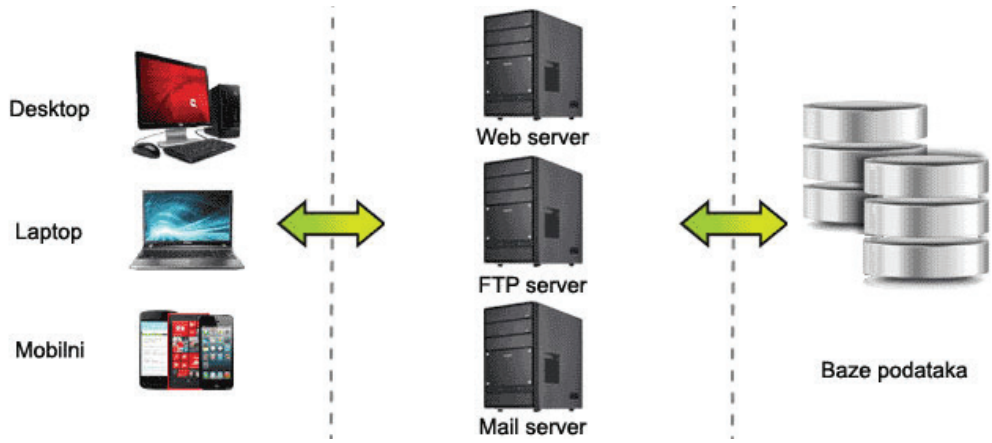
Veb aplikacija može imati fizičke nivoe (engl. *Tiers*) i logičke slojeve (engl. *Layers*). Segmentacija arhitekture aplikacije u slojeve omogućava razvoj fleksibilnih veb aplikacija koje mogu da odgovore različitim potrebama primene i čiji se pojedini delovi mogu iznova koristiti u različitim projektima, tzv. moduli. Izmene u jednom sloju su izolovane u odnosu na druge slojeve, što olakšava održavanje aplikacije i dodavanje novih funkcionalnosti. Međutim, što je broj implementiranih slojeva veći, to je i arhitektura aplikacije složenija i teže je pratiti tok podataka kroz slojeve.

Slika 55 - Troslojna arhitektura veb aplikacije



Izbor arhitekture veb aplikacije se vrši na osnovu trenutnih, ali i budućih potreba sistema. Potrebe sistema se procenjuju na osnovu više kriterijuma kao što su projektovani broj korisnika veb aplikacije, tehničkih mogućnosti primenjene platforme i softverskih alata, tipa i veličine baze podataka, tipa, prosečne veličine i količine dokumenata i sl.

Slika 56 - Višeslojna arhitektura veb aplikacije



12.4.1.1 Fizički nivoi

Fizički nivoi su mesta na kojima se segmenti infrastrukture potrebne za rad aplikacije nalaze i izvršavaju. Fizički nivoi se koriste da bi se omogućile ravnoteža između performansi aplikacije, skalabilnost u pogledu opterećenja u saobraćaju koji aplikacija može da podnese, tolerancija na greške i unapredila bezbednost sistema.

Osnovni fizički nivoi jedne veb aplikacije mogu da budu veb server, email server, jedan ili više servera baze podataka, mreža za distribuciju sadržaja (engl. *Content Delivery Network, CDN*) i sl, slike 55 i 56.

12.4.1.2 Logički slojevi

Logički slojevi predstavljaju metodologiju organizacije programskog koda veb aplikacije. To se može ostvariti segmentiranjem koda veb aplikacije kroz sistem dokumenata, klasa, funkcija, imenskih prostora i sl. Osnovni efekat koji se postiže primenom logičkih slojeva je pregledniji kod aplikacije, koji se lakše čita, ne zahteva dodatnu dokumentaciju da bi se razumeo, i samim tim implicira lakše održavanje softvera. Drugi, ali isto toliko važan efekat jeste da se primenom ovakve metodologije dobija arhitektura aplikacije koja je skalabilna, odnosno dozvoljava različite opsege funkcionalnosti i performansi u skladu sa konkretnim potrebama projekta.

Logički slojevi takođe omogućavaju lakšu kontrolu autorizacije pristupa podacima, softversku kontrolu opterećenja aplikacije, primenu šablona (engl. *template*) u kodu aplikacije, ali i u prezentacionom sloju.

12.4.2 Moduli aplikacije

Modularnost je tehnika u programiranju koja omogućava da se deo koda, obično određena funkcionalnost, strukturno izdvoji od ostatka koda u tzv. modul. Kod modula sadrži sve neophodne elemente za izvršavanje željene funkcionalnosti, i po potrebi se može uključiti ili isključiti iz aplikacije.

Modularno programiranje je u tesnoj vezi sa strukturnim i objektno orijentisanim programiranjem, a sve radi postizanja optimalne arhitekture složenih veb aplikacija.

PHP ne poseduje direktnu podršku za ovakav tip programiranja, ali to nije ograničenje da se prave modularne veb aplikacije. U tu svrhu se mogu koristiti PHP funkcije opisane u poglavlju 2.14:

- include('naziv_modula.php');
- include_once('naziv_modula.php');
- require('naziv_modula.php'); ili
- require_once('naziv_modula.php');

Slika 57 - Struktura dokumenata veb aplikacije

```
veb-aplikacija
├ moduli
│  ├ galerija.php
│  ├ kontakt.php
│  ├ pocetna.php
│  └ vesti.php
├ public
│  ├ css
│  │  └ style.css
│  ├ images
│  └ js
│     └ script.js
├ view
│  ├ galerija.html
│  ├ kontakt.html
│  ├ pocetna.html
│  ├ stranica-footer.html
│  ├ stranica-header.html
│  ├ stranica-meni.html
│  └ vesti.html
└ index.php
```

12.4.3 Struktura dokumenata

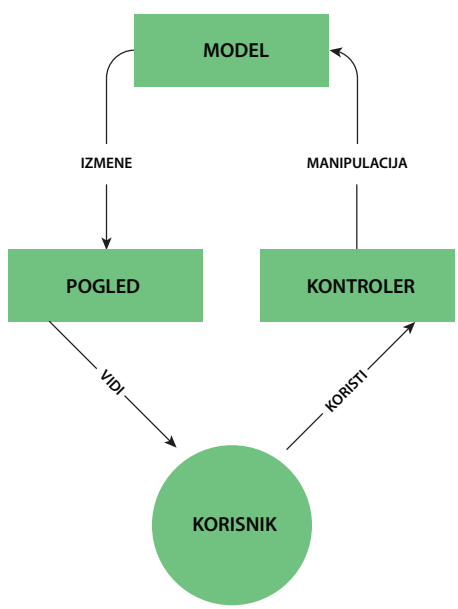
Kako je prethodno navedeno, cela PHP veb aplikacija može da se smesti u samo jedan dokument, uključujući HTML, CSS i JavaScript kod. Iako tehnologija dozvoljava ovakav pristup, iz perspektive održavanja i daljeg razvoja aplikacije to nije optimalno.

Radi lakšeg razvoja, kod aplikacije se može segmentirati u srodne dokumente, a dokumenti se mogu grupisati u foldere, slika 57.

U prikazanom primeru, veb aplikacija ima samo jedan ulazni dokument, *index.php* koji se nalazi u osnovnom direktorijumu projekta. Fajlovi koji pripadaju projektu su razvrstani u tri direktorijuma:

- *moduli* - sadrži fajlove, koji svaki za sebe predstavlja jedan modul aplikacije.
- *public* - sadrži javno dostupne fajlove, kao što su CSS i JavaScript dokumenti, ili slike.
- *view* - sadrži fajlove u kojima se nalazi kod za prikaz rezultata rada određenog modula, tzv. prezentacioni sloj, ili zajedničke elemente korisničkog interfejsa kao što su zaglavlje, futer ili navigacioni meni.

Slika 58 - MVC model



12.4.4 MVC arhitektura

MVC arhitektura (engl. *Model-View-Controller*) se koristi u složenim aplikacijama kao efikasan metod za razdvajanje funkcionalnih celina radi lakšeg razumevanja i održavanja koda, slika 58.

- *Kontroler*, ili *ruter* (engl. *controller*, ili *router*) aplikacije je deo koda koji je zadužen da protumači zahtev koji je korisnik poslao aplikaciji, po potrebi verifikuje i potom ga prosledi odgovarajućem modelu na dalju obradu.
- *Model* je deo koda aplikacije zadužen za obradu podataka. Ulazne informacije dobija od kontrolera.
- *Pogled* (engl. *View*) predstavlja vizuelnu prezentaciju modela, ili samo jednog dela modela. U kontekstu veb aplikacije, to je veb stranica, ili deo veb stranice.

U primeru prikazanom na slici 57, kontroler se može smestiti u *index.php* fajl koji se nalazi u osnovnom folderu projekta, modeli se nalaze u folderu *moduli,* dok se pogledi nalaze u folderu view.

12.4.5 Ruter aplikacije

Razmotrimo zadatak izrade veb aplikacije za pokretanje veb sajta sa sledećom strukturom:

- Početna strana
- Vesti
- Galerija slika
- Kontakt stranica

Arhitektura ovakve veb aplikacije može da se organizuje u svega četiri PHP dokumenta u osnovnom (*root*) direktorijumu projekta, kojima bi korisnik direktno pristupao bez potrebe da se pravi ruter aplikacije, što na prvi pogled deluje kao najjednostavnije rešenje:

- index.php
- vesti.php
- galerija.php
- kontakt.php

Ako se uzmu u obzir osnovne funkcionalnosti, strukturu svake od navedenih stranica je moguće modelirati na sledeći način:

1. Uspostavljanje veze sa bazom podataka
2. Provera kredencijala korisnika (npr. anoniman ili admin)
3. Čitanje podataka iz baze podataka
4. Formiranje stranice
   - Zaglavlje
   - Navigacioni meni
   - Glavni sadržaj stranice
   - Futer
5. Zatvaranje veze sa bazom podataka

Očigledno je da stranice imaju puno zajedničkih elemenata koda koji bi se bespotrebno ponavljali, što bi otežavalo dalji razvoj ili održavanje aplikacije.

Tabela 17 - Ruter veb aplikacije - upotreba URL parametara i semantičkih URL

URL parametri	*Semantički URL*	*Aktivirani modul*
/index.php	/	Početna strana
/index.php?stranica=vesti	/vesti	Vesti
/index.php?stranica=vesti&id=123	/vesti/123	Pregled vesti
/index.php?stranica=galerija	/galerija	Galerija
/index.php?stranica=galerija&id=123	/galerija/123	Pregled slike
/index.php?stranica=kontakt	/kontakt	Kontakt

Alternativno, arhitekturu aplikacije je moguće reorganizovati tako da se svim sekcijama, ili funkionalnostima aplikacije pristupa samo preko jednog fajla - *index.php* koji se nalazi u osnovnom direktorijumu projekta. Struktura aplikacije se modelira na sličan način kao u prethodnom slučaju:

1. Uspostavljanje veze sa bazom podataka
2. Ruter aplikacije
3. Uključivanje odgovarajućeg modula
   - Provera kredencijala korisnika (npr. anoniman ili admin)
   - Čitanje podataka iz baze podataka
   - Formiranje sadržaja za ispis

4. Formiranje stranice
   - Zaglavlje
   - Navigacioni meni
   - Glavni sadržaj stranice
   - Futer
5. Zatvaranje veze sa bazom podataka

Ruter aplikacije tumači URL parametre (*$_GET*) ili semantički URL (*$_SERVER ['REQUEST_URI']*) preko koga je izvršen poziv, i na osnovu toga pokušava da "razume" šta korisnik traži od aplikacije, npr. koju stranicu želi da pregleda, tabela 17.

Na osnovu izloženog, ruter se može formirati u samo jednoj liniji koda,

```
include("moduli/{$_GET['s']}");
```

što je primer loše prakse, zato što se bez provere veruje informacijama koje daje korisnik aplikacije. Ovako formiran ruter omogućava aktivaciju bilo kog PHP dokumenta bez obzira na direktorijum u kojem se nalazi, npr.

```
https://singidunum.ac.rs/index.php?s=. . /slike/napad.php
```

U slučaju da je napadač prethodno uspeo na veb server da ubaci svoj PHP fajl napad.php u folder slike, ovako formiran ruter bi mu omogućio aktivaciju takvog fajla.

Listing 158 - Ruter veb aplikacije

```php
<?php
$stranica = @$_GET['stranica'];
switch($stranica) {
  case '':
    $modul = 'pocetna';
    break;
  case 'vesti' :
  case 'galerija' :
  case 'kontakt' :
    $modul = $stranica;
    break;
  default :
    $modul = 'greska';
    break;
```

U primeru prikazanom na listingu 158, ruter tumači zahtev očitavanjem URL parametra stranica i na osnovu njegove vrednosti formira ime modula, linije 2-15, koji se potom uključuje u glavni tok programa, linija 16, što u potpunosti odgovara strukturi fajlova sistema prikazanoj na slici 57. Za razliku od prethodnog primera, ruter će reagovati u skladu sa predefinisanim vrednostima, dok će za sve ostale vrednosti koje ne pripadaju tom skupu aktivirati modul *greska*.

PRIMER PROJEKTNOG ZADATKA

13. PRIMER PROJEKTNOG ZADATKA

U ovom poglavlju je prikazan primer izrade veb aplikacije primenom tehnologije i tehnika opisanih u ovom udžbeniku. Aplikacija će biti *sistem za upravljanje sadržajem* (CMS), sa zadatkom da pokreće veb sajt Univerziteta, na kome će studenti moći da se informišu o najnovijim dešavanjima. Sadržaj sajta će kreirati administrator kroz korisnički interfejs, bez potrebe da se direktno pristupa dokumentima na veb serveru ili kodu same aplikacije. Arhitektura aplikacije će biti formirana kao radni okvir (engl. *framework*), što će omogućiti naknadnu ugradnju funkcionalnosti i modula, i upotrebu koda u drugim projektima, bez obzira na temu.

13.1 SISTEM ZA UPRAVLJANJE SADRŽAJEM

Pojam *sistem za upravljanje sadržajem* (engl. *Content Management System, CMS*) se odnosi na veb aplikaciju koja korisniku omogućava uređivanje veb sadržaja bez potrebe za poznavanjem programiranja.

Primenjuju se u situacijama kada je sadržaj veb sajta potrebno češće menjati kroz unos, izmenu ili brisanje informacija. To može biti lični veb sajt sa malim brojm stranica, ali i posećeni medijski portal gde se u toku dana objavljuje veći broj vesti, i gde je korisnicima omogućeno da ostavljaju komentare ili da ocenjuju članke. CMS se koriste i u slučajevima kada je posetiocima veb sajta dozvoljeno da samostalno postavljaju informacije i da upravljaju svojim sadržajem, npr. u *oglasnicima*, ili da od aplikacije zahtevaju određenu uslugu, npr. *eUprava*.

U skladu sa specifikacijom projektnog zadatka (poslovni model, tehnički zahtevi, budžet, vremenski okviri realizacije projekta, raspoloživi ljudski i materijalni resursi itd.), može se koristiti CMS već dostupan na tržištu ukoliko zadovoljava zadate parametre, ili se može pristupiti razvoju novog rešenja.

13.2 OSNOVNE POSTAVKE PROJEKTA

Kao prvi korak prilikom izrade projekta, potrebno je definisati osnovne elemente, i napraviti smernice na koji način će oni biti realizovani:

- Kakav digitalni sadržaj će biti prikazan na veb sajtu? Kakva je struktura sadržaja, odnosno da li i kako se informacije mogu grupisati? Koliko se često menja sadržaj i kako o tome obavestiti korisnika?
- Na osnovu toga se potom mogu formirati struktura glavnog navigacionog menija i struktura početne strane.
- Koliko će aplikacija imati korisnika, kakva je struktura korisnika, i koliko korisničkih nivoa je potrebno?
- Struktura baze podataka. Potrebne tabele, i struktura svake tabele.
- Kako će biti organizovan korisnički interfejs? Potrebno je predvideti elemente koji treba da budu prikazani na svakoj veb stranici, njihov raspored, i korake koje korisnik treba da izvede prilikom svake operacije u kojoj se očekuje njegova akcija.
- Arhitektura aplikacije. Definisati neophodne module koji će pokretati veb sajt, strukturu fajlova, arhitekturu koda.
- Da li postoji korisnička podrška, odnosno na koji način posetilac sajta može da stupi u kontakt sa administracijom?

13.3 GRUPISANJE SADRŽAJA

Javno dostupan sadržaj veb sajta se može grupisati na sledeći način:

- Početna strana
- Vesti
- Studenti
- Projekti
- Kontakt stranica

Na početnoj strani se prikazuje listing sa tri najnovije vesti, slika 59. Naslov vesti je veb link ka odgovarajućem članku, dok se ispod naslova prikazuje datum objave.

Slika 59 - Početna strana

Sekcije *Vesti, Studenti* i *Projekti* tematski grupišu srodne članke. Kako je u pitanju informativni veb sajt, svi članci su po svojoj strukturi identični i čine ih:

- Naslov članka
- Kratak opis
- Pun tekst članka
- Datum kada je članak objavljen

Prilikom ulaska u neku od ovih sekcija, na veb stranici se prikazuje spisak vesti, slika 60, sortiranih opadajućim redosledom po datumu objave tako da se najnovije vesti nalaze na početku spiska.

Ukoliko korisnik klikne na naslov vesti, aplikacija će prikazati veb stranicu sa člankom prikazanim u celini, slika 61. Dodatni element na ovakvoj stranici su tzv. mrvice hleba (engl. *breadcrumbs*), sa ulogom pomoćnog elementa navigacije koji se postavlja iznad naslova stranice (H1 element) i korisniku ukazuje na korake u navigaciji od početne do trenutne stranice. U ovom slučaju, to može biti

```
Početna > Vesti
```

Slika 60 - Sekcija Vesti, spisak objavljenih članaka

Slika 61 - Prikaz članka

Slika 62 - Kontakt stranica

Napokon, na kontakt stranici je prikazan veb obrazac u koji korisnik može da unese osnovne podatke kao što su ime i prezime, e-mail adresa i kratka poruka, koji će biti poslati administratoru veb sajta.

13.4 STRUKTURA KORISNIKA. TIPOVI KORISNIČKIH NALOGA.

U ovom projektnom zadatku, dovoljno je da veb aplikacija može da prepozna dva tipa korisnika:

- *Anonimni korisnik* ne mora da se identifikuje, može samo da čita sadržaj objavljen na veb sajtu
- *Administrator* se prijavljuje putem login obrasca unošenjem korisničkog imena i lozinke. Osim što ima pristup sadržaju kao i anonimni korisnik, može da unosi nove, menja ili briše postojeće članke.

Jednostavnosti radi, aplikacija će imati samo jednog administratora, sa pristupnim parametrima *admin* (korisničko ime) i *admin* (lozinka).

13.5 STRUKTURA BAZE PODATAKA

Sve vesti, bez obzira kojoj sekciji sajta pripadaju, imaju istu strukturu i zato mogu da se skladište u jednoj tabeli, listing 159. Pripadnost sekciji određuje numerički podatak u koloni *article_category_id*, koji odgovara koloni *category_id* u tabeli categories, listing 160. Ova tabela se popunjava u skladu strukturom kategorija usvojenom u poglavlju 13.3, listing 161. Kolona *article_id* je primarni ključ sa *AUTO_INCREMENT* svojstvom, i ima ulogu referentne koordinate prilikom izmene ili brisanja podataka, kao i u slučajevima kada se vrši spajanje podataka iz više tabela (*JOIN*).

Listing 159 - SQL upit za formiranje tabele articles

```
CREATE TABLE `articles` (
  `article_id` int(11) UNSIGNED NOT NULL,
  `article_category_id` int(11) UNSIGNED NOT NULL DEFAULT 0,
  `article_title` varchar(255) NOT NULL,
  `article_short` varchar(255) NOT NULL,
  `article_full` text NOT NULL,
  `article_date` datetime DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Listing 160 - SQL upit za formiranje tabele categories

```
CREATE TABLE `categories` (
  `category_id` int(10) UNSIGNED NOT NULL,
  `title` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Listing 161 - SQL upit za ubacivanje kategorija, tabela categories

```
INSERT INTO `categories` (`category_id`, `title`) VALUES
(1, 'Vesti'),
(2, 'Studenti'),
(3, 'Projekti');
```

Na listingu 162 je prikazan SQL kod koji odgovarajućim kolonama u tabelama *articles* i *categories* formira indekse.

Listing 162 - SQL upit za postavljanje indeksa u tabelama articles i categories

```
ALTER TABLE `articles`
  ADD PRIMARY KEY (`article_id`),
  ADD KEY `kategorija_id` (`article_category_id`);
ALTER TABLE `categories` ADD PRIMARY KEY (`category_id`);
ALTER TABLE `articles` MODIFY `article_id` int(11)
  UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;
ALTER TABLE `categories` MODIFY `category_id` int(10)
  UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
```

U sekciji 13.4 je navedeno da će aplikacija imati samo dva tipa korisničkih naloga, pri čemu su pristupni podaci za administratora predefinisani (korisničko ime *admin* i šifra *admin*). Priroda veb sajta koji će ova aplikacija pokretati je takva da je dovoljna samo jedna osoba za administraciju, bez potrebe za kreiranjem dodatnih korisničkih naloga. Iz ovih razloga tabela sa korisnicima neće biti formirana.

Listing 163 - SQL upit za formiranje tabele korisnici

```
CREATE TABLE `korisnici` (
  `id` int(11) UNSIGNED NOT NULL,
  `korisnicko_ime` varchar(32) NOT NULL,
  `lozinka` varchar(32) NOT NULL,
  `korisnicki_nivo` tinyint(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ukoliko bi se u kasnijim fazama razvoja ili eksploatacije javila potreba da se omogući korisnicima kreiranje naloga, tzv. proces registracije, neophodno bi bilo da se formira tabela u kojoj će se čuvati podaci o korisničkim nalozima. Na listingu 163 je prikazan SQL upit za kreiranje takve tabele sa imenom korisnici:

- *id* - primarni ključ, *AUTO_INCREMENT*
- *korisnicko_ime* - string dužine 32 karaktera
- *lozinka* - string dužine 32 karaktera. Ovaj podatak se iz bezbednosnih razloga u bazi podataka čuva kriptovan. Dužina od 32 karaktera odgovara *MD5* enkripciji, a u slučaju upotrebe drugačijeg algoritma potrebno je prilagoditi i dužinu podataka u ovoj koloni.

U ovom poglavlju je napravljen propust koji neće uticati na rad aplikacije, ali može uticati na proces održavanja - dve tabele su formirane sa imenima na engleskom jeziku (*articles* i *categories*), dok je predlog strukture tabele sa korisničkim nalozima dat na srpskom jeziku (tabela *korisnici*). Prilikom izrade projekta, potrebno je da se na samom početku odredi jezik na kome će se pisati kod, i potom to poštovati tokom celokupnog procesa razvoja.

13.6 KORISNIČKI INTERFEJS

Korisnički interfejs treba da bude jednoobrazan na svim stranicama, a kako je navedeno u poglavlju 12.3.1, krajnjem korisniku treba da obezbedi lakoću upotrebe veb sajta, uključujući jednostavnu navigaciju i jasan prikaz informacija. Na slici 69 je prikazan izgled početne strane veb sajta sa obeleženim osnovnim strukturnim elementima stranice, koji će se iskoristiti za formiranje šablona za izradu svih veb stranica u okviru ove aplikacije:

- *Zaglavlje* (engl. *header*) stranice, u kome se nalaze link za prijavu korisnika (engl. *login*) i logo vlasnika sajta.
- *Glavni meni*, gde su prikazani linkovi ka osnovnim sekcijama veb sajta. Od velike je važnosti da struktura glavnog menija na svim stranicama bude ista.
- *Naslov stranice*, H1 HTML element. To može biti naslov sekcije u kojoj se korisnik trenutno nalazi (*Vesti, Studenti, Projekti, Kontakt*), ili naslov članka koji se pregleda.
- *Sadržaj veb stranice* može biti spisak najnovijih vesti na početnoj strani, listing vesti u odgovarajućoj sekciji (*Vesti, Studenti, Projekti*), slika 60, tekst članka koji se pregleda, slika 61, ili veb obrazac za kontakt sa administratorom sajta na stranici *Kontakt*, slika 62.
- *Footer* se prikazuje na dnu stranice, i obično sadrži informaciju o vlasniku sata.

13.7 ARHITEKTURA APLIKACIJE

Prilikom formiranje strukture aplikacije, primeniće se MVC model opisan u poglavlju 12.4.4. Za pristup stranicama će se koristiti semantički linkovi, v. poglavlje 1.8.4, dok će aplikacija imati samo jednu pristupnu tačku putem dokumenta *index.php* koji se nalazi u osnovnom direktorijumu projekta, slika 63. U tabeli 18 su prikazani javno dostupni linkovi i odgovor rutera kojima mogu da pristupe anonimni korisnici, dok su u tabeli 19 prikazani linkovi i odgovor rutera kojima može da pristupi samo administrator.

Tabela 18 - Semantički linkovi, anoniman korisnik

Semantički link	***Reakcija rutera aplikacije***
/	*Početna strana*
/vesti	Sekcija *Vesti*, listing članaka
/vesti/1	Pregled članka u sekciji *Vesti, article_id*=1
/studenti	Sekcija *Studenti*, listing članaka
/studenti/2	Pregled članka u sekciji *Studenti, article_id=2*
/projekti	Sekcija *Projekti*, listing članaka
/projekti/3	Pregled članka u sekciji *Projekti, article_id=3*
/kontakt	*Kontakt strana*
/rss	*RSS feed*
/greska	*Ruter* nije razumeo adresu, redirekcija na stranicu Greška

Tabela 19 - Semantički linkovi, administrator

Semantički link	***Reakcija rutera aplikacije***
/login	Strana za prijavljivanje *(login)*
/logout	Odjava prijavljenog korisnika
/vesti/submit	Unos novog članka u sekciju Vesti
/vesti/edit/1	Izmena postojećeg članka u sekciji Vesti
/vesti/delete/1	Brisanje članka iz sekcije Vesti
/studenti/submit	Unos novog članka u sekciju Studenti
/studenti/edit/2	Izmena postojećeg članka u sekciji Studenti
/studenti/delete/2	Brisanje članka iz sekcije Studenti
/projekti/submit	Unos novog članka u sekciju Projekti
/projekti/edit/3	Izmena postojećeg članka u sekciji Projekti
/projekti/delete/3	Brisanje članka iz sekcije Projekti

Slika 63 - Struktura dokumenata veb aplikacije

```
projekat
├ includes
│ ├ config.php
│ ├ db.php
│ ├ funkcije.php
│ ├ funkcije_rss.php
│ ├ funkcije_vesti.php
│ ├ router.php
│ ├ stranica_greska.php
│ ├ stranica_kontakt.php
│ ├ stranica_login.php
│ ├ stranica_pocetna.php
│ ├ stranica_rss.php
│ ├ stranica_vesti.php
│ ├ stranica_vesti_admin.php
│ ├ stranica_vesti_spisak.php
│ └ stranica_vesti_vest.php
├ public
│ ├ css
│ │ ├ all.min.css
│ │ └ style.css
│ ├ images
│ │ └ singidunum-logo.png
│ ├ js
│ │ ├ jquery.js
│ │ └ script.js
│ └ webfonts
│   ├ fa-brands-400.eot
│   ├ fa-brands-400.svg
│   ├ fa-brands-400.ttf
│   ├ fa-brands-400.woff
│   ├ fa-brands-400.woff2
│   └ ...
├ template
│ ├ breadcrumbs.php
│ ├ footer.php
│ ├ greska.php
│ ├ header.php
│ ├ stranica_greska.php
│ ├ stranica_kontakt.php
│ ├ stranica_login.php
│ ├ stranica_pocetna.php
│ ├ stranica_rss.php
│ ├ stranica_vest.php
│ ├ stranica_vesti.php
│ └ stranica_vesti_edit.php
├ .htaccess
└ index.php
```

Listing 164 - Preusmeravanje tražene URL adrese na fajl *index.php*

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule . index.php [L]
```

Veb server prema osnovnim podešavanjima ne razume semantičke linkove. Da bi se omogućila njihova upotreba, Apache modul *mod_rewrit*e mora biti aktivan, a instrukcije za preusmeravanje tražene URL adrese na *index.php* fajl se nalaze u dokumentu *.htaccess*, listing 164, koji se takođe nalazi u osnovnom direktorijumu projekta.

Listing 165 - Ulazna tačka aplikacije, *index.php*

```
<?php
session_start();
define('STRANICA_GRESKA', '/greska');
define('DIR_INCLUDES', 'includes/');
define('DIR_TEMPLATE', 'template/');
define('DIR_PUBLIC', '/public/');
define('DIR_JS', DIR_PUBLIC . 'js/');
define('DIR_CSS', DIR_PUBLIC . 'css/');
define('HOST_URL', sprintf(
  '%s://%s',
  $_SERVER['REQUEST_SCHEME'],
  $_SERVER['HTTP_HOST']
 ));
define('PRIJAVLJEN', $_SESSION['prijavljen'] ?? 0);
include(DIR_INCLUDES . 'db.php');
$db = db();
include(DIR_INCLUDES . 'funkcije.php');
$kategorije = kategorije();
include(DIR_INCLUDES . 'router.php');
if (!defined('DISABLE_HEADER'))
 include(DIR_TEMPLATE . 'header.php');
include($modul_ime);
if (!defined('DISABLE_FOOTER'))
 include(DIR_TEMPLATE . 'footer.php');
$db->close();
?>
```

Struktura fajla *index.php* prikazanog na listingu 165 se može podeliti na sledeće segmente:

- Pokretanje sesije radi identifikacije korisnika, linija 2.
- Definisanje konstanti, linije 3-14.
- Uspostavljanje veze sa bazom podataka, linije 15-16.
- Očitavanje spiska kategorija iz baze podataka, linije 17-18.
- Uključivanje rutera aplikacije, linija 19.
- Aktiviranje modula i formiranje veb stranice, linije 20-25.

Osim fajlova *.htaccess* i *index.php* koji se nalaze u osnovnom direktorijumu projekta, svi ostali dokumenti aplikacije su grupisani u poddirektorijume, prema svojoj nameni:

- *includes* direktorijum sadrži PHP fajlove koji čine jezgro frejmvorka, ruter (kontroler u MVC arhitekturi), zajedničke funkcije i module (*model* u MVC arhitekturi).
- *public* direktorijum sadrži javno dostupne fajlove, kao što su CSS i JavaScript dokumenti, slike i fontove.
- *template* direktorijum sadrži segmente šablona za formiranje veb stranica, tzv. poglede u *MVC* modelu.

Listing 166 - Konfiguracioni podaci, *config.php*

```
1 <?php
2 $host = 'localhost';
3 $port = '3306';
4 $username = 'root';
5 $password = '';
6 $db_name = 'veb_programiranje';
7 ?>
```

13.7.1 Jezgro aplikacije

Jezgro aplikacije se nalazi u direktorijumu *includes*. Korisnik ne sme direktno da pristupa ovim dokumentima, već se oni pozivaju preko pristupnog dokumenta *index.php* i u skladu sa time kako je ruter protumačio traženu URL adresu.

Dokument *config.php*, listing 166, sadrži osnovne podatke za uspostavljanje veze sa bazom podataka, kao što su korisničko ime i šifra, linije 4-5, ime servera na kome se nalazi baza podataka i port na kome se pristupa, linije 2-3, kao i ime baze podataka, linija 6. Ovi podaci se mogu razlikovati u različitim projektima, i potrebno je omogućiti administratoru da može da ih prilagodi prema potrebi, zbog čega se nalaze u posebnom dokumentu izolovani od ostatka koda aplikacije.

Listing 167 - Uspostavljanje veze sa bazom podataka, *db.php*

```
1 <?php
2 function db() {
3  include(DIR_INCLUDES . 'config.php');
4  $db = new mysqli(
5         "{$host}:{$port}",
6         $username,
7         $password,
8         $db_name
9        );
10  if ($db->connect_errno)
11   die("Veza sa MySQL serverom nije
12     uspostavljena: {$db->connect_error}");
13  return $db;
14 };
15 ?>
```

Rad sa bazom podataka se može organizovati na različite načine u skladu sa konkretnim potrebama projekta, kako je opisano u poglavlju 6. U ovom primeru, koriste se *MySQL*i ekstenzija i *OOP* pristup, listing 167. Kod za uspostavljanje veze sa bazom podataka je formiran kao funkcija *db()* da bi se konfiguracioni podaci, linija 3, izolovali u odnosu na glavni tok aplikacije, v. poglavlje 2.3.1.

Ako je veza sa bazom uspostavljena, funkcija kao rezultat vraća objekat, linija 13, a u suprotnom zaustavlja aplikaciju i ispisuje grešku, linije 10-12.

U dokumentu *funkcije.php* se nalaze funkcije opšte namene, koje mogu biti korišćene po potrebi u ostatku aplikacije, listing 168.

Listing 168 - Zajedničke funkcije, *funkcije.php*

```
1 <?php
2 $kategorije = function() use ($db) {
3  $kategorije = [];
4  $sql = "SELECT * FROM `categories`";
5  $rezultat = $db->query($sql);
6  while ($kat = $rezultat->fetch_assoc()) {
7   $kategorije[$kat['category_id']] = $kat['title'];
8  }
9  return $kategorije;
10 };
11 function breadcrumbs($b) {
12  $breadcrumbs = [];
13  foreach($b AS $u => $t) {
14   $breadcrumbs[] =
15    sprintf(
16     '<a href="%s">%s</a>',
17     strtolower($u),
18     $t
19    );
20  }
21  return sprintf(
22   '<breadcrumbs>%s</breadcrumbs>',
23   implode('<razdelnik></razdelnik>', $breadcrumbs)
24  );
25 }
26 function redirect($location) {
27  header("Location: {$location}");
28  exit;
29 }
30 ?>
```

Očitavanje listinga kategorija je implementirano kao anonimna funkcija, linije 2-10, da bi se eliminisala potreba za navođenjem objekta *$db* kao atributa funkcije, ili njegovog deklarisanja kao globalne promenljive:

```
function kategorije($db) {
...
}
```

```
function kategorije() {
global $db;
...
}
```

Prilikom upotrebe aplikacije, pojedine situacije zahtevaju redirekciju, odnosno prelaz veb brauzera na drugu adresu, npr. :

- Nakon unosa, ili izmene članka, aplikacija neće ponovo prikazati obrazac za unos, već će izvršiti redirekciju na listing članaka u odgovarajućoj sekciji.
- Korisnik se odjavljuje iz aplikacije upotrebom semantičkog linka */logout*, kako je prikazano u tabeli 19. Ruter aplikacije će ovaj link razumeti na ispravan način, ali nakon podešavanja statusa korisnika na *odjavljen* ne može da prikaže prazan ekran, ili da aktivira modul koji je van konteksta ovog linka. Zato će izvršiti redirekciju na početnu stranu veb sajta.

Redirekcija se vrši upotrebom funkcije *redirect()*, linije 26-29, gde se kao atribut funkcije navodi putanja, odnosno adresa na koju veb brauzer treba da prebaci korisnika.

Zadatak funkcije *breadcrumbs()*, linije 11-25, je da formira HTML kod za prikaz navigacionih mrvica hleba, koje će biti postavljene iznad naslova stranice prilikom pregleda članka, slika 61. Navigacione mrvice (niz) se prosleđuju kao atribut *$b*, koji funkcija transformiše u HTML linkove (string). U ovom slučaju, ne koristi se fajl *breadcrumbs.php* prikazan u direktorijumu *template*, slika 63.

Alternativno, moguće je izbeći formiranje HTML koda na ovakav način. U poglavlju 12.4 je opisano kako se PHP i HTML kod mogu razdvojiti u posebne fajlove, radi lakšeg održavanja. U ovom primeru, funkcija *breadcrumbs()* se može transformisati u

```
function breadcrumbs($breadcrumbs) {
  include(DIR_TEMPLATE . 'breadcrumbs.php');
}
```

dok je sadržaj fajla breadcrumbs.php u direktorijumu template prikazan na listingu 169.

Listing 169 - Šablon (templejt) za prikaz mrvica hleba, *breadcrumbs.php*

```
1 <breadcrumbs>
2  <?php foreach($breadcrumbs AS $u => $t): ?>
3   <a href="<?= $u ?>"><?= $t ?></a>
4  <?php endforeach; ?>
5 </breadcrumbs>
```

Funkcije koje se koriste samo u određenom modulu, mogu se grupisati u posebne fajlove, slika 63, npr:

- Fajl *funkcije_rss.php* pripada modulu rss
- Fajl *funkcije_vesti.php* pripada modulu vesti

13.7.2 Ruter aplikacije

Zadatak rutera (kontrolera) aplikacije, v. poglavlje 12.4.5, je da protumači semantičku adresu, tabele 18 i 19 i na osnovu toga prosledi informaciju aplikaciji šta korisnik želi, odnosno koji modul aplikacije je potrebno pokrenuti, listing 170. Polazna tačka je URI zahteva, linija 2.

Ukoliko je korisnik pozvao veb adresu

```
https://www.singidunum.ac.rs/studenti
```

vrednost u superglobalnoj matrici *$_SERVER* sa ključem *REQUEST_URI* će biti

```
/studenti
```

U pitanju je string, koji počinje kosom crtom, bez obzira na upit koji je upućen veb serveru. Kosa crta se uklanja, funkcija *ltrim()*, nakon čega se dobija string čija vrednost je smeštena u promenljivoj *$stranica* i može da se uporedi sa uvedenim pretpostavkama:

- Ukoliko je string prazan, upućeni zahtev se odnosi na početnu stranu, linije 3-4.
- U suprotnom, pomoću regularnih izraza, v. poglavlje 9, se izdvaja niz od početka stringa koji sadrži alfanumeričke karaktere i srednju crtu, linije 6-8. Ukoliko regularni izraz ne može da izdvoji takav niz, prikazaće se stranica *Greška*.

Listing 170 - Ruter aplikacije, *router.php*

```
1 <?php
2 $request = ltrim($_SERVER['REQUEST_URI'], '/');
3 if ($request == '')
4  $stranica = 'pocetna';
5 else
6  $stranica =
7   preg_match('#^([\w-]+)#', $request, $m)
8   ? $m[0] : 'greska';
9 switch ($stranica) {
10  case 'rss' :
11   define('DISABLE_HEADER', true);
12   define('DISABLE_FOOTER', true);
13  case 'pocetna' :
14  case 'kontakt' :
15  case 'greska' :
16   $modul_ime = DIR_INCLUDES . "stranica_{$stranica}.php";
17   break;
18  case 'login' :
19  case 'logout' :
20   $modul_ime = DIR_INCLUDES . "stranica_login.php";
21   break;
22  default:
23   $stranice_kategorije = array_map('strtolower', $kategorije);
24   if (in_array($stranica, $stranice_kategorije))
25    $modul_ime = DIR_INCLUDES . "stranica_vesti.php";
26   else
27    $modul_ime = DIR_INCLUDES . "stranica_greska.php";
28   break;
29 }
30 ?>
```

Vrednost promenljive *$stranica* se napokon upoređuje sa predefinisanim vrednostima u *switch* bloku, linije 9-29, na osnovu čega se formira vrednost promenljive *$modul_ime*, koja će se potom iskoristiti za uključivanje odgovarajućeg modula u glavni tok aplikacije.

Listing 171 - HTML zaglavlje šablona, *header.php*

```
<html>
<head>
<link
  rel="stylesheet" type="text/css"
  href="<?= DIR_CSS ?>all.min.css"
>
<link
  rel="stylesheet" type="text/css"
  href="<?= DIR_CSS ?>style.css">
</head>
<body>
 <header>
  <login>
   <?php if (PRIJAVLJEN): ?>
    <a href="./logout">Odjavi se</a> |
    <a href="./rss">RSS</a>
   <?php else: ?>
    <a href="./login">Prijavi se</a>
   <?php endif; ?>
  </login>
 </header>
 <nav>
  <a href="/">Početna</a>
  <?php foreach($kategorije AS $k): ?>
   <a href="/<?= strtolower($k) ?>"><?= $k ?></a>
  <?php endforeach; ?>
  <a href="/kontakt">Kontakt</a>
 </nav>
 <stranica>
```

Listing 172 - HTML footer šablona, *footer.php*

```
</stranica>
<footer>
 &copy; <?= date('Y') ?> Univerzitet Singidunum
 <secondary>
  Veb programiranje |
  Miloš Dobrojević |
  mdobrojevic@singidunum.ac.rs
 </secondary>
</footer>
</body>
</html>
```

U slučaju da je izvršen poziv ka rss stranici, deklarišu se konstante *DISABLE_HEADER* i *DISABLE_FOOTER*, linije 11-12, koje potom u dokumentu *index.php* sprečavaju prikaz zaglavlja i futera veb stranice.

13.7.3 Formiranje veb stranice

Osnovna struktura formiranja veb stranice se može videti u dokumentu *index.php*, listing 165, linije 20-24:

- Ukoliko konstanta *DISABLE_HEADER* nije definisana, u tok aplikacije se uvodi dokument *header.php*, listing 171, koji se nalazi u direktorijumu *template*, sa zadatkom da obezbedi HTML zaglavlje veb stranice.
- Potom se uvodi glavni dokument modula, smešten u direktorijumu *includes*.
- Napokon, formira se HTML footer uvođenjem dokumenta *footer.php*, listing 172, lociranog u direktorijumu *template*, pod uslovom da konstanta *DISABLE_FOOTER* nije definisana.

13.8 MODULI

Aplikacija sadrži sledeće module, slika 63:

- *Početna*, prikaz početne strane
- *Vesti,* prikaz vesti u sekcijama *Vesti, Studenti* i *Projekti*
- *Kontakt*, prikaz kontakt obrasca
- *RSS*, prikaz RSS liste sa najnovijim člancima
- *Login*, prijavljivanje korisnika
- *Greška*, prikaz stranice sa greškom

Generički raspored i sistem imenovanja fajlova jednog modula može okvirno da se prikaže kroz sledeći primer:

- includes/stranica_vesti.php, model modula vesti.
- *includes/funkcije_vesti.php*, funkcije razvijene za modul *vesti* koje će se koristiti u modelu ili šablonu.
- public/css/style_vesti.css, stilizovanje modula vesti
- *public/js/script_vesti.js*, kontrola na klijentskoj strani
- *template/stranica_vesti.php*, šablon za prikaz stranice

Struktura fajlova modula se može razlikovati od prikazane, shodno konkretnim potrebama projekta.

Listing 173 - Model *početne* strane, direktorijum *includes, stranica_pocetna.php*

```php
<?php
$vesti = [];
$sql = "SELECT *
  FROM `articles`
  ORDER BY `article_date` DESC
  LIMIT 3";
$rezultat = $db->query($sql);
while ($red = $rezultat->fetch_assoc()) {
 $vesti[] = $red;
}
include(DIR_TEMPLATE . 'stranica_pocetna.php');
?>
```

Listing 174 - Šablon početne strane, direktorijum *template, stranica_pocetna.php*

```
<pocetna>
  <h2>Najnovije vesti</h2>
  <?php if (empty($vesti)): ?>
    Trenutno nema objavljenih vesti
  <?php else: ?>
    <?php foreach($vesti AS $vest): ?>
    <vest>
      <?php
        $url = sprintf(
          '%s/%d',
          strtolower($kategorije[$vest['article_category_id']]),
          $vest['article_id']
        );
      ?>
      <a href="./<?= $url ?>"><?= $vest['article_title'] ?></a>
      <datum><?= $vest['article_date'] ?></datum>
    </vest>
    <?php endforeach; ?>
  <?php endif; ?>
</pocetna>
```

13.8.1 Početna strana

Da bi se prikazala početna strana, nakon što ruter ispravno protumači zahtev korisnika, iz glavnog dokumenta *index.php* u tok aplikacije će se uključiti dokument *stranica_pocetna.php* koji se nalazi u direktorijumu *includes*, listing 173.

Kako je već prikazano na slici 69, na početnoj strani se prikazuju tri najnovije vesti. Da bi se to realizovalo, neophodno je formirati i izvršiti odgovarajući SQL upit, linije 3-7, a potom kroz petlju očitati rezultate upita i smestiti ih kao niz u promenljivu *$vesti*, linije 8-9.

Time su stečeni uslovi da se u liniji 11 pređe u šablon modula, dokument *stranica_pocetna.php* koji se nalazi u direktorijumu *template*, listing 174, gde će se formirati odgovarajući HTML kod.

Početak i kraj dela stranice koji je generisao modul, označen je tagom *<pocetna>* ...*</pocetna>*. Ova tehnika je primenjena u svim modulima aplikacije, radi lakše selekcije elemenata stranice prilikom formiranja CSS ili JavaScript koda.

U slučaju da SQL upit nije pronašao nijedan članak, niz *$vesti* je prazan, i šablon će prikazati adekvatnu poruku, linije 3-4. Ukoliko su vesti pronađene, odgovarajući HTML kod će biti formiran kroz petlju, linije 6-18.

Radi lakše selekcije elemenata (CSS, JS), HTML kod za svaku vest ograničen je tagom <*vest*>...</*vest*>.

13.8.2 Vesti

Modul *Vesti* kontroliše sekcije veb sajta *Vesti, Studenti* i *Projekti*, i ima najsloženiju strukturu.

Na listingu 175 je prikazan primer rešenja rutera za navedene sekcije sajta koje je ispravno, ali nedovoljno fleksibilno. U situaciji kada bi bilo potrebno da se izmeni struktura veb sajta dodavanjem ili uklanjanjem sekcije, neophodno bi bilo da se u skladu sa tim izmeni i ruter.

Listing 175 - Kruta struktura rutera

```
...
case 'vesti' :
case 'studenti' :
case 'projekti' :
 $modul_ime = DIR_INCLUDES . "stranica_vesti.php";
 break;
...
```

Kako je već opisano u poglavlju 13.5, struktura sekcija veb sajta se nalazi u bazi podataka, tabela *categories*. Sadržaj ove tabele se očitava u dokumentu *index.php* i smešta u promenljivu *$kategorije*, linija 18 na listingu 165.

Da bi aplikacija bila fleksibilnija i omogućila dodavanje ili uklanjanje određenih sekcija bez potrebe da se vrše izmene u ruteru, u switch bloku rutera za svaku vrednost promenljive *$stranica* koja nije konkretno navedena proverava se da li se nalazi u nizu *$kategorije*, linije 23-25 na listingu 170, i u tom slučaju uključuje u tok programa fajl *stranica_vesti.php*. Transformacija imena kategorija u mala slova je izvršena zato što će i semantičke adrese biti pisane malim slovima.

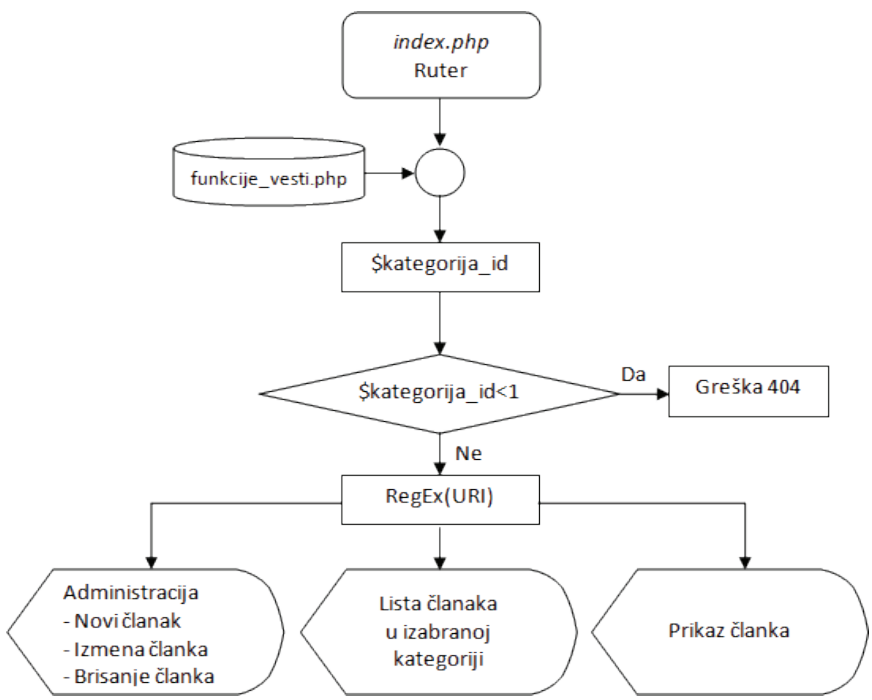
13.8.2.1 Struktura fajlova. Ruter.

Modul *Vesti* čine sledeći dokumenti:

```
includes
├─ funkcije_vesti.php
├─ stranica_vesti.php
├─ stranica_vesti_admin.php
├─ stranica_vesti_spisak.php
└─ stranica_vesti_vest.php
template
├─ stranica_vest.php
├─ stranica_vesti.php
└─ stranica_vesti_edit.php
```

Fajl *stranica_vesti.php* u direktorijumu *includes*, listing 176, je ulazna tačka modula, i može se predstaviti algoritmom prikazanim na slici 64:

Slika 64 - Algoritam modula Vesti, direktorijum *includes, fajl stranica_vesti.php*



- U tok programa se uključuju zajedničke funkcije koje se nalaze u fajlu *funkcije_vesti.php*, linija 2.
- Na osnovu vrednosti promenljive *$stranica*, određene u ruteru, pronalazi se odgovarajuća numerička vrednost koja odgovara koloni *category_id* u tabeli *categories*, linije 3-7. Ova vrednost se dodeljuje promenljivoj *$kategorija_id*.
- Vrednost promenljive *$kategorija_id* manja od 1 ukazuje na grešku, linije 8-9.
- Provera vrednosti promenljive *$stranica* upotrebom regularnih izraza, linije 10-19. U skladu sa dobijenim rezultatom provere, rutiranje ka odgovarajućim segmentima aplikacije:
- Administracija, odnosno menadžment članaka
- Prikaz liste članaka u izabranoj kategoriji
- Prikaz određenog članka

Listing 176 - Ruter modula vesti, direktorijum *includes, stranica_vesti.php*

```
1  <?php
2  include(DIR_INCLUDES . 'funkcije_vesti.php');
3  $kategorija_id =
4    (int)array_search(
5      $stranica,
6      $stranice_kategorije)
7    );
8  if ($kategorija_id < 1)
9   redirect(STRANICA_GRESKA);
10 if (preg_match(
11   "#^{$stranica}/(submit|edit|delete)(?:/(\d+))*$#",
12   $request,
13   $m)
14 )
15  include(DIR_INCLUDES . 'stranica_vesti_admin.php');
16 else if (preg_match("#^{$stranica}/(\d+)$#", $request, $m))
17  include(DIR_INCLUDES . 'stranica_vesti_vest.php');
18 else
19  include(DIR_INCLUDES . 'stranica_vesti_spisak.php');
20 ?>
```

Listing 177 - Funkcije modula vesti, direktorijum *includes, funkcije_vesti.php*

```php
<?php
function vesti_vreme($vreme) {
   $vreme = strtotime($vreme);
   $vreme = date('d.m.Y H:i', $vreme);
   return $vreme;
}
?>
```

13.8.2.2 Funkcije modula

Da bi se izbeglo nepotrebno ponavljanje koda, zajedničke funkcije na nivou modula mogu da se grupišu u dokument *funkcije_vesti.php*, listing 177, i potom pozivaju po potrebi.

Dokument sadrži samo jednu funkciju, namenjenu ispisu datuma objave članka u formatu koji se koristi na našim prostorima:

```
dan.mesec.godina čas:minuti
```

Slika 65 - Sekcija Vesti, spisak objavljenih članaka (*Administrator*)

Listing 178 - Spisak vesti, direktorijum *includes, stranica_vesti_spisak.php*

```
1 <?php
2 $vesti = [];
3 $sql = "SELECT *
4         FROM `articles`
5         WHERE `article_category_id`={$kategorija_id}
6         ORDER BY `article_date` DESC";
7 $rezultat = $db->query($sql);
8 while ($vest = $rezultat->fetch_assoc())
9   $vesti[] = $vest;
10 include(DIR_TEMPLATE . 'stranica_vesti.php');
11 ?>
```

13.8.2.3 Spisak vesti

Kod za prikaz spiska vesti, odnosno članaka u izabranoj sekciji sajta se nalazi u fajlovima

- *includes/stranica_vesti_spisak.php*
- *template/stranica_vesti.php*

U fajlu *stranica_vesti_spisak.php*, direktorijum *includes,* listing 178, SQL upit iz tabele *articles* čita članke koji pripadaju kategoriji označenoj promenljivom *$kategorija_id,* i sortira ih prema datumu objave u opadajućem redosledu, linije 3-7. Očitani članci se smeštaju u niz *$vesti*.

Fajl *stranica_vesti.php* u direktorijumu *template*, listing 179 formira HTML kod veb stranice koja će prikazati spisak vesti. Slično kao i na početnoj strani, svaka vest je označena tagom <*vest*>...</*vest*>, linije 21-37, u okviru koga se nalazi veb link sa naslovom vesti, linije 23-25, datum kada je vest objavljena, linija 35, i kratak opis vesti, linija 36.

Prilikom formiranja stranice mora se voditi računa o tome da li je korisnik prijavljen, a razlika se može uočiti poređenjem izgleda veb stranice na slikama 60 (*anoniman* korisnik) i 65 (*administrator*). Kao indikator statusa korisnika se koristi konstanta *PRIJAVLJEN*, linija 1

Slika 66 - Menadžment članaka, simbolički prikaz funkcionalnost

FA Ikona	*FA CSS klasa*[28]	*Značenje*	*Linije koda* [29]
	fa-plus-circle	Dodavanje novog članka	9-11
	fa-edit	Izmena postojećeg članka	27-29
	fa-trash	Brisanje postojećeg članka	30-32

Funkcionalnostima za uređivanje sadržaja navedenim u tabeli 19, administrator pristupa putem simboličkih linkova, slike 65 i 66. Za prikaz simbola su iskorišćene *Font Awesome ikone*.

Listing 179 - Spisak vesti, šablon, direktorijum *template, stranica_vesti.php*

```
<?php if (PRIJAVLJEN): ?>
<script type="text/javascript" src="<?= DIR_JS ?>jquery.js"></script>
<script type="text/javascript" src="<?= DIR_JS ?>script.js"></script>
<?php endif; ?>
<vesti>
<h1>
  <?= $kategorije[$kategorija_id] ?>
  <?php if (PRIJAVLJEN): ?>
  <a href="/<?= strtolower($kategorije[$kategorija_id]) ?>/submit">
    <i class="fas fa-plus-circle"></i>
  </a>
  <?php endif; ?>
</h1>
<?php if (empty($vesti)): ?>
  Trenutno nema objavljenih vesti
<?php else: ?>
```

28 https://fontawesome.com/download, u vreme pisanja udžbenika aktuelna verzija 6
29 Listing 179

```
17  <?php foreach($vesti AS $vest): ?>
18    <?php $kategorija = strtolower(
19            $kategorije[$vest['article_category_id']]
20          ) ?>
21    <vest>
22      <h3>
23        <a href="/<?= $kategorija ?>/<?= $vest['article_id'] ?>">
24          <?= $vest['article_title'] ?>
25        </a>
26        <?php if (PRIJAVLJEN): ?>
27          <a href="/<?= $kategorija ?>/edit/<?= $vest['article_id'] ?>">
28            <i class="fa fa-edit"></i>
29          </a>
30          <a href="/<?= $kategorija ?>/delete/<?= $vest['article_id'] ?>">
31            <i class="fa fa-trash"></i>
32          </a>
33        <?php endif; ?>
34      </h3>
35      <datum><?= vesti_vreme($vest['article_date']) ?></datum>
36      <kratko><?= $vest['article_short'] ?></kratko>
37    </vest>
38  <?php endforeach; ?>
39 <?php endif; ?>
40 </vesti>
```

Slika 67 - Zahtev za potvrdu akcije brisanja članka

13.8.2.4 Kontrola interakcije između korisnika i veb stranice

Interakcija između administratora i ikone za brisanje članka mora da sadrži međukorak u kome administrator mora da potvrdi da zaista želi da realizuje svoju nameru, za slučaj da je na ikonu slučajno kliknuto, slika 67.

Listing 180 - Kontrola interakcije sa linkovima za brisanje vesti

```
$(document).ready(function() {
  $("a[href*='delete']").click(function(){
    var clanak = $(this).parent().find('a').text();
    if (!confirm(
            'Da li želite da obrišete članak "'+clanak+'"?'
          )
    ) {
      return false;
    }
  })
});
```

Ovaj korak je realizovan na klijentskoj strani aplikacije, v. poglavlje 11, implementacijom *jQuery* biblioteke[30] i jednostavne JavaScript skripte, listing 180. Oba fajla, *jquery.js* i *script.js* se nalaze u direktorijumu *public/js*, kako je prikazano na slici 63.

jQuery može da se integriše u veb stranicu na dva načina:

- Kao hostovano rešenje, npr.

```
<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">
</script>
```

- Kod se može preuzeti i postaviti lokalno na veb server sa ostatkom projekta. Dostupan je u različitim varijantama, u ovom slučaju je iskorišćena varijanta za produkciono okruženje u kojoj je kod transformisan uklanjanjem preloma linija i tabulatora.

30 https://jquery.com/download/, u vreme pisanja udžbenika aktuelna verzija 3.6.0

Na stranici se nalazi više članaka, kao što je prikazano na slici 65. Zadatak skripte prikazane na listingu 180 je da osluškuje događaj *click* na svim elementima DOM stabla koje čini HTML tag a i koji u adresi, atribut *href*, sadrži string *delete*, linija 2.

Kada se desi osluškivani događaj, skripta pokušava da pronađe naslov članka, odnosno tekst linka na koji je kliknuto, linija 3. Mehanizam funkcionisanja ove linije se može objasniti na sledećem primeru.

Listing 181 - HTML model naslova vesti sa ikonama za menadžment članka

```
1 <h3>
2   <a href="/vesti/1">Naslov vesti</a>
3   <a href="/vesti/edit/1"><i class="fa fa-edit"></i></a>
4   <a href="/vesti/delete/1"><i class="fa fa-trash"></i></a>
5 </h3>
```

HTML kod naslova vesti je izolovan na listingu 181. Mišem je kliknuto na ikonu za brisanje članka, linija 4, i *jQuery* selektor *$(this)* se odnosi na nju. Njegov matični element (engl. *parent*) će biti element *h3*, u okviru koga je potrebno da se pronađe element sa tagom *a*, link. Napokon, izoluje se tekst koji sadrži link, a to je naslov članka, i on se čuva u promenljivoj *clanak*. Na kraju, od korisnika se putem JavaScript *confirm box-a*, v. poglavlje 11.2.1, traži saglasnost da li zaista želi da obriše članak čije se ime nalazi u promenljivoj *clanak*. U slučaju odričnog odgovora klikom na dugme *Cancel*, akcija se poništava, dok će u slučaju potvrdnog odgovora klikom na dugme *Ok* veb brauzer preći na link za brisanje članka.

13.8.2.5 Prikaz vesti

Kada se na početnoj strani, ili u nekoj od sekcija koje prikazuju spisak vesti klikne na naslov članka, u veb brauzeru će se otvoriti stranica sa prikazom cele vesti, slika 61. Prikaz vesti kontrolišu dva fajla:

- includes/stranica_vesti_vest.php
- template/stranica_vest.php

URL stranice za prikaz vesti se zadaje u sledećem obliku, tabela 18:

```
/vesti/1
```

koji aplikaciji daje signal da je potrebno da se prikaže članak u kategoriji *vesti* čiji je *article_id jednak 1*.

Kako je ranije opisano, ruter aplikacije prepoznaje string *vesti* kao validnu kategoriju i aktivira modul vesti uključujući u tok programa fajl *stranica_vesti.php*, listing 176. Regularni izraz poredi pozvani URI sa zadatim šablonom, u promenljivu *$m* skladišti *id* članka, linija 16 i u sledećoj liniji 17 koda poziva odgovarajući segment modula - fajl *stranica_vesti_vest.php* prikazan na listingu 182.

U dokumentu *stranica_vesti_vest.php,* SQL upit se formira tako da se traži članak čiji je *article_id* određen u prethodnom koraku, ali i koji pripada odgovarajućoj kategoriji, linije 3-8. Ukoliko takav članak ne postoji u bazi podataka, vrši se redirekcija na stranicu 404.

Ukoliko je članak pronađen u bazi podataka, formiraju se navigacione mrvice (engl. *breadcrumbs*, v. poglavlje 12.3.1), linije 13-16.

Listing 182 - Prikaz članka, direktorijum *includes, stranica_vesti_vest.php*

```
1  <?php
2  $article_id = $m[1];
3  $sql = "SELECT *
4          FROM `articles`
5          WHERE
6            `article_category_id`={$kategorija_id}
7            AND `article_id`={$article_id}
8            LIMIT 1";
9  $rezultat = $db->query($sql);
10 $vest = $rezultat->fetch_assoc();
11 if (empty($vest))
12     redirect(STRANICA_GRESKA);
13 $breadcrumbs = breadcrumbs([
14   '/' => 'Početna',
15   "/{$kategorije[$kategorija_id]}" => $kategorije[$kategorija_id]
16 ]);
17 include(DIR_TEMPLATE . 'stranica_vest.php');
18 ?>
```

Napokon, u tok programa se uključuje fajl *stranica_vest.php* koji se nalazi u direktorijumu *template* radi formiranja HTML koda stranice, listing 183.

Listing 183 - Prikaz članka, šablon, direktorijum *template, stranica_vest.php*

```
<?= $breadcrumbs ?>
<vest>
  <h1><?= $vest['article_title'] ?></h1>
  <datum><?= $vest['article_date'] ?></datum>
  <clanak><?= $vest['article_full'] ?></clanak>
</vest>
```

13.8.2.6 Menadžment vesti

Menadžment, odnosno administracija vesti, u skladu sa tabelom 19 podrazumeva sledeće operacije:

- unos novog članka, npr. */vesti/submit*
- izmena postojećeg članka, npr. */vesti/edit/1*
- brisanje postojećeg članka, npr. */vesti/delete/1*

Nakon što ruter u direktorijumu *includes* aktivira fajl *stranica_vesti.php*, listing 176, pomoću regularnih izraza se proverava da li URI upućuje na neku od ovih operacija, linije 10-14, i u tom slučaju uključuje fajl *stranica_vesti_admin.php* prikazan na listingu 184.

Listing 184 - Administracija vesti, direktorijum *includes, stranica_vesti_admin.php*

```
<?php
if (!PRIJAVLJEN)
    redirect(STRANICA_GRESKA);
$article_action = $m[1];
$article_id = @$m[2];
switch ($article_action) {
  case 'submit' :
  case 'edit' :
    if ($_POST) {
      $test = function() use ($article_action, $article_id, $db, $kategorije) {
```

```
    $post = $_POST;
    array_walk($post, function(&$v, &$k) use ($db) {
      $v = trim($v);
      $v = $db->real_escape_string($v);
    });
    extract($post);
    $greska = [];
    $kategorija = (int)$kategorija;
    if ($naslov_clanka == '') $greska[] = 'Unesite naslov';
    if ($kratak_opis == '') $greska[] = 'Unesite kratak opis';
    if ($pun_tekst == '') $greska[] = 'Unesite pun tekst';
    if (!empty($greska))
      return implode("<br>", $greska);
    switch ($article_action) {
      case 'submit' :
        $sql = "INSERT INTO `articles` (
                  `article_category_id`,
                  `article_title`,
                  `article_short`,
                  `article_full`
                ) VALUES (
                  {$kategorija},
                  '{$naslov_clanka}',
                  '{$kratak_opis}',
                  '{$pun_tekst}'
                )";
        break;
      case 'edit' :
        $sql = "UPDATE `articles` SET
                  `article_category_id`={$kategorija},
                  `article_title`='{$naslov_clanka}',
                  `article_short`='{$kratak_opis}',
                  `article_full`='{$pun_tekst}'
                WHERE `article_id`={$article_id}
                LIMIT 1";
        break;
    }
    $db->query($sql);
    $redirect = sprintf("/%s", strtolower($kategorije[$kategorija]));
    redirect($redirect);
  };
  $greska = $test();
  $vest = [
    'article_category_id' => $_POST['kategorija'],
    'article_title' => $_POST['naslov_clanka'],
    'article_short' => $_POST['kratak_opis'],
    'article_full' => $_POST['pun_tekst']
```

```
58        ];
59      }
60      switch ($article_action) {
61        case 'edit' :
62          $sql = "SELECT *
63                  FROM `articles`
64                  WHERE
65                    `article_category_id`={$kategorija_id}
66                    AND `article_id`={$article_id}
67                    LIMIT 1";
68          $rezultat = $db->query($sql);
69          $vest = $rezultat->fetch_assoc();
70          if (empty($vest))
71              redirect(STRANICA_GRESKA);
72          break;
73      }
74      $breadcrumbs = breadcrumbs([
75        '/' => 'Početna',
76        "/{$kategorije[$kategorija_id]}" => $kategorije[$kategorija_id]
77      ]);
78      include(DIR_TEMPLATE . 'stranica_vesti_edit.php');
79      break;
80    case 'delete' :
81      $sql = "SELECT `article_category_id`
82              FROM `articles`
83              WHERE
84                `article_id`={$article_id}
85                LIMIT 1";
86      $rezultat = $db->query($sql);
87      $vest = $rezultat->fetch_assoc();
88      $kategorija = $vest['article_category_id'];
89      $sql = "DELETE FROM `articles`
90              WHERE `article_id`={$article_id}
91              LIMIT 1";
92      $db->query($sql);
93      $redirect = sprintf("/%s", strtolower($kategorije[$kategorija]));
94      redirect($redirect);
95      break;
96  }
97  ?>
```

Kako je u pitanju deo aplikacije koji sadrži funkcionalnosti kojima samo administrator ima pristup, prvo se proverava da li je korisnik prijavljen, linija 2. U slučaju da anoniman korisnik pokušava da pristupi ovom delu aplikacije, izvršiće se redirekcija na stranicu 404, linija 3.

Listing 185 - Šablon administracije vesti, direktorijum *template, stranica_vesti_edit.php*

```
1  <vest>
2  <h1>Izmena članka</h1>
3  <?php include(DIR_TEMPLATE . 'greska.php'); ?>
4  <form method="post">
5    Naslov članka
6    <input name="naslov_clanka" value="<?= @$vest['article_title'] ?>">
7    Kategorija
8    <select name="kategorija">
9      <?php foreach($kategorije AS $k => $v): ?>
10       <option
11         value="<?= $k ?>"
12         <?= $k == @$vest['article_category_id'] ? 'selected' : '' ?>
13       >
14         <?= $v ?>
15       </option>
16     <?php endforeach; ?>
17   </select>
18   Kratak opis
19   <input name="kratak_opis" value="<?= @$vest['article_short'] ?>">
20   Pun tekst
21   <textarea name="pun_tekst"><?= @$vest['article_full'] ?></textarea>
22   <button type="submit">Pošalji</button>
23 </form>
24 </vest>
```

U sledećem koraku se iz rezultata provere regularnim izrazima, promenljiva *$m*, izdvajaju dve vrednosti koje su neophodne za dalji rad:

- *akcija, promenljiva $article_action*, linija 4, može imati vrednosti *submit, edit* ili *delete.*
- *id članka, promenljiva $article_id*, linija 5, je celobrojna vrednost.

U *switch* bloku, linije 6-96, se vrši provera vrednosti promenljive *$article_action* i u skladu sa tim izvršava željena aktivnost:

- *dodavanje (submit)* ili *izmena (edit*) članka, linije 7-79
- *brisanje (delete)* članka, linije 80-95

Slika 68 - Veb obrazac za unos novog i izmenu postojećeg članka

Članak se ubacuje ili menja putem veb obrasca prikazanog na slici 68. Kod za generisanje ove stranice se nalazi na listingu 185. Radi efikasnije strukture koda, isti veb obrazac se koristi u oba slučaja, i za unos, i za izmenu članka.

Kada se završi sa unosom podataka i klikne na dugme pošalji, podaci se *POST* metodom šalju ka veb serveru. Aplikacija proverava da li ima podataka u superglobalnoj promenljivoj *$_POST*, linija 9 na listingu 184, i ukoliko je ovaj uslov ispunjen, inicira sekvencu provere podataka, linija 52, realizovanu u obliku anonimne funkcije, linije 10-51. Ukoliko su podaci korektni, izvršava odgovarajući SQL upit i vrši redirekciju na stranicu sa spiskom vesti u odgovarajućoj kategoriji. U suprotnom, poruke o eventualnim greškama će biti isporučene u promenljivu *$greška*. Veb obrazac će ponovo biti prikazan, ali i poruke o greškama kako bi korisnik znao šta treba da ispravi.

U slučaju da je aktiviran link za brisanje članka, linija 80, prvo se iz baze podataka očitava kategorija članka, linije 81-88. Potom se članak uklanja iz baze podataka, linije 89-92. Napokon, formira se adresa i izvršava redirekcija, linije 93-94.

13.8.3 Kontakt stranica

Kontakt strana prikazuje veb obrazac u koji korisnik može da unese svoje ime, email adresu i kratku poruku, slika 62. Kada klikne na dugme *Pošalji*, veb aplikacija preuzima unete podatke, proverava ih i ukoliko su korektno uneti šalje email administratoru, dok u suprotnom ponovo prikazuje veb obrazac i opis greške kako bi korisnik znao šta treba da ispravi, slika 69.

Model kontakt strane se nalazi u direktorijumu includes, dokument *stranica_kontakt.php,* listing 186. Zadatak modula je da prikaže kontakt stranicu i da prihvati podatke koje je korisnik uneo u veb obrazac i da ih kao email pošalje administratoru sajta.

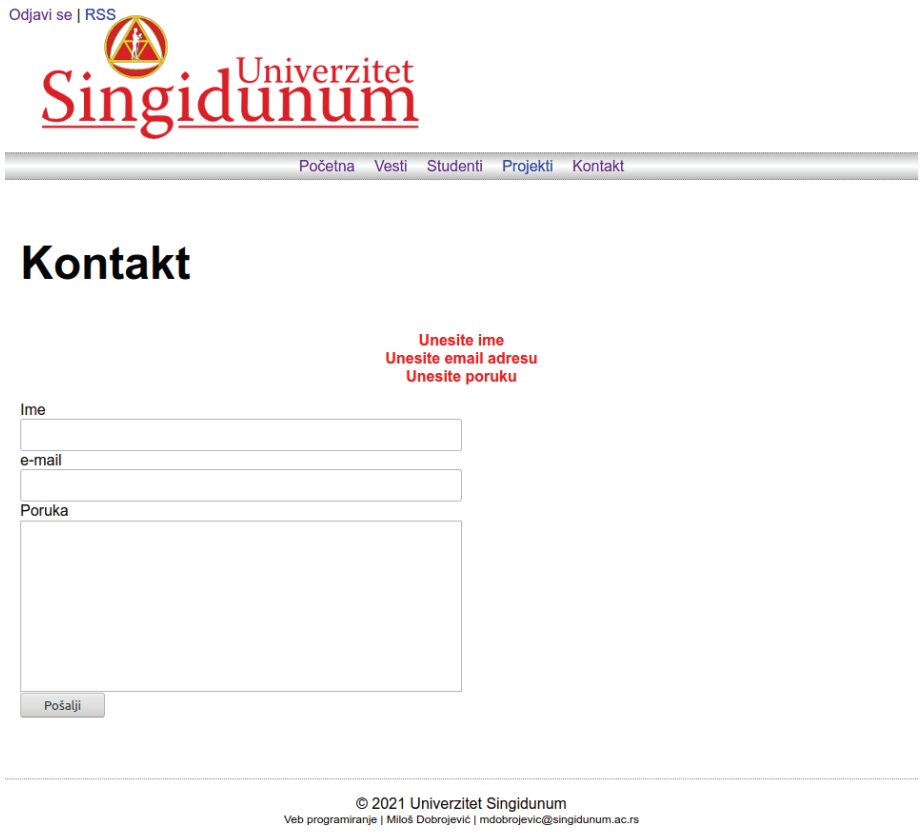
Ukoliko su podaci poslati POST metodom, izvršiće se testiranje podataka, linije 2-27. PHP funkcija *extract()*, linija 4, generiše promenljive čija imena odgovaraju indeksima u nizu *$_POST*. U slučaju zloupotrebe POST zahteva, upotreba ove funkcije u glavnom toku programa bi mogla da ugrozi integritet aplikacije, te je iz tog razloga smeštena u anonimnu funkciju, linije 3-25.

Ako je korisnik propustio da unese neophodne podatke, odgovarajuće poruke će biti smeštene u niz *$greska*, u suprotnom će aplikacija pokušati da pošalje mail, linije 13-20. U slučaju neuspeha, poruka će takođe biti smeštena u niz *$greška*.

Listing 186 - Model *kontakt* strane, direktorijum *includes, stranica_kontakt.php*

```
<?php
if ($_POST) {
  $test = function() {
    extract($_POST);
    $greska = [];
    if ($ime == '')
      $greska[] = 'Unesite ime';
    if ($email == '')
      $greska[] = 'Unesite email adresu';
    if ($poruka == '')
      $greska[] = 'Unesite poruku';
    if (empty($greska)) {
      $poslato = mail(
        "moja@mailadresa.com",
        'Poruka sa sajta',
        "{$poruka}\n\n
         {$ime}\n
         {$email}",
        "From: {$email}\r\n"
      );
      if (!$poslato)
        $greska[] = 'Poruka nije poslata';
    }
    return implode('<br>', $greska);
  };
  $greska = $test();
}
include(DIR_TEMPLATE . 'stranica_kontakt.php');
?>
```

Slika 69 - Unos podataka u kontakt obrazac, prijava greške



Napokon, niz *$greska* se pretvara u string sa HTML tagom <*br*> kao razdelnikom i vraća kao rezultat rada funkcije, linija 24. Šablon za prikaz veb obrasca se poziva u liniji 28, listing 187.

HTML sadržaj kontakt stranice je označen HTML tagom <*kontakt*>...</*kontakt*>. Ispod naslova stranice se po potrebi prikazuje poruka o uočenim greškama prilikom unosa podataka, na osnovu koje korisnik može da izvrši neophodne ispravke i pokuša ponovo sa slanjem podataka. Umesto da se pozove funkcija koja će ubaciti odgovarajući šablon, kao što je to slučaj sa prikazom navigacionih mrvica (engl. *breadcrumbs*), ovde je to direktno urađeno, linija 3.

Listing 187 - Šablon kontakt strane, direktorijum *template, stranica_kontakt.php*

```
<kontakt>
<h1>Kontakt</h1>
<?php include(DIR_TEMPLATE . 'greska.php'); ?>
<form method="post">
  Ime
  <input name="ime">
  e-mail
  <input name="email" type="email">
  Poruka
  <textarea name="poruka"></textarea>
  <button type="submit">Pošalji</button>
</form>
</kontakt>
```

Listing 188 - Šablon za ispis greške na veb stranici

```
<?php if (@$greska): ?>
  <greska>
    <?= $greska ?>
  </greska>
<?php endif; ?>
```

13.8.4 Login

Korisnik može da se prijavi (uloguje, engl. *login*) preko odgovarajućeg obrasca prikazanog na slici 70 u koji unosi svoje korisničko ime i lozinku.

U slučaju da su podaci ispravno uneti, korisnik stiče pravo pristupa funkcionalnostima aplikacije koje nisu dostupne anonimnom korisniku, u ovom slučaju to su uređivačka prava za menadžment vesti.

Ukoliko se pogreši prilikom unosa korisničkog imena ili lozinke, aplikacija će ponovo prikazati veb obrazac za prijavljivanje, sa odgovarajućom porukom, slika 71:

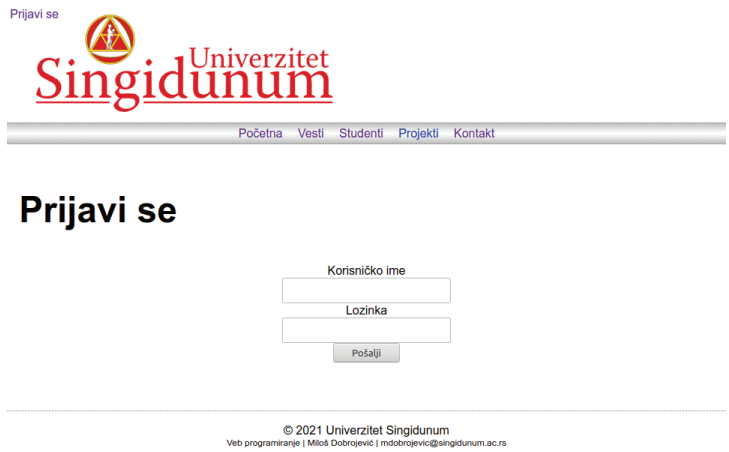
```
Podaci nisu dobri
```

Ova poruka očigledno nije precizna, tj. ne pruža informaciju o tome da li je pogrešno uneto korisničko ime ili lozinka. Osoba koja poseduje korisnički nalog će lako uočiti i ispraviti grešku, dok potencijalni napadač neće dobiti povratnu informaciju o tome da li uneto korisničko ime postoji ili ne.

Modul za prijavu i odjavu korisnika se sastoji iz dva fajla:

- *includes/stranica_login.php*
- *template/stranica_login.php*

Slika 70 - Login obrazac



Slika 71 - Obaveštenje o pogrešno unešenim podacima prilikom prijave

Listing 189 - Model *login* strane, direktorijum *includes, stranica_login.php*

```php
<?php
switch($request) {
  case 'login' :
      if (PRIJAVLJEN)
        redirect('./greska');
      if ($_POST) {
        $login_test = function () {
          extract($_POST);
          if (
            $username == 'admin'
            && $password == 'admin'
          ) {
            $_SESSION['prijavljen'] = true;
            redirect('./');
          } else
            return 'Podaci nisu dobri.';
        };
        $greska = $login_test();
      }
      include(DIR_TEMPLATE . 'stranica_login.php');
    break;
  case 'logout' :
    $_SESSION['prijavljen'] = 0;
    redirect('./');
    break;
}
?>
```

Listing 190 - Šablon *login* strane, direktorijum *template, stranica_login.php*

```
1  <login>
2    <h1>Prijavi se</h1>
3    <?php include(DIR_TEMPLATE . 'greska.php'); ?>
4    <form method="post">
5      Korisničko ime
6      <input name="username" type="text">
7      Lozinka
8      <input name="password" type="password">
9      <button type="submit">Pošalji</button>
10   </form>
11 </login>
```

Nakon što ruter aplikacije u tok programa uključi fajl *stranica_login.php* u direktorijumu *includes*, listing 189, sadržaj promenljive *$request* se upoređuje sa predefinisanim *URI* adresama *login* (prijava), linija 3 i *logout* (odjava), linija 22.

13.8.4.1 Prijava korisnika

U slučaju da je modul aktiviran radi prijave a da je korisnik već prijavljen, vrši se redirekcija na stranicu greška, linije 4-5. U suprotnom proverava se da li su podaci za prijavljivanje već poslati, linije 6-19.

Provera poslatih podataka se izvršava u okviru anonimne funkcije, linije 7-17, zato što je korišćena PHP funkcija *extract()* koja ne bi smela da se koristi u glavnom toku programa kako je već opisano u tački 3.5. Ukoliko su podaci korektno uneti, u superglobalnu promenljivu *$_SESSION* se unosi ključ *prijavljen* sa vrednošću *1* i potom se vrši redirekcija na početnu stranu veb sajta, linije 9-14. Ukoliko uneti podaci nisu tačni, ponovo će se prikazati veb obrazac za prijavu sa odgovarajućom porukom, slika 71.

Zbog prirode aplikacije, dovoljna je samo jedna osoba za administraciju objavljenog sadržaja. Radi optimizacije koda i uštede radnih časova prilikom razvoja aplikacije, poslati podaci za korisničko ime i lozinku se direktno porede sa predefinisanim vrednostima, linije 10-11.

U slučaju kada je potrebno da aplikacija ima veći broj korisnika, i/ili složeniju strukturu korisničkih naloga, v. 12.3.2, podaci za pristup će se porediti sa podacima o korisnicima koji se čuvaju u posebnoj tabeli u bazi podataka. Ovakav scenario bi zahtevao izradu dodatnog modula za menadžment korisničkih naloga.

13.8.4.2 Odjava korisnika

Kada prijavljen korisnik klikne na link za odjavu, koji se prikazuje u zaglavlju svake veb stranice, slike 65, 68 i 69, vrednost ključa *prijavljen* u superglobalnoj promenljivoj *$_SESSION* se postavlja na *0*, linije 22-25, i vrši se redirekcija na početnu stranu.

13.8.5 RSS

RSS feed je *XML* dokument, v. poglavlje 8.1, koji generišu veb sajtovi na kojima se često menja sadržaj, kao što su medijski portali, sajtovi za hosting slika i video klipova, forumi i sl.

Dokument sadrži spisak najnovijih članaka, pri čemu je članak obično predstavljen naslovom, linkom, sličicom (engl. *thumbnail*), datumom objave i kratkim opisom. Periodičnim praćenjem takvih dokumenata putem namenski napravljene aplikacije, korisnik može na jednom mestu da ima jedinstveni presek novosti iz više različitih izvora.

Modul za generisanje RSS feed-a čine dva fajla:

- *includes/stranica_rss.php*
- *template/stranica_rss.php*

Pozivom odgovarajuće adrese, ruter aplikacije će u glavni tok programa uključiti fajl *stranica_rss.php* koji se nalazi u direktorijumu *includes*, listing 191.

Listing 191 - Model *RSS feed*-a, direktorijum *includes, stranica_rss.php*

```
<?php
include(DIR_INCLUDES . 'funkcije_rss.php');
$vesti = [];
$sql = "SELECT *
        FROM `articles`
        ORDER BY `article_date` DESC
        LIMIT 50";
$rezultat = $db->query($sql);
while ($vest = $rezultat->fetch_assoc())
  $vesti[] = $vest;
include(DIR_TEMPLATE . 'stranica_rss.php');
?>
```

Potom se uključuje fajl *funkcije_rss.php*, listing 192, koji sadrži funkciju *rss_vreme()*, čiji je zadatak da prikazuje vreme u odgovarajućem formatu.

Listing 192 - Funkcije modula *RSS*, direktorijum *includes, funkcije_rss.php*

```
<?php
function rss_vreme() {
  $vreme = date('D, d M Y H:i:s O');
  return $vreme;
}
?>
```

Listing 193 - Šablon *RSS* strane, direktorijum *template, stranica_rss.php*

```
1  <?php header('Content-Type: application/rss+xml; charset=utf-8'); ?>
2  <?xml version="1.0" encoding="UTF-8"?>
3  <rss
4    version="2.0"
5    xmlns:atom="http://www.w3.org/2005/Atom"
6    xmlns:content="http://purl.org/rss/1.0/modules/content/"
7    xmlns:slash="http://purl.org/rss/1.0/modules/slash/"
8  >
9    <channel>
10     <title>Univerzitet Singidunum</title>
11     <description>Najnovije vesti</description>
12     <image>
13       <url>http://singidunum.ac.rs/public/images/singidunum-logo.png</url>
14       <title>Univerzitet Singidunum</title>
15       <link>https://www.singidunum.ac.rs</link>
16     </image>
17     <pubDate><?= rss_vreme() ?></pubDate>
18     <generator>Veb programiranje - Primer projektnog zadatka</generator>
19     <link>https://www.singidunum.ac.rs/vesti</link>
20     <?php if (empty($vesti)): ?>
21         <item>
22           <title>Trenutno nema objavljenih vesti</title>
23         </item>
24     <?php else: ?>
25       <?php foreach($vesti AS $vest): ?>
26         <?php $kategorija = strtolower(
27                               $kategorije[$vest['article_category_id']]
28                             ) ?>
29         <item>
30             <title><?= $vest['article_title'] ?></title>
31             <description><?= $vest['article_short'] ?></description>
32             <pubDate><?= rss_vreme($vest['article_date']) ?></pubDate>
33             <link><?= HOST_URL ?>/<?= $kategorija ?>/<?= $vest['article_id'] ?></link>
34             <guid><?= HOST_URL ?>/<?= $kategorija ?>/<?= $vest['article_id'] ?></guid>
35             <content:encoded><![CDATA[<p><?= $vest['article_full'] ?></p>]]></content:encoded>
36         </item>
37       <?php endforeach; ?>
38     <?php endif; ?>
39   </channel>
40 </rss>
```

Na osnovu SQL upita, linije 4-8, iz tabele *articles* u bazi podataka se očitava najnovijih 50 članaka i smešta u promenljivu *$vesti* kao niz, linije 9-10.

Na kraju se u tok programa uključuje fajl *stranica_rss.php* u direktorijumu *template,* zadužen za formiranje same stranice.

13.8.6 Greška

Greška je pomoćni modul, na koji se vrši redirekcija u situacijama kada aplikacija ne može da isporuči traženi sadržaj, npr:

- Poziva se nepostojeća stranica, npr. */izvestaji/1*
- Prijavljeni korisnik želi da se prijavi
- Anonimni korisnik želi da pristupi funkcionalnosti aplikacije kojoj može da pristupi samo administrator

Listing 194 - Model strane *Greška*, direktorijum *includes, stranica_greska.php*

```
1 <?php
2 include(DIR_TEMPLATE . 'stranica_greska.php');
3 ?>
```

Listing 195 - Šablon strane *Greška*, direktorijum *template, stranica_greska.php*

```
1 <h1>Greška</h1>
2 Došlo je do greške.
```

Struktura modula je jednostavna i sastoji se iz dva fajla:

- *includes/stranica_greska.php*, listing 194. Jedini zadatak ovog fajla je da uključi u tok programa fajl *stranica_greska.php* u direktorijumu template.
- *template/stranica_greska.php*, listing 195. Ovaj dokument sadrži HTML kod za ispis poruke.

13.9 JAVNO DOSTUPNI DOKUMENTI

Javno dostupni dokumenti su fajlovi koji se isporučuju posetiocu veb stranice:

- Slike
- JavaScript dokumenti
- CSS dokumenti
- Veb fontovi

Na veb serveru su grupisani u odgovarajuće direktorijume koji se nalaze u matičnom direktorijumu *public*, slika 72[31].

Slika 72 - Struktura javno dostupnih dokumenata

13.9.1 JavaScript

U ovoj veb aplikaciji JavaScript se koristi u modulu *Vesti* za kontrolu interakcije korisnika sa veb stranicom, što je detaljno opisano u tački 13.8.2.4. Zato će u nastavku biti navedene smernice za dalje razmatranje.

Shodno nameni projekta i usvojenoj arhitekturi aplikacije, JavaScript aplikacija po svom obimu može biti izuzetno složena, pa čak i teža za razumevanje i održavanje od dela aplikacije koji funkcioniše na veb serveru.

31 Izdvojeni segment slike 63

To treba imati u vidu prilikom projektovanja sistema, i blagovremeno usvojiti optimalnu strukturu fajlova i koda kako bi se, koliko je moguće, olakšao kasniji rad na poslovima razvoja i održavanja sistema. JavaScript kod može da se segmentira na

- *Osnovni sloj,* koji je zajednički na nivou cele aplikacije, npr. *fajl script.js*
- *Module*, slično kako je učinjeno i sa PHP kodom, npr. *script_vesti.js*. Po potrebi, jedan modul može da ima i više JavaScript fajlova, pri čemu svaki fajl sadrži kod za određeni set međusobno srodnih funkcionalnosti.

Takođe je čest slučaj da se prilikom razvoja frontend segmenta aplikacije koriste javno dostupne JavaScript biblioteke ili radni okviri, kao npr. *jQuery* ili *AngularJS*, na koje se potom oslanja JavaScript kod aplikacije. Ovakva eksterna rešenja mogu biti integrisana u projekat putem HTML <*script*> taga, na dva načina:

- Referencom ka zvaničnoj adresi za preuzimanje.
- Preuzimanjem i postavljanjem na veb server zajedno sa aplikacijom, kao što je učinjeno sa *jQuery* fajlom, slika 72.

13.9.2 CSS

Arhitektura CSS koda je u međusobnoj korelaciji sa strukturom HTML i JavaScript koda, i bitno utiče na proces razvoja i održavanja veb aplikacije.

CSS kod može biti organizovan tako da se u celosti nalazi u jednom fajlu, ili može biti podeljen u više fajlova koji po strukturi odgovaraju modulima ili grupama modula aplikacije.

Da bi se naglasila međuzavisnost HTML i CSS koda, kao i hijerarhijski način pisanja CSS koda, u ovom primeru se kompletan CSS kod nalazi u fajlu *style.css* lociranom u direktorijumu *public/css*, listing 196. Struktura dokumenta je sledeća:

- Deklaracija CSS promenljivih, linije 1-6
- Stilizovanje osnovnih HTML tagova (body, a, input elementi, ...), linije 7-24
- Stilizovanje osnovnih stukturnih elemenata veb stranice:
  - zaglavlje, linije 25-31
  - navigacioni meni, linije 32-44
  - h1-h3 tagovi, linije 45-54
  - datum objave članka, linije 55-58
  - footer, linije 120-130

- Struktura svake veb stranice koju generiše ova aplikacija je tako koncipirana da je glavni sadržaj stranice ugnežđen u tag <*stranica*>...</*stranica*>, listinzi 171 i 172, pa potom u tag koji označava prirodu stranice, npr. <*pocetna*>...</*pocetna*>, <*vesti*>...</*vesti*>, <*vest*>...</*vest*>, <*kontakt*> ...</*kontakt*> i sl. Na osnovu toga se stranice mogu nezavisno stilizovati, bez bojazni da će se opisi elemenata na različitim stranicama međusobno potirati:
  - Stranica greška, linije 70-76
  - Početna strana, linije 77-80
  - Listing vesti, linije 81-87
  - Prikaz vesti, linije 88-100
  - Unos ili izmena vesti, linije 101-108
  - Kontakt stranica, linije 109-112
  - Login stranica, linije 113-119

Listing 196 - CSS kod aplikacije

```
:root {
  --border-structural: 1px dotted gray;
  --razmak-v1: 1em 0;
  --razmak-v05: 0.5em 0;
  --razmak-v025: 0.25em 0;
}
body {
  font: 12pt Arial, sans-serif;
  margin: 0;
}
a {
  text-decoration: none;
}
button {
  width: auto !important;
}
input,
textarea {
  padding: 0.25em;
}
```

```
textarea {
  height: 10em;
  resize: none;
}
header {
  height: 150px;
  padding: 0.25em;
  background-image: url(../images/singidunum-logo.png);
  background-repeat: no-repeat;
  background-size: auto 100%;
}
nav {
  border: var(--border-structural);
  border-left: none;
  border-right: none;
  background-image: linear-gradient(silver, white, silver);
  padding: 0.25em;
  text-align: center;
  position: sticky;
  top: 0;
}
nav a:not(:last-child) {
  margin-right: 1em;
}
h1 {
  font-size: 3em;
  padding: var(--razmak-v025);
}
h2 {
  font-size: 2em;
}
h3 {
  margin: var(--razmak-v025);
}
datum {
  display: block;
  font-size: 0.8em;
```

```
58 }
59 stranica {
60   padding: 1em;
61   display: block;
62 }
63 stranica breadcrumbs {
64   font-size: 1em;
65 }
66 stranica breadcrumbs razdelnik:before {
67   margin: 0 0.25em;
68   content: '>';
69 }
70 stranica greska {
71   font-weight: bold;
72   color: red;
73   text-align: center;
74   display: block;
75   margin: var(--razmak-v1);
76 }
77 stranica pocetna vest {
78   margin-top: 0.5em;
79   display: block;
80 }
81 stranica vesti vest datum {
82   margin-bottom: 0.25em;
83 }
84 stranica vesti vest kratko {
85   color: gray;
86   font-style: italic;
87 }
88 stranica vest clanak {
89   margin: var(--razmak-v1);
90   display: block;
91 }
92 stranica vest clanak p,
93 stranica vest clanak br {
94   display: block;
95   margin: 0.25em 0;
```

```
}
stranica vest clanak img {
  display: block;
  margin: 0.5em auto;
}
stranica vest form * {
  display: block;
  width: 100%;
}
stranica vest textarea {
  height: 10em;
  resize: none;
}
stranica kontakt form * {
  display: block;
  width: 50%;
}
stranica login form {
  text-align: center;
}
stranica login form input {
  display: block;
  margin: auto;
}
footer {
  font-size: 1em;
  border-top: var(--border-structural);
  margin-top: 2em;
  padding-top: 1em;
  text-align: center;
}
footer secondary {
  display: block;
  font-size: 0.75em;
}
```

LITERATURA, INDEKSI LISTINGA, SLIKA I TABELA

LITERATURA

Bouras, A., *PHP and Algorithmic Thinking for the Complete Beginner*, Independently published, 2021, p. 704

Dobrojević, M., *Kako napraviti web aplikaciju? PHP, MySQL, JavaScript*, www.magma.rs, Beograd, 2016, ISBN 978-86-920027-0-0

Hansen, B., *Practical Oracle SQL*, APRESS, 2020, p. 460

Lindley, C., *DOM Enlightenment: Exploring JavaScript and the Modern DOM 1st Edition,* O'Reilly Publisher, 2013, p. 1232

Nikolić B., *Internet programiranje 1*, ETF, 2008, p. 378

Nixon, R., *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (Learning PHP, MYSQL, JavaScript, CSS & HTML5) 5th Edition,* O'Reilly Publisher, 2019, p. 1278

Pratt, L., *Advanced Guide to Learn the Realms of PHP Programming*, Independently published, 2021, p. 230

Veinović, M., Šimić, G., Jevremović, A., Tair, M., *Baze podataka*, UniverzitetSingidunum, Beograd, 2018, p.307

Welling, L., Thomson, L., *PHP and MySQL Web Development (Developer's Library) 5th Edition*, Addison-Wesley Professional, 2017, p. 329.

Wisborg, K., *MySQL 8 Query Performance Tuning*, APRESS, 2020, p. 965

CSS: Cascading Style Sheets, https://developer.mozilla.org/en-US/docs/Web/CSS

JavaScript, https://developer.mozilla.org/en-US/docs/Web/JavaScript

MariaDB Server Documentation, https://mariadb.com/kb/en/documentation/

MySQL Documentation, https://dev.mysql.com/doc/

PHP Documentation, https://www.php.net

INDEKS SLIKA

Slika 1 - Visual Studio Code, izgled radnog ekrana 7
Slika 2 - HTTP zaglavlje (engl. *header*) 12
Slika 3 - HTTP odgovor veb servera 12
Slika 4 - Izgled elementa na ekranu 14
Slika 5 - PHP veb aplikacija: korelacija između komponenti sistema 16
Slika 6 - Konfigurisanje ekstenzija u *php.ini* konfiguracionom dokumentu 17
Slika 7 - Upotreba programskih jezika na strani servera, april 2021. 17
Slika 8 - Preuzimanje XAMPP softverskog paketa, *apachefriends.org* 20
Slika 9 - Komponente XAMPP paketa koje su neophodne za razvoj PHP veb aplikacija 21
Slika 10 - Izbor imena direktorijuma u koji će komponente XAMPP-a biti instalirane 22
Slika 11 - XAMPP kontrolni panel za individualnu kontrolu instaliranih komponenti 22
Slika 12 - XAMPP dashboard 23
Slika 13 - Set pravila za osnovni direktorijum (DocumentsRoot), httpd.conf 25
Slika 14 - Zabrana pristupa fajlovima *.htaccess* i *.htpasswd*, dokument *httpd.conf* 26
Slika 15 - Primer parova korisničko ime i hešovana lozinka (MD5) 26
Slika 16 - Obavezna referenca ka *.htpasswd* u dokumentu *.htaccess* 26
Slika 17 - Apache veb server, listing direktorijuma kada nije navedeno ime dokumenta 28
Slika 18 - Struktura veb projekata u htdocs direktorijumu 29
Slika 19 - Definicija virtuelnog domena, Apache dokument *httpd-vhosts.conf* 30
Slika 20 - Ispis sadržaja niza 52
Slika 21 - Kontrolisani ispis članova niza, veb stranica 55
Slika 22 - Greška prouzrokovana pozivom funkcije bez argumenta 58
Slika 23 - Obrađeni članovi niza 59
Slika 24 - Serijalizovani podaci, rezultat 62
Slika 25 - Serijalizovani podaci vraćeni u izvorni oblik, rezultat 62
Slika 26 - Veb obrazac za pretragu sadržaja sajta - označavanje elemenata obrasca 69
Slika 27 - Veb obrazac za unos imena i prezimena 71
Slika 28 - Prijem dokumenta putem veb obrasca 76
Slika 29 - Unos, prijem i provera podataka uz prijavu greške 77

Slika 30 - Primer zaglavlja stranice sa setovanim kolačićem 102
Slika 31 - Veb obrazac za prijavu korisnika 109
Slika 32 - Uneti podaci nisu tačni. Prikaz greške iznad obrasca za prijavu. 109
Slika 33 - Korisnik je ulogovan 110
Slika 34 - Rangiranje sistema za upravljanje bazama podataka 130
Slika 35 - *PHPMyAdmin*, izvoz baze - I 148
Slika 36 - *PHPMyAdmin*, izvoz baze - II 149
Slika 37 - Rezultat nepohlepnog regularnog izraza 178
Slika 38 - Primena modifikatora na tekstu sa više linija 181
Slika 39 - Upozorenje o grešci u kodu 190
Slika 40 - Podaci o pozivima funkcije 194
Slika 41 - Struktura *Exception* objekta 196
Slika 42 - Prikaz *HTML DOM* stabla 201
Slika 43 - Faze izvršavanja AJAX koda 252
Slika 44 - Sadržaj fajla *rečnik.txt* 254
Slika 45 - Struktura fajla *carCatalog.xml* 261
Slika 46 - Osnovni elementi korisničkog interfejsa veb aplikacije 273
Slika 47 - Obrazac za registraciju korisničkog naloga 275
Slika 48 - Obrazac za prijavljivanje korisnika (login) 276
Slika 49 - Grupisanje u kategorije profila kompanija prema delatnosti 278
Slika 50 - Veb obrazac za jednostavnu pretragu sadržaja veb sajta 279
Slika 51 - Veb obrazac za pretragu oglasnika po osnovu više kriterijuma 279
Slika 52 - Veb obrazac za pretragu veb prodavnice 280
Slika 53 - Višejezičnost veb aplikacije na primeru obrasca za prijavu (login) korisnika 281
Slika 54 - Veb stranica kao rezultat koda prikazanog na listingu 157 283
Slika 55 - Troslojna arhitektura veb aplikacije 284
Slika 56 - Višeslojna arhitektura veb aplikacije 285
Slika 57 - Struktura dokumenata veb aplikacije 286
Slika 58 - MVC model 287
Slika 59 - Početna strana 297
Slika 60 - Sekcija *Vesti*, spisak objavljenih članaka 298
Slika 61 - Prikaz članka 298

Slika 62 - Kontakt stranica 299
Slika 63 - Struktura dokumenata veb aplikacije 304
Slika 64 - Algoritam modula *Vesti*, direktorijum *includes, fajl stranica_vesti.php* 317
Slika 65 - Sekcija *Vesti,* spisak objavljenih članaka (*Administrator*) 319
Slika 66 - Menadžment članaka, simbolički prikaz funkcionalnosti 321
Slika 67 - Zahtev za potvrdu akcije brisanja članka 322
Slika 68 - Veb obrazac za unos novog i izmenu postojećeg članka 330
Slika 69 - Unos podataka u kontakt obrazac, prijava greške 333
Slika 70 - Login obrazac 335
Slika 71 - Obaveštenje o pogrešno unešenim podacima prilikom prijave 335
Slika 72 - Struktura javno dostupnih dokumenata 342

INDEKS LISTINGA

Listing 1 - Osnovna struktura HTML dokumenta 8
Listing 2 - Primer stilizovanja elementa na veb stranici primenom CSS koda 14
Listing 3 - Primer PHP dokumenta 35
Listing 4 - PHP i HTML kod u istom dokumentu 35
Listing 5 - Imena funkcija nisu osetljiva na velika i mala slova 36
Listing 6 - Upotreba promenljivih u globalnom i lokalnom opsegu 38
Listing 7 - Primena statičkih promenljivih 39
Listing 8 - Upotreba jednostrukih i dvostrukih navodnika 44
Listing 9 - Matematičke operacije nad stringovima 44
Listing 10 - Kontrola toka programa - *if-else* struktura 45
Listing 11 - Kontrola toka programa - *switch blok* 46
Listing 12 - PHP *for* petlja 46
Listing 13 - PHP *foreach* petlja 47
Listing 14 - PHP *while* petlja 47
Listing 15 - PHP *do-while* petlja 47
Listing 16 - Očitavanje trenutnog vremena na veb serveru 48
Listing 17 - Formatiranje UNIX vremena u ljudima prepoznatljiv format 48
Listing 18 - Konverzija stringa u UNIX vreme 50
Listing 19 - Tipovi nizova: a) Indeksiran, b) asocijativan i c) višedimenzioni 51
Listing 20 - Transformacija stringa u niz 52
Listing 21 - Transformacija niza u string 53
Listing 22 - Kontrolisani ispis članova niza 54
Listing 23 - Dinamičko generisanje niza 55
Listing 24 - Deklaracija funkcije 57
Listing 25 - Prosleđivanje argumenata referencom 58
Listing 26 - Anonimna funkcija, dodela vrednosti promenljivoj 59
Listing 27 - Obrada članova niza 59
Listing 28 - Zamena dela stringa upotrebom regularnih izraza i anonimne funkcije 59
Listing 29 - Vidljivost promenljive u anonimnoj funkciji 60

Listing 30 - Serijalizacija niza i deserijalizacija podataka, kod 61
Listing 31 - Redirekcija upotrebom HTTP zaglavlja, funkcija *header()* 63
Listing 32 - Slanje koda statusa zahteva, greška 404 63
Listing 33 - Upotreba PHP funkcije *include()* 63
Listing 34 - Pozicioniranje oznake elementa u okviru 70
Listing 35 - Očitavanje vrednosti poslatih putem veb obrasca GET metodom 72
Listing 36 - Slanje dokumenta putem veb obrasca 75
Listing 37 - Unos, prijem i provera podataka uz prijavu greške, kod 78
Listing 38 - Sekvencijalno čitanje dokumenta 87
Listing 39 - Kreiranje novog dokumenta, *file_put_contents()* 88
Listing 40 - Kreiranje novog dokumenta, *fwrite()* 88
Listing 41 - Očitavanje spiska dokumenata u folderu, *funkcija glob()* 89
Listing 42 - Očitavanje sadržaja direktorijuma, *funkcija readdir()* 90
Listing 43 - Menadžment direktorijuma 91
Listing 44 - Provera da li dokument postoji, funkcija *file_exists* 92
Listing 45 - Kopiranje dokumenta, funkcija *copy* 92
Listing 46 - Premeštanje otpremljenog dokumenta, funkcija *move_uploaded_file()* 93
Listing 47 - Uklanjanje dokumenta, funkcija *unlink()* 94
Listing 48 - Očitavanje vremena kreiranja, izmene i pristupa dokumentu 95
Listing 49 - Izmena vremena izmene i pristupa dokumentu 96
Listing 50 - Postavljanje kolačića 103
Listing 51 - Smeštanje niza u kolačić 104
Listing 52 - Praćenje broja poseta istog korisnika putem sesije 106
Listing 53 - Prijava i odjava korisnika veb aplikacije 108
Listing 54 - Sintaksa definisanja klase 115
Listing 55 - Deklarisanje klase i instanciranje objekta - I 118
Listing 56 - Deklarisanje klase i instanciranje objekta - II 119
Listing 57 - Korišćenje konstante u deklarisanju klase 119
Listing 58 - Nasleđivanje klasa i instanciranje objekata 120
Listing 59 - Primer nasleđivanja - I 121
Listing 60 - Primer nasleđivanja - II 122

Listing 61 - Sprečavanje nasleđivanja klase i metode 123
Listing 62 - Implementiranje interfejsa 124
Listing 63 - Statički atributi i metodi 125
Listing 64 - Primer kreiranja i brisanja korisnika 134
Listing 65 - Dodela privilegija naredbom *GRANT* 134
Listing 66 - Povezivanje sa serverom baze podataka, funkcija *mysqli_connect()* 137
Listing 67 - Kreiranje i brisanje baze na lokalnom računaru, funkcija *mysqli_query()* 137
Listing 68 - Kreiranje tabela, funkcija *mysqli_query()* 138
Listing 69 - Upisivanje podataka u tabelu, funkcija *mysqli_query()* 139
Listing 70 - Prikaz rezultata izvršavanja upita, funkcija *mysqli_fetch_row()* 139
Listing 71 - Kreiranja i brisanje baze 141
Listing 72 - Kreiranje tabele 141
Listing 73 - Upis podataka u tabelu i prikaz trenutne *AUTO_INCREMENT* vrednosti 142
Listing 74 - Izvršavanje više upisa istovremeno 143
Listing 75 - Upotreba pripremljenih upita i vezivanje parametara 144
Listing 76 - Izvršavanje *SELECT* upita i ispis rezultata 145
Listing 77 - Izvršavanje *SELECT* upita i ubacivanje rezultata u HTML tabelu 146
Listing 78 - Brisanje slogova 146
Listing 79 - Ažuriranje slogova 147
Listing 80 - Primer XML sintakse 153
Listing 81 - Primer XML sintakse, tagovi 154
Listing 82 - XML struktura, root-child 156
Listing 83 - Jednostavan XML dokument 156
Listing 84 - Primer XML dokumenta koji opisuje vesti 157
Listing 85 - Primer XML dokumenta koji opisuje vremensku prognozu 158
Listing 86 - JSON sintaksa 160
Listing 87 - *JSON* format, podaci o raspoloživim knjigama 161
Listing 88 - Primer korišćenja *JSON*-a u HTML dokumentu 162
Listing 89 - Primer kreiranja niza *JSON* objekata u HTML-u 163
Listing 90 - *JSON* shema 165
Listing 91 - Poređenje sintakse: *JSON* (levo) i *XML* (desno) 166

Listing 92 - Parsiranje HTML koda regularnim izrazima 178
Listing 93 - Upotreba preg_match funkcije 182
Listing 94 - Upotreba preg_match_all funkcije 183
Listing 95 - Upotreba preg_split funkcije 184
Listing 96 - Upotreba preg_grep funkcije 185
Listing 97 - Upotreba nedeklarisane promenljive 189
Listing 98 - Greška u sintaksi 191
Listing 99 - Kritična greška 191
Listing 100 - Testiranje funkcije 193
Listing 101 - Praćenje poziva određene funkcije 194
Listing 102 - Aktiviranje izuzetka 195
Listing 103 - Primer sa *getElementById()* metodom i *innerHTML* svojstvom 203
Listing 104 - Ciljanje elementa i izmena sadržaja, *getElementById()* i i*nnerHML* 205
Listing 105 - Ciljanje elementa, metoda *getElementsByTagName()* 206
Listing 106 - Ciljanje elemenata, *getElementsByClassName()* 207
Listing 107 - Ciljanje HTML elemenata, CSS selektor *querySelectorAll()* 208
Listing 108 - Pronalaženje svih HTML elemenata koji pripadaju istoj HTML formi 209
Listing 109 - Primer sa *document.write()* metodom 210
Listing 110 - Promena vrednosti svojstva HTML objekta 211
Listing 111 - Promena vrednosti svojstva CSS stila 212
Listing 112 - Primer jednostavnog događaja - I 213
Listing 113 - Primer jednostavnog događaja - II 214
Listing 114 - Primer jednostavnog događaja - III 214
Listing 115 - Dodavanje događaja HTML elementima 215
Listing 116 - Primer onchange događaja 216
Listing 117 - Primer *onmouseup* i onmousedown događaja 217
Listing 118 - Primer sa *addEventListener* metodom - I 218
Listing 119 - Primer sa *addEventListener* metodom - II 219
Listing 120 - Primer sa *addEventListener* metodom - III 220
Listing 121 - Primer sa *addEventListener* metodom - IV 221
Listing 122 - Primer sa propagacijom događaja 223

Listing 123 - Primer sa *removeEventListener* metodom 224
Listing 124 - Primer sa DOM navigacijom - I 227
Listing 125 - Primer sa DOM navigacijom - II 228
Listing 126 - Primer sa DOM navigacijom - III 228
Listing 127 - Primer sa *nodeName* svojstvom 230
Listing 128 - Primer sa appendChild metodom 231
Listing 129 - Primer sa *insertBefore* metodom 231
Listing 130 - Brisanje HTML elementa pomoću imena roditelja 232
Listing 131 - Primer replaceChild() metode 233
Listing 132 - Prikazivanje veličine prozora pretraživača 235
Listing 133 - Upotreba svojstava *window.screen.width* i *window.screen.height* 236
Listing 134 - Svojstava *window.screen.availWidth* i *window.screen.availHeight* 237
Listing 135 - Svojstva *window.screen.colorDepth* i *window.screen.pixel Depth* 237
Listing 136 - Primer sa atributima *window.location* objekta - I 238
Listing 137 - Primer sa atributima *window.location objekta* - II 239
Listing 138 - Primer sa *window.history.back()* i *window.history.forward()* metodama 239
Listing 139 - Primer sa atributima *window.navigator* objekta - I 240
Listing 140 - Primer sa atributima *window.navigator* objekta - II 241
Listing 141 - Primer sa *window.alert()* 242
Listing 142 - Primer sa *window.confirm()* 243
Listing 143 - Primer sa *window.prompt()* 244
Listing 144 - Primer sa tajming događajima - I 245
Listing 145 - Primer sa tajming događajima - II 246
Listing 146 - Primer sa tajming događajima - III 247
Listing 147 - Primer sa kolačićima 250
Listing 148 - Čitanje sadržaja fajla pomoću AJAX tehnologije 255
Listing 149 - Modifikovani primer sa člitanjem sadržaja fajla pomoću AJAX tehnologije 256
Listing 150 - Primer slanja podataka serveru pomoću GET metode 257
Listing 151 - Kod fajla *JSAJAX3.php* 257
Listing 152 - Primer slanja podataka serveru pomoću POST metode 258
Listing 153 - Očitavanje XML fajla pod nazivom *carCatalog.xml* pomoću AJAX-a - I 262

Listing 154 - Očitavanje XML fajla pod nazivom *carCatalog.xml* pomoću AJAX-a - II 263
Listing 155 - Primer pozivanja PHP fajla pomoću AJAX-a 265
Listing 156 - Kod fajla *JSAJAX7.php* 266
Listing 157 - PHP, HTML, CSS i JavaScript kod u jednom dokumentu 282
Listing 158 - Ruter veb aplikacije 290
Listing 159 - SQL upit za formiranje tabele *articles* 300
Listing 160 - SQL upit za formiranje tabele *categories* 300
Listing 161 - SQL upit za ubacivanje kategorija, tabela *categories* 300
Listing 162 - SQL upit za postavljanje indeksa u tabelama *articles* i *categories* 301
Listing 163 - SQL upit za formiranje tabele *korisnici* 301
Listing 164 - Preusmeravanje tražene URL adrese na fajl *index.php* 305
Listing 165 - Ulazna tačka aplikacije, *index.php* 305
Listing 166 - Konfiguracioni podaci, *config.php* 306
Listing 167 - Uspostavljanje veze sa bazom podataka, *db.php* 307
Listing 168 - Zajedničke funkcije, *funkcije.php* 308
Listing 169 - Šablon (*templejt*) za prikaz mrvica hleba, *breadcrumbs.php* 310
Listing 170 - Ruter aplikacije, *router.php* 311
Listing 171 - HTML zaglavlje šablona, *header.php* 312
Listing 172 - HTML footer šablona, *footer.php* 313
Listing 173 - Model *početne* strane, direktorijum *includes, stranica_pocetna.php* 314
Listing 174 - Šablon početne strane, direktorijum template, stranica_pocetna.php 315
Listing 175 - Kruta struktura rutera 316
Listing 176 - Ruter modula *vesti*, direktorijum *includes, stranica_vesti.php* 318
Listing 177 - Funkcije modula vesti, direktorijum *includes, funkcije_vesti.php* 319
Listing 178 - Spisak vesti, direktorijum *includes, stranica_vesti_spisak.php* 320
Listing 179 - Spisak vesti, šablon, direktorijum *template, stranica_vesti.php* 321
Listing 180 - Kontrola interakcije sa linkovima za brisanje vesti 323
Listing 181 - HTML model naslova vesti sa ikonama za menadžment članka 324
Listing 182 - Prikaz članka, direktorijum *includes, stranica_vesti_vest.php* 325
Listing 183 - Prikaz članka, šablon, direktorijum *template, stranica_vest.php* 326
Listing 184 - Administracija vesti, direktorijum *includes, stranica_vesti_admin.php* 326

Listing 185 - Šablon administracije vesti, direktorijum *template, stranica_vesti_edit.php* 329
Listing 186 - Model *kontakt* strane, direktorijum *includes, stranica_kontakt.php* 332
Listing 187 - Šablon kontakt strane, direktorijum *template, stranica_kontakt.php* 334
Listing 188 - Šablon za ispis greške na veb stranici 334
Listing 189 - Model *login* strane, direktorijum *includes, stranica_login.php* 336
Listing 190 - Šablon *login* strane, direktorijum *template, stranica_login.php* 337
Listing 191 - Model *RSS feed*-a, direktorijum *includes, stranica_rss.php* 339
Listing 192 - Funkcije modula *RSS*, direktorijum *includes, funkcije_rss.php* 339
Listing 193 - Šablon *RSS* strane, direktorijum *template, stranica_rss.php* 340
Listing 194 - Model strane *Greška,* direktorijum *includes, stranica_greska.php* 341
Listing 195 - Šablon strane *Greška*, direktorijum *template, stranica_greska.php* 341
Listing 196 - CSS kod aplikacije 344

INDEKS TABELA

Tabela 1 - Popularni editori koda 8
Tabela 2 - Primer upotrebe URL parametara 10
Tabela 3 - Kontrola komponenti XAMPP paketa putem komandne linije u Windows OS 23
Tabela 4 - Najčešće korišćeni kodovi za formatiranje datuma pomoću funkcije *date()* 49
Tabela 5 - Mime tip dokumenata 74
Tabela 6 - Funkcija *fopen*, režimi rada sa dokumentom 86
Tabela 7 - Kontrola ponašanja funkcije *glob()* 89
Tabela 8 - Struktura MySQL tabele *user* 132
Tabela 9 - Struktura MySQL tabele *db* 133
Tabela 10 - Struktura MySQL tabele *tables_priv* 133
Tabela 11 - Struktura MySQL tabele *columns_priv* 133
Tabela 12 - Struktura MySQL tabele *procs_priv* 134
Tabela 13 - Glavne razlike između *XML* i *JSON* formata 167
Tabela 14 - Metode *XMLHttpRequest* objekta 253
Tabela 15 - Atributi *XMLHttpRequest* objekta 254
Tabela 16 - Atributi *XMLHttpRequest* objekta 260
Tabela 17 - Ruter veb aplikacije - upotreba URL parametara i semantičkih URL 289
Tabela 18 - Semantički linkovi, anoniman korisnik 303
Tabela 19 - Semantički linkovi, administrator 303

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд

004.42:004.738.5(075.8)

004.438PHP(075.8)

ДОБРОЈЕВИЋ, Милош, 1975-

Veb programiranje / Miloš Dobrojević, Nebojša Bačanin-Džakula. - 1. izd. - Beograd : Univerzitet Singidunum, 2021 (Beograd : Birograf). - VI, 362 str. : ilustr. ; 24 cm

Tiraž 1.200. - Bibliografija: str. 351. - Registri.

ISBN 978-86-7912-752-5

1. Бачанин Џакула, Небојша, 1983- [аутор]

a) Интернет - Програмирање b) Програмски језик "PHP"

COBISS.SR-ID 37240073

www.singidunum.ac.rs

9 788679 127525

VEB PROGRAMIRANJE

Miloš Dobrojević
Nebojša Džakula-Bačanin

Veb aplikacije su nezaobilazni deo svakodnevnog, poslovnog i privatnog života. Koriste se za pristup elektronskoj pošti, za obavljanje kancelarijskih poslova, za upravljanje poslovnim procesima i izveštavanje, za pristup društvenim mrežama, oglasnicima, medijskim portalima i sl.

Ova knjiga je namenjena studentima koji prate predmet Veb programiranje na drugoj godini studija Tehničkog fakulteta Univerziteta Singidunum, na studijskom programu Softversko i informaciono inženjerstvo. Od studenata se očekuje da su prethodno odslušali predmete Veb dizajn i Baze podataka, odnosno da su upoznati sa HTML markirnim jezikom, CSS-om, osnovama SQL jezika, kao i sa osnovnim principima objektno orijentisanog programiranja. Tehnologija za razvoj veb aplikacija sa kojom će se studenti upoznati na ovom predmetu je bazirana na WAMP/LAMP/XAMPP platformi koju čine Apache veb server, programski jezici PHP (backend) i JavaScript (frontend), uz upotrebu MySQL ili MariaDB baze podataka.

Univerzitet Singidunum