SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

**Zoran Antolović**

WEB SUSTAVI VISOKIH PERFORMANSI BAZIRANI NA PHP-u

DIPLOMSKI rad

Varaždin, 2017.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Zoran Antolović

Matični broj: 43579/14–R

Studij: Informacijsko i programsko inženjerstvo

WEB SUSTAVI VISOKIH PERFORMANSI BAZIRANI NA PHP-u

DIPLOMSKI rad

Mentor:

Prof. dr. sc. Dragutin Kermek

Varaždin, kolovoz 2017.

**Sadržaj**

1. Uvod 1

2. World Wide Web 3

2.1. Nastanak i razvoj World Wide Weba 3

2.2. Hypertext Transfer Protocol (HTTP) 4

2.3. Web sustavi 7

2.3.1. Web stranice 7

2.3.2. Web servisi 8

2.3.3. Web aplikacije 8

2.4. Evolucijske faze razvoja World Wide Weba 9

2.5. Tehnologije za razvoj web sustava 12

2.5.1. Razvoj web sustava na strani klijenta 13

2.5.2. Razvoj web sustava na strani poslužitelja 15

3. PHP programski jezik 17

3.1. Razvoj PHP programskog jezika 17

3.2. Razvoj suvremenih web sustava u PHP programskom jeziku 22

3.2.1. Tržišni udio i popularnost PHP programskog jezika 23

3.2.2. Razvojni ekosustav i zajednica 26

4. Performanse web sustava 29

4.1. Web sustavi visokih performansi 29

4.2. Performansne karakteristike web sustava 31

4.2.1. Propusnost web sustava 31

4.2.2. Vrijeme odziva web sustava 31

4.2.3. Dostupnost web sustava 31

4.3. Testiranje performansi web sustava 31

4.3.1. Testiranje web sustava pod opterećenjem 31

4.3.2. Testiranje granica web sustava 31

4.3.3. Regresijsko testiranje web sustava 31

5. Razvoj web sustava visokih performansi 32

5.1. Koraci razvoja i optimizacije web sustava 32

5.2. Mjerenje i optimizacija performansi 33

5.2.1. Optimizacija aplikacijskog sloja web sustava 33

5.2.2. Optimizacija podatkovnog sloja web sustava 34

5.2.2.1. Optimizacija upita prema bazi podataka 34

5.2.2.2. Prilagodba strukture baze podataka s ciljem optimizacije sustava 34

5.2.2.3. Pohrana rezultata u privremenu memoriju 34

5.2.3. Optimizacija infrastrukturnog sloja web sustava 34

5.2.3.1. HTTP zaglavlja i pohrana podataka u privremenu memoriju 34

5.2.3.2. Kompresija podataka i pohrana podataka u privremenu   
memoriju preglednika 34

6. Praktičan rad – web sustav visokih performansi 35

6.1. Opis web sustava 35

6.1.1. Infrastruktura 36

6.1.2. Model baze podataka 36

6.1.3. Korišteni alati 36

6.2. Optimizacija i mjerenje performansi web sustava 36

6.2.1. Performanse sustava bez optimizacijskih mehanizama 36

6.2.2. Optimizacija dodavanjem indeksa baze podataka 36

6.2.3. Optimizacija korištenjem privremen pohrane podataka na strani aplikacije 36

6.2.4. Optimizacija korištenjem HTTP poslužitelja za privremenu   
pohranu podataka 36

6.2.5. Sumarni prikaz rezultata i interpretacija 36

7. Zaključak 37

8. Literatura 38

9. Izvorni kod praktičnog primjera 42

1. Uvod

World Wide Web (WWW, web) nastao je 1989. godine kada je Tim Berners-Lee u CERN-u izradio prijedlog sustava za upravljanje i dijeljenje informacija kako bi kolegama znanstvenicima omogućio jednostavniju razmjenu informacija kako unutar institucije tako i između znanstveno-istraživačkih institucija[[1]](#footnote-1). Prva je web stranica objavljena 20. prosinca 1990. godine, a kao prvi web poslužitelj (eng. *web server*) Berners-Lee je iskoristio svoje NeXT računalo. Ta je web stranica opisivala koncept i osnovne funkcionalnosti weba, sadržavala je upute za pristupanje dokumentima drugih osoba te upute za postavljanje osobnog web poslužitelja[[2]](#footnote-2). Izvorni kod World Wide Weba javno je objavljen 1993. godine.

27 godina od objave prve web stranice, Internet broji više od 3.5 milijardi korisnika, 966 milijuna web stranica, a gotovo 50% svjetske populacije ostvaruje pristup Internetu uz značajnu korelaciju (0.87) stupnja razvijenosti države i mogućnosti pristupa Internetu (Poushter, 2016; Internet World Stats, 2017; VPN Mentor, 2017).

Razvojem weba i porastom broja korisnika weba značajno je prošireno područje informatičke i računalne industrije, marketinga, prodaje ali i drugih industrija kao što su industrija zabave i turizam. Stvorena su nova zanimanja poput web dizajnera i programera, upravitelja društvenim mrežama (eng. *community manager*, *social media manager*), a sam web preuzeo je ulogu vodećeg medija za distribuciju informacija. Iako primarno razvijen u akademske svrhe, web je danas centralno mjesto komunikacije i socijalizacije (npr. društvene mreže, platforme za diskusije i instant dopisivanje), poslovanja (digitalne trgovine, internet bankarstvo, poslovne aplikacije) ali i zabave (internet portali, mrežne igrice, platforme za multimediju). Web se pokazao kao pouzdan medij u kriznim situacijama, a trenutačna razmjena informacija kroz društvene mreže omogućila je transparentniju i distribuciju informacija bez cenzure.

Performanse web stranica (web sustava) značajno utječu na korisničko iskustvo, preferencije korisnika i njihovo ponašanje na webu, pogotovo kod korisnika koji webu pristupaju s mobilnih uređaja. Primjerice, ukoliko učitavanje sadržaja potraje više od 3 sekunde 53% korisnika koji pristupaju webu s mobilnog uređaja istu napušta. Brzina učitavanja sadržaja (eng. load time, render time) izravno utječe na efikasnost prikazivanja oglasa (tj. zarade od internet oglasa), korisničko zadržavanje na web mjestu i sklonost napuštanja web mjesta sa prve posjećene stranice (eng. bounce rate) (Shellhammer, 2017).

Ovaj rad donosi osnovne informacije o povijesti i razvoju weba, web sustavima i njihovu kategorizaciju, povijest i pregled razvoja PHP programskog jezika te optimizacijske mehanizme primijenjive za web sustave visokih performansi s praktičnim primjerom. Posebna je pozornost posvećena performansama web sustava, definiranju metrika, mjerenju performansi te primjenu principa pohrane podataka u privremenu memoriju (eng. *caching*) s ciljem optimizacije web sustava. Praktični dio rada prikazuje optimizaciju fiktivnog web portala razvijenog u PHP programskom jeziku, mjerenje njegovih performansi, identifikaciju uskih grla, optimizaciju te usporedbu performansi prije i nakon optimizacije s primjerima koda.

1. World Wide Web

Internet je često krivo poistovjećen s World Wide Webom (u nastavku rada web) koji je, iako najpopularniji, samo jedan njegov aspekt. Internet podrazumijeva "*mrežu međusobno povezanih prostorno distribuiranih računala*" (Collins, 2012), dok je web "*prostor informacija unutar kojeg se interesni pojmovi nazivaju resursima i identificiraju globalnim oznakama nazvanim Uniform Resource Identifiers (URI)*" (Berners-Lee *et al.*, 2004). Na web možemo gledati kao na uslugu razmjene informacija u obliku dokumenata tj. web stranica. Dakle, web je usluga bazirana na Internetu baš kao što je to i elektronska pošta (eng. email) ili *peer-to-peer* razmjena podataka. Nastavak poglavlja donosi pregled razvoja World Wide Weba, osnove HTTP protokola, kategorizaciju web sustava i pregled tehnologija za razvoj web sustava..

* 1. Nastanak i razvoj World Wide Weba

Tim Berners-Lee je 90-tih godina 20. stoljeća radio kao istraživač u CERN-u, najvećem laboratoriju za istraživanje čestica, gdje se više od tisuću znanstvenika bavilo znanstvenim istraživanjima radovima. U CERN-u je postojala ogromna količina informacija, ali pristup i pronalazak traženih dokumenata nije bio jednostavan. Znanstvenici su se često morali prijavljivati u različita računala, koristiti (i naučiti kako koristiti) različite sustave kako bi pristupili informaciji koja im je potrebna. Berners-Lee je bio frustriran takvim uvjetima gdje je znanstvenicima bilo jednostavnije "*pitati jedan drugoga za informacije u vrijeme kave*" nego pronaći informaciju u sustavu (Web Foundation, s.a.; Berners-Lee, 2017). U CERN-u se tada najviše komuniciralo emailom a dijeljenje informacija bilo je svedeno na izravno dijeljenje datoteka.

Berners-Lee je 1980. godine radio na projektu Enquire kako bi razvio rješenje za upravljanje i rad s dokumentima odnosno informacijama temeljeno na hipertekstu (eng. *hypertext*). Sustav je morao biti kompatibilan s različitim mrežama, formatima diska, formatima podataka i načinima kodiranja podataka. Enquire je prethodnik današnjeg weba u kojem se korisnik kretao kroz informacijski prostor pomoću tekstualnih poveznica (eng. *link, hyperlink*). Enquire je bio sličan aplikaciji Hypercard koju je Apple razvio za svoj Macintosh operacijski sustav. Enquire je, za razliku od Hypercarda, bilo moguće pokrenuti na različitim i višekorisničkim sustavima ali nije imao podršku za slikovne datoteke (Berners-Lee, 1989). Berners-Lee je privremeno napustio CERN da bi se ponovno vratio 1984. godine i nastavio s radom. Aktivno je koristio Enquire za pohranu i povezivanje svojih informacija ali je shvatio da previše vremena provodi ažurirajući pohranjene informacije. Shvatio je kako CERN treba sustav poput Enquirea ali dostupan svima, ne samo njemu. Berners-Lee je procijenio kako je za prvu fazu razvoja takvog sustava potreban tim od 2 znanstvenika i vrijeme od 6 do 12 mjeseci (Berners-Lee, 1989, no date b).

Berners-Lee je svoje rješenje za upravljanje i distribuciju informacija temeljio na hipertekstu, konceptu poveznica koje korisnicima omogućava kretanje kroz prostor informacija, koji su ranije opisali Vanevar Bush (1945.) i Ted Nelson (1965.). Doug Engelbart je u 60. godinama 20. stoljeća razvio sustav sličan webu pokretan na jednom računalu (internet još nije bio razvijen). Berners-Lee navodi kako je on "samo" povezao koncept hiperteksta s idejama TCP (Transmission Control Protocol) i DNS (Domain Name System) (Berners-Lee, no date a):

"*I just had to take the hypertext idea and connect it to the TCP and DNS ideas and -- ta-da! -- the World Wide Web.*"

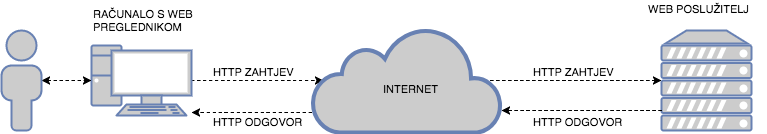
Iako web nije prvi sustav za upravljanje i dijeljenje informacija, jedini je uspješno probio barijere i omogućio dijeljenje informacija između korisnika na različitim računalima, bez obzira na njihovu lokaciju. Berners-Lee navodi kako razvoj weba nije bio toliko težak koliko je teško bilo uvjeriti ljude da ga počnu koristiti, standardizirati protokole i tehnologije. Berners-Lee je koristio svoje osobno NeXT računalo za razvoj weba, HTTP protokola, prve web stranice i web preglednika a isto je računalo i prvi web poslužitelj.

* 1. Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) protokol je aplikacijskog sloja ISO-OSI konceptualnog modela komunikacijskih sustava kojim se najčešće opisuju Internet sustavi. HTTP je definiran kao generički, višestruko primjenjiv, objektno orjentirani protokol bez stanja (eng. stateless). Protokol definira pravila komunikacije distribuiranih sustava za razmjenu informacija odnosno strukturu i semantiku digitalnih poruka (HTTP zahtjeva i odgovora) i temelj je infrastrukture World Wide Weba (Berners-Lee *et al.*, 1997).

Razvoj HTTP-a započeo je Tim Berners-Lee 1989. godine, a danas njegov razvoj nadziru organizacije Internet Engineering Task Force (IETF)[[3]](#footnote-3) i World Wide Web Consortium (W3C)[[4]](#footnote-4). Tim Berners-Lee danas je direktor W3C organizacije. Najraširenija verzija HTTP protokola, verzija HTTP/1.1, prvi je put opisana u dokumentu [RFC 2068](https://tools.ietf.org/html/rfc2068) 1997. godine, a kasnije je zamijenjen dokumentom [RFC 2616](https://tools.ietf.org/html/rfc2616) 1999. godine. Grupa dokumenata [RFC 7230](https://tools.ietf.org/html/rfc7230) (RFC 7230, RFC 7231, RFC 7232, RFC 7233, RFC 7234, RFC 7235) 2014. godine zamijenjuju i nadopunjuju HTTP protokol 2014. godine.

HTTP protokol temelji se na parovima poruka zahtjeva i odgovora. Klijent (najčešće web preglednik) generira zahtjev za željeni resurs (npr. web stranicu, multimedijski resurs i sl.) te u zahtjev uključuje različite metapodatke u obliku zaglavlja HTTP zahtjeva (npr. identifikator preglednika, podržane načine enkodiranja sadržaja, upravljačke direktive za kontrolu pohrane podataka u privremenu memoriju i sl.).



Slika : Princip rada HTTP protokola: zahtjev - odgovor

HTTP definiraju tzv. HTTP metode (npr. GET, POST, HEAD, PUT, DELETE, ...) koje semantički pobliže određuju zahtjev, što je posebno važno kod web servisa (npr. REST web servisi se u svojoj osnovi oslanjaju na HTTP metode). Web je poslužitelj dostupan na web adresi i portu (standardno port 80) te prihvaća HTTP zahtjeve na temelju kojih generira HTTP odgovore. HTTP odgovor sadrži statusni kod koji semantički određuje status obrade zahtjeva, zaglavlja koja sadrže dodatne informacije te tijelo poruke (eng. *message body*) u nekom formatu (npr. HTML, XML, JSON). HTTP statusni kodovi standardizirani su troznamenkasti brojevi raspoređeni u grupe:

* 1xx – Informativni odgovori
* 2xx – Uspješno obrađeni odgovori
* 3xx – Preusmjerenja (eng. redirections)
* 4xx – Greška na strani klijenta
* 5xx – Greška na strani poslužitelja

Brojni se servisi i aplikacije poput web preglednika, web pretraživača i robota za analizu sadržaja oslanjaju na semantiku HTTP statusnih kodova. Primjerice, roboti za indeksiranje sadržaja najpopularnije web tražilice Google različito tretiraju statusne kodove unutar iste grupe statusnih kodova. Tako Google-ovi roboti neće pokušati ponovno dohvatiti sadržaj koji je prethodno poslužen s HTTP statusom 301 (*Moved Permanently*) već će zapamtiti da je preusmjerenje trajno (permanentno). S druge strane, roboti će redovito ponavljati dohvaćanje sadržaja koji je poslužen s HTTP statusom 302 (*Moved Temporarily*). Pravilna uporaba i poštivanje semantike HTTP statusnih osigurava bolju poziciju na rezultatima pretrage, smanjuje rizik od negativnog bodovanja ocjene stranice kao i rizik od neočekivanih ponašanja web preglednika.

U prosincu 2014. godine radna skupina IETF organizacije pod nazivom "*Hypertext Transfer Protocol working group httpbis*" (bis dolazi iz latinskog i znači drugi) predstavila je odboru grupe Internet Engineering Steering Group (IESG)[[5]](#footnote-5) prijedlog prve nove verzije HTTP protokola nakon HTTP 1.1 pod nazivom HTTP/2. IESG je prijedlog odobrio i objavio ga kao predloženi standard 17. veljače 2015. godine. Specifikacija protokola HTTP/2 objavljena je u svibnju 2015. godine u obliku dokumenta RFC 7540. Razvoj HTTP/2 protokola temeljen je na radu Google-ovih inženjera i njihovom eksperimentalnom protokolu SPDY (Belshe, Thomson and Peon, 2014; Nottingham *et al.*, 2015). U kolovozu 2017. godine 16.3% svih web sustava podržava protokol HTTP/2 (W3Techs, 2017d).

Jedna od ključnih novosti koje donosi HTTP/2 protokol jest "*Server Push*" mehanizam koji omogućava poslužitelju posluživanje resursa koje klijent još nije zatražio, a za koje poslužitelj sigurno zna da će biti zatraženi u nekom od idućih HTTP zahtijeva. Novi je protokol u potpunosti kompatibilan s prethodnom verzijom te je moguće migrirati web sustave koji su do sada radili s protokolom HTTP 1.1. na HTTP/2 bez izmjena u kodu samog sustava. Radna skupina navodi sljedeće ciljeve razvoja novog protokola:

* Razvoj mehanizma pregovaranja koji bi klijentu i poslužitelju u omogućio odabir protokola (HTTP 1.1, 2.0 ili drugog protokola)
* Zadržati kompatibilnost s protokolom HTTP 1.1 (HTTP metode, statusne kodove te većinu zaglavlja)
* Optimizirati i poboljšati performanse web sustava kroz uvođenje kompresije HTTP zaglavlja, HTTP/2 "*Server Push*" mehanizam, ulančavanje HTTP zahtjeva, multipleksiranje više HTTP zahtjeva preko jedne TCP veze i druga poboljšanja.
  1. Web sustavi

Web sustavima nazivamo računalne sustave (aplikacije, platforme, stranice i servise) koji se temelje na HTTP protokolu, odnosno one računalne sustave koje resurse distribuiraju kroz web. Ovisno o razini dinamike sadržaja, razni korisničke mogućnosti interakcije sa sustavom, te o minimalno potrebnim tehnologijama za razvoj, web sustave dijelimo na web aplikacije, web servise i web stranice. Tablica u nastavku prikazuje osnovnu podjelu web sustava u navedene kategorije dok se detaljnija kategorizacija nalazi u nastavku rada.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Web stranice** | **Web servisi** | **Web aplikacije** |
| **Tehnologije izrade** | HTML, JavaScript, CSS | Programski jezici za razvoj poslužiteljske strane, HTML, JavaScript, CSS | Programski jezici za razvoj poslužiteljske strane, HTML, JavaScript, CSS |
| **Ciljani korisnici** | Ljudi | Računalni sustavi | Ljudi |
| **Razina moguće interakcije** | Niska | Srednja | Visoka |
| **Primjer** | Prezentacijska web stranica restorana | REST servisi za dohvat podataka vremenske prognoze | Društvena mreža, On line plaćanje |

Tablica : Kategorizacija web sustava

* + 1. Web stranice

Web stranice sastoje se od statičkog sadržaja koji se ne mijenja dinamički već je svaka promjena ručno inicirana (npr. autor web stranice mora ručno promijeniti sadržaj). Web stranice su najraniji i najjednostavniji oblik web sustava kod kojih je korisnička interakcija svedena na minimum – osim pasivnog konzumiranja sadržaja korisnici nemaju mnogo mogućnosti. Takvi su web sustavi uglavnom prezentacijskog karaktera i predstavljaju svojevrsnu digitalnu "posjetnicu" svojih vlasnika. Web stranice bile su iznimno popularne krajem 20. stoljeća. Tehnologije za razvoj statičnih web stranica osnovne su web tehnologije koje bi svaki web programer trebao temeljito poznavati (Flanagan, 2011):

* Cascading Style Sheets (CSS)
* Hypertext Markup Language (HTML)
* JavaScript (JS)

Detaljnije o tehnologijama slijedi u zasebnom poglavlju o web tehnologijama.

* + 1. Web servisi

Web servisima nazivamo one web sustave čiji su primarni korisnici računalni sustavi. Takvi web sustavi sadržaj nužno ne prezentiraju u vizualnom obliku prilagođenom za čovjeka, već u strukturiranim i standardiziranim formatima kao što su XML, JSON i dr. koje drugi računalni sustavi "razumiju". Web servisi najčešće predstavljaju javno dostupni dio nekog zatvorenog sustava (npr. web servisi za rezervaciju karata avioprjevoznika ili sustavi za plaćanja kreditnim karticama) odnosno sučelje za programsku interakciju web sustava i okoline (eng. *application programming interface, API*).

S ciljem povećanja interoperabilnosti međusobno različitih web sustava i servisa razvijeni su i standardizirani protokoli/prakse poput "Simple Object Access Protocol" (SOAP), "Representational state transfer" (REST) i sl.. Takvi standardi apstrahiraju implementacijske razlike web sustava i određuju protokol komunikacije sustava na njihovim programskim sučeljima. Web servisi gotovo su beskorisni bez popratne dokumentacije koja opisuje na koji se način web servisi mogu koristiti, koji su prihvatljivi formati poruka i druga ograničenja web servisa.

Početkom 21. stoljeća razvijen je "Web Service Definition Language" (WSDL), jezik za dokumentiranje i opisivanje dostupnih usluga web servisa. Definicija web servisa u WSDL formatu omogućila je razvojnim programerima korištenje širokog spektra različitih web servisa na jednostavniji način – učitavanjem WSDL definicije u razvojno okruženje programer je udaljene web servise koristio na isti način na koji bi koristio funkcije i procedure razvijene unutar svog web sustava (Christensen *et al.*, 2001).

* + 1. Web aplikacije

Web aplikacije najsloženiji su oblik web sustava koji osim distribucije informacija korisniku pružaju dodanu vrijednost kroz svoje funkcionalnosti. Društvene mreže (eng. *social networks*), sustavi za upravljanje sadržajem (eng. *content management system, CMS*),sustavi za rezervaciju i Internet bankarstvo samo su neki od primjera web aplikacija s kojima je prosječni korisnik weba upoznat. Web aplikacije obično se sastoje od javnog (eng. *public*) dijela koji je javno dostupan te privatnog dijela sustava koji je rezerviran za korisnike s aktivnim korisničkim profilom. Osim autentikacijskog i autorizacijskog sloja, web aplikacije obično implementiraju različite oblike obavijesti korisnika (eng. *notifications*), dvosmjerne kanale komunikacije u obliku razmjena poruka ili diskusije, mogućnost promjene korisničkih postavki i sl..

Web aplikacije često sadrže i do nekoliko različitih web servisa koji u pozadini prikupljaju podatke, obrađuju ih i pripremaju za prezentaciju i korištenje. Uzmemo li za primjer web sustav PayPal, najpoznatiji sustav za digitalni prijenos novca, možemo identificirati dio sustava koji nazivamo web aplikacijom (sučelja za prijavu, upravljanje računima, plaćanja i sl.) te dio sustava koji nazivamo web servisima, a koji na javno objavljenim adresama (eng. *endpoint*) omogućuju programsko korištnje usluga (npr. integraciju plaćanja PayPal-om u druge web aplikacije).

* 1. Evolucijske faze razvoja World Wide Weba

Krajem 20. i početkom 21. stoljeća web je proživljavao svoj vrhunac. Mnoge su kompanije prigrlile ovu tehnološku inovaciju te su sve više implementirale web i srodne tehnologije u svoje poslovanje. Takve su kompanije bile nazivane *dot-coms* (zbog najčešće korištene domene .com, eng. *dot* znači točka). Investitori su bili uvjereni kako su prepoznali obrasce uspješnih poduzeća u ranim fazama te je nastala euforija investiranja u mlade kompanije (eng. *startup*) koje su svoje poslovanje temeljile na webu. U periodu između 2000. i 2002. godine dogodilo se tzv. pucanje *dot-com* balona (eng. *dotcom bubble burst*) kada su na vidjelo počeli izlaziti pravi, ne tako pozitivni, poslovni rezultati tih kompanija. Vrijednosti dionica počele su padati, a velik broj kompanija zauvijek je završio s poslovanjem (Investopedia, 2017).

Brojni stručnjaci debatirali su oko trenutnog stanja weba i njegove sudbine. Dale Dougherty, jedan od pionira u području web tehnologija i jedan od ključnih osoba izdavačke kuće O'Reilly, zastupao je mišljenje kako web nipošto nije "srušen" ili "uništen" već upravo suprotno, web je bio važniji nego ikad, a nove i inovativne aplikacije nastajale su svakog dana. Dougherty je tvrdio kako među kompanijama koje su preživjele pucanje *dot-com* balona imaju neke zajedničke osobine. Stvoren je termin Web 2.0 koji je označavao prekretnicu u razvoju weba ("stara" verzija weba, odnosno tipovi web sustava nazivali su se Web 1.0) i koji je u samo godinu i pol postojanja bio spomenut na Google rezultatima pretrage više od 9.5 milijuna puta. Redovito se održavala konferencija Web 2.0, a uz sam termin vezale su se brojne diskusije jer nije bio potpuno jasno i jednoznačno definiran. Dok su jedni smatrali kako se radi o marketinškom triku, drugi su ga koristili bez pravog razumijevanja (O’Reilly, 2005).

Tim O'Reilly (direktor izdavačke kuće O'Reilly) navodi kako Web 2.0 nema jasne granice te kako nije jednostavno opisati web sustav kao Web 2.0 sustav, već kako se radi o skupu karakteristika web sustava. Web 2.0 sustavi web tretiraju kao platformu za pružanje svojih usluga krajnjim korisnicima, te integriraju usluge drugih Web 2.0 sustava. O'Reilly navodi primjer Internet oglašavanja kao prvog oblika integracije usluga nekoliko Web 2.0 sustava. Razlike Web 1.0 i Web 2.0 sustava autor opisuje na primjeru kompanija Netscape i Google, koji su bili nositelji razdoblja Web 1.0 i Web 2.0 respektivno. Dok je Netscape korisnicima pružao isključivo web preglednik koji je bilo potrebno instalirati na stolno računalo, Google je svoje usluge pružao u obliku web aplikacije odnosno usluge pretraživanja, bez karakteristika tzv. "stare računalne industrije" (npr. kupovina licenci i nadogradnji sustava, zastarjevanje verzija i sl.). Jedna od osnovnih karakteristika Web 2.0 sustava jest tranzicija fokusa sa samog računalnog sustava na podatke. Google kao sustav (tražilica) ne vrijedi mnogo bez podataka koje indeksira i obrađuje, baš kao i drugi primjeri Web 2.0 sustava eBay i Amazon. Sljedeća osnovna karakteristika Web 2.0 sustava jest omogućavanje kreiranje sadržaja krajnjim korisnicima, tako Wikipedia omogućava kolaborativno kreiranje unosa, Google povezuje sadržaj kojima nije vlasnik, a na eBayu korisnici prodaju artikle samostalno. Amazon je poticao i na koncu iskoristio mogućnosti korisničkih osvrta (eng. *user review*) te se na taj način pozicionirao kao izvor kvalitetnih proizvoda. Svi uspješni Web 2.0 sustavi iskoristili su doprinos korisnika kako bi postigli marketinšku dominaciju. Web 2.0 sustavi više pažnje posvetili su korisničkom iskustvu (eng. *user experience, UX*) za razliku od "sirovih" sučelja računalnih sustava "stare škole". Na taj način sustavi su postali dostupni korisnicima s nižom razinom informatičke pismenosti, a računalna i informacijska industrija ozbiljno je počela shvaćati osjećaje koje korištnje sustava izaziva kod korisnika. S tehnološke strane, AJAX (kratica od *Asynchronous JavaScript and XML*) je postao nositelj razvoja novih sučelja i boljeg korisničkog iskustva (O’Reilly, 2005).

Semantički web (Web 3.0) označava korištenje standardiziranih protokola i struktura podataka na webu, kako bi dostupne podatke i informacije mogli koristiti različiti web sustavi sa što većom razinom automatizacije (World Wide Web Consortium (W3C), 2011). Semantički web podrazumijeva pravilno strukturiranje i meta-opisivanje podataka kako bi se isti mogli kategorizirati i klasificirati. On-line zajednica Schema.org[[6]](#footnote-6) jedna je od inicijativa koja potiče korištenje strukturiranih podataka na webu, a najpopularnija tražilica Google bolje rangira web sustave koji koriste strukturirane podatke te ih drugačije vizualizira u rezultatima pretrage. S razvojem semantičkog weba web sustavi će postići veću autonomiju i smaostalno će moći pretraživati i povezivati informacije te korisnicima omogućiti relevantniji sadržaj i ugodnije iskustvo korištenja.

Web 4.0 označava sljedeću fazu evolucije otvorenog, povezanog i inteligentnog weba koji zahtijeva visoku razinu povezanosti korisnika i računala. Razvoj interneta stvari (eng. *Internet of things, IOT*) omogućava prikupljanje velikog broja podataka u realnom vremenu, a Web 4.0 sustavi bit će u mogućnosti "razumjeti" okolinu i uvjete u kojima se korisnik nalazi bez njegove nužne interakcije (Letts, 2015). Virtualni asistenti kao što su Siri tvrtke Apple ili Alexa tvrtke Amazon primjer su Web 4.0 sustava te uključuju visoku razinu praktičnog korištenja umjetne inteligencije i povezivanje različitih servisa (npr. kalendar, vremenska prognoza, planer, pronalazak termina za sastanak, planiranje putovanja i sl.) kako bi što više zadataka mogli obavljati samostalno.

Web 5.0 još je uvijek nedovoljno definirana faza evolucijskog razvoja weba, a podrazumijeva razvoj decentraliziranih web sustava u kojemu su korisnici i web sustavi usko povezani – simbiotički web sustavi (Patel, 2013). Takvi će se web sustavi zasnivat na distribuiranim tehnologijama kao što je block-chain i mogućnostima prepoznavanja emocija korisnika i predviđanju njihovog ponašanja.

Ne postoji jasna granica kojom bi bilo moguće definirati kriterije prema kojima bi se web sustav jednoznačno smjestio u određenu evolucijsku fazu. Jedan web sustav može imati osobine iz nekoliko faza evolucijskog razvoja. Tablica 1 sumarno prikazuje osobine web sustava pojedine faze.

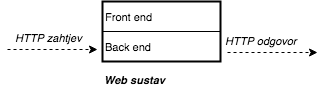
|  |  |  |
| --- | --- | --- |
| **Evolucijska faza** | **Opis i karakteristike** | **Primjer** |
| Web 0 | Rana faza razvoja weba kada je web uglavnom bio korišten od strane znanstvenika. | Razmjena informacija i podataka između znanstvenih institucija |
| Web 1.0 | Tzv. *read-only* web sustavi (stranice) kao digitalne posjednice poduzeća i pojedinaca. Korištenje se svodi na pregledavanje (eng. *browse*) i pasivno konzumiranje. | Web stranice *dot-com* kompanija i *news* portali. |
| Web 2.0 | Fokus je postavljen na korisnike i njihov aktivni angžaman. Kornisici stvaraju i ocjenjuju sadržaj, web sustavi pružaju usluge a ne proizvode. | Wikipedia, Google tražilica i skup alata, društvene mreže, sustavi za ocijenjivanje i kolaboraciju. |
| Web 3.0 | Standardizacija i strukturiranje podataka s ciljem povećanja razine autonomije web sustava. Web sustavi međusobno mogu komunicirati i razmjenjivati podatke. | Schema.org oznake i metapodaci, Resource Description Framework (RDF). |
| Web 4.0 | Povećano korištenje umjetne inteligencije, prepoznavanje obrazaca ponašanja, predviđanje korisničkih emocija i navika. Web sustavi su u mogućnosti prepoznati kontekst sadržaja te poduzimati akcije (npr. stvaranje podsjetnika temeljem sadržaja email poruke) | Virtualni asistenti, povezivanje različitih servisa (kalendar, planer, kontakti). |
| Web 5.0 | Simbiotički web sustavi koji su uvijek aktivni (eng. *always on*) i aktivno nadziru korisnika i njegovo ponašanje s ciljem predviđanja i neprimjetnog obavljanja potrebnih zadataka. | Virtualna i proširena stvarnost, distribuirani sustavi i tehnologije kao što je *block chain* |

Tablica : Evolucijske faze razvoja World Wide Weba

* 1. Tehnologije za razvoj web sustava

Mnogi su programski jezici kroz povijest proživljavali svoj rani razvoj, procvat i propadanje, a tehnološke inovacije u velikoj su mjeri određivale životni tijek i sudbinu programskih jezika. Web je kao jedna takva tehnološka inovacija, osim omogućavanja razvoja sasvim nove vrste računalnih sustava, unaprijedio načine komunikacije, razmjene informacija, diskusije i razvoja tehnologije. Resursi za samostalno svladavanje vještine programiranja te primjeri programskog koda i platforme za diskusiju o konkretnim programskim problemima postali su dostupni svima, a programiranje kao struka u potpunosti je liberalizirana. Pojavom softvera otvorenog koda (eng. *open source software, OSS*) i stvaranjem *online* zajednica (eng. *online community*) oko pojedinih tehnologija nastavljena je popularizacija IT struke i programiranja, a samim time i računalnih tehnologija. Danas, 28 godina od rođenja web-a, web tehnologije apsolutno dominiraju IT svijet kada je u pitanju popularnost, a zanimanja u IT industriji najpopularnija su na tržištu (CareerProfiles, 2017; Stackify, 2017).

Razvoj funkcionalnih web sustava podrazumijeva korištenje nekoliko tehnologija: tehnologije za razvoj web sustava na strani poslužitelja (eng. *frontend*), tehnologije za razvoj web sustava na strani poslužitelja (eng. *backend*) te poznavanje tehnologija za postavljanje i konfiguraciju web poslužitelja.



Slika : Shematski prikaz web sustava

U nastavku poglavlja detaljnije su opisane tehnologije za razvoj web sustava na strani poslužitelja i na strani klijenta.

* + 1. Razvoj web sustava na strani klijenta

Tehnologije za razvoj web sustava na strani poslužitelja (u nastavku frontend tehnologije) u početku su isključivo služile za prezentaciju podataka (statičnih ili pripremljenih od strane poslužitelja). Frontend tehnologije danas omogućavaju razvoj potpuno funkcionalnih web aplikacija na strani poslužitelja (tzv. progresivne web aplikacije). Frontend se temelji na tri osnovne tehnologije razvoja web sustava: HTML, CSS i JavaScript.

Hypertext Markup Language (HTML) semantički je jezik sličan XML-u koji služi definiranju strukture web dokumenata. Iako često krivo nazivan HTML nije programski jezik jer svojim funkcionalnostima ne omogućava razvoj dinamičnih aplikacija koji mogu biti u interakciji s korisnicima ili drugim sustavima. Osnovni gradivni elementi HTML jezika su HTML elementi (tagovi), a njihovim pravilnim povezivanjem nastaju složeni web dokumenti. HTML elementi mogu biti prezentacijski (npr. img, a, div tagovi) ili opisni (npr. meta, title, link tagovi) koji služe opisivanju dokumenta ili uključivanju resursa (npr. uključivanje CSS datoteka) ili semantičkom definiranju sadržaja. Standardno se koristi HTML verzije 5 (HTML 5) koji je uveo nove semantičke i grafičke elemente, pojednostavio sintaksu i ispravio nepravilnosti prethodne verzije.

Cascading Style Sheets (CSS) jezik je kojim se definira način prezentacije HTML (ili XML i srodnih) dokumenata na različitim medijima (zaslon, papir, čitači zaslona). Dok se HTML-om definira isključivo struktura web dokumenta, CSS je tehnologija koja web dokumentu daje vizualnu komponentu. Standardno se koristi CSS verzije 3 (CSS 3) koji je uveo nove vizualne mogućnosti, transformacije elemenata, animacije i prijelaze (eng. *transitions*). U razvoju modernih web sustava koriste se i tzv. CSS preprocesori (npr. SCSS) koji pojednostavljuju proces razvoja web sustava, uvode naslijeđivanje klasa, varijable i druge koncepte. CSS je standardiziran i web preglednici redovito ugrađuju podršku za postojeća i nova pravila.

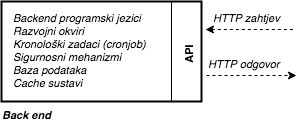
JavaScript (JS) je programski jezik visoke razine koji uvodi dinamiku u statične web dokumente. JavaScript omogućava manipulaciju HTML elementima, interakciju s korisnikom, rad s multimedijom, mrežni rad (npr. web socketi), a u zadnje vrijeme i sve više interakcije s hardverom (npr. senzori mobilnih uređaja, kamera, mikrofon i sl.) ovisno o okolini u kojoj se izvršava. JavaScript je u početku bio ugrađen samo u web preglednike, a u zadnje su vrijeme mogućnosti za rad s JavaScriptom proširene pa je tako moguće razvijati i backend dio web sustava isključivo u JavaScript programskom jeziku (npr. s Node JS platformom) ili programska proširenja za druge web i desktop aplikacije ili operacijske sustave. S obzirom da je JavaScript dostupan na gotovo svim web preglednicima (na stolnim i prijenosnim računalima), nositelj je tzv. hibridnog načina razvoja mobilnih aplikacija koji koristi web tehnologije za izgradnju mobilnih aplikacija. Razvojem i uvođenjem naprednih koncepata u JavaScript programski jezik nestala je potreba za drugim tehnologijama i možemo reći kako ih je JavaScript "izbacio iz igre" (npr. Flash, Java applets). JavaScript je danas najpopularniji programski jezik. U ovom radu promatrat će se isključivo frontend aspekt JavaScript programskog jezika.

Arhitekti web sustava odabiru koliki će dio poslovne logike prebaciti s poslužitelja na klijent te u skladu s time definiraju zahtjeve za frontend tehnologijama. Razvoj tzv. *single-page* aplikacija (eng. *single-page application, SPA*)omogućava stvaranje potpuno funkcionalnih web sustava u obliku jednog web dokumenta i iskustvom korištenja sličnom desktop aplikacijama (npr. bez kretanja između različitih web dokumenata), kod kojih je sva logika smještena na strani klijenta, a backend dio sustava služi isključivo za dohvaćanje podataka iz baze podataka (najčešće u JSON formatu). Tehnološki lideri, kompanije Facebook i Google, prepoznali su prednosti ovog pristupa kod razvoja dinamičnih web sustava s visokim prometom te su čak i razvili vlastite metodologije i razvojne okvire (React i AngluarJS).

* + 1. Razvoj web sustava na strani poslužitelja

Tehnologije za razvoj web sustava na strani poslužitelja (u nastavku backend tehnologije) osnova su razvoja web servisa i web aplikacija. Upravo backend tehnologije omogućavaju razvoj funkcionalnosti koje se temelje na radu s bazom podataka, pozadinskim servisima i asinkronom i/ili paralelnom obradom podataka, strojnim učenjem i sl.. Razvoj web sustava drugačiji je u odnosu na ostale računalne sustave zbog svoje *stateless* prirode - svaki HTTP zahtjev postoji neovisno o prethodnima, sustavi se ne mogu oslanjati na kontekst, a svaki se HTTP zahtjev pojedinačno obrađuje.

Web sustav zaprima HTTP zahtjev, obrađuje ga te generira HTTP odgovor. Moderni programski jezici za razvoj backend dijela web sustava apstrahiraju obradu izvornog ("sirovog") HTTP zahtjeva već ga internim mehanizmima transformiraju u objekt. Takav objekt sa sobom nosi atribute HTTP zahtjeva kao što su zaglavlja, tijelo poruke, kolačići i sl. koji čine ulazni skup podataka web sustava. Web sustav tada može primijeniti poslovna pravila, manipulirati resursima (npr. zapisima u bazi podataka ili datotekama na disku) i generirati HTTP odgovor. Moderne backend tehnologije pružaju bogat API za rad s HTTP zahtjevima i odgovorima stoga su implementacijski detalji i izvorni tekstualni format skriveni od programera. Javne pristupne točke backend dijela web sustava nazivamo *API endpoint-*ima.



Slika : Shematski prikaz backend dijela web sustava

HTTP protokol jednoznačno je definiran i temelji se na porukama u tekstualnom i čitljivom formatu. Pažljivi će čitatelj zaključiti kako tehnologija za razvoj backend dijela web sustava može biti bilo koji programski jezik koji ima mrežnu komponentu - ako programski jezik omogućava stvaranje TCP socketa, moguće je realizirati web sustav. Ne čudi stoga raznolikost i broj dostupnih programskih jezika za razvoj backend dijela web sustava, a izbor konkretnog programskog jezika najčešće ovisi o preferencijama programera te specifičnim zahtjevima web sustava. U nastavku je prikaz 15 najpopularnijih programskih jezika za razvoj web sustava na strani poslužitelja (prema broju predloženih izmjena, eng. *pull request*), napravljen prema sumarnim podacima najvećeg *open-source* Git repozitorija Github.com (*Github Language Stats*, 2017).

|  |  |
| --- | --- |
| **R.br.** | **Programski jezik** |
| 1 | JavaScript |
| 2 | Python |
| 3 | Java |
| 4 | PHP |
| 5 | Ruby |
| 6 | Go |
| 7 | C# |
| 8 | TypeScript |
| 9 | Scala |
| 10 | Rust |
| 11 | CoffeeScript |
| 12 | Haskell |
| 13 | Lua |
| 14 | Perl |
| 15 | Elixir |

Tablica : 15 najpopularnijih programskih jezika često korištenih za razvoj web sustava na strani poslužitelja (prema broju pull request unosa)

Iako u zadnje vrijeme i frontend tehnologije sve više koriste vanjske servise (npr. baza podataka, autorizacijski mehanizmi i sl.), za koordinaciju i rad sa servisima najčešće je zadužen backend. Optimizacija performansi i sigurnosti web sustava u najvećoj se mjeri tiče optimizacije backenddijela web sustava.

1. PHP programski jezik
   1. Razvoj PHP programskog jezika

Današnji programski jezik PHP (PHP je rekurzivni akronim od eng. *Hypertext Preprocessor*) naslijednik je jezika PHP/FI (eng. *Personal Home Page/Forms Interpreter*). Rasmus Lerdorf 1994. godine razvio je prvu verziju PHP/FI jezika kako bi javno objavio, i pratio posjete svom digitalnom životopisu te je skupinu programskih skripti grupirao pod nazivom "*Personal Home Page Tools*" ili skraćeno "*PHP Tools*". Lerdorf je nastavio s razvojem svojih skripti te je u lipnju 1995. javno objavio izvorni kod prve verzije PHP-a s ciljem popularizacije novostvorenog programskog jezika, ali i suradnje te poboljšanja koje bi eventualno drugi programeri izradili (The PHP Group, 2017a). Sam Lerdorf izjavio je kako u svemu tome nije planirao razvoj novog programskog jezika:

"*Nisam znao kako stati, nije bilo namjere razviti novi pgorgramski jezik... Niti ne znam kako razviti programski jezik, samo sam dodavao elemente koji su mi se logički činili potrebnima*" (IT Conversations, 2003).

PHP je već u svojim prvim godinama doživio nekoliko velikih revizija i izmjena iz temelja, ubrzo su dodane funkcionalnosti potrebne za komunikaciju s bazom podataka, a već krajem 1998. godine PHP je brojao nekoliko tisuća korisnika u svijetu. Istraživanje Netcraft-a pokazalo je kako je u svibnju 1998. godine više od 60 000 ili 1% ukupnog broja web stranica na svijetu bilo pokretano na poslužiteljima koji podržavaju PHP (The PHP Group, 2017a).

PHP verzija 3.0 najbliža je PHP-u kakvog danas poznajemo. Programeri Andi Gutmans i Zeev Suraski radili su na razvoju sustava elektronske trgovine (eng. *ecommerce*) baziranog na PHP-u te su uočili brojne nedostatke prethodne verzije jezika PHP/FI 2.0. Kontaktirali su Lerdorfa i digitalnim kanalima raspravljali o poboljšanjima na kojima su Gutmans i Suraski već radili. Konačno, dogovorena je suradnja trojice programera te su zajedno započeli rad na razvoju novog i neovisnog programskog jezika pod novim nazivom "*PHP: Hypertext Preprocessor*" kako bi se rješili ograničavajućeg prizvuka korištenja isključivo za osobne potrebe. PHP 3.0 privukao je brojne programere koji dali svoj doprinos razvojem različitih programskih modula. Na vrhuncu popularnosti verziju PHP 3.0 podržavalo je približno 10% svih web poslužitelja na svijetu (The PHP Group, 2017a).

U zimu 1998. godine, nedugo nakon službene objave PHP 3.0, Gutmans i Suraski započeli su rad na poboljšanju jezgre PHP-a s ciljem optimizacije performansi kompleksnih aplikacija te poboljšanje modularnosti izvornog koda PHP-a. Razvili su novi sustav koji je pokretao PHP (eng. engine) i nazvali ga "*Zend Engine*" (prema kombinaciji imena autora, Zeev i Andi). PHP 4.0 temeljen na novom sustavu "*Zend Engine*" objavljen je u svibnju 2000. godine a nova je verzija podržavala različite web poslužitelje, HTTP sesije, nove programske konstrukte te poboljšane sigurnosne mehanizme (The PHP Group, 2017a).

PHP 5 objavljen je u srpnju 2004. godine nakon nekoliko iteracija razvoja i objavljenih pred-verzija. Nova je verzija PHP-a bila pokretana sustavom "*Zend Engine 2.0*" (The PHP Group, 2017a). Nova je verzija programskog jezika donijela brojne funkcionalnosti i poboljšala podršku za objektno-orjentirano programiranje u PHP-u, "PHP Data Objects" (PDO) ekstenzija omogućila je jednostavnije i sigurnije pristupanje bazama podataka a performansna poboljšanja omogućila su razvoj kompleksnih web sustava u PHP-u (Trachtenberg, 2004). U kolovozu 2014. godine objavljena je verzija PHP 5.6 koja je donijela značajna poboljšanja i pristupačniju sintaksu a u kolovozu 2017. godine PHP 5.6 je najkorištenija podverzija PHP programskog jezika verzije 5 s gotovo 30% tržišnog udjela (W3Techs, 2017a). PHP verzija 5 koristi se na 91,9% web sustava koji koriste PHP (W3Techs, 2017b).

PHP je često bio na meti kritika zbog nedostatka podrške za *unicode* način enkodiranja tekstualnih podataka (PHP je podržavao samo tzv. *byte strings*). Andrei Mievski započeo je rad na uvođenju nativne podrške za *unicode* način enkodiranja 2005. godine kroz ugradnju "International Components for Unicode" (ICU) biblioteke i enkodiranje tekstualnih podataka UTF-16 načinom kodiranja. Značajnost promjene koju bi objava nove verzije programskog jezika donijela nalagala je da se izmjena uvede kroz novu verziju programskog jezika – PHP 6 (tzv. *major release*). Nedostatak programera koji su prepoznali važnost predloženih promjena i performansni problemi nastali prilikom promjene načina kodiranja tekstualnih podataka doveli su do zastoja u razvoju što je rezultiralo objavom verzije PHP 5.3 2009. godine koja je donijela funkcionalnosti prethodno planirane za objavu u sklopu verzije PHP 6. Razvoj verzije PHP 6 napušten je u ožujku 2010. godine a verzija PHP 5.4 donijela je preostale funkcionalnosti planirane za napuštenu verziju (Zmievski, 2005, 2011).

Tijekom 2014. i 2015. godine razvijena je nova značajna (eng. *major*) verzija PHP programskog jezika – PHP 7. Imenovanje nove verzije uzrokovalo je brojne debate. Postojale su dvije struje, jedna koja se zalagala za objavu nove verzije pod nazivom PHP 6 budući da nije postojala službena verzija s tim nazivom i druga koja se zalagala za preskakanje verzije i objavu nove verzije programskog jezika pod nazivom PHP 7. Radna verzija PHP 6 iako nikada nije bila javno objavljena, bila je dostupna u repozitoriju koda i poznata (i očekivana) u programerskim krugovima. Zanimljiva je činjenica da su izdavači stručnih knjiga već objavili nekoliko izdanja koja su referencirala PHP verziju 6, što bi objavom nove verzije pod nazivom PHP 6 stvorilo nesporazum. Na kraju je za službeni naziv verzije odabran PHP 7 a diskusija oko konačne odluke traje i danas (Sturgeon, 2014). Tim programera Dmitry Stogov, Xinchen Hui i Nikita Popov razvili su novu verziju programskog jezika sa značajno poboljšanim performansama koje su postigli doradom sustava "Zend Engine". Performnansa mjerenja (eng. *benchmark*) pokazala su 100% poboljšanje performansi u novoj verziji PHP 7 u odnosu na prethodnu verziju. Izmjene u sustavu "Zend Engine" objavljene su pod nazivom "Zend Engine 3" koji je naslijedio "Zend Engine 2" korišten u verziji PHP 5. Osim performansnih poboljšanja i dorada izvornog koda, PHP 7 donio je nove programske konstrukte u PHP programski jezik kao što su definiranje povratnog tipa podataka funkcije i definiranje tipa podataka argumenta funkcija (Stogov and Suraski, 2014). PHP verzija 7 koristi se na 7,2% web sustava koji koriste PHP (W3Techs, 2017b)

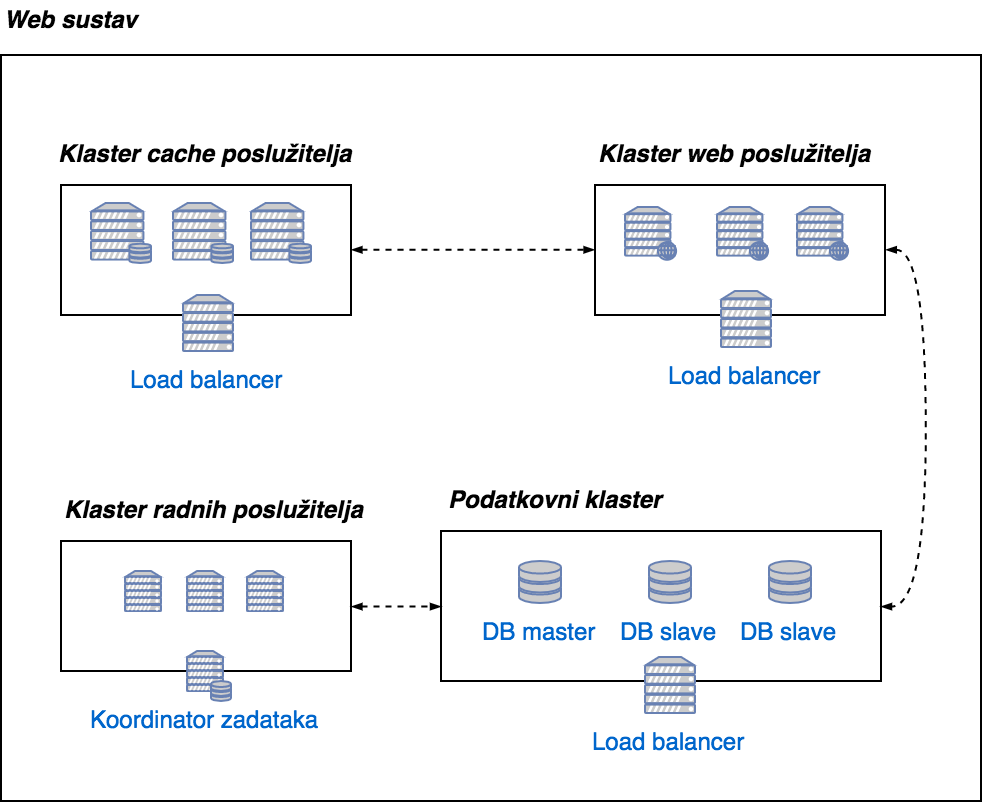
U nastavku je napravljen pregled do sada objavljenih verzija programskog jezika PHP s kratkim opisom novosti pojedine verzije te stanjem podrške za pojedinu verziju.

|  |  |  |  |
| --- | --- | --- | --- |
| **Verzija** | **Datum objave** | **Podrška do** | **Opis** |
| ***Legenda****: ■ Neaktivno izdanje, ■ Stabilno izdanje s doradama sigurnosnih propusta, ■ stabilno izdanje s doradama bugova i sigurnosnih propusta, ■ planirano (buduće izdanje)* | | | |
| 1.0 ■ | 8. lipnja 1995. | *Nema podataka* | Prvi puta korišten naziv PHP – službeni naziv: "Personal Home Page Tools (PHP Tools)" |
| 2.0 ■ | 1. studenoga 1997. | *Nema podataka* | Službeni naziv "PHP/FI 2.0". Prva verzija samostalnog programskog jezika u punom smislu, s funkcionalnostima od kojih su neke zadržane do danas |
| 3.0 ■ | 6. lipnja 1998. | 20. listopada 2000. | Proširenje razvojnog tima - Zeev Suraski i Andi Gutmans dorađuju temelje nove verzije |
| 4.0 ■ | 22. svibnja 2000. | 23. lipnja 2001. | Dodan je sustav obrade i izvođenja – tzv. Zend engine |
| 4.1 ■ | 10. prosinca 2001. | 12. ožujka 2002. | Uvedene su superglobalne varijable |
| 4.2 ■ | 22. travnja 2002. | 6. rujna 2002. | Sigurnosna poboljšanja |
| 4.3 ■ | 27. prosinca 2002. | 31. ožujka 2005. | Uvedeno je konzolno sučelje CLI |
| 4.4 ■ | 11. srpnja 2005. | 7. kolovoza 2008. | Ispravljen je problem u radu s memorijom |
| 5.0 ■ | 13. srpnja 2004. | 5. rujna 2005. | Uveden je sustav Zend Engine 2 s novim objektnim modelom |
| 5.1 ■ | 24. studenoga 2005. | 24. kolovoza 2006. | Performansna poboljšanja, uvođenje PHP Data Objekata (PDO) kao konzistentnog sučelja za rad s bazama podataka |
| 5.2 ■ | 2. studenoga 2006. | 6. siječnja 2011. | Nativna podrška za JSON |
| 5.3 ■ | 30. lipnja 2009. | 14. kolovoza 2014. | Podrška za *namespace* kasno statičko povezivanje (eng.  *late static binding*), anonimne funkcije, PHP arhive i druge dorade. |
| 5.4 ■ | 1. ožujka 2012. | 3. rujna 2015. | Podrška za *trait* mehanizam, sintaksne dorade, ugrađeni web poslužitelj, performansna poboljšanja. |
| 5.5 ■ | 20. lipnja 2013. | 21. srpnja 2016. | Uvedeni su generatori, poboljšano je upravljanje iznimkama |
| 5.6 ■ | 28. kolovoza 2014. | 31. prosinca 2018. | Poboljšane konstante, funkcije s neograničenim brojem argumenata, operator potenciranja, novi razvojni alati i druge dorade. |
| 6.x | *Nije objavljeno* |  | Napuštena verzija koja je trebala uvesti podršku za unicode način kodiranja |
| 7.0 ■ | 3. prosinca 2015. | 3. prosinca 2018. | Uveden je novi sustav Zend Engine 3 sa značajnim poboljšanjem performansi, novom sintaksom, novim operatorima i drugim doradama. |
| 7.1 ■ | 1. prosinca 2016. | 1. prosinca 2019. | Uvođenje nedefiniranog (eng. *void*) povratnog tipa, bolje upravljanje iznimkama i druge dorade. |
| 7.2 ■ | 30. studenoga 2017 | 30. studenoga 2020. |  |

Tablica : Povijest verzija programskog jezika PHP prema (The PHP Group, 2017b, 2017c).

* 1. Razvoj suvremenih web sustava u PHP programskom jeziku

Suvremeni se web sustavi najčešće sastoje od javnog (eng. *public*) i privatnog (eng. *private*) dijela. Javni je dio web sustava dostupan svim korisnicima, a najčešće sadrži početnu stranicu (eng. *home page, landing page*), opisne stranice te sučelja za registraciju i prijavu. Nakon uspješne prijave, korisnicima je omogućen pristup privatnom dijelu web sustava, uz ograničenja definirana korisničkim ulogama. Performanse web sustava degradiraju se s vremenom u ovisnosti o broju korisnika, intenzitetu korištenja, količini sadržaja i geolokacijskom rasporedu korisnika. Web sustavi najčešće su smješteni na nekoliko web poslužitelja, poslužitelji baza podataka su replicirani, podaci se pohranjuju u poslužitelje za privremenu pohranu podataka grupirane u klaster, statičke se datoteke poslužuju kroz mrežu distribuiranih poslužitelja za isporuku sadržaja (eng. *content delivery network, CDN*) a u slučajevima visoke posjećenosti (eng. *peak*) sustavi se mogu automatski skalirati i osiguravati razinu usluge. Dijagram u nastavku prikazuje primjer arhitekture suvremenog web sustava.



Slika : Primjer arhitekture suvremenog web sustava

Suvremeni se web sustav sastoji od nekoliko logičkih grupa (klastera) manjih sustava: Web poslužitelji (eng. *web servers*), poslužitelji baze podataka (eng. *database servers, DB servers*), poslužitelja za privremenu pohranu podataka (eng. *cache servers*) te sustava za obradu radnih zadataka (eng. *job servers*). Svaki se klaster sastoji od gotovo identičnih jedinica te nadzorne jedinice zadužene za koordinaciju i raspored prometa (eng. *load balancer*). Tako primjerice load balancer klastera web poslužitelja raspoređuje HTTP zahtjeve na odgovarajući web poslužitelj (npr. prema kriteriju prostornog razmještaja odabire se najbliži poslužitelj kako bi se maksimizirala brzina posluživanja), a load balancer cache klastera temeljem ključa (eng. *cache key*) pronalazi poslužitelj koji čuva traženi podatak. Koordinator zadataka radnih poslužitelja raspoređuje poslove radnih poslužitelja te upravlja i ažurira popis preostalih zadataka za obradu. Suvremeni su web sustavi sastavljeni od manjih nezavisnih sustava.

Suvremeni sustav baziran na PHP programskom jeziku može biti sastavljen primjerice od Apache ili Nginx web poslužitelja, Nginx ili HAProxy web load balancera, Redis ili Memcache poslužitelja za povremenu pohranu podataka, te MySQL ili Postgre SQL baze podataka. Budući da su podsustavi u potpunosti autonomni, razdvojeni (eng. *decoupled*) i modularni, moguće je kombinirati različite tipove podsustava te ih u vremenu mijenjati bez nužne prilagodbe ostatka sustava. Upravo ta osobina suvremenih web sustava omogućava inkrementalan razvoj – u početku nije potrebno uvoditi sve elemente (npr. web sustav može biti sastavljen od jednog web poslužitelja i jednog poslužitelja baze podataka, bez radnih poslužitelja i poslužitelja za privremenu pohranu podataka), već se oni uvode kada se za to javi potreba odnosno kada performanse web sustava takve arhitekture padnu ispod zadovoljavajuće razine.

* + 1. Tržišni udio i popularnost PHP programskog jezika

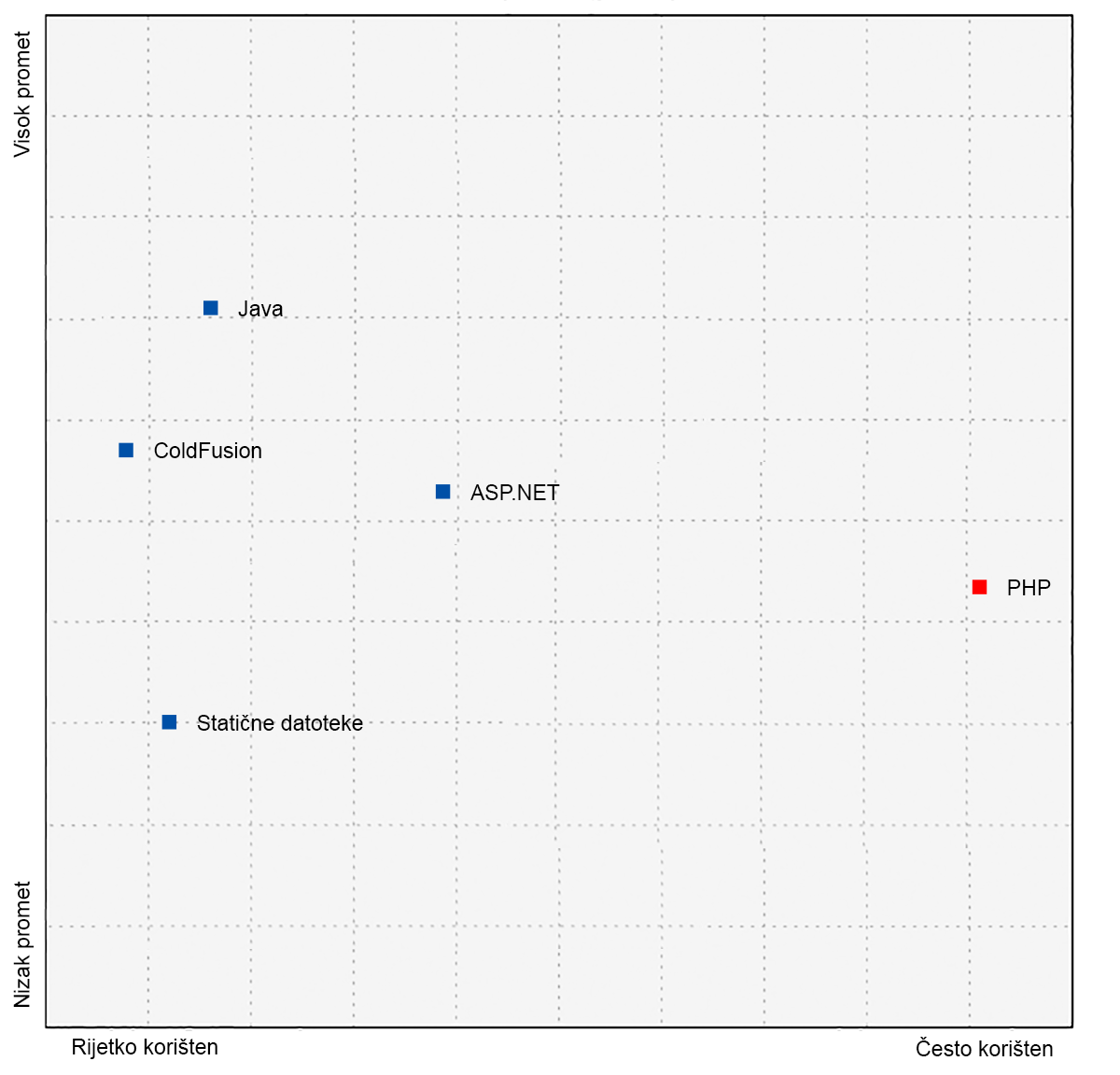
Najpopularnija društvena mreža Facebook u svojim je početcima bila u potpunosti bazirana na PHP programskom jeziku (zbog svog viralnog rasta i velike posjećenosti Facebook je kasnije uveo i razvio brojne tehnologije i poboljšanja PHP programskog jezika), Yahoo i Wikipedia su najpoznatiji primjeri suvremenih web sustava baziranih na PHP-u, a najpopularnija platforma za objavu sadržaja (eng. *blogging platform*) i sustav za upravljanje sadržajem (eng. *content management system, CMS*) WordPress (i wordpress.com) u potpunosti su bazirani na PHP-u. Popularni web sustavi s visokim prometom poput VKontakte (2,2 milijarde posjeta mjesečno) i XVideos (1,6 milijarde posjeta mjesečno) uspješnos u razvijeni na PHP programskom jeziku i komplementarnim tehnologijama (SimilarTech, 2017).

Suvremeni sustavi vrlo su složeni i najčešće koriste desetke različitih programskih jezika i tehnologija u svom radu, s ciljem optimizacije performansi, stabilnosti i sigurnosti. Najnoviji trendovi u razvoju web sustava, razvoj sustava baziran na mikroservisima, odlazi i korak dalje te razdvaja pojedine funkcionalnosti web sustava na neovisne i autonomne web servise koji međusobno komuniciraju isključivo programski, gdje jedna funkcionalnost web sustava može biti razvijena u potpuno drugom skupu tehnologija od ostatka web sustava (Richardson, 2017).

Programski jezik PHP često je na udaru kritičara kao "loš i zastario programski jezik" koji "nije prikladan za razvoj ozbiljnih web sustava". Najčešće mu se zamjera manjak konzistentnosti u sintaksi (funkcije su ponekad imenovane bez razmaka, ponekad se koristi podvlaka, a ponekad su pisani *camel case* stilom), neočekivano upravljanje tipovima podataka (npr. neočekivana konverzija tipa podataka prilikom korištenja operatora ==), ovisnost o konfiguraciji te loše upravljanje greškama i iznimkama (eng. *error and exception handling*) (Eevee, 2012). PHP zajednica i razvojni tim aktivno rade na ispravljanju grešaka i nedostataka programskog jezika, a neki su članovi zajednice mišljenja kako je PHP na lošem glasu zbog svoje jednostavnosti odnosno mogućnosti da bilo tko krene razvijati web sustave u PHP programskom jeziku, pa od tamo dolazi mnoštvo lošeg programskog koda. Najveći pomak u razvoju PHP programskog jezika napravljen je s verzijom 7 i uvođenjem kontrole tipova podataka, performansna i sintaksna poboljšanja. Prethodno navedeni primjeri web sustava razvijenih u PHP programskom jeziku dokazuju kako je apsolutno moguće razviti složene web sustave u PHP-u. U trenutku pisanja ovog rada PHP je najčešće korišten programski jezk za razvoj web sustava na strani poslužitelja, s tržišnim udjelom od 82,8% ispred tehnologija ASP.NET (14,19%), Java (2,6%) te statičkih datoteka (1,5%). Programski jezici ColdFusion, Ruby, JavaScript (na strani poslužitelja), Perl, Python te Erlang zastupljeni su u iznosu manjem od 1% (W3Techs, 2017c).

Graf : Tržišni udio programskih jezika na strani poslužitelja   
prema (W3Techs, 2017c)

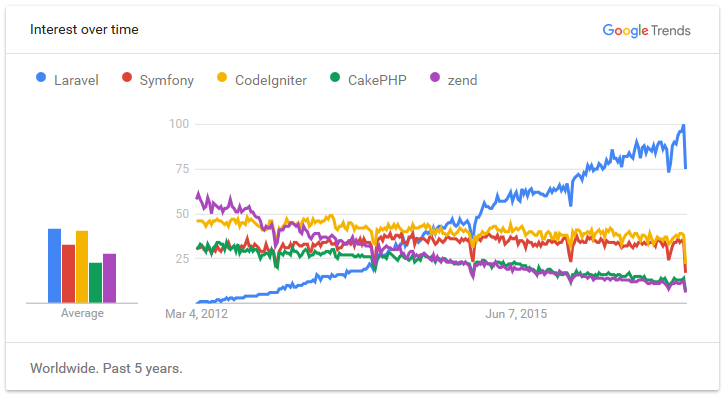
Prema kriteriju popularnosti mjerenom u broju pretraga s ključnim riječima pojedine tehnologije tj. programskog jezika, PHP zauzima 7. mjesto nakon Java, C, C++, C#, Python i Visual Basic .NET programskih jezika (TIOBE, 2017). Iako očito nije najpopularniji izbor za razvoj web sustava na strani poslužitelja, PHP je i dalje jedan od ključnih programskih jezika prema broju web sustava koji ga koriste. Neki autori navode kako je teže nabrojati popularne web sustave koji ne koriste PHP (barem u nekoj mjeri) nego one koji ga koriste (MacKay, 2016). Dvodimenzionalna usporedba programskih jezika prema kriterijima količine prometa i broju web sustava pokazuje kako je PHP programski jezik često korišten od strane web sustava sa srednjom količinom prometa (W3Techs, 2017b).



Graf : Usporedba programskih jezika prema količini prometa i učestalosti korištenja   
prema (W3Techs, 2017b)

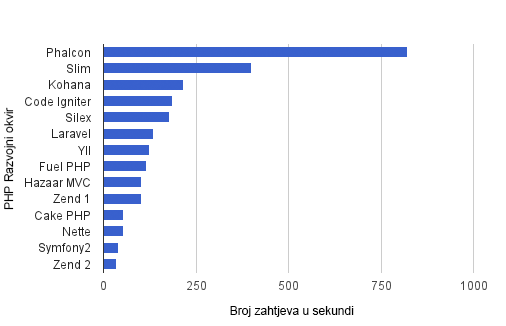
* + 1. Razvojni ekosustav i zajednica

U razvoju računalnih sustava (softvera) sve je više popularno korištenje razvojnih okvira (eng. *framework*)i/ili biblioteka programskog koda (eng. *library*) koji pojednostavljuju razvoj, apstrahiraju različite segmente rada računalnog sustava kao što je rad s bazom podataka, datotekama ili mrežnim slojem te na korištenje pružaju gotove funkcionalnosti. Phalcon, Laravel, Symfony i Yii samo su neki od razvojnih okvira za programski jezik PHP, oko kojih se okupila velika zajednica programera i koji aktivno rade na njihovom razvoju i održavanju te na razvoju novih modula. Laravel razvojni okvir u posljednjih je 5 godina postao najpopularniji zbog svoje jednostavnosti, kvalitetne dokumentacije i lakoće korištenja (coderseye.com, 2017).



Slika : Popularnost PHP razvojnih okvira   
(preuzeto s https://coderseye.com/best-php-frameworks-for-web-developers/, 28.8.2017)

Najvažnije osobine razvojnog okvira tiču se njegovih performansi. Propusnost definira broj zahtjeva u jedinici vremena najbolje opisuje performanse backend dijela web sustava. Phalcon razvojni okvir ima najbolju propusnost (približno 760 zahtjeva u sekundi), slijede ga razvojni okviri Slim (približno 330 zahtjeva u sekundi) te Kohana (približno 230 zahtjeva u sekundi (coderseye.com, 2017). Razvojni okviri Symfony 2 i Zend, najčešće korišteni za razvoj složenih web sustava (tzv. *enterprise* web sustavi), imaju najlošiju propusnost.



Graf : Propusnost PHP razvojnih okvira   
prema (coderseye.com, 2017)

Tehnologije otvorenog koda (eng. *open source*) temelje se na javno dostupnim komponentama (bibliotekama koda), najčešće objavljenima u obliku Git repozitorija. Alat Composer omogućava jednostavno upravljanje potrebnim bibliotekama (eng. *dependency management*) na sličan način na koji to rade alati Bundler za Ruby programski jezik odnosno npm za JavaScript programski jezik (Composer, 2017). Composer je osnova razvoja modernih PHP aplikacija, a mehanizam automatskog pronalaska i uključivanja klasa propisan PSR standardom (eng. *autoload*) dolazi ugrađen u alat. Composer omogućava definiranje verzija željenih biblioteka te ih automatski preuzima s njihovih Git repozitorija, registrira u interni mehanizam otkrivanja i omogućava njihovo korištenje. Javno dostupni Composer paketi objavljeni su u sustavu Packagist[[7]](#footnote-7). Composer koristi mehanizme Git sustava za verzioniranje koda kako bi osigurao da će točno definirana verzija zahtjevane biblioteke biti preuzeta s repozitorija (*commit hash*, potpis revizije). Symfony razvojni okvir verzije 3 i Doctrine ORM (eng. *Object-relational mapping*) jedna su od nekoliko najpopularnijih kombinacija odnosno par alata za razvoj PHP aplikacija (najpopularniji par je Laravel razvojni okvir i Eloquent ORM) i bit će korišteni u praktičnom dijelu ovog rada.

1. Performanse web sustava

Osnovna je svrha svih web sustava prihvaćanje i obrada HTTP zahtjeva te generiranje HTTP odgovora, odnosno posluživanje resursa koji su bili zathjevani. Brzina kojom web sustav može obraditi korisnički zahtjev jedna je od najvažnijih njegovih karakteristika. Web sustav koji brže obrađuje zahtjeve može podnijeti veći promet i ne zahtijeva značajna ulaganja u infrastrukturu kada dođe do povećanja prometa. Ukoliko web sustav doživi ekspanziju i prosječan se broj korisničkih zahtjeva u minuti (eng. *request per minute, RPM*) značajno poveća, potrebno je aktivno nadzirati i optimizirati sustav, ukloniti uska grla, poboljšati infrastrukturu i na druge načine poboljšati performanse kako bi web sustav mogao služiti svojoj svrsi. Navedene aktivnosti nazivamo skaliranjem web sustava, a kako bi se performanse web sustava mogle poboljšati, prvo ih je potrebno identificirati i izmjeriti.

U nastavku poglavlja postavljena je ljestvica za kategorizaciju web sustava prema posjećenosti, opisane su performansne karakteristike web sustava uz primjere metrika za svaku o performansnih karakteristika.

* 1. Web sustavi visokih performansi

Za razliku od aplikacija na stolnim računalima (eng. *desktop applications*) koji su smješteni (instalirani) na korisničkom računalu, web sustavi podatke prenose s (često i stotinama kilometara) udaljenih poslužitelja. Korisnici ne mare previše za infrastrukturne karakteristike i razlike sustava koje koriste, zanima ih ono što vide odnosno "osjete" prilikom korištenja, stoga su metrike za kvalitetu web sustava jednake kao i za sve druge računalne sustave (Souders, 2007).

Ukoliko računalni sustav na korisnički zahtjev odgovori unutar 100 milisekundi (0.1 sekunda), kod korisnika je stvoren dojam "instantnosti" – rezultat je prikazan u vremenu u kojem korisnik nije stigao osjetiti da se nešto događa odnosno nije primjećen zastoj rada sustava. Ukoliko korisnik čeka na odgovor sustava više od 1 sekunde, on gubi osjećaj izravnog rada nad podacima i osjeća usporenje u radu, ali i dalje zadržava fokus na zadacima koje obavlja. Sustavi kojima je potrebno 10 sekundi ili više za generiranje odgovora na korisničku interakciju ne mogu računati s koncentracijom korisnika – u tom vremenu korisnici se obično okreću drugim zadacima dok čekaju da sustav završi obradu. Takvi sustavi obavezno moraju korisnicima knstantno pružati povratnu informaciju o napretku (eng. *progress*) obrade (npr. instalacija zahtjevnih računalnih programa i traka napretka koja prikazuje postotke dovršenosti) (Miller, 1968).

Autori web sustava moraju u obzir uzeti nekoliko performansno nepovoljnih okolnosti: vrijeme prijenosa podataka korisnik percipira kao vrijeme obrade, korisnici često pristupaju web sustavima s mobilnih ili drugih uređaja s ograničenom brzinom pristupa Internetu, svi resursi (multimedijske datoteke i druge datoteke, artefakti) putuju prema korisniku kroz mrežu i vrijeme njihovog prijenosa ne smije se zanemariti, više korisnika može istovremeno pristupati web sustavu i na taj ga način opteretiti. Imajući u vidu navedena ograničenja pažljivi čitatelj može zaključiti da web sustavi često moraju raditi mnogo brže kako bi kompenzirali nedostatke i ograničenu brzinu prijenosa podataka.

Čak i kada su svi podaci uspješno preneseni na korisnikovo računalo, web sustav nije završio s radom – reprezentacija pristiglih podataka, dohvaćanje povezanih resursa kao i obrada na strani klijenta samo su neke od operacija koje web preglednici obavljaju nakon što su podaci uspješno preneseni. Gotovo 80-90% (Souders, 2007) posla otpada na obradu i prezentaciju podataka na strani klijenta (tzv. frontend) što ostavlja vrlo malo prostora za optimizacijski manevar na strani poslužitelja (tzv. backend).

Iako se većina optimizacija web sustava svodi na minimiziranje podataka koji se prenose kroz mrežu (pohranom u privremenu memoriju, minifikacijom i kompresijom datoteka ili drugim tehnikama), optimizacija na strani poslužitelja važna je zbog smanjenja troškova i ulaganja u infrastrukturu te potrošnje energije (Souders, 2007).

Teško je postaviti ljestvicu koja bi jednoznačno kategorizirala promet web sustava kao visok ili nizak, ali moguće je povući paralelu s postojećim web sustavima i kategorizaciju prilagoditi trenutnom stanju. Tablica u nastavku prikazuje kategorije web sustava prema njihovom prometu (izraženom u broju jedinstvenih korisnika mjesečno, eng. *unique monthly visitors*)[[8]](#footnote-8):

|  |  |  |
| --- | --- | --- |
| **Kategorija** | **Promet (jedinstveni korisnici mjesečno)** | **Opis** |
| Web sustav sa sitnim prometom | 0 – 100.000 | Lokalno poznati portali i blogovi, manje web stranice |
| Web sustav s niskim prometom | 100.000 – 1.000.000 | Popularni regionalni *news* portali, regionalno popularni blogovi |
| Web sustavi s osrednjim prometom | 1.000.000 – 10.000.000 | Svjetski poznati blogovi, web trgovine i viralni sadržaji |
| Web sustavi sa srednje visokim prometom | 10.000.000 – 100.000.000 | Svjetski poznati news portali, atraktivni forumi i grupe za diskusiju (npr. StackExchange) |
| Web sustavi sa značajnim prometom | 100.000.000 – 500.000.000 | npr. Wikipedia, Amazon |
| Web sustavi s visokim prometom | 500.000.000 – 1.000.000.000 | npr. Facebook, YouTube, Yahoo |
| Web sustavi s iznimno visokim prometom | Više od 1.000.000.000 | npr. Google |

Tablica : Kategorizacija web sustava prema mjesečnom broju jedinstvenih posjetitelja

* 1. Performansne karakteristike web sustava

Performansne su karakteristike web sustava one osobine koje opisuju brzinu kojom web sustav može obraditi određenu količinu zahtjeva, koju količinu prometa web sustav može podnijeti prije nego performanse značajno opadnu ili sustav postane nedostupan te koliko je jednostavno (ili kompleksno) skalirati i pripremiti sustav za nekoliko redova veličine prometa.

@todo objasniti pojedinu metriku s primjerima

* + 1. Propusnost web sustava
    2. Vrijeme odziva web sustava
    3. Dostupnost web sustava
  1. Testiranje performansi web sustava

@todo objasniti što je testiranje web sustava, što se mjeri i kako, te kako se intepretiraju rezultati na primjerima alata siege i ab. Objasniti pojedinu vrstu testiranja

* + 1. Testiranje web sustava pod opterećenjem
    2. Testiranje granica web sustava
    3. Regresijsko testiranje web sustava

1. Razvoj web sustava visokih performansi

Web sustavi visokih performansi svoje zadatke najčešće obavljaju u uvjetima visokog prometa, a sastoje se od većeg broja različitih poslužitelja (web poslužitelj, poslužitelj baze podataka, poslužitelj za raspored prometa i sl.). Takvi web sustavi najčešće obradu na strani poslužitelja moraju dovršiti unutar 100 milisekundi kako bi se stvorili uvjeti za postizanje efekta instantnosti kod krajnjeg korisnika. U nastavku rada napravljen je pregled procesa razvoja web sustava visokoih performansi, metrika za mjerenje performansi web sustava te mehanizmi za optimizaciju web sustava.

* 1. Koraci razvoja i optimizacije web sustava

Jedan od najvećih znanstvenika u području računalnih znanosti, Donald E. Knuth, izjavio je kako je "*preuranjena optimizacija izvor svog zla (ili barem većine) u programiranju*"[[9]](#footnote-9) (Knuth, 1974), a razvoj web sustava nije iznimka. U početku razvoja web sustava nije moguće u potpunosti predvidjeti sve scenarije niti količinu prometa koja će zapravo pristizati - poslovna očekivanja i planirani promet (a i prihod) često su pod subjektivnim utjecajem, stoga optimizaciju web sustava treba temeljiti na realnim brojkama i podacima analitičkih alata.

Nakon što je web sustav razvijen i u potpunosti funkcionalan potrebno je napraviti analizu očekivanog prometa te identificirati sučelja koja će najčešće biti "na udaru" velikog broja korisnika odnosno HTTP zahtjeva. Takvu analizu najlakše je napraviti prilikom redizajna postojećeg web sustava, kada su dostupni podaci o posjećenosti prethodne verzije, no kada se razvija potpuno novi web sustav (tzv. "od nule"*)* potrebno je analizirati slične web sustave i njihov promet te se osloniti na iskustvo arhitekta sustava i drugih stručnjaka iz područja. Nakon što su identificirana sučelja koja će biti pod značajnim opterećenjem potrebno je utvrditi mogućnost pohrane podataka u mehanizme za privremenu pohranu (eng. *cache*). Takvi mehanizmi omogućavaju pohranu potpunog skupa podataka koji će kasnije biti dostupan bez obrade na strani poslužitelja, a osvježavanje takvih podataka odnosno poništavanje može biti ručno ili temeljeno na vremenu (eng. *time base cache invalidation*).

Optimizacijski se mehanizmi uvode u web sustav tek nakon što je potvrđeno usko grlo koje će takav mehanizam poboljšati, a kako bi se takvo poboljšanje potvrdilo potrebno je provesti identične testove na obje verzije sustava (prije i nakon optimizacije) u gotovo identičnim uvjetima testiranja.

* 1. Mjerenje i optimizacija performansi

William Thomson, poznat kao Lord Kelvin, škotsko-irski je znanstvenik u svojim je radovima 1883. navodio godine naveo kako "ne možemo poboljšati ono što ne možemo mjeriti"[[10]](#footnote-10) te "mjerenje je znanje"[[11]](#footnote-11). Peter Drucker, poslovni konzultant i autor, te izjave primijenio je tijekom svoje konzultantske karijere kako bi istaknuo važnost mjerenja performansi prilikom optimizacije performansi poslovnih sustava. Isto pravilo primijenjivo je i na računalne sustave, gdje je najvažnije postaviti relevantne metrike, provesti mjerenja prije i nakon optimizacije te donijeti zaključke. Web je sustave moguće optimizirati na nekoliko načina, a nastavak poglavlja donosi pregled mehanizama i koncepata za optimizaciju performansi web sustava na aplikacijskom, podatkovnom i infrastrukturnom sloju.

* + 1. Optimizacija aplikacijskog sloja web sustava

Aplikacijski sloj obuhvaća programski kod konkretnog web sustava (uključujući i vanjske biblioteke) odnosno poslovna pravila kojima sustav podliježe. Postupak analize performansi programskog koda naziva se profiliranje (eng. *profiling*), a rezultat profiliranja aplikacije daje uvid u trajanje i kompleksnost izvođenja pojedinog odsječka programskog koda. Analizom rezultata profiliranja moguće je identificirati uska grla odnosno dijelove programskog koda koji relativno gledajući zahtijevaju najviše vremena za izvršavanje. Takve problematične cjeline potrebno je optimizirati ili minimizirati njihovo korištenje pohranom rezultat u privremenu memoriju.

@todo screenshot profilinga na primjeru PHP Symfony aplikacije

Optimizacija programskog koda u PHP programskom jeziku najčešće se svodi na korištenje nativnih funkcija programskog jezika umjesto ručne implementacije, kada takve funkcije postoje, korištenje manje fleksibilnih operatora i sintaksnih elemenata kao što su operator === umjesto == i jednostrukih navodnika: 'vrijednost' umjesto dvostrukih: "vrijednost". Takvi, manje fleksibilni elementi ne rade automatsku konverziju tipova podataka niti provjeravaju postojanje varijabli u tekstualnim podacima te brže dolaze do rezultata. Rezultate izračuna uvijek je bolje pohraniti u varijablu (ili u privremenu memoriju) umjesto ponavljanja izračuna, pogotovo ako se radi o petljama. Iteriranje kroz niz podataka s lijeve strane neopitmalno je jer se u svakom koraku petlje iznova računa veličina samog niza. S desne strane prikazuje se optimalna varijanta iteracije:

|  |  |  |
| --- | --- | --- |
| **Neoptimalno:** |  | **Optimalno:** |
| for ($i=0; i < count($array); $i++){  someFunction($array);  } |  | $length = count($array);  for ($i=0; i < length; $i++){  someFunction($array);  } |

Primjer : Neopitmalno i optimalno iteriranje kroz niz podataka u PHP-u

Broj upita prema bazi podataka treba svesti na minimum, a često korištene podatke pohraniti u privremenu memoriju. S podacima koji se pohranjuju i koriste iz privremene memorije treba raditi oprezno, jer je nepravilno poništavanje (invalidacija) takvih podataka često uzrok problema odnosno programskih bugova (Isaac, 2014)*,* a Phil Karlton prema (Fowler, 2009) navodi kako je poništavanje podataka pohranjenih u privremenu memoriju (eng. *cache invalidation*) jedan od dva najteža zadatka u računalnim znanostima (kao drugu navodi pravilno imenovanje).

* + 1. Optimizacija podatkovnog sloja web sustava

Optimizacija podatkovnog sloja web sustava podrazumijeva optimizaciju upita prema bazi podataka, prilagodbu strukture baze podataka i ciljano narušavanje normalizirane strukture baze podataka s ciljem optimizacije performansi te intenzivno korištenje privremene memorije (eng. *cache*). U nastavku je napravljen pregled načina optimizacije web sustava s naglaskom na podatkovni sloj aplikacije.

Izvršavanje upita prema bazi podataka performansno je "skupa" operacija koja, ovisno o složenosti upita i količini spremljenih podataka u bazi, može potrajati i do nekoliko sekundi (znatno složeniji upiti nad velikim skupom podataka mogu potrajati i do nekoliko minuta). U uvjetima velike količine prometa (tj. velikog broja zahtjeva u jedinici vremena) baza podataka najčešće postaje usko grlo, a budući da sustavi za upravljanje bazama podataka nisu optimizirani za velik broj paralelnih konekcija najčešće dolazi do potpune nedostupnosti baze podataka (npr. MySQL greška "*ERROR 2006 (HY000) at line 1: MySQL server has gone away*"). Često je za obradu jednog HTTP zahtjeva potrebno izvršiti nekoliko upita prema bazi podataka, pa čak i desetak paralelnih korisnika može znatno usporiti rad web sustava (posebice kada je riječ o velikoj količini podataka koja se dohvaća). Zlatno je pravilo da se upiti prema bazi podataka izvršavaju što rjeđe, odnosno da se podaci dohvaćeni iz baze podataka (rezultati upita) čuvaju u privremenoj memoriji za sljedeće upite. Uzmemo li za primjer naslovnicu *news* portala, možemo identificirati sadržaj koji će često biti potraživan iz baze podataka odnosno potreban za pravilan rad sustava – to su najčešće prikazi sažetaka članaka prema različitim kriterijima (npr. posljednje dodani članci, najčitaniji članci, najkomentiraniji članci i sl.). Ako detaljnije promotrimo odnos operacija čitanja i pisanja takvih podataka zaključujemo kako se podaci češće čitaju iz baze podataka (odnosno prikazuju) nego što se upisuju u bazu podataka (odnosno ažuriraju). Sadržaj naslovnice *news* portala mijenja se u nepravilnim intervalima u razmaku od nekoliko minuta do nekoliko sati (npr. noću), a sadržaj članaka nakon objave mijenja se vrlo rijetko. S druge strane svaki posjetitelj naslovnice portala zahtjeva prikaz tih podataka, a u uvjetima povećane posjećenosti naslovnica se prikazuje i do nekoliko desetaka puta u minuti. Postavlja se pitanje je li potrebno svaki put izvršavati upit prema bazi podataka ili je moguće ponovno iskoristiti podatke koji su prethodno već bili dohvaćeni? Pohranom rezultata upita moguće je povećati propusnost web sustva za nekoliko redova veličine (detaljnije je prikazano u praktičnom dijelu rada). Sadržaj se pohranjuje u privremenu memoriju te se poništava kada dođe do promjene sadržaja (npr. urednik zamijeni sadržaj naslovnice).

Analizom čestih upita prema bazi podataka moguće je odrediti atribute u relacijskoj shemi koji su pogodni za uvođenje indeksa. Indeksi u bazama podataka predstavljaju mehanizam optimizacije pronalaska traženih podataka prema unaprijed određenim ključevima (atributima), baš kao što indeksi u knjigama omogućuju pronalazak definicije pojma pregledom sortiranog popisa. Uvođenjem indeksa u strukturu baze podataka usporavaju se operacije ažuriranja podataka stoga indekse treba uvoditi ciljano. Indeksi se najčešće uvode za atribute primarnog ključa, atribute koji se često koriste za sortiranje sadržaja (npr. vrijeme objave članka, broj pregleda) te jedinstvene (eng. *unique*) vrijednosti u shemi.

* + - 1. Prilagodba strukture baze podataka s ciljem optimizacije sustava
      2. Pohrana rezultata u privremenu memoriju
    1. Optimizacija infrastrukturnog sloja web sustava
       1. HTTP zaglavlja i pohrana podataka u privremenu memoriju
       2. Kompresija podataka i pohrana podataka u privremenu memoriju preglednika

1. Praktičan rad – web sustav visokih performansi

U nastavku ovog poglavlja praktično je obrađen pristup mjerenju, optimizaciji i poboljšanju performansi web sustava tj. fiktivnog *news* portala (u nastavku portal). Regionalni *news* portali ostvaruju srednje visoki promet prema tablici 3, a često su podložni trendovima (npr. prilikom prirodnih katastrofa, terorističkih napada, značajnih događaja i sl.) pa je robusnost sustava i mogućnost skaliranja od iznimne važnosti.

* 1. Opis web sustava

Obično se *news* portal sastoji od nekoliko standardiziranih sučelja ili "tipskih stranica" koje se međusobno razlikuju prema tipu sadržaja i količini prometa koju svaka od stranica zaprima:

* **Naslovnica portala** – obično najposjećenije sučelje *news* portala, na jednom mjestu prikazuje popularne novosti, posljednje dodane članke. Ovo je često i najčešće ažurirana stranica portala s obzirom da je objava članaka na naslovnici strateški i poslovni alat *news* portala.
* **Sučelje/naslovnica pojedine kategorije** – sučelje vizualno često slično naslovnici portala koje prikazuje popularne članke iz jedne kategorije. Sučelje se ažurira svaki puta kada dođe do promjene redoslijeda objavljenih članaka ili promjene unutar članka te kategorije.
* **Sučelje pojedinog članka** – sučelje koje sadrži cjelokupan sadržaj pojedinog članka, u pravilu se ne ažurira često, osim u slučajevima kada se radi o posebnom tipu članka, tzv. *live* članak.
* **Statičke stranice** (npr. impressum ili cjenik oglašavanja) – sučelja koja se ažuriraju vrlo rijetko a prikazuju statičke informacije.

S obzirom da je naslovnica portala najčešće posjećeno sučelje te da se sadržaj naslovnice iz baze podataka dohvaća pomoću nekolicine upita prema bazi podataka i prema različitim kriterijima (npr. jedan modul može prikazivati članke sortirane prema broju posjeta, drugi prema broju komentara i sl.) naslovnica je i najkompleksnije sučelje *news* portala. U nastavku praktičnog dijela rada fokus će biti stavljen na mjerenje i optimizaciju performansi naslovnice portala. Korišteni mehanizmi optimizacije primijenjivi su i na sva ostala sučelja portala.

* + 1. Infrastruktura

Portal je izrađen na standardnom LAMP/LEMP skupu tehnologija (LAMP i LEMP su akronimi od naziva korištenih tehnologija Linux, Apache/NGINX, MySQL/MariaDB i PHP). Sustav se sastoji od jednog web Nginx poslužitelja (eng. *web server*), jednog aplikacijskog poslužitelja (eng. *application server, app server*) koji pokreće PHP aplikaciju, jednog MySQL poslužitelja baze podataka (eng. *database server, db server*) i Redis poslužitelja za privremenu pohranu podataka (eng. *cache server*).

@todo dijagram arhitekture

* + 1. Model baze podataka

@todo ERA dijagram

* + 1. Korišteni alati

Web sustav implementiran je u PHP programskom jeziku verzije 7.0, korištena je MySQL baza podataka verzije 5.7.19, Nginx web poslužitelj verzije 1.13.3, Redis sustav za privremenu pohranu podataka verzije 3.2.10, Docker sustav za kontejnerizaciju servisa i Docker Compose alat za orkestriranje (upravljanje i održavanje) Docker kontejnera. Korišteni su siege i ab alati za testiranje performansi web sustava.

@todo dodati linkove za korištene alate

* 1. Optimizacija i mjerenje performansi web sustava

U nastavku su prikazani rezultati stres testova i testova opterećenja web sustava prije i nakon implementacije mehanizama optimizacije. Rezultati su interpretirani i sumarno vizualizirani u obliku grafova koji jasno prikazuju kako je koji mehanizam optimizacije utjecao na performanse web sustava.

* + 1. Performanse sustava bez optimizacijskih mehanizama
    2. Optimizacija dodavanjem indeksa baze podataka
    3. Optimizacija korištenjem privremen pohrane podataka na strani aplikacije
    4. Optimizacija korištenjem HTTP poslužitelja za privremenu pohranu podataka
    5. Sumarni prikaz rezultata i interpretacija

1. Zaključak
2. Literatura

Belshe, M., Thomson, M. and Peon, R. (2014) ‘Hypertext Transfer Protocol Version 2 (HTTP/2)’. Available at: https://tools.ietf.org/html/rfc7540 (Accessed: 20 August 2017).

Berners-Lee, T. (1989) *Information Management: A Proposal*. Available at: https://www.w3.org/History/1989/proposal.html (Accessed: 28 August 2017).

Berners-Lee, T. *et al.* (1997) ‘RFC2068 - Hypertext Transfer Protocol -- HTTP/1.1’. Available at: https://tools.ietf.org/html/rfc2068 (Accessed: 18 August 2017).

Berners-Lee, T. *et al.* (2004) ‘Architecture of the World Wide Web, Volume One’, *W3C Recommendation 15 December 2004*. Available at: https://www.w3.org/TR/webarch/ (Accessed: 20 August 2017).

Berners-Lee, T. (no date a) *Answers for young people - Tim Berners-Lee*. Available at: https://www.w3.org/People/Berners-Lee/Kids.html (Accessed: 28 August 2017).

Berners-Lee, T. (no date b) *Frequently asked questions by the Press - Tim BL*. Available at: https://www.w3.org/People/Berners-Lee/FAQ.html (Accessed: 28 August 2017).

CareerProfiles (2017) *Explore the Top 100 Careers for 2017*. Available at: http://www.careerprofiles.info/top-100-careers.html (Accessed: 27 August 2017).

Christensen, E. *et al.* (2001) *Web Service Definition Language (WSDL)*. Available at: https://www.w3.org/TR/wsdl (Accessed: 20 August 2017).

coderseye.com (2017) *11 Best PHP Frameworks for Modern Web Developers in 2017*. Available at: https://coderseye.com/best-php-frameworks-for-web-developers/ (Accessed: 28 August 2017).

Collins, W. (2012) *Collins English Dictionary - Complete & Unabridged 2012 Digital Edition*. HarperCollins Publishers.

Composer (2017) *Introduction - Composer*. Available at: https://getcomposer.org/doc/00-intro.md (Accessed: 28 August 2017).

Eevee (2012) *PHP: a fractal of bad design / fuzzy notepad*. Available at: https://eev.ee/blog/2012/04/09/php-a-fractal-of-bad-design/ (Accessed: 28 August 2017).

Flanagan, D. (2011) *JavaScript: The Definitive Guide, 6th Edition - O’Reilly Media*. 6th edn. O’Reilly Media. Available at: http://shop.oreilly.com/product/9780596805531.do (Accessed: 20 August 2017).

Fowler, M. (2009) *TwoHardThings*. Available at: https://martinfowler.com/bliki/TwoHardThings.html (Accessed: 28 August 2017).

*Github Language Stats* (2017). Available at: https://madnight.github.io/githut/ (Accessed: 27 August 2017).

Internet World Stats (2017) *World Internet Users Statistics and 2017 World Population Stats*. Available at: http://www.internetworldstats.com/stats.htm (Accessed: 18 August 2017).

Investopedia (2017) *Dotcom Bubble*. Available at: http://www.investopedia.com/terms/d/dotcom-bubble.asp.

Isaac, L. P. (2014) *15 Tips to Optimize Your PHP Script for Better Performance for Developers*. Available at: http://www.thegeekstuff.com/2014/04/optimize-php-code/ (Accessed: 24 August 2017).

IT Conversations (2003) ‘Rasmus Lerdorf, Senior Technical Yahoo: PHP, Behind the Mic’. Available at: https://web.archive.org/web/20130728125152/http://itc.conversationsnetwork.org/shows/detail58.html (Accessed: 19 August 2017).

Knuth, D. E. (1974) ‘Computer programming as an art’, *Communications of the ACM*. ACM, 17(12), pp. 667–673. doi: 10.1145/361604.361612.

Letts, S. (2015) *What is Web 4.0?* Available at: https://stephenletts.wordpress.com/web-4-0/ (Accessed: 22 August 2017).

MacKay, J. (2016) *Building Facebook today: Is PHP still relevant in 2017? - Crew.co*. Available at: https://crew.co/blog/is-php-still-relevant-in-2017/ (Accessed: 28 August 2017).

Miller, R. B. (1968) ‘Response time in man-computer conversational transactions’, in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS ’68 (Fall, part I)*. New York, New York, USA: ACM Press, p. 267. doi: 10.1145/1476589.1476628.

Nottingham, M. *et al.* (2015) *Hypertext Transfer Protocol (httpbis)*, *IETF Datatracker*.

O’Reilly, T. (2005) *What Is Web 2.0*. Available at: http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html (Accessed: 22 August 2017).

Patel, K. (2013) ‘Incremental Journey for World Wide Web: Introduced with Web 1.0 to Recent Web 5.0 – A Survey Paper’, *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(10), pp. 2277–128. Available at: http://www.ijarcsse.com/docs/papers/Volume\_3/10\_October2013/V3I10-0149.pdf (Accessed: 22 August 2017).

Poushter, J. (2016) ‘Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies’, *Pew Research Center*, pp. 1–45. doi: 10.1017/CBO9781107415324.004.

Richardson, C. (2017) *Microservice Architecture pattern*. Available at: http://microservices.io/patterns/microservices.html (Accessed: 28 August 2017).

Shellhammer, A. (2017) *The need for mobile speed: How mobile latency impacts publisher revenue*. Available at: https://www.doubleclickbygoogle.com/articles/mobile-speed-matters/ (Accessed: 18 August 2017).

SimilarTech (2017) *PHP Market Share and Web Usage Statistics*. Available at: https://www.similartech.com/technologies/php (Accessed: 28 August 2017).

Souders, S. (2007) *14 Rules for Faster-Loading Web Sites*. Available at: http://stevesouders.com/hpws/ (Accessed: 20 August 2017).

Stackify (2017) *Trendiest and Most Popular Programming Languages in 2017*. Available at: https://stackify.com/trendiest-programming-languages-hottest-sought-programming-languages-2017/ (Accessed: 27 August 2017).

Stogov, D. and Suraski, Z. (2014) *PHP RFC: Move the phpng branch into master*. Available at: https://wiki.php.net/rfc/phpng (Accessed: 20 August 2017).

Sturgeon, P. (2014) *The Neverending Muppet Debate of PHP 6 v PHP 7 | Phil Sturgeon*. Available at: https://philsturgeon.uk/php/2014/07/23/neverending-muppet-debate-of-php-6-v-php-7/ (Accessed: 20 August 2017).

The PHP Group (2017a) *History of PHP and Related Projects*. Available at: http://php.net/manual/en/history.php.php (Accessed: 19 August 2017).

The PHP Group (2017b) *PHP: Supported Versions*. Available at: http://php.net/supported-versions.php (Accessed: 20 August 2017).

The PHP Group (2017c) *PHP: Unsupported Branches*. Available at: http://php.net/eol.php (Accessed: 20 August 2017).

TIOBE (2017) *TIOBE Index | TIOBE - The Software Quality Company*. Available at: https://www.tiobe.com/tiobe-index// (Accessed: 28 August 2017).

Trachtenberg, A. (2004) *Upgrading to PHP 5*. O’Reilly Media. Available at: http://shop.oreilly.com/product/9780596006365.do (Accessed: 20 August 2017).

VPN Mentor (2017) *Internet Trends 2017. Stats & Facts in the U.S. and Worldwide*. Available at: https://www.vpnmentor.com/blog/vital-internet-trends/ (Accessed: 18 August 2017).

W3Techs (2017a) *Historical trends in the usage of PHP version 5, August 2017*. Available at: https://w3techs.com/technologies/history\_details/pl-php/5 (Accessed: 20 August 2017).

W3Techs (2017b) *Usage Statistics and Market Share of PHP for Websites, August 2017*. Available at: https://w3techs.com/technologies/details/pl-php/all/all (Accessed: 28 August 2017).

W3Techs (2017c) *Usage Statistics and Market Share of Server-side Programming Languages for Websites, August 2017*. Available at: https://w3techs.com/technologies/overview/programming\_language/all (Accessed: 28 August 2017).

W3Techs (2017d) *Usage Statistics of HTTP/2 for Websites, August 2017*. Available at: https://w3techs.com/technologies/details/ce-http2/all/all (Accessed: 20 August 2017).

webfoundation.org (no date) *History of the Web – World Wide Web Foundation*, *Web Foundation*. Available at: https://webfoundation.org/about/vision/history-of-the-web/ (Accessed: 28 August 2017).

World Wide Web Consortium (W3C) (2011) *W3C Semantic Web Activity*.

Zmievski, A. (2005) *[PHP-DEV] PHP Unicode support design document*. Available at: https://marc.info/?l=php-internals&m=112365908921757&w=1 (Accessed: 20 August 2017).

Zmievski, A. (2011) *The Good, the Bad, and the Ugly: What Happened to Unicode and PHP 6*. Available at: https://www.slideshare.net/andreizm/the-good-the-bad-and-the-ugly-what-happened-to-unicode-and-php-6 (Accessed: 20 August 2017).

1. Izvorni kod praktičnog primjera

1. <https://home.cern/topics/birth-web> Službeni podaci o rođenju weba [↑](#footnote-ref-1)
2. <http://info.cern.ch/hypertext/WWW/TheProject.html> Prva web stranica koju je Tim Berners-Lee objavio 1990. godine [↑](#footnote-ref-2)
3. <https://www.ietf.org/> - Internet Engineering Task Force organizacija [↑](#footnote-ref-3)
4. <https://www.w3.org/> - World Wide Web Consortium (W3C) konzorcij [↑](#footnote-ref-4)
5. <https://www.ietf.org/iesg/> - službena web stranica IESG grupe zadužene za tehničko upravljanje aktivnosti IETF te donošenje Internet standarda. [↑](#footnote-ref-5)
6. <http://schema.org> - Schema.org zajednica stvorena s ciljem promocije i definiranja strukturiranih podataka na Internetu. [↑](#footnote-ref-6)
7. https://packagist.org/ - Packagist, the PHP package repository [↑](#footnote-ref-7)
8. <https://webmasters.stackexchange.com/questions/17914/what-is-a-lot-of-traffic> diskusija glede kategorizacije web sustava prema prometu [↑](#footnote-ref-8)
9. "*premature optimization is the root of all evil (or at least most of it) in programming*" [↑](#footnote-ref-9)
10. "If you can not measure it, you can not improve it." [↑](#footnote-ref-10)
11. "To measure is to know." [↑](#footnote-ref-11)