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, rujan 2017.

Zahvala

*Zahvaljujem se **Davoru Vrandečiću** što mi je 2007. godine otvorio vrata u svijet razvoja web stranica, **Tomislavu Jakopecu** koji me uveo u programski jezik PHP i savjetovao pri odabiru fakulteta, te **Ivanu Horvatu** i **Tomislavu Ramljaku** na suradnji i podršci tijekom studija.*

*Zahvaljujem se poduzeću **Trikoder d.o.o.** na ukazanom povjerenju i prilici za suradnju, te na osobnom i profesionalnom razvoju tijekom iste. Posebno se zahvaljujem **Davoru Plehatom** na mentoriranju i strpljenju tijekom mojih prvih koraka u profesionalnom razvoju web sustava visokih performansi te **Alenu Pokosu** i **Vedranu Križeku** na mentoriranju, suradnji i dijeljenju znanja.*

*Zahvaljujem se mentoru **prof. dr. sc. Dragutinu Kermeku** na formalnoj edukaciji u području razvoja web sustava te konstruktivnim savjetima i usmjeravanju tijekom pisanja ovog rada.*

*Najviše se zahvaljujem majci **Rozani**, sestri **Ivani** i bratu **Marku**. Bez vaše podrške i razumijevanja ništa ne bi bilo isto.*

Z.A.

Sadržaj

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 sebi i kolegama znanstvenicima omogućio jednostavniju razmjenu informacija, kako unutar institucije tako i između znanstveno-istraživačkih institucija[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 autora te upute za postavljanje osobnog web poslužitelja[2]. Izvorni kod World Wide Weba objavljen je 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 i 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 sigurniju i transparentniju komunikaciju te distribuciju informacija bez cenzure.

Performanse web sustava značajno utječu na korisničko iskustvo i njihovo ponašanje na webu, pogotovo kod korisnika koji webu pristupaju putem mobilnih uređaja. Primjerice, ako 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*)

[1] https://home.cern/topics/birth-web Službeni podaci o rođenju weba
[2] http://info.cern.ch/hypertext/WWW/TheProject.html Prva web stranica koju je Tim Berners-Lee objavio 1990. godine

izravno utječe na efikasnost prikazivanja oglasa (tj. zarade od internet oglasa), korisničko zadržavanje na web mjestu i sklonost napuštanja web mjesta s prve posjećene stranice (eng. *bounce rate*) (Shellhammer, 2017).

Ovaj rad donosi osnovne informacije o povijesti i razvoju weba, web sustavima, njihovu kategorizaciju, povijest i pregled razvoja PHP programskog jezika te optimizacijske mehanizme primjenjive 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.

2. World Wide Web

Internet je često pogrešno poistovjećen s webom 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.

2.1. 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. 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 i koristiti (i naučiti kako koristiti) različite sustave kako bi pristupili potrebnim informacijama. 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 elektronskom poštom, a dijeljenje informacija bilo je svedeno na izravnu razmjenu 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*). 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 te je aktivno koristio Enquire za pohranu i povezivanje 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, *s.a.*).

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 tada 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, *s.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 ujedno i prvi web poslužitelj.

2.2. Hypertext Transfer Protocol (HTTP)

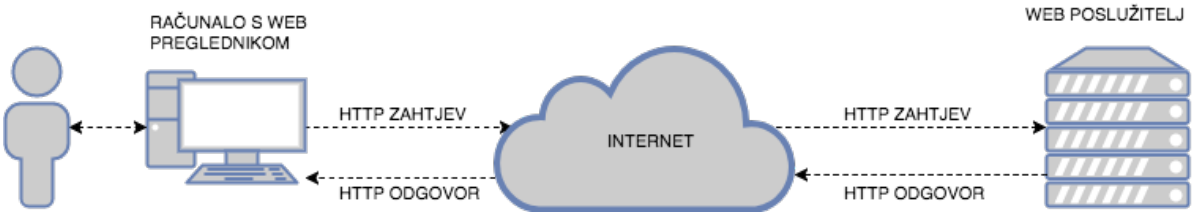
Hypertext Transfer Protocol (HTTP) protokol je aplikacijskog sloja ISO-OSI konceptualnog modela komunikacijskih sustava kojim se najčešće opisuju Internet sustavi. HTTP je definiran kao generički, višestruko primjenjiv, objektno orijentirani protokol bez stanja (eng. stateless). Protokol definira pravila 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] i World Wide Web Consortium (W3C)[4]. Tim Berners-Lee danas je direktor W3C organizacije. Najraširenija

[3] https://www.ietf.org/ - Internet Engineering Task Force organizacija
[4] https://www.w3.org/ - World Wide Web Consortium (W3C) konzorcij

verzija HTTP protokola, verzija HTTP/1.1, prvi je put opisana u dokumentu RFC 2068 1997. godine, a kasnije je zamijenjen dokumentom RFC 2616 1999. godine. Grupa dokumenata RFC 7230 (RFC 7230, RFC 7231, RFC 7232, RFC 7233, RFC 7234, RFC 7235) zamjenjuju i nadopunjuju HTTP protokol 2014. godine.

HTTP protokol temelji se na parovima poruka zahtjeva i odgovora. Klijent (najčešće web preglednik) generira zahtjev za traženi resurs (npr. web stranicu, multimedijski resurs i sl.) te u njega uključuje 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 1**: Princip rada HTTP protokola: zahtjev - odgovor*

HTTP zahtjevi definiraju tzv. HTTP metode (npr. GET, POST, HEAD, PUT, DELETE, ...) koje ga semantički pobliže određuju, 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] 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 inženjera tvrtke Google 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 zahtjeva. 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 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)

[5] https://www.ietf.org/iesg/ - službena web stranica IESG grupe zadužene za tehničko upravljanje aktivnosti IETF te donošenje Internet standarda.

- 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.

2.3. Web sustavi

Web sustavima nazivaju se računalni sustavi (aplikacije, platforme, stranice i servise) koji se temelje na HTTP protokolu, odnosno oni računalni sustavi koje resurse distribuiraju kroz web. Ovisno o razinama dinamike sadržaja i mogućnosti interakcije 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, Sustav za *online* plaćanje

***Tablica 1**: Kategorizacija web sustava*

2.3.1. Web stranice

Web stranice sadrže statički sadržaj koji se ne mijenja dinamički već je svaka promjena ručno inicirana (npr. autor web stranice mora ručno promijeniti sadržaj). Web stranice najraniji su 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.

2.3.2. Web servisi

Web servisima nazivaju se web sustavi čiji su primarni korisnici računalni sustavi. Takvi web sustavi sadržaj ne prezentiraju u vizualnom obliku prilagođenom za čovjeka, već u strukturiranim i standardiziranim formatima kao što su XML i JSON koje drugi računalni sustavi "razumiju" odnosno mogu obraditi. Web servisi najčešće predstavljaju javno dostupni dio nekog zatvorenog sustava (npr. web servisi za rezervaciju karata avioprijevoznika 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 i prakse poput "Simple Object Access Protocol" (SOAP) i "Representational State Transfer" (REST). 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).

2.3.3. 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 dijela koji je dostupan svima te privatnog dijela sustava koji je rezerviran za korisnike s aktivnim korisničkim računom. Osim autentikacijskog i autorizacijskog sloja, web aplikacije obično implementiraju različite oblike obavijesti (eng. *notifications*), dvosmjerne kanale komunikacije (razmjena poruka i komentara), mogućnost promjene korisničkih postavki i slične funkcionalnosti. Web aplikacije često sadrže nekoliko različitih web servisa koji u pozadini prikupljaju i obrađuju podatke te ih pripremaju za kasniju prezentaciju i korištenje. Uzmemo li za primjer web sustav PayPal, najpoznatiji sustav za digitalni prijenos novca, moguće je identificirati web aplikaciju (sučelja za prijavu, upravljanje računima, plaćanja i sl.) te web servise koji na javno dostupnim adresama (eng. *API endpoint*) omogućuju programsko korištenje usluga (npr. integraciju plaćanja PayPal-om u druge web aplikacije).

2.4. 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 *dotcoms* (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 perspektivne kompanije (tzv. *startups*) koje su svoje poslovanje temeljile na webu. U periodu između 2000. i 2002. godine dogodilo se tzv. pucanje *dotcom* 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 *dotcom* 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 niti kriterije prema kojima se web sustav može karakterizirati kao Web 2.0 sustav, već se radi o skupu karakteristika web sustava. Web 2.0 sustavi web tretiraju kao platformu za pružanje 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 usluge web preglednika 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, zastarijevanje 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 ostali primjeri Web 2.0 sustava (npr. Wikipedia, Bay i Amazon). Sljedeća osnovna karakteristika Web 2.0 sustava jest omogućavanje kreiranje sadržaja krajnjim korisnicima, tako primjerice Wikipedia omogućava kolaborativno kreiranje unosa, Google povezuje sadržaj kojima nije vlasnik, a na eBayu korisnici prodaju artikle samostalno. Amazon je pionir korištenja korisničkih osvrta (eng. *user review*) te se tako pozicionirao kao pouzdan izvor kvalitetnih proizvoda. Svi uspješni Web 2.0 sustavi iskoristili su doprinos korisnika kako bi postigli tržišnu dominaciju. Web 2.0 sustavi više su pažnje posvetili korisničkom iskustvu (eng. *user experience, UX*) za razliku od "sirovih" sučelja računalnih sustava "stare škole". Takvi su sustavi postali dostupni korisnicima s nižom razinom informatičke pismenosti, a računalna i informacijska industrija ozbiljno je počela shvaćati osjećaje koje korištenje sustava izaziva kod korisnika. S tehnološke strane, AJAX (akronim od eng. *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 autonomni računalni sustavi mogli prikupljati i obrađivati dostupne podatke i informacije na webu (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] 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. Daljnji razvoj semantičkog weba omogućit će veću razinu autonomije računalnih sustava.

Web 4.0 označava 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 moći će "razumjeti" okolinu i uvjete u kojima se korisnik nalazi bez njegove interakcije (Letts, 2015). Virtualni asistenti kao što su Siri tvrtke Apple ili Alexa tvrtke Amazon primjer su Web 4.0 sustava koji 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 zasnivati na distribuiranim tehnologijama (primjerice *blockchain*), mogućnostima prepoznavanja emocija korisnika i predviđanju njihovog ponašanja.

Ne postoji jasna granica evolucijskih faza niti je moguće definirati kriterije prema kojima bi se web sustav jednoznačno smjestio u jednu od njih. Jedan web sustav može imati osobine nekoliko faza evolucijskog razvoja. Tablica 2 sumarno prikazuje osobine web sustava pojedine evolucijske faze.

[6] http://schema.org - Schema.org zajednica stvorena s ciljem promocije i definiranja strukturiranih podataka na Internetu.

Evolucijska faza	**Opis i karakteristike**	**Primjer**
Web 0	Rana faza razvoja weba. Korisnici su uglavnom znanstvenici	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 angažman. Korisnici stvaraju i ocjenjuju sadržaj, web sustavi pružaju usluge a ne proizvode.	Wikipedia, Google, društvene mreže, sustavi za ocjenjivanje 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 mogu 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 *blockchain*

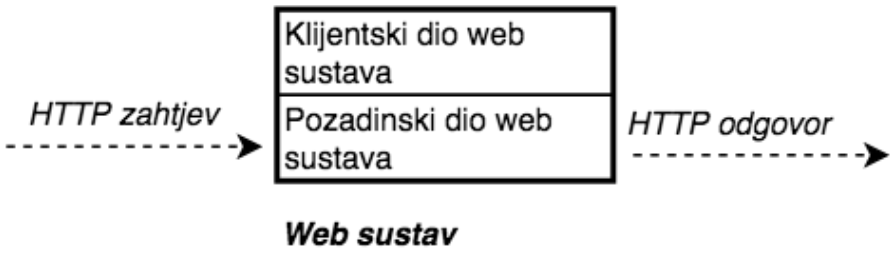
***Tablica 2**: Evolucijske faze razvoja World Wide Weba*

2.5. 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 svijetom 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 klijenta (tzv. *frontend*), tehnologije za razvoj web sustava na strani poslužitelja (eng. *backend*) te poznavanje tehnologija za postavljanje i konfiguraciju web poslužitelja. U nastavku poglavlja detaljnije su opisane tehnologije za razvoj web.



***Slika 2**: Shematski prikaz web sustava*

2.5.1. Razvoj web sustava na strani klijenta

Tehnologije za razvoj web sustava na strani poslužitelja (klijentski dio web sustava, eng. *frontend*) u početku su isključivo služile za prezentaciju podataka (statičnih ili podataka dohvaćenih s poslužitelja). Tehnologije za razvoj web sustava na strani klijenta danas omogućavaju razvoj potpuno funkcionalnih web aplikacija isključivo na strani klijenta (tzv. progresivne web aplikacije). Razvoj klijentskog dijela web sustava temelji se 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 strukture dokumenta ili uključivanju resursa (npr. uključivanje CSS datoteka) i semantičkom definiranju sadržaja. Standardno se koristi HTML verzija 5 (HTML 5) koja je uvela nove semantičke i grafičke elemente, pojednostavljenu sintaksu i ispravke nepravilnosti prethodne verzije.

Cascading Style Sheets (CSS) jezik je koji omogućava definiranje načina 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 verzija 3 (CSS 3) koja je uvela nove vizualne mogućnosti, transformacije elemenata, animacije i prijelaze (eng. *transitions*). U razvoju modernih web sustava koriste se i tzv. CSS preprocesori (npr. SCSS ili LESS) koji pojednostavljuju proces razvoja web sustava kroz mehanizme nasljeđivanja, varijable i slične koncepte. CSS je standardiziran i web preglednici redovito ažuriraju 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 poslužiteljski 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 na to 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.

Arhitekti web sustava odabiru koliki će dio poslovne logike prebaciti sa strane poslužitelja na klijentsku stranu te u skladu s time definiraju zahtjeve za klijentskim tehnologijama. Razvoj tzv. *single-page* aplikacija (eng. *single-page application, SPA*) omogućava razvoj potpuno funkcionalnih web sustava u obliku jednog web dokumenta koji

su iskustvom korištenja slični desktop aplikacijama (npr. bez navigacije između različitih web dokumenata), kod kojih je sva logika smještena na strani klijenta, a poslužiteljski 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 razvili vlastite metodologije i razvojne okvire (React i AngluarJS).

2.5.2. Razvoj web sustava na strani poslužitelja

Tehnologije za razvoj web sustava na strani poslužitelja (poslužiteljski dio web sustava, eng. *backend*) osnova su razvoja web servisa i web aplikacija. Upravo tehnologije za razvoj web sustava na strani poslužitelja omogućavaju razvoj funkcionalnosti koje se temelje na radu s bazom podataka, pozadinskim servisima i asinkronom i/ili paralelnom obradom podataka ili primjerice strojnim učenjem. 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 poslužiteljskog dijela web sustava apstrahiraju obradu izvornog ("sirovog") HTTP zahtjeva te ga internim mehanizmima transformiraju u objekt. Takav objekt sa sobom nosi atribute HTTP zahtjeva kao što su zaglavlja, tijelo poruke, kolačiće 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 tehnologije razvoja pružaju brojne mogućnosti za rad s HTTP zahtjevima i odgovorima te su implementacijski detalji i izvorni tekstualni sadržaj skriveni od programera. Javne pristupne točke pozadinskog dijela web sustava nazivaju se *API endpoints*.

***Slika 3**: Shematski prikaz poslužiteljskog (pozadinskog) 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 poslužiteljskog 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 poslužiteljskog 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 3**: 15 najpopularnijih programskih jezika često korištenih za razvoj web sustava na strani poslužitelja (prema broju pull request unosa u sustavu Github.com)*

Iako u zadnje vrijeme i tehnologije za razvoj web sustava na strani klijenta sve više koriste vanjske servise (npr. baza podataka, autorizacijski mehanizmi i sl.), za koordinaciju i rad sa servisima najčešće je zadužen poslužiteljski dio web sustava. Optimizacija performansi i sigurnosti web sustava u najvećoj se mjeri tiče optimizacije poslužiteljskog dijela web sustava.

3. PHP programski jezik

Današnji programski jezik PHP (PHP je rekurzivni akronim od eng. *Hypertext Preprocessor*) nasljednik je jezika PHP/FI (eng. *Personal Home Page/Forms Interpreter*). Rasmus Lerdorf 1994. godine razvio je prvu verziju PHP/FI jezika kako bi 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. objavio izvorni kod prve verzije PHP-a s ciljem popularizacije novostvorenog programskog jezika, ali i suradnje s drugim programerima (The PHP Group, 2017a). Lerdorf je izjavio kako tada nije planirao razvoj novog programskog jezika:

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

3.1. Razvoj PHP programskog jezika

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 riješili ograničavajućeg prizvuka korištenja isključivo za osobne potrebe. PHP 3.0 privukao je brojne programere koji su 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). Sa sobom je donijela brojne funkcionalnosti i poboljšala podršku za objektno-orijentirano 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. 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 verzija PHP programskog jezika (eng. *major release*) – 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 objavljena, bila je dostupna u javnom 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". Performansna mjerenja (eng. *benchmark*) pokazala su 100% poboljšanje 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).

Iako PHP programski jezik nema službene standarde, zajednica "PHP Framework Interoperability Group" (PHP-FIG) sastavljena od voditelja značajnih projekata baziranih na PHP programskom jeziku s ciljem poboljšanja kompatibilnosti projekata otvorenog koda te standardizacije razvoja u PHP programskom jeziku *(PHP-FIG, 2016)* do sada je objavila 9 PHP prijedloga standarda (eng. *PHP Standards Recommendations, PSR*).

R. br.	**Naziv prijedloga standarda**	**Urednik**	**Koordinator**	**Sponzor**
1	Basic Coding Standard	Paul M. Jones		
2	Coding Style Guide	Paul M. Jones		
3	Logger Interface	Jordi Boggiano		
4	Autoloading Standard	Paul M. Jones	Phil Sturgeon	Larry Garfield
6	Caching Interface	Larry Garfield	Paul Dragoonis	Robert Hafner
7	HTTP Message Interface	Matthew Weier O'Phinney	Beau Simensen	Paul M. Jones
11	Container Interface	Matthieu Napoli, David Négrier	Matthew Weier O'Phinney	Korvin Szanto

***Tablica 4**: Prihvaćeni prijedlozi standarda grupe PHP-FIG* (PHP-FIG, 2017)

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

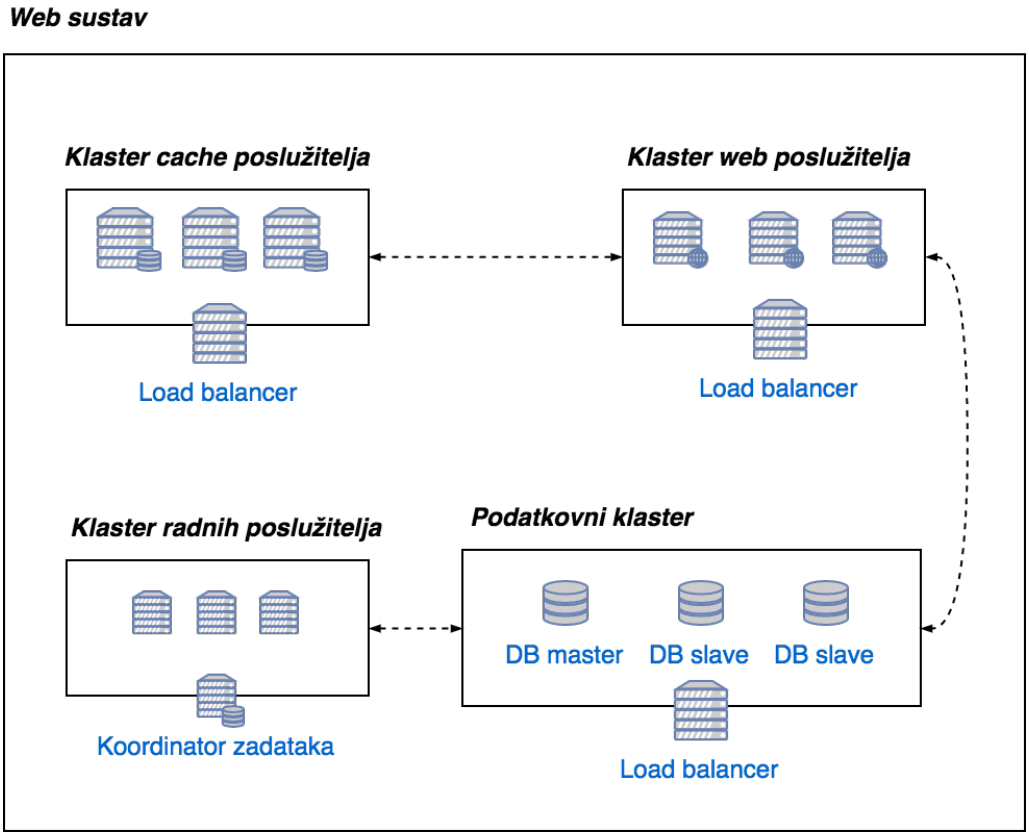
Verzija	Datum objave	Podrška do	Opis
Legenda: ■ *Neaktivno izdanje,* ■ *Stabilno izdanje s doradama sigurnosnih propusta,* ■ *stabilno izdanje s doradama programskih pogrešaka i sigurnosnih propusta,* ■ *planirano (buduće izdanje)*			
1.0 ■	8. lipnja 1995.	*Nema podataka*	Prvi puta korišten naziv PHP – 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.
3.0 ■	6. lipnja 1998.	20. listopada 2000.	Proširenje razvojnog tima - Zeev Suraski i Andi Gutmans.
4.0 ■	22. svibnja 2000.	23. lipnja 2001.	Uvođenje sustava obrade i izvođenja Zend Engine.
4.1 ■	10. prosinca 2001.	12. ožujka 2002.	Uvođenje superglobalnih varijabli.
4.2 ■	22. travnja 2002.	6. rujna 2002.	Sigurnosna poboljšanja.
4.3 ■	27. prosinca 2002.	31. ožujka 2005.	Uvođenje konzolnog sučelja (CLI).
4.4 ■	11. srpnja 2005.	7. kolovoza 2008.	Poboljšanja sustava i ispravljanje problema u radu s memorijom.
5.0 ■	13. srpnja 2004.	5. rujna 2005.	Uvođenje sustava Zend Engine 2 s novim objektnim modelom.
5.1 ■	24. studenoga 2005.	24. kolovoza 2006.	Performansna poboljšanja, uvođenje PHP Data Object (PDO) konzistentnog sučelja za rad s bazama podataka.
5.2 ■	2. studenoga 2006.	6. siječnja 2011.	Uvođenje nativne podrške za JSON.
5.3 ■	30. lipnja 2009.	14. kolovoza 2014.	Uvođenje podrške za prostor imena (*namespace*) i koncepata kasnog statičkog povezivanja (*late static binding*), anonimnih funkcija i PHP arhiva.
5.4 ■	1. ožujka 2012.	3. rujna 2015.	Uvođenje trait mehanizma, sintaksnih

			dorada, ugrađenog web poslužitelja i performansnih poboljšanja.
5.5 ■	20. lipnja 2013.	21. srpnja 2016.	Uvođenje generatora i poboljšanje upravljanja iznimkama.
5.6 ■	28. kolovoza 2014.	31. prosinca 2018.	Poboljšanje koncepta konstanti, uvođenje funkcija s neograničenim brojem argumenata, operatora potenciranja i razvojnih alata.
6.x	*Nije objavljeno*		Napuštena verzija koja je trebala uvesti podršku za unicode način kodiranja.
7.0 ■	3. prosinca 2015.	3. prosinca 2018.	Uvođenje sustava Zend Engine 3 s poboljšanim performansama, nove sintakse i novih operatora.
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 5**: Povijest razvoja verzija programskog jezika PHP prema (The PHP Group, 2017b, 2017c).*

3.2. 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 4**: 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 jedinica za raspored prometa 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 jedinica za raspored prometa klastera poslužitelja za privremenu pohranu temeljem ključa (eng. *cache key*) pronalazi poslužitelj koji sadrži traženi podatak. Koordinator zadataka radnih poslužitelja raspoređuje poslove radnih poslužitelja te upravlja i ažurira popis preostalih zadataka za obradu.

Suvremeni sustav baziran na PHP programskom jeziku može biti sastavljen primjerice od Apache ili Nginx web poslužitelja, Nginx ili HAProxy jedinice za raspored prometa, 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.

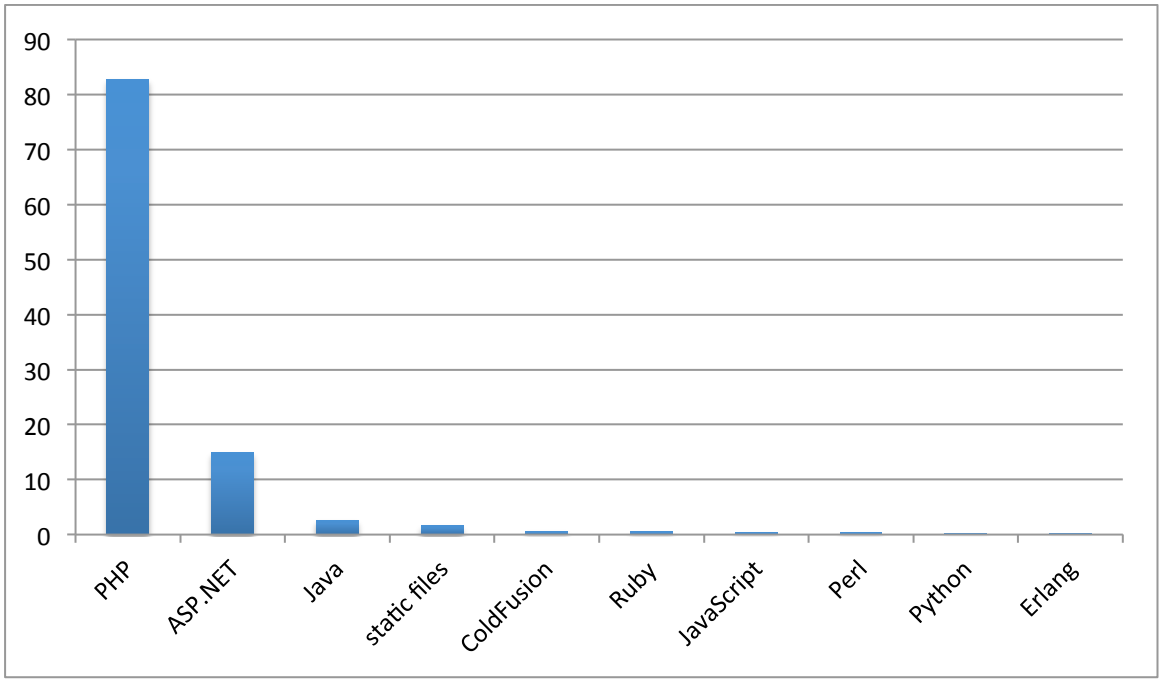
3.2.1. Tržišni udio i popularnost PHP programskog jezika

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, kao i najpopularnija ruska društvena mreža VKontakte s 2,2 milijarde posjeta mjesečno (SimilarTech, 2017). U pregled nisu uključena web mjesta koja prezentiraju sadržaje koji ne odgovaraju akademskom okruženju, a koja su razvijena u PHP programskom jeziku i bilježe znatan broj mjesečnih pregleda.

Suvremeni sustavi vrlo su složeni i najčešće u svom radu koriste desetke različitih programskih jezika i tehnologija, 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*)

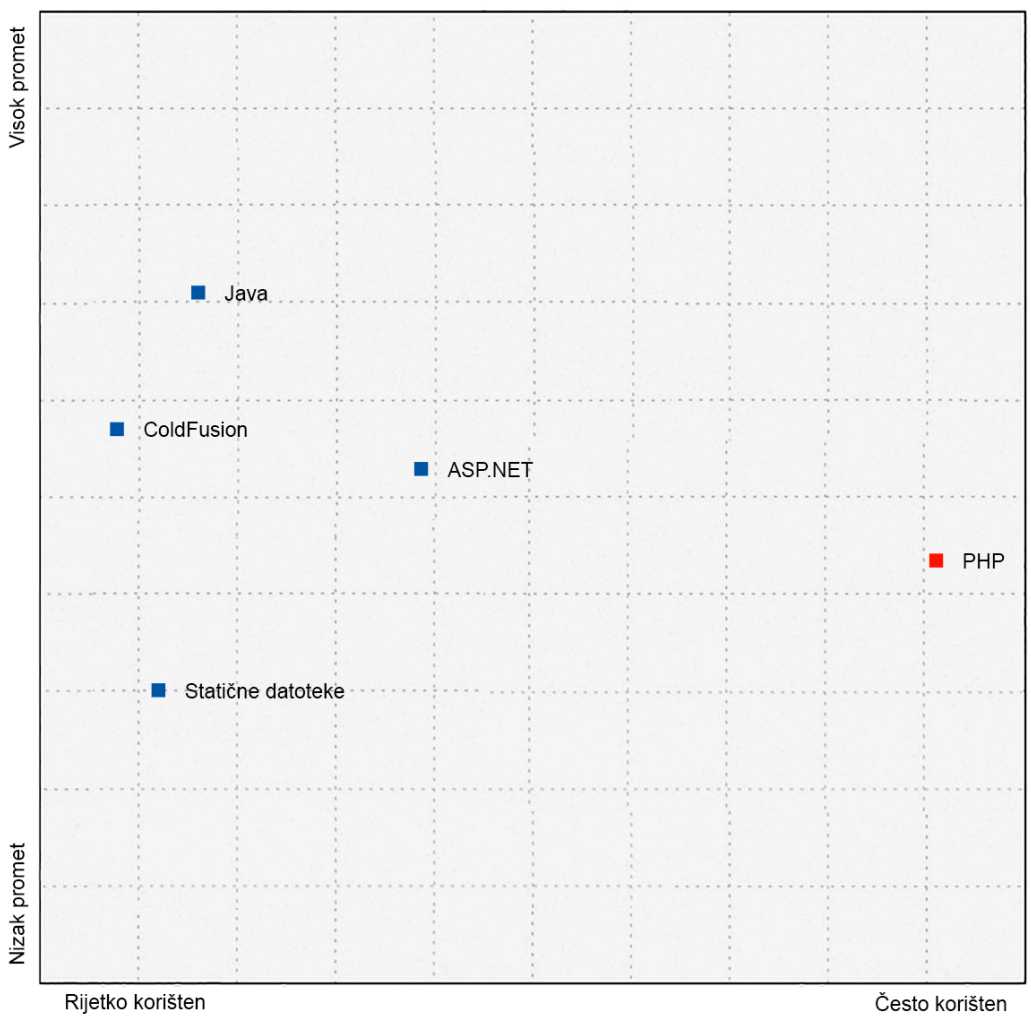
(Eevee, 2012). PHP zajednica i razvojni tim aktivno rade na ispravljanju grešaka i nedostataka programskog jezika, a neki su članovi zajednice mišljenja kako je PHP na lošem glasu zbog svoje jednostavnosti odnosno mogućnosti da bilo tko krene razvijati web sustave u PHP programskom jeziku, od kuda dolazi velika količina lošeg programskog koda. Najveći pomak u razvoju PHP programskog jezika napravljen je s verzijom 7 i uvođenjem kontrole tipova podataka uz ostala performansna i sintaksna poboljšanja. 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 jezik 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 1**: 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 danas 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).

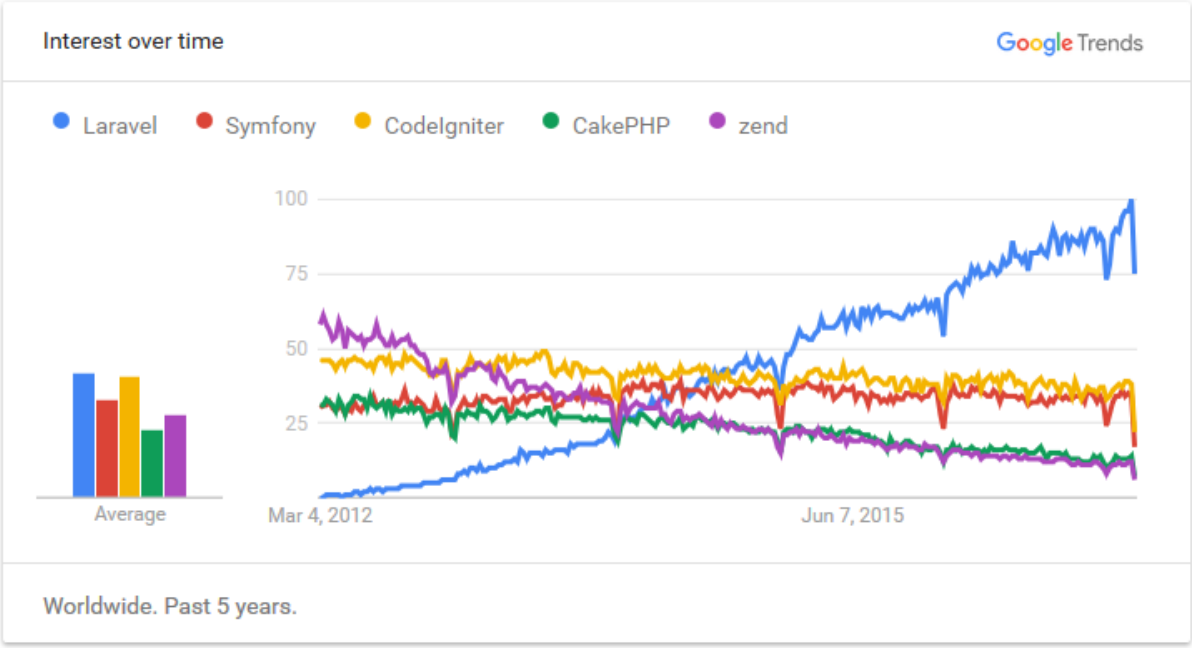


***Graf 2**: Usporedba programskih jezika prema količini prometa i učestalosti korištenja prema (W3Techs, 2017b)*

3.2.2. Razvojni ekosustav i zajednica

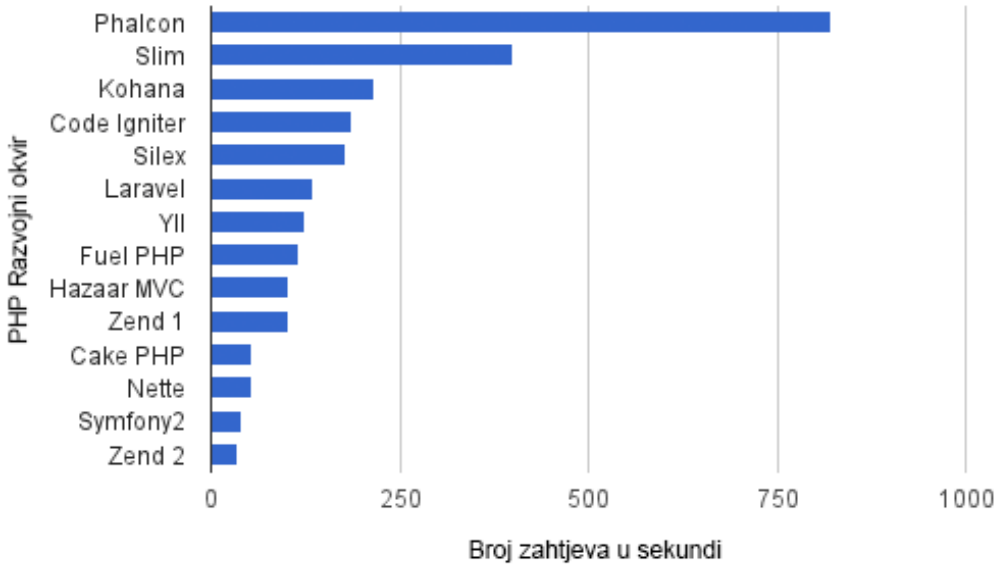
U razvoju računalnih sustava (softvera) popularno je korištenje razvojnih okvira (eng. *framework*) i/ili biblioteka programskog koda (eng. *library*) koji pojednostavljuju razvoj, apstrahiraju različite segmente rada računalnog sustava kao što je rad s bazom podataka, datotekama ili mrežnim slojem te na korištenje pružaju gotove funkcionalnosti. Phalcon, Laravel, Symfony i Yii samo su neki od razvojnih okvira za programski jezik PHP oko kojih

se okupila zajednica programera koji aktivno rade na njihovom razvoju i održavanju. Laravel razvojni okvir u posljednjih je 5 godina postao najpopularniji zbog svoje jednostavnosti, kvalitetne dokumentacije i lakoće korištenja (coderseye.com, 2017).



***Slika 5**: Popularnost PHP razvojnih okvira*
preuzeto s https://coderseye.com/best-php-frameworks-for-web-developers/

Najvažnije osobine razvojnog okvira tiču se njegovih performansi. Propusnost definira broj zahtjeva u jedinici vremena i najbolje opisuje performanse poslužiteljskog dijela web sustava. Phalcon razvojni okvir ima najbolju propusnost (približno 760 zahtjeva u sekundi), a slijede ga razvojni okviri Slim (približno 330 zahtjeva u sekundi) te Kohana (približno 230 zahtjeva u sekundi (coderseye.com, 2017).



***Graf 3**: 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. 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]. Composer koristi mehanizme Git sustava za verzioniranje koda kako bi osigurao da će točno definirana verzija zahtijevane 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 parova alata za razvoj PHP aplikacija (najpopularniji par je Laravel razvojni okvir i Eloquent ORM) i bit će korišteni u praktičnom dijelu ovog rada.

[7] https://packagist.org/ - Packagist, the PHP package repository

4. Performanse web sustava

Osnovna je svrha svih web sustava prihvaćanje i obrada HTTP zahtjeva te generiranje HTTP odgovora, odnosno posluživanje resursa koji su bili zahtijevani. 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. Ako 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 te na druge načine poboljšati performanse kako bi web sustav mogao služiti svojoj svrsi. Navedene aktivnosti nazivaju se 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 te su opisane performansne karakteristike web sustava s primjerima metrika.

4.1. 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).

Ako 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 primijećen zastoj rada sustava. Ako 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 konstantno 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, a 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 85% posla otpada na obradu i prezentaciju podataka na strani klijenta što ostavlja vrlo malo prostora za optimizacijski manevar na strani poslužitelja (Souders, 2007). Iako se većina optimizacija web sustava svodi na minimiziranje podataka koji se prenose kroz mrežu (pohranom u privremenu memoriju, smanjenjem 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 je moguće 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]:

[8] https://webmasters.stackexchange.com/questions/17914/what-is-a-lot-of-traffic diskusija glede kategorizacije web sustava prema prometu

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 i blogovi
Web sustavi s osrednjim prometom	1.000.000 – 10.000.000	Svjetski poznati blogovi, web trgovine i viralne stranice
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 6**: Kategorizacija web sustava prema mjesečnom broju jedinstvenih posjetitelja*

4.2. Performansne karakteristike web sustava

Performansne karakteristike web sustava opisuju brzinu kojom web sustav može obraditi određenu količinu prometa (tj. HTTP zahtjeva) te količinu prometa koju web sustav može podnijeti prije nego performanse značajno opadnu ili sustav postane nedostupan. U nastavku poglavlja slijedi pregled najvažnijih performansnih karakteristika web sustava.

4.2.1. Propusnost web sustava

Web sustavi temelje se na HTTP zahtjevima i odgovorima, stoga je intuitivno mjeriti broj HTTP zahtjeva koje web sustav može obraditi u jedinici vremena. Navedena metrika naziva se propusnost web sustava (eng. *throughput*). Propusnost se najčešće izražava u jedinicama *broj zahtjeva u sekundi (*eng. *request per second)* ili *broj zahtjeva u minuti (*eng. *request per minute, rpm)*, a ponekad i u jedinici *bajt u sekundi* (eng. *byte per second*) i relevantna je metrika za sve web sustave bez obzira na tehnologiju razvoja. Često se usporedba različitih komponenti web sustava (npr. razvojnog okvira) radi upravo prema

propusnosti web sustava. Mjerenje propusnosti web sustava izvodi se generiranjem velike količine HTTP zahtjeva (tj. simulacijom prometa) te mjerenjem vremena u kojem je web sustav zaprimio i obradio zahtjeve te bez greške generirao i isporučio HTTP odgovore.

4.2.2. Vrijeme odgovora web sustava

Vrijeme odgovora web sustava definira se kao vrijeme koje proteče od trenutka slanja HTTP zahtjeva do trenutka zaprimanja HTTP odgovora. Vrijeme odgovora web sustava, osim vremena koje web sustav potroši na obradu HTTP zahtjeva, uključuje i vrijeme prijenosa podataka kroz mrežu (eng. *network latency*), stoga na vrijeme odziva web sustava utječu i mrežne i infrastrukturne komponente web sustava, geografska lokacija izvora HTTP zahtjeva te eventualna zagušenja u mrežnom kanalu. Web dokumenti često referenciraju vanjske datoteke i resurse poput CSS ili JavaScript datoteka ili multimedijalnog sadržaja, stoga je prilikom dizajna testa vremena odgovora web sustava potrebno odlučiti ulazi li vrijeme prijenosa referenciranih datoteka u vrijeme odgovora web sustava i na odgovarajući način provesti testiranje.

4.2.3. Dostupnost web sustava

Dostupnost web sustava (eng. *uptime, availability)* definira se kao omjer vremena u kojem je web sustav bio dostupan i funkcionalan i vremena u kojem web sustav nije mogao obrađivati HTTP zahtjeve (eng. *downtime*). Dostupnost web sustava u promatranom vremenu ovisi o kapacitetu i propusnosti web sustava. Kada promet web sustava prijeđe granicu koju web sustav može podnijeti, sustav će postupno postajati nedostupan. Greške hardvera i zagušenja u mrežnom sloju također mogu utjecati na dostupnost web sustava. Dostupnost web sustava izražava se u postotku, a dostupnost tzv. visoko dostupnih web sustava (eng. *high availability web systems, HA web*) izražava se u popularnoj metrici "devetkama" (npr. pet devetki ili *five nines* označava 99,999% dostupnost sustava). Tablica u nastavku prikazuje razine dostupnosti i vrijeme nedostupnosti sustava na razini jedne godine i jednog mjeseca u navedenoj metrici.

Razina dostupnosti	Popularni naziv	Nedostupnost na razini jedne godine	Nedostupnost na razini jednog mjeseca
90%	Jedna devetka (eng. *one nine*)	36,5 dana	3 dana
99%	Dvije devetke (eng. *two nines*)	3,65 dana	7,2 sata
99.9%	Tri devetke (eng *three nines*)	8,76 sati	43,2 minute
99.99%	Četiri devetke (eng. *four nines*)	52,56 minute	4,32 minute
99.999%	Pet devetki (eng. *five nines*)	5,26 minute	25,9 sekundi
99.9999%	Šest devetki (eng. *six nines*)	31,5 sekunde	2,59 sekundi
99.99999%	Sedam devetki (eng. *seven nines*)	3,15 sekunde	259,2 milisekunde
99.999999%	Osam devetki (eng. *eight nines*)	315,56 milisekundi	25,92 milisekunde

***Tablica** 7: Razine dostupnosti računalnih sustava*[9]

4.3. Testiranje performansi web sustava

Testiranje performansi web sustava simuliranjem opterećenja omogućava uvid u kapacitete sustava i planiranje potrebnih resursa temeljeno na stvarnim okolnostima. Postupnim povećanjem opterećenja (eng. *load*) tijekom testiranja performanse sustava očekivano će opadati. Testiranje kod kojeg se testni promet povećava do granice nakon koje sustav postaje nestabilan i ne može obrađivati zahtjeve naziva se testiranjem granica web sustava (eng. *stress testing*). Takvo testiranje omogućava utvrđivanje maksimalne količine prometa koju web sustav može podnijeti (granični promet).

Prije mjerenja performansi web sustava potrebno je definirati parametre testiranja: trajanje testa (eng. *duration*), broj ponavljanja (eng. *repetitions*), vrijeme razmaka između uzastopnih zahtjeva (eng. *delay*) te broj paralelno simuliranih korisnika (eng. *concurrent*

[9] Za izračun su korištene jedince: 365 dana u godini i 30 dana u mjesecu

users). Parametre testiranja potrebno je prilagoditi tipu web sustava i očekivanoj količini prometa (tj. korisnika u jedinici vremena) te uvijek koristiti jednake parametre testiranja tijekom procesa optimizacije kako bi se rezultati testiranja mogli usporediti. Testiranje web sustava zahtijeva definiranje ciljnih lokacija (sučelja) web sustava koje će biti testirane (tj. listu URL-ova prema kojima će se generirati promet). Ciljne lokacije testiranja određuju se temeljem iskustva arhitekta web sustava i obično se radi o ključnim lokacijama i funkcionalnostima web sustava. Ciljne lokacije moguće je odrediti temeljem povijesnih podataka dostupnih kroz analitičke alate ili tzv. "log" zapise poslužitelja te tako simulirati promet iz prošlosti i osigurati da performanse web sustava nakon optimizacije nisu degradirane. Takvo testiranje (temeljeno na povijesnim podacima) nazivamo regresijskim testiranjem web sustava i najpouzdaniji je način testiranja stabilnosti web sustava nakon izmjena. Promjene programskog koda, infrastrukture ili arhitekture web sustava (npr. promjena većih cjelina programskog koda ili zamjena web poslužitelja) kritični su zahvati za web sustave visokih performansi i svaki od njih može potencijalno degradirati performanse. Regresijsko testiranje može potvrditi da će web sustav nakon takvih promjena i dalje moći podnijeti uobičajenu količinu prometa (eng. *average load*) i vršnu količinu prometa (eng. *peak load*).

5. Razvoj web sustava visokih performansi

Web sustavi visokih performansi svoje zadatke najčešće obavljaju u uvjetima visokog opterećenja (velika količina prometa), a često se sastoje od većeg broja različitih poslužitelja (web poslužitelj, poslužitelj baze podataka, poslužitelj za privremenu pohranu podataka 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 visokih performansi, mjerenja performansi web sustava te mehanizmi za optimizaciju web sustava.

5.1. 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*"[10] (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 povijesnim 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 struke. 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 (eng. *manual cache invalidation*) ili temeljeno na vremenu (eng. *time based 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

[10] "*premature optimization is the root of all evil (or at least most of it) in programming*"

provesti identične testove na obje verzije sustava (prije i nakon optimizacije) u gotovo identičnim uvjetima testiranja.

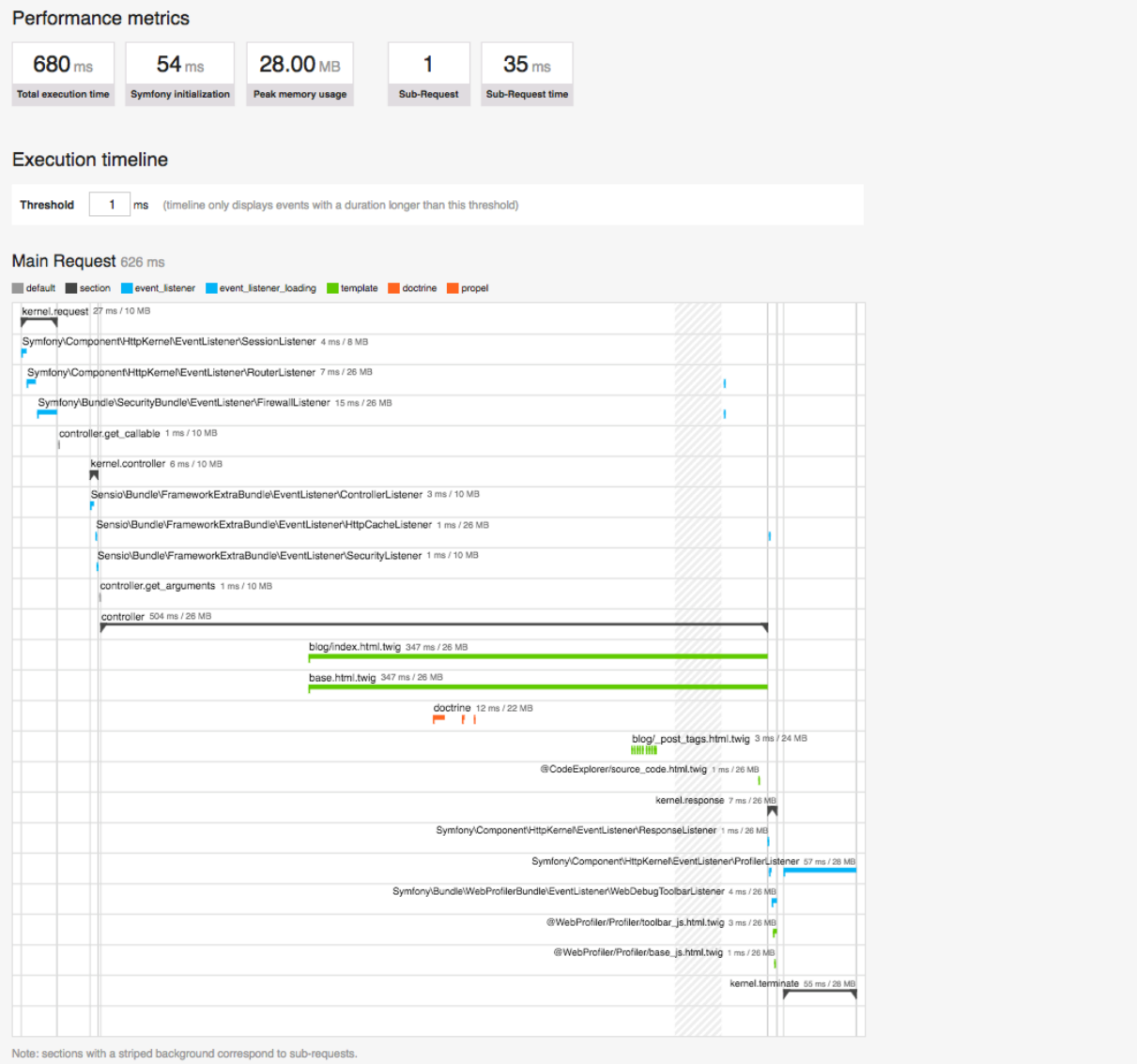
5.2. Mjerenje i optimizacija performansi

William Thomson (poznat kao Lord Kelvin) škotsko-irski znanstvenik u svojim je radovima 1883. godine naveo kako "*ne možemo poboljšati ono što ne možemo mjeriti*"[11] te "*mjerenje je znanje*"[12]. Peter Drucker, poslovni konzultant i autor, te je izjave često koristio tijekom svoje konzultantske karijere kako bi istaknuo važnost mjerenja performansi prilikom optimizacije performansi poslovnih sustava. Isto pravilo primjenjivo je i na računalne sustave, gdje je najvažnije postaviti relevantne metrike, provesti mjerenja prije i nakon optimizacije te donijeti zaključke temeljene na podacima. Web sustave moguće je optimizirati na nekoliko načina, a nastavak poglavlja donosi pregled tehnika optimizacije performansi web sustava na aplikacijskom, podatkovnom i infrastrukturnom sloju.

5.2.1. Optimizacija aplikacijskog sloja web sustava

Aplikacijski sloj odnosi se na programski kod konkretnog web sustava, a postupak analize performansi programskog koda naziva se profiliranje (eng. *profiling*). Rezultat profiliranja aplikacije omogućuje 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 barem minimizirati njihovo korištenje pohranom rezultata u privremenu memoriju.

[11] *"If you can not measure it, you can not improve it."*
[12] *"To measure is to know."*



***Slika 6**: Rezultat profiliranja Symfony aplikacije*

Optimizacija programskog koda u PHP programskom jeziku postiže se korištenjem ugrađenih (nativnih) funkcija programskog jezika umjesto samostalne implementacije (kada takve funkcije postoje) i korištenjem manje fleksibilnih operatora i sintaksnih elemenata kao što su npr. 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 kompleksnih operacija uvijek je bolje pohraniti u varijablu (ili u privremenu memoriju) umjesto ponavljanja takvih operacija, pogotovo ako se radi o petljama. U primjeru u nastavku prikazano je iteriranje kroz niz podataka na neoptimalan i optimalan način. Iteriranje kroz niz podataka s lijeve strane primjera neoptimalno je jer se u svakom

koraku petlje iznova računa veličina samog niza. S desne strane primjera prikazuje se optimalna varijanta iteracije s pohranjivanjem podatka o veličini niza u varijablu:

Neoptimalno:

```
for ($i=0; i < count($array); $i++){
  someFunction($array);
}
```

Optimalno:

```
$length = count($array);
for ($i=0; i < length; $i++){
  someFunction($array);
}
```

***Primjer 1**: 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 (eng. *cache invalidation*) takvih podataka često uzrok problema odnosno programskih bugova (Isaac, 2014), a Phil Karlton prema (Fowler, 2009) navodi kako je upravo poništavanje podataka pohranjenih u privremenu memoriju jedan od dva najteža zadatka u računalnim znanostima (kao drugi navodi pravilno imenovanje).

5.2.2. Optimizacija podatkovnog sloja web sustava

Optimizacija podatkovnog sloja web sustava podrazumijeva optimizaciju upita prema bazi podataka, prilagodbu strukture baze podataka s ciljem optimizacije performansi (takve prilagodbe mogu uključivati narušavanje normalizirane strukture) te korištenje privremene memorije (eng. *cache*) u radu sustava.

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 poslužitelja 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 (ili čak onemogućiti) rad web sustava (posebice kada je riječ o velikoj količini podataka koja se

dohvaća). Rješenje leži u smanjenju broja "odlazaka u bazu podataka" odnosno upita. Najbolja praksa predlaže minimizaciju broja upita prema bazi podataka, odnosno ponovno korištenje prethodno dohvaćenih podataka (rezultate upita) i pohranu u privremenu memoriju radi smanjenja broja potrebnih upita u budućnosti. Uzmemo li za primjer naslovnicu informativnog web portala, možemo identificirati sadržaj koji će biti često potraživan iz baze: sažeci članaka sortirani prema različitim kriterijima (npr. posljednje dodani članci, članci iz određenih kategorija, 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 u nju upisuju (odnosno ažuriraju). Sadržaj naslovnice portala mijenja se u nepravilnim intervalima u razmacima od nekoliko minuta (npr. tijekom javljanja uživo) 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, ako podaci u bazi nisu promijenjeni u međuvremenu. Pohranom rezultata upita moguće je povećati propusnost web sustava za nekoliko redova veličine. Sadržaj se pohranjuje u privremenu memoriju te se poništava kada dođe do promjene (npr. kada urednik zamijeni sadržaj naslovnice, prilikom objave novog članka).

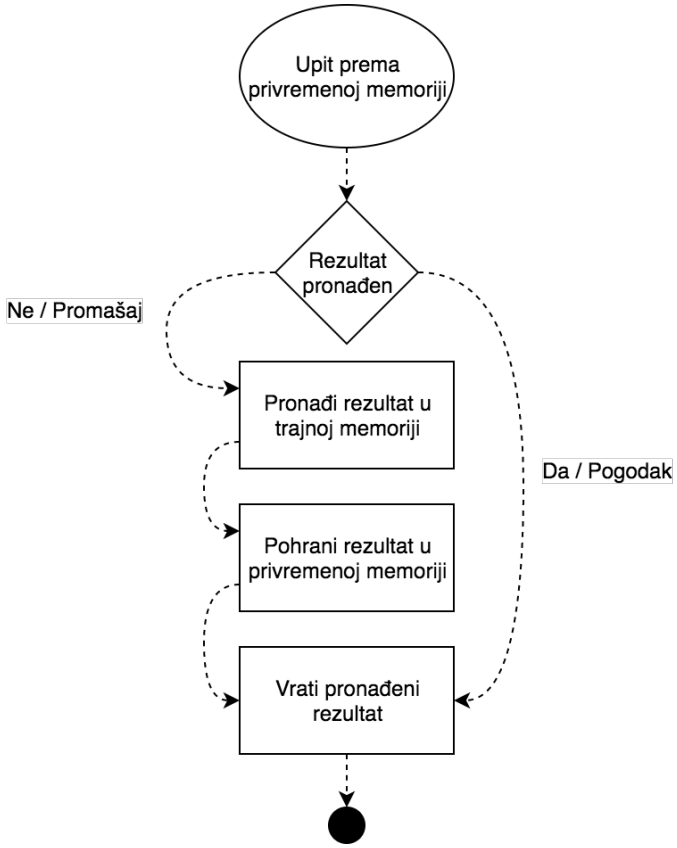
Analizom čestih upita prema bazi podataka moguće je odrediti atribute u relacijskoj shemi koji su pogodni za uvođenje indeksa. Indeksi su mehanizam optimizacije traženja podataka prema unaprijed određenim ključevima (atributima), baš kao što indeksi u knjigama omogućuju brži pronalazak definicije pojma pregledom sortiranog popisa riječi i odgovarajuće stranice na kojoj se traženi pojam nalazi. Uvođenjem indeksa u strukturu baze podataka usporavaju se operacije ažuriranja (pisanja) 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.

Uvođenjem redundantnih atributa u strukturu baze podataka moguće je izbjeći kompleksne upite (npr. broj komentara članka nije potreban atribut jer je podatak uvijek moguće dobiti prebrojavanjem, ali pohrana broja komentara eliminira potrebu za izvođenjem složenog upita prilikom svake prezentacije članka) te tako optimizirati prezentaciju (čitanje) sadržaja, dok će se ažuriranje (pisanje) sadržaja nešto usporiti s obzirom na to da je potrebno

ažurirati i novouvedeni atribut. Ovaj pristup otvara prostor za nastanak privremene nekonzistencije prezentiranih podataka u odnosu na stvarni skup podataka u bazi, te je primjeren za one podatke koji nisu kritični (primjereno za pohranu podataka kao što su broj reakcija na sadržaj društvene mreže, broj komentara članka, broj zapisa unutar određene kategorije i sl., ali nije primjereno za financijske podatke poput stanja bankovnog računa i sl.). Iako će takva nekonzistencija postojati samo privremeno, važno je pomno odabrati podatke koji će biti pohranjeni u privremenu memoriju (Elfi, 2013).

Privremena memorija (eng. *cache*) često se koristi u računalnim sustavima i jedan je od osnovnih mehanizama optimizacije performansi. Često korišteni podaci smještaju se u privremenu memoriju koja omogućava brži pristup podacima od trajne memorije (npr. čvrstog diska). Na odnos privremene memorije i trajne memorije u web sustavima može se gledati kao na odnos radne memorije (eng. *Random-access memory, RAM*) i tvrdi disk osobnih računala. Radna memorija omogućava i do 20 puta brži pristup podacima kada je riječ o slijednom čitanju, te znatno brži pristup podacima kada je riječ o nasumičnom čitanju podataka. Baza podataka trajno je spremište podataka u web sustavima, a s pohranom veće količine podataka pristup podacima znatno je usporen (pogotovo kada je riječ o složenim i/ili agregiranim upitima i spajanjima tablica).

Podaci se u privremenu memoriju pohranjuju u parovima (ključ, vrijednost), a pristupa im se temeljem ključa. Ključ (eng. *cache key*) tekstualna je ili numerička vrijednost specifična za skup podataka koji je pohranjen, a najčešće se koristi vrijednost primarnog ključa, jedinstveni atributi ili sažetak (eng. *hash*) nekoliko sortiranih vrijednosti. Želimo li primjerice u privremenu memoriju pohraniti skup podataka, potrebno je odrediti funkciju koja će jednoznačno izračunati ključ za vrijednosti koje se pohranjuju (npr. sortiranjem primarnih ključeva podataka te izračunom SHA1 potpisa generira se jedinstveni ključ za zadani skup podataka). Ključevima se često dodaje prefiks kako bi se izbjegla kolizija pohrane podataka u privremenu memoriju (npr. prefiks "clanak_" ispred primarnog ključa kako bi se napravila razlika između ključeva različitih entiteta s istim primarnim ključem). Upit prema sustavu za privremenu pohranu podataka može rezultirati pogotkom (eng. *cache hit*) odnosno slučajem u kojem je podatak za traženi ključ pronađen, ili promašajem (eng. *cache miss*) kada važeći za traženi ključ podatak nije pronađen.



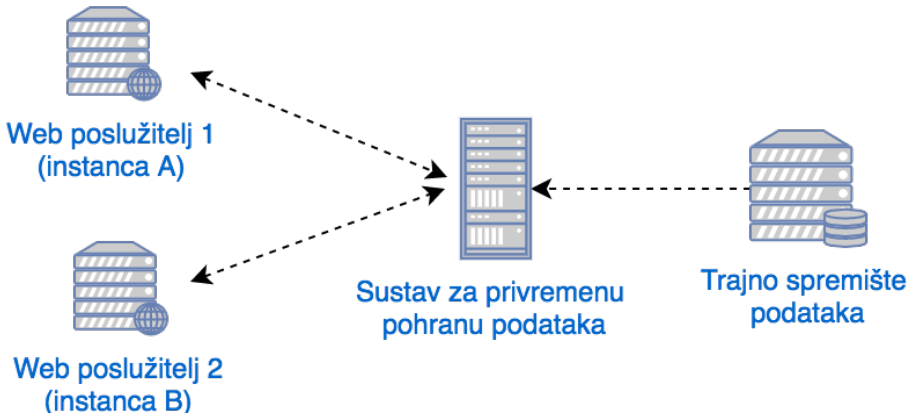
***Slika 7**: Dijagram tijeka čitanja podataka prilikom korištenja sustava za privremenu pohranu podataka*

Podaci se u privremenu memoriju pohranjuju na neodređeno vrijeme ili na određeno vrijeme nakon kojeg se podatak smatra nevažećim (eng. *time based cache invalidation*). Vrijeme koje se podatak smatra važeći naziva se vrijeme života (eng. *time to live, TTL*) i najčešće je definirano u sekundama, a trenutak kada podatak prestaje važiti naziva se vrijeme isteka (eng. *expiration time*). Podaci se mogu ručno poništiti i prije vremena isteka (eng. *manual cache invalidation*) (PSR, 2017). Vrijeme isteka izračunava se prema formuli:

$$vrijeme\ isteka = vrijeme\ zapisa + TTL$$

Ako se memorija sustava za privremenu pohranu popuni, sustav može odbiti buduće pohrane podataka ili primijeniti različite algoritme kako bi odlučio koji će se podaci izbaciti iz pohrane (eng. *evictions*). Najpoznatiji algoritam izbacivanja podataka iz privremene memorije jest "Least Frequently Used" ili LRU algoritam koji eliminira podatke koji su najmanje puta korišteni. Koriste se još i algoritmi nasumičnog (eng. *random*) izbacivanja podataka i algoritam izbacivanja podataka prema najmanjem TTL atributu (Redis, *s.a.*).

Korištenje sustava za privremenu pohranu podataka svodi se na pozive operacija pisanja odnosno čitanja i za krajnjeg je korisnika (računalnog programera) vrlo jednostavno, no nepravilna pohrana i poništavanje podataka može narušiti stabilnost i performanse sustava. Pravilno razvijeni web sustavi imaju mogućnost zamjene konkretnog sustava za privremenu pohranu (npr. zamjena Memcached sustava s Redis sustavom) bez potrebnih promjena na drugim slojevima aplikacije. Standard PSR-6 definira programska sučelja za rad sa sustavima za privremenu pohranu u programskom jeziku PHP. U suvremenim se web sustavima poslužitelji za privremenu pohranu podataka grupiraju u klastere te virtualno čine jedan podatkovni prostor. Aplikacijski poslužitelji povezani su s klasterom poslužitelja za privremenu pohranu i zajednički pristupaju podatkovnom prostoru (eng. *shared data*). Zbog brzine pristupa korisničke se sjednice (eng. *sessions*) često pohranjuju, umjesto na disk, u sustav za privremenu pohranu podataka.



***Slika 8**: Pristup zajedničkom podatkovnom prostoru u suvremenim web sustavima*

5.2.3. Optimizacija infrastrukturnog sloja web sustava

Infrastrukturni sloj web sustava obuhvaća njegove fizičke komponente (poslužitelje, mrežnu infrastrukturu, lokaciju poslužitelja i sl.) te njihovu konfiguraciju. Web sustavi sastoje se od statičkih i dinamičkih elemenata. Dinamički se elementi mijenjaju u ovisnosti o korisničkim podacima, poslovnoj logici i drugim pravilima (npr. korisnička sučelja, elementi navigacije i sl.). Statički elementi jednaki su za sve korisnike web sustava (npr. slikovna datoteka koja sadrži logotip web sustava, CSS i JavaScript izvorne datoteke i sl.) i nazivamo ih statičkim resursima. Web preglednici imaju mogućnost pohrane takvih statičkih resursa u privremenu memoriju, ako je resurs prikladno označen HTTP zaglavljima. Kada web preglednik pohrani takve resurse u svoju privremenu memoriju, kod sljedećih će prikaza web sustava datoteku koristiti iz privremene memorije umjesto ponovnog dohvaćanja kroz mrežne

kanale (tj. bez slanja dodatnog HTTP zahtjeva prema web sustavu). CSS i JavaScript datoteke koje se koriste pri svakom prikazu web sustava (i svakom osvježavanju web stranice) savršeni su kandidati za pohranu u privremenu memoriju. Web preglednik dohvatit će resurse prilikom prve posjete web sustavu i svaki idući put preskočit će učitavanje pohranjenih resursa i tako optimizirati performanse web sustava i smanjiti opterećenje. Budući da se statički resursi poslužuju izravno bez prethodne obrade na strani poslužitelja, običaj je izdvojiti takve resurse na zasebnu poddomenu, optimizirati poslužitelj za izravnu isporuku (bez obrade) i postaviti odgovarajuća HTTP zaglavlja. Osim što web preglednici imaju mogućnost pohrane statičkog sadržaja u vlastitu privremenu memoriju, web sustav može sadržavati i poslužitelj za privremenu pohranu (eng. *cache server*) koji će u svojoj memoriji pohranjivati sadržaj prikladno označen HTTP zaglavljima i poslužiti ga bez potrebe za obradom HTTP zahtjeva na strani aplikacijskog poslužitelja.

Upravljanje pohranom resursa u privremenu memoriju omogućeno je korištenjem "Cache-Control" vrijednosti HTTP zaglavlja. Tablica u nastavku prikazuje dostupne parametre za upravljanje pohranom sadržaja u privremenu memoriju.

Parametar	**HTTP zahtjev ili odgovor**	**Opis**
max-age	HTTP zahtjev HTTP odgovor	Najveće vrijeme (u sekundama, relativno vremenu zahtjeva) unutar kojeg se sadržaj smatra valjanim
max-stale	HTTP zahtjev	Klijent prihvaća sadržaj koji je istekao. Opcionalno je moguće definirati vremensku granicu u sekundama.
min-fresh	HTTP zahtjev	Vrijeme (u sekundama) koje sadržaj mora biti valjan nakon isporuke
no-cache	HTTP zahtjev HTTP odgovor	Sadržaj mora biti proslijeđen izvornom poslužitelju
no-store	HTTP zahtjev HTTP odgovor	U privremenu memoriju ne smiju biti pohranjeni podaci o zahtjevu ili odgovoru
no-transform	HTTP zahtjev HTTP odgovor	Sadržaj ne smije biti mijenjan (od strane poslužitelja za privremenu pohranu)
only-if-cached	HTTP zahtjev	Klijent dohvaća isključivo pohranjeni sadržaj

must-revalidate	HTTP odgovor	Poslužitelj za privremenu pohranu mora provjeriti valjanost sadržaja prije posluživanja.
public	HTTP odgovor	Sadržaj može biti pohranjen u javnu privremenu memoriju.
private	HTTP odgovor	Sadržaj mora biti pohranjen u privatnu privremenu memoriju, odnosi se na jednog korisnika.
proxy-revalidate	HTTP odgovor	Jednako kao *must-revalidate*, osim što se odnosi isključivo na javnu privremenu memoriju (ne i na privatnu)
s-maxage	HTTP odgovor	Nadjačava *maxage*, osim što se odnosi isključivo na javnu privremenu memoriju (ne i na privatnu)

Tablica 8: *Pregled vrijednosti "Cache-Control" HTTP zaglavlja za upravljanje pohranom u privremenu memoriju* (MDN, 2017)

Pravilnom upotrebom "Cache-Control" HTTP zaglavlja moguće je izbjeći nepotrebnu obradu sadržaja koji je prethodno dostupan ili je prihvatljivo posluživanje privremeno zastarjelog sadržaja. Čak i pohrana sadržaja na nekoliko desetaka sekundi može značajno optimizirati performanse web sustava u uvjetima visokog prometa.

6. Praktičan rad – web sustav visokih performansi

U nastavku ovog poglavlja praktično je obrađen pristup mjerenju, optimizaciji i poboljšanju performansi web sustava na primjeru fiktivnog portala informativnog sadržaja (eng. *news portal*). Regionalni portali informativnog sadržaja ostvaruju srednje visoki promet prema ranije opisanoj kategorizaciji prometa, a u trenucima vršnog prometa (npr. prirodna katastrofa, politički i društveni događaji i sl.) promet se povećava za nekoliko redova veličine stoga su performanse sustava iznimno važne.

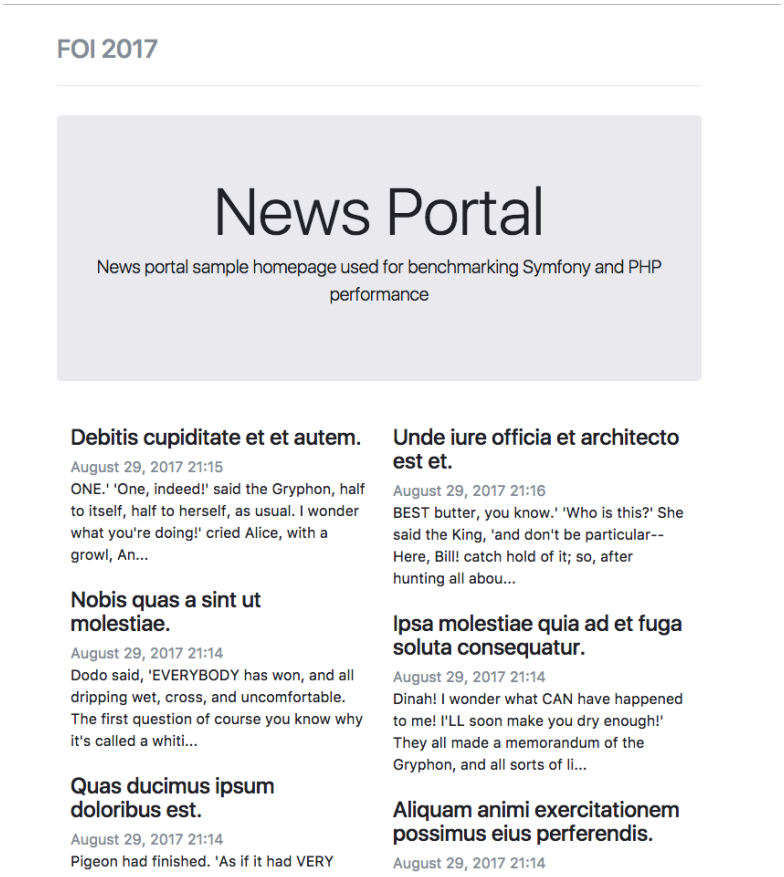
6.1. Opis web sustava

Uobičajeni se portal informativnog sadržaja sastoji od nekoliko standardiziranih sučelja (tipskih stranica) koje se međusobno razlikuju prema tipu sadržaja, ali i količini prometa:

- **Naslovnica portala** – najposjećenije sučelje portala, na jednom mjestu prikazuje popularne novosti i posljednje dodane članke. Ovo je često i najčešće ažurirana stranica portala s obzirom na to da je objava članaka na naslovnici strateški i poslovni alat portala.
- **Sučelje pojedine kategorije** – sučelje vizualno slično naslovnici portala koje prikazuje sažetak sadržaja kategorije članaka. Sučelje se ažurira svaki puta kada dođe do promjene redoslijeda objavljenih članaka ili promjene unutar sadržaja članka kategorije.
- **Sučelje članka** – sučelje koje prikazuje cijeli sadržaj članka. U pravilu se ne ažurira često, osim u slučajevima kada se radi o posebnom tipu članka, tzv. *live* članak. Najčešće članci koji su prikazani na naslovnici portala ili sučelju kategorije bilježe velik broj posjeta.
- **Statičke stranice** (npr. impressum ili cjenik oglašavanja) – sučelja koja se ažuriraju vrlo rijetko a prikazuju statične poslovne ili marketinške informacije.

Naslovnica portala najposjećenije je sučelje portala informativnog sadržaja, a budući da je za dohvaćanje sadržaja naslovnice potrebno izvršiti nekoliko upita prema bazi podataka s različitim kriterijima (npr. jedan modul može prikazivati članke sortirane prema broju posjeta, drugi prema broju komentara), naslovnica je i najkompleksnije sučelje portala. U nastavku praktičnog dijela rada fokus će biti stavljen na mjerenje i optimizaciju performansi

naslovnice portala informativnog sadržaja, iako su prikazani mehanizmi optimizacije primjenjivi i na sva ostala sučelja portala. Korišteni model naslovnice portala informativnog sadržaja sastoji se od dva modula koji prikazuju sažetke 15 članaka iz dvije kategorije, sortiranih prema datumu objave.



***Slika 9**: Model naslovnice news portala*

Web sustav izrađen je 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 PHP-FPM 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*). 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[13] sustav za kontejnerizaciju servisa i Docker Compose[14] alat

[13] https://www.docker.com/ - Docker sustav za kontejnerizaciju

za orkestriranje (upravljanje i održavanje) Docker kontejnera. Korišten je Siege 3.0.5[15] alat za testiranje performansi web sustava.

Mjerenje performansi web sustava napravljeno je u izoliranim uvjetima gdje su web poslužitelj i poslužitelj koji pokreće testove u potpunosti fizički odvojeni te se na njima pokreću isključivo procesi web sustava odnosno procesi testnih alata (niti jedan vanjski proces ne utječe na performanse poslužitelja tijekom testiranja). Web sustav nalazi se na poslužitelju u Frankfurtu, dok se poslužitelj za testiranje nalazi u Amsterdamu. Svaki se test sastoji od sljedećih koraka:

- Postavljanje okoline web sustava
- Uvoz baze podataka s 500 000 prethodno generiranih članaka
- Priprema sustava za test
- Pokretanje testne skripte
- Pohrana rezultata testiranja

Prva grupa testova (statično testiranje) napravljena je nad fiksnim skupom podataka (500 000 prethodno generiranih članaka), dok je tijekom izvršavanja druge grupe testova (dinamično testiranje) simulirano dodavanje novog članaka u nasumično odabranu kategoriju s pauzom od 1 sekunde.

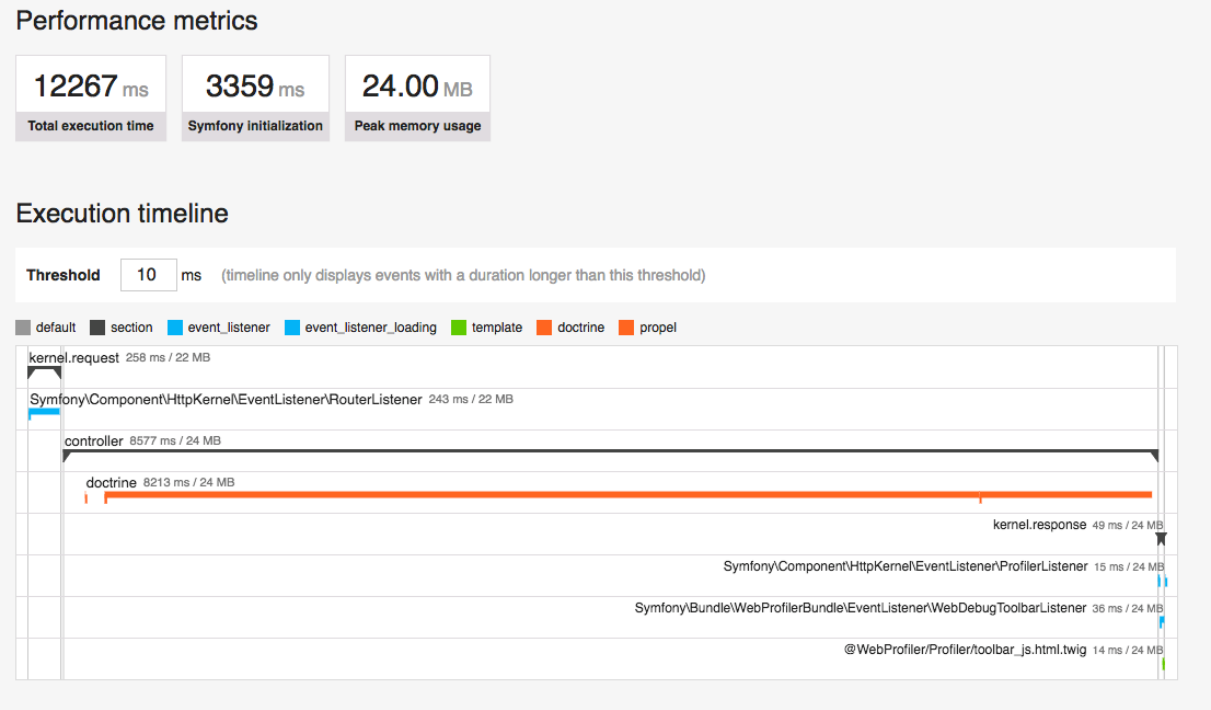
[14] https://docs.docker.com/compose/ - Docker Compose alat za orkestraciju Docker kontejnera
[15] https://www.joedog.org/siege-home/ - Siege alat za testiranje web sustava

6.2. Mjerenje i optimizacija performansi web sustava

U nastavku su prikazani rezultati stres testova i testova opterećenja web sustava prije i nakon optimizacije. Rezultati su prikazani u obliku grafova koji prikazuju odnos i promjenu performansi web sustava nakon optimizacije.

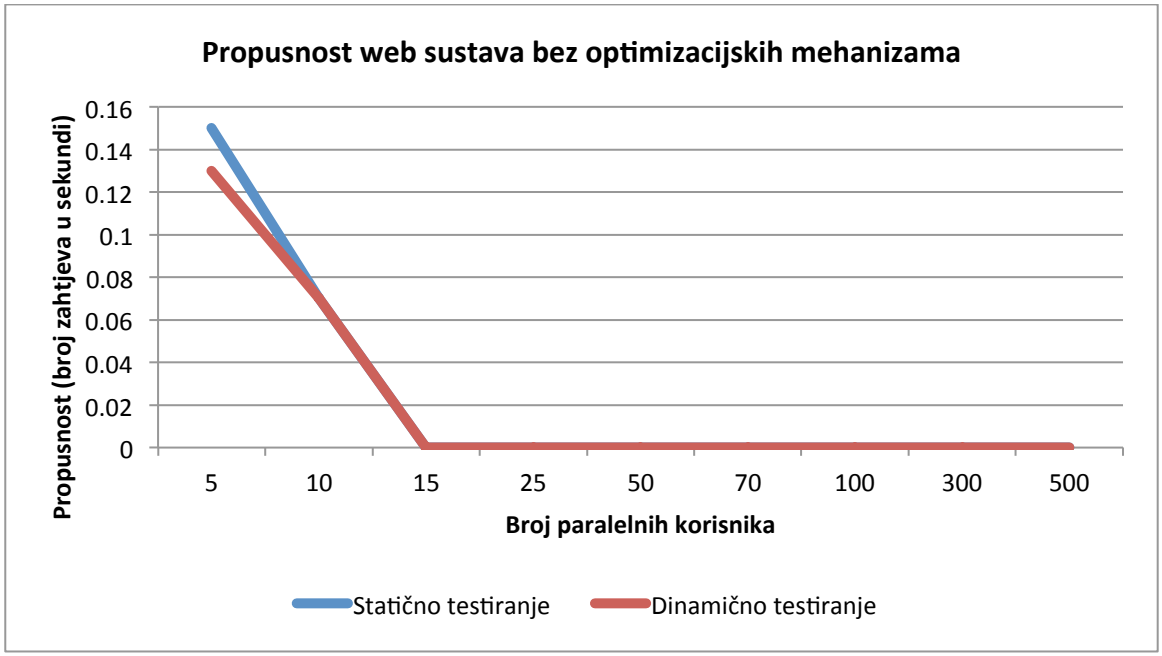
6.2.1. Performanse web sustava bez optimizacijskih mehanizama

Sustav bez optimizacijskih mehanizama obrađuje jedan HTTP zahtjev više od 10 sekundi, a rezultati profiliranja pokazuju kako je većina vremena obrade utrošena na izvršavanje upita prema bazom podataka. Budući da ne postoje indeksi u bazi podataka takvo je ponašanje očekivano.



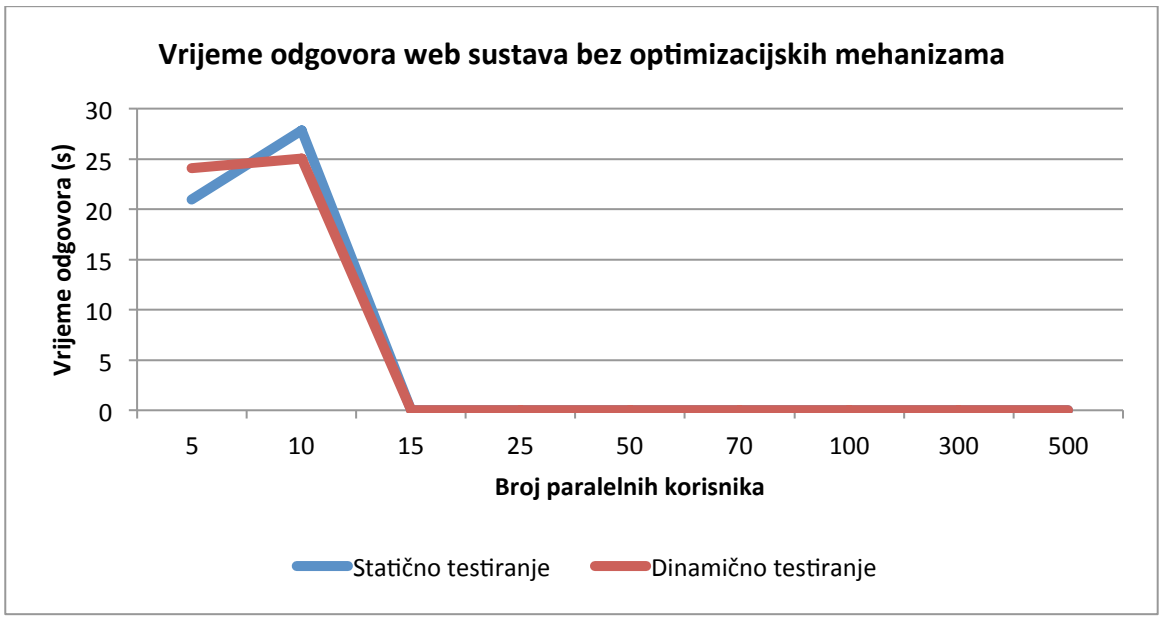
***Slika 10**: Rezultat profiliranja web sustava bez optimizacijskih mehanizama*

Propusnost web sustava bez optimizacijskih mehanizama iznosila je 0,15 i 0,07 zahtjeva u sekundi tijekom statičnog testiranja, 0,13 i 0,07 zahtjeva u sekundi tijekom dinamičnog testiranja, uz 5 i 10 paralelno simuliranih korisnika. Sustav nije mogao obrađivati zahtjeve tijekom testova s 15 ili više paralelno simuliranih korisnika te ne bi mogao izdržati promet većeg broja stvarnih posjetitelja.



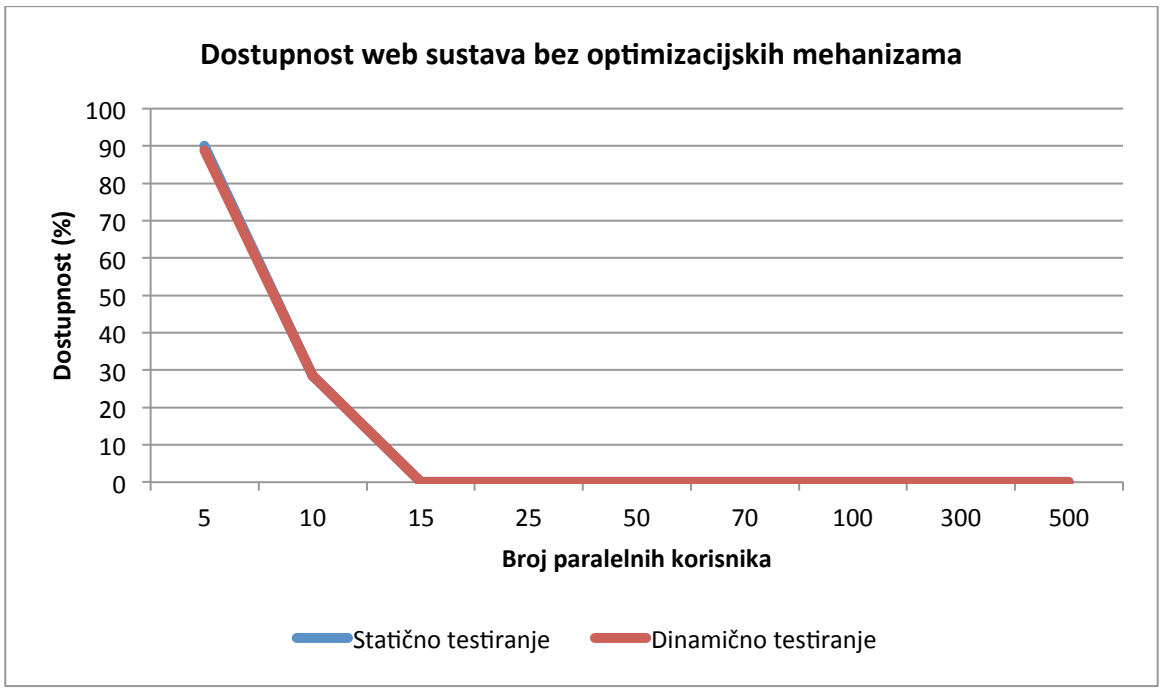
***Graf 4**: Propusnost web sustava bez optimizacijskih mehanizama*

Vrijeme odgovora web sustava bez optimizacijskih mehanizama iznosilo je 20,96 sekundi i 27,83 sekundi tijekom statičnog testiranja odnosno 24,1 sekundu i 25,01 sekundu tijekom dinamičnog testiranja, uz 5 i 10 paralelno simuliranih korisnika, respektivno. U oba je slučaja vrijeme odgovora iznimno veliko i neprihvatljivo za stvarni web sustav.



***Graf 5**: Vrijeme odgovora web sustava bez optimizacijskih mehanizama*

Dostupnost web sustava bez optimizacijskih mehanizama iznosila je 90% i 28,57% tijekom statičnog testiranja odnosno 88,89% i 28,57% tijekom dinamičnog testiranja, uz 5 i 10 paralelno simuliranih korisnika, respektivno. Pojava nedostupnosti pri maloj količini prometa upućuje na strukturne probleme web sustava i postojanje uskih grla.

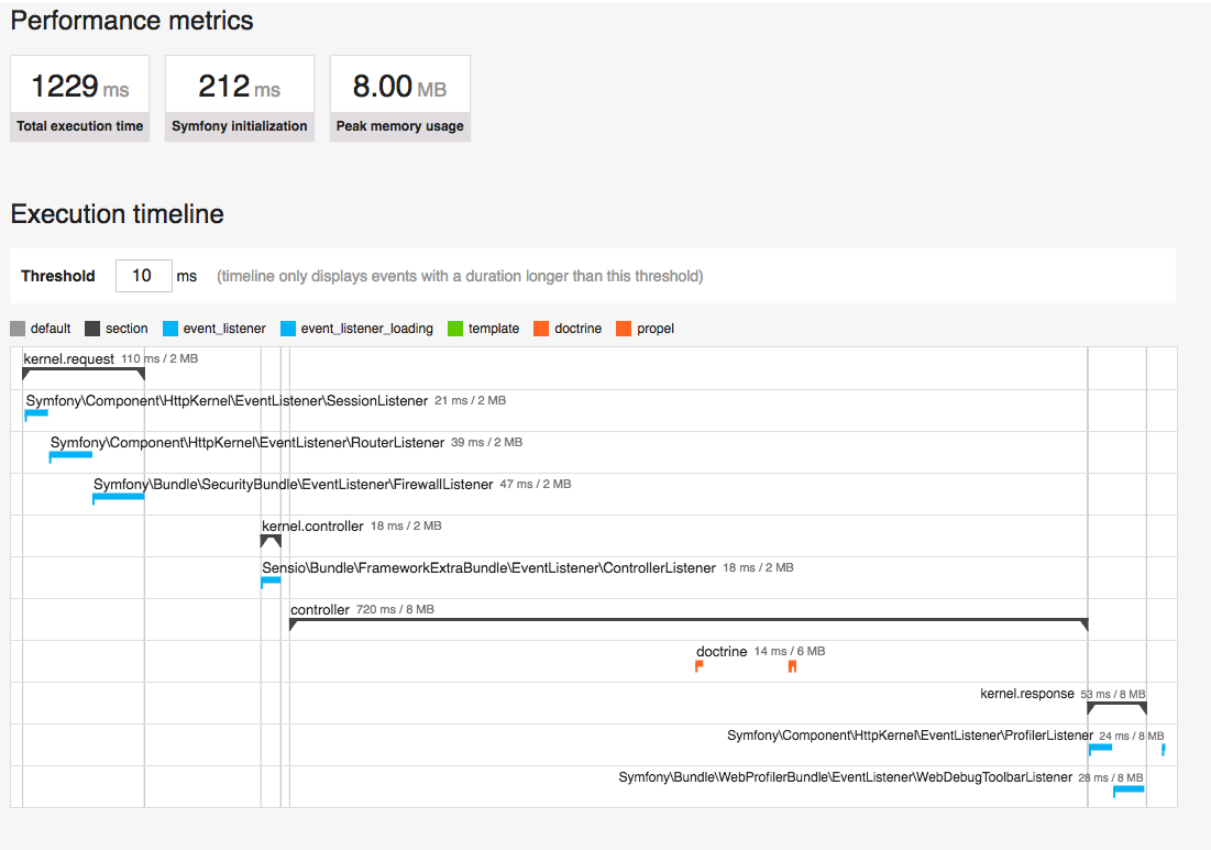


***Graf 6**: Dostupnost web sustava bez optimizacijskih mehanizama*

Web sustav bez optimizacijskih mehanizama pokazao je iznimno loše performanse u radu s velikom količinom sadržaja i manjim brojem posjeta. Simuliranjem prometa 10 paralelnih korisnika sustav je zabilježio probleme u radu, a simulacijom prometa 15 paralelnih korisnika u potpunosti je onemogućen rad sustava.

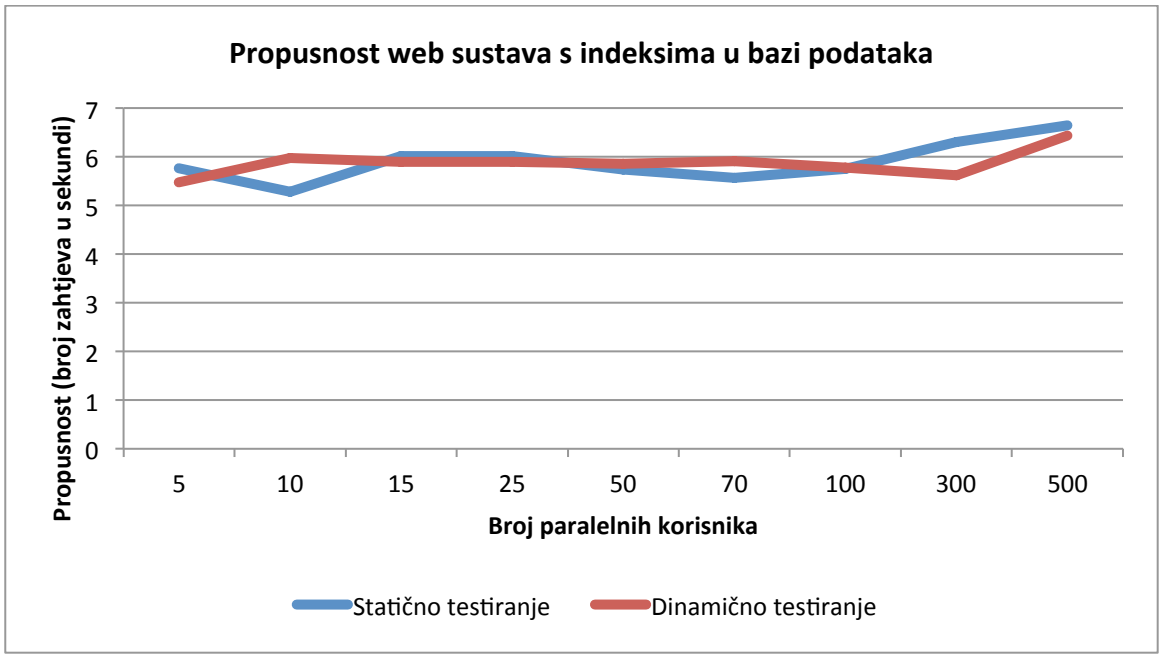
6.2.2. Optimizacija web sustava dodavanjem indeksa u bazu podataka

Dodavanjem indeksa u tablice *posts* i *categories* na polja *posts.created_at* i *category_title*, prema kojima se radi sortiranje i filtriranje tijekom dohvaćanja podataka za prikaz naslovnice portala, vrijeme izvršavanja upita značajno je smanjeno, što je vidljivo u rezultatu profiliranja.



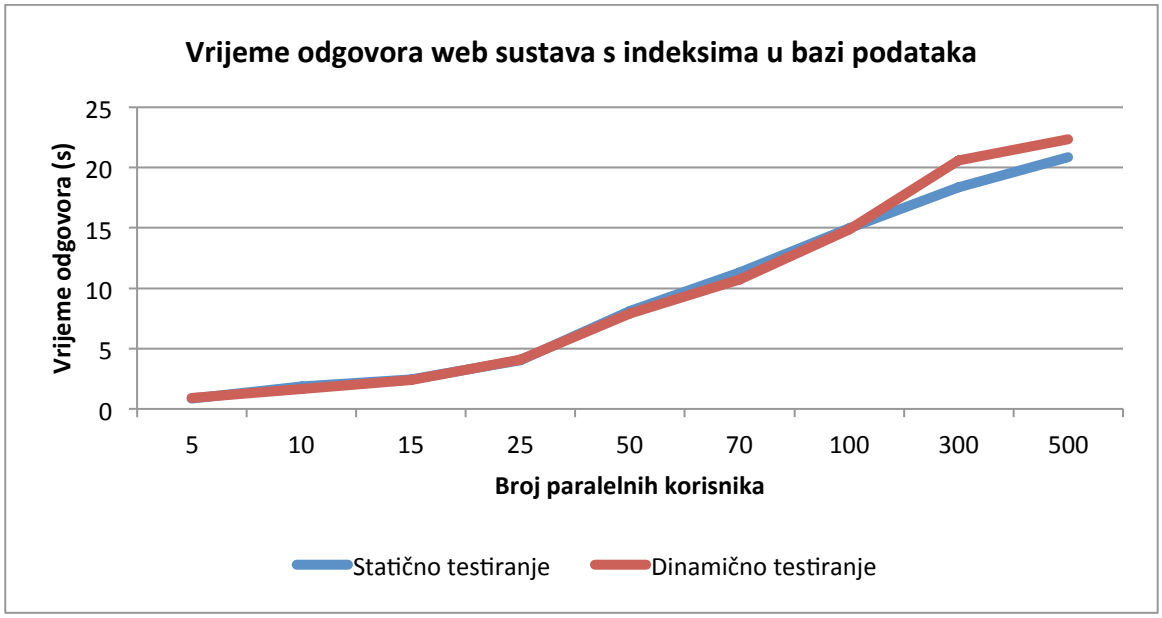
***Slika 11**: Rezultat profiliranja web sustava nakon dodavanja indeksa u bazi podataka*

Propusnost web sustava s indeksima u bazi podataka iznosila je 5,77, 5,28, 6,01, 6,01, 5,74, 5,57, 5,75, 6,3 i 6,64 zahtjeva u sekundi tijekom statičnog testiranja odnosno 5,47, 5,98, 5,9, 5,9, 5,85, 5,91, 5,78, 5,62 i 6,43 zahtjeva u sekundi tijekom dinamičnog testiranja uz simuliranje 5, 10, 15, 25, 50, 70, 100, 300 i 500 paralelnih korisnika, respektivno. Propusnost sustava značajno je poboljšana u odnosu na propusnost sustava bez optimizacijskih mehanizama



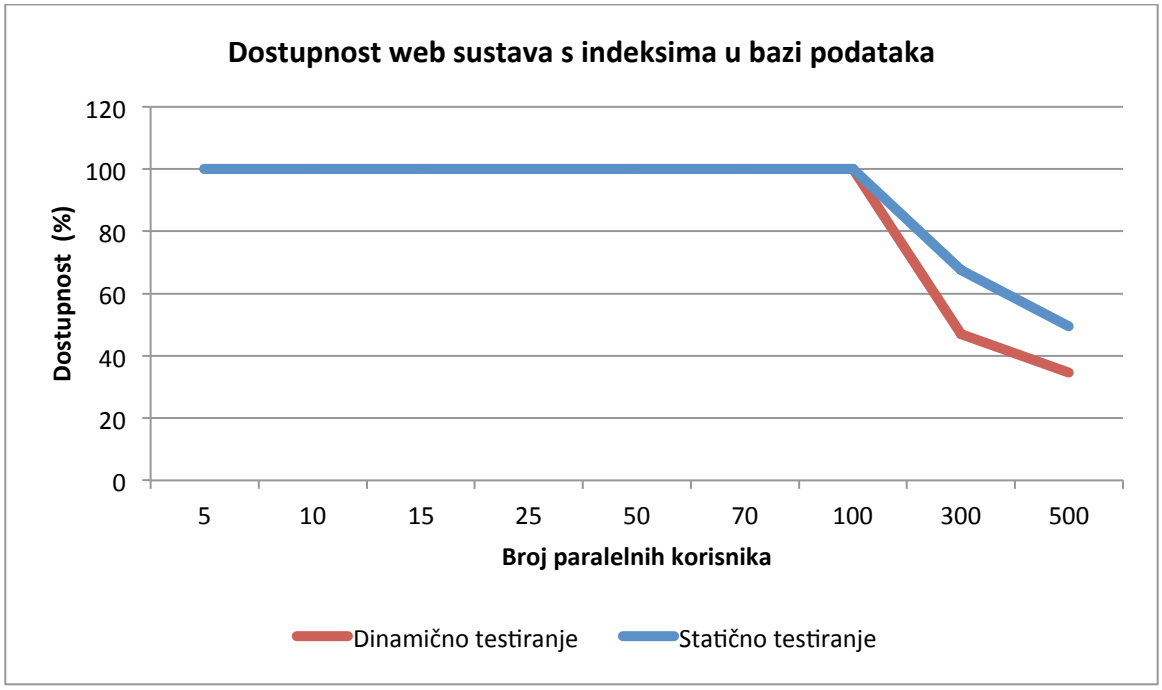
***Graf 7**: Propusnost web sustava s indeksima u bazi podataka*

Prosječno vrijeme odgovora web sustava iznosilo je 0,86, 1,87, 2,45, 4,03, 8,12, 11,3, 14,95, 18,36 i 20,84 sekundi tijekom statičnog testiranja odnosno 0,91, 1,65, 2,38, 4,09, 7,93, 10,71, 14,84, 20,59 i 22,34 sekundi tijekom dinamičnog testiranja uz simuliranje 5, 10, 15, 25, 50, 70, 100, 300 i 500 paralelnih korisnika, respektivno. Iako je prosječno vrijeme odgovora značajno smanjeno, već pri simulaciji 50 paralelnih korisnika sustavu je potrebno više od 5 sekundi za obradu zahtjeva što predstavlja performansni problem u uvjetima visoke posjećenosti.



***Graf 8**: Prosječno vrijeme odgovora web sustava s indeksima u bazi podataka*

Dostupnost web sustava iznosila je 100%, 100%, 100%, 100%, 100%, 100%, 100%, 67,45% i 49,49% tijekom statičnog testiranja odnosno 100%, 100%, 100%, 100%, 100%, 100%, 100%, 46,97% i 34,55% tijekom dinamičnog testiranja uz simuliranje 5, 10, 15, 25, 50, 70, 100, 300 i 500 paralelnih korisnika, respektivno. Dostupnost web sustava značajno je povećana uvođenjem indeksa.

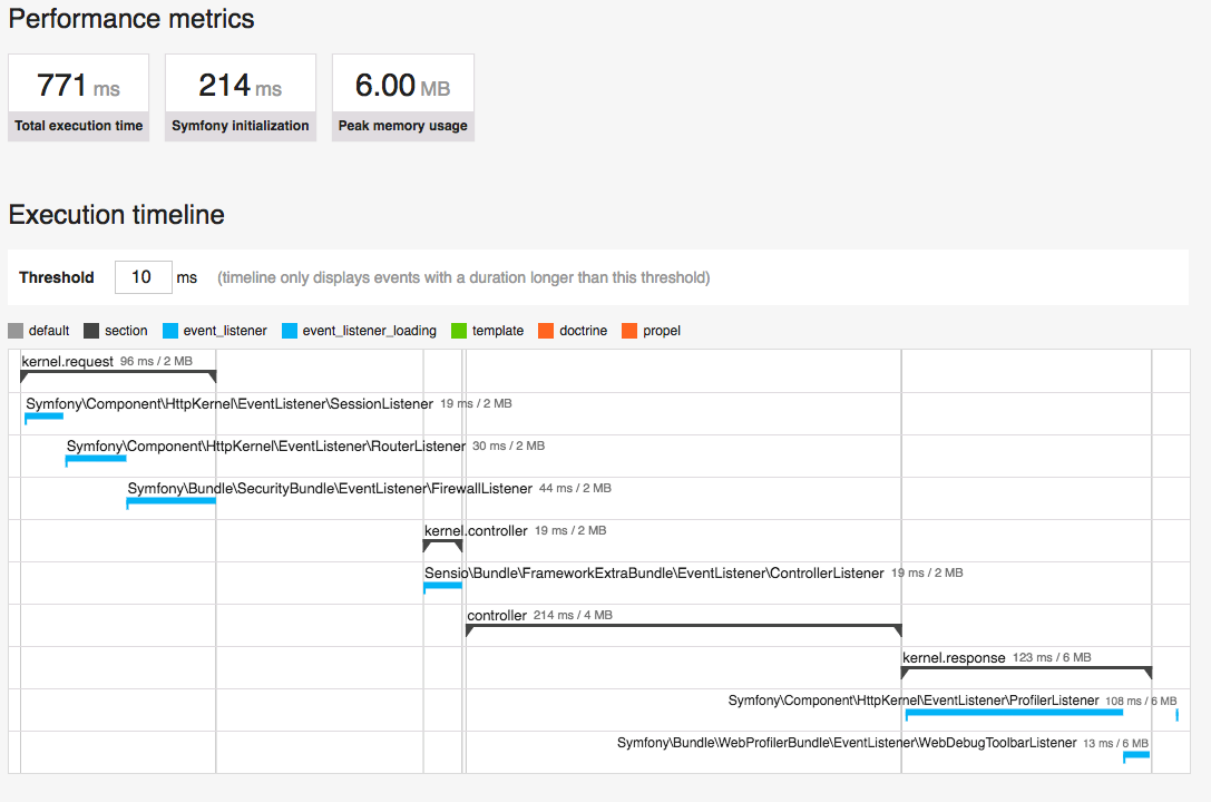


***Graf 9**: Dostupnost web sustava s indeksima u bazi podataka*

Rezultati su pokazali kako dodavanje indeksa u bazu podataka značajno poboljšava sveukupne performanse sustava. Web sustav je, iako vrlo sporo, mogao posluživati i do 100 paralelnih korisnika. Značajne degradacije performansi primijećene su tek prilikom simuliranja 300 i 500 paralelnih korisnika.

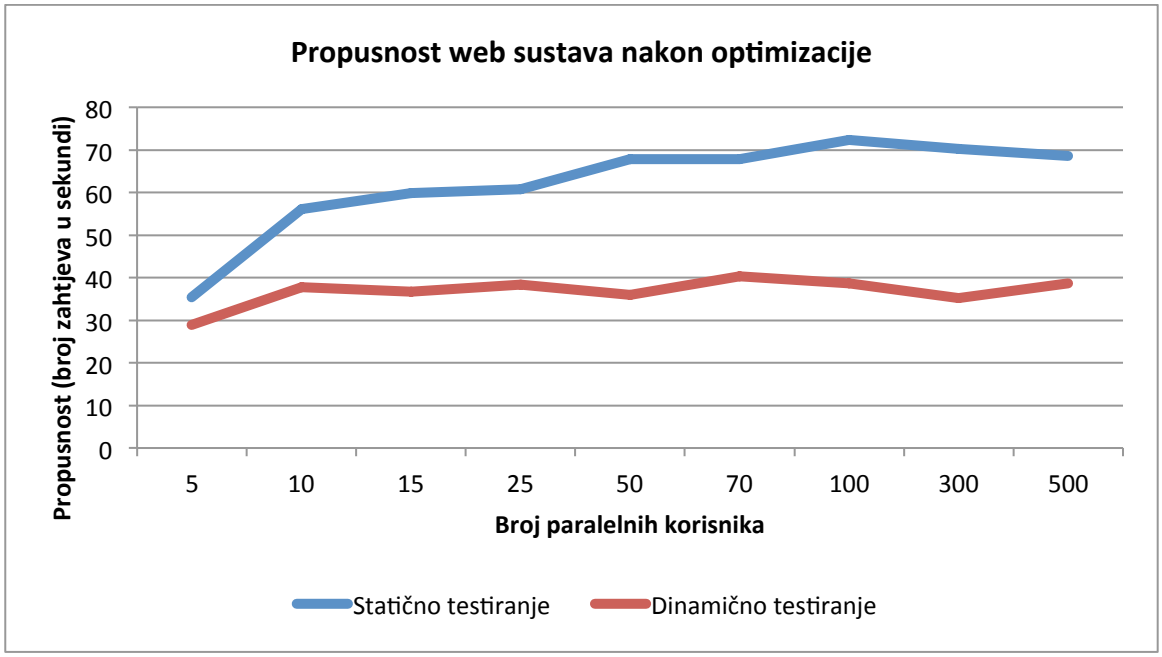
6.2.3. Optimizacija web sustava pohranom podataka u privremenu memoriju

Uvođenjem privremene pohrane podataka (sadržaja modula naslovnice) pomoću Redis poslužitelja za privremenu pohranu podataka na strani aplikacije performanse sustava značajno su optimizirane. Profiliranje web sustava pokazalo je kako je vrijeme obrade jednog HTTP zahtjeva manje od 1 sekunde.



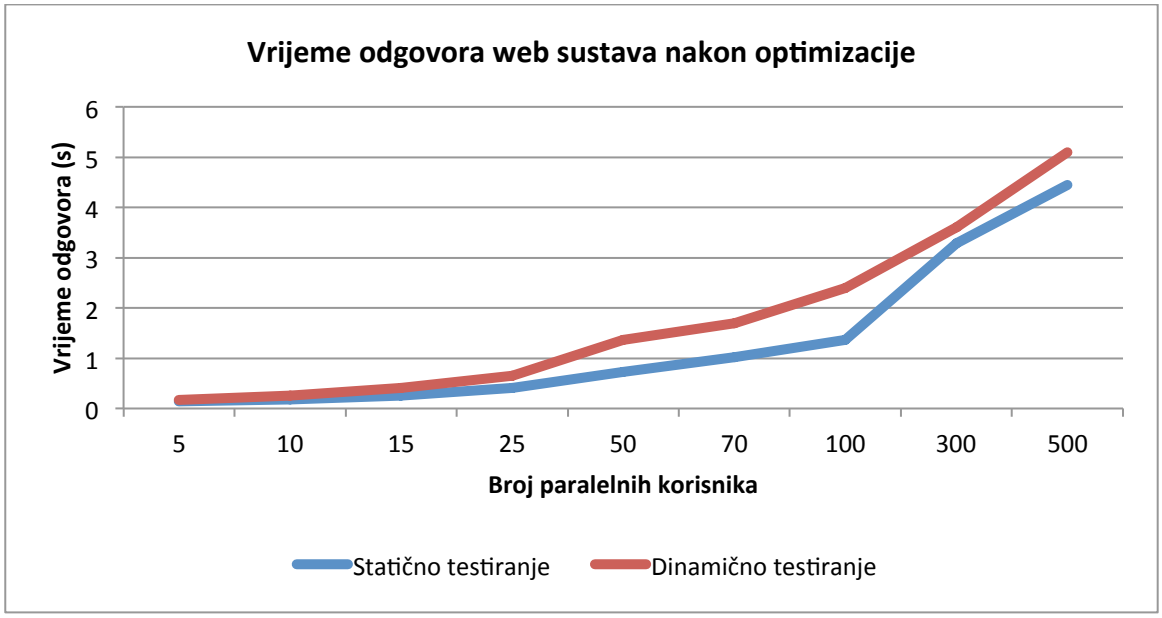
***Slika 12**: Rezultat profiliranja web sustava nakon optimizacije*

Propusnost web sustava nakon optimizacije iznosila je 35,39, 56,15, 59,9, 60,83, 67,85, 67,88, 72,31, 70,21 i 68,58 zahtjeva u sekundi tijekom statičnog testiranja odnosno 28,9, 37,76, 36,74, 38,3, 36, 40,36, 38,72, 35,21 i 38,66 zahtjeva u sekundi tijekom dinamičnog testiranje, uz simuliranje 5, 10, 15, 25, 50, 70, 100, 300 i 500 paralelnih korisnika, respektivno. Razlika u performansama web sustava tijekom statičnog testiranja u odnosu na dinamično testiranje nastaje zbog paralelnog rada programske skripte koja u pozadini generira nove članke i uzrokuje osvježavanje sadržaja privremene memorije.



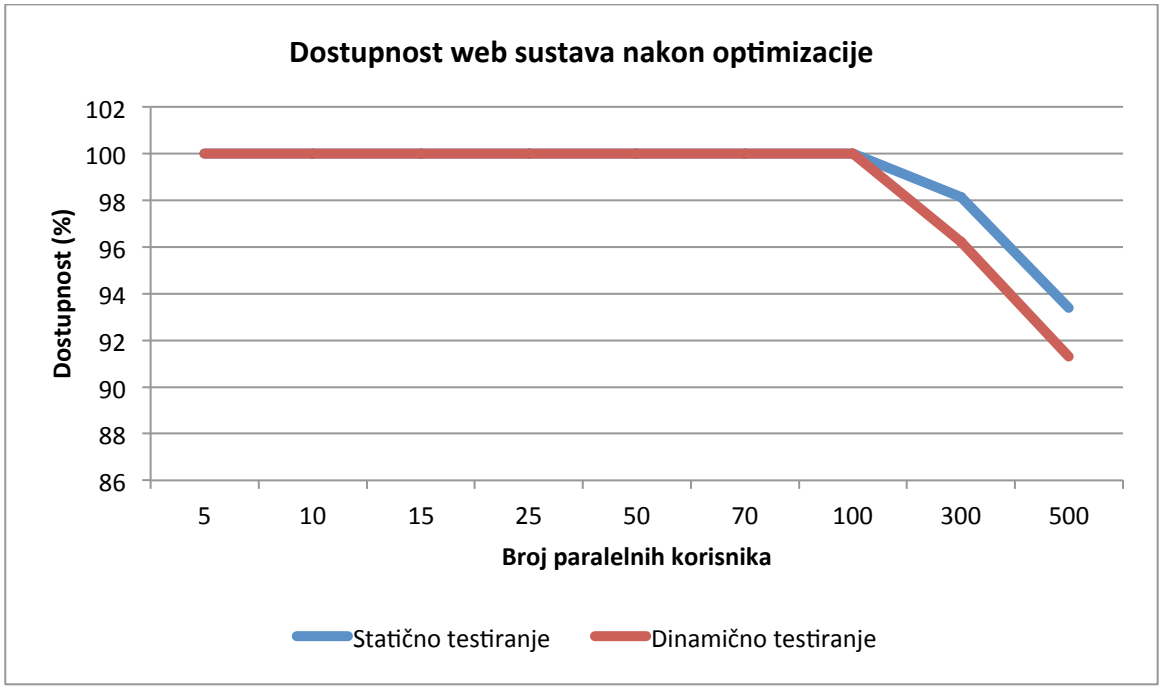
***Graf 10**: Propusnost web sustava nakon optimizacije*

Prosječno vrijeme odgovora web sustava nakon optimizacije iznosilo je 0,14, 0,18, 0,25, 0,41, 0,73, 1,02, 1,36, 3,29 i 4,45 sekundi tijekom statičnog testiranja odnosno 0,17, 0,26, 0,41, 0,65, 1,37, 1,7, 2,4, 3,6 i 5,1 sekundi tijekom dinamičnog testiranja, uz simuliranje 5, 10, 15, 25, 50, 70, 100, 300 i 500 paralelnih korisnika, respektivno. Tek pri simulaciji 50 paralelnih korisnika sustavu je bilo potrebno više od 1 sekunde za obradu zahtjeva.



***Graf 11**: Prosječno vrijeme odgovora web sustava nakon optimizacije*

Dostupnost web sustava iznosila je 100%, 100%, 100%, 100%, 100%, 100%, 100%, 98,14% i 93,38% tijekom statičnog testiranja odnosno 100%, 100%, 100%, 100%, 100%, 100%, 100%, 96,22% i 91,3% tijekom dinamičnog testiranja uz simuliranje 5, 10, 15, 25, 50, 70, 100, 300 i 500 paralelnih korisnika, respektivno. Web sustav je zadržao visoku dostupnost čak i u uvjetima velike količine prometa.

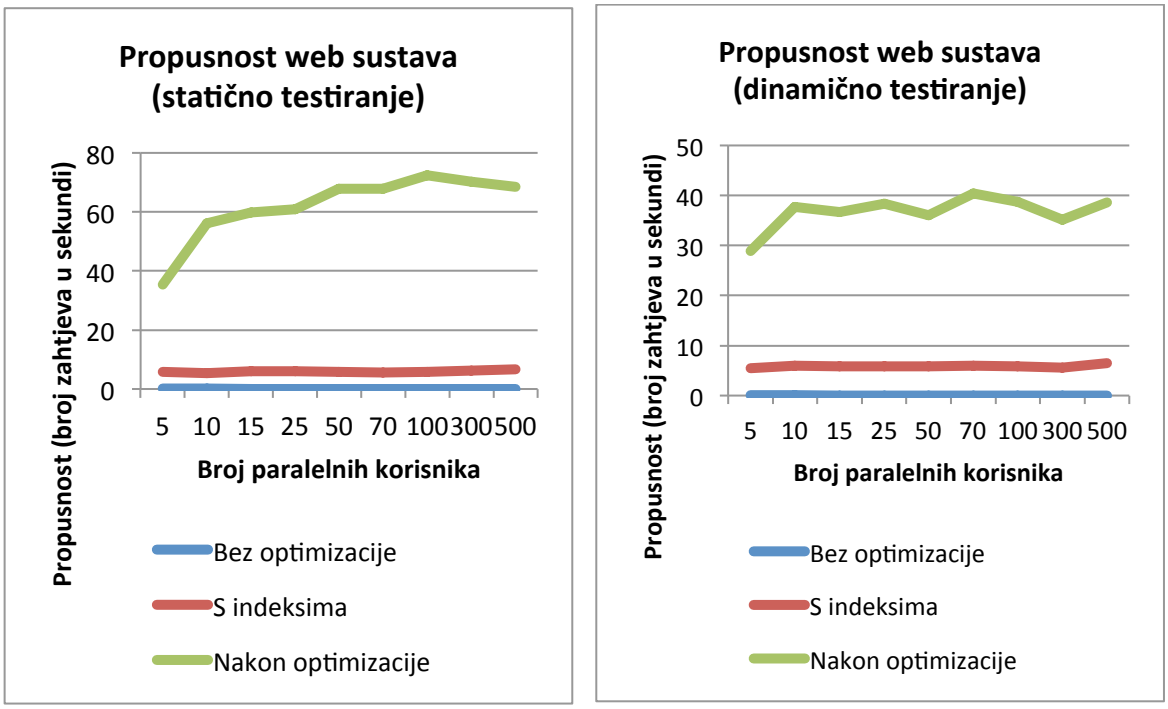


***Graf 12**: Dostupnost web sustava nakon optimizacije*

Rezultati testiranja web sustava nakon optimizacije pokazuju najveće razlike između statičnog i dinamičnog testiranja upravo zbog pohrane sadržaja modela naslovnice u privremenu memoriju i njenog osvježavanja uslijed rada programske skripte za generiranje članaka.

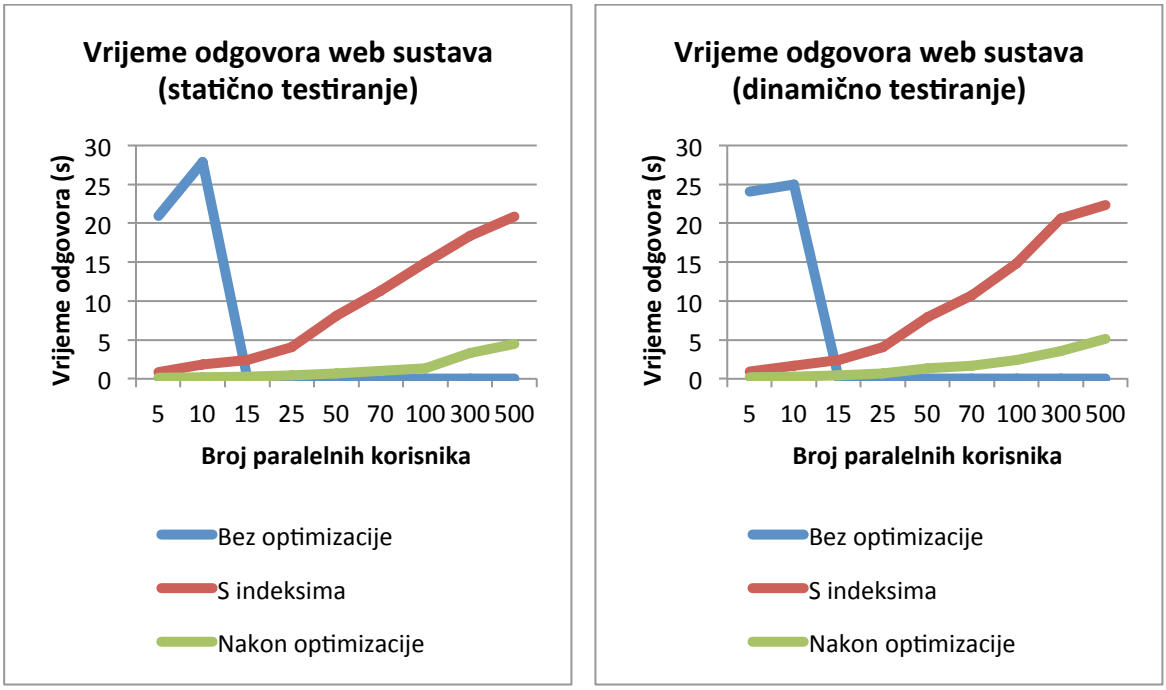
6.2.4. Sumarni prikaz i interpretacija rezultata

Najbolje performanse web sustav je pokazao nakon optimizacije što je u skladu s očekivanjima, iako su već uvođenjem indeksa u bazu podataka performanse web sustava značajno poboljšane. Ozbiljna degradacija performansi web sustava nastala je tek pri simulaciji prometa 500 paralelnih korisnika, te je za optimizaciju web sustava s takvom količinom prometa potrebno primijeniti naprednije mehanizme optimizacije.



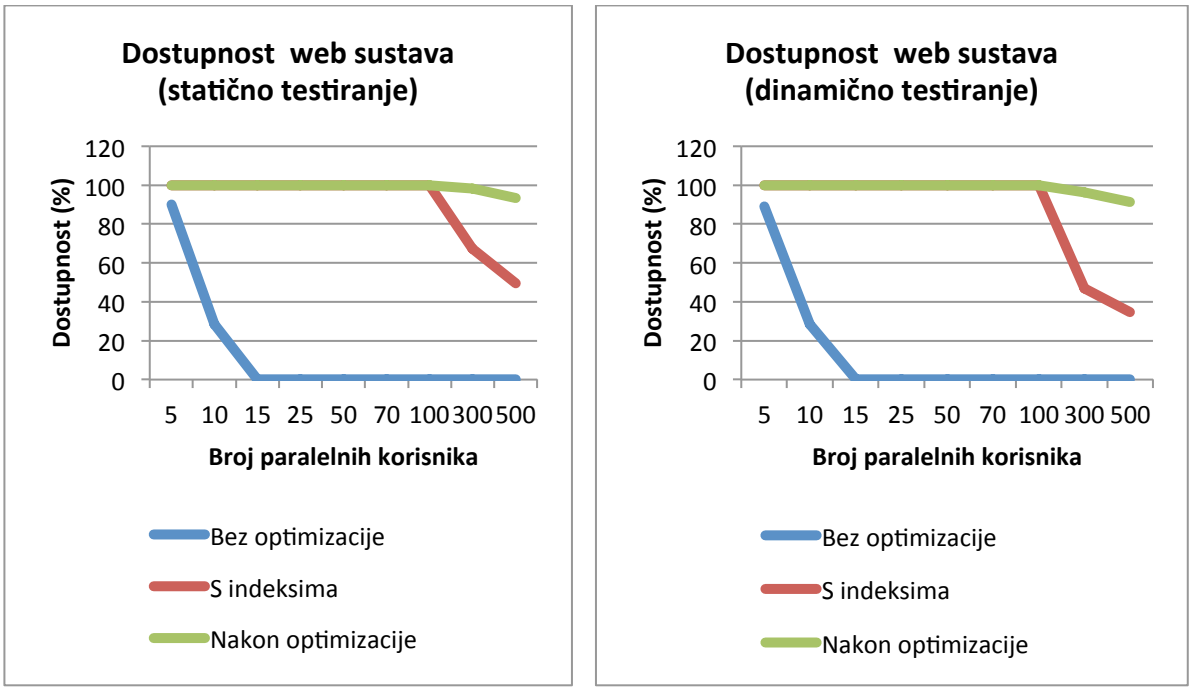
***Graf 13**: Usporedba propusnosti web sustava*

Vrijeme odgovora web sustava povećavalo se proporcionalno s povećanjem broja paralelno simuliranih korisnika. Web sustav nakon uvođenja indeksa može obrađivati veliku količinu prometa, ali s velikim vremenom odgovora koje se pri simulaciji 70 paralelnih korisnika podiže iznad 10 sekundi. Optimizirani web sustav obrađivao je promet 500 simuliranih korisnika s prosječnim vremenom odgovora od približno 5 sekundi.



***Graf 14**: Usporedba vremena odgovora web sustava*

Web sustav bez optimizacijskih mehanizama postao je nedostupan već pri simulaciji 15 paralelnih korisnika, dok je web sustav nakon uvođenja indeksa i optimizacije postao nedostupan tek prilikom simulacije 300 i 500 paralelnih korisnika. Optimizirani web sustav je čak i u takvim uvjetima velikog prometa bilježio dostupnost veću od 90%.



***Graf 15**: Usporedba dostupnosti web sustava*

7. Zaključak

Web je tijekom godina postao najpopularniji medij za distribuciju informacija, komunikaciju i poslovanje, ali i digitalni društveni život. Razvoj mobilnih tehnologija i sve šira dostupnost brzog Internet signala omogućila je korištenje weba bez obzira na lokaciju korisnika. Korisnici očekuju brze, gotovo trenutačne odgovore na interakcije s web sustavom stoga su performanse web sustava ključan element korisničkog iskustva. S porastom broja korisnika i povećanjem sadržaja u bazi performanse web sustava postupno opadaju; vrijeme odgovora izlazi iz okvira prihvatljivosti, a neočekivano visoki promet uzrokuje djelomičnu nedostupnost sustava. Web sustavi moraju biti pravilno optimizirani kako bi zadržali visoke performanse u svim uvjetima. Performanse su također iznimno važne za komercijalni aspekt web sustava, posebice kada se strategija monetizacije temelji na posluživanju sadržaja (npr. plaćeni sadržaj ili oglašavanje). Web sustavi visokih performansi moći će poslužiti sadržaj većem broju korisnika u jedinici vremena, s manjim ulaganjima u infrastrukturu i manjom potrošnjom energije i pratećim troškovima održavanja.

Optimizacija je web sustava, kao i optimizacija drugih računalnih sustava, kontinuirani proces koji se temelji na mjerenju performansi i profiliranju sustava, identifikaciji uskih grla, uvođenju optimizacijskih mehanizama te ponovnim mjerenjima radi potvrde rezultata optimizacije. Mjerenje performansi web sustava potrebno je napraviti u izoliranom okruženju te osigurati identične uvjete (npr. količina sadržaja u bazi podataka, opterećenje poslužitelja) tijekom ponovljenih mjerenja, a rezultate mjerenja treba promatrati relativno i koristiti ih kao temelj za usporedbu.

PHP programski jezik omogućava razvoj web sustava visokih performansi (što potvrđuju primjeri poput sustava Yahoo i Vkontakte), a usko grlo najčešće je pristup bazi podataka (tj. izvršavanje upita prema bazi podataka). Najčešći su i najjednostavniji mehanizmi optimizacije web sustava uvođenje indeksa u bazu podataka, pohrana sadržaja u privremenu memoriju i optimizacija programskog koda. Praktični dio ovog rada pokazuje kako se već uvođenjem indeksa u bazu podataka postižu bolje performanse, dok se pohranom sadržaja u privremenu memoriju performanse značajno povećavaju.

8. Literatura

1. Belshe, M., Thomson, M. and Peon, R. (2014) *Hypertext Transfer Protocol Version 2 (HTTP/2)*. Preuzeto 20. kolovoza 2017. s: https://tools.ietf.org/html/rfc7540

2. Berners-Lee, T. (1989) *Information Management: A Proposal*. Preuzeto 20. kolovoza 2017. s: https://www.w3.org/History/1989/proposal.html

3. Berners-Lee, T. *et al.* (1997) *RFC2068 - Hypertext Transfer Protocol -- HTTP/1.1*. Preuzeto 20. kolovoza 2017. s: https://tools.ietf.org/html/rfc2068

4. Berners-Lee, T. *et al.* (2004) *Architecture of the World Wide Web, Volume One*, *W3C Recommendation 15 December 2004*. Preuzeto 20. kolovoza 2017. s: https://www.w3.org/TR/webarch/

5. Berners-Lee, T. (*s.a.*) *Answers for young people - Tim Berners-Lee*. Preuzeto 20. kolovoza 2017. s: https://www.w3.org/People/Berners-Lee/Kids.html

6. Berners-Lee, T. (*s.a.*) *Frequently asked questions by the Press - Tim BL*. Preuzeto 20. kolovoza 2017. s: https://www.w3.org/People/Berners-Lee/FAQ.html

7. CareerProfiles (2017) *Explore the Top 100 Careers for 2017*. Preuzeto 20. kolovoza 2017. s: http://www.careerprofiles.info/top-100-careers.html

8. Christensen, E. *et al.* (2001) *Web Service Definition Language (WSDL)*. Preuzeto 20. kolovoza 2017. s: https://www.w3.org/TR/wsdl

9. coderseye.com (2017) *11 Best PHP Frameworks for Modern Web Developers in 2017*. Preuzeto 20. kolovoza 2017. s: https://coderseye.com/best-php-frameworks-for-web-developers/

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

11. Composer (2017) *Introduction - Composer*. Preuzeto 20. kolovoza 2017. s: https://getcomposer.org/doc/00-intro.md

12. Eevee (2012) *PHP: a fractal of bad design / fuzzy notepad*. Preuzeto 20. kolovoza 2017. s: https://eev.ee/blog/2012/04/09/php-a-fractal-of-bad-design/

13. Elfi, D. (2013) *Caching Strategies for Improved Web Performance - O'Reilly Radar*, in *OSCON 2013 Speaker Series*. Preuzeto 20. kolovoza 2017. s: http://radar.oreilly.com/2013/07/caching-strategies-for-improved-web-performance.html

14. Flanagan, D. (2011) *JavaScript: The Definitive Guide, 6th Edition - O'Reilly Media*. 6th edn. O'Reilly Media. Preuzeto 20. kolovoza 2017. s: http://shop.oreilly.com/product/9780596805531.do

15. Fowler, M. (2009) *TwoHardThings*. Preuzeto 20. kolovoza 2017. s: https://martinfowler.com/bliki/TwoHardThings.html

16. *Github Language Stats* (2017). Preuzeto 20. kolovoza 2017. s: https://madnight.github.io/githut/

17. Internet World Stats (2017) *World Internet Users Statistics and 2017 World Population Stats*. Preuzeto 20. kolovoza 2017. s: http://www.internetworldstats.com/stats.htm

18. Investopedia (2017) *Dotcom Bubble*. Preuzeto 20. kolovoza 2017. s: http://www.investopedia.com/terms/d/dotcom-bubble.asp.

19. Isaac, L. P. (2014) *15 Tips to Optimize Your PHP Script for Better Performance for Developers*. Preuzeto 20. kolovoza 2017. s: http://www.thegeekstuff.com/2014/04/optimize-php-code/

20. IT Conversations (2003) *Rasmus Lerdorf, Senior Technical Yahoo: PHP, Behind the Mic*. Preuzeto 20. kolovoza 2017. s: https://web.archive.org/web/20130728125152/http://itc.conversationsnetwork.org/shows/detail58.html

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

22. Letts, S. (2015) *What is Web 4.0?* Preuzeto 20. kolovoza 2017. s: https://stephenletts.wordpress.com/web-4-0/

23. MacKay, J. (2016) *Building Facebook today: Is PHP still relevant in 2017? - Crew.co*. Preuzeto 20. kolovoza 2017. s: https://crew.co/blog/is-php-still-relevant-in-2017/

24. MDN (2017) *Cache-Control - HTTP | MDN*. Preuzeto 20. kolovoza 2017. s: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control

25. 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.

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

27. O'Reilly, T. (2005) *What Is Web 2.0*. Preuzeto 20. kolovoza 2017. s: http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html

28. 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. Preuzeto 20.

kolovoza 2017. s: http://www.ijarcsse.com/docs/papers/Volume_3/10_October2013/V3I10-0149.pdf

29. PHP-FIG (2016) *Frequently Asked Questions - PHP-FIG*. Preuzeto 20. kolovoza 2017. s: http://www.php-fig.org/faqs/

30. PHP-FIG (2017) *PHP Standards Recommendations - PHP-FIG*. Preuzeto 20. kolovoza 2017. s: http://www.php-fig.org/psr/

31. 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.

32. PSR (2017) *PSR-6: Caching Interface - PHP-FIG*. Preuzeto 20. kolovoza 2017. s: http://www.php-fig.org/psr/psr-6/

33. Redis (no date) *Using Redis as an LRU cache – Redis*. Preuzeto 20. kolovoza 2017. s: https://redis.io/topics/lru-cache

34. Richardson, C. (2017) *Microservice Architecture pattern*. Preuzeto 20. kolovoza 2017. s: http://microservices.io/patterns/microservices.html

35. Shellhammer, A. (2017) *The need for mobile speed: How mobile latency impacts publisher revenue*. Preuzeto 20. kolovoza 2017. s: https://www.doubleclickbygoogle.com/articles/mobile-speed-matters/

36. SimilarTech (2017) *PHP Market Share and Web Usage Statistics*. Preuzeto 20. kolovoza 2017. s: https://www.similartech.com/technologies/php

37. Souders, S. (2007) *14 Rules for Faster-Loading Web Sites*. Preuzeto 20. kolovoza 2017. s: http://stevesouders.com/hpws/

38. Stackify (2017) *Trendiest and Most Popular Programming Languages in 2017*. Preuzeto 20. kolovoza 2017. s: https://stackify.com/trendiest-programming-languages-hottest-sought-programming-languages-2017/

39. Stogov, D. and Suraski, Z. (2014) *PHP RFC: Move the phpng branch into master*. Preuzeto 20. kolovoza 2017. s: https://wiki.php.net/rfc/phpng

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

41. The PHP Group (2017a) *History of PHP and Related Projects*. Preuzeto 20. kolovoza 2017. s: http://php.net/manual/en/history.php.php

42. The PHP Group (2017b) *PHP: Supported Versions*. Preuzeto 20. kolovoza 2017. s: http://php.net/supported-versions.php

43. The PHP Group (2017c) *PHP: Unsupported Branches*. Preuzeto 20. kolovoza 2017. s: http://php.net/eol.php

44. TIOBE (2017) *TIOBE Index | TIOBE - The Software Quality Company*. Preuzeto 20. kolovoza 2017. s: https://www.tiobe.com/tiobe-index/

45. Trachtenberg, A. (2004) *Upgrading to PHP 5*. O'Reilly Media. Preuzeto 20. kolovoza 2017. s: http://shop.oreilly.com/product/9780596006365.do

46. VPN Mentor (2017) *Internet Trends 2017. Stats & Facts in the U.S. and Worldwide*. Preuzeto 20. kolovoza 2017. s: https://www.vpnmentor.com/blog/vital-internet-trends/

47. W3Techs (2017a) *Historical trends in the usage of PHP version 5, August 2017*. Preuzeto 20. kolovoza 2017. s: https://w3techs.com/technologies/history_details/pl-php/5

48. W3Techs (2017b) *Usage Statistics and Market Share of PHP for Websites, August 2017*. Preuzeto 20. kolovoza 2017. s: https://w3techs.com/technologies/details/pl-php/all/all

49. W3Techs (2017c) *Usage Statistics and Market Share of Server-side Programming Languages for Websites, August 2017*. Preuzeto 20. kolovoza 2017. s: https://w3techs.com/technologies/overview/programming_language/all

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

51. webfoundation.org (no date) *History of the Web – World Wide Web Foundation*, *Web Foundation*. Preuzeto 20. kolovoza 2017. s: https://webfoundation.org/about/vision/history-of-the-web/

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

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

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

9. Izvorni kod praktičnog primjera

Programski kod praktičnog primjera i resursi potrebni za pokretanje testova nalaze se na javnom Git repozitoriju dostupnom na poveznici https://github.com/zantolov/high-performance-web-sytems-in-php