Prednosti i nedostaci uporabe Node.js platforme

Raguž, Roberta

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I TELEKOMUNIKACIJA

Stručni studij

PREDNOSTI I NEDOSTACI UPORABE Node.js PLATFORME

Završni rad

Roberta Raguž

Osijek, 2017.

ELEKTROTEHNIČKI FAKULTET OSIJEK
Sveučilište Josipa Jurja Strossmayera u Osijeku

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za obranu završnog rada na stručnom studiju

Osijek,

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu završnog rada na stručnom studiju

Ime i prezime studenta:	
Studij, smjer:	
Mat. br. studenta, godina upisa:	
Mentor:	
Sumentor:	
Predsjednik Povjerenstva:	
Član Povjerenstva:	
Naslov završnog rada:	
Primarna znanstvena grana rada:	
Sekundarna znanstvena grana (ili polje) rada:	
Zadatak završnog rada	
Prijedlog ocjene pismenog dijela ispita (završnog rada):	
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: Postignuti rezultati u odnosu na složenost zadatka: Jasnoća pismenog izražavanja: Razina samostalnosti:

Potpis sumentora:	Potpis mentora:

Dostaviti:

1. Studentska služba

U Osijeku, godine Potpis predsjednika Odbora:

Sveučilište Josipa Jurja Strossmayera u Osijeku

ELEKTROTEHNIČKI FAKULTET OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek,

Ime i prezime studenta:	
Studij :	
Mat. br. studenta, godina upisa:	

Ovom izjavom izjavljujem da je rad pod nazivom**:**

izrađen pod vodstvom mentora

i sumentora

mojvlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora inobrazaccija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada,osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, ašto je izričito navedeno u radu.

Potpis studenta:

Popis tablica

Sadržaj

1. UVOD

JavaScript je programski jezik nastao 1995. godine. Zajedno s HyperText Markup Language-om (HTML) i Cascading Style Sheet-om (CSS) čini osnovu *World Wide Web*-a (WWW). Platforma koja služi za izradu brzih *real-time* aplikacija i koja je napravljena pomoću JavaScript programskog jezika naziva se Node.js. Node.js implementiran je 2009. godine s ciljem da omogući korisniku skriptiranje web stranice. Daljnjim razvojem platforme, glavna funkcija s korisničkog skriptiranja prešla je na skriptiranje od strane poslužitelja. S time se postiglo brže učitavanje stranica jer se sadržaj učitava prvo na poslužitelju. Poslužitelj preuzima informacijsko opterećenje te zbog toga korisničko računalo brže učitava web sadržaj. Dakle, umjesto da svako računalo zasebno učitava sadržaj, ona ih preuzimaju s poslužitelja koji odrađuje taj posao. Programerska industrija se sve više interesira za Node.js te sa starijih alata prelaze na njega. Programeri teže *user friendly* (hrv. prijateljskim) alatima. Node.js se može nazvati *user friendly* alatom jer posjeduje sve karakteristike za to. Vrlo je bitno koristiti alate koji pojednostavljuju sam tijek i proces programiranja. *User friendly* alati uvelike pomažu i štede korisniku vrijeme te se on može posvetiti samoj logici programa. Zahvaljujući automatizaciji programskog raspisivanja, programiranje se više ne svodi na puko kucanje koda, već na čistu logiku i razmišljanje. Ovaj rad će objasniti što je Node.js platforma, gdje se primjenjuje i koje su prednosti, odnosno nedostaci platfome. Osim toga, u narednim poglavljima bit će prikazana i usporedba s programskim jezikom Hypertext Preprocessor (PHP) te nekim pomoćnim alatima.

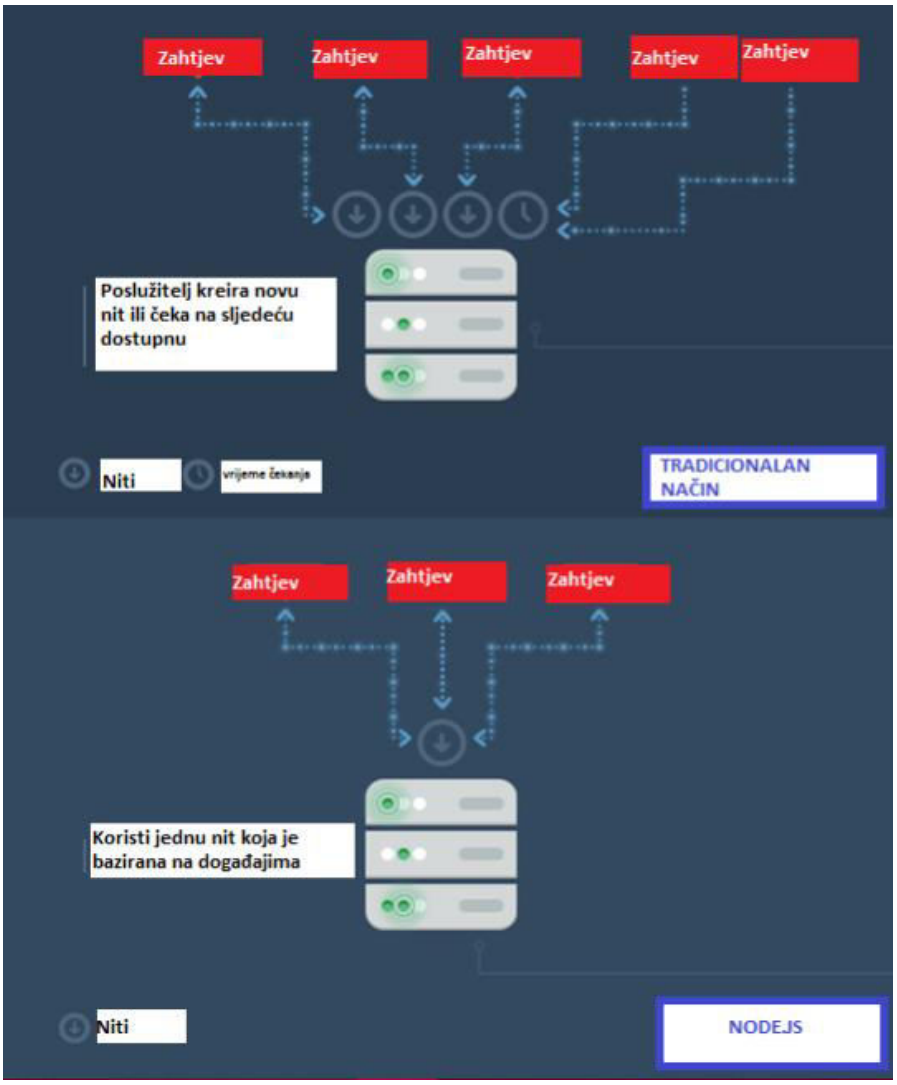
1.1. Povijest Node-js

Node.js razvio je čovjek pod imenom Ryan Dahl zajedno sa svojim kolegama u 2009. godini. Dvije godine kasnije postao je *open-source* projekt. Karakteristično za *open-source*, tj. projekte otvorenog izvora je što se oni mogu uređivati i poboljšavati od strane svih korisnika. To ga čini besplatnim alatom koji je svima dostupan. S obzirom da se ovaj princip rada temelji na dobrovoljnom usavršavanju, produktivnost i kreativnost korisnika je neograničena te se takvi proizvodi neprestano poboljšavaju. Budući da ih poboljšava sam korisnik, a ne firma, cijeli proces razvoja je ubrzan. Firma prvo stvori proizvod, testira ga, šalje u distribuciju nakon čega prikuplja povratne informacije korisnika i onda šalje natrag na usavršavanje. S *open-source* principom rada je taj povrat informacije i naknadni popravak već plaćenog projekta izbjegnut jer korisnik u suradnji s drugima direktno usavršava, tj. njegov rad je svojevrsna povratna informacija koja se izravno implementira u proizvod. Kako je Node.js zajednica rasla tako su se sve više pojavljivali konflikti između vođa projekta, ali i same zajednice te oko novih verzija platforme. U 2015. godini nastao je io.js [1] koji je napravljen kao otvorena alternativa za Node.js [2]. Glavna razlika je bila što su autori platforme planirali održati io.js ažuriran sa zadnjim verzijama Google V8 JavaScript *engine*-a.

2. OSNOVE NODE.js - a

Glavni cilj Node.js platforme je pružiti siguran i lak način za izgradnju mrežnih aplikacija visoke performanse u JavaScriptu.JavaScript se tradicionalno izvršava u web pregledniku, odnosno zadaci pisani u tom programskom jeziku se osvježavaju u korisničkim pretraživačima. Stoga se još prije dva desetljeća javila potreba za pokretanjem na poslužiteljskoj razini te se Node.js pokazao kao najpraktičnije rješenje za to.

Node.js je platforma izrađena na JavaScript *engine*-u (hrv. programsko- logičkom motoru) koja služi za izradu aplikacija za internet. Informatičke kompanije koje temelje svoje aplikacije na radu u pomoćnom alatu Node.js-u su: GoDaddy, IBM, LinkedIn, Microsoft, Netflix, Groupon, Walmart, SAP, Cisco Systems, Paypal i slične usluge [3]. Rješenja koja funkcioniraju slično su: Node.js-u Jaxer i Narwhal [4], ali Node.js se razlikuje od njih jer se bazira na događajima, a ne na nitima kao svi drugi. Glavni cilj Node.js platforme je pružiti siguran i lak način za izgradnju aplikacija visoke performanse u JavaScriptu.



Slika 2.1. Node.js način rad [5]

Slika 2.1. prikazuje tradicionalni način izvršavanja niti te način na koji radi Node.js. Korisnik zadaje programu naredbe koje se dolaze do poslužitelja te ih on istovremeno i u realnom vremenu obrađuje unutar iste niti, tj. spaja ih u zajedničko rješenje.

Neke od važnijih prednosti Node.js-a:

a) **Velika brzina** – brzina izvršavanja programskog koda je, zahvaljujući Google Chrome V8JavaScript *engine*-u, iznimno brza.

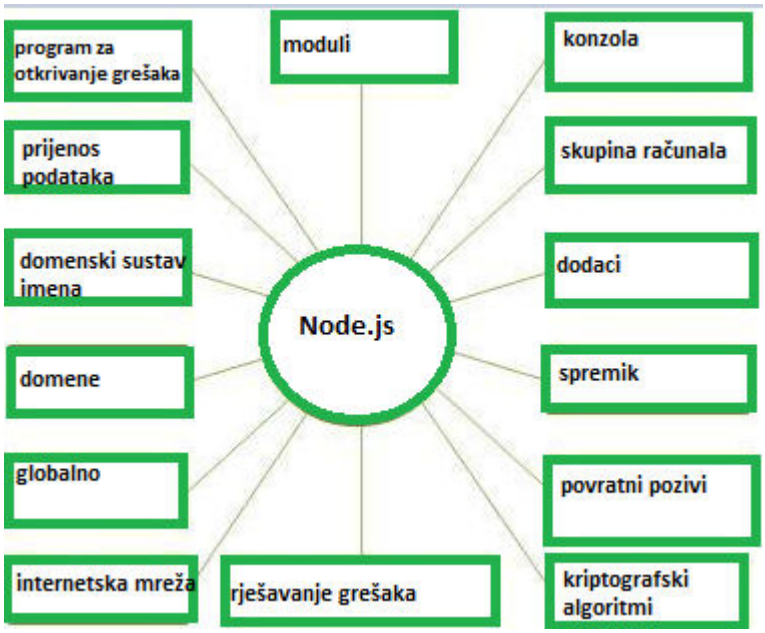
b) **Licenca** –Node.js je izdan pod MIT licencom [6].
Mit licenca je licenca otvorenog (slobodnog) koda koja dozvoljava ponovno

korištenje *software*-a pod tom licencom dokle god sve kopije licenciranog *software*-a sadrže pun tekst i sve uvjete ove licence. Licenca je kompatibilna i može se koristiti u kombinaciji sa GNU općom javnom licencom.

c) **Jedna nit** – cijela platforma bazirana na događajima, a ne na nitima.

d) **„*Cross platform*“** – može se koristiti na svim platformama odnosno izrađen je za Linux, Mac i Windows.

e) **„*Open source*“** –Node.js ima zajednicu koja dizajnira i objavljuje razne dodatke već postojećim mogućnostima Node.js aplikacija.



Slika 2.2. Node.js koncept [7]

Slika 2.2. prikazuje koncept po kojem je zamišljem i realiziran Node.js. Prikazane su bitne sastavnice i karakteristike koje čine Node.js.

2.1. Alternative Node.js-u

U današnje vrijeme Node.js je najpopularniji i najkorišteniji sustav za izvršavanje JavaScripta na poslužitelju. Osim Node.js-a postoje i ostali sustavi, odnosno alternative njemu. Te alternative su također vrlo dobre, a neke od njih su: Vert.x, RingoJS, Nodyn te PurpleJS.

2.1.1. Vert.x

Eclipse Vert.x je alat za izrađivanje web aplikacija na JavaScript platformi. Lako se implementira i podržava različite jezike. To znači da možete sami odabrati vrstu jezika koju želite koristiti, a to uključuje i JavaScript. Kod ove platforme korisnik ima potpunu kontrolu nad izvršavanjem aplikacije na poslužitelju. Najbitnija značajka Vert.x alata je to što je reaktivan. Reaktivan znači da po prirodi ne blokira izvršavanje [8], baš kao i Node.js.

Vert.x	***Node.js***
Bolje performanse	Bolja podrška
Princip više jezika	Razvijan ekosustav
Lak model konkurencije	Lakše implementiranje i ažuriranje
Podržava događaje	Korisniji za manje projekte

Tablica 2.1. Vert.x u odnosu na Node.js

Iz priložene tablice može se zaključiti kako Vert.x alat imabolje performanse, dok je kod Node.js-a bolja korisnička podrška. Što je tiče korištenja programskih jezika, prvi je višejezični, dok je Node.js napravljen za samo jedan, JavaScript, za kojeg ima posebno razvijen softverski ekosustav.

2.1.2. Ringo.js

Ringo.js je već jako dugo na tržištu, te je dugo bio prva i jedina alternativa za Node.js. Ringo.js nije reaktivan kao Vert.x ili Node.js i izrađen je na starom Rhino JavaScript sustavu. On se koristi kao podrška poslužitelju i podržava velike *developerske* zahvate.

Ringo.js	*Node.js*
Velika zajednica	Mlada zajednica
Vrlo dobra Java interoperabilnost	Bez interoperabilnosti
Koristi niti	Korištenje događaja

Tablica 2.2. Ringo.js u odnosu na Node.js

Tablica 2.2. prikazuje usporedbu Ringo.js-a i Node.js-a. Iz tablice je vidljivo da je glavna prednost Ringo.js sustava je to što je on, s obzirom da koristi niti (*threads*) i nije reaktivan, ipak lak za koristištenje i dobar za početak programiranja.

Ringo.js je pogodniji za korištenje unutar velikih programerskih zajednica koje se bave razvojem informatičkog rješenja, dok je Node.js elegantniji za korištenje u manjim zajednicama. Vrlo dobra JavaScript interoperabilnost je karakteristična za Ringo.js. Za razliku od njega, Node.js nije interoperabilan. Interoperabilnost je sposobnost, odnosno mogućnost informacijskog ili informatičkog sustava za pružanje i primanje usluga (dvosmjerno) od drugih i srodnih sustava. Ovo je veliki nedostatak za Node.js, jer se umanjuje šansa za međusobno djelovanje, pa tako i brži rast potaknut uzajamnim razvijanjem i komentiranjem programa.

2.1.3. Nodyn

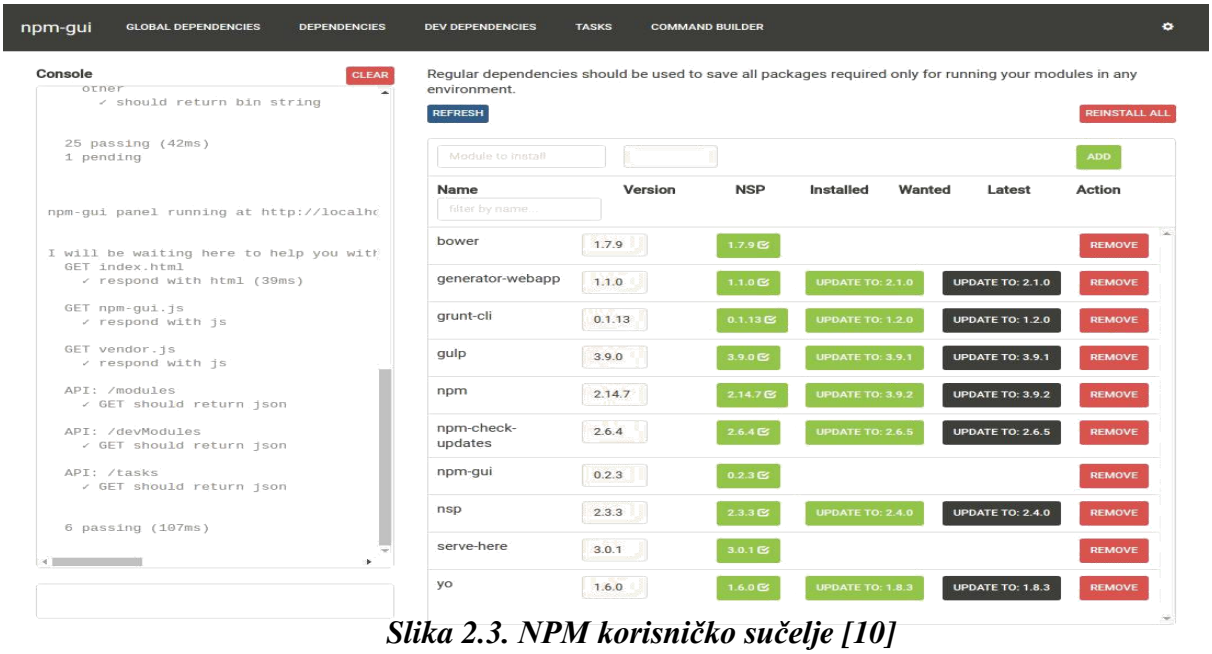
Treći alternativni alat je Nodyn. On je drugačiji od prethodno nabrojanih jer implementira kompletnu Node.js kompatibilnost što znači da nudi zapravo apsolutno drugačija rješenja, a ne alternativni pristup. JavaScript *engine* koji se koristi je Dyn.js koji je ujedno i bolji u usporedbi s Rhino kojeg koristi Ringo.js. Nodyn se može koristi na dva načina: kao kompletna zamjena za Node.js i/ili kao dodatak postojećem sustavu.

2.1.4. PurpleJS

PurpleJS je jedan od novijih projekata koji se koristi Nashorn JavaScript *engine*. Ovaj alat je napravljen za brzo razvijanje aplikacije, koristi se nitima te nije reaktivan. To može biti ograničenje, ali i prednost kod korisnika koji žele lakše izraditi i pregledati svoju aplikaciju. Kod PurpleJS-a smanjena je količina alata na alatnoj traci i to mu daje najveću prednost u pogledu jednostavnijeg korištenja i preglednosti. PurpleJS koristi se: kada je potreban JavaScript programski kod, ali aplikacija treba biti pokrenuta na JVM (Java Virtual Machine) [9]; kada je potrebno kreirati poslužiteljske aplikacije s JavaScriptom te pristupiti i koristiti Java biblioteku; kada je potrebno pokrenuti isti programski kod na obje strane- poslužiteljskoj i klijentskoj (npr. pretraživač); kada je potrebno proširiti Java aplikacije s JavaScriptom; kada je potrebno kreirati JavaScript poslužitelj koji će se moći izvršiti na bilo kojoj infrastrukturi te prilikom niza slične problematike.

2.2. Node Package Manager (NPM)

Kada se priča o Node.js-u, svakako se mora napomenuti korištenje Node Package Managera (NPM). To je alat koji dolazi zajedno sa svakom Node.js instalacijom. Ideja NPM nadogradnje je javna dostupnost dodatnih komponenti i resursa koji su dostupni kroz jednostavnu instalaciju. NPM služi kao *framework* (hrv. programerski okvir) za Node.js *development* (hrv. razvoj projekta) s velikom količinom raznih dodataka za razne Node.js aplikacije.



Slika 2.3. NPM korisničko sučelje [10]

Slika 2.3. prikazuje kako izgleda NPM korisničko sučelje. U korisničkom sučelju nalazi se s konzola za upisivanje programskog koda s lijeve strane. Crveni gumb s natpisom *Clear* čisti konzolu od napisanog programskog koda. Na desnoj strani slike se nalazi popis instaliranih paketa s odgovarajućom oznakom verzije te ponuđenim ažuriranjima.

NPM funkcionira na način da stvori jednostavan web poslužitelj iz datoteke koja se nalazi u projektu. NPM se može koristiti u nizu primjera: poslužuje GUI za preglednike (dostupno na http://localhost:1337/), modificira package.json, pokreće naredbe s klijentske strane (npr. „*npm install angular –save*“), prebacuje konzolu preko *websocket*-a do klijentske strane i tako dalje.
Neki od najpopularnijih NPM *package*-a [11] (paketa odnosno biblioteka) su: Lodash, Request, Async, Underscore i Express.

2.3. Express.js dodatak

Slika 2.4. Express.js dodatak [12]

Slika 2.4. prikazuje razloge zašto korisiti Express.js dodatak za razvoj web stranica. Uspoređuje se Express.js s drugim raznim JavaScript dodacima kao što su: Compound.JS, sails.js, partial.js i

derby.js. Dijagram sa slike prikazuje kako je Express.js daleko bolji od ostalih ako se gledaju bitne karakteristike poput konfiguracije, usmjeravanja, *middleware*-a, pogleda i predložaka, sesija te sigurnosti.

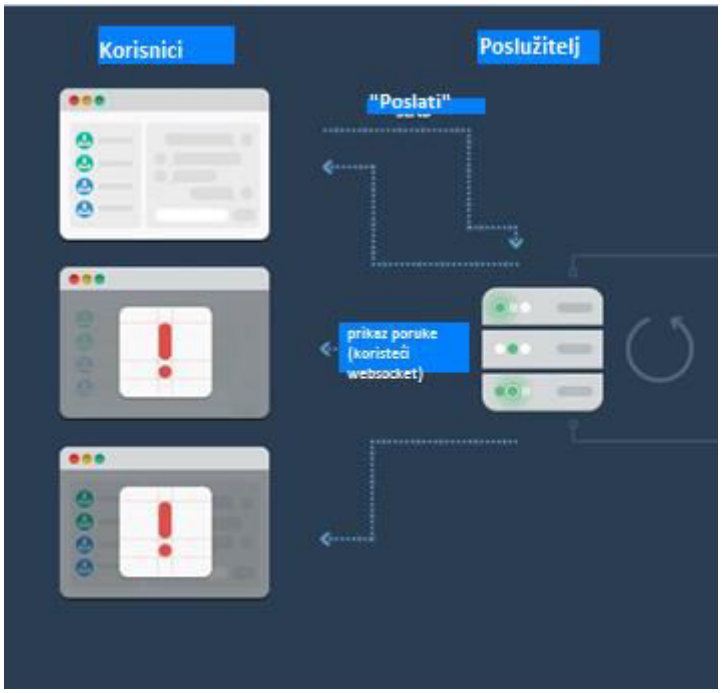
Express.js je najviše korišten Node.js-ov alat koji omogućuje skriptama da se izvršavaju konstantno i beskonačno. Riječ je o nekoliko linija programskog koda koje ubrzavaju prikaz napravljenih izmjena prilikom pregledavanja web stranice. HTTP *server framework* za Node.js služi kao osnova za dodatak Express. Express je poslužiteljski dodatak sa dvije najviše korištene web komponente na tržištu. Sadrži najpopularniju biblioteku (Redis) za JavaScript koja je pripremljena da se koristi na Node.js te zanimljivu skriptu koja korisnicima omogućuje da uz pomoć „Coffee“ (hrv. Kava) pišu svoje Node.js programe.

2.4. Primjena Node.js-a

Tipični primjeri gdje se Node.js može i treba koristiti jesu:

1) Chat

Chat (razgovori) je jedna od najviše korištenih aplikacija koje moraju konstantno biti pokrenuti i imaju više korisnika.



Slika 2.5. Node.js CHAT - način rada [13]

Slika 2.5. prikazuje način rada. Prikazana je jedna soba za *chat* gdje korisnici ulaze i gdje mogu izmjenjivati poruke. Na primjer, u *chat* sobi je troje ljudi koji su povezani. Na poslužiteljskoj strani nalazi se Express.js aplikacija koja implementira dvije stvari: 1. GET '/' zahtjev koji služi web stranici za inicijalizaciju unosa nove poruke, 2. *Websocket* poslužitelj koji zaprima nove poruke poslane od strane *websocket* korisnika.Na korisničkoj strani nalazi se HTML stranica koja sadrži *„Send"* gumb koje zaprima ulazne poruke i šalje ih prema *websocket*-u te još jedno koje osluškuje dolazne poruke. Kada jedan od korisnika pošalje poruku, događa se slijedeće: 1. Preglednik registrira pritisak na *„Send"* gumb preko JavaScript *handlera*, zaprima vrijednost poruke s polja za unos i prikazuje poruku koristeći *websocket* korisničku vezu na poslužitelja. 2. Poslužiteljska komponenta zaprima poruku i prosljeđuje ju. 3. Svi korisnici primaju nove poruke kao *push* poruke (notifikacije) preko *websocket* klijentske komponente pokrenute zajedno s web stranicom. Nakon zaprimanja poruke, poruka se prikazuje.

2) Proxy

Proxy poslužitelj može se definirati kao računalo- posrednik koji stoji između korisnika i glavnog poslužitelja. Proxy poslužitelji najčešće se koriste za posluživanje web stranica (uporaba Interneta). Postoje dvije vrste Proxy poslužitelja, to su: obični Proxy i obrnuti (reverzni) Proxy.
Node.js lako se može koristiti za izradu poslužiteljskog Proxy servisa jer može podnijeti više konekcija u isto vrijeme skoro bez ikakvog ograničenja. Posebno je koristan za korištenje Proxy poslužitelja za različite servise u različito vrijeme ili prikupljanje podataka sa različitih točaka.

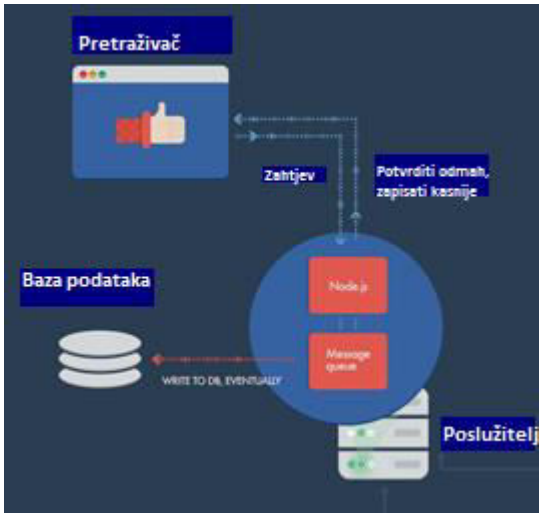
3) *Data Streaming* (tok podataka)

Kod tradicionalnih web platformi HTTP zahtjevi i odgovori tretiraju se kao izolirani događaji te su oni zapravo tokovi. To se može primijeniti pomoću Node.js-a. Moguće je procesirati datoteke dok se one još uvijek *uploadaju* na poslužitelj jer se zahtjevi ne tretiraju kao svaki zaseban događaj. Na primjer, tok podataka kod PHP-a započinje od diska prema memoriji te se nastavlja do poslužitelja. Kod Node.js-a postupak je efikasniji jer tok podataka ide iz memorije direktno prema poslužitelju, prije nego je početni protok prema disku završen.

4) *Inputs* (unosi)

Ako program primi veliku količinu podataka, baza podataka može se usporiti ili promet može potpuno zaustaviti tok podataka. Node.js može se prilagoditi svim količinama nadolazećeg podatkovnog prometa. No, pristup bazi podataka je operacija koja može blokirati trenutno pokrenuti program. Rješenje je preuzeti i zapisati podatke korisnika prije nego su zapisani u

samu bazu podataka. S takvim pristupom moguće je „sa strane“ pisati sve unose u bazu podataka. Primjeri takvog načina su: prijava korisnika u sustave s podacima koji se obrađuju u serijama i koji se ne koriste odmah, operacije čije odražavanje nije potrebano odmah (npr. broj pritisaka na gumb „sviđa mi se“ na Facebook-u).



Slika 2.6. Rješenje za veliki promet prema bazi podataka [14]

Slika 2.6. prikazuje rješenje za veliki promet informacija prema bazi podataka. Baza izvrši proces pretrage te nakon toga šalje rješenje na Node.js-ov poslužitelj. Tim procesom se korisnikov poslužitelj rasterećuje.

3. USPOREDBA PROGRAMSKIH JEZIKA I ALATA: PHP i NODE.js

PHP je napravljen 1994. godine od strane Rasmusa Lerdorfa [15]. PHP se danas koristi na više od 80% poslužitelja u svijetu [16], a tome najviše pridonosi WordPress platforma za izradu web stranica. Node.js je, kao što je već spomenuto, izrađen 2009. godine. Relativno je nov pa je još u fazi rasta i razvoja. Postoje dva glavna razloga zašto je odabrana usporedba upravo ova dva jezika te zašto su ostali poput Ruby, Python, C+, Java, Erlang itd. preskočeni. Razlozi su slijedeći:

- Sličnost ova dva jezika. Oba su *open source*, usmjereni su na web programiranje i mogu se primijeniti na slične projekte.

- Brzina rasta i razvoja Node.js programa. PHP programeri možda bi trebali razmisliti o prebacivanju na Node.js. U budućnosti Node.js bi mogao zamijeniti sveprisutni PHP.

3.1. Osnove PHP-a

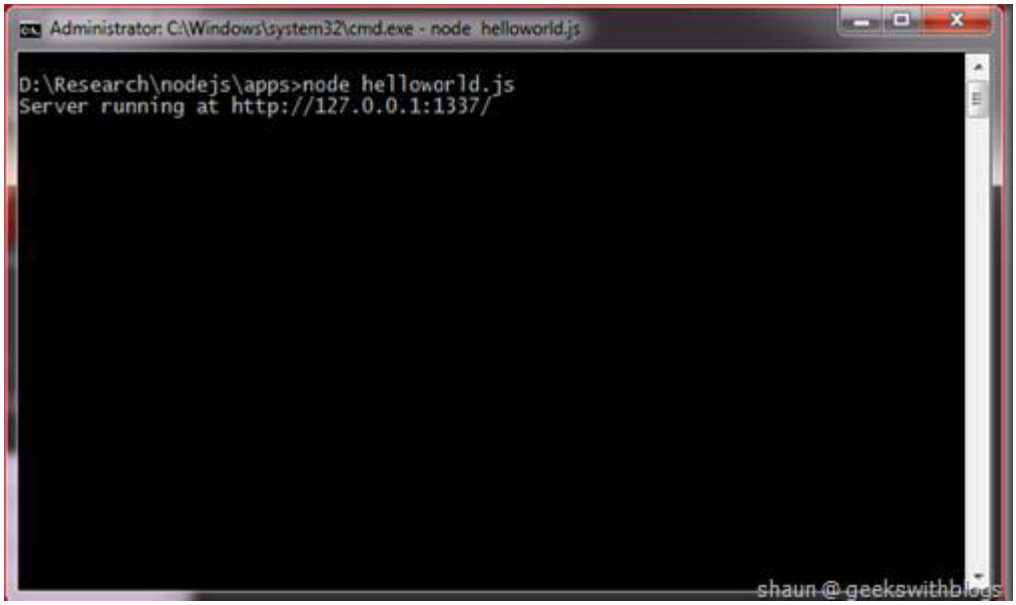
Kod pisanja programskog koda u PHP-u bitno je znati kako se može implementirati u bilo koju datoteku koja može biti interpretirana od strane PHP poslužitelja, a obično mora imati .php ekstenziju. Najjednostavniji primjer PHP programskog koda prikazuje slika 3.1. Usporedbe radi, isti takav primjer prikazuje i slika 3.2., samo za Node.js.

Slika 3.1. „Hello World“ u PHP-u [17]

Slika 3.2. „Hello World“ u Node.js-u [18]

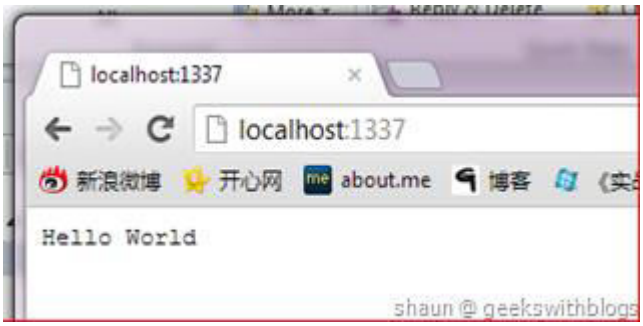
Kako bi se u PHP-u ispisalo „Hello World“ potrebna su svega tri retka te jedna funkcija. Prvo je potrebno staviti <? Php ?> kako bi se označilo da se radi o PHP programskom kodu. Zatim se koristi *echo()*[19] funkcija nakon koje slijede navodni znakovi. *Echo()* funckija u principu nije prava funkcija jer ona služi samo za ispis i nešto je brža od klasične funkcije *print()*. Unutar navodnih znakova unosi se *string* koji se ispisuje. Naposljetku je potrebno staviti interpukcijski znak ; kako bi se označio završetak linije programskog koda. Kod Node.js-a stvar je malo kompliciranija. Potrebno je koristiti više funkcija i dobro poznavati način rada same platforme. Prvo se koristi *var http = require ('http')* koji provjerava HTTP vezu, odnosno umeće ugrađeni HTTP model kako bi se mogao kreirati HTTP poslužitelj. HTTP poslužitelj kreira se naredbom createServer(). Naredba *res.writeHead()* postavlja zaglavlje i odgovara na zahtjev, dok *res.end()* služi za ispis i zatvara zahtjev.

U primjeru na slici 3.2. zadani programski kod će kreirati web poslužitelj na 1337 portu i ispisat će „*Hello World*“ za sve zahtjeve koje zaprimi. Poslužitelj se pokrene u komandnom prozoru na slijedeći način prikazan slikom 3.3. :



Slika 3.3. Komandni prozor

U komandni prozor prvo se treba napisati „node“ kako bi se prikazala Node.js konzola. Zatim treba otvoriti web preglednik i otići na adresu: http://localhost:1337/ kako bi se vidjela ispisana poruka „*Hello World*“.



Slika 3.4. Prikazana poruka na web pregledniku

Može se primijetiti da se prilikom korištenja Node.js-a ne kreira web aplikacija, već poslužitelj i to je jedna od bitnih prednosti korištenja Node.js-a.

3.2. Sintaksa PHP-a

Struktura pisanja je također drugačija. PHP sintaksa ponekad se mijenja između verzija, ali kompatibilnost uvijek ostaje zadovoljavajuća. Najvažnije je pisati PHP programski kod između <?php i ?> tagova kao što je već navedeno u primjeru na slici 3.1. Također je bitno koristiti točku-zarez ; na kraju programskog koda jer taj interpunkcijski znak govori da je PHP programski kod gotov. U odnosnu na JavaScript, PHP je kompliciraniji. JavaScript je precizniji, sintaksa je jednostavnija te na taj način privlači sve više programera. S druge strane, PHP pruža mogućnost za pravo objektno programiranje te ima velik broj gotovih ugrađenih fukcija. Takve gotove funkcije mogu uštediti puno vremena jer mogu sve izvršiti same umjesto pisanja dugačkih potprograma. Unatoč tomu JavaScript i dalje ima prednost nad PHP-om u ovoj kategoriji. U konačnici, JavaScript je sažetiji, može se u isto vrijeme koristiti na poslužiteljskom i klijentskom računalu bez prebacivanja fokusa na jedno od njih.

3.3. Podrška

S obzirom da je PHP stariji jezik i da je nastao puno prije lako je zaključiti kako PHP ima veću podršku prema korisnicima. PHP ima veći broj službene dokumentacije, priručnika, uputa i ostalih dokumenata koji korisnicima služe za olakšavanje primjene samog jezika.

S druge strane, imamo nedavno nastali Node.js koji ima puno manje dostupne literature u kojoj se opisuje i olakšava način korištenja. U ovom segmentu PHP vodi bitku.

3.4. Alati

Obje tehnologije imaju veliki broj alata koji pomažu pri izradi aplikacija. Node.js ima NPM odnosno Node Package Manager koji omogućava instaliranje i mijenjanje ovisnosti, postavljanje konfiguracije varijabli, definiranje skripti i slično [20].

3.5. Okolina

Web developeri često moraju raditi aplikacije koje nisu direktno povezane sa webom. Postoje načini kako koristiti PHP na *desktop* okolini, ali se oni rijetko koriste. PHP se uglavnom koristi na poslužiteljskoj strani. Node.js je JavaScript jezik napravio slobodnim i sveprisutnim. Prije nekoliko godina JavaScript je bio prilično ograničen te je njegovo mjesto primjene bilo usko povezano s preglednikom. Danas se JavaScript, zahvaljujući Node.js, može koristiti bilo gdje, na poslužitelju, u terminalu, na *desktopu* ili bilo gdje drugdje.

3.6. Intergracija

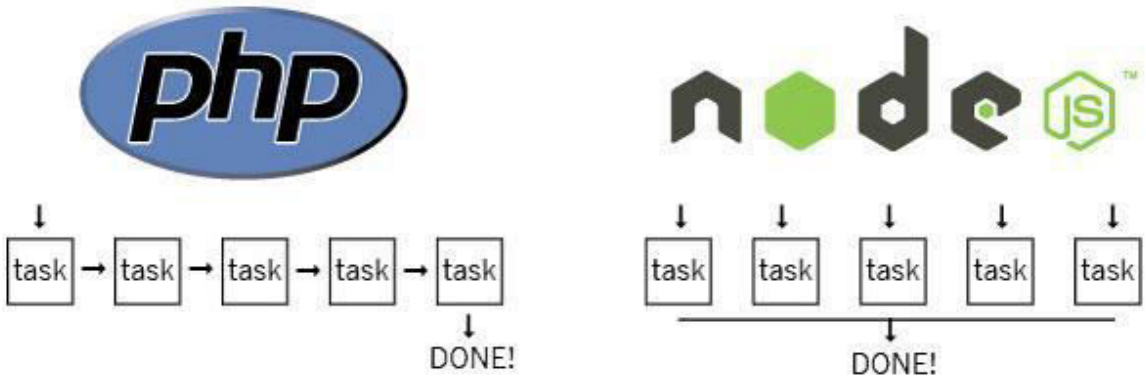
Tehnologije za daljnje razvijanje programa često su ograničene za korištenje ukoliko se ne mogu integrirati s odgovarajućim *driverima* i bazama podataka. U tom dijelu PHP je iznimno snažan jer ima sustav nadogradnji koje omogućuju komunikaciju sa velikim brojem API-ja. Node.js također je dobar u ovom području, ali ne toliko jer je nova, nedovoljno razvijena platforma. Brzo sustiže korak, ali odgovarajuće integracije za neke starije i manje popularne tehnologije i dalje nedostaju.

3.7. Hosting i primjena

Jedna od bitnih stvari na koje se treba misliti kod odabira programskog jezika i platforme je koliko je lako postaviti i pokrenuti aplikaciju na web poslužitelju. PHP je u ovom području dominantan jer postoji veliki broj tvrtki koje pružaju web *hosting*. Također veliki broj tvrtki nudi i podršku za PHP. Node.js koristi različit način. Potrebno je imati specifičnu konfiguraciju na poslužitelju, a ponekad i mogućnost SSH pristupa. Često se dogodi da je to ipak preveliki zahtjev za web *hosting* pa su ponude takvih usluga dosta rjeđe. Node.js će kroz vrijeme sigurno omogućiti drugačiji i još lakši način, ali će biti teško dostići PHP razinu.

3.8. Perfomanse PHP-a i Node.js-a

Performanse najčešće ovise o projektima i iskustvima *developera* koji razvijaju programski kod, odnosno o količini znanja koje posjeduju te o funkcionalnosti koju programski kod posjeduje. U nekim istraživanjima pokazano je kako je Node.js prilično brži od PHP-a. To se događa zbog određenih prednosti koje Node.js ima, odnosno zbog drugačijih metoda koje koristi prilikom izvršavanja.



Slika 3.5. Niti i događaji

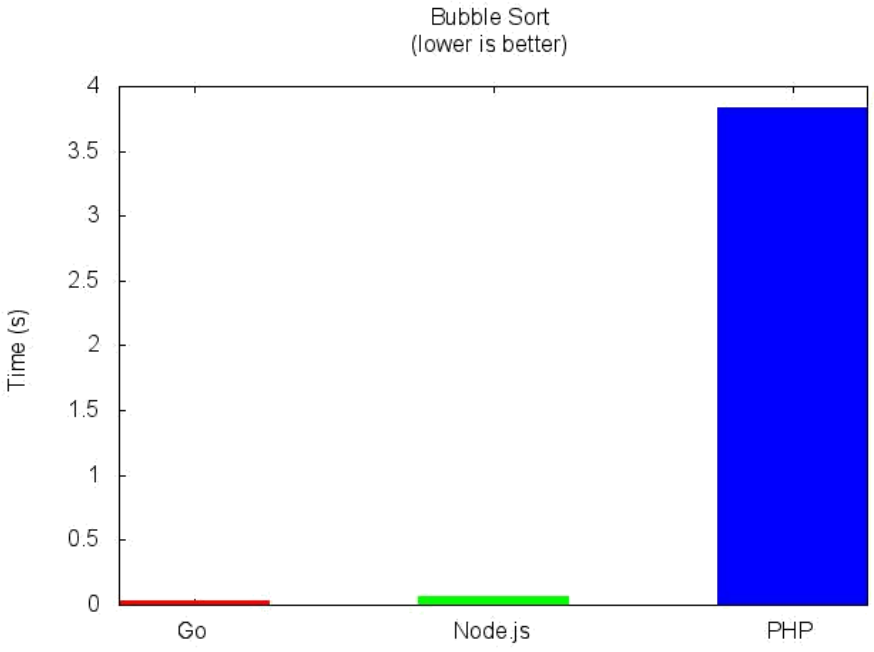
Slika 3.5. prikazuje usporedbu toka procesa prilikom rada u PHP-ovim događajima u odnosu na niti kod Node.js-a koje zadatak izvršavaju paralelno. PHP izvršava jedan po jedan događaj zbog uvjeta i petlji koje koristi [21].

Neke od bitnijih preformansi Node.js-a su: manje ovisnosti o web poslužitelju, uvijek uključene aplikacije koje su potrebne za izvršavanje zadanog programskog koda i korištenje događaja umjesto niti. Node.js je manje ovisan o web poslužitelju iz razloga što PHP zahtjeve mora provesti kroz web poslužitelj, dok Node.js ne mora nužno. Nadalje, Node.js uvijek ima uključene aplikacije koje se prilikom korištenja, odnosno izvršavanja zahtjeva, samo jednom se trebaju učitati. Kod PHP-a način je malo drugačiji. Kod svih zahtjeva svaki put se moraju konfigurirati potrebni parametri nakon čega se spaja na bazu podataka, dohvaćaju informacije i prevodi HTML. Na posljetku, korištenje događaja umjesto niti. PHP koristi niti i treba pokrenuti naredbu koja će dohvatiti informacije iz baze podataka. Nakon toga, na sljedeću naredbu će čekati dok se prijašnja ne izvrti. Kod Node.js-a takvog čekanja obično nema jer koristi događaje.

3.8.1. Preformanse PHP-a i Node.js-a: Primjer za naredbu *sort*

Provedeno je istraživanje koje procjenjuje razlike u performansama između Node.js-a, PHP-a i Go.js-a. Go.js je programski jezik koji sadrži JavaScript biblioteku i omogućuje jednostavno kreiranje interaktivnih dijagrama u svim modernim web pretraživačima. Podržava grafičke predložke i objekte. Uz PHP i Node.js uzeta je i Go.js platforma kako bi se na što boljem i jasnijem primjeru prikazale njihove preformanse. U istraživanju za procjenu razlika uzet je obični *Bubble sort* programski kod. *Bubble sort* je sortirajući algoritam, odnosno algoritam za razvrstavanje. Radi na principu ponavljanja koraka kroz određene uvjete. Uspoređuje svaki par susjednih stavki te ih zamjenjuje ako se nalaze u pogrešnom redoslijedu. Postupak se ponavlja sve dok ne prestanu zahtjevi za zamjenom, što znači da je sve razvrstano prema uvjetima. Navedeni programski kod uzet je za primjer jer je jednostavan, većini korisnika poznat i daje dobar i jasan osvrt koliko je dugo potrebno pojedinom jeziku za izvedbu.

Bubble sort programski kod napisan je u tri jezika: Node.js, Go.js i PHP te je nakon toga pokrenut. Rezultati na slici 3.6. pokazuju kako je Go.js najbrža opcija, odmah iza njega slijedi Node.js dok je PHP pokazao veliku količinu sporosti u usporedbi s druge dvije opcije.



Slika 3.6. "Bubble Sort" – GoJS, Node.js i PHP [22]

Slika 3.6. Prikazuje usporedbu. Što je niži vremenski period (Time) to je opcija bolja.

Kao što je vidljivo iz ovog primjera s *Bubble sort* programskim kodom, prilikom računanja i izvođenja operacija PHP ima lošije preformanse u odnosu na Node.js. Znatno je sporiji u izvođenju.

3.9. Budućnost PHP-a i Node.js-a

Bez obzira na kojem poslužiteljskom jeziku korisnik radi, moći će na njemu raditi još duži niz vremena bez obzira čak i ako je njegovo razvijanje bude prekinuto. Ako usporedimo Node.js i PHP, može se reći kako će Node.js sigurno preuzeti dio tržišta od PHP-a jer se razvija velikom brzinom te postaje sve više konkurentan.

3.10. Zaključak: Node.js i PHP

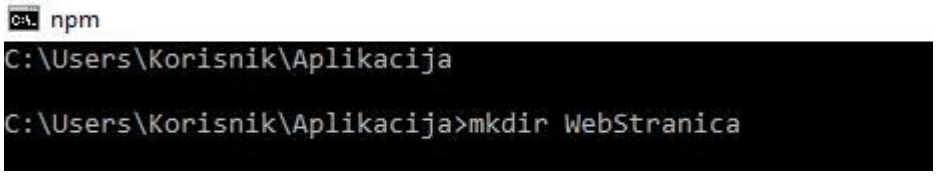
Node.js je kompliciraniji za učenje i shvaćanje te nije preporučljiv potpuno novim korisnicima u web programiranju. PHP se lakše uči, postoji ogromna količina korisničke podrške i dokumentacije s profesionalnim tehnikama učenja. Pomoć možete dobiti na bilo kojem mjestu, a integracija je jednostavna. PHP je definitivno bolji izbor za sve jednostavnije web stranice i aplikacije. Iz tih razloga bolje je početi s PHP-om i zatim se prebaciti na Node.js ili ih koristiti u kombinaciji, koristeći prednosti jednog i drugog.

4. NODE.js PRIMJER – WEBSTRANICA

Cilj projekta je prikazati kako funkcioniraju Node.js i Express.js te izraditi funkcionalnu web stranicu.

4.1. Proces izrade projekta

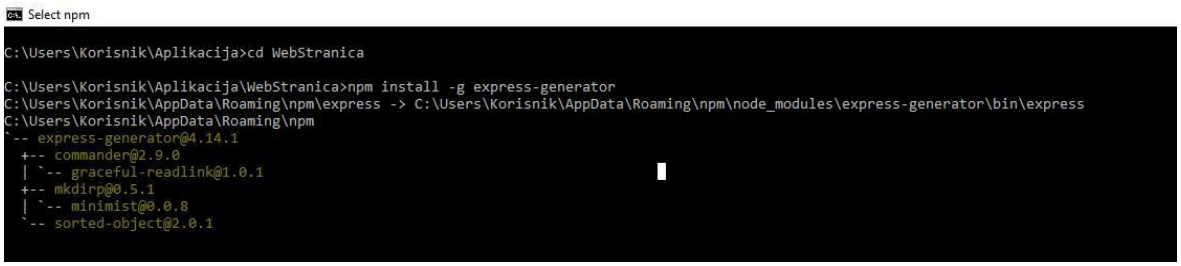
Za početak je potrebno izraditi mapu u kojoj će se nalaziti Node.js i naša stranica.



Slika 4.1. Pravljene radne mape

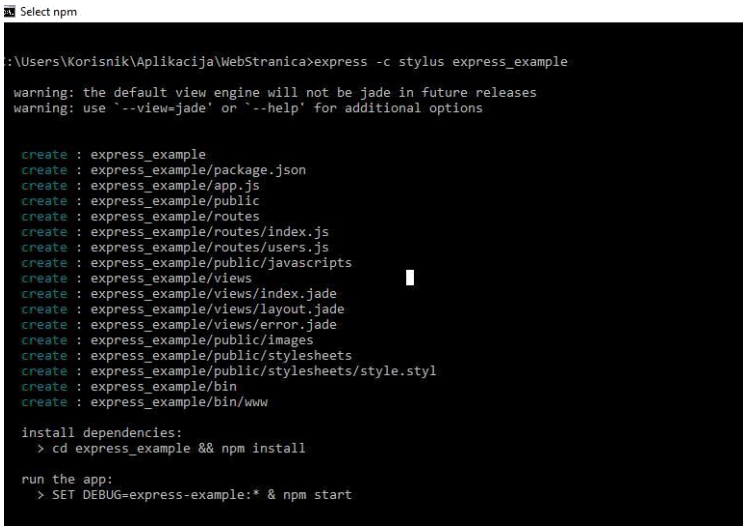
Slika 4.1. Prikazuje kako izgleda kreiranje mape projekta. Mapa je nazvana „Aplikacija“. Funkcija *mkdir* stvara direktorij „Web stranica“ u mapi „Aplikacija“.

Budući da će stranica biti izrađena uz pomoć Express.js alata, potrebno ga je instalirati. Postupak instalacije je prikazan na slici 4.2.



Slika 4.2. Instalacija alata Express.js

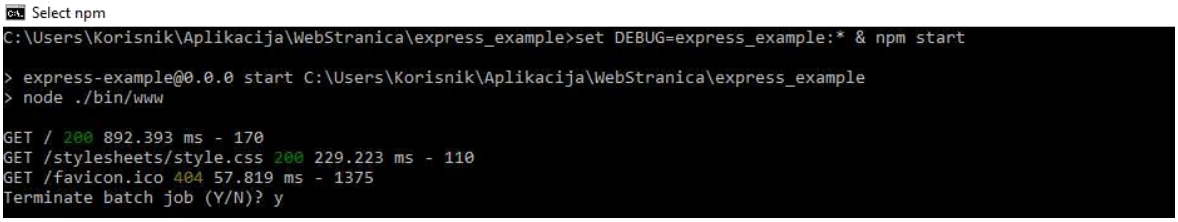
U konzolu se unese naredba za instalaciju: „npm install –g express-generator“. Može se dogoditi da dođe do greške nakon unosa ove naredbe. Ako se to dogodi, potrebno je koristiti naredbu *sudo*: „sudo npm install –g express-generator“. Oznaka –g označava da se alat Express.js instalira globalno.
Nakon instalacije Express.js alata potrebno je kreirati Express aplikaciju.



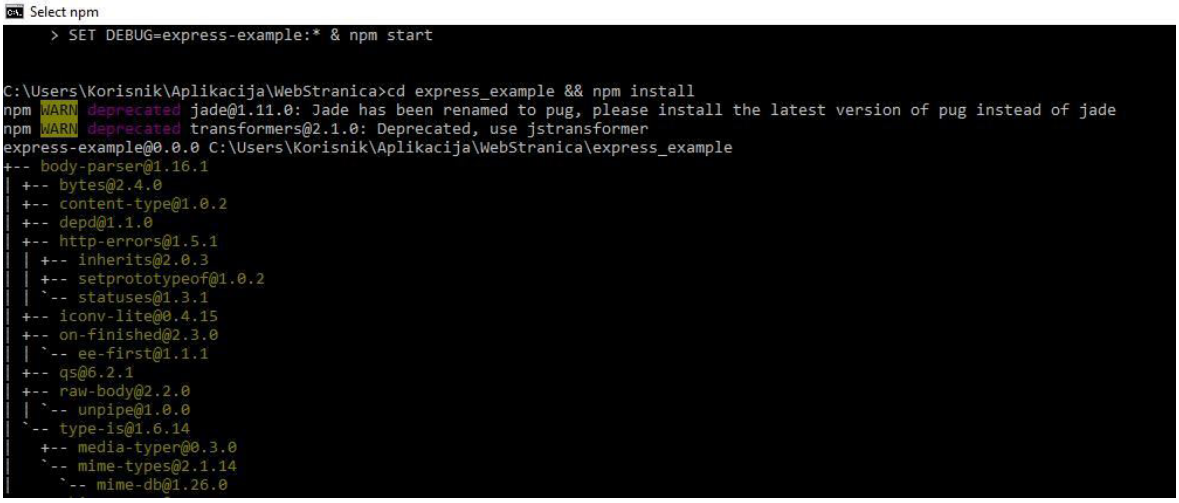
Slika 4.3. Kreiranje Express.js stranice

Slika 4.3. Prikazuje kreiranje Express.js stranice. U komandni prozor unese se naredba: „express –c stylus express_example“. Oznaka –c označava da će se koristiti CSS. Nakon toga, unesu se potrebne linije koda za kreiranje datoteka za stranicu.

Nakon kreiranja svih potrebnih datoteka za stranicu, sljedeći korak je pratiti navedene instrukcije odnosno pokrenuti dvije komande prikazane na kraju instalacije.



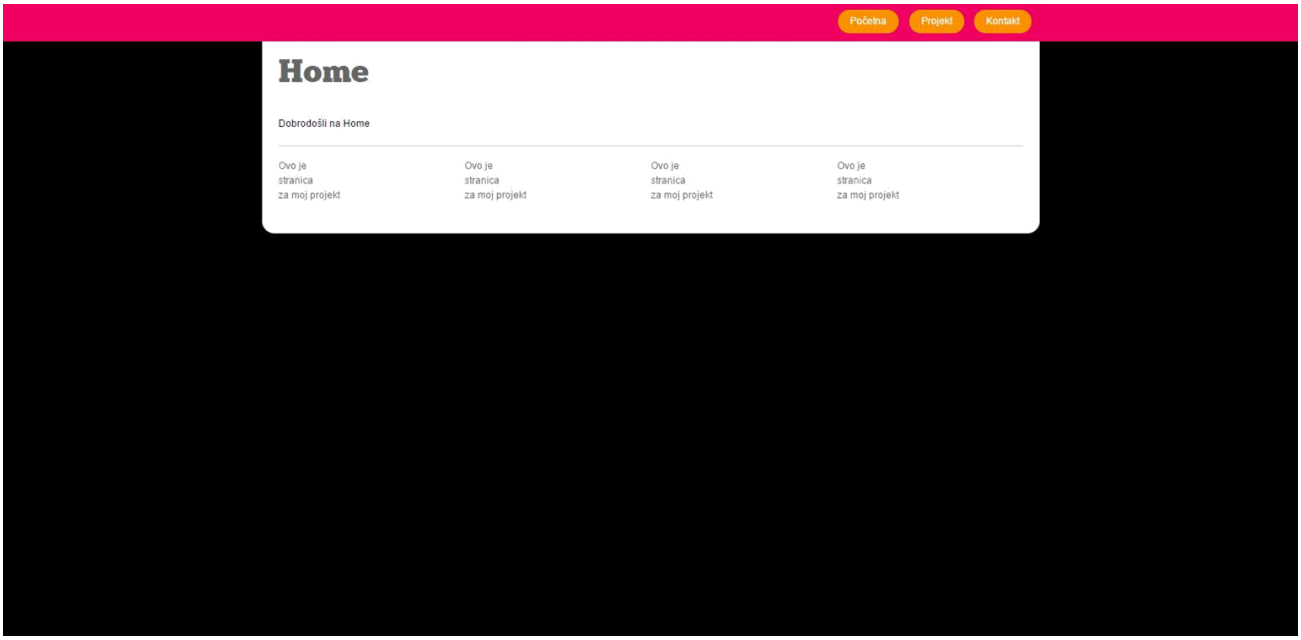
Slika 4.4. Instalacija



Slika 4.5. Pokretanje aplikacije

Slika 4.5. prikazuje način pokretanja aplikacije. Koristi se naredba: „set DEBUG=express_example:* & npm start“ (za Windows operacijski sustav) ili „DEBUG=express_example:* npm start“ (za OSX ili Linux operacijske sustave).
Adresa za provjeru web stranice odnosno svega što je napravljeno do sada je: http://127.0.0.1:300

4.2. Rezultat



Slika 4.7. Konačan rezultat

Slika 4.7. prikazuje konačan izgled stranice. Kao što je navedeno na početku, konačan rezultat je osnovni dio stranice izrađene na Node.js platformi. Ovo je samo osnovni prikaz, za daljnje uređivanje stranice potrebno je naprednije znanje HTML i CSS jezika, kao i znanje Node.js i Express alata.

5. ZAKLJUČAK

Node.js namijenjen je za rad na namjenskom HTTP poslužitelju. Node.js aplikacije su događaji pokrenuti asinkrono. Programski kod napisan na Node.js platformi ne slijedi tradicionalni model primanja, obrade, slanja i čekanja podataka. Node.js obrađuje dolazne zahtjeve i stalne događaje šaljući zahtjeve jedan za drugim, bez čekanja odgovora koristeći događaje. Platforma je bazirana na Google-ovom V8 *engine*-u, što mu je istovremeno i prednost i mana. Prednost je iz razloga što je brz i aplikacije se na obje strane (na poslužiteljskoj i korisničkoj) pišu u jednom jeziku. Mana je što podržava samo određene arhiktekture. U odnosu na svepristuni PHP, Node.js je nešto kompliciraniji i teži za početnike. Za određene stvari PHP nudi lakša, brža i elegantnija rješenja. Jedna od glavnih prednosti Node.js je to što ne blokira ulaz/izlaz (I/O). Međutim, neki od programera vrlo su kritični prema Node.js-u. Ističu slijedeće: ako jedan proces zahtijeva znatan broj CPU (eng. *Central processing unit)* ciklusa, moguće je da će aplikacija blokirati, a blokiranje može srušiti aplikaciju. Zagovornici Node.js modela tvrde da je vrijeme za procesiranje podataka znatno brže i efikasnije zbog velikog broja malih procesa na kojima se Node.js temelji. Svaki programer, odnosno korisnik treba razmotiriti koje su njegove potrebe i zahtjevi te prema tome odlučiti koju platformu će koristiti. Naposljetku jedno je sigurno, Node.js se razvija i širi velikom brzinom te postaje ozbiljan konkurent i suparnik.

6. LITERATURA

[1] https://gist.github.com/maxogden/d96123138522c84cdb25 5.5.2017.

[2] https://en.wikipedia.org/wiki/Node.js#History10.2.2017.

[3] https://www.quora.com/What-companies-are-using-Node-js-in-production 10.02.2017.

[4] https://github.com/smith/narwhal-jaxer 5.5.2017.

[5] https://www.growthaccelerationpartners.com/blog/node-js-and-express-part-i-server-side-event-driven.programming/ 10.2.2017.

[6] https://raw.githubusercontent.com/joyent/node/v0.12.0/LICENSE 10.2.2017

[7] https://www.tutorialspoint.com/nodejs/nodejs_quick_guide.htm10.2.2017.

[8] https://en.wikipedia.org/wiki/Reactive_programming 11.02.2017.

[9] https://hr.wikipedia.org/wiki/Java_(virtualni_stroj) 5.5.2017.

[10] https://www.npmjs.com/package/npm-gui12.2.2017

[11] https://www.npmjs.com/12.2.2017

[12] http://www.slideshare.net/EyalV/nodejs-express-37327755 12.2.2017

[13] http://blog.apcelent.com/why-the-hell-would-i-use-nodejs.html 13.2.2017.

[14] http://www.slideshare.net/Toptal/why-the-hell-would-i-use-nodejs13.2.2017

[15] https://en.wikipedia.org/wiki/Rasmus_Lerdorf13.2.2017

[16] http://www.onlinetechtutorials.com/2011/11/hello-world-program-in-php.html13.2.2017

[17] http://geekswithblogs.net/shaunxu/archive/2012/09/14/node.js-adventure---node.js-on-windows.aspx13.2.2017

[18] https://w3techs.com/technologies/overview/programming_language/all 13.2.2017

[19] https://www.npmjs.com/14.2.2017

[20] https://www.w3schools.com/php/func_string_echo.asp 5.5.2017.

[21] https://servercheck.in/blog/moving-functionality-nodejs-increased-server14.2.2017

[22] https://jaxbot.me/articles/benchmarks_nodejs_vs_go_vs_php_3_14_2013 8.5.2017.

7. SAŽETAK

Node.js je relativno nova platforma koja još treba dostići svoj vrhunac. Glavni cilj Node.js platforme je pružiti siguran i lak način za izgradnju mrežnih aplikacija visoke performanse u JavaScriptu. U radu su prikazane glavne preformanse platforme, njene prednosti kao i nedostaci, alternative koje mogu zamijeniti Node.js. Na kraju je napravljena usporedba programskih jezika i alata (PHP i Node.js).

Ključne riječi: Node.js, platforma, JavaScript, mrežne aplikacije, programski jezik, programski alat, PHP

ABSTRACT

Node.js is relatively new platform which still isn't reach its peak. The main goal of Node.js platform is to provide a safe and easy way to build high-performance network applications in JavaScript. In this final work the main platform performances are shown, its advantages, disadvantages and alternatives that can replace Node.js. At the end a comparison of program languages and tools (PHP and Node.js) is made.

Key words: Node.js, platform, JavaScript, network applications, program language, program tool, PHP

ŽIVOTOPIS

Roberta Raguž rođena je 07. siječnja 1995. godine u Požegi. Adresa prebivališta je Vilima Korajca 38, 34000 Požega. Osnovnu i srednju školu završila je u Požegi. Osnovna škola koju je pohađala nosi naziv Dobriša Cesarić. Naziv srednje škole koju je pohađala je Gimnazija Požega, smjer prirodoslovno- matematički. Akademske godine 2013/14. upisala je stručni studij Informatike na Fakultetu elektrotehnike, računarstva i telekomunikacija u Osijeku.
