Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4
DOI: https://doi.org/10.24867/30BE30Ruvceski

PERFORMANSE MIKROSERVISNE APLIKACIJE U NODE.JS I .NET OKRUŽENJU POSTAVLJENO NA AWS-U

PERFORMANCES OF MICROSERVICE APPLICATION IN NODE.JS AND .NET ENVIRONMENT DEPLOYED ON AWS

Teodora Ruvčeski, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Sa rastom upotrebe mikroservisne arhitekture u razvoju softverskih aplikacija, postaje važno razumeti kako različite tehnologije, kao što su Node.js i .NET, utiču na performanse ovakvih sistema. Ovaj rad istražuje i upoređuje performanse mikroservisnih aplikacija razvijenih u Node.js i .NET tehnologijama, kroz analizu tri specifična toka podataka. Merenja su izvršena na identičnim aplikacijama, koje implementiraju iste funkcionalnosti, ali koriste različita tehnološka okruženja. Rezultati pokazuju razlike u efikasnosti i brzini odziva između ove dve platforme, pružajući uvide u njihove prednosti i slabosti u realnim uslovima mikroservisnih okruženja.*

Ključne reči: *.NET, Node.js, Mikroservisi, AWS, Performanse*

Abstract – *With the increasing adoption of microservice architecture in software development, it becomes important to understand how different technologies, such as Node.js and .NET, impacts the performance of such systems. This paper explores and compares the performance of microservice applications developed in Node.js and .NET technologies through the analysis of three specific data flows. The measurements were conducted on identical applications, implementing the same functionalities but using different technological environments. The results highlight differences in efficiency and response speed between these two platforms, providing insights into their strengths and weaknesses in real-world microservice environments.*

Keywords: *.NET, Node.js, Microservices, AWS, Performances*

1. UVOD

Savremeni razvoj softverskih rešenja sve češće se oslanja na mikroservisnu arhitekturu, koja omogućava modularni pristup kreiranju aplikacija kroz male, nezavisne servise. Ovaj pristup donosi brojne prednosti, uključujući veću skalabilnost i fleksibilnost, ali i jednostavnije održavanje sistema. Međutim, pri izboru tehnologije za razvoj mikroservisa, performanse igraju ključnu ulogu.

U ovoj studiji analizirane su performanse aplikacija razvijenih u dva popularna okruženja – Node.js i .NET. Oba tehnološka rešenja imaju široku primenu u industriji, ali se razlikuju u pristupu obradi zadataka i upravljanju resursima. Node.js [1] se oslanja na asinhroni model koji omogućava visoku skalabilnost u realnim vremenskim aplikacijama, dok .NET pruža robustan okvir za izgradnju visoko performantnih rešenja sa naglaskom na stabilnost i integraciju sa različitim sistemima.

Cilj istraživanja jeste da se kroz seriju testova proceni efikasnost i brzina odziva aplikacija razvijenih u ova dva okruženja. Testovi su uključivali različite scenarije, poput interakcije sa bazom podataka, komunikacije sa eksternim servisima i vremena potrebnog za pokretanje mikroservisa. Ova analiza pruža uvid u specifične prednosti i izazove svake od tehnologija, čime doprinosi boljem razumevanju njihovih performansi u kontekstu realnih zahteva softverskih sistema.

2. KORIŠĆENE TEHNOLOGIJE

U ovom istraživanju korišćeno je nekoliko ključnih tehnologija kako bi se omogućilo testiranje i poređenje performansi aplikacija. Aplikacije su razvijene u mikroservisnom okruženju, arhitekture koja omogućava modularan pristup razvoju softverskih rešenja [5]. Svaki mikroservis predstavlja nezavisnu komponentu zaduženu za specifične funkcionalnosti, što olakšava skaliranje i održavanje aplikacija. Ovakva arhitektura omogućava bolju distribuciju resursa i bržu obradu zahteva, a koristi se u modernim aplikacijama koje zahtevaju visoku efikasnost i fleksibilnost.

Za implementaciju mikroservisa korišćen je .NET, tehnologija koja omogućava razvoj robustnih, sigurnih i visoko performantnih aplikacija. Jedna od glavnih prednosti .NET platforme je njena sposobnost da se lako integriše sa širokim spektrom sistema i tehnologija, što je posebno važno u kompleksnim poslovnim okruženjima. .NET podržava *multithreading* i paralelno izvršavanje zadataka [3], čime se postiže optimizovano upravljanje resursima i smanjuje vreme izvršavanja zahteva. Osim toga, .NET pruža napredne alate za praćenje performansi, debugovanje i profilisanje aplikacija, što olakšava održavanje sistema u produkcionim uslovima [4].

S druge strane, Node.js je korišćen zbog svoje sposobnosti da efikasno upravlja velikim brojem istovremenih konekcija zahvaljujući asinhronom modelu [2] izvršavanja. Ova tehnologija je poznata po svojoj brzini i skalabilnosti, što je čini idealnom za aplikacije

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Darko Čapko, red. prof.

koje zahtevaju visok nivo interakcije sa korisnicima i brze odgovore. Node.js koristi *event-driven* arhitekturu [6], što omogućava neblokirajuće operacije, idealne za aplikacije u realnom vremenu i one koje često komuniciraju sa eksternim API-jevima. Za razvoj API-ja unutar Node.js okruženja korišćen je Express, lagan i fleksibilan framework koji značajno ubrzava proces razvoja, omogućavajući jednostavno rukovanje HTTP zahtevima i integraciju sa različitim servisima.
Aplikacije su postavljene na AWS EC2 instancama, pružajući *cloud* okruženje za pokretanje i testiranje performansi u identičnim uslovima. Za upravljanje bazama podataka korišćena je Supabase platforma, zajedno sa PostgreSQL bazom podataka, koja je omogućila efikasno skladištenje i pristup podacima za potrebe aplikacija.
Testiranje zahteva i performansi aplikacija izvršeno je korišćenjem alata Postman, koji je korišćen za simulaciju stvarnih *HTTP* zahteva prema aplikacijama, omogućavajući preciznu analizu vremena odgovora i opterećenja aplikacija.
Ove tehnologije su odabrane zbog njihove fleksibilnosti, skalabilnosti i mogućnosti da podrže moderne aplikacije u realnom vremenu, što ih čini idealnim za ovo istraživanje performansi.

3. ARHITEKTURA APLIKACIJE

Aplikacija korišćena za testiranje dve tehnologije, .NET i Node.js, predstavlja složenu web aplikaciju u formi više različitih API-ja koji čine jedan mikroservis (*Application Programming Interface*). Ovi API-ji omogućavaju komunikaciju između klijentskih aplikacija i servera putem definisanih pristupnih tačaka (*endpoints*), pružajući standardizovanu razmenu podataka. API se može posmatrati kao jedan od načina primene principa Crne kutije (*Black Box*), gde klijent koristi funkcije koje sistem pruža, bez potrebe da zna kako su te funkcije implementirane unutar sistema. API pristup se često koristi kako bi se unifikovala komunikacija između različitih sistema i svih klijentskih aplikacija. Struktura i arhitektura aplikacije biće detaljnije opisano u nastavku rada, sa posebnim fokusom na tehničke aspekte implementacije API-ja i testiranja u oba tehnološka okruženja.

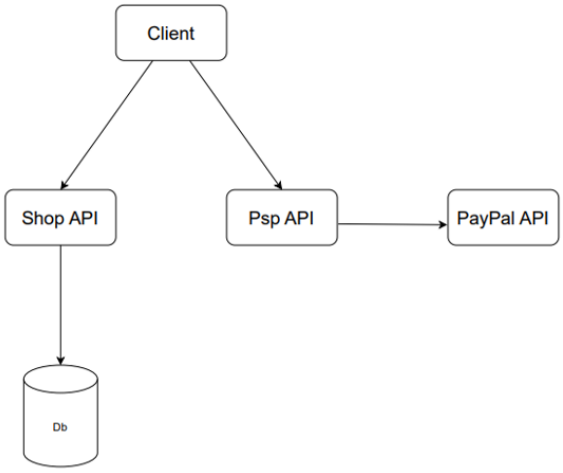
3.1 API

API (Programski interfejs za aplikacije) omogućava komunikaciju između softverskih sistema definišući pravila interakcije sa funkcijama i podacima sistema. Kroz pristupne tačke (*endpoints*), API unifikuje komunikaciju između različitih sistema, bez potrebe da klijent poznaje detalje implementacije. U dokumentaciji su opisane pristupne tačke, parametri i očekivani odgovori, čineći API ključnim za efikasnu razmenu podataka.

3.2 Test Aplikacija

Testirana aplikacija je platforma za potragu za poslom, omogućavajući kompanijama objavu oglasa i kandidatima da kreiraju profile i postavljaju CV-jeve. Ključna funkcionalnost je kupovina premium paketa, uz podršku za sigurne online transakcije putem PayPal-a. Aplikacija je implementirana kao sistem mikroservisa (*shop-api, psp-api, paypal-api, bank-api*), uz NGINX koji orkestrira komunikaciju.

Back-end aplikacije razvijen je u oba okruženja, .NET i Node.js, sa identičnim *endpoints*-ima i logikom, što omogućava direktno poređenje performansi i skalabilnosti ova dva tehnološka *stack-a*.



Slika 1. *Dijagram komponenti sistema*

3.3 Deployment aplikacije na AWS

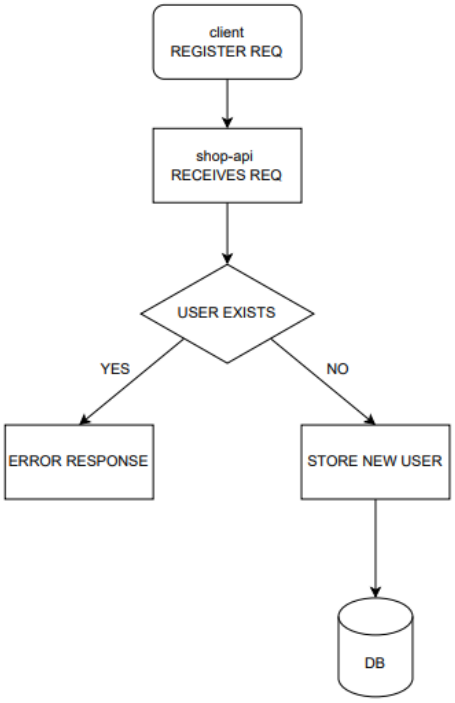
Proces deploymenta mikroservisne aplikacije na AWS infrastrukturu izveden je korišćenjem EC2 instanci, kako bi se omogućilo kompetentno poređenje performansi između Node.js i .NET aplikacija. Oba okruženja koriste *t2.micro* instance sa istim konfiguracijama, uključujući 1 vCPU, 1 GB RAM-a i 8 GB EBS skladišta.

Za Node.js aplikaciju, EC2 instanca sa *Amazon Linux 2 AMI* je konfigurisana za pokretanje aplikacije, uz instalaciju Node.js, NGINX-a i *deployment* putem SCP-a. Za .NET aplikaciju, EC2 instanca sa Windows Server 2019 je korišćena, uz instalaciju IIS-a i .NET Core Runtime-a, te deployment unutar IIS-a. Nakon *deploymenta*, oba sistema su testirana kako bi se osiguralo ispravno funkcionisanje aplikacija, omogućavajući poređenje njihovih performansi u identičnim *cloud* okruženjima.

4. REZULTATI TESTIRANJA

U narednim poglavljima biće prikazani rezultati testiranja aplikacija u različitim scenarijima kako bi se detaljno analizirale performanse oba tehnološka okruženja. Testovi su sprovedeni kako bi se procenila efikasnost aplikacija u ključnim aspektima kao što su brzina obrade zahteva, komunikacija sa eksternim servisima i vreme pokretanja sistema.
Testiranje je vršeno merenjem vremena potrebnim za izvršenje svakog pojedinačnog zahteva do trenutka dobijanja odgovora od servera. Zahtevi prema API-ju slati su putem *Postman* aplikacije. Za svaki scenario merenje je izvođeno više puta zatim je računata srednja vrednost merenja, devijacija vrednosti i distribucija izmerenih vremena.
Ova analiza pruža jasne uvide u mogućnosti i ograničenja korišćenih tehnologija, omogućavajući bolje razumevanje njihovih performansi u realnim uslovima.

4.1 Interakcija sa bazom podataka – registracija korisnika

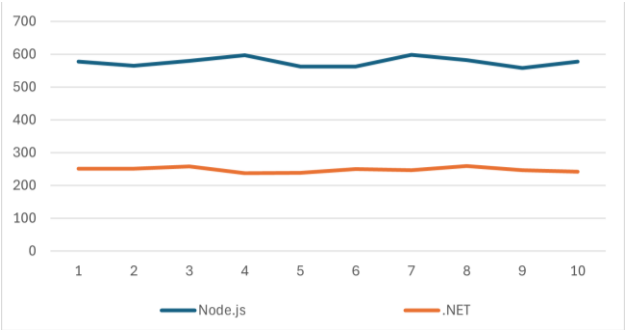


Slika 2. *Dijagram toka podataka registracije korisnika*

Prvi test obuhvata interakciju sa bazom podataka tokom registracije korisnika, pri čemu su merenja vršena za vreme potrebno da korisnički zahtev bude obrađen i podaci upisani u bazu. U oba okruženja izvršeno je 10 merenja.

Prosečno vreme izvršavanja za Node.js iznosi 575 ms, uz standardnu devijaciju od 14.46 ms. S druge strane, za .NET aplikaciju prosečno vreme izvršavanja iznosi 248 ms, sa standardnom devijacijom od 7.55 ms.

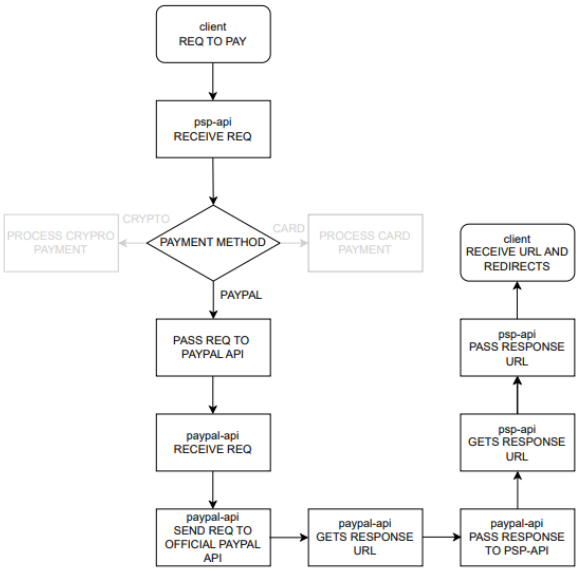
Rezultati pokazuju da je .NET značajno brži u obradi zahteva i upisivanju podataka u bazu, što ukazuje na efikasniji način upravljanja resursima tokom I/O operacija u ovom okruženju.



Slika 3. *Distribucija izmerenih vremena za registraciju*

4.2 Komunikacija sa eksternim servisom – plaćanje putem PayPal-a

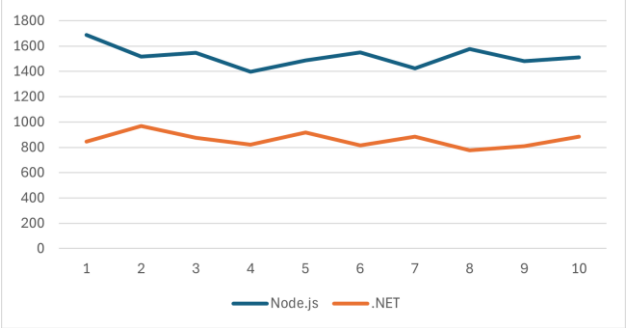
U drugom testu mereno je vreme komunikacije sa PayPal API-jem prilikom obrade plaćanja. I u ovom slučaju je izvršeno 10 merenja za oba okruženja.



Slika 4. *Dijagram toka podataka paypal plaćanja*

Prosečno vreme izvršavanja za Node.js je 1517 ms, dok je standardna devijacija 81.39 ms, što ukazuje na varijacije u merenjima. Za .NET aplikaciju, prosečno vreme iznosi 859 ms, uz standardnu devijaciju od 57.26 ms, što pokazuje nešto manju varijabilnost u odnosu na Node.js.

Ovde .NET takođe pokazuje prednost, sa bržim vremenom odgovora u odnosu na Node.js, što je važno za aplikacije koje zahtevaju brzu i sigurnu obradu plaćanja.



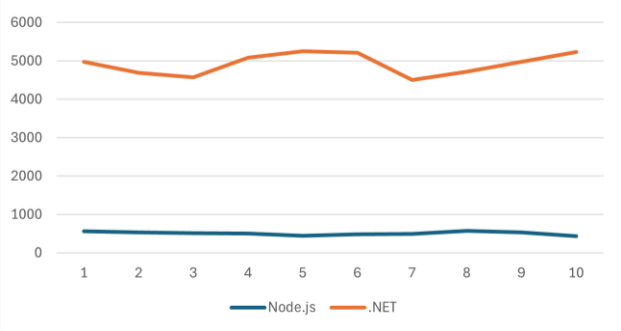
Slika 5. *Distribucija izmerenih vremena za paypal plaćanje*

4.3 Inicijalizacija i pokretanje mikroservisa

Treći test fokusirao se na vreme potrebno za kompletno pokretanje mikroservisa, uključujući inicijalizaciju resursa i povezivanje sa bazama podataka. Test je obavljen sa 10 merenja u oba okruženja.

Prosečno vreme izvršavanja za Node.js iznosi 507 ms, uz standardnu devijaciju od 45.03 ms. S druge strane, za .NET aplikaciju, prosečno vreme je značajno veće, iznoseći 4919 ms, sa standardnom devijacijom od 282.19 ms, što ukazuje na veću varijabilnost u merenjima.

Node.js pokazuje veliku prednost u brzini pokretanja, zahvaljujući svojoj jednostavnijoj arhitekturi i brzini JavaScript izvršavanja, dok .NET zahteva značajno duže vreme za inicijalizaciju zbog složenijih procesa prevođenja i optimizacije aplikacije za dugotrajno izvršavanje.



Slika 6. *Distribucija izmerenih vremena za pokretanje servisa*

Ovi rezultati pokazuju kako se performanse Node.js i .NET razlikuju u zavisnosti od specifičnog scenarija, pri čemu .NET nudi bolju efikasnost u operacijama koje zahtevaju obradu podataka, dok Node.js briljira u brzini pokretanja i upravljanju istovremenim konekcijama.

5. ZAKLJUČAK

Zaključak ove studije pruža dublji uvid u performanse aplikacija razvijenih u dva vodeća tehnološka okruženja – Node.js i .NET – i pokazuje kako se svaka od ovih tehnologija ponaša u specifičnim realnim scenarijima. Testovi su obuhvatili ključne aspekte performansi mikroservisnih aplikacija, uključujući interakciju sa bazama podataka, komunikaciju sa eksternim servisima i inicijalizaciju sistema.

Rezultati pokazuju da je .NET ostvario superiorne rezultate u oblastima kao što su brzina obrade podataka i stabilnost prilikom komunikacije sa eksternim servisima, što je naročito važno za aplikacije koje zahtevaju visoku pouzdanost i procesorsku efikasnost. .NET koristi *multi-thread* model, koji omogućava paralelnu obradu zadataka, čime postiže bolje rezultate u situacijama koje uključuju intenzivne operacije sa bazom podataka i zahtevaju visoku efikasnost u radu sa resursima. Ova karakteristika čini .NET pogodnim za aplikacije koje zahtevaju brzu obradu velikog broja zahteva i stabilnu integraciju sa složenim ekosistemima.

Sa druge strane, Node.js se istakao svojom efikasnošću kada je reč o brzom pokretanju aplikacija i upravljanju velikim brojem istovremenih konekcija, zahvaljujući svojoj asinhronoj prirodi i *event-driven* arhitekturi. Node.js koristi jedinstveni *event loop*, što omogućava lako skaliranje aplikacija i efikasno rukovanje brojnim zahtevima koji dolaze od klijenata. Ova tehnologija se posebno pokazala efikasnom u scenarijima gde je brzina pokretanja sistema ključna, što je čini pogodnom za aplikacije koje zahtevaju agilnost i brzu reakciju na zahteve korisnika.

Sveobuhvatna analiza pokazuje da oba tehnološka okruženja imaju svoje specifične prednosti, koje dolaze do izražaja u zavisnosti od zahteva projekta. .NET se pokazao kao bolji izbor za aplikacije koje zahtevaju stabilnost, sigurnost i efikasnost u radu sa bazama podataka i eksternim servisima, dok Node.js pruža prednosti u aplikacijama koje zahtevaju brzo startovanje i jednostavno skaliranje. Ova razlika u performansama ukazuje na to da izbor tehnologije ne treba da bude vođen isključivo preferencijama, već da mora biti pažljivo prilagođen specifičnim potrebama i ciljevima projekta.

Ovi rezultati mogu poslužiti kao vodič za inženjere softverskih sistema pri odlučivanju o tehnologijama koje će koristiti u budućim projektima. U zavisnosti od zahteva u pogledu performansi, brzine odziva, pouzdanosti i fleksibilnosti, Node.js i .NET pružaju različite prednosti.

Buduća istraživanja u ovoj oblasti mogla bi se fokusirati na dodatnu analizu skalabilnosti i energetske efikasnosti oba tehnološka okruženja.

Takođe, dodatno istraživanje moglo bi obuhvatiti evaluaciju sigurnosnih karakteristika svake platforme, kao i analizu ponašanja ovih okruženja u produkcionim uslovima pod velikim opterećenjem. Uvođenje novih tehnologija, poput *serverless* arhitektura i *edge computing-a*, u ova okruženja moglo bi doneti nove mogućnosti za optimizaciju i unapređenje performansi sistema, što predstavlja plodno tlo za dalja istraživanja.

6. LITERATURA

[1] Mario Casciaro, *"Node.js Design Patterns"*, 2016.

[2] Lakshmi Prasanna Chitra, Ravikanth Satapathy, *"Performance comparison and evaluation of Node.js and traditional web server"*, International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies, 2017.

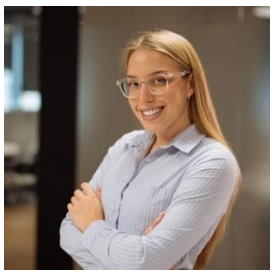
[3] Adam Freeman, "*Pro ASP.NET Core 6*", 2021.

[4] Iris Classon, Kenneth Yagen, *"Microservices in .NET Core: With Examples in Nancy"*, Apress, 2017.

[5] Sam Newman, *"Building Microservices: Designing Fine-Grained Systems"*, O'Reilly Media, 2021.

[6] David Mark Clements, *"Node Cookbook: Discover solutions, techniques, and best practices for server-side web development with Node.js 14"*, Packt Publishing, 2020.

Kratka biografija:



Teodora Ruvčeski rođena je u Novom Sadu 1999. godine. Osnovne akademske studije na Fakultetu tehničkih nauka Univerziteta u Novom Sadu upisala je 2018. godine. Diplomirala je 2022. godine na smeru Primenjeno softversko inženjerstvo i iste godine upisala master akademske studije na smeru Računarstvo i informatika.