SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

IZRADA SKALABILNE CHAT APLIKACIJE KOJA SE IZVRŠAVA NA KUBERNETES SUSTAVU

Ivo Kovačević

Split, rujan 2020.

SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Diplomski studij: **FESB**

Smjer/Usmjerenje: **Računarstvo**

Oznaka programa: 250

Akademska godina: 2019./2020.

Ime i prezime: **IVO KOVAČEVIĆ**

Broj indeksa: 744-2018

**ZADATAK DIPLOMSKOG RADA**

Naslov: **IZRADA SKALABILNE CHAT APLIKACIJE KOJA SE IZVRŠAVA NA KUBERNETES SUSTAVU**

Zadatak: Izučiti principe dizajna i izrade skalabilnih web aplikacija. Izučiti način korištenja Kubernetes sustava kao i prednosti koje donosi u rada sa skalabilnim aplikacijama. Izradom chat aplikacije demonstrirati način i prednosti korištenja Kubernetes sustava.

Prijava rada: 05.10.2020. (početak semestra u kojem se prijavljuje rad)

Rok za predaju rada: 05.02.2020. (deset dana prije završetka semestra u kojem je rad prijavljen)

Rad predan:

Predsjednik   
Odbora za diplomski rad: Mentor:

prof. dr. sc. Ime Prezime izv. prof. dr. sc. Marin Bugarić

**IZJAVA**

Ovom izjavom potvrđujem da sam završni rad s naslovom (Izrada skalabilne chat aplikacije koja se izvršava na Kubernetes sustavu) pod mentorstvom (prof. dr. sc. Marin Bugarić) pisao samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući naveo/la u završnom radu citirao/la sam i povezao/la s korištenim bibliografskim jedinicama.

Student

Ivo Kovačević

**SADRŽAJ**

[1 UVOD 1](#_Toc41387262)

[2 MOTIVACIJA ZA UVOĐENJE KUBERNETES SUSTAVA 2](#_Toc41387263)

[2.1 Prelazak na mikroservis arhitekturu 2](#_Toc41387264)

[2.2 Uvod u container tehnologiju 4](#_Toc41387265)

[2.3 Potreba za Kubernetes sustavom 4](#_Toc41387266)

[3 ZAKLJUČAK 5](#_Toc41387267)

[LITERATURA 6](#_Toc41387268)

[POPIS OZNAKA I KRATICA 7](#_Toc41387269)

[SAŽETAK 8](#_Toc41387270)

# UVOD

Novi trend razvijanja aplikacija su svakako mikroservisi. Mikroservis arhitektura se uveliko razlikuje od tradicionalnog razvoja aplikacija gdje je cijela aplikacija „zapakirana“ kao jedna velika cjelina. Takve aplikacije nazivaju se monolitne aplikacije.

Monolitan razvoj je i dalje vrlo raširen iako ima mnogo nedostataka. Samim tim što je aplikacija jedna velika cjelina gdje su svi „servisi“ strogo povezani, jako je teško nadograđivati i mijenjati aplikaciju pa su nove verzije aplikacija vrlo rijetke. Nakon što se takve aplikacije naprave, programeri aplikaciju proslijede Ops timu. Ops tim potom aplikaciju ručno migrira na „zdravi“ (engl. healthy) server. Ukoliko dođe do prekida rada servera, aplikaciju je potrebno opet ručno migrirati na drugi „zdravi“ server.

Mikroservis arhitektura je skup više nezavisnih, samostalnih, odvojenih servisa. Svaki servis se razvija zasebno, najčešće od strane manjeg tima unutar organizacije te se potom migrira (engl. deploy) i održava neovisno o ostalim servisima. Ovakva arhitektura donosi mnoge prednosti u odnosu na tradicionalni razvoj. Glavna prednost je svakako mogućnost učestalijeg mijenjanja pojedinih komponenti ovisno o poslovnim zahtjevima.

Kod velikih aplikacija mikroservis arhitektura može rezultirati velikim brojem zasebnih servisa. Kod kompanija kao što su Amazon i Netflix taj broj prelazi nekoliko tisuća. Upravljati s nekoliko tisuća zasebnih mikroservisa bi svakako bilo nemoguće da ne postoje alati koji nam olakšavaju i omogućavaju taj posao. Jedan od tih alata je i Kubernetes sustav koji će biti objašnjen u okviru ovog diplomskog rada.

# MOTIVACIJA ZA UVOĐENJE KUBERNETES SUSTAVA

## Prelazak na mikroservis arhitekturu

Monolitne aplikacije sastoje se od više strogo povezanih aplikacija ili servisa koji se razvijaju, migriraju i održavaju kao cjelina. Takva aplikacija je u biti jedan proces koji se vrti unutar servera. Za svaku i najmanju promjenu neke od komponenti, cijela aplikacija se treba iznova migrirat na server (engl. redeploy).   
Ovakva arhitektura predstavlja ograničenje u skaliranju aplikacije pri promjeni opterećenja. Postoje dva tipa skaliranja: vertikalno i horizontalno skaliranje. Vertikalno skaliranje podrazumijeva nadogradnju servera dodavanjem procesorske snage, memorije, povećanje frekvencije procesora itd. Horizontalno skaliranje odnosi se na dodavanje više servera koji su najčešće slabijih performansi nego kod vertikalnog skaliranja, ali je ključ u njihovom broju. Ti serveri imaju istu instancu aplikacije te se opterećenje raspoređuje između servera. Ukoliko je neki server u velikom opterećenju novi zahtjevi će se prebaciti na drugi, manje opterećen server.   
Monolitne aplikacije vrlo se lako vertikalno skaliraju. Zamislimo scenarij da posjedujemo Internet trgovinu koja prodaje kreme za sunčanje. Lako je zaključiti da će u periodu od jeseni do proljeća posjećenost naše Internet trgovine biti vrlo mala. Za takvo opterećenje će vrlo vjerojatno biti dovoljni prosječni serveri. Pri dolasku sunčanijih ljetnih mjeseci, za očekivati je da će posjećenost naše stranice naglo porasti. Moguć je scenariji da naš prosječni server neće biti u mogućnosti podnijeti nagli porast opterećenja. Jedno od rješenja bi bilo vertikalno skalirati našu aplikaciju dodavanjem procesorske snage i memorije. Ukoliko se radi o manjoj Internet trgovini ovo će vrlo vjerojatno riješiti problem. U ovoj priči radi se o velikoj Internet trgovini koja drži monopol prodaje krema za sunčanje u cijelom svijetu. Može se pomisliti: „Pa zašto nebi dodali još, još i još procesorske snage i memorije?“. Naime, vertikalno skaliranje ima svoja ograničenja. Glavno ograničenje svakako bi bila cijena. Snažni i moćni hardver je vrlo skup. Drugo ograničenje je proizvodnja i dostupnost hardvera na tržištu. Iako danas na tržište vrlo brzo izlaze nove i bolje verzije hardvera opet ćemo doći do točke gdje više nećemo moći dobiti bolji server nego što ga imamo. Može se naslutiti da smo dosegli vrhunac vertikalnog skaliranja i da moramo pronaći neko drugo rješenje. To bi bilo horizontalno skaliranje. Međutim, monolitne aplikacije je najčešće vrlo teško horizontalno skalirati jer su komponente strogo povezane. Ukoliko samo jedan dio monolitne aplikacije nije skalabilan, cijela aplikacije će biti neskalabilna. Uvjet za horizontalno skaliranje je neovisnost komponenti. Da bi ispunili taj preduvjet monolitnu aplikaciju moramo „razbiti“ na više neovisnih servisa gdje svaki servis predstavlja jedan nezavisni proces (Slika 2.1). Procesi mogu međusobno komunicirati. Komunikacija među procesima može se ostvariti velikim brojem protokola koji nisu strogo vezani za određeni programski jezik nego su implementirani gotovo u svakom programskom jeziku. To su HTTP, TCP, AMQP itd.

A screenshot of a cell phone

Description automatically generated

Slika 2.1 Prelazak s tradicionalnog razvoja na mikroservis arhitekturu

Budući da je svaki mikroservis neovisni proces, moguće ih je migrirati, razvijati i održavati neovisno o drugim mikroservisima. Promjena na jednom mikroservisu ne uzrokuje „redeploy“ ostalih mikroservisa. Posljedično tome, kao rezultat dobijemo mogućnost skaliranja svakog mikroservisa zasebno, neovisno o drugim mikroservisima (Slika 2.2). Na taj način možemo skalirati samo one mikroservise koji zahtjevaju više resursa, a ne cijelu aplikaciju.

A close up of a logo

Description automatically generated

Slika 2.2 Svaki mikroservis može biti skaliran zasebno

## Uvod u container tehnologiju

## Potreba za Kubernetes sustavom

# ZAKLJUČAK

# LITERATURA

# POPIS OZNAKA I KRATICA

Ops – Information-technology operations  
HTTP - HyperText Transfer Protocol  
TCP – Transmission Control Protocol  
AMQP – Advanced Message Queueing Protocol

# SAŽETAK