Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.5:339
DOI: https://doi.org/10.24867/10BE23Mijatov

TESTIRANJE SOFTVERA U OKRUŽENJU SIMULACIJE U REALNOM VREMENU

SOFTWARE TESTING IN REAL-TIME SIMULATION ENVIRONMENT

Vanja Mijatov, Branko Milosavljević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu se opisuju principi testiranja hardver u petlji i kontejnerizacije aplikacije. Opisuje sistem za testiranje softvera u okruženju simulacije u realnom vremenu koristeći specijalizovan sistem za testiranje po principu hardver u petlji pri čemu je softverski deo sistema kontejnerizovan korišćenjem Docker tehnologije.*

Ključne reči: *testiranje softvera, softver u realnom vremenu, hardver u petlji, kontejnerizacija, TyphoonTest, Docker*

Abstract – *Paper describes principles of hardware-in-the-loop testing and application containerization. It describes system for software testing in real-time simulation environment by using specialized system for hardware-in-the-loop testing where software part of the system is containerized by using Docker technology.*

Keywords: *software testing, real-time software, hardware-in-the-loop, containerization, TyphoonTest, Docker*

1. UVOD

Testiranje softvera je proces validacije ponašanja softverskog sistema. Testiranjem se otkrivaju potencijalni bagovi i proverava se pravilnost rada sistema u različitim situacijama, kako u trenucima normalnog rada, tako i u ekstremnim situacijama ili situacijama kada je sistem pod opterećenjem. Takođe, proverava se da li softverski sistem podržava očekivane i potrebne funkcionalnosti.

Kod sistema u realnom vremenu (engl. *real-time system*) testiranje rada je kompleksnije zato što osim što sistem treba da ostvari očekivan rezultat, taj rezultat mora biti ostvaren u određenom vremenskom periodu, odnosno postoje vremenska ograničenja u kojima sistem mora izvršiti posao.

Za testiranje ovakvih sistema najčešće je potrebno obezbediti posebno softversko i hardversko okruženje i posebne alate koji omogućavaju izvršavanje testova.

Simulacija sa hardverom u petlji je tip simulacije u realnom vremenu i način da se putem simulacija testira kompleksan sistem u realnom vremenu. Umesto da se rad gotovog sistema testira tako što se pusti u privremen rad i na taj način rizikuju potencijalne nepredviđene i opasne situacije kao i veliki troškovi usled potencijalnog oštećenja proizvoda ili okoline u kojoj radi, može se simulirati kako bi taj sistem radio u realnim uslovima i na taj način predvideti mnogi potencijalni problemi i nedostaci. Takođe, simulacije mogu pomoći i u fazi razvoja različitih električnih komponenti sistema i na taj način se mogu blagovremeno u fazi razvoja identifikovati problematični delovi sistema i ispraviti, što može predstavljati veliku uštedu finansijskih sredstava i vremena.

Kako je softver koji vrši simuliranje izuzetno kompleksan i treba da kompajlira modele kojima se definišu sistemi, njegovo ponašanje zavisi od konfiguracije i operativnog sistema mašine na kojoj je instaliran i na kojoj se koristi. Drugačija implementacija delova softvera da bi se prilagodio različitim mašinama može biti veliki i skup proces i gotovo je nemoguće podržati pravilan rad na apsolutno svakoj mašini. Stoga, rešenje za korišćenje softvera na različitim platformama bi bilo da se zapakuje u neki kontejner koji bi mogao da se jednako ili približno jednako uspešno koristi na različitim mašinama. Kontejnerizacija različitih softvera sa okruženjem za njihovo pravilno izvršavanje radi lakšeg korišćenja na različitim mašinama bez potrebe da se mašina dodatno konfiguriše postaje standardna praksa poslednjih nekoliko godina i sve više kompanija se odlučuje da svoj softver distribuira i na ovaj način.

Ovaj rad se bavi problematikom testiranja sistema u realnom vremenu i jednim načinom postavke sistema za testiranje ovakvih sistema pomoću specijalizovanog softvera. Takođe, opisuje način kontejnerizovanja specijalizovanog softvera za testiranje i njegovo okruženja, pri čemu se dobija rešenje koje je moguće koristiti na računarima sa različitim operativnim sistemima koji imaju podršku za korišćenje kontejnerizovanih aplikacija. Zatim, opisuje prednosti sistema, probleme koji mogu da se javljaju i njihova rešenja i moguća proširenja sistema. Konačno rešenje je validirano u sistemu koji omogućava automatizovanje testiranje različitih električnih kola i električnih sistema, pri čemu se kao rezultat ovog procesa dobija izveštaj i omogućen je pregled istorijata testiranja, kao i pretraživanje rezultata prethodno izvršenih testova.

2. KONCEPTI TESTIRANJA SISTEMA U REALNOM VREMENU

2.1. Hardver u petlji

Hardver u petlji (engl. *Hardware-in-the-loop,* skraćeno *HIL*) je način ili tehnika za testiranje sistema u realnom vremenu tako što se korišćenjem posebnih hardverskih komponenti simulira okruženje realnog sistema i na taj način testira njegovo ponašanje u određenim situacijama. Vršenjem simulacija u različitim situacijama i scenarijima korišćenja moguće izvršiti veoma veliki broj testova za

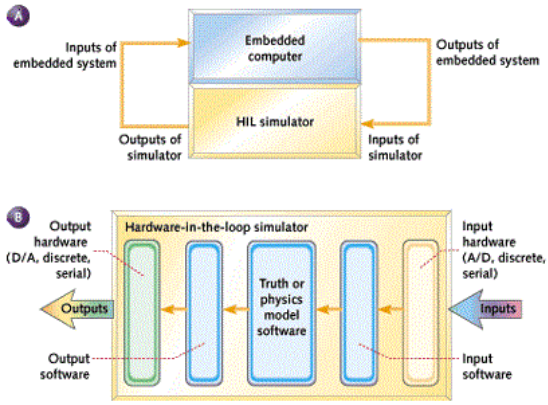
---

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

kratko vreme bez većih troškova što znači da nije potrebno vršiti fizičko testiranje sistema u realnim uslovima.

Autor rada [1] opisuje hardver u petlji simulator kao uređaj koji je u stanju učiniti da sistem misli da se nalazi u okruženju u realnom svetu slanjem i primanjem signala kao u realnom svetu te na taj način proverava ponašanje sistema u stvarnom okruženju. Autori rada [2] gledaju na hardver u petlji kao simulator koji predstavlja sistem kojim se delovi testiranog podsistema zamene virtuelnim reprezentacijama i na taj način se testira ponašanje ostatka sistema. U radu [3] autor opisuje koncept hardvera u petlji kao metoda koja koristi koncept simuliranja modela procesa i hardvera, pri čemu model simulacije obezbeđuje potrebne signale koji se transformišu i dostavljaju kontrolerima, koji zatim generišu drugi tip signala i na taj način se čini da se hardverske komponente ponašaju što bliže pravim uređajima u realnom sistemu. Sistem za testiranje sistema u realnom vremenu po principu hardvera u petlji koji je korišćen za potrebe ovog rada sastoji se od računara na kom se nalazi softver koji interpretira modele i podržava izvršavanje testova i uređajem za testiranje sistema u realnom vremenu sa kojim je računar povezan i sa kojim može da komunicira.

Slika 1 reprezentuje tipičan primer sistema za testiranje po principu hardver u petlji, odnosno prikazuje komponente simulatora. A segment slike prikazuje računar povezan sa simulatorom hardvera u petlji koji razmenjuju informacije. B segment slike prikazuje unutrašnju strukturu simulatora i tok signala kroz njega. Na osnovu slike i interpretacija ovog tipa sistema može se uvideti simulator dobija kao ulaz različite vrednosti signala od računara i vraća računaru signale kao odgovor, odnosno daje informaciju o odzivu i ponašanju sistema u toku simulacija.



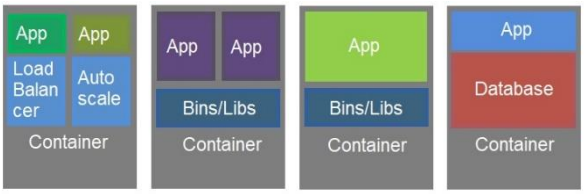
Slika 1. *Primer sistema za testiranje po principu hardver u petlji [1]*

2.2. Kontejnerizacija aplikacije

Kontejnerizacija aplikacije je proces virtuelizacije okruženja i aplikacije u kom se ona izvršava. Na ovaj način dobija se na lakšem pakovanju, instalaciji, distribuciji i korišćenju aplikacije. Krajnji korisnik stoga ne mora da podešava svoj računar i prilagođava okruženje aplikaciji koju želi da koristi, već pomoću softvera koji radi sa kontejnerima koristi aplikaciju zapakovanu u kontejner sa čitavim okruženjem neophodnim za njeno izvršavanje. Kako navodi autor rada [4], kontejnerizacijom se dobija na interoperabilnosti sa ciljem da se distribuira softver. Takođe, ističe jednostavnost kontejnera, portabilnost izvršavanja, mogućnost povezivanja kontejnera i mogućnost razvoja, testiranja i plasiranja na veliki broj servera.

Kako je na jednoj mašini moguće pokrenuti više različitih kontejnera ili više instanci identičnih kontejnera pri čemu se oni nezavisno izvršavaju, preporučljivo je više aplikacija pokrenuti u više različitih kontejnera. U slučaju da ove aplikacije treba da interaguju i razmenjuju informacije ili poruke, potrebno je obezbediti način komunikacije između njih. Slika 2 demonstrira primer jednog sistema zasnovanog na kontejnerima. U različitim kontejnerima se nalaze aplikacije sa različitom namenom koje obavljaju svoje poslove, a po potrebi mogu da međusobno komuniciraju i svoje funkcionalnosti koriste za ostvarenje funkcionalnosti drugih aplikacija u drugim kontejnerima.

U kontejner je moguće spakovati softver za testiranje sistema u realnom vremenu po principu hardver u petlji ako je moguće obezbediti sve potrebne biblioteke i alate od kojih zavisi njegovo izvršavanje. Takođe, potrebno je omogućiti aplikaciji unutar kontejnera da pristupa priključnim mestima računara domaćina da bi mogla da interaguje sa hardverom koji učestvuje u simulacijama. Na ovaj način bi se dobilo simuliranje sistema kao da je softver na tradicionalan način instaliran na računar, a samo korišćenje bi bilo daleko lakše i ne bi trebalo da se izvrši značajnije podešavanje okruženja osim da se obezbedi adekvatna tehnologija za rukovanje kontejnerom.



Slika 2. *Arhitektura sistema zasnovanog na kontejnerima [4]*

3. MODEL SISTEMA

3.1. Komponente sistema

Sistem za testiranje softvera u okruženju simulacije u realnom vremenu se sastoji od platforme za testiranje koja se nalazi na računaru i simulatora koji radi po principu hardvera u petlji koji je povezan sa računarom. Platforma za testiranje omogućava testiranje sistema u realnom vremenu. Zapakovana je u Docker kontejner koji ima pristup simulatoru. Na ovaj način nema potrebe da se značajno podešava okruženje računara, već je dovoljno samo obezbediti tehnologiju za upravljanje kontejnerom te nema potrebe za manuelnom instalacijom platforme.

3.2. Platforma za testiranje softvera

Platforma za testiranje softvera je specijalizovan softver sa velikim brojem alata što čini ovaj softver zahtevnim sa stanovišta performansi računara. U saradnji sa eksternom hardverskom jedinicom, odnosno simulatorom, omogućava simulaciju sistema u realnom vremenu. U skladu sa principom testiranja hardver u petlji, slanjem različitih signala hardverskom modulu od strane aplikacije vrši se simuliranje sistema tako što simulator reaguje na ulazne signale onako kako bi reagovao pravi sistem (softver) u realnom svetu i vraća signalne izlaze kao rezultat koji se evaluiraju i ako su u skladu sa očekivanim vrednostima

definisanim u testu, test će biti uspešan. Na kraju izvršavanja testova, kao rezultat se dobije izveštaj generisan od strane platforme koji demonstrira kako je tekla simulacija, koji su bili koraci u okviru nje i na koji način se simulirani sistem ponašao.

Platforma je kontejnerizovana korišćenjem Docker tehnologije i ima mogućnost korišćenja potpunih resursa računara na kom se pokreće. Takođe, potrebno je da ima pristup priključnim mestima računara da bi mogla aplikacija da komunicira sa simulatorom. Kako se testiranje često automatizuje, u kontejner je za tu potrebu dodata i aplikacija koja automatizuje testiranje u okviru master-agent arhitekture. Ideja master-agent arhitekture (u literaturi ranije poznato i kao *master-slave*) jeste da master definiše zadatak koji agent izvršava.

4. IMPLEMENTACIJA SISTEMA

4.1. Okruženje za testiranje u realnom vremenu

Typhoon HIL kontrolni centar (engl. *Typhoon HIL Control Center*) je aplikacija koja predstavlja skup alata za dizajniranje modela sistema u realnom vremenu, definisanje simulacija rada sistema, testiranje dizajniranog sistema, automatizovano testiranje, pregled izveštaja testiranja itd. Osim softverskih alata koje pruža kontrolni centar, korišćen je i „Typhoon HIL602+“ uređaj koji predstavlja simulator rada električnih sistema u realnom vremenu koji omogućava razvoj, optimizaciju, testiranja i proveru kvaliteta elektronskih sistema.

TyphoonTest je biblioteka koja omogućava izvršavanje testova pisanih u Python programskom jeziku. Može da izvršava testove pisane za testiranje sistema u realnom vremenu dizajniranih pomoću alata kontrolnog centra i generiše datoteke sa rezultatima izvršenih testova.

4.2. Kontejnerizacija aplikacije pomoću Docker tehnologije

Docker je najpopularnija tehnologija za kontejnerizaciju aplikacija u ovom trenutku. Kako razvojni tim ove tehnologije ističe [5], kontejneri u Docker-u su jedinice softvera u koje se zapakuju kod i njegove zavisnosti i koji može da se koristi u različitim okruženjima, a slike kontejnera su mali, nezavisni i izvršivi paketi koji sadrže sve što je potrebno za pravilan rad aplikacije. Na osnovu slike se instanciraju kontejneri i moguće je instancirati više kontejnera na osnovu jedne slike na jednom ili više različitih mašina. Rešenje koje je opisano ovim radom je bazirano na Linux CentOS 8 operativnom sistemu (korišćen kao osnovna slika) i instaliranim većim brojem standardnih biblioteka za Linux operativni sistem.

4.3. Platforma za testiranje softvera

U rešenju je Typhoon HIL kontrolni centar instaliran unutar Docker kontejnera. Kontejener sadrži potrebne sistemske biblioteke i Python koji je kompajliran iz izvornih datoteka, kao i potrebne biblioteke za testiranje. Na osnovu kontejnera je napravljena slika.

Da bi se pokretanjem kontejnera zasnovanim na ovoj slici moglo vršiti testira-nje, potrebno je koristiti Docker *volume* kojim se konte-jneru da pristup datoteci licence i navesti da kontejner može da koristi priključna mesta mašine domaćina da bi komunikacija sa simulatorom bila moguća.

Kako se testiranje učestalo izvršava i testovi se često dorađuju, korišćenje dobijenog rešenja tako što se svaki put testovi manuelno kopiraju unutar kontejnera i ručno pokreće testiranje, a nakon čega se rezultati manuelno izvlače iz kontejnera da bi se mogli analizirati, nije dobro rešenje sa stanovišta zadovoljstva korisnika. Stoga, rešenje je potrebno automatizovati da bi se lakše koristilo i korisnici bili zadovoljniji. Korišćenjem Jenkins aplikacije koja se uz kontrolni centar zapakuje unutar kontejnera, može se postići automatizacija. Rešenje je unapređeno u dva smera tako da su dobijene dve slike koje pružaju različite funkcionalnosti. Prva slika sadrži Jenkins agentsku aplikaciju koja omogućava ponašanje kontejnera kao agenta kojem se daju naredbe od strane mastera. Druga slika osim agentske aplikacije sadrži i celu Jenkins aplikaciju koja omogućava definisanje zadataka koji se izvršavaju na toj mašini. Prilikom pokretanja kontejnera na osnovu ove dve slike potrebno je navesti podatke potrebne za povezivanje kontejnera sa ulogom agenta sa master aplikacijom u vidu promenljivih okruženja.

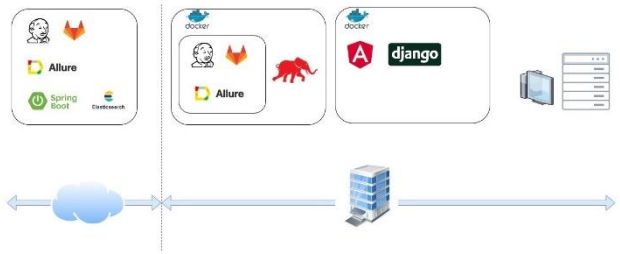
5. EVALUACIJA REZULTATA

Sistem za testiranje softvera u okruženju simulacije u realnom vremenu je pokazao zadovoljavajuće rezultate. Trajanje testiranja se ne razlikuje značajno u odnosu na trajanje pokretanja istih testova na mašini domaćina iz razloga što Docker kontejner ima mogućnost korišćenja svih dostupnih resursa mašine domaćina. Takođe, proširenjem opisanog sistema i automatizovanjem testiranja se dobija velika ušteda vremena za korisnike sistema čime se dobija na njihovom zadovoljstvu rada korišćenog sistema. Distribucija i upotreba dobijenog rešenja u vidu Docker slike je daleko jednostavnije od tradicionalnog pristupa i može da radi na mašinama sa različitim operativnim sistemima koji podržavaju Docker tehnologiju. Kako se koristi Pytest radni okvir za testiranje, moguće je integrisati dodatne priključke napisane u Python programskom jeziku koji mogu raditi uz TyphoonTest i obavljati programski definisane zadatke u okviru testiranja.

Ono što predstavlja manu dobijenog rešenja jeste količina memorije koju zauzima slika usled ogromnog prostora na disku koji zahteva Typhoon HIL kontrolni centar, zbog čega i objavljivanje slika na repozitorijum i preuzimanje slike dugo traje. Kako se za potrebe testiranja ne koristi veći broj funkcionalnosti koje pruža kontrolni centar, particionisanjem softvera na veći broj modula tako da se mogu koristiti isključivo moduli sa potrebnim funkcionalnostima bi se dobilo na smanjenju trajanja izgradnje slike kao i na smanjenju konačne veličine slike.

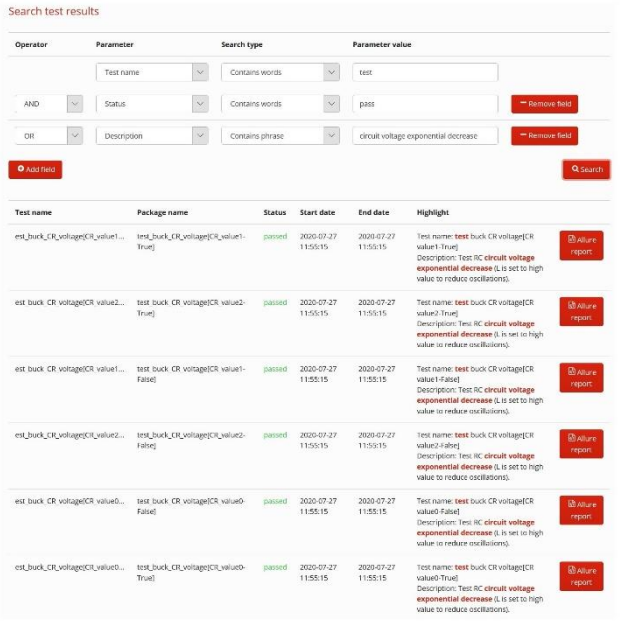
Druga mana rešenja proizilazi iz nedostataka Jenkins aplikacije, odnosno problema sa slanjem i čuvanjem velikih izveštaja i nemogućnošću pretraživanja dobijenih izveštaja. Nakon testiranja, generiše se izveštaj koji se šalje Jenkins master aplikaciji koji u slučaju izvršavanja velikog broja testova može zauzeti značajnu količinu memorije na disku, a slanje izveštaja preko mreže između agenta i mastera može dugo da traje zbog njegove veličine. Primljeni izveštaj Jenkins aplikacija čuva i omogućava pregledanje što nije problem za manji broj izveštaja. U slučaju čestih testiranja velikog obima, Jenkins aplikacija ne odgovara u potpunosti potrebama korisnika.

Sa ciljem rešavanja drugog problema koji prouzrokuju nedostaci Jenkins aplikacije, sistem za automatizovano testiranje je proširen aplikacijom čija je uloga da čuva testove i omogućava njihovo pregledanje i pretraživanje, a koja se nalazi u oblaku. U drugom kontejneru koji se pokreće na istoj mašini kao i kontejner koji vrši testiranje, nalazi se Django aplikacija koja nadgleda testiranje. Rezultate testiranja ova aplikacija šalje Spring Boot aplikaciji koja se nalazi u oblaku i koja čuva rezultate testiranja i omogućava njihovo pretvaranje u izveštaj čime se rešava jedna od mana Jenkins aplikacije. Pored Django aplikacije, pomenuti kontejner sadrži i Angular aplikaciju kojom je definisan korisnički interfejs. Spring Boot aplikacija koja se nalazi u oblaku komunicira sa Elasticsearch platformom koja se takođe nalazi u oblaku i koja omogućava čuvanje izvršenih testova koji se predstavljaju u vidu zasebnih dokumenata i pretraživanje po više različitih parametara. Model proširenog sistema prikazuje Slika 3.



Slika 3. *Sistem za automatizovano testiranje i evaluaciju rezultata testiranja*

Slika 4 prikazuje formu za definisanje upita za pretragu izvršenih testova i tabelu sa rezultatima pretrage. Za svaki parametar upita nakon prvog parametra moguće je definisati logički operator i/ili u odnosu na prethodni parametar ili skup parametara. Tabela rezultata prikazuje maksimalno tri stotine najbolje rangiranih rezultata te u slučaju da korisnik traži izvršeni test po parametru koji se često pojavljuje u polju koje je izabrano kao parametar pretrage ili je za parametar pretrage unesena reč visoke frekvencije, potrebno je da se preciznije definiše upit kako bi se pronašao rezultat koji je korisnik priželjkivao.



Slika 4. *Stranica za pretragu izvršenih testova*

6. ZAKLJUČAK

U ovom radu dat je opis sistema za testiranje softvera u okruženju simulacije u realnom vremenu. Opisane su komponente sistema, principi u skladu sa kojima one rade i osobine rešenja sa njegovim proširenjima.

Sistem za testiranje softvera u okruženju simulacije u realnom vremenu omogućava testiranje električnih sistema i dozvoljava integrisanje sa drugim alatima radi promene standardnog ponašanja sistema kao i davanje višestruke uloge sistemu. Specijalizovani softver koji omogućava testiranje je kontejnerizovan korišćenjem Docker tehnologije tako da su olakšani korišćenje i distribucija softvera. Testiranje se može automatizovati tako što se vrši integracija sa Jenkins aplikacijom koja se takođe nalazi u kontejneru i koja zadaje komande za pokretanje testiranja koje izvršava specijalizovani softver, a sama automatizacija radi prema master-agent arhitekturi.

Moguće je proširiti sistem sa drugim aplikacijama koje omogućavaju evaluaciju i pretraživanje rezultata testiranja. Dalje istraživanje bi bilo usmereno u pravcu integrisanja sistema za testiranje sa nekim od popularnih oblak kompjuting veb servisa koji nudi veliki izbor alata što bi smanjilo potrebu za korišćenjem dosta različitih alata i tehnologija za potrebe automatizacije testiranja i evaluacije dobijenih rezultata izvršenih testova.

7. LITERATURA

[1] Gomez, Martin. "Hardware-in-the-loop simulation." *Embedded Systems Programming* 14.13 (2001): 38-49.

[2] Fathy, Hosam K., et al. "Review of hardware-in-the-loop simulation and its prospects in the automotive area." *Modeling and simulation for military applications*. Vol. 6228. International Society for Optics and Photonics, 2006.

[3] Grega, Wojciech. "Hardware-in-the-loop simulation and its application in control education." *FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education. Conference Proceedings (IEEE Cat. No. 99CH37011*. Vol. 2. IEEE, 1999.

[4] Pahl, Claus. "Containerization and the paas cloud." *IEEE Cloud Computing* 2.3 (2015): 24-31.

[5] https://www.docker.com/resources/what-container (pristupljeno u septembru 2020.)

Kratka biografija:

Vanja Mijatov rođen je u Novom Sadu 1996. god. 2015. god. se upisao na osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu na smer Softversko inženjerstvo i informacione tehnologije. Diplomirao je na osnovnim akademskim studijama 2019. godine i iste godine je upisao master akademske studije na Fakultetu tehničkih nauka u Novom Sadu, smer Softversko inženjerstvo i informacione tehnologije, na usmerenju Elektronsko poslovanje.
kontakt: vanja.mijatov@uns.ac.rs