Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/15BE09Ercegovac

TESTIRANJE JEZIKA SPECIFIČNOG ZA DOMEN ENERGETSKE RAZMENE BEZ POSREDNIKA BAZIRANOG NA JETBRAINS MPS MBDDR I KERNELF SPECIFIKACIJI

TESTING OF A DOMAIN SPECIFIC LANGUAGE FOR P2P ENERGY TRADING BASED ON JETBRAINS MPS MBDDR AND KERNELF SPECIFICATIONS

Marko Ercegovac, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu će biti predstavljene mogućnosti primene KernelF jezika na pravljenje i testiranje DSL-a za domen P2P energy trading-a.*

Abstract – *The paper presents application of the KernelF language to the creation and testing of DSL (Domain-specific language) for the domain of P2P energy trading.*

Ključne reči: *domenski jezici, blokčejn, energetska razmena, pametni ugovori, KernelF, MPS.*

Keywords: *domain specific lanugage, blockchain, energy P2P trading, KernelF, MPS.*

1. UVOD

U ovom radu biće predstavljeno testiranje jezika specifičnog za domen energetske razmene bez posrednika implementacijom pametnih ugovora (Smart Contract-a) i korišćenjem interpretera uz podršku MPS (MethaProgramingSystem) okruženja mbeddr platforme i KernelF *embeddable* jezika. U daljem izlaganju, umesto pojma „energetska razmena bez korisnika" biće korišćen izraz "*P2P Energy trading* ".

Smart contract ili pametni ugovor je računarski program ili transakcioni protokol koji se izvršava automatski u skladu sa pravilima opisanim u sadržaju ugovora [1]. Pametni ugovori se izvršavaju na *block chain*-u koji predstavlja distribuiranu bazu podataka koja je transparentna svim korisnicima sistema zahvaljujući decentralizaciji sistema i kriptografiji [2]. Jedan od najpoznatijih *blockchain*-ova je Etherium *blockchain* koji predstavlja decentralizovanu mrežu koja nije pod kontrolom nijedne centralne organizacije i pomoću nje i jezika Solidity koji komunicira direktno sa Etherium *blockchain*-om moguće je pravljenje decentralizovanih aplikacija. Pisanje dobrih pametnih ugovora predstavlja izazov i programeri moraju detaljno testirati pametne ugovore pre njihove distribucije. Zbog sve većeg razvoja decentralizovanih aplikacija, neophodno je omogućiti automatizaciju razvoja i pisanja pametnih ugovora i povezati domenskog eksperta i programera.

U ovom radu će biti ukratko predstavljen domenski jezik sa interpreterom za opisivanje pametnih ugovora i automatski testovi koji su razvijeni za podršku daljeg razvoja i testiranja jezika. DSL (*Domain specific language* – domenski jezik) je programski jezik koji nudi softverska rešenja u nekom određenom domenu. Domenskim jezicima se mogu smatrati i saobraćajni znakovi ili notni sistem u muzici. Domenski jezici postaju sve popularniji zbog direktne povezanosti programera i domenskog eksperta. Domenskim ekspertima je omogućeno da povećaju svoju produktivnost i do 10 puta.

KernelF je funkcionalan jezik izgrađen na osnvu MPS-a. Dizajniran je da bude proširiv i ugradiv kao podrška jezgru domenskog jezika. KernelF je korišćen u širokom spektru jezika uključujući domenske jezike u oblasti medicine, finansija, obračuna plata, pametnih ugovora [3]. Jedni od najpoznatijih projekata izgrađenih na jezgru KernelF-a su Voluntis Healtcare i DATEV Payroll aplikacije. Voluntis je francusko-američka firma koja je napravila Healtcare platformu za lečenje personalizovanom digitalnom terapijom [4]. Koristeći algoritme, kako bi pronašli pravu dozu za pacijenta, platforma koristi već postojeće medicinske algoritme koji su se dobro pokazali u praksi. Aplikacija funkcioniše tako što telefon prikuplja sve podatke vezane za pacijenta, koje analizira i na osnovu medicinskih algoritama formira terapiju koju može da vidi lekarski tim i u svakom trenutku da kontaktira korisnika aplikacije. Projekat je nastao kao *startup* izgrađen na jezgru KernelF jezika. DATEV je registrovano društvo koje je pre svega pružalac tehničkih informacionih usluga za poreze, računovodstvo i advokate. U početku je bio dobavljač usluga, a sada softver direktno pruža usluge krajnjim korisnicima. Politika DATEV-a se zasniva u pružanju usluga na poreskom tržištu.

2. ENERGETSKI P2P TRADING

P2P (*Peer to Peer Service*) predstavlja decentralizovanu platformu gde dva pojedinca uzajamno interaguju bez prisustva trećeg lica. Umesto toga prodavac i kupac vrše transakcije direktno jedan ka drugome preko P2P servisa [5].

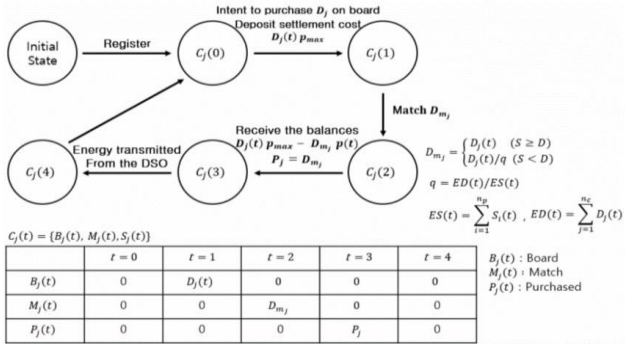
Osnovni koncepti energetske P2P razmene su dinamičko izračunavanje cene, DSO (*distribution system operator*) i korisnici koji učestvuju u razmeni energije (*Consumers*, *Prosumers*). Sa povećanim razvojem čiste (zelene) energije tradicionalni potrošači (*Consumers*) postaju proizvođači-potrošači (*Prosumers*) koji, koristeći foto-

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila prof. dr Gordana Milosavljević.

ćelije ili energiju vetra, mogu da generišu energiju, skladište je i prodaju višak energije koju nisu potrošili. Veliki problem predstavljaju zakoni koji još uvek nisu usklađeni u odnosu na visok razvoj i potražnju za zelenom energijom. Sam proces razmene i prodaje energije ne može da bude siguran i pouzdan bez nekog operatera od poverenja kao što je DSO [6]. Svaki od učesnika u energetskoj razmeni mora proći kroz nekoliko stanja. To stanje isključivo zavisi da li je korisnik *Consumer* (slika 1) ili *Prosumer* (slika 2).

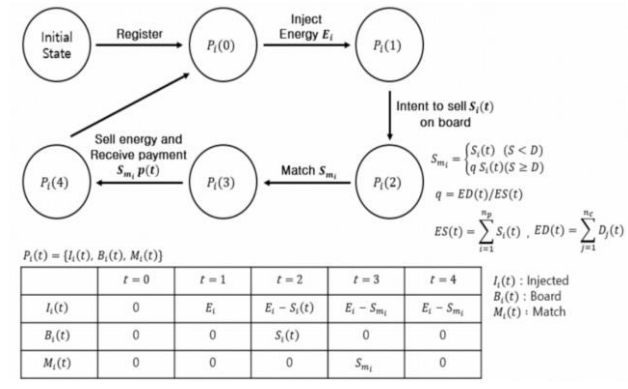
Na samom početku energetske razmene korisnik se registruje u sistem. Nakon toga prelazi u stanje *injected* (u tom stanju korisnik odlučuje koliko energije želi da proda), a zatim prelazi u stanje *on board* u koje objavljuje prodaju.

Ukoliko se nađe potencijalni *Consumer* koji kupuje tu količinu energije, prelazi u stanje *match* (upareno stanje) nakon kojeg prodaje energiju i dobija isplatu u odnosu na količinu prodate energije. Nakon uspešno obavljene energetske razmene proces se završava.



	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$
$B_j(t)$	0	$D_j(t)$	0	0	0
$M_j(t)$	0	0	D_{m_j}	0	0
$P_j(t)$	0	0	0	P_j	0

Slika 1. *Prikaz dijagrama stanja consumer-a u toku energetske razmene [6]*



	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$
$I_i(t)$	0	E_i	$E_i - S_i(t)$	$E_i - S_{m_i}$	$E_i - S_{m_i}$
$B_i(t)$	0	0	$S_i(t)$	0	0
$M_i(t)$	0	0	0	S_{m_i}	0

Slika 2. *Prikaz dijagrama stanja prosumer-a u toku energetske razmene [6]*

Bilo bi nezgodno u svakom mometu i za svaku pojedinačnu razmenu da potrošači i potrošači-proizvođači ponude ili traže za svaki period određenu vrednost. Da bi ovo izbegli, u sistemu se određuje pojedinačna cena [6]. Pojedinačna cena se određuje kao funkcija ukupne potražnje i ukupne ponude i jedna takva cena se koristi u toku procesa razmene.

Na samom početku procesa razmene označićemo totalnu ponudu (*supply*) sa ES(t) i totalnu potražnju (*demand*) sa ED(t) kao što je predstavljeno u jednačinama (1) i (2) [6].

$$ES(t) = \sum_{i=1}^{n_p} S_i(t)\ (S_i \geq 0) \quad (1)$$

$$ED(t) = \sum_{j=1}^{n_c} D_j(t)\ \left(D_j \geq 0\right) \quad (2)$$

$S_i(t)$ totalna ponuda od *prosumer*-a, $D_j(t)$ totalna potražnja od *consumer*-a i n_p broj *prosumer*-a i n_c broj *consumer*-a [6]. U formulama (3) i (4) su prikazani odnos R(t) i razlika D(t) između totalne potražnje (*supply*) i totalne ponude (*demand*) [6].

$$R(t) = \frac{ED(t)}{ES(t)} \quad (3)$$

$$D(t) = ED(t) - ES(t) \quad (4)$$

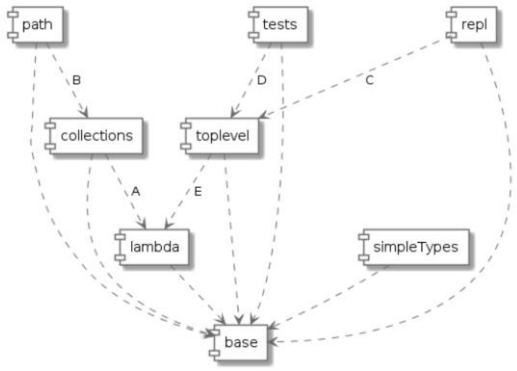
Chekired, Khoukhi and Mouftah [7] su predložili dinamičko izračunavanje cene korišćenjem R(t) i D(t) što je iskorišćeno u radu i na osnovu kog je izneta fukcija za dinamičko izračunavanje cene (5).

$$p(t) = \frac{2}{\pi}(p_{con}) \cdot tan^{-1}((lnR(t))^k) \ + p_{balance} \quad (5)$$

$p_{balance}$ je cena kada je totalna potražnja jednaka totalnoj ponudi to jest kada je R(t) = 1. p_{con} određuje opseg cene, dok koeficijent k „kontroliše“ opseg cene [6]. Formula postiže stabilnu cenu energije koja se ne menja puno u odnosu na ponudu i potražnju.

3. OSNOVNI KONCEPTI JEZIKA

KernelF je nastao kao potreba da se implementira jezik u MPS-u koji ima predefinisane principe: tipove podataka, promenljive, funkcije itd. Sam jezik KernelF je izgrađen od nekoliko manjih jezika koji su predstavljeni konceptima, što se može videti na slici 3.



Slika 3. *Kolekcija jezika u MPS-u od kojih se KernelF sastoji [8]*

KernelF je moguće proširiti drugim jezicima u MPS-u, takođe moguće je izgraditi DSL samo na osnovu KernelF-a. Jezik je izvršiv u javi i to preko javine virtualne mašine. KernelF predstavlja grupu različitih jezika implementiranih u MPS-u. Osnovni koncepti KernelF-a korišćeni u ovom radu su: *simple types*, *enumerations*, *state machines*, *functions*, *boxes*, *transactions* i *unit tests*. Postoje tri osnovna tipa koja koristi KernelF, a to su: *boolean*, *number* i *string*. *String* tip podržava interpolaciju

(konkatenaciju) *string*-ova, dok je *number* poseban tip koji sadrži opseg i preciznost. Ukoliko broj nije u dobrom opsegu program prijavljuje greške. Osnovne operacije u KernelF-u su unarne i binarne i koristi se infiksna notacija. KernelF podržava kolekcije i to liste, setove i mape. Tip tag je posebna vrsta tipa u KernelF-u na osnovu kog se može implementirati *custom* tip neke promenljive. Enumeracije koje podržava KernelF su: regularne i *valued flavors*. Enumeracije su potrebne za opisivanje stanja (*register*, *injected*, *on board*, *matched*, *purchased*).

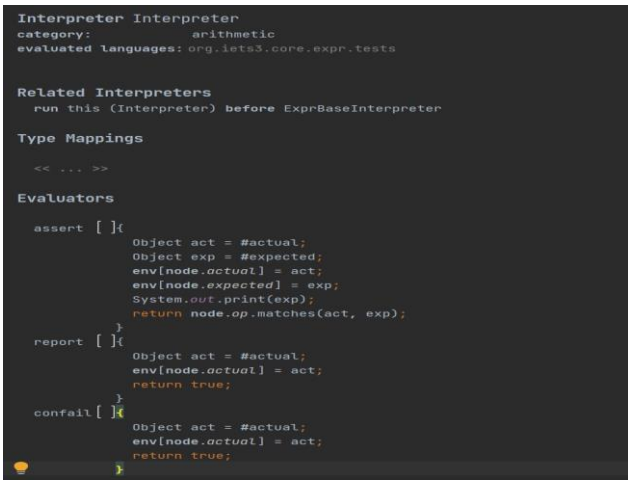
Konačni automati (*State machines*) su najbitniji koncept korišćen u ovom radu. Da bi se opisala pomenuta stanja *Prosumer*-a i *Consumer*-a koriste se konačni automati koje predstavljaju promenljivu (*mutable*) strukturu podataka. Konačni automati sadrže stanja, događaje i varijable i definisani su za jednu virtualnu mašinu. Događaje je moguće pozivati samo iz stanja u kome su ti događaji definisani. Svaki konačni automat ima inicijalno stanje sa inicijalnim vrednostima.

Funkcije u KernelF-u sadrže ime, listu argumenata, opcionu povratnu vrednost i telo funkcije. Takođe, funkcije pamte efekte i KernelF uvodi koncepte R-*read*, RM-*read/modify* koji omogućavaju na neki način kontrolu pristupa nad funkcijama. U koncepte KernelF-a uveden je tip nepromenljivog (*immutable*) podatka kao što je *boxes* koji nakon kreiranja nije moguće izmeniti. Korišćen je i koncept transakcija koje iplementira KernelF. Transakcioni blok je isti kao i obični blok ali ako nešto „pukne“ prilikom izvršavanja, sve izmene nad podacima se ponište unutar bloka. Testiranje je izuzetno bitno prilikom razvoja DSL-a. Testovi pomažu da se napiše dobar kod i da se proveri funkcionisanje koda. Testovi se sastoje od imena testa i različitih *elemenata* kao što su: *assertion, confail, report*. Klasično poređenje se postiže sa elementom *assert* koji poredi vrednost koju očekujemo sa dobijenom vrednošću. *Confail* proverava da li je test pukao (*fail*). *Report* vraća vrednost i tip elementa koji je testiran.

4. INSTALACIJA I KONFIGURACIJA

U ovom radu opisana je detaljna instalacija MPS-a sa sajta https://www.jetbrains.com/mps/ kao i upustvo kako instalirati mbeddr i IETS3 sa https://build.mbeddr.com. Takođe je opisan proces instalacije i konfiguracije KernelF-a. IETS3 predstavlja trenutnu verziju KernelF-a i stalno je u daljem razvoju i na sajtu https://build.mbeddr.com moguće je videti razne repozitorijume i različite artefakte. Da bi se implemetirao mbbeddr neophodno je sa glavnog repozitorijuma skinuti sa *master* grane na putanji mbbeddr/Main/Platform/master *zip* folder. Za IETS3 neophodno je skinuti sa repozitorijuma IETS3 Open Source/Build IETS3 opensource/master *zip* folder.
Preporučljivo je koristiti najnoviju verziju MPS-a, kao i najnovije verzije sa pomenutih repozitorija. Nakon uspešene instalacije *plugins*-a MPS JetBrains će prikazati i učitati prilikom pokretanja pomenute *plugin*-e.

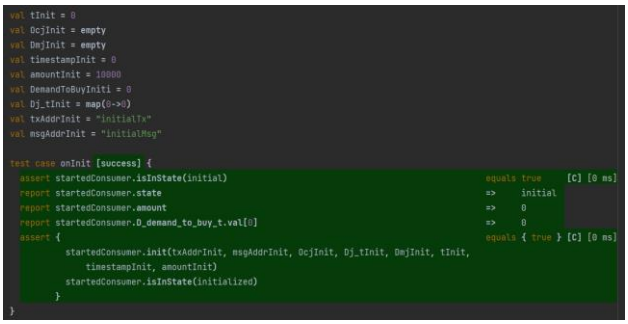
Nakon pokretanja MPS-a pojavi se prozor u kome se bira da li se pravi *solution* ili *language* projekat. Pošto je KernelF izgrađen kao predefinisani language projekat, potrebno je izabrati *solution* (rešenje) i nakon toga importovati pomenute koncepte. Moguće je napraviti i prazan projekat (*empty*), a naknadno da se izabere *solution* projekat. Da bi se moglo uopšte pisati u KernelF-u koriste se *Library* fajlovi koji se pozivaju tako što se implementira predefinisani koncept org.iets3.core.expr.toplevel, koje je za implementaciju testova neophodno importovati. Da bi testovi uopšte funkcionisali neophodno je implementirati interpreter testova (slika 4) koji je u ovom radu implementiran i opisana je detaljno njegova konfiguracija. Interpreter testova se pokreće pomoću Ctrl + Alt + Enter.



Slika 4. *Prikaz implementacije interpretera za testove u KernelF-u*

5. IMPLEMENTACIJA REŠENJA

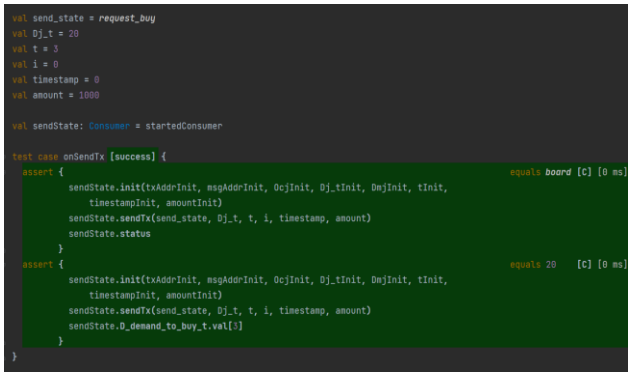
Celokupna implementacija jezika je prikazana u delovima i objašnjenjima šta koji deo koda radi [8] . Celokupan kod rešenja nije prikazan ali je prikazano njegovo uspešno izvršavanje u sklopu *unit* testova. Rešenje se sastoji iz nekoliko *Library* fajlova koji su međusobno povezani i moguće je koristiti definisane koncepte iz različitih *Library* fajlova. Opisani su u potpunosti Prosumeri i Consumeri sa njihovim stanjima kao i događaji koji se izvršavaju kad se pomenuti korisnici nalaze u različitim stanjima.



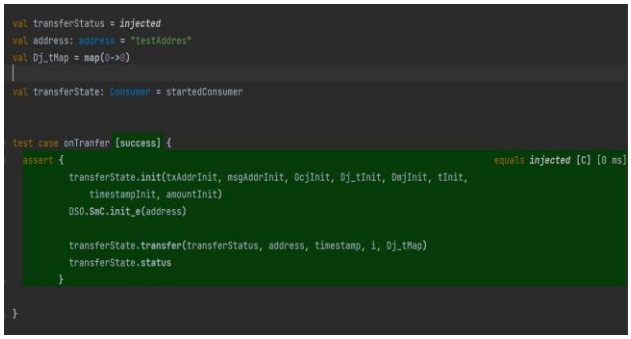
Slika br.5 *Test Consumer-a na inicijalni događaj*

Implementirani su događaji za prelazak stanja, kao i događaji za međusobnu komunikaciju i razmenu energije. Implemetirana je funkcionalnost registracije novih korisnika u sistem. Celokupno rešenje je prikazano kao DSL koji olakšava korišćenje nekih od već pomenutih jezika za pisanje pametnih ugovora (*Smart Contract*-a). Jezik je jednostavan, mali i modularno implemetiran. Svi testovi su uspešno izvršeni i pokazana je uspešna

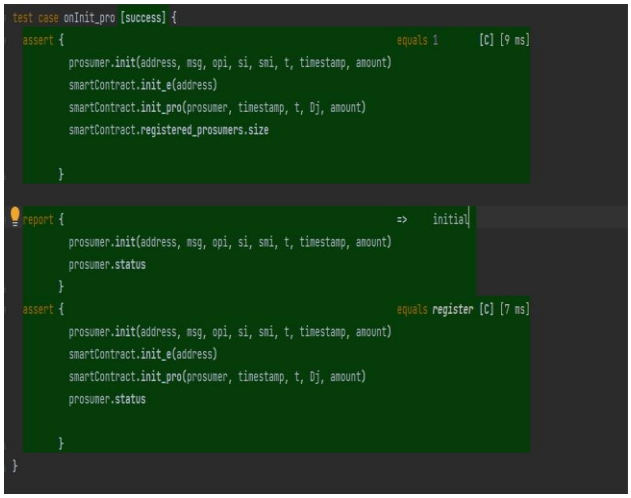
funkcionalnost trenutnog jezika. Korišćeno je više test slučajeva (slike od 5 do 8).



Slika br.6 *Test Consumer-a na događaj sendTx()*



Slika 7. *Test Consumera-a na događaj transfer()*



Slika 8. *Prikaz testa Smart Contract-a na događaj init_pro()*

6. ZAKLJUČAK

U radu je opisano testiranje DSL-a (*Domain-specific language*) – domenski specifičnog jezika za *energy* P2P (*Peer-to-Peer Service*) razvijenog u MPS okruženju uz podršku jezika KernelF-a. Opisani su učesnici *Prosumers* i *Consumers* kao i *Smart Contract* sa pripadajućim testovima i funkcionalnostima. Objašnjene su i implemetirane funkcionalnosti za učesnike, njihove događaje i stanja kao i za pametne ugovore.
Prednost razvoja domenskog jezika pomoću KernelF-a je u tome što su osnovni koncepti već implementirani u vidu *plugin*-a za jezik, a neophodno je samo konfigurisati okruženje i pravilno iskoristiti definisane koncepte. Takođe, omogućeno je veoma brzo izvršavanje i provera validnosti pomoću interpretera testova kojeg je neophodno konfigurisati. Prednost KernelF-a je u tome što je dobar za manje projekte kojima još osnovni koncepti nisu definisani, a moguće ih je veoma brzo testirati ili za jezgra velikih jezika izgrađenih na vrhu KernelF-a.

Mana KernelF-a je u tome što je nov jezik koji je i dalje u razvoju i zbog toga ne postoji puno primera i literature kao pomoći pri implementaciji jezika i generatora ili interpretera koda u MPS-u. Takođe, MPS je vezan za javu, pa se stoga celokupna aplikacija pokreće preko javine virtuelne mašine što smanjuje performanse.

Ovim radom su opisani i testirani osnovni koncepti jezika, ali implementacija još nije završena u potpunosti. U daljem razvoju bilo bi potrebno implementirati i DSO i primeniti algoritam za izračunavanje cena, nakon čega bi bila u potpunosti implementirana „menjačnica za energiju“. Posle ovoga bi se mogao izgenerisati kod za željenu platformu.

7. LITERATURA

[1] Smart Contracts dostupno na: https://www.ibm.com/topics/smart-contracts.
[2] Blockchain dostupno na: https://www.investopedia.com/terms/b/blockchain.asp
[3] Markus Vöelter, Design, evolution and use of KernelF
[4] Voluntis Healtcare dostupno na: https://www.voluntis.com/solutions/
[5] P2P Service dostupno na: https://www.investopedia.com/terms/p/peertopeer-p2p-service.asp
[6] Jae Geu Song, Eung soun Kang, Hyeon Woo Shin and Ju Wook Jang, A Smart Contract-Based P2P Energy trading Syistem with Dynamic Pricing on Etherum Blockchain
[7] Chekired, D.A.; Khoukhi, L.; Mouftah, H.T.; Decentralized cloud-SDN architecture in smart grid: A dynamic pricing model. IEEE Trans. Ind. Inform. 2018, 14, 1220–1231
[8] Markus Vöelter, KernelF- an Embeddable and Extensible Functional Language reference
[9] Marija Borisov, Jezik specifičan za domen za energy P2P trading baziran na MPS mbddr i KernelF specifikaciji, Ispitni rad iz Odabranih poglavlja savremenih metoda razvoja softvera (predmet na toktorskim studijama), 2021

Kratka biografija:

Marko Ercegovac rođen je u Sremskoj Mitrovici, Republika Srbija, 1997 god. Osnovne akademske stududije je upisao na Fakultetu tehničkih nauka Univerziteta u Novom Sadu 2016. Diplomirao je 2020. god.