УНИВЕРЗИТЕТ У НОВОМ САДУ
**ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ**

Мијановић Бојан

# Дизајн и имплементација *Blockchain* механизма са криптовалутом

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2024

Образац **Q2.НА.15-04** - Издање 1

# Садржај

# 1 Увод

*Blockchain* технологија представља дистрибутивну, децентрализовану и јавну базу свих трансакција [1].

Први концепт *blockchain* технологије помиње се у 1982. години, када је Давид Чаум у својој дисертацији описао дистрибуирану базу података која користи криптографију [2]. Овај рани рад није био директно повезан са дигиталним валутама, али је поставио темеље за будући развој *blockchain* - а.

Права револуција долази 2008. године када Сатоши Накамото објављује рад "*Bitcoin*: *Peer-to-peer* електронски готовински систем", уводећи први модерни *blockchain* и криптовалуту *Bitcoin*. Генесис блок, први блок *Bitcoin blockchain* - а, ископан је 3. јануара 2009. године, означавајући почетак *blockchain* технологије какву данас познајемо [3].

*Etherium*, лансиран 2015. године од стране Виталика Бутерина, увео је паметне уговоре који омогућавају сложеније трансакције и аутоматизацију различитих процеса. Овај развој проширио је примену *blockchain* технологије далеко изван дигиталних валута, омогућавајући креирање децентрализованих апликација [4].

*Blockchain* технологије су се од свог настанка имплементирале у различитим програмским језицима и окружењима. У својим раним фазама, *blockchain* технологије су се углавном развијале користећи језике као што су *C++* и *Java*, захваљујући њиховој ефикасности и широкој употреби у индустрији. Касније, с појавом паметних уговора, *Solidity* је постао стандард за развој на *Etherium* платформи.

Овај рад се фокусира на имплементацију основних концепата *blockchain* технологије у програмском језику *Rust*, који је познат по својој сигурности, перформансама и могућности превенције грешака при руковању меморијом.

Рад је структуиран X целина

# 2 Основе *Rust* програмског језика

*Rust* је савремени програмски језик који је развијен да буде безбедан и брз. Развијен од стране *Mozilla Research*-а, *Rust* је од свог настанка привукао велику пажњу због својих изузетних безбедносних карактеристика и перформанси [5].

## 2.1 Увод у *Rust* програмски језик

*Rust* је системски програмски језик, а уместо интерпретираног језика, као што су *JavaScript* или *Ruby*, има компајлер, као што имају *Go*, *C* или *Swift*. Не комбинује активни *runtime*, али обезбеђује језичку ергономију. Све је ово могуће захваљујући компајлеру који спречава грешке било којег типа и осигурава да не дође до проблема у меморији пре него што се покрене апликација [6].

*Rust* обезбеђује перформансе (нема *runtime*, нити прикупљање "смећа"), безбедност (компајлер осигурава да је све безбедно за меморију, чак и у асинхроним окружењима) и продуктивност (његове уграђене алатке за тестирање, документацију и "менаџер"пакета чине га лаким за израдз и одржавање) [6].

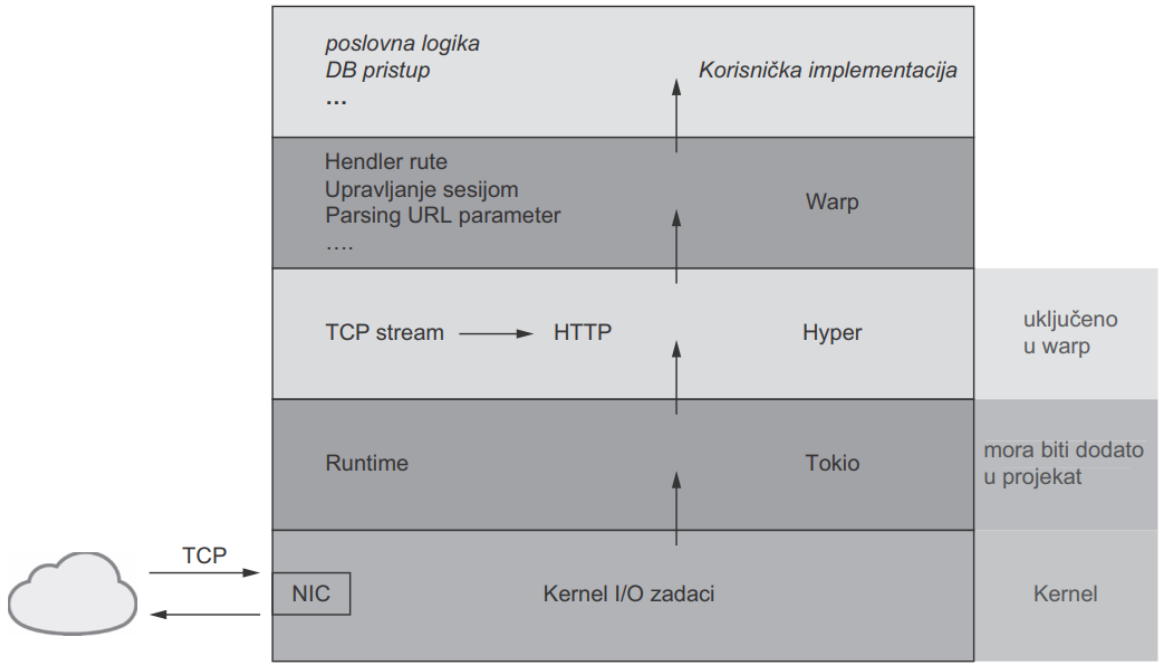
## 2.2 Зашто *Rust* за *blockchain*?

Када је у питању развој blockchain апликација, *Rust* се истиче као одличан избор из неколико разлога:

1. **Безбедност меморије**: *Rust*-ов систем власништва и провера за време компилације осигуравају да програмери избегну уобичајене грешке у раду са меморијом, што је критично за сигурност *blockchain* система [7].
2. **Перформансе**: *Rust* је дизајниран да буде брз и ефикасан. Његов минималан *overhead* и високо оптимизован компајлер резултирају брзим извршавањем кода, што је важно за обраду великог броја трансакција у реалном времену [7].
3. **Паралелизам и конкурентност**: *Rust* нуди снажну подршку за паралелно и конкурентно програмирање, омогућавајући оптимално коришћење мултијезгарних процесора [7].

*Rust* нуди низ алата и библиотека које олакшавају развој сложених апликација. Две од најзначајнијих библиотека за развој *blockchain* апликација су *Tokio* и *libp2p*. Ове библиотеке пружају подршку за асинхроне позиве и *peer-to-peer* комуникацију, што је кључно за функционалност и ефикасност *blockchain* система.

Слика 2.1 прииказује технички стек који је укључен у одабир радног оквира. ***Warp*** је довољно мали да се "склони са пута", довољно се користи да се њиме управља активно и има активну заједницу. Заснован је на *Tokio runtime* - у.




Слика 2.1: *Warp* веб радни оквир

***libp2p*** је модуларни мрежни стек који омогућава *peer-to-peer* комуникацију. У контексту *blockchain*-а, *libp2p* се користи за омогућавање комуникације између различитих чворова у мрежи. Ова библиотека је флексибилна и подржава различите протоколе за пренос података, што је чини идеалном за развој децентрализованих апликација.

# 3 Увод у *blockchain* технологију

*Blockchain* технологија представља савремен приступ складиштењу и дистрибуцији података. Основни принципи и концепти *blockchain* технологије нуде дубоку промену у начину на који се информације похрањују, проверавају и дистрибуирају путем децентрализоване мреже рачунара.

## 3.1 Основни принципи и концепти

*Blockchain* се може дефинисати као дистрибуисана дигитална књига трансакција. Основна идеја је стварање низа блокова који садрже податке. Блкови су криптографски повезани тако да је немогуће мењати податке у претходним блоковима без мењања свих следећих блокова [8].

Кључни елементи *blockchain*-а укључују:

1. **Децентрализација**: Подаци се похрањују и управљају путем мреже чворова уместо централизованог ауторитета, што осигурава транспарентност и отпорност на цензуру.
2. **Дистрибуираност**: Сваки чвор у мрежи садржи комплетан или део *blockchain*-а, омогућујући свима у мрежи да виде исте податке. Ово спречава појединачне тачке квара и повећава отпорност на нападе.
3. **Сигурност**: Криптографски алгоритми осигуравају да је свака промена у *blockchain*-у лако проверљива, а трансакције се потврђују кроз консензус мреже.
4. **Неповратност**: Након што је трансакција забележена у *blockchain*-у, тешко ју је променити или обрисати без сагласности већине чворова у мрежи, чиме се осигурава поверење и интегритет података.

## 3.2 Поређење са традиционалним базама података

Насупрот традиционалним базама података које су често централизоване и ослањају се на поверење у једну јединицу, *blockchain* нуди неколико кључних разлика:

1. **Централизација у односу на децентрализацију**: Традиционалне базе података често су централизоване под контролом једне организације. *Blockchain* дистрибуише податке широм мреже, елиминишући потребу за централним ауторитетом.
2. **Транспарентност и проверљивост**: *Blockchain* омогућава свим корисницима увид у све трансакције које су се догодиле, што повећава транспарентност и смањује могућност манипулације.

3. **Сигурност и отпорност**: Због своје дистрибуиране природе, *blockchain* је отпорнији на нападе и кварове у поређењу са традиционалним базама података које су осетљиве на појединачне тачке квара.

4. **Ефикасност и трошкови**: Иако *blockchain* може бити спорији у обради података у поређењу са централизованим базама података, његова сигурност и транспарентност могу надмашити трошкове и ризике традиционалних система.

# 4 Архитектура апликације

*Blockchain* технологија се састоји од неколико кључних компоненти које омогућавају њено функционисање. Основне јединице података су блокови, који садрже информације о трансакцијама, временским ознакама и криптографским хеш функцијама претходних блокова. Ови блокови су повезани у секвенцијални ланац, познат као *blockchain*, који осигурава неповредивост података.

Дистрибуирана мрежа чворова заједнички одржава и верификује *blockchain*, омогућавајући децентрализацију. Консензус алгоритми, као што су *Proof of Work (PoW)* и *Proof of Stake (PoS)*, омогућавају учесницима мреже да се сложе око валидности нових блокова. Криптографија осигурава сигурност и приватност података унутар *blockchain* -а, користећи хеш функције и дигиталне потписе.

У наредним подсецијама, детаљно ћемо описати сваку од ових компоненти, укључујући процесе као што су PoW и мајнинг, који су кључни за додавање нових блокова у ланац.

## 4.1 Блокови

Блокови су основне јединице података у *blockchain* технологији. Сваки блок садржи скуп података који су повезани са трансакцијама и другим важним информацијама. У контексту *blockchain*-а, блокови су организовани у ланац, где сваки блок садржи хеш претходног блока, што обезбеђује интегритет и сигурност података. Следећи код приказује структуру блока у Rust програмском језику:

```rust
pub struct Block {
    pub timestamp: DateTime<Utc>,
    pub last_hash: String,
    pub hash: String,
    pub data: Vec<Transaction>,
    pub nonce: u64,
    pub difficulty: u64,
}
```

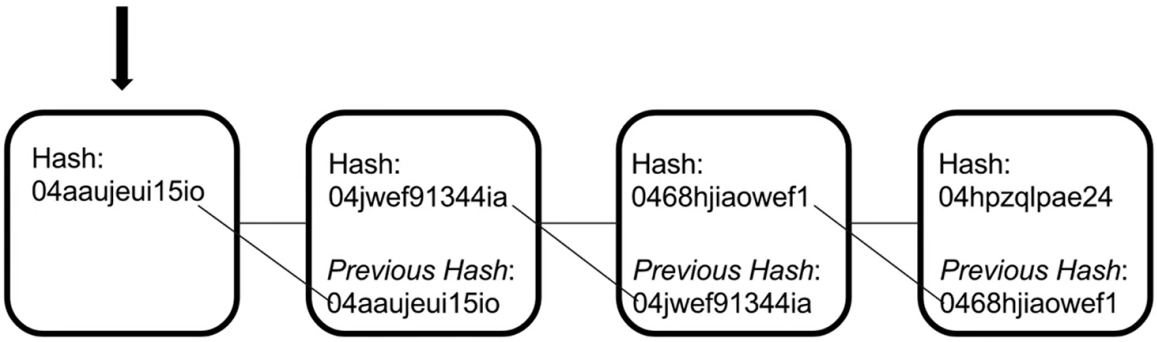
Атрибути блока су:

- **timestamp**: Време када је блок креиран. Овај атрибут омогућава праћење хронологије трансакција у *blockchain*-у.
- **last_hash**: Хеш вредност претходног блока у ланцу. Овај атрибут обезбеђује да сваки блок буде повезан са својим претходником, чиме се осигурава интегритет ланца.

- **hash**: Хеш вредност тренутног блока. Ова вредност се добија применом хеш функције на садржај блока и служи као јединствени идентификатор блока.
- **data**: Податке у блоку, који обично укључују трансакције. У овом случају, то је вектор трансакција *(Vec<Transaction>)*.
- **nonce**: Произвољни број који рудари мењају током процеса рударења како би добили хеш вредност блока која задовољава критеријуме тешкоће.
- **difficulty**: Ниво тежине који одређује колико је сложено пронаћи важећи хеш за блок. Тежина рударења се прилагођава да би се одржала константна брзина креирања блокова у мрежи.

### 4.1.1 Генесис блок

Генесис блок је први блок у ланцу блокова и служи као темељ целокупног *blockchain* система (Слика 4.1). Он нема претходника и обично је ручно креиран од стране креатора *blockchain*-а. Генесис блок обично садржи посебне параметре и почетне вредности које су специфичне за дат *blockchain*. Његова важност лежи у чињеници да сваки наредни блок у ланцу зависи од њега кроз хеш вредности.




Слика 4.1: Приказ генесис блока у *blockchain*-у

### 4.1.2 Рударење

Рударење је процес додавања нових блокова у *blockchain*. Рудари користе своју рачунарску снагу да реше комплексне математичке проблеме који су потребни за валидацију нових трансакција и креирање нових блокова. Овај процес захтева значајну количину енергије и ресурса, али је кључан за одржавање безбедности и децентрализације *blockchain* мреже. У процесу рударења, рудари се такмиче да пронађу одговарајући *nonce* који ће произвести хеш вредност која испуњава одређене критеријуме тешкоће.

### 4.1.3 Хеш функција

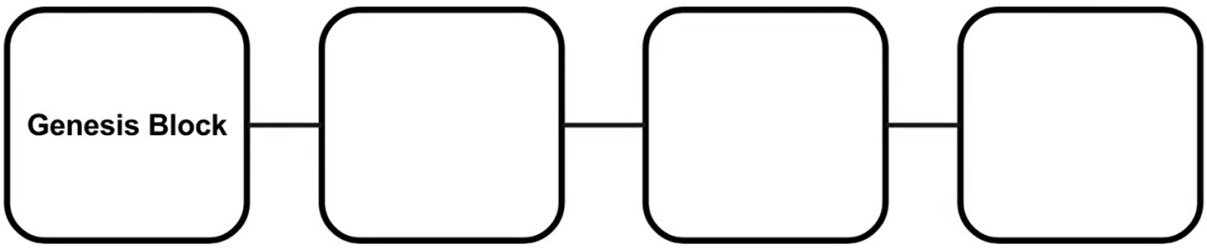
Хеш функција је критичан елемент у *blockchain* технологији, јер обезбеђује сигурност и интегритет података у блоковима. Хеш функција узима улазне податке произвољне дужине и генерише фиксну дужину излазне вредности, која је јединствена за те улазне податке. У контексту *blockchain*-а, хеш функција се користи да повезује сваки блок са претходним блоком, чиме се обезбеђује да свака промена у подацима било ког блока одмах утиче на све наредне блокове, што чини *blockchain* изузетно отпорним на манипулацију.

## 4.2 Ланац

*Blockchain* је структура података која се састоји од низа повезаних блокова, где сваки блок садржи хеш претходног блока, чиме се обезбеђује интегритет и сигурност ланца. Следећи код приказује структуру *blockchain*-а у *Rust* програмском језику:

```
pub struct Blockchain {
    pub chain: Vec<Block>,
}
```

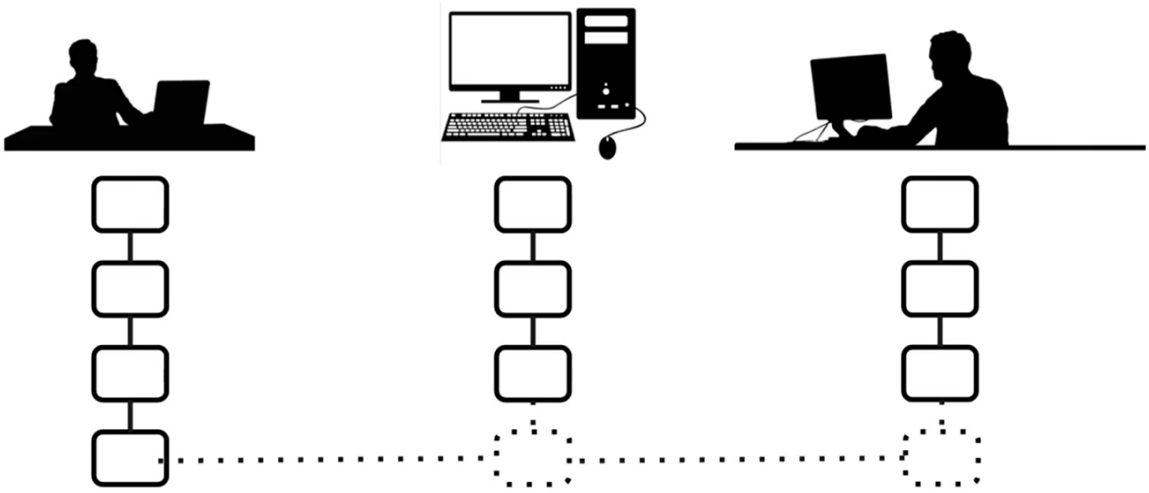
Атрибут ***chain*** је вектор који чува редоследно повезане блокове, формирајући ланац блокова. Слика 4.2 приказује структуру ланца са генесис блоком на почетку.




Слика 4.2: Приказ идеје *blockchain*-а

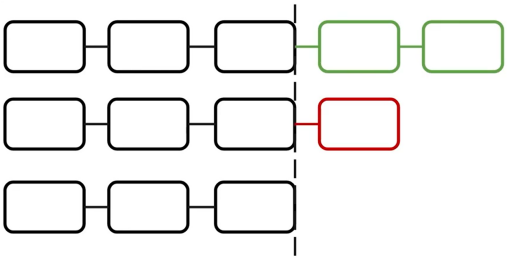
### 4.2.1 Валидација више ланаца

Идеја овог механизма је да подржи више доприносиоца, при чему ће више доприносиоца додавати блокове у *blockchain*. Сваки рудар ће имати своју верзију истог ланца. Када један рудар дода нови блок у ланац, мораће да пошаље тај нови блок осталим ланцима у систему како би они прихватили ту промену и ажурирали целокупни систем. На тај начин сви добијају ажурирану копију са тим новим блоком, чиме се осигурава да сви ланци буду конзистентни.

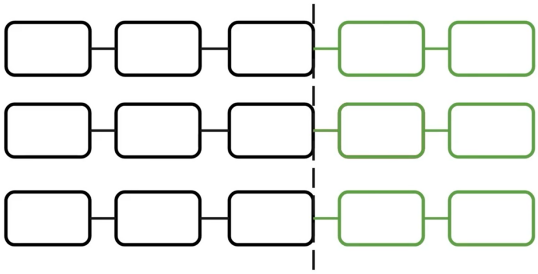


Слика 4.3: Приказ дељења ланаца

Међутим, да би сви рудари прихватили ове нове ланце, мора постојати неки облик валидације који ће осигурати да је нови блок валидан и да треба да буде прихваћен. Главни облик валидације је прихватање дужих ланаца који стигну. На пример, ако сви имају договорени *blockchain* који је већ дуг три блока, и један рудар дода два блока у ланац, док други рудар дода само један блок у исто време, систем ће прихватити дужи ланац. На тај начин се осигурава да договорени ланац за све увек буде онај који садржи највише података.

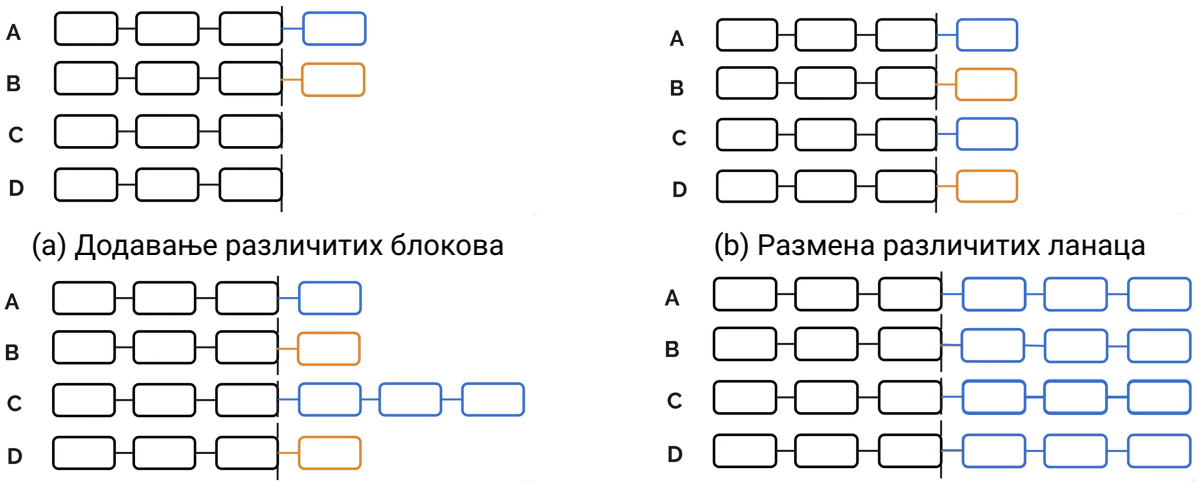


Слика 4.4: Стање пре валидације



Слика 4.5: Стање после валидације

Ово такође решава проблем рачвања у ланцу. На пример, ако две одвојене инстанце *blockchain*-а истовремено произведу један блок на основу претходног блока, настаје рачвање у систему где оба рудара производе блок на основу истог претходног блока (слика 4.6a). Пола рудара ће имати ланац који је произвео рудар А, а друга половина ће имати ланац који је произвео рудар Б (слика 4.6b). Коначно, систем треба да дође до договора о томе који ланац ће прихватити. Ако неко дода неколико блокова на ланац рудара А, тај ланац ће сада бити дужи од свих осталих у систему (слика 4.6c). Сви ће морати да прихвате најдужи ланац, који садржи блок од рудара А, чиме се решава рачвање прихватањем оригиналног блока од рудара А (слика 4.6d).




(a) Додавање различитих блокова

(b) Размена различитих ланаца

(c) Додавање нових блокова

(d) Прихватање најдужег ланца

Слика 4.6: Решавање рачвања

Ово не значи да блокови са ланца рудара Б губе оригиналне податке, јер се блок који није укључен у рачвање може сада додати на крај новоприхваћеног ланца.

Други облик валидације је провера вредности хеша произведених за сваки блок ланца. Сваки *blockchain* има приступ хеш функцији која генерише хеш на основу података блока. Када *blockchain* прими нови ланац, може осигурати да је хеш исправно генерисан тако што ће сам поново генерисати тај хеш. Ако се хешеви не поклапају, вероватно су подаци мењани, и због тога *blockchain* неће прихватити нови ланац.

# 5 Литература


[1] Zheng, Zibin, Shaoan Xie, Hong Ning Dai, Xiangping Chen, i Huaimin Wang: *An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends*. strana 1, Jun 2017.

[2] Sherman, Alan, Farid Javani, Habin Zhang, i Enis Golaszewski: *On the Origins and Variations of Blockchain Technologies*. University of Maryland, Baltimore County (UMBC) Baltimore, Maryland 21250, Octobar 2018.

[3] Nakamoto, Satoshi: *Bitcoin: A Peer-to-Peer Electronic Cash System*. Cryptography Mailing list at https://metzdowd.com, Mart 2009.

[4] Sixt, Elfriede: *Ethereum*, strane 189–194. Januar 2017, ISBN 978-3-658-02843-5.

[5] *Rust (programming language)*. `https://en.wikipedia.org/wiki/Rust_(programming_language)`. прегледано 16. јул 2024.

[6] Gruber, Bastian: *Rust Web Development*. Manning, 2023, ISBN 978-1617299001.

[7] *The Rust Reference*. `https://doc.rust-lang.org/reference`. прегледано 16. јул 2024.

[8] Aggarwal, Shubhani i Neeraj Kumar: *Chapter Seven - Basics of blockchain*. U: Aggarwal, Shubhani, Neeraj Kumar, i Pethuru Raj (urednici): *The Blockchain Technology for Secure and Smart Applications across Industry Verticals*, tom 121 iz *Advances in Computers*, strane 129–146. Elsevier, 2021. `https://www.sciencedirect.com/science/article/pii/S0065245820300620`.

# 6 Подаци о кандидату

Кандидат Бојан Мијановић је рођен 2002. године у Зрењанину. Завршио је средњу школу у Зрењанину, 2020. године као ђак генерације. Факултет Техничких Наука у Новом Саду је уписао 2020. године. Испунио је све обавезе и положио је све испите предвиђеним студијским програмом са просечном оценом од 9.75.