Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Evidenčné číslo: FIIT-100241-97019

Tomáš Rafaj

DeFi herná platforma

Bakalárska práca

Vedúci záverečnej práce: Ing. Lukáš Mastiľak
Konzultant: Ing. Kristián Košťál, PhD.

September 2021

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Evidenčné číslo: FIIT-100241-97019

Tomáš Rafaj

DeFi herná platforma

Bakalárska práca

Študijný program: Informatika (konverzný)

Študijný odbor: Informatika

Školiace pracovisko: Ústav počítačového inžinierstva a aplikovanej informatiky

Vedúci záverečnej práce: Ing. Lukáš Mastiľak
Konzultant: Ing. Kristián Košťál, PhD.

September 2021

Čestné prehlásenie

Čestne prehlasujem, že som bakalársku prácu na tému Internet of value a DeFi vypracoval samostatne, na základe vedomostí nadobudnutých samoštúdiom, s použitím uvedenej odbornej literatúry a na základe konzultácií pod odborným vedením Ing. Lukáša Mastiľaka a Ing. Kristiána Košťála, PhD.

V Bratislave, 16.5.2022 .....

Tomáš Rafaj

Poďakovanie

Týmto spôsobom by som sa chcel poďakovať vedúcemu bakalárskej práce Ing. Lukášovi Mastiľakovi a Ing. Kristiánovi Košťálovi, PhD. za odbornú pomoc a cenné rady, poskytnuté počas celej doby vypracovávania.

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný program: Informatika (konverzný)

Autor: Tomáš Rafaj
Bakalárska práca: DeFi herná platforma
Vedúci projektu: Ing. Lukáš Mastiľak
Konzultant: Ing. Kristián Košťál, PhD.
September 2021

Cieľom práce Internet of value a Defi bolo poukázať na aktuálnu pripravenosť a možnosť tvoriť decentalizované hry vrámci blockchainových sietí. Zaoberá sa vhodnosťou jednotlivých blockchainov a ich implementácii, pričom hľadá vhodný prienik, ktorý slúži ako formovanie návrhu samotného výstupu práce. Dotýka sa o existujúce riešenia, ktoré riešili obdobné problémy, hodnotí ich prístup, prípadne vyzdvihne chyby, ktorým sa treba vyvarovať. Skúma a opisuje možnosti a rozvinutosť konkrétne siete Ethereum na tvorbu a rozvoj herných platforiem, ktorá sa zdá ako najvhodnejšia sieť z mnohých dôvodov. V kapitolách sú opísané vyžadované funkcionálne požiadavky, detailne opísaný samotný návrh riešenia na základe analýzy už spomínaných riešení. Práca obsahuje teóriu siete Ethereum, úvahu o tom ako by zmenili plánované zmeny sieti Ethereum návrh na zapajánie hier a implementáciu. Záver práce obsahuje celkové zhodnotenie problémov a výsledkov práce. Práca obsahuje základ pre hry na blockchaine, pričom dbá na poukázanie možnej škálovateľnosti a ukážku hry.

Annotation

Slovak University of Technology in Bratislava
Faculty of informatics and information technologies
Degree Course: Information Security

Author: Tomáš Rafaj
Bachelor's thesis: DeFi Gaming platform
Supervisor: Ing. Lukáš Mastiľak
Consultant: Ing. Kristián Košťál, PhD.
September 2021

The aim of the work Internet of value and Defi was to point out the current readiness and possibility to create decentalized games within blockchain networks. It deals with the suitability of individual blockchains and their implementation, while looking for a suitable penetration, which serves as the formation of the design of the work output itself. It touches on existing solutions that have solved similar problems, evaluates their approach, or highlights mistakes that need to be avoided. It examines and describes the possibilities and development of the specific Ethereum network for the creation and development of gaming platforms, which seems to be the most suitable network for many reasons. The chapters describe the required functional requirements, describe in detail the solution itself based on the analysis of the already mentioned solutions. The work contains the theory of the Ethereum network, a consideration of how the planned changes of the Ethereum network would change the proposal for game engagement and implementation. The conclusion of the thesis contains an overall evaluation of the problems and results of the work. The work contains the basis for blockchain games, while paying attention to the possible scalability and demonstration of the game.

Obsah

Zoznam obrázkov

Zoznam použitých skratiek

PoW	Proof of work
PoS	Proof of stake
EVM	Ethereum virtual machine
NFT	Non fungible token
ETH	Ethereum

Kapitola 1

Úvod do problematiky

Blockchain technológie majú v poslednej dobe vysokú pozornosť vo viacerých odvetviach. Najväčšie zameranie poňala zatiaľ finančná sféra, ktorá vytvára hlavný use case blockchainu. Typický blockchain pozostáva z dát, ktoré sa ukladajú do reťazcov dát nazývaných blokov. A všetky bloky dokopy vytvárajú blockchain, ktorý obsahuje kompletnú históriu transakcií. Prvým projektom bol Bitcoin, neskôr začali vznikať viaceré projekty a medzi ne patrí aj samotné Ethereum. Vitalik Buterin sa začal zaujímať o bitcoin, trápilo ho, že bitcoin si môžeme predstaviť ako banku, ktorá dokáže robiť 3 základné úkony : vlož peniaze na účet, vyber peniaze z účtu, presuň peniaze z môjho účtu na iný. Chcel presadiť zmeny ako napríklad vykonávanie pokročilejších operácii ako je napríklad "dovoľ mi vyberať si každý mesiac maximálne 50 eur", "urob účet troch majiteľov, kde Alica môže vyberať bez limitov ale Bob iba ak mu povolí ešte niekto iní z majiteľov" a iné. Neskôr sa, teda rozhodol pre vlastný projekt s názvom Ethereum, ktorý slúži ako vysoko levelový neobmedzený počítač. Pre realizáciu mojej aplikácie som si vybral platformu Ethereum, nakoľko ponúka dobrú funkcionalitu elektronických zmlúv a je na nej vybudovaných viacero funkčných projektov [1].

Kapitola 2

Analýza

Táto kapitola sa zameriava na opis, zhrnutie a opísanie funkčných platforiem a projektov na Ethereum, ktoré majú herné aplikácie riešené pomocou onchain kontraktov. Opisuje taktiež potrebné teoretické znalosti a predpoklady vrámci blockchain vývoja. Poukazuje na slabé a silné stránky Ethereum siete.

2.1 Ethereum blockchain

Ethereum sa dá chápať ako transakčný stroj, ktorý zachováva stavy siete a všetkého, čo sa v sieti deje. Začiatkom je genesis stav, môžeme ho označovať aj genesis blok, nesúci prvý počiatočný stav [2]. V Ethereum tento blok vznikol Jul-30-2015 03:26:13 PM +UTC [1]. Postupne sa teda inkrementuje zo stavu genesis až do finálneho stavu. Bloky siete Ethereum teda fungujú ako zápisník, ktorý zapisuje jednotlivé transakcie, predošlý blok a identifikátor finálneho stavu. Bloky neobsahujú samotný konečný stav, z dôvodu príliš veľkej veľkosti [2]. Bloky v prípade Ethereum platformy obsahujú základné veci ako sú transakcie, vrátane smart kon-

[1] https://etherscan.io/block/0

traktov, obtiažnosť, číslo bloku, a odkaz na najnovší stav [3]. Ethereum blockchain pracuje teda tak, že zoberie vstupy, na základe vstupov vyráta nové stavy a tie zapíše do blokov. Preto aby bola transakcia platná, je potrebné aby prešla validačným procesom s názvom "mining". Táto skutočnosť vyplýva z toho, že Ethereum je platforma využívajúca konsenzus proof of work. Miner, respektíve ťažiar, ktorý validuje blok dostáva odmenu za overenie a zapísanie do Ethereum blockchainu [4].

2.1.1 Proof of work

Ethereum je platforma využívajúca a rešpektujúca konsenzus fungovania s názvom proof of work, ako už bolo spomenuté vyššie. Pod týmto pojmom rozumieme to, že uzly vrámci peer-to-peer Ethereum siete zdieľajú pohľad na aktuálny globálny stav siete. Užívateľ tejto siete interaguje tak, že urobí transakciu, ktorá sa zmení v platný stav. Proces zabezpečujú uzly, ktoré vyberajú z mempool (zbierka neschválených transakcií) transakcie a verifikujú ich validitu, potrebné výpočty, zmenu stavu účtov. Následne v prípade ich validity menia stav siete [5]. Ethereum zabezpečovací algoritmus, ktorý mineri riešia sa nazýva ethhash, kde úlohou je nájsť nonce, ktorý bude vstupom do algoritmu a výsledok je menší ako určená obtiažnosť siete. V priemere trvá vytvorenie, respektíve vyťaženie jedného bloku približne 15 sekúnd [6].

Bloky

Blok v Ethereum vzniká ako výsledok a dôsledok proof of work konsenzu. Blok obsahuje veľmi veľa informácii, obsahuje číslo bloku, ktoré definuje koľký blok je v poradí. Ďalej obsahuje vlastný 32 binárny hash bloku, obsahuje hash bloku nad ním v stromovej štruktúre a tým pádom vlastní aj rodiča. Obsahuje tiež podstatnú informáciu ako je timestamp, to určuje kedy bol vytvorený, nonce, veľkosť

v byte-och, minerovu adresu, ktorý blok vyťažil, obtiažnosť siete daného bloku vyjadrenú číslom typu integer, totálnu obtiažnosť siete po tento blok, gas limit, použitý gas, obe gas parametre sú vyjadrené číslom typu integer. Ďalej extradáta, binárne dáta rôznej veľkosti, obsahuje ďalej hash prvej transakcie, a taktiež hash hodnoty všetkých transakcií, samotné transakcie, počet transakcií vyjadrených číslom typu integer, hash uncií, uncie, zbierka blokov uncií [7].

2.1.2 EVM

Ethereum virtuálna mašina nedokáže predpokladať deterministicky, či program niekedy skončí, spôsobuje to problém ako je napríklad útok zamietnutie-služby, kde by užívateľ vytváral nekonečné úlohy a problémy v cykle bez konca a zamedzil by tak prístup ostatným. Na vyriešenie tohoto problému, ethereum protokol obsahuje cenník [5]. Smart kontrakty sú vykovanávané práve pomocou EVM. Dá sa povedať, že smart kontrakty sú program na EVM. Väčšina smart kontraktov je nasadených ako EVM byte kód, pričom málokedy je priamo písaný. Najpopulárnejšie prostredie na programovanie je v Solidity, ktorý má bohatú syntax. V jednoduchosti sa dá EVM označiť ako stavová mašina, ktorá vykonáva programy [8].

2.1.3 Smart kontrakty

Smart kontrakty sú programy spúšťané na EVM. Majú asociované dáta, teda stavy, ktoré dokážu automaticky ovplyvňovať stav účtov Ethereum. Všetky zmeny spôsobené volaním tohoto programu sú následné zapísané do siete Ethereum. Samotné kontrakty sú izolované od systému, kde sú vykonávané. Každý kontrakt stojí poplatok nazývaný gas. Bežná prax je písať ethereum smart kontrakty v už spomínanom programovacom jazyku Solidity, ktorý má objektovo orientované koncepty [9]. Viaceré smart kontrakty definujú štandard Ethereum Request for Comment pre základ tokenizácie a najznámenjší je kontrakt štandardu ERC-20.

2.1.4 Solidity

Solidity je jazyk používaný k tvorbe smart kontraktov na Ethereum blockchaine [10]. Najväčšiu podobnosť má s jazykom JavaScript a jazykom C. Je objektovo orientovaný, citlivý na veľkosť písmen, a statický jazyk. Kompilácia kódu prebieha do bytecode a má automatické kontrolovanie syntax problémov. [11] Jazyk vyvinul Gavin Wood, jeden zo spoluzakladateľov Ethereum siete [12].

2.1.5 Poplatky

Poplatky sú počítané pomocou jednotky gas, odhad veľkosti gasu je úzko prepojený s EVM a jej inštrukciami. Bežna cena je 21000 jednotiek gasu , pričom 1 gas je rovný 0.000000001 ethereum, avšak v prípade viacerých zmien a ukladania stavov, vytvárania kontraktov, ktoré stoja veľa výkonu sa táto cena zvyšuje, aby sa tak napríklad zamedzilo aj spomínaným problémom ako je zamietnutie služby s nekonečnými cyklami a podobne. V prípade chyby kódu v rámci smart kontraktu alebo chybného skoku v rámci inštrukcií sa minie všetok gas, stavy sa nezmenia, a výsledok bude zlyhanie so správou out-of-gas exception [12].

2.1.6 Ethereum 2.0

Cieľom Ethereum siete je naďalej zväčšovať a zlepšovať sieť, čo sa týka bezpečnosti, priepustnosti, škálovateľnosti a spotreby elektrickej energie. Vďaka tomu vznikla myšlienka takzvaného Ethereum 2.0 konsenzus protokolu, ktorý upúšťa od pôvodného proof of work protokolu a aplikuje proof of stake protokol. Tento mechanizmus spočíva v tom, že účastníci, ktorí úvádzajú stavy sú vyberaní tak často ako veľká je proporcia ich vkladu voči celej sieti. Ďaľšou veľkou zmenou oproti terajšiemu chodu protokolu je možné rozdelovanie blockchainových blokov do viacero menších blokov, čo povoľuje a umožňuje vytváranie menších a pravdepodobne

rýchlejších chainov, okrem hlavného blockchainu. Hlavnou súčastou tejto aktualizácie je prvá fáza spustenie takzvaného beacoin chainu. Upgrade sa plánuje roky, avšak podľa aktuálnych informácii z ethereum.org je úplné spustenie plánované na December 2022.

Beacon chain

Beacon chain je špeciálny typ prokotolu, ktorý presadzuje konsenzus týkajúci sa uzlov na základe veľkosti stávky. Tieto uzly sa zúčastňujú na zabezpečovaní siete a spravujú stavy. Účastníci sa nazývajú validátori a ich hlavnou úlohou je navrhovať a hlasovať za nové Beacon bloky, ktoré budú pripojené k blockchainu. Zloženie a sada týchto validátorov je premenlivá, noví validátori sa môžu zúčastniť vložením stávky ethereum, minimálne 32, po registrácii sú validátori oprávnení zúčastniť sa na hlasovaní a tvorbe nových blokov. Beacon chain sa započal 1. decembra 2020, a v čase 14. október 2021 už má viac ako 250 tisíc validátorov, ktorí vložili cez 7.7 milióna ethereum [13].

Zmena mojeho projektu v prípade Ethereum 2

Na základe nadobudnutých informácií a zistení som dospel k záveru, že v prípade fungovania Ethereum 2 blockchainu by neboli potrebné offchain transakcie ani state kanály, ktoré sú opísané v ďalšej časti tejto práce. Hlavný dôvod tvorby hier mimo blockchain sú poplatky a nízka škálovateľnosť siete, ale v prípade Ethereum 2 môžeme očakávať oveľa väčšiu priepustnosť, nižšie poplatky, tým pádom by sa mohli stať riešenia typu Layer 2 postavené nad blockchain redundantné. Avšak v čase tvorenia projektu je Ethereum stále proof of work mechanizmus a nie je možné využiť potenciál a možnosti proof of stake škálovania a priepustnosti.

2.1.7 Offchain

Potreba vykonávať operácaie mimo blockchain sa vyskytuje vtedy, keď potrebujeme rýchlejší a lacnejší výpočet našich úkonov, niekedy je dobré zobrať časť výpočetnej sily a dát z blockchainu, čo môže viesť k zmene výkonu, ceny a aj súkromia. Pričom na výsledky a dáta sa dá odkazovať pomocou kryptografických hashov do blockchainu, ktoré odkazujú na dáta mimo blockchain. Dá sa, teda povedať, že off-chain výpočty by mohli zvýšiť škálovateľnosť a zlepšiť súkromie. Avšak v tomto modeli by sa mohla stratiť hlavná myšlienka blockchainu a to je nedôveryhodnosť voči každému. Na riešenie tohto problému je potrebné, aby uzol dodával dôkaz o tom, že všetko prebehlo férovo, správne a využíva tiež model zero-knowledge, čo znamená to, že overovač nevie nič len fakt, že sa vykonal výsledok správne [14].

2.1.8 State channels

State channel je špecifický druh kanálu, ktorý sa uchováva mimo blockchain. Tieto kanály sa stále vyznačujú vysokým zabezpečením a taktiež finalitou, ktorá znamená to, že ak sa niečo udeje, stále sa môžeme vrátiť späť k blockchainu s aktuálnym stavom kanálu, nemusíme držať zmeny a stavy už naďalej v state kanáli. Spôsob fungovania state kanálov sa dá vysvetliť najjednoduchšie na prípade z praxe: Predstavíme si systém, kde sú dvaja hráči Alica a Bob a hrajú spolu jednoduchú hru s názvom hod mincou. Je dôležité, aby sme vrámci blockchainu urobili, čo najmenej transakcií, ideálne dve - jedna pre počiatočný stav, druhá pre koncový stav. Výsledky a stavy jednotlivých účtov Alice a Boba sa teda ukladajú off-chain, ale obaja môžu mať dôveru, že v prípade potreby sa vrátia naspäť na blockchain Ethereum siete. Je tu teda smart kontrakt, ktorý chápe pravidlám hodu mince a identifikuje našich hráčov Alica a Boba. Počas toho ako Bob a Alica hrajú sa

vytvárajú transakcie medzi nimi, komunikujú iba cez interenet, pričom ešte nič nie je odoslané ani zapísané v blockchaine. Tieto kroky a transakcie sú v podstate blockchainovo certifikované šeky. Podobne ako v banke, v prípade potreby môžete prísť s platnými šekami hocikedy a nechať si vyplatiť právom získané peniaze, v našom prípade napríklad tokeny, či ethereum mince. V prípade konca hry, napríklad Alica alebo Bob nebudú mať dostatok prostriedkov hrať ďalej alebo v prípade, že by niekto nechcel pokračovať. Smart kontrakt zavrie kanál a zapíše finálny stav - zoznam všetkých transakcií do blockchain a zaplatí jeden poplatok sieti [15].

2.1.9 Defi

Defi je skratka pre decentralizované financie. Tieto financie si vyznačujú viacerými prvkami od bežných centralizovaných financií ako napríklad v bežnej banke. Platobný systém a systém overovania prebieha jasne, rýchlo, bez potreby prostredníka na spracovanie platby, bez limitov a často aj za oveľa menšie poplatky ako v bežných financiách. Dobrým oporným bodom je prístupnosť, toto je rozhodne jedna z vecí, ktorú ma defi zvladnútu dokonalo, k používaniu defi nemusíte spĺňať žiadne podmienky ako je napríklad vek, príslušnosť štátu a podobne. Defi je dostupné každému, kto má internet a zariadenie, na ktorom bude vedieť pracovať. Nadväzujúc na prístupnosť sa vyznačuje tento systém transparentnosťou a decentralizovaním celého systému. Celá sieť, napríklad v prípade Ethereum blockchain je open-source. Celé defi protokoly sú teda ako open-source kódy, a v kóde sa nedá podvádzať, pretože ten sa správa ku každému účastníkovi bez rozdielu. Defi si treba predstaviť ako súbor produktov, ktoré ponúkajú riešenia z bežného sveta, ako sú pôžičky, bankové operácie, poistenia, dlhopisy [16].

2.1.10 Tokeny

Sieť Ethereum ponúka vlasnosť tvoriť a vytvárať takzvané tokeny. Tokeny na sieti ethereum sú nadtyp, sú definované smart kontraktom, fungujú ako napríklad mince, ale je dôležité, že nemajú svoj vlastný blockchain ako napríklad ethereum minca má Ethereum blockchain, tak tokeny bežia všetky na Ethereum blockchaine. Existuje niekoľko základných štandardov týchto tokenov, medzi ktoré patria :

- ERC-20 je základný štandard pre tokeny vrámci siete, je najpoužívanejší a prináša základné funkcie ako je napríklad prenášanie tokenov.
- ERC-721 je známy aj ako NFT, kde prináša unikátnosť v tom, že každý token je rozdielny od iného tokenu, čo dodáva aj možnosť sledovať jednotlivé tokeny.
- ERC-777 je pokročilejší štandard, ktorý je kompatibilný aj s ERC-20, ponúka možnosť ovládať operátorom tokeny aj na iných adresách.
- ERC-1155 je štandard, ktorý povoľuje kombináciu NFT a obyčajných tokenov vrámci jedného kontraktu [17].

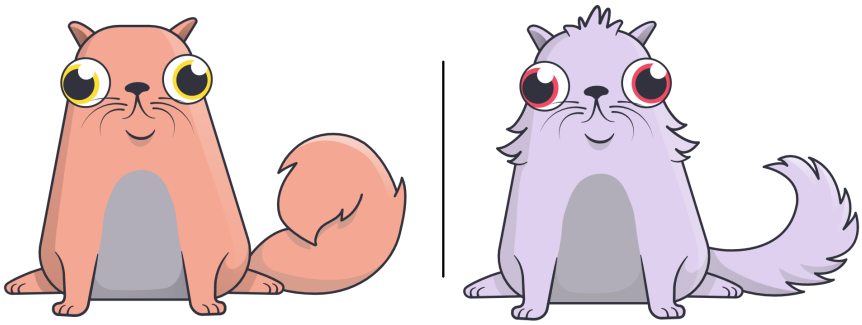
2.1.11 Gaming

Herný priemysel dnešnej doby sa stretáva s viacerými problémami, častý problém je potrebné nabiť si kredit na účet, kde sa hrá napríklad poker, ruleta a iné hry. Problém sa vyskytuje z dôvodu, že ak sa využíva platobná brána tretej strany ako je napríklad Paypal, Visa, Stripe a iné, často dochádza k momentom, kde povolenie a prejdenie transakcie trvá dni. Avšak ďalším a to hlavným problém je nedôvera v dané stránky, z dôvodu rizika podvádzania, možnosti krachu alebo nevôle vyplatiť vyhrané prostriedky. Ethereum v tomto segmente prináša riešenia a zaručuje, že pomocou open-source kódu sú hry férové podľa jasných pravidiel a nikto nemá

ako ovplyvniť výsledky, výplaty môžu prebiehať okamžite priamo cez blockchain Ethereum siete alebo vrámci state kanálov, avšak všetko toto zo sebou nesie časté a nemalé poplatky, čo môže od používania Ethereum siete odrádzať [18].

2.1.12 NFT

NFT je skratka pre non fungible token, z predošlých informácii možno nebolo jasné, ale v prípade, že máme 1 ethereum, ten pošleme preč, neskôr nám niekto iný pošle 1 ethereum, tak sme na tom rovnako, pretože každé ethereum je zameniteľné za iné, čiže je jedno, ktorú mincu presne máme. Avšak pri NFT je tento rozdiel dôležitý, pre príklad si uvedieme, že použijeme jednu z prvých kolekcií NFT cryptokitties. V prípade, že pošleme niekam našu aktuálnu cryptokittie a neskôr dostaneme inú cryptokittie, nemáme rovnakú ale úplne inú a sú nezameniteľné. Na obrázku 2.1 môžeme vidieť grafickú reprezentáciu NFT tokenov cryptokitties, naľavo je povedzme náš pôvodný token a napravo je token, ktorý nám potom poslal niekto iný. Existujú už viaceré blockchainové NFT hry mimo cryptokitties, kde vznikajú nové mačky / tokeny pomocou kríženia dvoch a zdedia určité prvky z pôvodných tokenov. Najznámejšie sú cryptopunks, meebits, axie infinity, gods unchanged. NFT tokeny nesú viaceré vlastnosti, najčastejšie sú to, že vlastník alebo tvorca tokenu si často berie províziu z každého predaja pri smart kontrakte daného tokenu. Vo výsledku to znamená, že ak tvorca vytvorí napríklad 10 tisíc kusov tokenu, všetky sa vypredajú a niekto ich bude následne predávať za sebou stanovenú cenu, majiteľ automaticky dostane vopred zvolené percentá z predaja napríklad 5%. Medzi ďalšie vlastnosti patrí napríklad to, že majitelia nejakého tokenu vedia dokázať svoje vlastníctvo tohoto tokenu a dostať nejaké extra funkcie alebo prístup niekam [19].



Obr. 2.1: Príklad rozdielnych NFT tokenov

2.2 Existujúce riešenia

Pre kvalitný návrh našej platformy a aplikácie, je dôležité nájsť vhodné existujúce riešenia, ktoré používajú blockchainové overovanie stavov, state kanály pre hry.

2.2.1 Ethex.bet

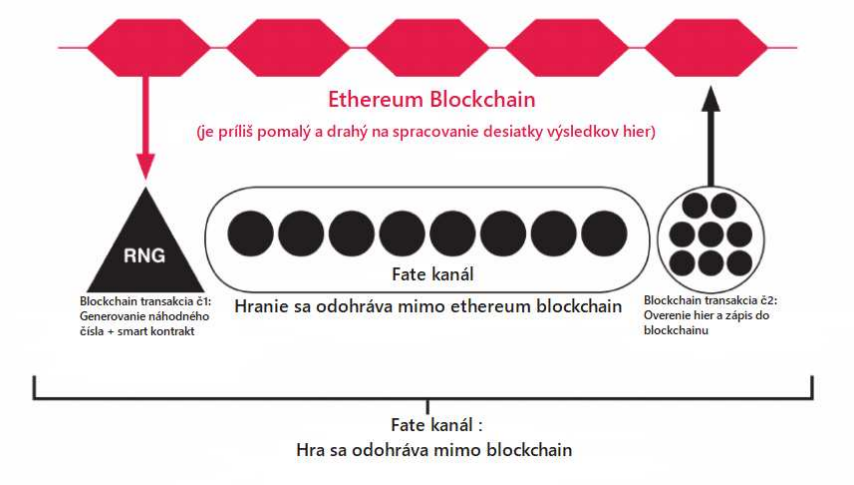
Ethex je Ethereová smart kontrakt lotéria s jednoduchými pravidlami, kde úlohou je úhadnuť 1 až 6 posledných znakov zo 16-kovej sústavy 0-9, A-F blocku v ktorom sa uzavrie smart kontrakt. Obsahom ich práce na github sú viaceré kontrakty, avšak hlavný, ktorý rieši logiku overovania výhry pracuje relatívne jednoducho. Pridá do radu číslo bloku, stavenú čiastku, id transakcie, stávku - tým rozumieme pole, v ktorom sú na pozícii 0 až 5 umiestnené znaky z 16-kovej sústavy. Následne hex bloku konvertuje pomocou shiftovania o 4 a AND s hodnotou 0x0F. Ku koncu prebehne podmienkami a v prípade zhody pridá zhodu do počítadla. Webové rozhranie platformy na adrese ether.bet ponúka demo rozhranie, pričom nie je nutné využívať prepojenie web3 na demo verziu ale v prípade, že nechcete demo, môžete importovať, vytvoriť wallet alebo prípadne prepojiť cez web3 roz-

hranie pomocou metamask. Minimálna stávka je v kontrakte nastavená tak, že za každý znak, ktorý hádate sa zvyšuje minimálna stávka o 0.01 ETH, to znamená, že v prípade, že tipujete 1 znak minimálna stávka je 0.01 ETH, a v prípade 6 znakov je stávka minimálne 0.06 ETH. Vďaka Ethereum consensus mechanizmu je nemožné vedieť aký hash bude mať samotná transakcia alebo blok pred samotným vykonaním transakcie alebo vytvorenia bloku. V prípade, že sa transakcia nepotvrdí do 256 blokov od začatia zruší sa z dôvodu toho, že v tomto prípade EVM má limitáciu na dostupnosť iba posledných 256 hashov [2].

2.2.2 Funfair

Funfair je herná platforma zameraná na hranie kasína a iných hier. Je vybudovaná a implementovaná na sieti Ethereum. Prínosom v rámci decentralizovaných financií a tokenizácie bolo to, že platforma mala vlastný token FUN, ktorý slúžil na lepšie fungovanie na stránke, to znamená samotné stávkovanie, vyplácanie. Z ich analýz bolo jasné, že v prípade, že ľudia pri hre blackjacku priemerne stávkujú sumy okolo doláru, tak je nutné iné riešenie ako hrať priamo onchain aj z dôvodu, že nie každému sa chce čakať jeden až niekoľko blokov na výsledok. Ich riešením je vytvorenie fate channelu, je to v podstate layer 2 riešenie. Fate channel je spôsob riešenia, kde otvárajú state channel. Na obrázku 2.2 vidíme ako to vlastne vyzerá v realite. Namiesto n zápisov do blockchainu, pričom n je väčšie ako 0 až nekonečno, sa vykonajú iba 2 zápisy do blockchainu. Z tohoto sa dá prepokladať, čím viac hráč hrá v danom off-chain kanál, tak tým menšie poplatky platí. Výhodou riešenia sú ušetrené poplatky sieti, zníženie zaťaženia siete, a daľšou výhodou, že všetky hry sú vyhodnotené okamžite a nie je potrebné čakať na zaradenie do bloku [3].

[2] https://ethex.bet/faq
[3] funfair.io/wp-content/uploads/FunFair-Commercial-White-Paper.pdf



Obr. 2.2: Príklad Funfair off chain state channelu [a=]
[a=] The footnote-like comment under the caption

2.2.3 Sunrise casino DAO

Sunrise je platforma na hranie gamblingových hier, pričom sa snažia zamerať na poker, blackjack a viac stolových hier. Z ich návrhov riešenia vyplýva to, že ich hlavný problém sa týkal tokenizácie a jej ekonomiky, narážali na to ako dať hodnotu svojim tokenom. Ich cieľom je vytvoriť token podobný akciám, kde ani po vystupení ceny veriaci v ekosystém nepredávajú za krátkodobý finančný profit, lebo vedia, že samotný token je ten projekt. Iný problém, ktorý sa naskytol je problém pseudo-random funkcie, ktorý nevyriešili. Pôvodný návrh bol implementovať a bežať projekt na Ethereum blockchaine. Návrh je nakoniec teda na privátnom blockchaine, ktorý bude komunikovať s web3, pričom ich token bude SUSD, ktorý bude swapovaný k tokenu USDT (Tether) - stablecoin viazaný k cene 1 doláru. Voľba tohoto blockchainu bola ovplyvnená aj vysokými poplatkami na Ethereum sieti a nemožnosti prekonať spomínané problémy [4].

[3] funfair.io/wp-content/uploads/FunFair-Commercial-White-Paper.pdf
[4] `sunrisegaming-dao.com/pdf/Release_SUNRISECASINO_by_DAO_WHITEPAPER_v1.pdf`

2.2.4 Stixex

Stixex je herná platforma, ktorá ponúka možnosť hrať iba jednému hráčovi pričom stávkuje na kurz obchodovaného páru. Prvý problém v tejto myšlienke je to, že stávkovanie na kurz komodity, konkrétne ethereum v pomere s americkým dolárom USD je nutné čerpať z centralizovaného riešenia, ktoré ponúka API. Problém nastáva v momente, keby sa z nejakého dôvodu táto centralizovaná služba zastavila alebo by posielala nesprávne údaje. Vyzdvihnúť sa na tejto platforme dá to, že pre jednoduchosť využívajú offchain transakcie bez poplatkov a majú na to vytvorený state channel a tiež transparentnosť smart kontraktu, ktorý zabezpečuje férovosť. Stránka komunikuje pomocou web3 na Ethereum blockchain-e, neponúka možnosť registrácie. V rámci tokenizácie a fungovania pracujú s USDT, tiež Tether, ktorý má stabilnú hodnotu voči americkému doláru USD [5].

2.2.5 Degens

Herná platforma určená najmä na stávkovanie zápasov pomocou mechanizmu hráč proti hráčovi. Majú vlastný token DEGENS, ktorý sa využíva na viaceré funkcie v rámci ich platformy. Stávkovanie umožňujú cez ethereum alebo prípadne token DAI, ktorý je naviazaný na cenu amerického dolára USD. Vrámci vyhodnocovania zápasov využívajú viaceré prvky offchain metód. Avšak každá stávka obsahuje otvorenie aj zatvorenie takéhoto kanálu z dôvodu, že stávky nie sú nadväzujúce a nešetria tým poplatky, ale vyhodnucujú tak zápasy, pričom samotné interakcie nestoja gas a ušetria tak určité poplatky pri vyhodnocovaní zápasov a zvýšia si tak súkromie. Čo sa týka párovania hráčov, pri stávke vytvoríte bet a niekto sa k vám musí pridať a staviť na opačný výsledok. V prípade, že sa tak nestane stávka sa neuskutoční. Na stránke po prihlásení môžete vidieť aktuálne čakajúce hry, ktoré chcú nájsť hráča aby stávka mohla začať. Komunikácia je zabezpečená pomocou

[5] stixex.io/

web3, a férovosť je zabezpečená verejným smart kontraktom [6].

2.3 Zhrnutie jednotlivých platforiem

Porovnávané platformy priniesli viaceré slabosti ale aj viaceré silné stránky. Vzhľadom na môj projekt je najvhodnejšia platforma Funfair, ktorá mala využité viaceré prvky zvládnuté veľmi dobre. Vyzdvihol by som ich vlastný off-chain kanál, ktorý slúži na ukladanie výsledkov hry, ktorý sme v prípade Stixex platformy mohli nájsť iba pre výbery a vklady kapitálu, kde išlo o veľmi podobnú implementáciu ale s problémami centralizácie API a mimo blockchainových riešení. Platforma FUNFAIR, aj napriek offchain riešeniu, ponúka možnosť tokenizácie, veľký počet hier. Ostatné platformy mali viaceré problémy medzi ktoré môžeme vyzdvihnúť u platformy Degens nie úplne transparentné overovanie výsledkov, ja to označujem za značnú centralizáciu, a označujú to tak aj samotní tvorcovia Degens a ospravedlňujú to tým, že ak by nevyhodnocovali správne výsledky, prestali by ich používať a nikto by u nich nehral. V rámci sunrise casino je chyba to, že ste takmer nútený využívať ich vlastný token bez takmer možnosti hrať priamo s ethereum mincami, kde by ste si znížili poplatky a aj by ste mohli mať vyššiu dôveru v hodnotu mince.

Názov	Ethex.bet	Funfair	Sunrise casino	Stixex	Degens
Blockchain	ETH	ETH	ETH,private chain	ETH	ETH
Whitepaper	No	Yes	Yes	No	No
Tokens	ETH	ETH,FUN	ETH,SUNC	ETH	DAI
Provably fair	Yes	Yes	No	Yes	Yes
Decentralized	Yes	Yes	Yes	Yes	No
Offchain	No	Yes	No	Yes	Yes

[6] degensprotocol.github.io/degens-contract/protocol.html

Kapitola 3

Funkcionálne a nefunkcionálne požiadavky

Zoznam požiadaviek:

Funkcionálne požiadavky:

1. Systém zabezpečiť možnosť autentifikácie ako hráča - urobí tak cez web3 interakciu
2. Systém musí zabezpečiť komunikáciu medzi hráčmi - táto komunikácia sa vytvára pomocou kanála špecifického pre jednotlivé hry. Prebieha cez Pubnub API.
3. Systém musí zabezpečiť otvorený stavový kanál hry počas celého trvania hernej relácie
4. Používateľ bude mať možnosť si vybrať výšku stávky
5. Rozhranie umožní používateľovi vytvárať herné relácie
6. Rozhranie umožní používateľovi pridávať sa do herných relácii

Nefunkcionálne požiadavky:

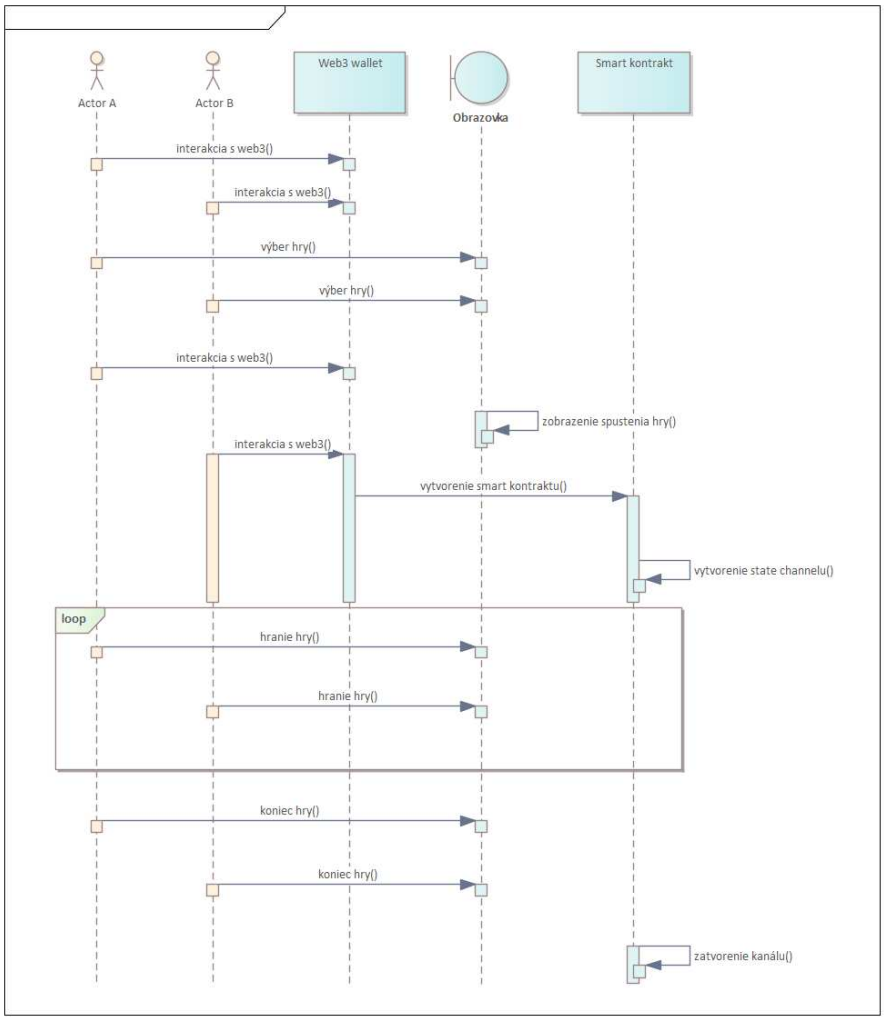
1. Aplikačné alebo webové rozhranie - okno alebo skupina okien, ktorá zobrazí používateľovi dostatočne zrozumiteľné grafické vyobrazenie na volanie a spúšťanie metód smart kontraktu, slúži taktiež na hranie samotných hier
2. Kompabilita prehliadačov Brave a Google Chrome
3. Kompabilita web3 rozhrania peňaženky Metamask
4. Ušetrenie poplatkov pri používaní smart kontraktu a state kanálu

Kapitola 4

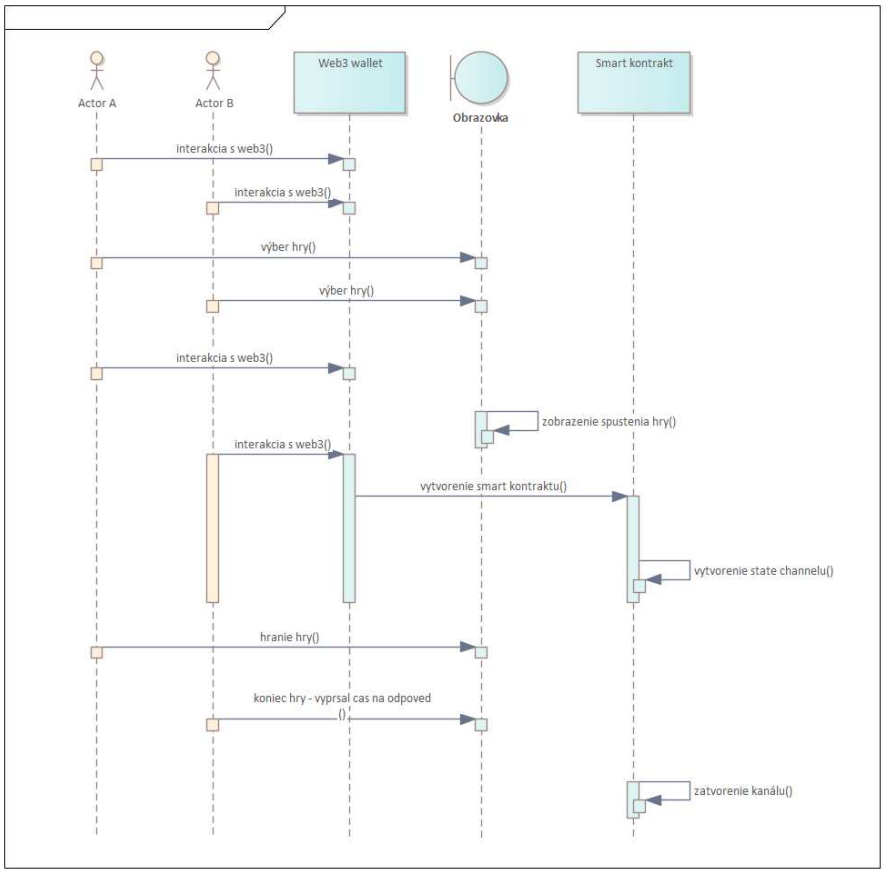
Návrh riešenia

Cieľom mojeho projektu je navrhnúť hernú platformu, kde existuje niekoľko hier vrámci blockchainu. Snaha je navrhnúť viaceré druhy implementácie, poukázať na ich výhody a nevýhody. Vo výsledku je snaha implementovať 2 druhy hier. Jedna hra overuje platnosť stavov pomocou podpisovania jednotlivých state kanálov, kde obaja hráči podpisujú 2 posledné stavy a druhá hra obsahuje overovanie pomocou merkle hash stromu. Snaha bude poukázať na výhody druhej vrstvy, ktorá ponúka vysokú škálovateľnosť, viac možností vrámci implementácie a v neposlednom rade ušetrí poplatky v priamej úmere s počtom krokov a dĺžky samotnej hry. Prvotný sekvenčný diagram predpokladá s 2 hráčmi, hráč A a hráč B, ktorý si zapnú webové rozhranie aplikácie vyberú si hru a v prípade spojenia začnú hrať. Hranie nie je nijako časovo obmedzené, avšak bude rozumné nastaviť si určitý časovy timeout, v prípade ak by jeden hráč prestal hrať počas hry alebo by mu iné okolnosti nedovolili ďalej hrať. Čo sa týka otvárania kanálu, je nutné otvoriť ho až po pridaní sa dvoch hráčov, v prípade otvárania kanálu pri jednom hráčovi dochádza vrámci implementácie k viacero problémom, čo ak k hráčovi A nedokážeme nájsť oponenta a otvoril si kanál, bude ho musieť zavrieť s 0 zahranými hrami, čo nedáva zmysel.

Každý hráč by mal mať možnosť hocikedy ukončiť svoju hru a tým uzavrieť hraný kanál.



Obr. 4.1: Sekvenčný diagram zobrazujúci hráčov hrajúcich hru bez problémov.



Obr. 4.2: Sekvenčný diagram zobrazujúci hráčov hrajúcich hru s problémom.

Spájanie hráčov

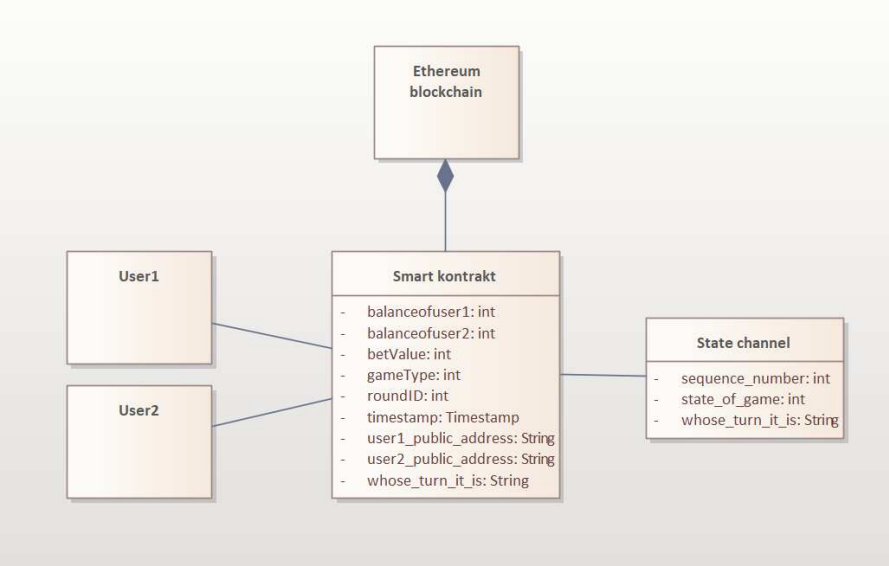
V rámci návrhu sa vyskytol prvotný problém toho, že ako spájať hráčov. Riešenie tohoto problému sa dá poňať rôzne, jedno s najjednoduchších riešení je spájanie random. V našom prípade ale chceme zachovať možnosť vybrať si oponenta, a preto je lepšie ak existuje list, v ktorom je zoznam všetkých aktívne hľadajúcich hráčov čakajúcich na oponenta, kde sa hráči môžu pridať a hra môže začať. Rozlišovanie hráčov sa dá pomocou verejných klúčov, tieto verejené kľúče má každý unikátne, z toho dôvodu vieme povedať, že každý hráč bude mať iné označenie.

Smart kontrakt

Smart kontrakt, ktorý plánujem využívať v rámci mojich hier bude musieť obsahovať viaceré smerodajné údaje. Prepokladám, že väčšina hier je medzi dvoma hráčmi, preto je nutné aby som si zachovával podstatné údaje o týchto dvoch hráčoch. Konkrétne informácie sú kredit hráča 1 a hráča 2, ich verejné adresy na rozlíšenie, ktorý je ktorý, v neposlednom rade to, kto je na rade. Čo sa hry samotnej týka, je potrebné vedieť aké číslo kola sa vykonáva, každý krok protihráča môžeme brať ako jedno kolo a to iteratívne zvyšujeme od čísla 0. Ďalej je potrebné určiť akú hru hráme, pretože vrámci platformy ich môže byť viac a je potrebné rozlišovať ich z dôvodu zmeny pravidiel, počtu hráčov, maximálneho počtu kôl a podobne. Ukladáme si taktiež timestamp, ideálny čas je čas zapísania do bloku alebo krátko predtým, pričom nastavíme vhodnú dobu na uzatvorenie kanála ak by sa niektorí z hráčov alebo obaja rozhodli prestať hrať počas hry. V rámci tokenizácie a hry je podstatné zapísať si aj stávku, alebo jej hodnotu napríklad 0.0754814 ethereum. Poslednú vec, ktorú by mal obsahovať smart kontrakt je informácia o statuse hry, status číslom 0 je ukončená bežne, 1 ukončená vypršaním času, 2 ukončená užívateľom, 3 prebieha.

State kanál

Všetky potrebné zmeny spomínané v kapitole smart kontrakt bude nutné zapisovať do state kanálu z dôvodu rýchlosti, ceny a flexibility riešenia. Samotný stav, bude definovaný ako štruktúra, ktorá obsahuje verejnú adresu toho, kto je na rade, sekvenčné číslo, a stav hry, pričom ostatné môže byť zapísané vrámci blockchainu, z dôvodu, že ide prevažne o konštanty. Ukončenie hry prebehne zavretím tohoto kanálu a zapísanie stavov hier pomocou transfer(adresa(this).hodnota).

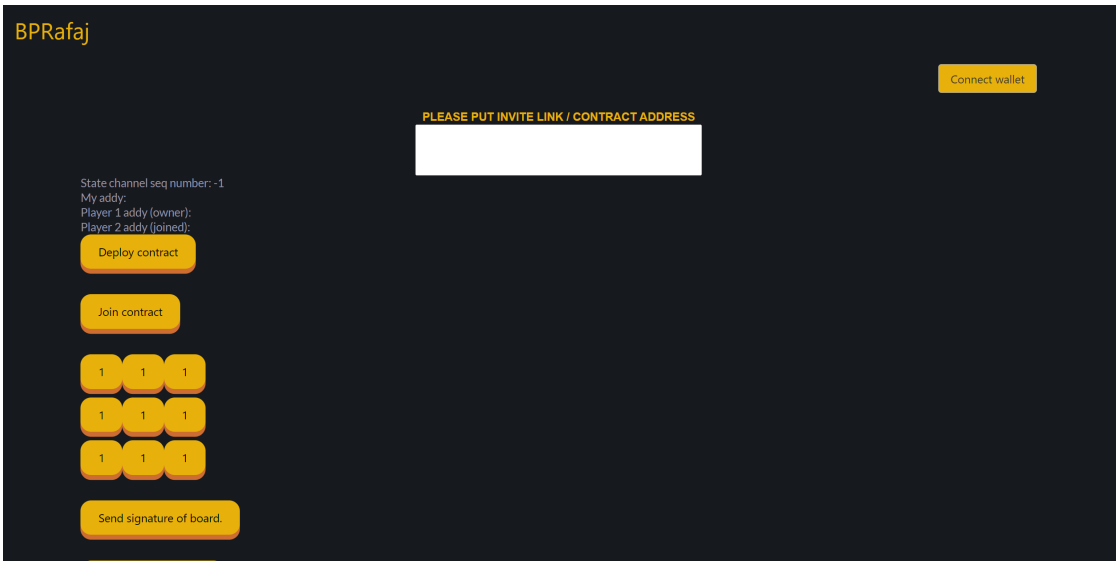


Obr. 4.3: Diagram tried opisujúci triedy potrebné v rámci implementácie

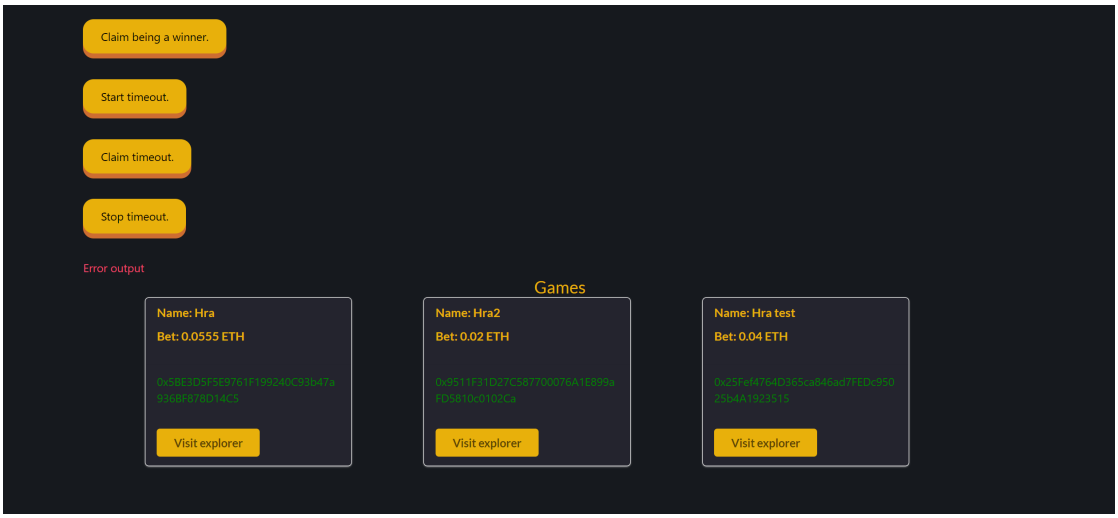
Kapitola 5

Implementácia

Samotná implementácia bola realizovaná pomocou react aplikácie, kde sa využívala testovacia sieť Rinkeby. V rámci vzhľadu som využíval bulma css, ktorá dodala určitý priateľský vzhľad aplikácii. React aplikáciu som si zvolil z dôvodu, že je vhodná pre ukážku decentralizovaných aplikácii a ponúka vhodnú funkcionalitu a prepojenia s knižnicami ako je web3. Výsledok práce som si rozdelil na 2 hlavné časti, prvá časť je front-end, kde sa nachádza kompletný javascript, css, knižnice, samotná react aplikácia a druhá časť blockchain, kde je kontrakt. Kontrakt som kontroloval a implementoval cez rozhranie Remix IDE, front-end cez Visual Studio IDE, avšak celý front-end a back-end sa neskôr spúšťali cez node.js. V rámci komunikácie medzi prehliadačom alebo hráčmi som využil api služby Pubnub, ktorá poskytuje komunikačné kanály zadarmo do určitej priepustnosti. Hra a interakcia stránky prebieha pomocou webového rozhrania, ktoré môžeme vidieť na obrázkoch č. 5.1. a 5.2. . Vytváranie kontraktov prebieha taktiež cez webové rozhranie, ktoré môžeme vidieť na obrázku č. 5.3. .



Obr. 5.1: Výsledne GUI hlavnej stránky časť 1



Obr. 5.2: Výsledne GUI hlavnej stránky časť 2

Obr. 5.3: Výsledne GUI deploy stránky

5.1 Použité technológie

Vrámci frontend aplikácie sa využilo viacero knižníc a jazykov. Medzi hlavné patria node.js, react, a javascript. Taktiež prvky html, css, ktoré dávajú vzhľad a vyobrazenie webovej stránky. Využitá bola konkrétne next-js react aplikácia, ktorá ponúka pre renderovanie jednotlivých stránok, podpora fast refresh funkcionality, ktorá počas implementácia bola extrémne nápomocná a ušetrila veľa času. Moja aplikácia disponuje aj prepojením na databázu mongoDB ale vrámci implementácie sa to využívalo na ukladanie adries vytvorených kontraktov, respektíve hier a popisu k nim. Na základe týchto zápisov sa vedia hráči spájať.

Vrámci komunikácie medzi klientami sa využívala už spomenutá Pubnub knižnica. Táto knižnica ponúka API pubnub aplikácie, ktorá dokáže spájať viacero klientov najrôznejších zariadení a zdieľať tak správy. Na back-ende je napojený pubnub kanál, ktorý počúva zdieľané správy na kanál s jeho kódom, kód je hash kontraktu, čiže sa počúvajú vždy dvaja hráči, ktorí hrajú spoločnú hru (jeden kontrakt).

Vrámci programovania blockchain aplikácie som využíval jazyk Solidity, ktorý patrí

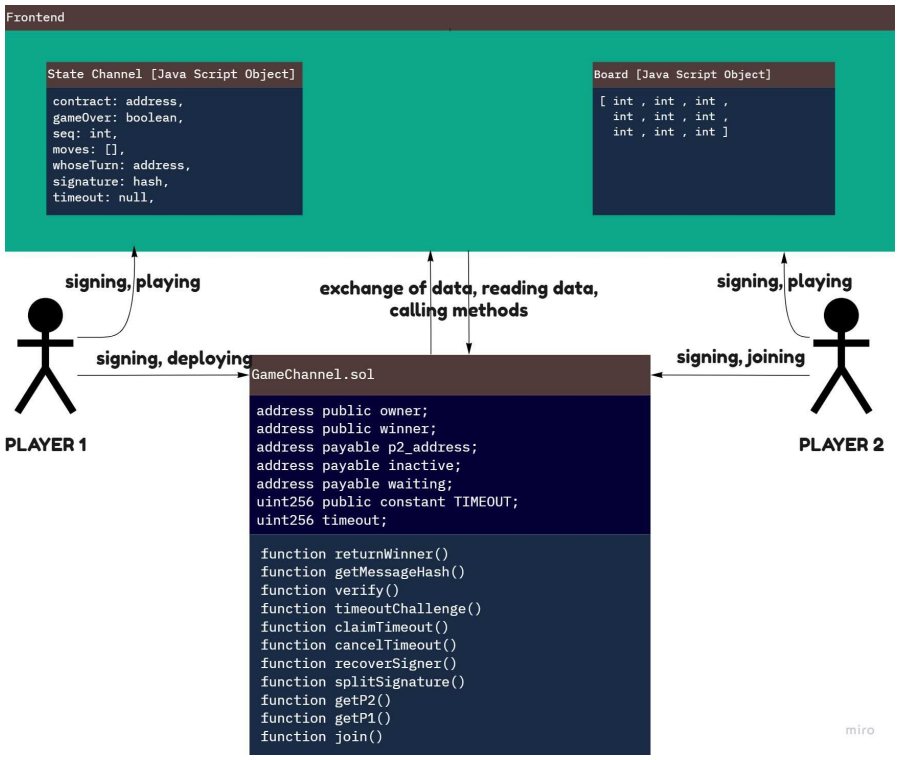
medzi najpopulárnejšie vrámci vývoja ethereum aplikácii. Pri testovaní som využíval technológie hardhat vývojové prostredie, web3 knižnice, ganache aplikáciu, ktorá poskytuje lokálny ethereum blockchain s privátnymi kľúčmi. Hardhat prostredie poskytuje jasné výsledky testov, časové merania, merania spotreby. Spustenie a testovanie je veľmi používateľsky prívetivé.

5.2 Zmeny oproti návrhu

Pri samotnej implementácii došlo k viacerým zmenám voči návrhu, medzi hlavnú zmenu patrí vnútorná štruktúra smart kontraktu, ktorý obsahuje značne menšie množstvo dát a nepracuje priamo so state kanál ale iba s hernými plochami, čím sa ušetril potrebný výkon na vyhodnotenie a nepriamo aj poplatky hráčov, ktoré by museli platiť. Kompletná štruktúra je reprezentovaná pomocou obrázka č. 5.4.
.

Spájanie hráčov

Spájanie hráčov prebieha pomocou výberu jednotlivých adries kontraktov, ktoré sú reprezentované ako okná, ktoré možno vidieť na obrážku č 5.2. . Hry sú zobrazené na spodnej časti hlavnej stránky, kde vidno kód hry, názov hry a stávku danej hry v ETH. Následne si môže hráč vložiť kód hry do okna s nápisom PLEASE PUT INVITE LINK / CONTRACT ADDRESS a následne stlačí Connect wallet, v prípade ak už to spravil, iba opätovne zatlačí tlačidlo svojej adresy, čo načíta údaje kontraktu, uvidí zmeny vrámci P1(owner) a prípadne aj P2(joined) ak už hra začala. Ak hra nemá najdeného druhého hráča, hráč sa môže pripojiť pomocou Join contract. Vrámci sekvencií samotnej hry bez problému alebo s problémom nedošlo k prakticky žiadnym zmenám.



Obr. 5.4: Diagram tried opisujúci objekty vrámci hry.

5.3 Opis react aplikácie - front-end a back-end

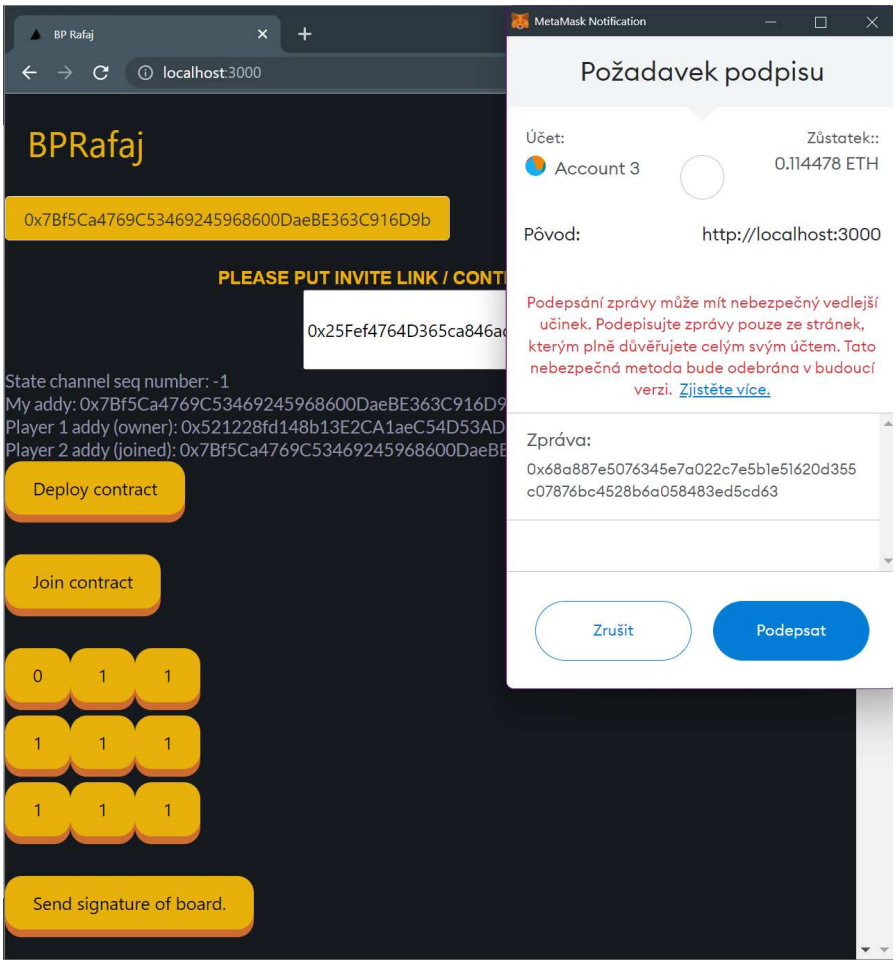
Vrámci front-end aplikácie som implementoval viacero funkcií. Začal by som v podstate tou, že bolo nutné spracovať web3 prepojenie samotného hráča. To znamená, že predpokladáme, že hráč má peňaženku alebo doplnok metamask a má na ňom napojenú sieť Rinkeby. Testované boli prehliadače Brave a Google Chrome. O túto funkcionalitu sa stará funkcia s názvom *connectWalletHandler*, ktorá spracováva požiadavku prepojenia web3 účtu a našej aplikácie. V prípade úspešného prepojenia sa text na front-ende zmení z "Connect wallet"na samotnú verejnú adresu pripojeného hráča. Ďalej máme 2 funkcie, ktoré súvisia s tým, že aby hráč mohol začať hrať hru alebo sa k nej pridať, potrebuje takéto informácie najprv zapísať do

blockchainu, pravidlá hry, výška stávky, hráči a podobne. Preto funkcia s názvom *deployContract* po zavolaní vyvolá web3 okno, kde je žiadosť na vytvorenie kontraktu s údajmi, ktoré sú predpísané vrámci back-endu. Defaultne sú nastavené nasledovné hodnoty: stávka 0.02 ethereum, majiteľ konraktu je samotný tvorca, volajúci funkciu *deployContract*. Druhou nadväzujúcou funkciou k *deployContract* je funkcia nesúca názov *joinHandler*, ktorá očakáva, že už je kontrakt aktívny a nasadený na sieť. Táto funkcia odhaduje a očakáva, že druhý hráč B zavolá funkciu *joinHandler*, pričom prvý hráč A deployoval kontrakt a pripojí sa tak do hry. Po samotnom pripojení druhého hráča B, to znamená že funkcia join prebehla správne a bola zapísaná do blockchainu vrámci kontraktu, je otvorený kanál a hra sa môže začať. Hru začína vždy hráč A, a striedajú sa až kým jeden hráč nepríde do konečného stavu, to berieme stav kedy jeden z hráčov vyhrá. S remízou nepočítame. V prípade, že jeden hráč vyhrá, druhý hráč už nebude môcť vykonať nový krok ale očakáva sa, že zavolá funkciu *sendBoard*, ktorá pošle podpísanú dosku, kde očividne prehral, čím potvrdí prehrávajúci stav a výhernú dosku máme podpísanú od oboch hráčov. V tomto momente sa očakáva, že vyhrávajúci hráč zavolá funkciu *claimWinner*, kde pošle podpísanú finálnu dosku obama hráčmi, podpisy oboch, a samotnú plain text dosku. V prípade, že kontrakt vyhodnotí všetko ako platné, bude mu po zavolaní tejto funkcie automaticky vyplatená výhra mínus poplatky siete.

Je vhodné spomenúť, že front-end aplikácia ráta aj s tým, že v prípade ak zahrá každý hráč aspoň jeden krok môže sa stať, že hráč odíde. Môže vopred vidieť, že už prehral alebo mu vypadne spojenie, či môže mať iný dôvod opustenia hry. V tomto prípade sa implementovala funkcionalita pomocou funkcií : *timeoutChallenge, claimTimeout, cancelTimeout*. Tieto funkcie slúžia na možnosť zavolať timeout výzvu na hráča, ktorý prestal hrať alebo neodpovedá počas hry. Na zavolanie tejto funkcie je nutný zápis do blockchainu. Scenár je nasledovný : Hráč A alebo

B zavolá na druhého hráča funkciu *timeoutChallenge*, bežný timeout je nastavený na 120 sekúnd (2 minúty) od zápisu zavolania tejto funkcie do bloku. Pubnub knižnica, chytá aj takýto druh správy a hráčovi, ktorý dostal timeout challenge je okamžite zobrazené upozornenie, že je na neho zavolaný timeout. V prípade ak ide o planý poplach, hráč môže zavolať funkciu *cancelTimeout*, krorá kontroluje, či nevypršal čas. Ak čas nevypršal, timeout sa zruší. Ak vypršal, dochádza k revertu a nevykonaniu funkcie vrámci kontraktu. V prípade ak hráč, ktorý mal zrušiť timeout tak nevykoná. Výherný hráč si môže svoju výhru vyzdvihnúť pomocou volania funkcie *claimTimeout*, ktorá vyhodnotí, či to hráč, na ktorého bola výzva timeoutu stihol zrušiť alebo, či nereagoval a tým pádom challenger vyhral.

Vrámci celej hry je zdieľaný state kanál, môžeme ho volať aj herný kanál. Tento kanál slúži ako komunikácia medzi dvoma uzlami, v našom prípade hráč A a hráč B. Herný kanál obsahuje : adresu kontraktu, gameOver status, číslo kroku, jednotlivé kroky, kto je na rade, podpis a timeout. Kanál je zdieľaný počas každého nového kroku a iteruje sa, je to veľmi dôležitá súčasť implementácie. Avšak pre ušetrenie zápisov do blockchainov sa tento kanál neposiela do blockchainu ale posiela sa iba herná plocha s podpismi, ktorá nám stačí na rozoznanie výhercu. Počas hrania sa očakáva, že hráč nerefreshuje stránku ani nevypína a nezapína prehliadač, pretože kanál sa drží iba v pamäti prehliadača oboch hráčov. Klasická databáza bola proti myšlienke blockchainovej technológie a bezpečnosti, a preto sa táto možnosť ukladania herného kanála nevyužila. Vrámci hrania sa vyžadujú podpisy každého kroku, ktoré môžeme vidieť na obrázku č. 5.5. .



Obr. 5.5: Podpisovanie.

5.4 Smart kontrakt

Kontrakt som testoval na 2 verziách solidity, verzia 0.8.4 a 0.8.11 . Snaha bola vytvoriť, čo najkratší a funkčný kontrakt, ktorý zabezpečí bezpečné hranie ale zároveň aj ušetrí používateľovi malú až väčšiu časť poplatkov. Medzi základné premenné definujeme viacero adries ako owner, winner, player2 address, inactive player, waiting player a konštantu timeoutu.

Transakcia returnWinner

Transakcia returnWinner poskytuje funkcionalitu vracajúcu aktuálny stav výhercu. Predpoklad je, že ak hra nebola ukončená alebo je v stave hrania tak táto funkcia vráti hodnotu null adresy 0x0 inak vráti adresu hráča 1 alebo 2.

```
1     function returnWinner() public
2         view
3         returns (address){
4             return winner;
5     }
```

Kód 5.1: Funkcia returnWinner()

Transakcia getMessageHash

Transakcia getMessageHash poskytuje funkcionalitu vracajúcu hash argumentu, ktorý príde ako vstup. Ako argument očakáva hraciu dosku, ktoré hráči vykonali. Vracia Keccak256 hash hodnotu, ktorá sa neskôr využíva na podpisovanie hráčmi.

```
1     function getMessageHash(uint[] calldata _num) public
2         pure
3         returns (bytes32) {
4             return keccak256(abi.encode(_num));
5     }
```

Kód 5.2: Funkcia getMessageHash()

Transakcia verify

Transakcia verify obsahuje najhlavnejšiu funkcionalitu kontraktu a tou je to, že dokáže deklarovať výhercu, overiť výsledok a aj vyplatiť výhercu. Ako prvý argument volanej funkcie prichádza pole _num, ktoré by malo obsahovať nezahashovanú hraciu dosku, následne druhý argument _board by mal byť keccak256 hash tejto

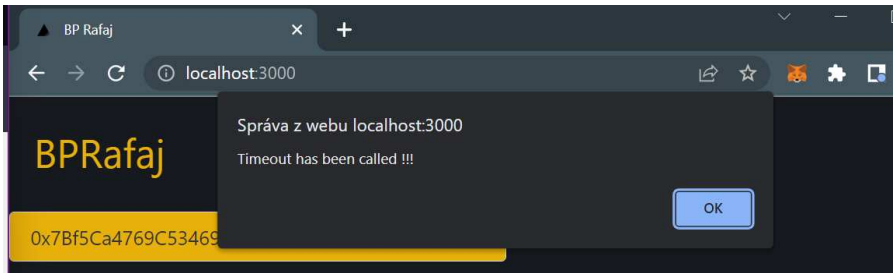
hracej dosky, argumenty _signature1 a _signature2 sú podpisy _board hashu. Táto funkcia by mala obsahovať špecifické pravidlá pre jednotlivé hry, kde sa dá overiť výherca z poskytnutej dosky. Funkcia by mala taktiež obsahovať v prípade finálneho stavu hry na doske a platných podpisov tejto dosky aj automatické vyplatenie výherciu a uzavretie kanála. Naša hra predpokladá kanál za uzatvorený v momente ako je deklarovaný výherca a je vyplatený.

```
1     function verify(uint[] calldata _num, bytes32 _board, bytes
2         memory _signature1, bytes memory _signature2) public
3         returns (bool) {
4             .... // telo funkcie
5     }
```

Kód 5.3: Funkcia verify()

Transakcia timeoutChallenge, claimTimeout, cancelTimeout

Transakcia timeout slúži ku komplementej obsluhe vyvolávania, rušenia a potvrdzovania timeoutov medzi hráčmi. Dôvod a potreba timeoutu je, že môže nastať situácia kedy sa jeden hráč rozhodne nehrať. Rozhodnúť sa tak môže z dôvodu videnia nevyhnutnej prehry už niekoľko krokov vopred, výpadok spojenia alebo z iných dôvodov. Druhý hráč by sa takýmto konaním dostal do takzvaného mŕtveho bodu. Hra mu preto dovoľuje zavolať timeout výzvu a ukončiť hru. Nadväzujúca funkcia *timeoutChallenge* očakáva aspoň elementárne dôkazy o tom, že hra prebiehala a to znamená, že očakáva aktuálny krok na doske podpísaný aktívnym hráčom a krok predtým s doskou podpísanú hráčom už neaktívnym, očakáva taktiež podpisy oboch týchto dosiek. V prípade ak je všetko v poriadku začne bežať timeout, bežne je nastavený na 120 sekúnd, ak si používateľ, ktorý prestal hrať nechal otvorený prehliadač alebo ide o planý poplach, okamžite sa mu zobrazí upozornenie, že na neho bol vyvolaný timeout.



Obr. 5.6: Upozornenie na timeout.

On ako hráč, ktorý je aktívny alebo si uvedomil, že už je načase hrať musí zavolať funkciu cancelTimeout, ktorá neberie žiadne argumenty a v prípade ak ešte neubehlo 120 sekúnd od začiatku timeoutu, daný timeout zruší a hra pokračuje ďalej. V prípade ak by sa neozval alebo by tento timeout zrušiť nestihol, hráč, ktorý vytvoril tento challenge môže požiadať o vyplatenie výhry a byť deklarovaný za výhercu pomocou zavolania metódy claimTimeout.

```
1     function timeoutChallenge(bytes32 _board, bytes32 _boardBefore,
2         bytes memory _challenger, bytes memory _challenged) public
3         returns (bool){
4             .... // telo funkcie
5     }
```

Kód 5.4: Funkcia timeout()

Transakcia recoverSigner, splitSignature

Funkcia recoverSigner je veľmi dôležitá funkcionalita asynchrónnych šifier, kde dokážeme zo správy a podpisu povedať, kto ich podpísal. To sa využíva v našom prípade tak, že ak hráč zdieľa hernú plochu so svojim novým krokom, zdieľa hash danej hracej plochy, následne podpis tejto hracej plochy a my na základe funkcie recoverSigner a splitSignature, ktorá rozdeluje daný hash hracej plochy na 3 časti dokážeme povedať, kto podpísal danú správu. Očakávame teda, že podpísaný je náš protihráč a výstupom bude jeho verejná adresa, ak tak nie je, tak sa snaží

podvádzať a ide o neplatný krok alebo podpis.

```
    function recoverSigner(bytes32 _ethSignedMessageHash, bytes
        memory _signature) public
        pure
        returns (address){
            .... // telo funkcie
    }
```

Kód 5.5: Funkcia recoverSigner()

Transakcia getP1 a getP2

Transakcia getP1 a getP2 poskytujú funkcionalitu vracajúcu aktuálnu adresu žiadaného hráča. Predpoklad je, že ak je hráč 2 null adresa 0x0, tak hra sa ešte nezačala. V opačnom prípade už sa hrala buď minimálne začala, prebieha alebo už skončila. Obe funkcie slúžia na jednoduché získavanie adries oboch hráčov hry a využívajú sa vrámci back-endu aj front-endu. Hráč 1 je zároveň aj tvorcom kontraktu.

```
    function getP1() public
        view
        returns (address) {
            return (owner);
    }
```

Kód 5.6: Funkcia getP1()

Transakcia join

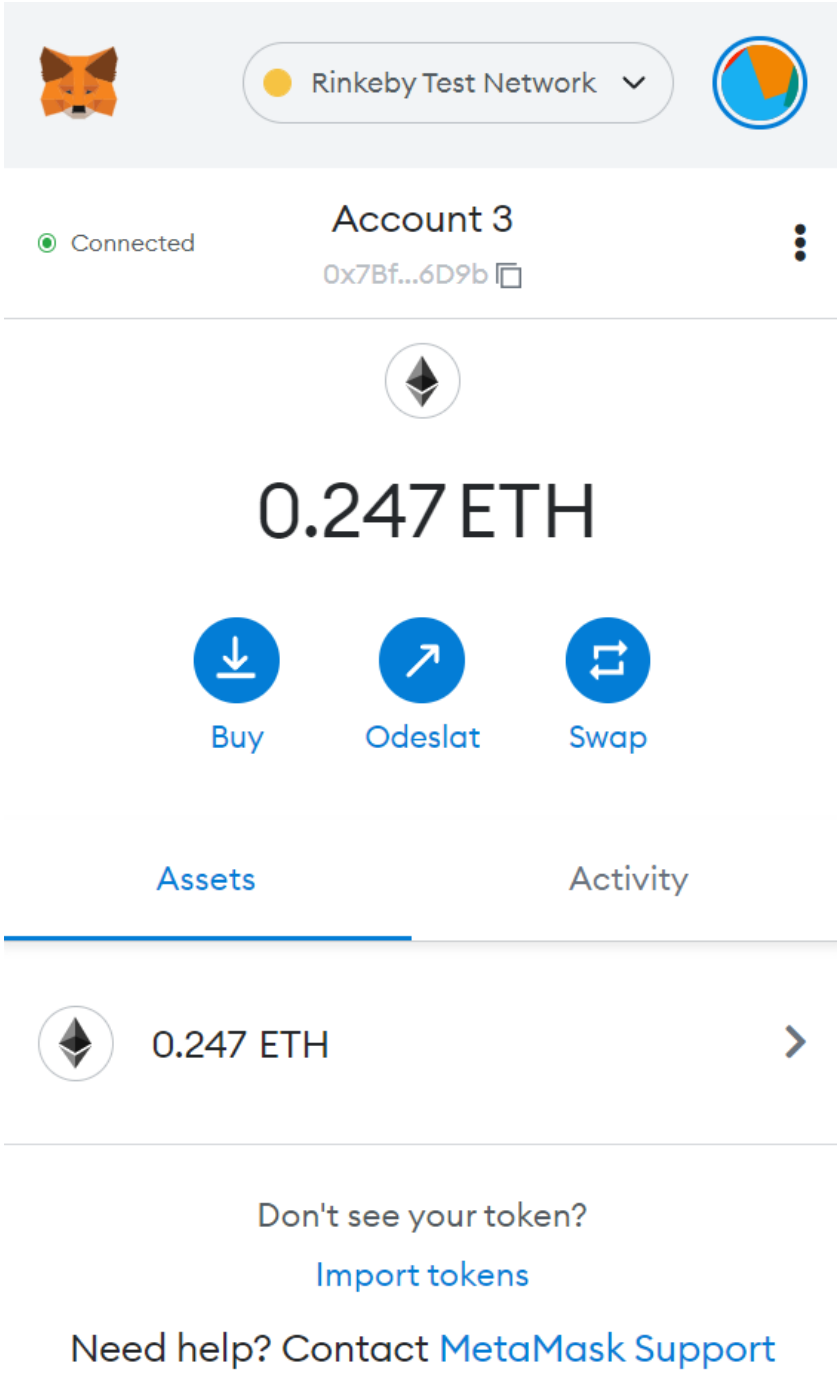
Funkcia join slúži na pripojenie hráča 2 do hry, očakáva sa, že vloží rovnaký vstup ako tvorca hry, hráč 1. Je dôležité aby sa výška stávky zhodovala so stávkou hráča 1, táto čiastka sa definuje pri vytváraní hry. V prípade ak už hráč 2 existuje, nie je možné sa pridať znova alebo prepísať aktuálneho hráča.

```
1    function join() public
2         payable {
3              require(msg.value == bet);
4              require(p2_address == address(0));
5              p2_address = payable(msg.sender);
6     }
```

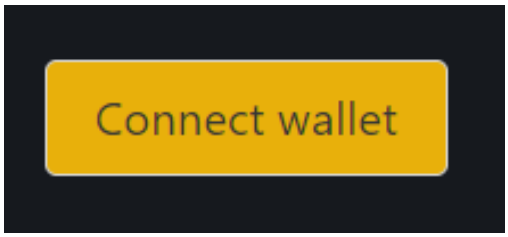
Kód 5.7: Funkcia join()

sectionInterakcia

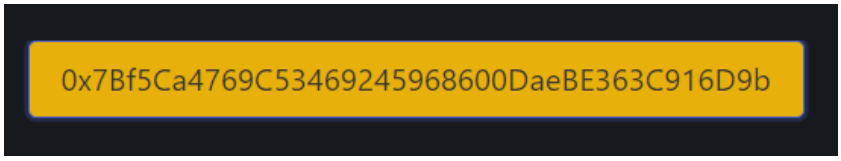
Po spustení node.js aplikácie a otvorení front-end stránky je dôležité pripraviť si odomknutú metamask peňaženku na sieti Rinkeby. Vyzerať by to malo ako na obrázku č. 5.4 . Následne pokračujem podľa scenára. Odhaduje sa, že konkrakt už bol nasadený na sieť a čaká na druhého hráča.



Obr. 5.7: Odomknutá metamask peňaženka pripojená na rinkeby testnet sieť.



Obr. 5.8: Pred pripojením metamask.



Obr. 5.9: Po pripojení metamask .

Ak som hráč, ktorý sa chce pripojiť do hry, postup je nasledovný:

1. Do baru v strede obrazovky vložím kód hry (adresu kontraktu),
2. Kliknem na tlačidlo Connect wallet,
3. Očakávam, že sa tlačidlo zmení na moju verejnú adresu,
4. Na základe načítaného kontraktu vidím, či už mám oponenta alebo nie. To, či oponenta mám, zistím na základe textu Player 2 addy (joined), kde uvidím verejný kľúč oponenta. V prípade ak je tam 0x0 respektíve null, hráča ešte nemám,
5. Druhý hráč si otvorí v inom prehliadači rovnakú aplikáciu, zadá kód hry a dá Connect wallet, následne stlačí tlačidlo Join Contract, potvrdí danú akciu v okne metamask,
6. Po úspešnom pridaní môžu obaja hráči refreshnúť stránku, obaja stlačiť Connect wallet a mali by vidieť stav hry, kde vidia oponenta aj tvorcu hry,

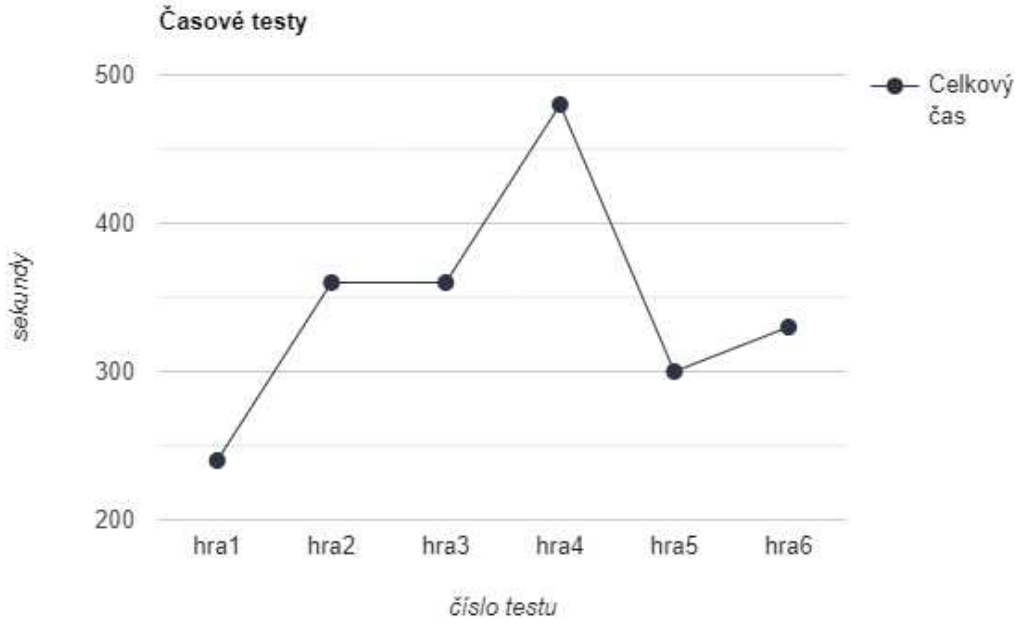
7. Následne môžu hrať a je dôležité aby už stránku nerefreshovali ani nezatvárali alebo sa nepokúšali otvárať ju viackrát,
8. Hrajú pomocou hracej plochy 3X3, kde sú na začiatku všetky políčka označené číslom 1,
9. Hru začína hráč 1 (po stlačení sa jeho políčko zmení na číslo 0, po stlačení hráča 2 sa políčko zmení na číslo 2),
10. Po kliknutí na prvé políčko, musí podpísať daný krok v metamask,
11. Následne pokračuje oponent, ktorý si vyberie políčko a musí taktiež podpísať daný krok,
12. Takto sa to opakuje krok 9 a 10 až kým sa hra neskončí, nie je jasný výherca,
13. V prípade, že váš oponent podpísal výherný krok, tak ste prehrali. Vašou úlohou ako prehrávajúceho je uznať porážku kliknutím tlačidla Send signature of board, kde podpíšete výherný stav oponenta, ktorý si bude môcť potom prebrať výhru,
14. V prípade ak ste vyhrali, čakáte na podpis uznanej porážky oponenta a následne stlačíte tlačidlo Claim being a winner,
15. Koniec hry.

Kapitola 6

Testovanie

6.1 Časové testovanie bežnej hry

Boli vykonané aj ručné časové testy, kde som sa snažil nájsť používateľov, ktorí nikdy nepoužívali web3 a získať tak reálne výsledky od používateľov bez skúseností. Časovo ukončiť hru pod 5 minút sa mi nepodarilo ani s jedným hráčom, často bol problém, že zavreli podpisovacie okno. Inak hry prebiehali plynulo. Prvý test som robil sám so sebou a podarilo sa mi to aj s podpismi, aj vyplatením dosiahnuť za približne 240 sekúnd. Dáta som čerpal z verejných adries kontraktov jednotlivých hier.



Obr. 6.1: Testovanie

6.2 Hardhat testovanie

Testovanie smart kontraktu bolo vykonané pomocou Hardhat nástroja, ktorý ponúka rozsiahle informácie a možnosti testov. Vykonali sa 4 základné testy. Prvý test obsahoval test základných metód kontraktu ako volanie konštruktora, kde je nutné vložiť vklad a následne je dôležité overiť, či je možné sa do hry aj pripojiť, čo sa overilo zavolaním a vyhodnotením funkcie join. V druhom a treťom teste sa simuloval obdobný scenár s tým, že došlo až k ukončeniu hry pomocou zavolania verify funkcie, ktorá vyhodnocuje, či sa hra ukončila, kto vyhral a či ju obaja podpísali. Ak je všetko v poriadku tak sa očakáva vyplatenie výhry výhercovi a zmena premennej winner. A ako posledný test funkcionality som urobil hru, kde jeden z

hráčov zavolá timeout na protihráča. Potom testovacia funkcia počká 60 sekúnd. V časti testovania som nastavil tento timeout na 1 minútu z pôvodných 2, a po uplynutí času sa pokúsi nárokovať si výhru. V prípade ak sa všetko splnilo, timeout bol naplnený alebo prekročený, výherca neurčený, tak sa volajúci deklaruje ako výherca a vyplatí sa mu stavená čiastka.

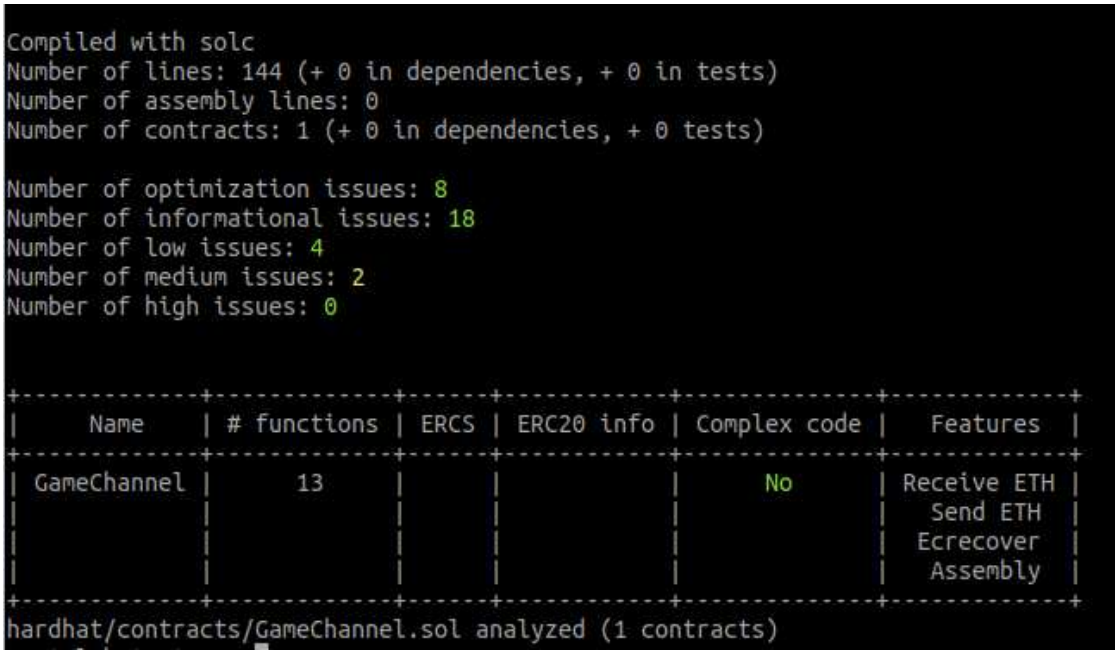
Z gas reportu na obrázku č. 6.2. môžeme vidieť, že najdrahší úkon je samotné vytvorenie a prvotný zápis do blockchainu, ktorý stojí až cez 1 milión gas jednotiek, pričom najlacnejšia funkcia začína na 40 tisíc jednotkách gasu a dokáže sa vyšplhať až na 100 tisíc jednotiek gasu v prípade volania timeoutu, kde sa kontrolujú herné dosky a podpisy.

```
λ npx hardhat test
Compiled 1 Solidity file successfully

  GameChannel
    √ Deployed contract with 0.02 ETH value. User 2 joined successfully. (1439902 gas)
    √ Simulated game where player 1 should be a winner. (1449412 gas)
    √ Simulated game where player 2 should be a winner. (1474985 gas)
        Waiting 60 seconds...
    √ Simulation of timeout situation. (1561279 gas)

·-------------------------------------|---------------------------|-------------|-----------------------------·
|       Solc version: 0.8.4           ·  Optimizer enabled: true  ·  Runs: 200  ·  Block limit: 6718946 gas  |
·-------------------------------------|---------------------------|-------------|-----------------------------·
|  Methods                                                                                                    |
·-------------·-----------------------|-------------|-------------|-------------|--------------|-------------·
|  Contract   ·  Method               ·  Min        ·  Max        ·  Avg        ·  # calls     ·  eur (avg)  |
·-------------·-----------------------|-------------|-------------|-------------|--------------|-------------·
|  GameChannel  ·  claimTimeout       ·          -  ·          -  ·      52080  ·           1  ·          -  |
·-------------·-----------------------|-------------|-------------|-------------|--------------|-------------·
|  GameChannel  ·  join               ·          -  ·          -  ·      43724  ·           5  ·          -  |
·-------------·-----------------------|-------------|-------------|-------------|--------------|-------------·
|  GameChannel  ·  timeoutChallenge   ·          -  ·          -  ·     101995  ·           2  ·          -  |
·-------------·-----------------------|-------------|-------------|-------------|--------------|-------------·
|  GameChannel  ·  verify             ·      67781  ·      69297  ·      68539  ·           4  ·          -  |
·-------------·-----------------------|-------------|-------------|-------------|--------------|-------------·
|  Deployments                        ·                                         ·  % of limit  ·             |
·-------------------------------------|-------------|-------------|-------------|--------------|-------------·
|  GameChannel                        ·          -  ·          -  ·    1294183  ·      19.3 %  ·          -  |
·-------------------------------------|-------------|-------------|-------------|--------------|-------------·

  4 passing (1m)
```

Obr. 6.2: Hardhat výsledky testov.

6.3 Security testovanie

Vrámci security testovania sa vykonali podrobné testy smart kontraktu. Na obrázku č. 6.3. môžeme vidieť detailné informácie a prehľad funkcionality našeho kontraktu. Vidíme, že kontrakt neobsahuje žiadne závažné problémy.



Obr. 6.3: Test 1.

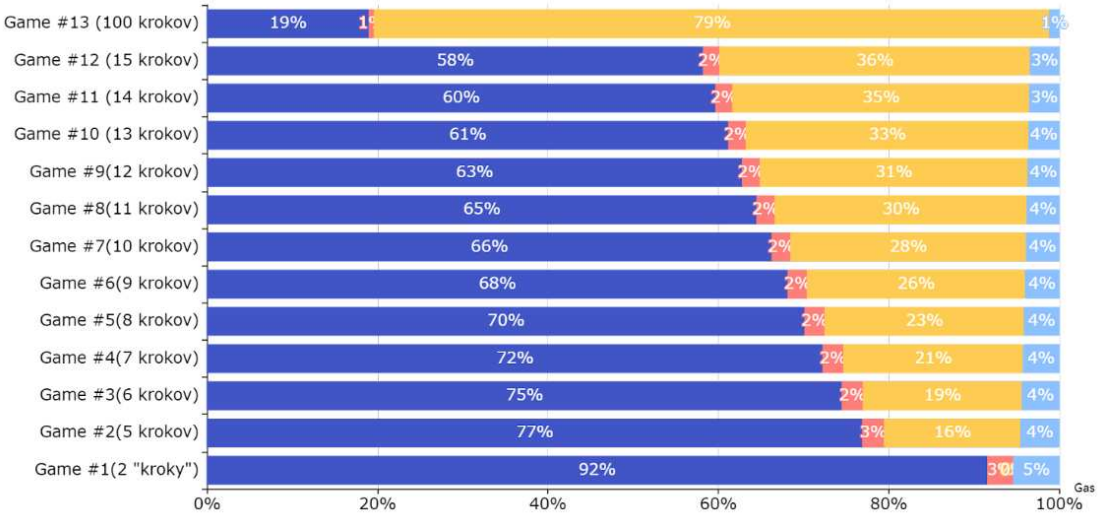
Na obrázku č. 6.4. môžeme vidieť detailne popísané problémy, ktoré našlo v teste z obrázka č. 6.3. . Ide o nie vážne problémy, nedodržaná konvencia názvov alebo podobný názov premenných. Najvážnejší problém má s posielaním ethereum mincí na adresy cez nízkoúrovňové volanie a pristupovaním k času bloku. Avšak tieto problémy sa nedajú odstrániť, pretože pre správne fungovanie hry, timeouty potrebujeme. Na security testovania sa použil framework Slither.

Obr. 6.4: Test Security

6.4 Výsledky testov

Na základe získaných výsledkov vieme predpokladať spotrebu gas jednotiek a nasledne vidíme výsledky úspešnosti našej práce. Aktuálna implementácia obsahuje štýl hry, kde je potrebný určitý počet krokov k tomu, aby hra došla k finálnemu stavu. Pri testovaní išlo o hru piškvôrky na ploche 3x3, kde je minimálny počet krokov potrebných k výhre na úrovni piatich krokov ak berieme do úvahy spotrebu gasu oboch hráčov dokopy. Môžeme odhadovať cenu zápisu jedného kroku na približne 50 tisíc jednotiek gasu. V našom prípade sa jedná v ideálnom scenári iba o 2 potrebné kroky, v prípade hráča A robíte deploy a ak vyhráte tak druhý krok je vzatie výhry pomocou verify funkcie, v tomto prípade by vás hra stála približne 1 milión a 300 tisíc jednotiek gasu, čo je v hodnote približne 26 krokov. V prípade hráča B dochádza k značnému ušetreniu, pretože na pridanie sa do hry je potrebných iba 43 tisíc jednotiek gasu a na ukončenie približne 70 tisíc

jednotiek gasu. Pre ilustráciu, môžeme vidieť, že v prípade ak by sme vykonávali už 100 krokov, gas za samotné kroky by tvoril až 80 percent. V prípade Game 1, kde predpokladáme našu hru je najväčší poplatok vyznačený tmavomodrou farbou, čo reprezentuje deploy kontraktu a samotné kroky hry žltou netvoria žiadnu časť, pretože sa vykonávajú off-chain. Červenou a bledomodrou som vyznačil cenu volania join funkcie a verify funkcie, ktorá ukončuje hru. Tu môžeme vidieť jasný dôkaz a význam herného kanálu ako dokáže ušetriť poplatky.



Obr. 6.5: Testovanie

Kapitola 7

Zhrnutie a záver

V rámci mojej práci som sa v prvej až tretej kapitole venoval prevažne nutným teoretickým základom k implementácii a návrhu riešenia. Podrobne som opísal problematiku Ethereum blockchainu, jeho konsenzu, a taktiež som sa venoval problematike hier vrámci blockchainových sietí. V rámci existujúcich riešení sme našli viacero odlišných riešení, ktoré nám odhalilli aj problémy ale preniesli nespočet odpovedí nespočet otázok. Ku koncu analýzy, sme urobili porovnanie jednotlivých platforiem, pričom sme vyzdvihli jednotlivé kľúčové body ako to, či využívajú off chain technológie alebo nie. Ďalej sme si definovali základné funkcionálne a nefunkcionálne požiadavky, ktoré bude náš obsahovať.

Kapitola 4 sa venuje už konkrétnemu návrhu, kde sa jasne popísali jednotlivé štruktúry a potrebné objekty vrámci projektu. Návrh koštry kontraktu, návrh spájania hráčov, riešenie tých najdôležitejších problémov. Jasný diagram sekvencií, kde vidno ako bude hra prebiehať a prípadne aj ako sa budú riešiť problémy vrámci hry.

Kapitola 5 sa venuje implementácii návrhu, kde sú jasné opísané použité tech-

nológie, odôvodnené ich použitie, sú tam podrobne opísané zmeny voči návrhu, išlo najmä o vylepšenia voči návrhu. Obsahuje podrobný opis celej aplikácie back-endu aj front-endu, obsahuje opis celého smart kontraktu v jazyku solidity, opis transakcií a je opísaná aj ukážková interakcia.

Kapitola 6 sa venuje testovaniu, konkrétne časového testovania samotnej hry s bežnými užívateľmi. V druhej časti sa venuje testovanie pomocou hardhat, ktorá ponúka podrobné výsledky a možnosti testovania ako cena v jednotkách gasu, čas potrebný na kompiláciu a iné.

Projekt je výsledok bakalárskej práce, pričom bol dodržiavaný harmonogram práce a aj jej pravidlá, ktoré boli stanovené. Výsledok jasne ukazuje, že je možné nielen implementovať hru na blockchainovej technológii ale, že je možné držať stavy mimo blockchain a zvýšiť tak možnosti škálovateľnosti. Riešenie splnilo očakávania, avšak vrámci budúceho vývoja je možné rozšíriť o viaceré hry, či urobiť porovnania na side chaine ako je napríklad polygon sieť.

Zoznam použitej literatúry

[1] M. Yano, C. Dai, K. Masuda a Y. Kishimoto, ed., *Blockchain and Crypto Currency: Building a High Quality Marketplace for Crypto Data*, en, ed. Economics, Law, and Institutions in Asia Pacific. Singapore: Springer Singapore, 2020, ISBN: 9789811533754 9789811533761. DOI: 10/hbpd. (cit. 31. 12. 2021).

[2] D. G. Wood, "ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER," en, s. 34, 2014. url: https://gavwood.com/paper.pdf.

[3] V. Buterin et al., "Ethereum white paper," *GitHub repository*, roč. 1, s. 22–23, 2013.

[4] "How does Ethereum work, anyway? – Preethi Kasireddy – Medium," en, s. 40, 2017, https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369.

[5] S. Tikhomirov, "Ethereum: State of Knowledge and Research Perspectives," en, in *Foundations and Practice of Security*, A. Imine, J. M. Fernandez, J.-Y. Marion, L. Logrippo a J. Garcia-Alfaro, ed., zv. 10723, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2018, s. 206–221, ISBN: 978-3-319-75649-3 978-3-319-75650-9. DOI: 10.1007/978-3-319-75650-9_14.

[6] P Sajana, M Sindhu a M Sethumadhavan, "On blockchain applications: hyperledger fabric and ethereum," *International Journal of Pure and Applied Mathematics*, roč. 118, č. 18, s. 2965–2970, 2018.

[7] S. Bragagnolo, H. Rocha, M. Denker a S. Ducasse, "Ethereum query language," en, in *Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*, Gothenburg Sweden: ACM, máj 2018, s. 1–8, ISBN: 978-1-4503-5726-5. DOI: 10.1145/3194113.3194114.

[8] Y. Hirai, "Defining the Ethereum Virtual Machine for Interactive Theorem Provers," en, in *Financial Cryptography and Data Security*, M. Brenner, K. Rohloff, J. Bonneau, A. Miller, P. Y. Ryan, V. Teague, A. Bracciali, M. Sala, F. Pintore a M. Jakobsson, ed., zv. 10323, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2017, s. 520–535, ISBN: 978-3-319-70277-3 978-3-319-70278-0. DOI: 10/gf8xr8.

[9] J. Toroman, "Application of Ethereum Smart Contracts in Purpose of Generating New Cryptocurrencies," *Global Journal of Computer Science and Technology*, url: globaljournals.org/GJCST_Volume18/4-Application-of-Ethereum-Smart.pdf.

[10] S. Bragagnolo, H. Rocha, M. Denker a S. Ducasse, "SmartInspect: solidity smart contract inspector," in *2018 International workshop on blockchain oriented software engineering (IWBOSE)*, IEEE, 2018, s. 9–18. DOI: 10.1109/IWBOSE.2018.8327566.

[11] R. Modi, *Solidity Programming Essentials: A beginner's guide to build smart contracts for Ethereum and blockchain*. Packt Publishing Ltd, 2018, ISBN: 978-1788831383.

[12] C. Signer, "Gas Cost Analysis for Ethereum Smart Contracts," en, 39 p. 2018, Artwork Size: 39 p. Medium: application/pdf Publisher: ETH Zurich. DOI: 10.3929/ETHZ-B-000312914.

[13] F. Cassez, J. Fuller a A. Asgaonkar, "Formal Verification of the Ethereum 2.0 Beacon Chain," en, *arXiv:2110.12909 [cs]*, okt. 2021, arXiv: 2110.12909.

[14] J. Eberhardt a S. Tai, "ZoKrates - Scalable Privacy-Preserving Off-Chain Computations," en, in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Halifax, NS, Canada: IEEE, júl 2018, s. 1084–1091, ISBN: 978-1-5386-7975-3. DOI: 10/hbjh.

[15] J. Stark, "Making Sense of Ethereum's Layer 2 Scaling Solutions: State Channels, Plasma, and Truebit," en, *2018-07-23]. url= https://medium.com/l4-media/making-senseof-ethereums-layer-2-scalingsolutions-state-channels-plasmaand-truebit-22cb40dcc2f4, year=2018*, s. 17,

[16] G. C. e. al, *How to defi:: Beginner*. 2021, ISBN: 979-8640579109.

[17] M. di Angelo a G. Salzer, "Tokens, Types, and Standards: Identification and Utilization in Ethereum," en, in *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, Oxford, United Kingdom: IEEE, aug. 2020, s. 1–10, ISBN: 978-1-72816-978-1. DOI: 10/hbjm. url: https://ieeexplore.ieee.org/document/9126009/ (cit. 28. 12. 2021).

[18] K. Iyer a C. Dannen, *Building Games with Ethereum Smart Contracts*, en. Berkeley, CA: Apress, 2018, ISBN: 978-1-4842-3491-4 978-1-4842-3492-1. DOI: 10.1007/978-1-4842-3492-1.

[19] Q. Wang, R. Li, Q. Wang a S. Chen, "Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges," en, *arXiv:2105.07447 [cs]*, okt. 2021, arXiv: 2105.07447.