**Slovenská technická univerzita v Bratislave**

**Fakulta informatiky a informačných technológií**

**Ilkovičova 2, 842 16 Bratislava 4**

Predmet

**– Digitálne meny a Blockchain –**

**- Dokumentácia -**

**Smart kontraktový systém pre námorne bitky**

Ak. Rok : 2021/2022, letný semester

**Cvičiaci:**

**Ing. Viktor Valaštín**

**Študent:**

Martin Rudolf 97029

**Obsah:**

[1 Implementačné prostredie 2](#_Toc100692216)

[2 Doimplementované časti kódu 2](#_Toc100692217)

[a. contract\_starter.sol 2](#_Toc100692218)

[i. store\_bid(): 2](#_Toc100692219)

[ii. clear\_state() 2](#_Toc100692220)

[iii. store\_board\_commitment() 3](#_Toc100692221)

[iv. check\_one\_ship() 3](#_Toc100692222)

[v. claim\_win() 3](#_Toc100692223)

[vi. forfeit() 3](#_Toc100692224)

[vii. accuse\_cheatin() 4](#_Toc100692225)

[viii. claim\_opponent\_left() 4](#_Toc100692226)

[ix. handle\_timeout() 4](#_Toc100692227)

[x. claim\_timeout\_winnings() 4](#_Toc100692228)

[b. BattleshipPlayer.js 4](#_Toc100692229)

[i. place\_bet() 4](#_Toc100692230)

[ii. Initialize\_board() 5](#_Toc100692231)

[iii. recieve\_response\_to\_guess() 5](#_Toc100692232)

[iv. accuse\_timeout() 5](#_Toc100692233)

[v. handle\_timeout\_accusation() 5](#_Toc100692234)

[vi. claim\_timout\_winnings() 5](#_Toc100692235)

[vii. accuse\_cheating() 5](#_Toc100692236)

[viii. claim\_win() 5](#_Toc100692237)

[ix. forfeit\_game() 6](#_Toc100692238)

[3 Testovanie 6](#_Toc100692239)

[4 Test coverage 7](#_Toc100692240)

[5 Security analýza 8](#_Toc100692241)

[6 Otázky 9](#_Toc100692242)

# Implementačné prostredie

Ako implementačné prostredie som si zvolil Virtual Studio Code, pomocou ktorého som doimplementoval požadované funkcie v zdrojovom súbore contract\_starter.sol v programovacom jazyku Solidity a v súbore BattleshipPlayer.js v jazyku JavaScript. Pre kompiláciu, deploy a testovanie som použil nástroj Truffle teda som si pre dané zadanie vytvoril truffle projekt ktorý bol pripojený na Ganache. Pre jednoduchšie overovanie funkcií smart kontraktu som využíval aj Remix IDE.

# Doimplementované časti kódu

## contract\_starter.sol

### store\_bid():

Táto funkcia priradí hodnoty stavovým premenný reprezentujúce adresy hráčov a uloží čiastky ktoré poslali na kontrakt. Ak hodnota stavovej premennej player1je default tak sa jej priradí adresa ktorá komunikuje s touto funkciou a namapuje sa na tu adresu čiastka ktorú daná adresa poslala na kontrakt. Inak sa to iste vykoná pre stavovú premennú Player2.

Text

Description automatically generated

### clear\_state()

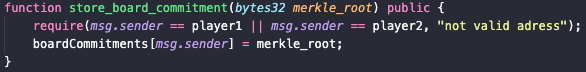
Pomocou tejto funkcie inicializujeme stavové premenné na default hodnoty a tak pripravíme kontrakt na novú hru.

Text

Description automatically generated

### store\_board\_commitment()

Funkcia najprv overí či adresa ktorá volá funkciu patrí niektoremu z hráčóv ak ano tak sa na ňu namapuje koreň mekrleho stromu, čiže počiatočný stav hracej dosky.



### check\_one\_ship()

Funkcia pomocou už implementovanej funkcie verify\_opening() overí na základe vstupných argumentov (opening\_nonce, proof teda hashe susedov, index daného listu, a počiatočný stav hracej dosky daného hráča). Ak je adresa ktorá interaguje s kontraktom rovná owner, vstupnému argumentu funkcie, a zároveň pomocná funkcia vrátila true, tak sa zvýši counter namapovaný na adresu volajúceho o 1. Teda overím pozíciu mojej lode. To isté sa udeje keď je owner protivník ale zvýši sa counter počitajúci zásahy.

Text

Description automatically generated

### claim\_win()

Pri volaní tejto funkcií sa najprv overí, či hráč ktorý učinil volanie má počítadlá, ktoré nesú informáciu o počte trafených a počte položených lodí sú rovné 10. Ak áno tak vyplatí sa mu výhra, nastaví sa stavová premenna gameOver na true a zavolá sa funkcia clear\_state().

Text

Description automatically generated

### forfeit()

Overí či adresa ktorá volá danú funkciu a adresa zo vstupného argumentu funkcie patria hráčom. Ak áno tak sa výhra odošle oponentovi a nastaví sa kontrakt pre novú hru.

Text

Description automatically generated

### accuse\_cheatin()

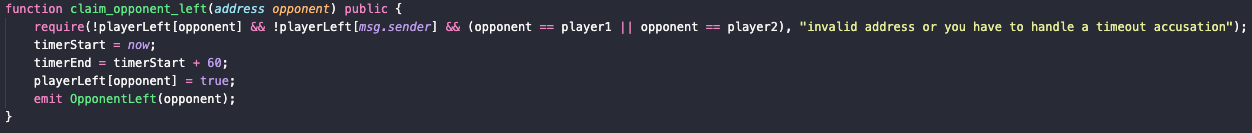
Rovnako ako v predchádzajúcich funkciách overím či interagujúca adresa patrí jednému z hráčov a skontrolujem daný ťah či bol podvod pomocou už spomenutej funkcii verify\_opening(). Ak pomocná funkcia vráti false tak sa výhra pripíše na adresu ktorá zavolala túto funkciu.

Text, application

Description automatically generated

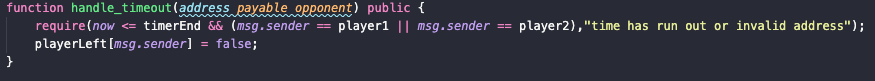
### claim\_opponent\_left()

Skontroluje či obviňovaný hráč už nie je náhodou obvinený a zároveň či obviňovateľ nie je rovnako obvinený. Spustí sa časovač na jednu minútu a emitne sa event hovoriaci že hráč je neprítomný alebo zdržuje.



### handle\_timeout()

Skontroluje či čas už nevypršal, ak nie tak nastaví hodnotu namapovanú na obvineného adresu na false čo zastaví časovač.



### claim\_timeout\_winnings()

Ak je oponent stále obvinený a časovač ubehol tak sa hráčovi vyplatí výhra a nastaví sa kontrakt na novú hru.

Text

Description automatically generated

## BattleshipPlayer.js

### place\_bet()

Interaguje s kontraktom, konkrétne s funkciou store\_bid() na ktorú pošle hodnotu stávky vo wei a adresu hráča.

### Initialize\_board()

Doimplementovaná časť rovnako len interaguje so smart kontrakt funkciou store\_board\_commit.

### recieve\_response\_to\_guess()

Do tejto funkcie bola doimplementovana časť kódu, ktorý vytvorí objekt s hodnotami proof, index, a opening\_nonce ktoré potrebujeme na overenie pravosti lode. Uložím si tento objekt ako last\_guess a ešte skontrolujem príznak opening, ak je true tak si vytvorený objekt uložím do poľa zničených lodí.

Text

Description automatically generated

### accuse\_timeout()

Funkcia len jednoducho volá funkciu smart kontraktu claim\_opponent\_left().

### handle\_timeout\_accusation()

Zavolá metódu smart kontraktu handle\_timeout() a vráti true ak je koniec hry.

### claim\_timout\_winnings()

Interaguje s funkciou claim\_timeout\_winnings smart kontraktu

### accuse\_cheating()

Funkcia využíva vyšie spomínaný objekt last\_guess, ktorý ak je definovaný tak sa odošlu jeho atribúty do funkcie smart kontraktu accuse\_cheating().

### claim\_win()

Ak sa počet hráčových lodí a počet zostrelených lodí rovná 10, tak sa pravosť každej z nich overí na smart kontrakte pomocou funkcie check\_one\_ship(). Následne sa zavolá funkcia claim\_win zo smart kontraktu.

Text

Description automatically generated

### forfeit\_game()

Interakcia s funkciou smartkontraktu forfeit()

# Testovanie

Prostredie na testovanie som si zvolil truffle, kde som si v adresary testy vytvoril súbor testBattleship.test.js v ktorom som implementoval testy.

Prvý test kontroluje hodnotu stavovej premennej gameOver. Ďalší kontroluje či su inicializované správne default adresy hráčov. V treťom sa kontroluje či funkcia check\_one\_ship vráti správnu hodnotu pre nevalidnú loď. Nasledujúci kontroluje, či sa zmení counter, po skontrolovaní nevalidnej lode. Posledný test kontroluje správnosť hodnoty stávky poslanej na smart kontrakt.

Text

Description automatically generated

Text

Description automatically generated

# Test coverage

Na test coverage som použil nástroj dostupný [tu](https://github.com/sc-forks/solidity-coverage). Priemerny test coverage je 25,9.

A picture containing text, scoreboard, clapperboard

Description automatically generated

# Security analýza

Bezpečnostnú analýzu prebehla v remixe s pluginom SOLIDITY STATIC ANALYSIS s takýmito výsledkami:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

# Otázky

**Predpokladajme, že hráč 1 umiestni na hraciu plochu menej ako 10 lodí, ale nikdy neklame o zásahoch alebo minutiach. Môže hráč 2 dostať svoje peniaze späť? Prečo áno, prečo nie?**

Nemôže lebo náš systém nepočíta s takouto situáciou. Jediná možnosť ako ich môže dostať spať je, že sa súper vzdá alebo nestihne zareagovať na timeout.

**Prečo neobmedzujeme hráčov v umiestňovaní viac ako 10 lodí na ich dosku?**

Lebo keď je pravidlo, že si ma hráč umiestniť práve 10 lodí a keďže hrá o peniaze, nebude riskovať, nemalo by to pre neho žiaden zmysel. Aj keby ich mal viac ako 10 a súper by vystrieľal 10 tak súper si môže nárokovať na výhru ak ma položených svojich 10 lodí.

**Nemáme mechanizmus, ktorý by hráča obviňoval z umiestnenia menej ako 10 lodí na hraciu plochu. Ako by ste ho vedeli implementovať?**

Hráč ktorý ma menej ako 10 lodí nikdy nevyhrá a tak jeho súper sa môže dostať do stavu keď ostanú len tie polička prázdne kde musia byť lode (keď uvažujem že nepodvádzal). Ak vystrelím a odpoveď je že som netrafil tak ho môžem obviniť z toho že položil menej lodí ako 10. V jedenej premennej si počítam výstrely, ak 30 výstrelov a z toho som 4 trafil, tak ostatné 6 políčka musia byť lode. Ak po nasledujúcom výstrele je odpoveď že som netrafil obviním protivníka.

**Napadajú vám scenáre útoku alebo konkrétne zraniteľné miesta v niektorom z uvedených kódov, proti ktorým by ste sa nedokázali ubrániť?**

Nevidím potencionálnu možnosť útoku v žiadnom z uvedených kódov.

1. Záver

V rámci tohto zadania som si osvojil praktické zručnosti s nástrojmi na vývoj decentralizovaných aplikácií (DAPP). Naučil som sa najmä aká dôležitá je bezpečnosť, a efektívnosť takto vyvíjanej aplikácie. Nikto predsa nebude používať aplikáciu v ktorá spotrebuje veľké množstvo gassu a tak produkuje veľké fíčka.

Oboznámil som sa taktiež s nástrojmi pre vývoj smart kontraktov, akými sú programovací jazyk Solidity, Remix IDE na jednoduché testovanie a overovanie funkčnosti smart kontrakt funkcií. Nástroj Truffle mi poslúžil na manažovanie, testovanie a deployovanie aplikácie na lokálnu eth sieť ktorú spravoval nástroj Ganache.

Pri tomto zadaní som sa veľa naučil a hodnotím ho kladne. Aj keď s JavaScriptom som nebol veľký kamarát, no musím sa priznať, po tej troške čo som s ním pracoval pri tomto zadaní mu prichádzam na chuť. Avšak ako veľké plus vnímam to, že som porozumel ako fungujú DAPP a ako prebieha komunikácia medzi frontendom a kontraktom deploynutým na blockchaine.