

---

**UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA  
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,  
TÎRGU-MUREŞ,  
SPECIALIZAREA CALCULATOARE**

**Bursă descentralizată  
PROIECT DE DIPLOMĂ**

**Coordonator științific:**

**Dr. Kátai Zoltán,  
conferențiar universitar**

**Absolvent:**

**Körmöci Csaba**

**2023**

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș  
Specializarea: **Calculatoare**

Viza facultății:

## LUCRARE DE DIPLOMĂ

Coordonator științific:  
**ș.l. dr. ing. Káta Zoltán**

Candidat: **Körmöci Csaba**  
Anul absolvirii: **2023**

a) Tema lucrării de licență:  
Bursă descentralizată

b) Problemele principale tratate:

- Studierea lumii tehnologiilor cripto și blockchain
- Explorarea aplicațiilor similare (Uniswap)
- Implementarea unei aplicații mobile care să permită tranzacționarea între criptomonede.

c) Desene obligatorii:

- Schema bloc al aplicației
- Diagrame UML privind software-ul realizat.

d) Softuri obligatorii:

- Aplicație mobilă care vă permite să tranzacționați criptomonede

e) Bibliografia recomandată:

- Hayden Adams, N. Z. (2020. Március). *Uniswap V2 Core*. Forrás: Whitepaper: <https://uniswap.org/whitepaper.pdf>
- Ethereum. (Június. 16 2023). Ethereum Whitepaper. Forrás: <https://ethereum.org/en/whitepaper/>

f) Termene obligatorii de consultații: săptămânal

g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca,  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș

Primit tema la data de: 31.03.2020

Termen de predare: 27.06.2021

Semnătura Director Departament

Semnătura responsabilului  
programului de studiu

Semnătura coordonatorului

Semnătura candidatului

---

### **Declarație**

Subsemnata/ul KÖRMÖCI CSABA absolvent(ă) al/a specializării Calculatoare, promoția 2023 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea,

Data:

Absolvent

Semnătura.....

---

# Bursă descentralizată

## Extras

Pe măsură ce criptomonedele au devenit din ce în ce mai răspândite în zilele noastre, site-urile și aplicațiile de criptomonde au devenit foarte populare. Este vital să avem instrumente și platforme care să permită tranzacționarea criptomonedelor. În lumea cripto, acestea sunt bursele descentralizate. Acestea sunt platforme sau aplicații care permit schimbul de criptomonde direct între ele, fără a fi nevoie de un intermediar. În plus, bursele descentralizate oferă o modalitate de a furniza lichiditate sistemului. În același timp, ele permit și retragerea de lichidități.

Pe baza acestor informații, scopul tezei mele este, de a dezvolta o aplicație mobilă care să permită utilizatorilor să facă schimb de criptomonde atât pe Android, cât și pe iOS. Rețineți că nu le permitem utilizatorilor să cumpere criptomonde, ci doar să le schimbe.

În crearea tezei mele, a fost nevoie să învăț despre tehnologiile, diferitele principii și soluții care fac posibil un schimb descentralizat. Acestea au inclus diferite tehnologii blockchain, contracte inteligente și principalele funcționalități ale schimburilor descentralizate.

**Cuvinte cheie:** bursă descentralizată, contract intelligent, cripto, blockchain

---

**SAPIENTIA ERDÉLYI MAGYAR  
TUDOMÁNYEGYETEM**

**MAROSVÁSÁRHELYI KAR  
SZÁMÍTÁSTECHNIKA SZAK**

**Decentralizált tőzsde  
DIPLOMADOLGOZAT**

**Témavezető:**

**Dr. Kátai Zoltán,  
egyetemi docens**

**Végzős hallgató:**

**Körmöci Csaba**

**2023**

---

# Kivonat

Mivel napjainkban egyre elterjedtebbé váltak a kriptovaluták, ezért az ezzel foglalkozó oldalak és alkalmazások rendkívül felkapottak lettek. Létfontosságú, hogy létezzenek olyan eszközök, platformok, amelyek lehetővé teszik a kriptovaluták kereskedését. A kripto világban ezek alkotják a decentralizált tőzsdéket. Ezek olyan platformok vagy alkalmazások, amelyek lehetővé teszik a kriptovaluták közvetlen cseréjét egymás között anélkül, hogy egy közvetítőre lenne szükség. Ezenfelül a decentralizált tőzsdék lehetőséget adnak arra, hogy likviditást szolgáltassunk a rendszerbe. Ugyanakkor lehetővé teszik a likviditás kivételét is.

Ezen információk alapján a dolgozatom célja, hogy egy olyan mobil alkalmazást fejlesszek, amely lehetővé teszi a kriptovaluták cseréjét a felhasználók számára úgy Android-on, mint iOS-en. Figyelembe kell vegyük, hogy a kriptovaluták vásárlását nem tesszük lehetővé a felhasználók számára, csupán azok cseréjét.

A dolgozatom létrehozása során meg kellett ismerjem azokat a technológiákat, különböző elveket és megoldásokat, amelyek lehetővé teszik egy decentralizált tőzsde működését. Ezek között voltak a különböző blokklánc technológiák, okos szerződések, illetve a decentralizált tőzsdék főbb funkcionalitásai.

**Kulcsszavak:** blokklánc, kriptovaluta, decentralizált tőzsde, okos szerződés

---

# Abstract

As cryptocurrencies have become more and more widespread these days, cryptocurrency sites and applications have become more popular. It is vital to have tools and platforms that allow trading cryptocurrencies. In the crypto world, these are the decentralised exchanges. These are platforms or applications that allow cryptocurrencies to be exchanged directly between each other without the need for an intermediary. In addition, decentralised exchanges provide a way to supply liquidity to the system. At the same time, they also allow liquidity to be withdrawn.

Based on this information, the aim of my thesis is to develop a mobile application that allows users to exchange cryptocurrencies on both Android and iOS. Note that we do not allow users to buy cryptocurrencies, only to exchange them.

In creating my thesis, I needed to learn about the technologies, different principles and solutions that make a decentralised exchange possible. These included different blockchain technologies, smart contracts and the main functionalities of decentralised exchanges.

**Keywords:** decentralized exchange, smart contract, crypto, blockchain

## Tartalomjegyzék

1. Bevezető .....	12
2. Célkitűzések .....	13
3. Elméleti megalapozás.....	13
3.1. Elméleti alapok.....	13
3.1.1. Blokklánc .....	13
3.1.2. DEX vs CEX.....	14
3.1.3. Uniswap .....	16
3.2. Ismert hasonló alkalmazások.....	19
3.2.1. Uniswap .....	20
3.2.2. PancakeSwap .....	20
3.2.3. 1inch.....	21
3.2.4. Curve Finance .....	22
3.3. Felhasznált technológiák .....	23
3.3.1. JavaScript.....	23
3.3.2. TypeScript.....	23
3.3.3. Node.js .....	24
3.3.4. React Native.....	24
3.3.5. Expo .....	26
3.3.6. MongoDB .....	26
3.3.7. Solidity .....	27
3.3.8. Etherscan.....	28
3.3.9. Alchemy .....	29
3.3.10. Goerli Testnet.....	30
3.3.11. Metamask.....	30
3.3.12. Tenderly .....	32
3.3.13. WalletConnect.....	32
3.3.14. Hardhat.....	33
4. A rendszer specifikációi .....	33
4.1. Felhasználói követelmények .....	33
4.2. Rendszer követelmények.....	36
4.2.1. Funkcionális követelmények .....	36
4.2.2. Nem-funkcionális követelmények .....	37
5. A rendszer architektúrája és üzembe helyezése .....	39
5.1. Backend .....	40
5.1.1. Adatbázis-kezelő rendszer .....	40
5.1.2. Blokklánc kommunikáció .....	42
5.2. Frontend.....	44
6. Az alkalmazás bemutatása és felhasználása .....	47
6.1. Főoldal.....	47
6.1.1. Kriptovaluták kilistázása.....	47
6.1.2. Kriptovaluták részletes nézete .....	48
6.2. Kereskedési oldal.....	50
6.2.1. Swap.....	50
6.2.2. Add liquidity .....	56
6.2.3. Remove liquidity.....	58
6.3. Beállítások oldal .....	60

7. Következtetések .....	61
8. Összefoglalás.....	62
8.1. Megvalósítások.....	62
8.2. Főbb funkciók és összehasonlítás hasonló alkalmazásokkal.....	62
8.3. Továbbfejlesztési lehetőségek .....	63
9. Bibliográfia.....	64

## Ábrák jegyzéke

ábra 1 - Az első blokk (Genesis Block) [Forrás 1].....	14
ábra 2 – Uniswap működése [Forrás 5].....	16
ábra 3 - Uniswap konstans formula működése [Forrás 5].....	17
ábra 4 - Uniswap ármeghatározás [Forrás 5] .....	18
ábra 5 – TypeScript mintakód (Forrás: [10]) .....	24
ábra 6 – React Native mintakód (Forrás: [11]) .....	25
ábra 7 – Példa egy egyszerű MongoDB token documentum-ról .....	27
ábra 8 – Solidity mintakód .....	28
ábra 9 – Etherscan weboldal.....	29
ábra 10 – Metamask mobilapplikáció .....	31
ábra 11 - Alkalmazás use-case diagramja .....	35
ábra 12 - DEX alkalmazás architektúrája.....	39
ábra 13 - Node.js szerver mappa struktúrája .....	41
ábra 14 - Főbb Uniswap V2 contract-ok [Forrás 12] .....	43
ábra 15 - Swap szekvencia diagram .....	45
ábra 16 - Swap kódrészlet (swapExactTokensForTokens) .....	46
ábra 17 - Kriptovaluták kilistázása.....	48
ábra 18 – Bitcoin kriptovaluta részletes nézete .....	49
ábra 19 - Swap oldal.....	51
ábra 20 - Swap oldal: tokenek kiválasztása.....	52
ábra 21- Swap oldal: értékek beírása és számolása.....	53
ábra 22 - Tranzakció beállítása.....	54
ábra 23 - Wallet kiválasztása.....	54
ábra 24 - Metamask tranzakció elfogadása .....	55
ábra 25 - Transaction hash értéke.....	55
ábra 26 - Add liquidity kapcsoló .....	56
ábra 27- Add liquidity oldal .....	56
ábra 28 - Add liquidity oldal: minimum likviditási érték.....	57
ábra 29 - Remove liquidity gomb.....	58
ábra 30 - Remove liquidity modal.....	59
ábra 31- Remove liquidity token egyenleg számolás.....	60

ábra 32 - Beállítások oldal.....	61
----------------------------------	----

## 1. Bevezető

Napjainkban egyre elterjedtebbé vált olyan technológiák alkalmazása, amelyek azt a célt szolgálják, hogy egyszerűbbé tegyék az emberek életét, így a kriptovilág is egyre felkapottabbá vált ezen a területen. Mivel a minden napokban a fizetőeszközök száma, illetve típusa nagyon megnőtt, ezért olyan megoldások jelennek meg, amelyek különféle módokat biztosítanak a gyors és biztonságos fizetés végrehajtásához. Gondolhatunk akár a telefonos fizetésre, akár az online lebonyolítandó tranzakciókra, amelyet az otthonunkból is elvégezhetünk. Mivel a tranzakciókat már nem csak natív pénznemben bonyolíthatjuk le, mint például a Dollár, Euró, Ron stb., hanem különböző kriptovaluták segítségével, ezért létfontosságú, hogy létezzen egy olyan eszköz, vagy platform, ahol különböző kriptovalutákat tudunk beváltani más kriptovalutákra. Ezeket nevezzük decentralized exchange-nek, röviden DEX-nek.

A DEX-ek olyan eszközök, amelyek megkönnyítik, illetve elősegítik a kriptovalutákkal való kereskedést. A hagyományos valuta váltók esetében szükséges egy harmadik fél (kormányzati intézmény, bank, kereskedési felület), amely kezeli egy személy letétjét vagy pénzét, illetve biztonságos körülmenyeket biztosít két személy vagy eszköz közötti tranzakció lebonyolítására. A DEX-ek esetében nincs szükség egy harmadik fél által nyújtotta szolgáltatásokra, mivel ezek teljes mértékben automatizált rendszerként működnek felhasználva a blokklánc technológia által nyújtotta előnyöket.

Mivel személy szerint mindenkorán is vonzott a kriptovilág ezért szerettem volna egy olyan alkalmazást létrehozni, amely felhasználja a blokklánc technológiákat. Éppen ezért a dolgozatom során egy olyan mobil alkalmazást terveztem, illetve hoztam létre, amellyel képesek vagyunk egy megszokott valuta váltóhoz hasonlóan kriptovalutákkal kereskedni. Az alkalmazást felhasználva, ha rendelkezünk egy virtuális pénztárcával és bizonyos kriptovalutákkal, akkor lehetőségünk van kereskedelmi műveleteket végrehajtani.

## **2. Célkitűzések**

A dolgozat fő célja, hogy megismerkedjek a kripto világával, illetve a blokklánc technológiákkal. Egy másik cél, hogy létrehozzak egy olyan alkalmazást, amely lehetővé teszi a decentralizált tőzsdékhez hasonló kereskedelmi műveleteket. Ezek magában foglalják a következőket:

- Kriptotárca csatlakoztatás/leválasztás
- Kereskedelmi műveletek kriptovalutákkal:
  - o Swap
  - o Add Liquidity
  - o Remove Liquidity
- Egyszerű, intuitív és letisztult felület
- Különböző kriptovaluták megtekintése
- Több platform támogatottsága kihasználva a React Native előnyeit
- Tranzakciók testre szabása
- Gyors és megbízható tranzakciók

## **3. Elméleti megalapozás**

### **3.1. Elméleti alapok**

Annak érdekében, hogy megértsük a decentralizált tőzsdék működését fontos, hogy megismerjük azokat az eszközöket, technológiákat és hasonló alkalmazásokat, amelyek ezekre épülnek.

#### **3.1.1. Blokklánc**

A blokklánc egy osztott és decentralizált adatbázis. Abban tér el egy hagyományos adatbázistól, hogy az adatok és az információk nem egy központi szerveren vannak eltárolva, hanem egy elosztott hálózaton.

A blokklánc technológia története a Bitcoin kriptovaluta megjelenésével kezdődik. 2008-ban egy ismeretlen szerző, aki a Satoshi Nakamoto [1] álnevet használta, közzétett egy fehér könyvet (white paper), amelyben bemutatta a Bitcoin és a blokklánc alapelveit. A Bitcoin a világ első decentralizált digitális valutájává vált, amely blokklánc technológiára épült. A blokklánc

létrehozásának a fő célja az volt, hogy elszakadjanak a hagyományos pénzügyi rendszerektől, ugyanis ezek minden központosított adatbázison alapultak, melyben az adatok kormányzati szervezetek felügyelete alatt álltak. E technológia lehetővé tette, hogy a tranzakciók közvetlen végrehajtódjanak a résztvevők között, megszüntetve a közvetítők szerepét, ezáltal decentralizálta téve az egész rendszert.

A Bitcoin blokklánc mellett számos más hálózatok is vannak, mint a Solana, Ripple, Corda, Polygon stb. Viszont az Ethereum hálózat különösen népszerű és elismert, mivel ezt tekintik a második legnagyobb blokkláncnak a Bitcoin mellett. Az Ethereum hálózatot olyan személyek futtatják, akik rendelkeznek a blokklánc aktuális változatával. Abban tér el az Ethereum hálózat a Bitcoin blokklanctól [2], hogy programozható, így saját alkalmazásokat tudunk létrehozni és feltölteni a hálózatra. Olyannyira megbízható infrastruktúrát alakított ki az Ethereum vállalat, hogy 2015 óta még nem történt leállás a hálózatban a több ezer csomópontnak (node-nak) köszönhetően.

Bitcoin Genesis Block	
Raw Hex Version	
00000000	01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000020	00 00 00 00 3B A3 ED FD 7A 7B 12 B2 7A C7 2C 3E ....;Ííý{.^zç,>
00000030	67 76 8F 61 7F C8 1B C3 88 8A 51 32 3A 9F B8 AA gv.a.È.À^ŠQ2:Ý,à
00000040	4B 1B 5E 4A 29 AB 5F 49 FF FF 00 1D 1D AC 2B 7C K.^J)«_Iÿy...¬+
00000050	01 01 00 00 00 01 00 00 00 00 00 00 00 00 00 00 .....
00000060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000070	00 00 00 00 00 00 FF FF FF FF 4D 04 FF FF 00 1D .....ÿÿÿM.ÿy..
00000080	01 04 45 54 68 65 20 54 69 6D 65 73 20 30 33 2F ..EThe Times 03/
00000090	4A 61 6E 2F 32 30 30 39 20 43 68 61 6E 63 65 6C Jan/2009 Chancel
000000A0	6C 6F 72 20 6F 6E 20 62 72 69 6E 6B 20 6F 66 20 lor on brink of
000000B0	73 65 63 6F 6E 64 20 62 61 69 6C 6F 75 74 20 66 second bailout f
000000C0	6F 72 20 62 61 6E 6B 73 FF FF FF FF 01 00 F2 05 or banksÿÿÿ..ò.
000000D0	2A 01 00 00 00 43 41 04 67 8A FD B0 FE 55 48 27 *....CA.gŠýºþUH'
000000E0	19 67 F1 A6 71 30 B7 10 5C D6 A8 28 E0 39 09 A6 .gñ q0..Ó~(à9.
000000F0	79 62 E0 EA 1F 61 DE B6 49 F6 BC 3F 4C EF 38 C4 ybàè.ab*Iók?LiøÄ
00000100	F3 55 04 E5 1E C1 12 DE 5C 38 4D F7 BA 0B 8D 57 6U.À.Á.P\BM+°..W
00000110	8A 4C 70 2B 6B F1 1D 5F AC 00 00 00 00 ŠLp+kñ._¬....

ábra 1 - Az első blokk (Genesis Block) [Forrás 1]

### 3.1.2. DEX vs CEX

Mivel a dolgozatom egy decentralizált tőzsde (DEX) megvalósításáról szól, ezért fontos volt, hogy jobban megértem a hagyományos tőzsdék működését. A hagyományos tőzsde egy központosított és szervezett piac, amelyen általában bárki kereskedhet. Gyakran a kereskedéshez brókerek segítségére van szükség, vagyis olyan köztes személyekre, akik közvetítik a vevő és az

eladó közötti tranzakciókat. A kripto világban ezeket centralizált tőzsdéknek (Centralized Exchange) [3] nevezzük. Ezek olyan kereskedési platform-ok, melyek lehetővé teszik a kriptovaluták vásárlását, eladását és cseréjét. A működésük alapja a megrendelési könyv vagy jegyzőkönyv (order book), amely egy közvetítő szerepét tölti be az eladó és vevő között. Ez az eszköz nyomon követi az összes függőben lévő tranzakciót. Mivel ezek a tőzsdék központosítottak és jogilag megbízhatóak kell legyenek, ezért általában ezek a rendszerek KYC (Know Your Customer) alapúak. Ha valaki kereskedni szeretne egy ilyen platformon akkor szükséges a személyazonossága igazolása, illetve a személy állandó lakcíme.

A decentralizált tőzsdék lehetővé teszik bárki számára a kriptovaluták biztonságos és nyitott kereskedését, közvetítő nélkül. A fontos különbség a centralizált és a decentralizált tőzsdék között, hogy a CEX esetében a tranzakciók egy központi entitás által vannak közvetítve, míg a DEX esetében a tranzakciók a felhasználók között zajlanak. Emellett a DEX esetében a virtuális pénztárcánk (privát és publikus kulcsa) teljes mértékben a mi birtokunkban van, míg a CEX esetében a központosított rendszer felügyelete alatt áll. Míg a CEX-ek zárt rendszert alkotnak, korlátozott fejlesztési lehetőséggel, addig a DEX-ek nyílt forráskóddal rendelkeznek, így teljesen publikusak a közösség számára. A DEX-ek alapját képezik a likviditás szolgáltatók, amelyek nélkül nem leletezhetne egy ilyen rendszer. A likviditás határozza meg, hogy egy kriptovalutához milyen ár-érték arányban tudunk hozzájutni.

Mivel minden tőzsde típus rendelkezik előnyivel és hátránnal [4] ezért fontos, hogy figyelembe vegyük ezeket:

### **CEX előnyei:**

- Könnyed felület, sok funkcionálitás
- Nagyobb pénzforgalom
- Fiat alapú ki- és befizetések

### **CEX hátrányai:**

- Centralizált, egy felhasználónak nincs hatalma a saját pénztárcája fölött
- Jogi csapatok folytonos ellenőrzése alatt áll

### **DEX előnyei:**

- Decentralizált, így nincsenek kormányzati korlátozások

- Nincs köztes közvetítő
- Felhasználói anonimitás

#### **DEX hátrányai:**

- Lassabb, mint a centralizált tőzsde
- Likviditás szükséges

#### **3.1.3. Uniswap**

Mivel az alkalmazásom a Uniswap v2-re épül, ezért fontos volt a részletes megértése. A Uniswap egy decentralizált tőzsde és egyben egy automatizált likviditási protokoll, melyet egy konstans képlet (constant product formula) [5] működtet. Ez a rendszer az Ethereum blokkláncon található meg smart contract-ok formájában. Mindegyik Uniswap smart contract egy likviditási medencét (liquidity pool) kezel, amelyek ERC20<sup>1</sup>-as tokenek-ból tevődnek össze. Bárki likviditásszolgáltatóvá válhat, aki hajlandó egy token párt letétbe helyezni likviditás tokenért cserébe. Ez a token arányos a letétbe helyezett összeggel. Vagyis, ha mi letétbe helyezünk egy token párt, akkor ezzel arányosan kapunk egy harmadik token-t, amelyet bármikor kivehetünk, mely a saját részesedésünket jelképezi.



*ábra 2 – Uniswap működése [Forrás 5]*

Mivel a Uniswap alapja az automatikus likviditás protokoll (Automated Market Maker AMM), ezért olyan smart contract-ot használ, amelyek automatikusan létrehozzák és fenntartják a likviditást a kereskedéshez. A likviditás alapja a likviditási medencék (liquidity

---

<sup>1</sup> ERC20: szabványrendszer, mely az Ethereum hálózat tokeneire érvényes

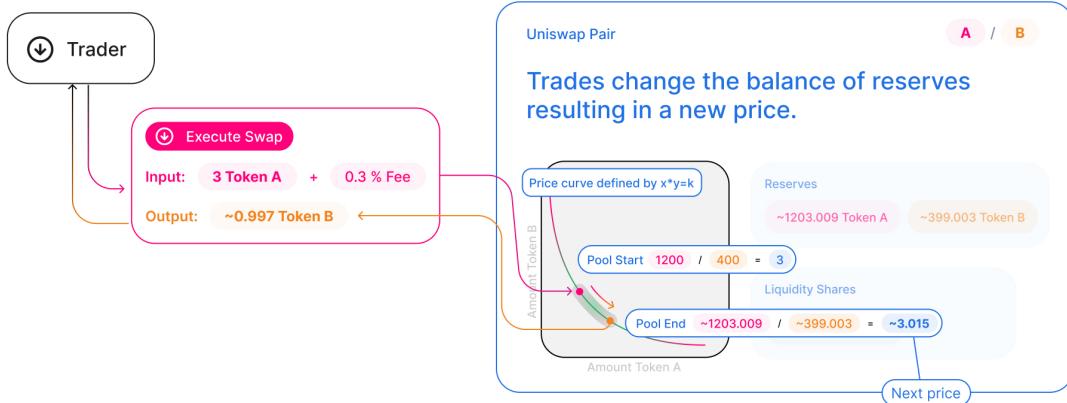
pools), amelyekben a felhasználók letétbe helyezhetik a saját tokeneiket. Ahogy fentebb is említve volt a működésének alapja a **Constant Product Market Maker** modell, amely az  $x * y = k$  képletet használja, amely kimondja, hogy a kereskedés nem változtathatja meg egy pár tartalékegyenlegének ( $x$  és  $y$ ) szorzatát ( $k$ ). Mivel  $k$  változatlan marad, ezért gyakran invariánsnak nevezik [5]. Ez a képlet határozza meg az árfolyamot a két token között, és biztosítja, hogy az árak automatikusan alkalmazkodjanak a kereslet és kínálat változásához. Ennek a képletnek is megvan a sajátos tulajdonsága, hiszen a nagyobb kereskedések exponenciálisan rosszabb arányban teljesítenek, mint a kisebbek.



ábra 3 - Uniswap konstans formula működése [Forrás 5]

A gyakorlatban a Uniswap 0.30%-os kereskedelmi díjat számol fel, melyet a tartalékegyenlegekhez (reserves) adnak hozzá. Ezáltal minden egyes kereskedés növeli a  $k$  invariáns értékét. Ez a likviditásszolgáltatók kifizetéseként működik. Amikor kiveszik a token párokat a medencéből (liquidity pool), akkor elégetik (burn) a medencében található rájuk eső token részesedésüket. Mivel az ár csak kereskedés által változható meg ezért a Uniswap és a külső ár közötti különbséget lehetőséget adnak az arbitrázsra<sup>2</sup>. Ezáltal a mechanizmus által a Uniswap árak mindig megközelítik a piaci egyensúly árát.

<sup>2</sup> arbitrázs: egy olyan művelet, amely során a piaci árkülönbségeket kihasználva próbálunk nyereséget elérni



ábra 4 - Uniswap ármeghatározás [Forrás 5]

A Uniswap rendszere smart contract-okra épül, viszont ezen belül fontos megemlítenünk abból két fő részét a Factory és a Router contract-okat. A Factory contract generikus bytekóddal rendelkezik, mely a token párok létrehozásáért felel [6]. Az ő feladata, hogy egyetlen egy smart contract-ot hozzon létre minden egyedi token párra. A Router contract feladata, hogy rendelkezzen minden olyan funkcionálitással, amely szükséges, hogy egy felhasználó interakcióba léphessen a rendszerrel. Ilyen például a kereskedési és likviditáskezelési funkciók.

A smart contract-okon kívül még fontos beszélnünk a **Swap**-okról és a **Pool**-okról. A Uniswap rendszerén belül egy **swap** [8] (csere) a következőképpen zajlik (ábra 4): a felhasználó kiválaszt egy bemeneti és egy kimeneti token-t. Ezután megad egy bemeneti token mennyiséget és az algoritmus kiszámolja, hogy a kimeneti token-ból mennyit fog kapni. Ezután egy egyszerű kattintással végrehajtja a cserét és a kiszámolt kimeneti token megjelenik a saját kriptotárcájában. De hogyan is történik mindez a háttérben? A Uniswap nem egy hagyományos tőzsdére alapszik, ahol egy megbízási könyvet (order book) alkalmaznak a likviditás kezelésére vagy az árak meghatározására. Ehelyett az Automated Market Maker mechanizmus felel, hogy visszajelzést adjon az árfolyamokról és a csúszásról (slippage)<sup>3</sup>. A Uniswap-on minden egyes pár valójában egy likviditási medencét alkot. Ezek olyan smart contract-ok, amelyek két egyedi token egyenlegét tartják és felelnek a tokenek letétbe helyezésükért és kivételükért. Mindez a konstans formula segítségével történik, mivel amikor valamelyik token-t megvásárolják

<sup>3</sup> slippage: az az összeg, amennyit az ár egy kereskedési párban mozog a tranzakció benyújtása és végrehajtása között.[7]

(withdrawn) a másikból arányos mennyiséget kell letétbe helyezni (deposit) annak érdekében, hogy a konstans az mindig konstans maradjon.

Most beszéljünk a **pool**-okról [9]. Ahogy említve volt minden likviditási medence egy ERC20-as token párt tartalmaz. Amikor egy ilyen medence létrejön akkor minden egyes token párnak az egyenlege nulla. Ahhoz, hogy kereskedni lehessen egy token párral, először szükséges, hogy valaki feltöltsse a medencét egy kezdeti értékkel. Ez a személy határozza meg a token pár kezdetleges árát. Azok a személyek, akik először hoznak létre, azaz töltenek fel egy medencét arra vannak felkérve, hogy minden token párból egyenlő összeget adjanak hozzá. Ez azért fontos mivel, ha az aktuális piaci árfolyamtól eltérő token-eket töltene fel, akkor arbitrázst teremtene, amelyet egy külső fél ki fog használni. Emellett ár instabilitást és likviditáshiányt okozna. Amikor már egy meglévő pool-hoz szeretne egy felhasználó likviditást hozzáadni, akkor az aktuális árfolyammal arányos token párt kell letétbe helyeznie. Amikor egy felhasználó letétbe helyez egy token párt egy pool-ba, akkor egy likviditási token-t kap cserébe. Ezek a tokenek reprezentálják a pool-hoz viszonyított részesedésüket. A pool likviditásához biztosított likviditási arány határozza meg azt, hogy mennyi likviditási token-t kap az illető. Ha a felhasználó egy új pool-t hoz létre, akkor a likviditási tokenek száma megegyezik a  $\sqrt{x * y}$  képlettel, ahol  $x$  és  $y$  jelöli a token pár mennyiségét. minden egyes tranzakció után egy 0.3%-os díj lesz felszámolva, amely arányosan szétosztásra kerül a pool-ban lévő összes likviditásszolgáltató között.

### 3.2. Ismert hasonló alkalmazások

A kriptovaluták piaca és érdekeltségi köre az elmúlt években egyre nagyobb méreteket öltött, így a decentralizált tőzsdék (DEX-ek) is egyre népszerűbbé váltak a felhasználók körében. A számuk olyannyira megnőtt, hogy 2028-ban egyes vélemények szerint a keresletük eléri majd a 1902,5 millió dollárt. Mivel az NFT-k és a digitális valuták egyre népszerűbbek, ezért azok a platformok melyek lehetővé teszik a kereskedésüket, versengenek egymással, hogy minél több felhasználót csábítanak magukhoz. Ezeket különböző új funkciókkal és a befektetés minél egyszerűbbé tételevel próbálják elérni.

### **3.2.1. Uniswap**

Az Uniswap az egyik legismertebb decentralizált tőzsde (DEX), amely az Ethereum blokkláncra épül. Jelenleg ez a DEX rendelkezik a legtöbb kereskedhető token-nel, így lehetővé teszi a kereskedést bármely két Ethereum token között. Jelentős hatást gyakorolt a decentralizált pénzügyek (DeFI) területére, mivel egy olyan új technológiát vezetett be (AMM), amely forradalmian átalakította a tőzsdék eddigi működését.

A Uniswap rendelkezik egy egyszerű és letisztult felhasználói felülettel, amely által a felhasználók könnyedén tudnak tokenekkel kereskedni. A felhasználók úgynevezett cseréket (swap-eket) hajtanak végre, amelyek során az egyik meglévő tokenüket egy másikra cserélik a likviditási medencék által meghatározott árak alapján. Ez a folyamat lehetővé teszi a gyors és hatékony kereskedést az Ethereum alapú tokenek között. Mivel a Uniswap nyílt forráskódval rendelkezik, ezért azt bárki megtekintheti a blokkláncon.

A Uniswap is rendelkezik egy saját kriptovalutával a UNI token-nel, mint minden más DEX. UNI token-t kapnak az a személyek, akik a platform irányításáért felelnek, illetve akik likviditást szolgáltatnak. A UNI token tulajdonosok szavazási jogokat kapnak a platform fejlesztéseivel és döntéseiivel kapcsolatban, így egyfajta kormányzási token-né (governement token) válik.

#### **Előnyei:**

- Hatalmas piaci résszel rendelkezik
- Irányítási jogokat kínál a UNI token által
- Biztonságos körülményeket biztosít a kereskedéshez

#### **Hátrányai:**

- Lassú hálózattal rendelkezik
- Likviditás okozta vesztességek
- Magas tranzakciós költségek

### **3.2.2. PancakeSwap**

A PancakeSwap egy olyan DEX, amely jelenleg két főbb blokkláncon is működik, Binance Chain-en (BSC), illetve az Ethereum hálózaton. Az alacsonyabb tranzakciós díjak és gyorsabb tranzakciók miatt népszerű vált, illetve a szokásos kereskedés mellett, lehetőséget

biztosít a felhasználóknak, hogy részt vegyenek kereskedési versenyeken, ár előrejelzés versenyeken, különböző nyereményjátékokon és sorsolásokon. Hasonlóan a Uniswap-hoz itt is a befektetők, a likviditás szolgáltatók saját token-t kapnak a letétbe helyezett pénzmennyiségről. A PancakeSwap-ot több mint 1,6 millió felhasználó használta az elmúlt 30 napban, körülbelül 24 millió kereskedést hajtott végre, és több mint 1.9 millió dollár lett befektetve.

#### **Előnyei:**

- Gyönyörű interfésszel rendelkezik
- Különböző szolgáltatásokat biztosít
- Alacsony kezelési költségek

#### **Hátrányai:**

- Csak weboldalon működik
- Nem kompatibilis BTC-vel
- Kisebb token kompatibilitás

### **3.2.3. 1inch**

A 1inch különlegessége, hogy automatizált piac aggregátorként és decentralizált tőzsdeként is működik. Piac aggregátor alatt olyan platformokat, vagy eszközöket értünk, amelyek összegyűjtik és megjelenítik különböző tőzsdék árait, illetve ajánlatait. Ehhez hasonlóan a 1inch is összegyűjti az árakat és a likviditást több DEX-ről, és lehetővé teszi a kliensei számára, hogy a lehető legjobb ár érték arányban tudjanak kereskedni. Az 1inch automatikus likviditásprotokollja által a felhasználók képesek könnyedén tranzakciókat kezdeményezni anélkül, hogy közvetlenül a likviditásforrásokkal kellene dolgozniuk. Az 1inch algoritmusai optimalizálják a tranzakciókat, hogy az ügyfelek a lehető legjobb ár érték arányban szerezzenek vagy adjanak el kriptovalutákat, így csökkentve a tranzakciós költségeket és az árkülönbségekből származó veszteségeket.

#### **Előnyei:**

- Jobb árak és likviditás
- Automatizált likviditásprotokoll
- Több blokklánc hálózatot is támogat

- Nincs szolgáltatási díj

**Hátrányai:**

- Bonyolultabb használat
- Nem támogatja a FIAT valuták használatát

### 3.2.4. Curve Finance

A Curve Finance egy olyan DEX, amely a likviditási pool-okat optimalizálja a stablecoin<sup>4</sup> (például a DAI, USDC, USDT stb.) kereskedelméhez. Célja a stablecoin-ok alacsonyabb árfolyam-ingadozásainak minimalizálása. A stabilitás érdekében a Curve több stablecoin-t likviditását gyűjtő össze egyetlen helyen. Ez nagyban megkönnyíti a felhasználók dolgát, hogy könnyen és hatékonyan cseréljenek stablecoin-okat anélkül, hogy jelentős árkülönbséggel vagy piaci eltérésekkel kellene szembenézniük. A Curve olyan eszközöket használ, amelyek optimalizálják a likviditási swap-okat, ezáltal csökkenti az árkülönbségek okozta veszteségeket. Emellett lehetőségünk van kereszt-swapokat végrehajtani, amelyek közvetlen cserét biztosítanak a stablecoin-ok között, így ez nem egy központi liquidity pool-ban történik meg.

A Curve Finance rendelkezik egy saját token-nel a CRV, amelyet működteti autonóm módon az egész rendszert. Ezt a token-t folyamatosan szétosztják a protokoll likviditásszolgáltatói között, viszont ez az arány évente csökken. A platform kereskedési díja 0.04%-os, amelyet minden egyes kereskedésért felszámolnak. Ez az Ethereum hálózatra érvényes, ugyanis a Polygon hálózat esetében már 0.10%-os díjról beszélhetünk. Amikor egy személy letérbe helyezi a kriptovalutáit, akkor az az összeg egy smart contract segítségével lesz eltárolva, illetve nyomon tartva.

**Előnyei:**

- Alacsony kezelési díjak
- Sokoldalú cserék
- Yield farming

**Hátrányai:**

---

<sup>4</sup> stablecoin: egy olyan kriptovaluta, mely egy stabil árfolyamhoz van rögzítve így csökkenti a volatilitást

- Nem kezdőknek való
- Komplex

### 3.3. Felhasznált technológiák

#### 3.3.1. JavaScript

A JavaScript egy objektumorientált programozási nyelv, mely által weboldalakat és webalkalmazásokat tudunk készíteni. Mivel jelenleg az egyik leggyakrabban használt programozási nyelv, ezért olyan szinten kinötte magát, hogy olyan területeken is használják, mint például: mobil alkalmazások fejlesztése, virtuális valóság, számítógépes játékok, neurális hálózatok.

A JavaScript egy interpretált nyelv, ezért nem tartozik a kompilált nyelvek közé. Ameddig egy Java vagy C++ programot le kell kompilálnunk mielőtt futtatnánk, addig a JavaScript esetében a böngészőben lévő fordítóprogram értelmezi a kódunkat sorról sorra, majd lefuttatja a JavaScript motor segítségével. Ezért rendelkeznek a böngészők egy dedikált JavaScript motorral (JavaScript engine), amely dinamikusan fordítja gépi kóddá a natív kódunkat, amely azonnal végrehajtódik.

#### 3.3.2. TypeScript

A TypeScript a Microsoft által fejlesztett programozási nyelv, amely hasonlít a JavaScript-re, de statikus típusdeklarációkat használhatunk a hibák megelőzésére. Emiatt változókhöz, függvényekhez és objektumokhoz típusokat kell deklárnunk, így könnyebben kiszűrhetjük a különböző hibákat, típushibákat. Ilyen típusok lehetnek például a: *Number*, *Array*, *Tuple*, *Boolean*, *String*, stb. Lehetőségünk van TypeScript-et írni típusok nélkül is (*any* típus használata), viszont így elvesztődik a nyelv jellegzetessége, így amennyiben lehetséges kerülnünk kell az ilyen eshetőségeket. Ahogy az ábra is mutatja (ábra 5) lehetőségünk van különböző interfészeket, osztályokat, változó típusokat létrehoznunk.

```

1  class Student {
2    fullName: string;
3    constructor(public firstName: string, public middleInitial: string, public lastName: string) {
4      this.fullName = firstName + " " + middleInitial + " " + lastName;
5    }
6  }
7
8  interface Person {
9    firstName: string;
10   lastName: string;
11 }
12
13 function greeter(person: Person) {
14   return "Hello, " + person.firstName + " " + person.lastName;
15 }
16
17 let user = new Student("Jane" "M" "User");
18   function greeter(person: Person): string
19   document.body.textContent = greeter(user);

```

ábra 5 – TypeScript mintakód (Forrás: [10])

### 3.3.3. Node.js

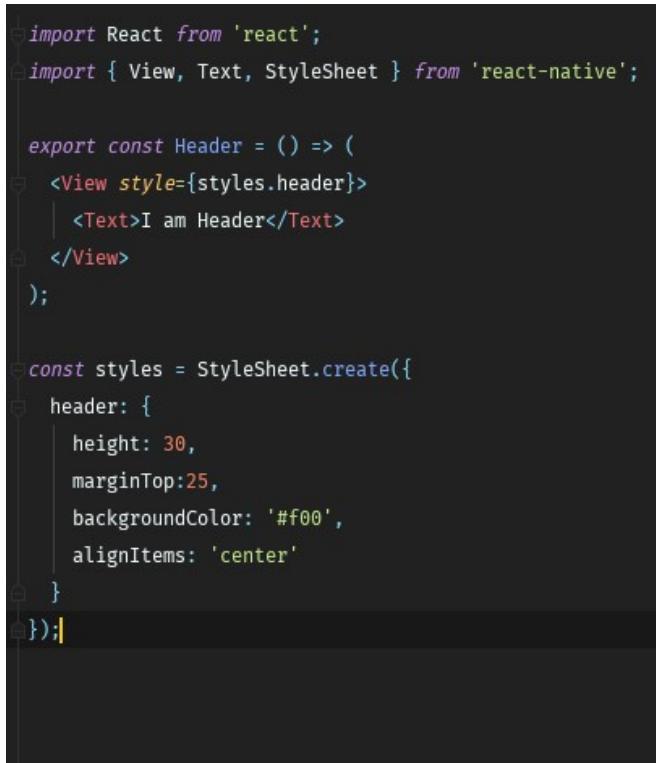
A Node.js egy szerver oldali platform, mely a Google Chrome JavaScript Engine (V8 Engine) -re épül. Emiatt a végrehajtási idő nagyon gyors és megbízható. A Node.js alkalmazások JavaScript-ben íródnak, ezért nagyban megkönnyíti a fejlesztők munkáját, hiszen egy nyelv segítségével lehetőségünk van kliens és szerver oldali alkalmazásokat létrehozni. Ezenkívül egyaránt futtathatóak OS X, Microsoft Windows és Linux rendszereken is.

A Node.js által könnyedén tudunk létrehozni olyan alkalmazásokat, amelyek egy adatbázissal kommunikálnak (pl: MongoDB). Amikor egy kliens egy kérést (request-et) küld, a kérés először a szervernek lesz továbbítva, amely feldolgozza és érvényesíti azt. Ha ez megtörtént, akkor a válasz elküldésre kerül a kliens számára.

### 3.3.4. React Native

A React Native egy Facebook által fejlesztett programozási keretrendszer, mely lehetővé teszi a mobil alkalmazások cross-platformra való fejlesztését. A React Native JavaScriptet használ az alkalmazás felhasználói felületének összeállításához. Az összetettebb funkciók esetében lehetőségünk van a kód implementálását natív-OS nyelveken (Swift és Objective-C iOS esetén, illetve Java és Kotlin Android esetén) megírni. A fő erőssége a keretrendszernek, hogy egy kód bázist felhasználva ugyanazt a működést és felhasználói felületet tudjuk elérni úgy iOS-en, mint Android-on.

Ez a keretrendszer egy úgynevezett híd (bridge) koncepciót használ, így aszinkron módon kommunikál egymással a JavaScript nyelv a natív elemekkel.



```
import React from 'react';
import { View, Text, StyleSheet } from 'react-native';

export const Header = () => (
  <View style={styles.header}>
    <Text>I am Header</Text>
  </View>
);

const styles = StyleSheet.create({
  header: {
    height: 30,
    marginTop: 25,
    backgroundColor: '#f00',
    alignItems: 'center'
  }
});
```

ábra 6 – React Native mintakód (Forrás: [11])

Ahogy a képen is látható (ábra 6.) egy függvényt (Header) szükséges létrehoznunk annak érdekében, hogy natív elemeket tudjunk megjeleníteni. Ha ez megtörtént, akkor egy View-ba kell beleraknunk azokat az elemeket, amelyeket felszeretnénk használni, ilyen például a Text, Image, FlatList, TextInput, ScrollView, Button, Switch, stb. Egy View-t tudunk kinézetileg testre szabni, illetve egy oldalon megjeleníteni. A Text komponens segítségével tudunk szöveget megjeleníteni. A View és a Text komponensekre képesek egy stílust adni a StyleSheet nevezetű komponens-el, így tudjuk elérni a kívánt kinézetet, eredményt.

A DEX alkalmazásomat React Native-ban írtam, mivel egy nyári gyakorlat keretin belül lehetőségem volt megismerkedni a nyelvvel, így megkedvelteim azt. Megtetszett, hogy minden könnyen lehet mobil alkalmazásokat létrehozni úgy iOS-re, mint Androidra is. A nyelv egyszerűsége, logikus felépítése és a rengeteg fejlesztő által karbantartott különböző csomagok (package-k) rengeteget segítenek a gyors és hatékony munkavégzéshez. Így könnyedén lehetett egy blokklánc technológiára alapuló kriptovaluta váltó alkalmazást létrehozni.

Példa React Native nyelvben megírt mobil alkalmazásokra: Facebook, Messenger, Microsoft Office, Skype, Shopify, Discord, PlayStation App, Tesla, Pintereset.

### **3.3.5. Expo**

Az Expo egy olyan eszközökészlet vagy keretrendszer, amely direkt React Native-hoz lett kialakítva, annak érdekében, hogy megkönnyítse és felgyorsítsa a mobil alkalmazás fejlesztését. Ennek segítségével percek alatt létrehozhatunk egy alkalmazást, amely a saját eszközünkön is fut. Rengeteg olyan beépített eszközzel rendelkezik, amely elősegíti a fejlesztést, mint például a Hot Reload, Debugger Mode, Performance Monitor stb. Rengeteg könyvtárral rendelkezik, így nem kell megírnunk teljesen a nulláról bizonyos részeit az alkalmazásunknak, csupán fel kell használnunk ezeket. Pontos és naprakész dokumentációval lát el bennünket. Ha azt szeretnénk, hogy a saját eszközünkön fusson az általunk megírt alkalmazás, akkor csupán le kell töltenünk az Expo Go nevezetű alkalmazást, majd egy QR kódot kell bescannelnünk, amelyet az alkalmazásunk generál ki. Az Expo használatával könnyedén tudunk a Google Play Áruházba és az App Store-ba applikációt feltölteni, ugyanis a szükséges kulcsokat és aláírás hitelesítést mind elvégzi helyettünk. Nagyon sok mindenben megkönnyítette az alkalmazásom fejlesztését az Expo Go, hiszen így nem kellett külön emulator-t használjak ahhoz, hogy az alkalmazásomat tesztelhessem, hanem egyszerűen a saját eszközömön Wifi segítségével tudtam élesben tesztelni.

### **3.3.6. MongoDB**

A MongoDB egy dokumentumalapú adatbázis-kezelő rendszer. Ezek a dokumentumok jól strukturáltak, és JSON-ként vagy bináris JSON-ként lehet őket elmenteni. Ez azért különbözik a többi tárolótól, mert nem táblázatszerűen rendezi az információkat, hanem egyedi dokumentumokként.

Mivel a MongoDB egy nem relációs adatbázis, ezért nem használ táblákat, mint a hagyományos adatbázisok. Ehelyett sajátos módon tárolja az adatokat, gyűjteményekben (collections) és dokumentumokban (documents). Az adatbázis lehetővé teszi az adatok lekérdezését JavaScript objektumok vagy a nyelvtől független MongoDB Query Language használatával. Különböző módokat biztosít a MongoDB az adatok biztonságos hozzáféréséhez, mint például az indexelés, összesítés és egyéb műveletek.

```

1  _id: ObjectId('63cbbf37c8cd85dc88f318db')          ObjectId
2  name: "Ethereum"                                     String
3  symbol: "ETH"                                       String
4  address: "0xB4FBF271143F4FBf7B91A5ded31805e42b2208d6" String

```

CANCEL UPDATE

*ábra 7 – Példa egy egyszerű MongoDB token documentum-ról*

Ahogy a képen látható (ábra 7) egy dokumentum rendelkezik egy saját `_id`-val amelyet a MongoDB rendszer automatikusan generál ki, ezáltal biztosít egy egyedi azonosítót. Ennek az `_id`-nak saját típusa van, egy úgynevezett MongoDB ObjectId. Egy dokumentumban minden egyik mezőnek (field-nek) van egy értéke, amely lehet String, Int, Double, Boolean, Array, stb.

### 3.3.7. Solidity

A Solidity egy objektum-orientált programozási nyelv, melyet az Ethereum hálózat csapata hozott létre annak érdekében, hogy smart contract-okat<sup>5</sup> tudjuk megtervezni és létrehozni. Különböző felhasználási területeken alkalmazzák, mint például az NFT-k, Decentralized Finance (DEFI) alkalmazások és Decentralized Apps (dApps) alkalmazások. A smart contract-ok fél-automatizált egységek, vagyis szükséges egy emberi személy, aki interakcióba lép a contract-al, viszont nem szükséges emberi beavatkozás ahhoz, hogy végrehajtódjon, emiatt nem lehet ezeket manipulálni vagy módosítani. Alapvetően az Ethereum blokkláncre volt kifejlesztve a Solidity nyelv, viszont a népszerűsége miatt annyira kinőtte magát, hogy más blokkláncok is támogatják, mint például a Tron, Tendermint vagy a Kin. A Solidity nyelv egy erősen típushatigénylő nyelv, ezért fontos, hogy a változókat megfelelő típussal lássuk el, mint például: int256, uint256, bool, address, byte\_array, string, stb. Ahhoz, hogy egy smart contract-ot tudunk használni, először szükséges, hogy azt lekompilláljuk, majd úgymond feltöltsük (deploy-oljuk) az Ethereum blokkláncre, amelyet majd az EVM (Ethereum Virtual Machine) értelmez, majd lefordítja gépi kódra. Ha ez megtörtént akkor tudunk interakcióba lépni a saját smart contract-unkkal és így tudjuk elérni azokat az adatokat és függvényeket, amelyek benne találhatóak.

---

<sup>5</sup> smart contract: egy olyan kódbázis vagy függvények összesége, mely egy saját címmel (address-el) rendelkezik az Ethereum blokkláncon.

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract TestContract {
5     uint256 public myNumber;
6
7     function setNumber(uint256 newValue) public {    22520 gas
8         myNumber = newValue;
9     }
10
11    function getNumber() public view returns (uint256) {    2459 gas
12        return myNumber;
13    }
14 }
```

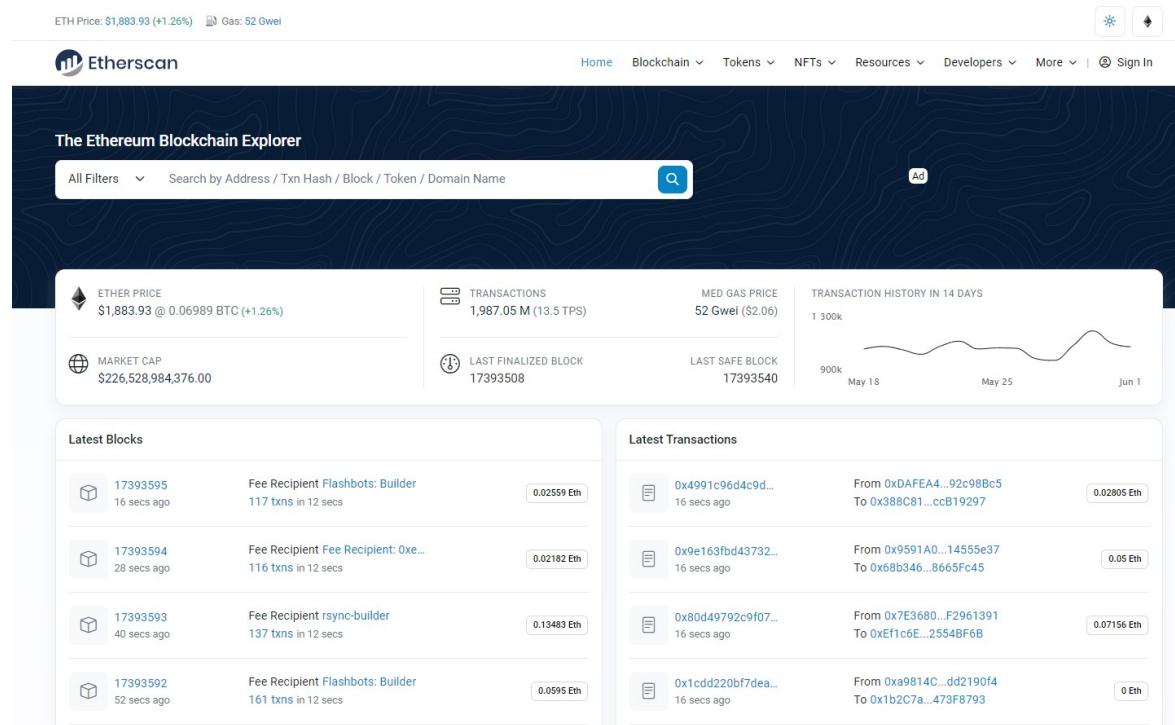
*ábra 8 – Solidity mintakód*

A képen található (ábra 8) programész magyarázata: a **pragma** (2. sor) kifejezés által tudjuk megmondani a fordítónak, hogy milyen verziót használjon a fordítás során. Ezután a **contract** (4.sor) kulccsóval határozzuk meg azokat az adatokat, illetve függvényeket, amelyeket szeretnénk, hogy a smart contract tároljon egy saját Ethereum címen. Ezután egy **uint256**-os típusú **myNumber** nevezetű változóban (5. sor) tárolunk egy számot. Mivel ez a változó egy **public** kulccsóval van ellátva, ezért az értékét egyenesen eltudjuk érni a contract meghívása során, így nincs feltétlenül szükség a **getNumber** metódusra. Ezt a változót tudjuk a **setNumber** (7. sor) nevezetű függvénnnyel módosítani, illetve a **getNumber** (11. sor) függvénnnyel ugyancsak lekérni.

### 3.3.8. Etherscan

Az Etherscan egy olyan platform, amely az Ethereum blokkláncra épül. Itt láthatjuk a blokklánc aktuális állapotát. Különféle tranzakciókat nyomon követhetünk, illetve részletesen megtekinthetjük őket. Hasonlóan működik, mint egy kereső motor, hiszen különböző paraméterek alapján keresni tudunk, mint például: Address, Txn Hash, Blokk, Token, Domain cím. Mivel minden smart contract rendelkezik egy Ethereum címmel, ezért ezen az oldalon részelesen megtekinthetjük annak működését, hiszen minden feltöltött contract teljes mértékben publikus. Ezen az oldalon lehetőségünk van a smart contract-unkat meghívni, tesztelni és ellenőrizni a helyességét. Az alkalmazásom elkészítése során sokat segített abban,

hogy megértsem az Ethereum blokklánc működését, illetve abban, hogy tesztelni tudjam az alkalmazásomat és más hasonló rendszereket.



ábra 9 – Etherscan weboldal

### 3.3.9. Alchemy

Az Alchemy egy blokklánc fejlesztői platform, amely segítségével alkalmazásokat tudunk létrehozni az Ethereum blokkláncon. Az Alchemy lehetővé teszi a fejlesztők számára, hogy könnyedén hozzáérhessenek a blokklánchoz, egy node<sup>6</sup> segítségével. Ezáltal az API által tudunk interakcióba lépni a blokklánc aktuális változatával, így nem kell a saját számítógépünkre letöltenünk majd szinkronizálnunk a blokkláncot, hanem a kommunikációt egy API kulccsal, HTTPS kérésekkel vagy WebSocket segítségével hajtuk végre. Az API lehetővé teszi, hogy nyomon követhessük a blokkláncon történő eseményeket, mint például a tranzakciók küldését, fogadását, az aktuális blokkszámot, az aktuális gas értékét stb. Az Alchemy nagyban megkönnyítette az alkalmazásom és a blokklánc közötti kommunikációt, mivel gyors és

<sup>6</sup> node: egy olyan számítógép, amelyen az Ethereum hálózathoz futtatásához szükséges szoftver fut.

megalapító eszközök bizonyságtartók, így nem kellett időt és energiát befektetni abba, hogy lokálisan rendelkezzem az Ethereum blokklánccal.

### **3.3.10. Goerli Testnet**

A Goerli Testnet egy nyílt forráskódú, közösségi alapú projekt, mely lehetővé teszi egy teszthálózat segítségével az Ethereum alapú alkalmazások futtatását és tesztelését. Ezek olyan hálózatok, melyek nagy mértékben hasonlítanak a valós Ethereum hálózathoz, de virtuális ETH (Ether)-el működnek. Így a fejlesztők különböző funkcionálitásokat és smart contract-okat tudnak tesztelni, mielőtt a valódi Ethereum hálózatra töltenék fel őket. Ezáltal rengeteg hibát és nem kívánt működést tudnak kiküszöbölni. Több publikus teszthálózat is létezik, mint például az Olympic, Morden, Ropsten, Kovan és a Rinkeby. Mivel egy teszthálózatról van szó, ezért a rajta található tokenek limitálthatók, mint a valódi hálózaton, illetve más árban lehet hozzájuk jutni. Az alkalmazásom fejlesztése során nekem is szükségem volt egy teszthálózatra, hiszen az Ethereum hálózatra való feltöltés valódi pénzbe kerül, illetve minden tranzakció után szükséges fizetnünk egy úgynevezett “gas fee-t”, amely ugyancsak pénzbe kerül. A Goerli Test-netre esett a választás hiszen ez volt a legyakrabban használt hálózat, illetve ezt a hálózatot támogatta úgy a Metamask<sup>7</sup>, mint az Alchemy is. A Goerli Test-net is rendelkezik egy saját Etherscan-nel így nyomon tudjuk követni az ott lévő tranzakciókat, smart contract-okat. Ahhoz, hogy saját ETH-val rendelkezzünk ezen a teszthálózaton, szükséges igényelnünk azt, egy úgynevezett “faucet” -en (csapon) keresztül. Ahhoz, hogy igényelni tudunk ETH-t szükségünk van egy kriptotárcára, pontosabban egy Ethereum címre, amelyre a rendszer kiküldi az igényelt ETH értéket. Léteznek olyan “faucet” -ek, amelyek POW-on (Proof of Work) alapszanak. Ez egy védelmi módszer, amely biztonságosabbá teszi a pénz kezelését. Sajnos az év végeire a Goerli Testnet lezárásra kerül, ezért azt ajánlják, hogy a Sepolia teszthálózatot használjuk helyette.

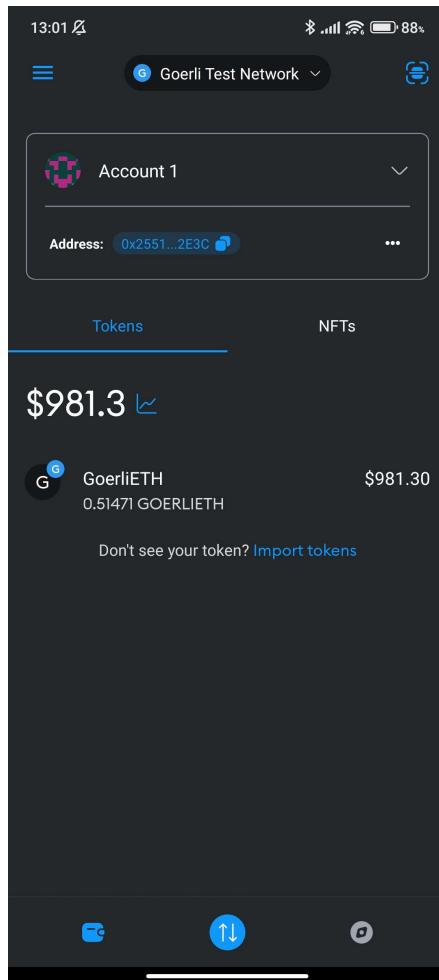
### **3.3.11. Metamask**

A Metamask egy kriptotárcá, amely böngészőbővítményként és mobil alkalmazásként érhető el. Ahogyan a való világban is szükség van egy pénztárcára ahhoz, hogy tárolni, illetve kezelni tudjuk a pénzüket, így a Metamask is egy olyan eszköz, amellyel hozzáférhetünk a

---

<sup>7</sup> Metamask: virtuális kriptopénztárcá

tokeneinkhez, ennek segítségével tudunk interakcióba lépni decentralizált alkalmazásokkal és ezáltal lehetőségünk van Ethereum-mal kereskedni. Bárki létrehozhat egy Metamask pénztárcát, még e-mail címre sincs szükség, csak egyszerűen egy jelszóra. A képen (ábra 10) látható a Metamask Android-os verziója. Itt megtudjuk tekinteni az egyenlegünket, és kiválaszthatjuk azt a fiókot, amellyel kereskedni szeretnénk. Ezenkívül kitudjuk választani, hogy milyen hálózaton szeretnénk tevékenykedni és a fiókunkhoz társított Ethereum címet is láthatjuk. (0x2551...2E3C)



ábra 10 – Metamask mobilapplikáció

Ebben az alkalmazásban lehetőségünk van tranzakciókat aláírni (sign), tranzakciókat kezdeményezni, illetve részletesen megtekinteni őket. Megnézhetjük azokat az NFT-ket<sup>8</sup>, amelyekkel rendelkezünk, azokat az tokenek-et (coin-okat) amely a pénztárcánkon van, illetve

<sup>8</sup> NFT: egy digitális token, amely a blokklánc segítségével igazolja és azonosítja azokat a tartalmakat, amellyel rendelkezünk.

Rengeteg kriptotárca létezik, mint például a Trust Wallet, Coinbase Wallet, Atomic, Electrum stb. Azért választottam a Metamask-ot, mivel ez a bizonyult a legmegbízhatóbbnak és ezzel tudtam könnyedén összekötni a DEX alkalmazásomat egy pénztárcával. Emellett több mint 700 000 token-t támogat és legfőképp támogatja a Goerli teszthálózatot, amely fő részét képezi az alkalmazásomnak.

### **3.3.12. Tenderly**

A Tenderly egy fejlesztői platform, amely elősegíti az Ethereum smart contract-ok fejlesztését, tesztelését és monitorozását. E platform által tudunk egy smart contract-ot mélyebbre szemügyre venni, betekintést nyújt a smart contract részletes működésében. Lépésről lépére mutatja, hogy mi történik egy smart contract-on belüli függvényel vagy változóval, így nagyban elősegíti a hibakeresést. Ha valamilyen hibába ütköztem, vagy nem sikerült egy bizonyos tranzakció, akkor a Tenderly által könnyedén sikerült azonosítanom, majd javítanom a hibát.

### **3.3.13. WalletConnect**

A WalletConnect egy olyan protokoll és egy nyílt forráskódú szabvány, mely lehetővé teszi a böngészőalapú kriptotárcák és a decentralizált alkalmazások (dApp-ok) közötti biztonságos és gyors kommunikációt. A WalletConnect segítségével a felhasználóknak lehetőségük van arra, hogy csatlakozzanak különböző dApps-hoz az okostelefonjukon vagy egy

másik kriptovaluta tárcán keresztül, és arra, hogy biztonságosan tudják kezelni a tranzakcióikat. Az alkalmazásoknak csak az engedélyezett műveleteket kell végezniük, és a felhasználók a kriptovaluta tárcájukban adnak jóváhagyást a tranzakciók végrehajtásához. A protokoll lehetővé teszi a felhasználóknak, hogy egyszerűen és biztonságosan csatlakozzanak és kezeljék a dApps-okat a saját kriptovaluta tárcájukból anélkül, hogy kiadnák a privát kulcsaikat vagy megbízzanak harmadik fél közvetítő szolgáltatásokban.

### **3.3.14. Hardhat**

A Hardhat egy olyan keretrendszer, amelyet az Ethereum smart contract-ok és dApp-ok (Decentralizált alkalmazások) fejlesztéséhez alakítottak ki. A Hardhat lehetővé teszi a fejlesztők számára az okosszerződések lokális fejlesztését, tesztelését és hibakeresését. A Hardhat integrálható más népszerű fejlesztői eszközökkel és tesztelési keretrendserekkel, így könnyedén alkalmazható az Ethereum-alapú projektek fejlesztéséhez és hibakereséséhez. A Hardhat segítségével sikerült a Solidity programozási nyelvben megírt smart contract-jaimat lekompillálni majd ezeket feltölteni a Goerli Testnet-re.

## **4. A rendszer specifikációi**

Annak érdekében, hogy a rendszert megfelelően használhassuk szükséges, hogy néhány előfeltételnek eleget tegyünk. Ezekről lesz szó a következőkben.

### **4.1. Felhasználói követelmények**

A legfontosabb követelmény, hogy a felhasználó rendelkezzen egy olyan mobiltelefonnal, amelyen iOS vagy Android rendszer fut. Ha ez megvan, akkor az alkalmazás elindításához szükséges lépések után megfog jelenni a főoldal. A főoldalon (Home) egy kilistázást talál az aktuális, illetve népszerű kriptovalutákról. Görgetve közöttük, megtekinthető ezeknek a neve, rövidítése (symbol), jele (icon), aktuális ára, illetve a 24 órához viszonyított árváltozása. A megszokotttól eltérően az alkalmazásban nincs regisztráció, illetve bejelentkezés, hiszen minden blokklánc hálózata által van megvalósítva. Éppen ezért térjünk rá a második fontos követelményre, amely ezzel kapcsolatos.

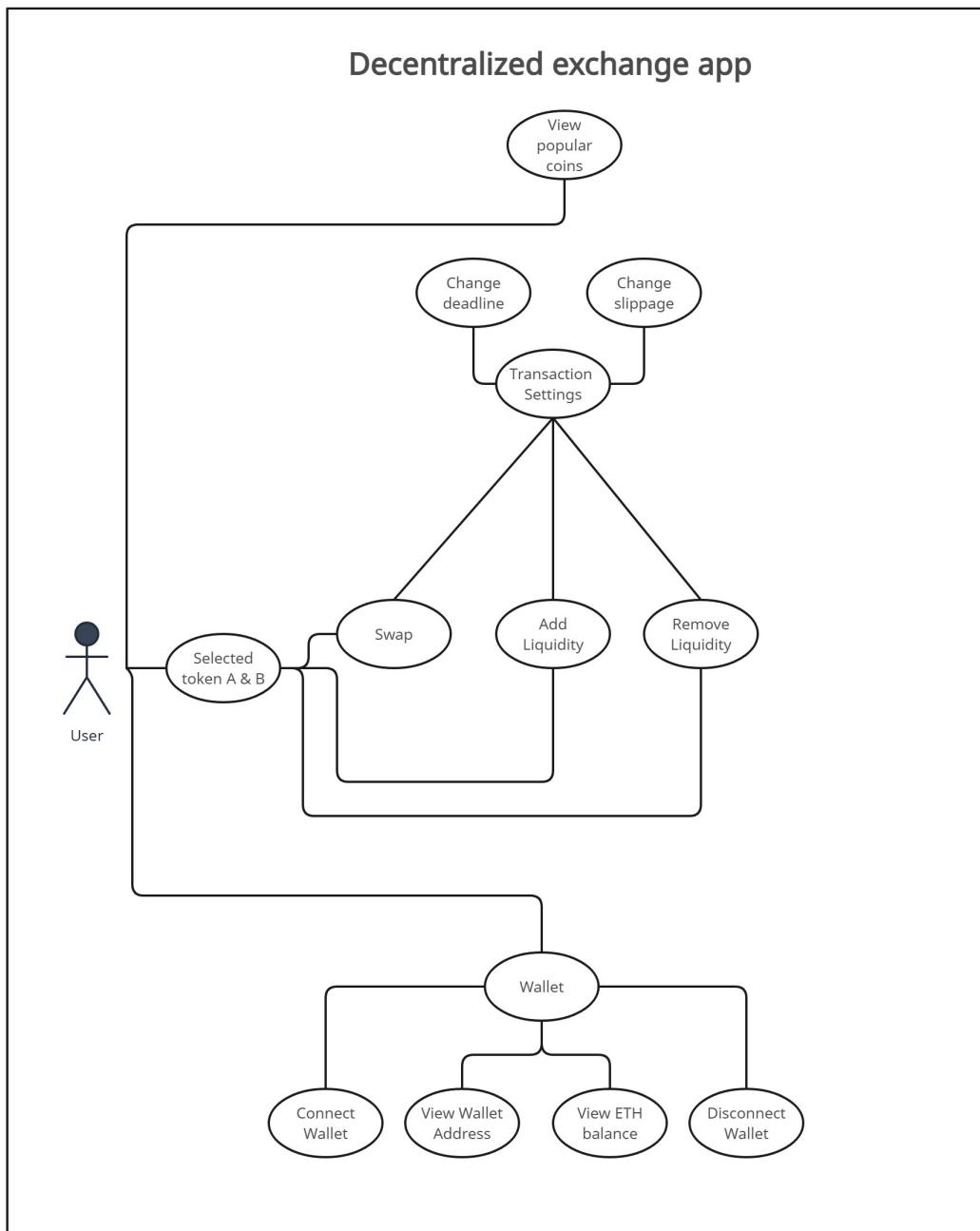
Létfontosságú, hogy a felhasználó rendelkezzen egy kriptotárcával, hiszen enélkül az alkalmazás fő funkcionálitása nem lesz elérhető. Egy kriptotárcát egyszerűen csak telepíteni kell

a rendszerre OS-től függően. Egy kriptotárca létrehozásához nincs szükség e-mailre, csak egyszerűen egy jelszóra. A kompatibilitás okáért, mivel az alkalmazásom a WalletConnect csomagra épül, ezért a legjobb módja annak, hogy ne ütközzünk hibába, hogyha a **Metamask** applikációt használjuk a DEX használata során.

A következő fontos követelmény, hogy a felhasználónak a pénztárcáján legyenek olyan kriptovaluták, melyek kompatibilisek a Goerli Testnet-el és az alkalmazással. Emellett fontos, hogy Goerli Ethereum-mal is rendelkezzünk, hiszen enélkül nem leszünk képesek fizetni a Testnet-en lévő tranzakciós díjakat. Mivel egy Testnet-ről van szó, ezért az ott megtalálható kriptovaluták száma és típusa nem mindenkor egyezik meg a valódi Ethereum hálózattal. Ez okból kifolyólag ki kellett választanom néhány kriptovalutát, amelyek megfelelnek az elvárt működésnek. A rendszerben megtalálható kriptovaluták, amelyekkel tesztelhető az alkalmazás a következő:

- ETH (Goerli Ethereum)
- WBTC (Wrapped Bitcoin)
- UNI (Uniswap)
- DAI (DAI stablecoin)
- WETH (Wrapped Ethereum)

Ezek mellett fontos megemlíteni, hogy szükséges internet kapcsolat az alkalmazás futtatásához, illetve szükséges valamely böngésző telepítve legyen a mobileszközre. Ez lehet akár Chrome, Edge, Safari, Opera stb. Azért fontos ez, mivel egy általunk végrehajtott tranzakciót interneten keresztül egy böngésző által lehet megtekinteni.



*ábra 11 - Alkalmazás use-case diagramja*

## 4.2. Rendszer követelmények

### 4.2.1. Funkcionális követelmények

- **Wallet kezelése:**
  - *Connect:* A felhasználó szükséges, hogy rendelkezzen egy virtuális pénztárcával, amelyet a mobil alkalmazáshoz tud hozzákötni. Enélkül az alkalmazás lényeges része használhatatlan, hiszen csak a főoldalt fogja látni, ahol különböző kriptovaluták és azok árai látszanak. Ahhoz, hogy egy wallet-et tudjon hozzá csatolni az alkalmazáshoz, akkor először a mobileszközén fel kell legyen telepítve egy wallet alkalmazás. Emellett el kell navigáljon a Settings oldalra, ahol ezt megteheti. Ha sikeresen csatlakoztatta a wallet-jét, akkor láthatóvá válik az egyenlege, illetve az address-e. Ezután az alkalmazásban hozzáférést kap a kereskedési funkciókhoz.
  - *Disconnect:* A felhasználónak lehetősége kell legyen, hogy a wallet-jei között tudjon váltogatni, így biztosítanunk kell azt, hogy a meglévő wallet-jét tudja lecsatlakoztatni az alkalmazástól. Ha ez megtörtént akkor egy újból párosítás segítségével egy másik wallet-et tud hozzákötni a rendszerhez.
- **Kereskedési funkciók:**
  - Az alkalmazás lehetőséget kell biztosítson a felhasználó számára, hogy különböző kereskedési funkciókat tudjon végrehajtani, mint például a **swap**, **add liquidity** és a **remove liquidity**. Ez a három fő alkotó eleme az alkalmazásnak.
  - *Swap:* A Swap keretein belül lehetősége lesz kiválasztani két token-t, amelyet swap-olni szeretne egyikről a másikra. Ezt két input mezőt segítségével teheti meg, ugyanis a kívánt értéket, amelyet szeretne swap-olni azt ide írhatja be. Fontos megjegyezni, hogy NEM a valódi értéket fogja tudni a felhasználó beírni, amelyre majd később kitérek, hogy miért is van ez így. Ha a felhasználó beírta az egyik input mezőbe a kívánt értéket, akkor a másik input mező értéke automatikusan kiszámítódik. Ezután a felhasználó a Swap gombra kattintva végrehajtja a swap-ot.
  - *Add liquidity:* Az Add liquidity a második fontos követelmény, hiszen ezen keresztül tudunk új pool-okat, illetve likviditást hozzáadni a rendszerhez.

Hasonlóan a Swap-hoz itt is két input mező fog a rendelkezésünkre állni, ami után kiválasztottuk a két token-t. Ha már létezik az a pool, amelybe likviditást szeretnénk hozzá adni, akkor a rendszer automatikusan kiszámolja a másik input mező értékét. Ha nem létezik, akkor ránk van bízva, hogy mekkora kezdő értékkel szeretnénk a pool-t feltölteni. Fontos, hogy ugyanolyan arányba tegyük ezt meg. A minimális likviditás az 1100 token.

- *Remove liquidity*: A Remove liquidity által lehetőségünk van eltávolítani a hozzáadott likviditást a saját pool-ból. Először is fontos, hogy kiválasszuk azt a két token-t, amelyhez likviditást adtunk hozzá. Ha ez megvan, akkor egy csúszka által eldönthetjük, hogy a hozzá adott likviditásnak hány százalékát szeretnénk kivenni.
  - Egy másik fontos opció, amely részét képezi a kereskedésnek, illetve az alkalmazásnak, hogy a tranzakciókat testre szabhatjuk. Ez azt jelenti, hogy minden egyes tranzakció, amely a swap rendszeren belül történik, beállíthatjuk, hogy mekkora deadline-al, illetve slippage-el szeretnénk őket elküldeni.
- 
- **Kriptovaluták megtekintése:**

A főoldalon láthatunk különböző ismert, illetve kevésbé ismert kriptovalutákat, láthatjuk azok árait, illetve árkülönbségeit az elmúlt 24 órához képest. Emellett azokat a kriptovalutákat, amelyekkel rendelkezünk, megtekinthetjük az egyenlegüket a Swap menüpont alatt.

#### 4.2.2. Nem-funkcionális követelmények

A következő rendszerigényű feltételeknek kell eleget tennünk, amennyiben szeretnénk a DEX mobil alkalmazásunkat futtatni:

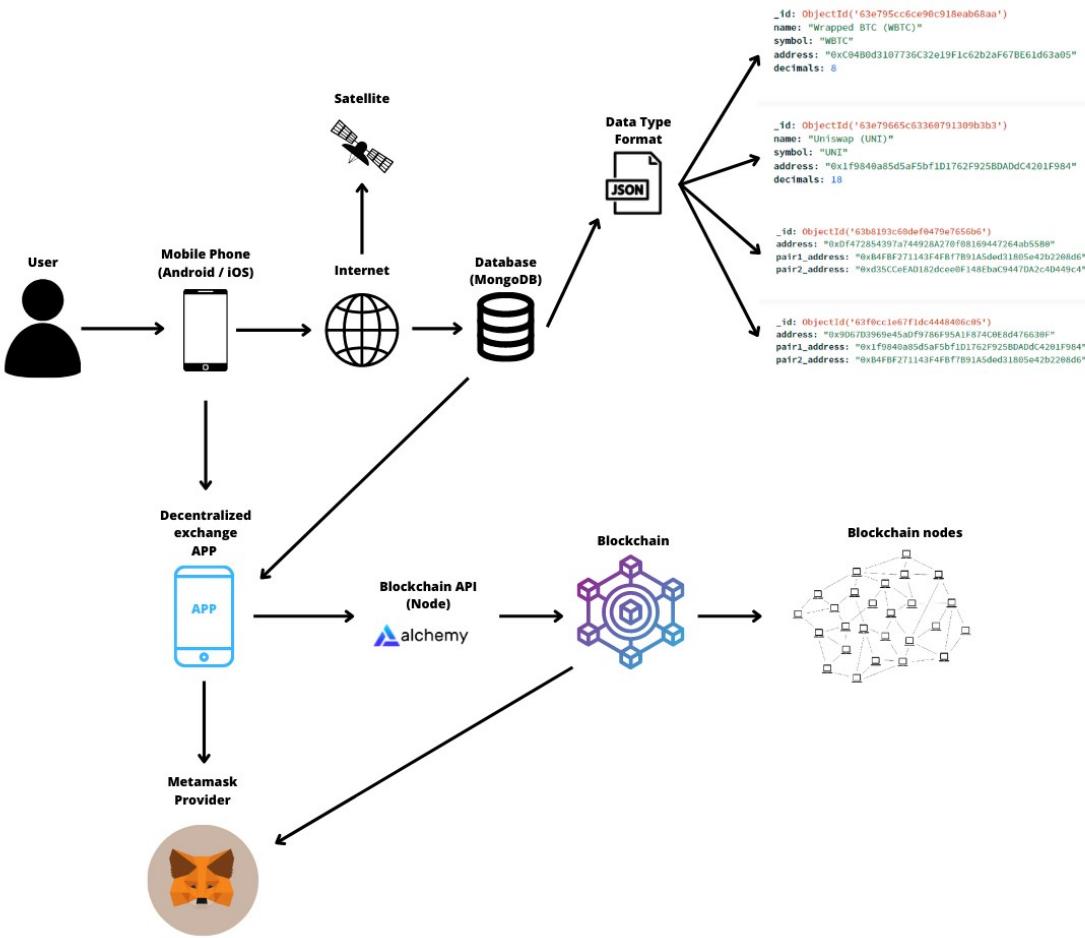
- Android: bármilyen Androidos eszköz, amely rendelkezik legalább **Android 5.0** verzióval
- iOS: olyan iOS eszköz, amely rendelkezik **iOS 13** verzióval
- Tárhely: szükséges legalább **50 MB** szabad tárhely
- Alkalmazások:

- Ha lokálisan szeretnénk futtatni az alkalmazást, akkor szükséges, hogy az Expo Go alkalmazás fel legyen telepítve az eszközünkre.
- Szükséges, hogy egy kriptotárca alkalmazással is rendelkezzünk. Ebben az esetben erősen ajánlott a Metamask kriptotárca, amely elérhető úgy iOS-re, mint Androidra.

Fejlesztéshez szükséges követelmények:

- GitHub követőrendszer
- Node.js és a szükséges csomagok telepítése
- Egy fejlesztői környezet, mint például a Microsoft Visual Studio Code

## 5. A rendszer architektúrája és üzembe helyezése



ábra 12 - DEX alkalmazás architektúrája

A rendszer architektúrája a 12-es ábrán látható. Láthatjuk amint egy felhasználó egy telefon és internet segítségével hozzáfér az alkalmazáshoz, amely kommunikál egy adatbázissal, illetve a blokkláncjaloggal egy node-on keresztül. A rendszert backend, illetve frontend részre bonthatjuk, hiszen a mobil alkalmazás része felel a kinézetért, míg a blokkláncjaloggal való kommunikációt a backend végzi.

## 5.1. Backend

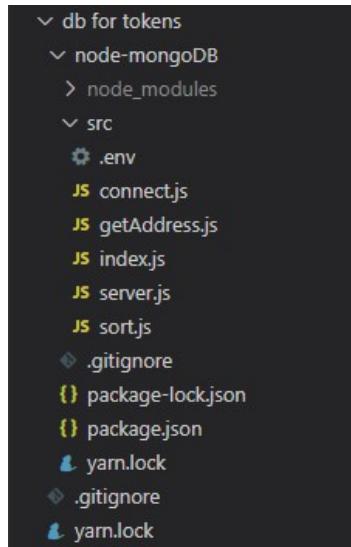
A backend két részből tevődik össze: adatbázis-kezelő rendszer és blokklánckal való kommunikáció.

### 5.1.1. Adatbázis-kezelő rendszer

Annak érdekében, hogy láthassuk és kereshessünk a meglévő tokenek-el a rendszerben, szükséges volt egy minimális adatbázis létrehozása. Az adatbázis azt a célt szolgálja, hogy képesek legyünk eltárolni a ERC20-as tokenek-et, illetve azokat a pool-okról információkat, amelyek léteznek a rendszerben. Azt a szervert, amely kiszolgálja a klienst Node.js segítségével hoztam létre. Az adatok pedig MongoDB-ben vannak eltárolva. Az adatbázis cluster-ben van eltárolva, amely célja, hogy bárhonnan elérhető legyen. Ahhoz hogy könnyedén hozzáférjünk a rendszerhez először Github-ról szükséges klónozzuk a repository-t [14], majd a mobile-app nevezetű branch-re kell váltanunk és végső soron adatbázis csatlakozáshoz szükséges útvonalat egy “`.env`” fájlba kell beleírjuk [15].

A szerver eléréséhez egyszerű HTTP kérésekre van szükség. Ha elszeretnénk indítani a szervert, akkor először is szükséges telepítenünk a Node Package Manager-t (NPM). Ezután, a következő parancsokat kell lefuttatnunk, annak érdekében, hogy működésre bírjuk a szervert:

- **npm install:** ezzel a parancssal automatikusan feltelepít minden szükséges package-t a rendszer.
- **node server.js:** ezzel elindítjuk a szerver-t, amely lokálisan fog futni.



*ábra 13 - Node.js szerver mappa struktúrája*

A szerver két endpoint-al rendelkezik, amelyet kliens oldalról elérhetünk. Az egyik a **tokenInfo**, amely lekéri az adatbázisból az összes token-t és mellé az összes információt. Ez az adatstruktúra a következőképpen néz ki:

- **id**: az objektum ID-ja
- **name**: a token neve
- **symbol**: a token szimbóluma
- **address**: a token címe

Ha fut a szerverünk, akkor ezt a következő link-en érhetjük el:

- **localhost:3000/tokenInfo**

A másik endpoint a **poolsInfo**, amely lekéri az összes pool-t az adatbázisból. Ennek segítségével eltudjuk érni azokat a pool-okat, amelyek a rendszerben vannak. A következőképpen néz ki az adatstruktúrája:

- **id**: az objektum ID-ja
- **address**: a token párt alkotó pool címe

- **pair1\_address**: az első token címe
- **par2\_address**: a második token címe

Hasonlóan az előző kéréshez a következő link-en érhetjük el ezt az endpoint-ot:

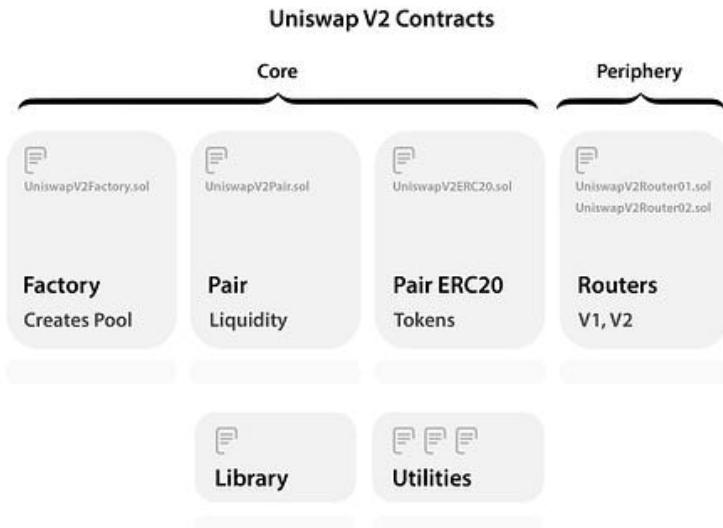
- **localhost:3000/poolsInfo**

### **5.1.2. Blokklánc kommunikáció**

Alapvetően az alkalmazásunk a blokkláncjal kommunikál, ezeken keresztül hajtja végre a tranzakciókat. A Uniswap V2 rendszere megalapozta azokat a műveleteket, amelyek szükségesek ehhez. Mivel a Uniswap V2 teljesen publikus és nyílt forráskódval rendelkezik, ezért a GitHub repository forkolásával hozzáférhetünk ezekhez a műveletekhez. Ezek után néhány módosítást végeztem, minimálisan testre szabtam, illetve kiszűrtettem a fontos funkcionálitásokat, amelyek összhangban vannak a Uniswap licensz feltételeivel [13]. Ezek a következők:

- UniswapV2Pair.sol (az én esetben DexPair.sol) fájlban eltávolítottam a “data” paramétert a swap függvényből, illetve kivettem a flash swap funkcionálitást, mivel erre nem volt szükség.
- Töröltetem az IUniswapV2Callee.sol (az én esetben DexCallee.sol) interfészét
- Töröltetem a test és example fájlokat
- minden Uni kifejezést Dex-re cseréltem
- Init hash-t csere, a pool-ok létrehozása érdekében

A továbbiakban bemutatom az architektúráját a Uniswap V2-nek, illetve a saját blokkláncra deployolt rendszeremet, útmutatókkal ellátva. A rendszer bárki számára klónozható és tesztelhető [14].



ábra 14 - Főbb Uniswap V2 contract-ok [Forrás 12]

A képen (ábra 14) két fő contract-ot találhatunk: **v2-core** és **v2-periphery**. A v2-core a Factory contract és a v2-periphery pedig a Router contract-nak felel meg. Ezekről részletesebben beszéltem a 3.1.3-as fejezetben. Ahhoz, hogy a rendszert tesztelni tudjuk, először szükséges a Factory és a Router contract-ot deploy-olni. Ez a két contract már deploy-olva lett a Goerli Testnet-re a következő címekre:

- 0xA16A20D39409112077d98c9Dc0b6f7ff93Cb059D (Factory)
- 0xE4c76722C7c5a60A62F6aF1Ec7C3C2E303c0dA4f (Router)

Ezek teljesen publikusak és megtekinthetőek az Etherscan weboldal segítségével. Ha mégis szeretnénk saját Factory és Router contract-ot deployolna, akkor ehhez a hardhat könyvtárat kell használnunk, amelyet az “**npm install hardhat**” -el tudunk telepíteni. Mielőtt nekifognánk a deploy-nak fontos, hogy beállítsuk egy `.env` fájlba a saját wallet-ünk privát kulcsát, `PRIVATE_KEY` kulcsszóval ellátva. Ha ez megtörtént, akkor két script áll rendelkezésünkre, amelyet a hardhat segítségével lefuttathatunk.

- `deploy.js` (Factory contract)
- `deployRouter.js` (Router contract)

A parancs, amivel ezek a scriptek futtathatóak a következő:

- **npx hardhat run --network goerli scripts/deploy.js**

Ha sikerült lefuttatni, akkor a terminálban meg kell jelenjen egy address, amely a kideploy-olt Factory contract-nak az elérhetősége lesz. Ezután a contract-ot és a függvényeit megtekinthetjük az Etherscan oldalon úgy, hogy egyszerűen beírjuk a kideploy-olt address-t a keresőmezőbe.

Ugyanezt a műveletet kell megtennünk a másik smart contract-ra, a Router-re. Ehhez a következő script-et kell lefutassuk:

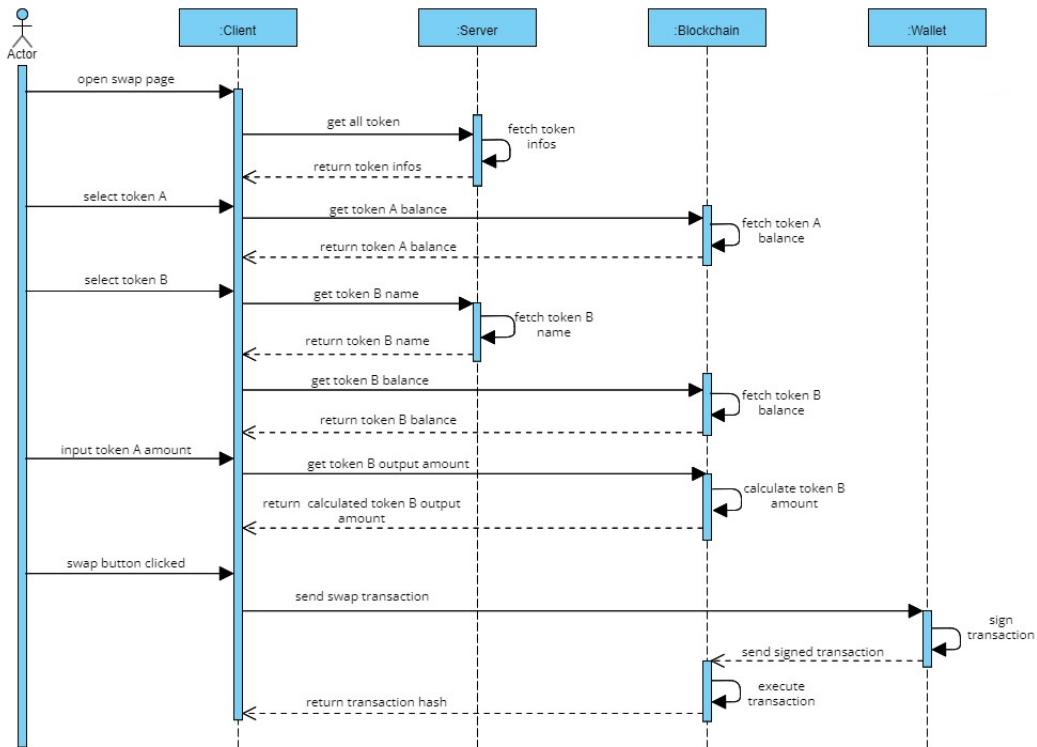
- **npx hardhat run --network goerli scripts/deployRouter.js**

Ugyanúgy ahogy előzőleg itt is a terminal-ban egy address kell megjelenjen, amelyet az Etherscan-en tudunk tovább vizsgálni.

## 5.2. Frontend

Mivel az alkalmazás React Native-ban van írva Expo környezetet használva, így a repository egyszerű klónozásával [14], illetve branch váltással (mobile-app branch) hozzáférhetünk az alkalmazáshoz. A mobil alkalmazás kommunikál a Node.js szerverrel, ahonnan a token és pool információkat tudjuk lekérni. Emellett direktbe kommunikál a Factory és Router smart contract-al az Alchemy API és Metamask segítségével. A Metamask lehetőséget biztosít a tranzakciók aláírására. Ha telepítettük az Expo Go mobilalkalmazást és elindítottuk a szervert (lásd 5.1.1 fejezet), akkor a következő parancsokra van szükségünk az alkalmazás futtatásához:

- **npm install** – minden szükséges csomagot telepít
- **npm start** – elindítja az expo alkalmazást, amelyet egy QR kód segítségével tudunk bescannelni



ábra 15 - Swap szekvencia diagram

A következő ábrán (ábra 15) egy swap művelet található, amikor is a felhasználó token A-t szeretne token B-re cserélni. Fontos megjegyezni, hogy a sorrend nem mindegy. Különbség van aközött, hogy token A-t szeretnénk token B-re cserélni, vagy token B-t szeretnénk token A-ra cserélni. Ugyanis a Router smart contract különbséget tesz ezek között, illetve aközött, hogy Ethereum-e az egyik kiválasztott token. Ez a felületen abban mutatkozik meg abban, hogy melyik bemeneti mezőbe volt írva. Ha az elsőben, akkor a token B fog kiszámításra kerülni, ha a másodikban, akkor a token A lesz kiszámítva. Visszatérve a szekvencia diagramra amint a felhasználó megnyitja az alkalmazást és hozzácsatlakoztatja a pénztárcáját, akkor lehetősége lesz swap-olni. Ha kiválasztott két token-t (A & B) akkor ezeknek az egyenlege láthatóvá válik majd, hiszen ezt a két értéket visszaküldi a kiválasztott tokenek smart contract-ja. Ezután a felhasználó egy értéket ír az első mezőbe, amely token A értékét reprezentálja. Ezután az alkalmazás elküldi ezt az értéket egy tranzakció formájában a Router smart contract-nak. Ez a contract visszaküldi a kiszámított token B értékét. A swap gomb megnyomása után a megfelelő swap metódus elküldésre kerül a contract-nak, amelyet a Metamask-al kell aláírni. Ha ez sikeresen megtörtént,

akkor a Metamask visszaküld egy transaction hash-t, amely jelképezi az elküldött tranzakciót.

```
454 //10 swap - swapExactTokensForTokens
455 const swapExactTokensForTokens = async () => {
456   const iRouter = new ethers.utils.Interface(abiRouter);
457
458   const deadline = Math.floor(Date.now() / 1000) + 60 * Number(stateDeadline);
459
460   const slippage = parseInt(stateSlippage) / 100;
461
462   const minValue = parseInt(getAmountOut) * slippage;
463
464   const remainingAmount = parseInt(getAmountOut) - minValue;
465
466   const swapABI = iRouter.encodeFunctionData("swapExactTokensForTokens", [
467     firstText,
468     Math.round(remainingAmount).toString(),
469     [getTokenAddress(fromTokenIndex), getTokenAddress(toTokenIndex)],
470     accounts[0]!,
471     deadline,
472   ]);
473
474   const tx = {
475     from: accounts[0]!,
476     to: routerAddress,
477     data: swapABI,
478     gasLimit: 1000000,
479     nonce: await alchemyProvider.getTransactionCount(accounts[0]!),
480   };
481
482   try {
483     const send = connector.sendTransaction(tx);
484     console.log("Transaction hash: ", await send);
485     setTransactionTX(await send);
486     getTokenBalance(fromTokenIndex, true);
487     getTokenBalance(toTokenIndex, false);
488   } catch (error) {
489     console.log(error);
490   }
491};
```

ábra 16 - Swap kódrészlet (*swapExactTokensForTokens*)

Az képen (ábra 16) egy kódrészlet található, mely a szekvencia diagram (ábra 15) által bemutatott swap műveletet mutatja be két token között, abban az esetben amikor a felhasználó A token-t szeretne B token-re váltani. A 456. sorban egy interfész definíálunk, amely megkapja az abiRouter-t. Ez az objektum segít az Ethereum smart contract-ok metódusainak kódolásában és dekódolásában. Ezek után a 458. sorban a deadline-t állítjuk be, amely meghatározza, hogy mennyi idő alatt kell végrehajtódjon a tranzakció. A 460., 462., 464. sorokban a slippage értékével végzünk műveletek, hogy a kiszámított token B értékét tudjuk megfelelően módosítani. A 466. sorban egy swapABI nevezetű változóban kódolva elmentjük a swapExactTokensForTokens nevezetű függvény és annak paramétereit. Ezután a 474. sorban

felépítjük a tranzakciót, amelyben különböző beállításokat végzünk annak érdekében, hogy sikeres legyen. A 482-től 490-ig terjedő sorokban pedig megpróbáljuk egy try-catch segítségével elküldeni a felépített tranzakciónkat a wallet provider-nek (Metamask a mi esetünkben), majd különböző értékeket állítunk be annak érdekében, hogy a képernyőn is megjelenjenek a változások.

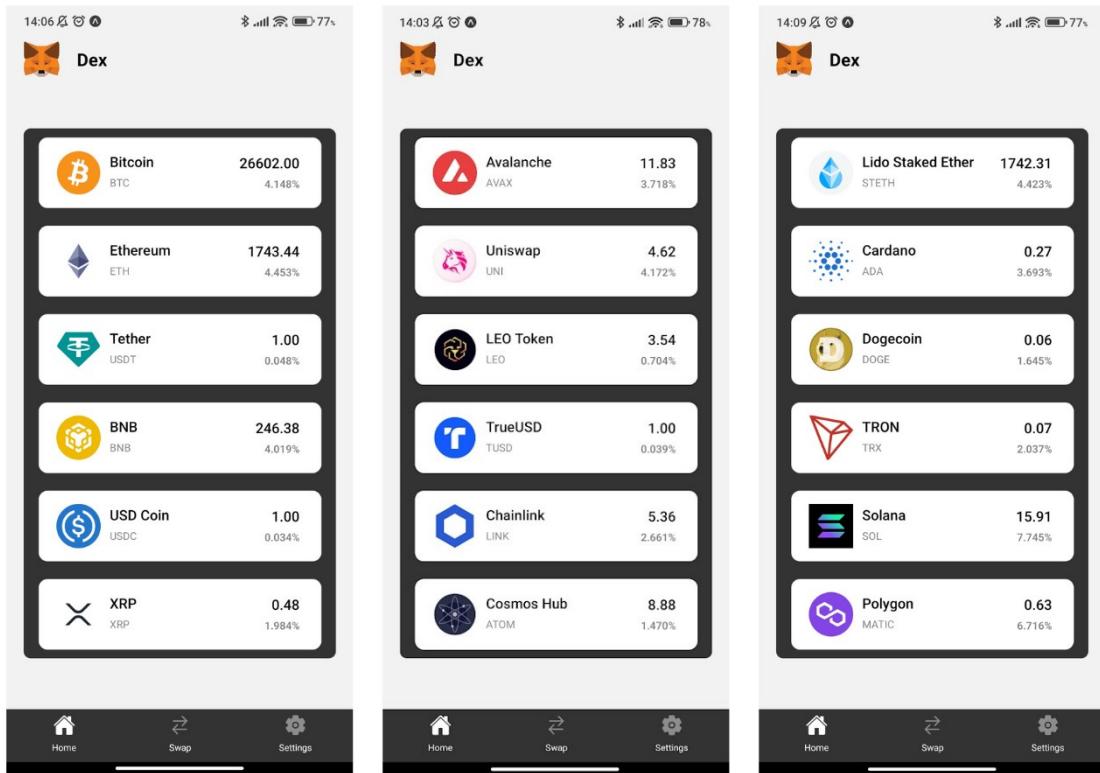
## **6. Az alkalmazás bemutatása és felhasználása**

Ebben a fejezetben a mobil alkalmazást fogom bemutatni, illetve azt, hogy milyen funkcionálisokra képes a rendszer.

### **6.1. Főoldal**

#### **6.1.1. Kriptovaluták kilistázása**

A főoldalon láthatunk egy listanézetet (ábra 17), amely görgethető. Ebben a listában különböző kriptovaluták vannak, amelyek egy API-n keresztül vannak lekérve, pontosabban a CoinGecko API-on keresztül. Ez egy teljesen ingyenes és bárki számára elérhető API, amely rengeteg információt tartalmaz kriptovalutákról.



*ábra 17 - Kriptovaluták kilistázása*

A kilistázott kriptovaluták esetében láthatjuk a szimbólumot, amely képviseli a kriptovalutát, a nevüket, az árát, illetve a 24 órához képest lévő ármozgás, amely lehet negatív, illetve pozitív.

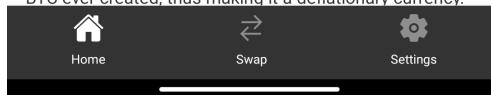
### 6.1.2. Kriptovaluták részletes nézete

Ha a főoldalon kiválasztunk egy kriptovalutát, akkor az alkalmazás átirányít egy másik oldalra (ábra 18), ahol bővebb információt találunk a kiválasztott valutáról. Ezen az oldalon láthatjuk a BTC-ben, illetve ETH-ban átszámított ár értékét. Ezenkívül különböző kifejezéseket láthatunk, amelyek az adott kriptovalutához viszonyítva változnak az értékük.



## Description

Bitcoin is the first successful internet money based on peer-to-peer technology; whereby no central bank or authority is involved in the transaction and production of the Bitcoin currency. It was created by an anonymous individual/group under the name, Satoshi Nakamoto. The source code is available publicly as an open source project, anybody can look at it and be part of the developmental process. Bitcoin is changing the way we see money as we speak. The idea was to produce a means of exchange, independent of any central authority, that could be transferred electronically in a secure, verifiable and immutable way. It is a decentralized peer-to-peer internet currency making mobile payment easy, very low transaction fees, protects your identity, and it works anywhere all the time with no central authority and banks. Bitcoin is designed to have only 21 million BTC ever created, thus making it a deflationary currency.



ábra 18 – Bitcoin kriptovaluta részletes nézete

Ezeknek a magyarázata a következő:

- **Market cap:** az adott kriptovaluta forgalomban lévő készletének a teljes piaci értéke.
  - **24 Hour Trading Vol:** a kriptovaluta kereskedési volumenének az értéke 24 órás távlatban.
  - **Fully Diluted Valuation:** a piaci kapitalizáció értéke, ha a kriptovaluta teljes kínálata van forgalomban. Ez az érték felmérése akár 3, 5, 10 évbe is beletelhet. [16]
  - **Circulation Supply:** piacon forgalomban lévő és a nyilvánosság által kereskedhető kriptovaluta mennyisége.

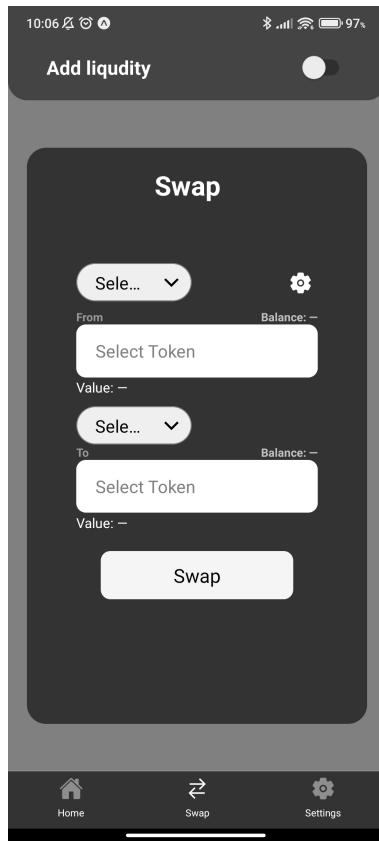
- **Total Supply:** az adott érme mennyisége, amely létre lett hozva, mínusz az elégetett érmék száma.
- **Max Supply:** a kriptovaluta maximális mennyisége, amely kibocsátásra került.

Legutolsó sorban pedig egy leírást találhatunk, amely az adott kriptovaluta történetéről, érdekességeiről szól.

## 6.2. Kereskedési oldal

### 6.2.1. Swap

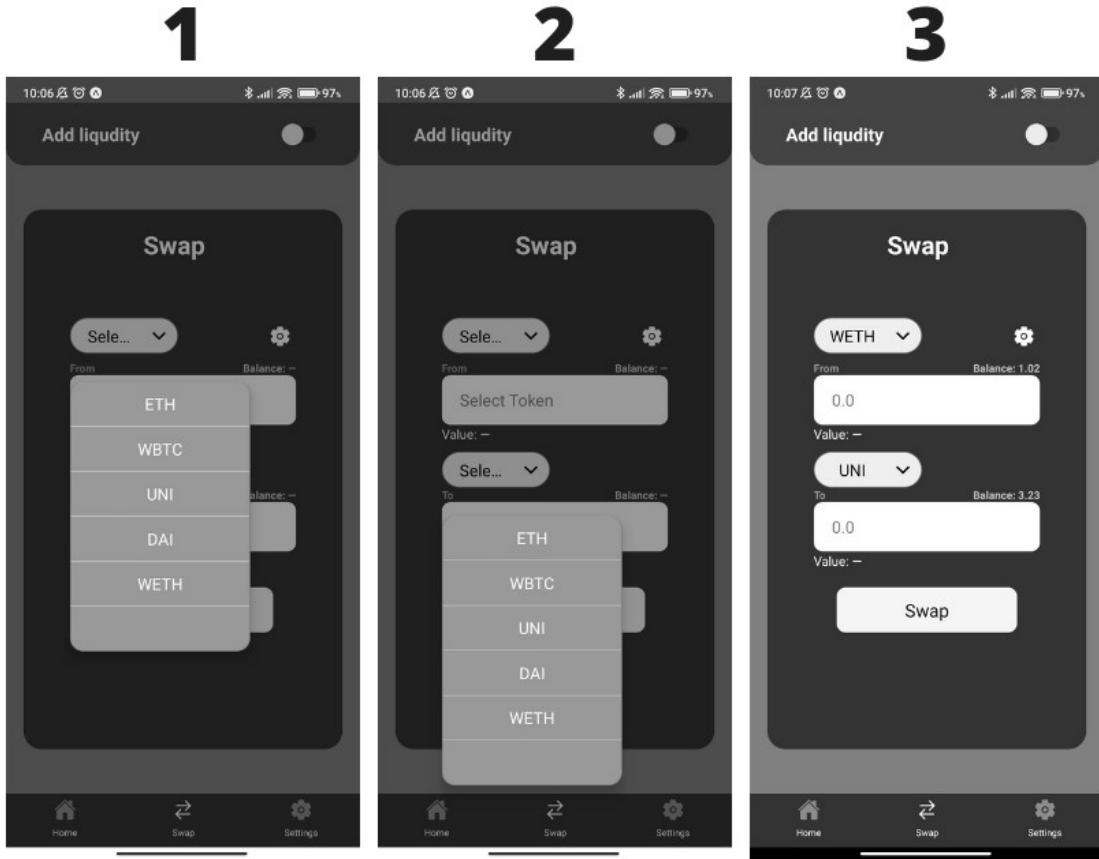
A kereskedési oldalon (ábra 19) lehetőségünk van swap-olni egy kriptovalutát egy másikra. Ennek feltétele, hogy rendelkezzünk azzal a kriptovalutával, amelyet átváltani szeretnénk, vagyis az egyenlegünk, amely az applikációban is megvan jelenítve nagyobb legyen, mint nulla. Itt láthatunk két lenyíló menüt, illetve két bemeneti mezőt, amelyben értékeket tudunk beírni. Emellett egy kapcsolót (switch) is láthatunk az oldal tetején, amellyel az add liquidity állapotot tudjuk beállítani. Mivel lehetőségünk van testre szabni a Swap tranzakciókat, ezért az első lenyíló menü mellett párhuzamosan találunk egy fogaskereket, amellyel ezt megtehetjük. Legutolsó sorban pedig egy Swap gombot látunk a második bemeneti mező alatt, amellyel a tranzakciónkat tudjuk elküldeni.



ábra 19 - Swap oldal

A következőkben röviden bemutatom az oldal működését, vagyis azt, hogyan lehet egy swap tranzakciót elejtől a végéig végrehajtani.

Elsősorban ahogy a képen is látható (ábra 20) ki kell választanunk az első lenyíló menüből (ábra 20, első kép) egy kriptovalutát, amellyel rendelkezünk, illetve amellyel kereskedni szeretnénk. A következő lépés, hogy a második lenyíló menüből (ábra 20, második kép) ugyancsak kiválasszunk egy kriptovalutát, pontosabban azt a kriptovalutát, amelyet kapni szeretnénk. Nem szükséges, hogy rendelkezzünk ezzel a kriptovalutával. Ha sikeresen kiválasztottuk mindkét valutát, akkor a 20. ábra harmadik képéén láthatjuk, hogy az első és a második bemeneti mező fölött jobb oldalt megjelenik külön-külön az egyenlegünk (Balance) kriptovaluta specifikusan.



ábra 20 - Swap oldal: tokenek kiválasztása

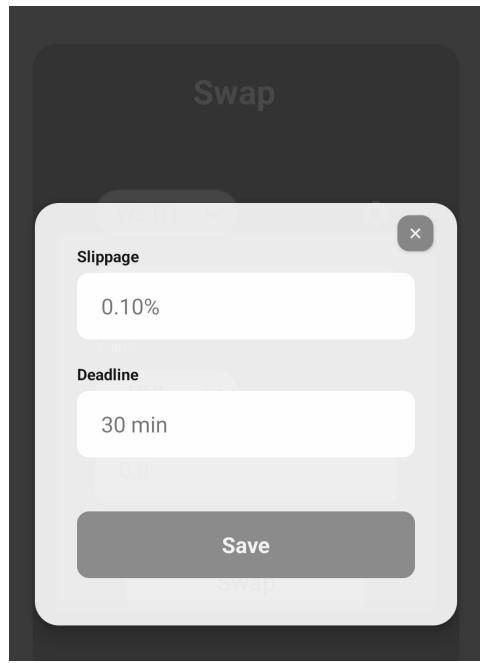
A következő lépésekben az első input mezőben szükséges beírnunk egy bizonyos értéket, amelyet átszeretnénk váltani az általunk kiválasztott kriptovalutára (ábra 21. első kép). Fontos megjegyezni, hogy az első és a második bemeneti mező alatt található egy „value” érték, amely jelzi a beírt kriptovaluta valódi értékét. Egy egyszerű példa erre: ha beírunk a mezőbe 1-et, akkor azt az értéket el kell osztani  $10^{18}$ -cal, amely majd reprezentálja a token valódi értékét. A hatvány érték token specifikusan változhat. Vagyis a végeredmény  $0.0000000000000000000001$  lesz. Ez csak a felhasználó számára van feltüntetve, hogy ne tévessze össze a valódi értéket a beírt értékkel. Erre azért volt szükség, mivel egy testnet-ről van szó, amelyen kevés ethereum-unk és kriptovalutánk volt és hogy ne egész tokenek-kel kereskedjünk, hanem a tört részükkel, így spórolva velük.



ábra 21- Swap oldal: értékek beírása és számolása

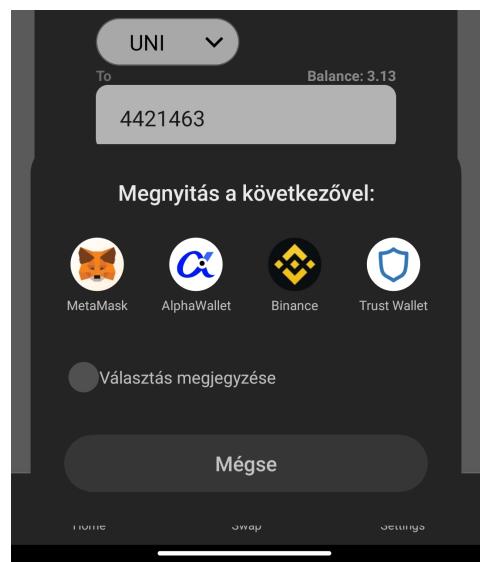
Ha beírtunk egy értéket az első mezőben, akkor a rendszer automatikusan kiszámolja a második mezőbe az értéket, amelyet kapnánk. Itt is a „value” érték automatikusan kiszámolásra kerül annak fényében, hogy mennyi a kiválasztott token decimal értéke, amely annak a bizonyos hatvány értéknek felelne meg. Ez látható az 21. ábra második képén.

Ahogy említve volt lehetőségünk van a tranzakciókat testre szabni, pontosabban betudjuk állítani a slippage és a deadline értékét. Ez látható a 22. ábrán.



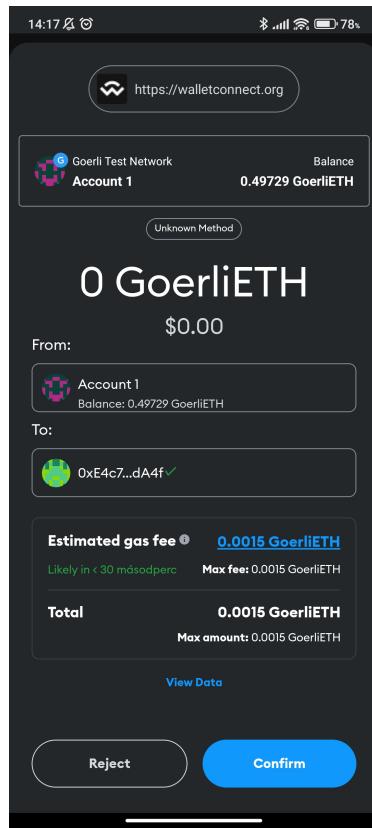
ábra 22 - Tranzakció beállítása

Miután beállítottuk ezeket a Swap gombra kell rányomnunk, amely a második input mező alatt található (ábra 21). Ezután az applikáció kérni fogja tölünk, hogy nyissunk meg egy wallet alkalmazást. Mivel valamilyen szinten a Metamask-ra épül az alkalmazás ezért ésszerű ezt használni (ábra 23)



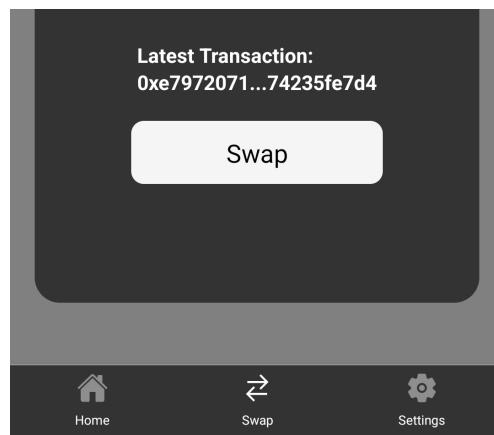
ábra 23 - Wallet kiválasztása

Miután kiválasztottuk a Metamask-ot, akkor az alkalmazás átirányít a Metamask applikációba, ahol alá kell írnunk, vagy más szóval el kell fogadnunk a tranzakciót (ábra 24)



ábra 24 - Metamask tranzakció elfogadása

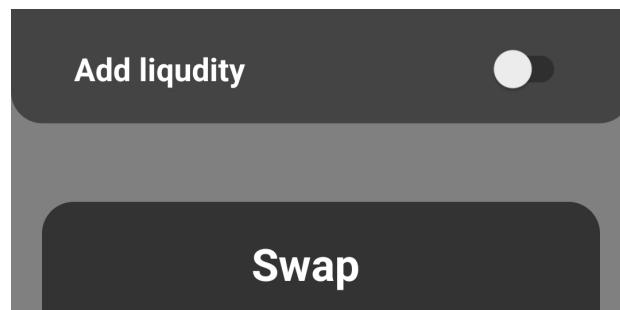
Ezután a Metamask alkalmazás átirányít a saját alkalmazásunkba. A Swap gomb fölött láthatóvá válik az elfogadott tranzakciónk transaction hash értéke, amelyre, ha rákattintunk akkor átdob az Etherscan oldalára, ahol nyomon tudjuk követni ezt.



ábra 25 - Transaction hash értéke

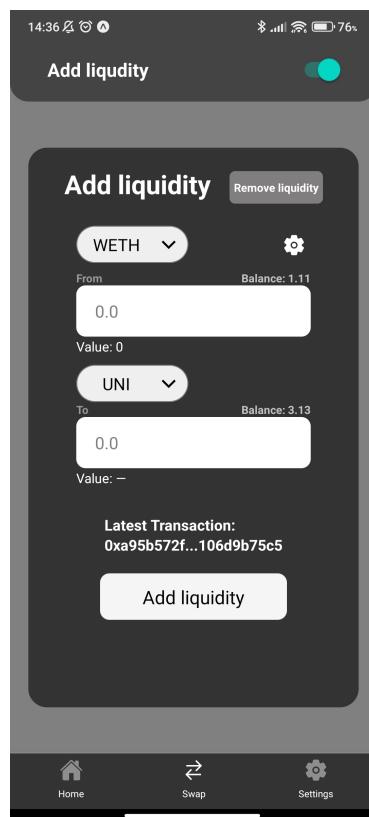
### 6.2.2. Add liquidity

Ahhoz, hogy likviditást adjunk hozzá egy bizonyos token párhoz, először az alkalmazásban át kell kapcsolunk egy kapcsolót, amely fent található (ábra 26).



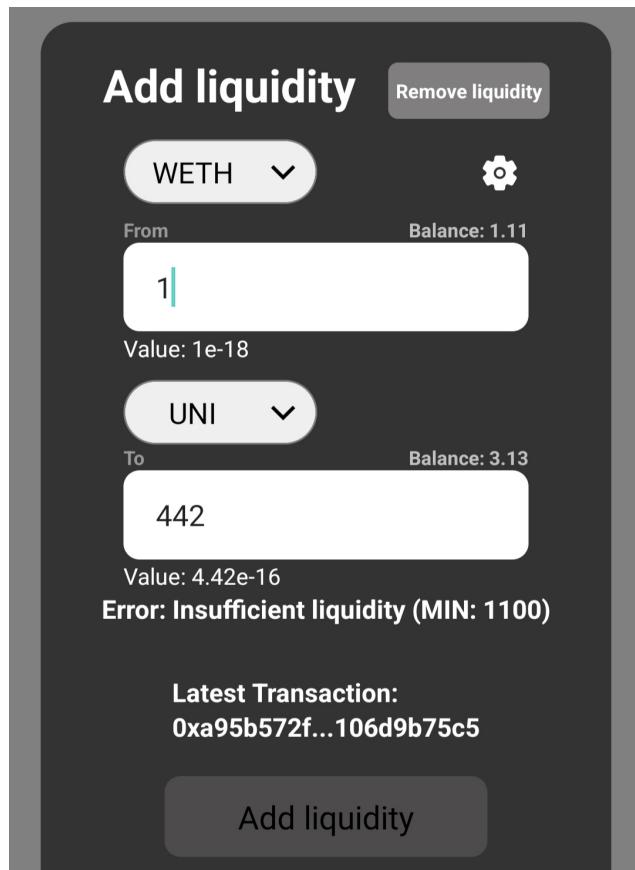
ábra 26 - *Add liquidity* kapcsoló

Ezután a Swap oldalhoz hasonló kinézet fog fogadni bennünket (ábra 27). A fő különbség a két oldal között, hogy megjelent egy Remove liquidity gomb, amellyel likviditást tudunk eltávolítani a saját token párunkból.



ábra 27- *Add liquidity* oldal

Az oldal működése nagyban hasonlít a Swap oldaléhoz, hiszen itt is ki kell választanunk két token-t a legördülő menüből, majd az egyik bemeneti mezőbe értéket kell beírnunk. Abban az esetben, ha egy bizonyos token pár pool-ja még nem létezik a rendszerben, akkor minden bemeneti mezőben értékeket kell beírunk. Erről bővebben szó volt a 3.1.3-mas fejezetben.



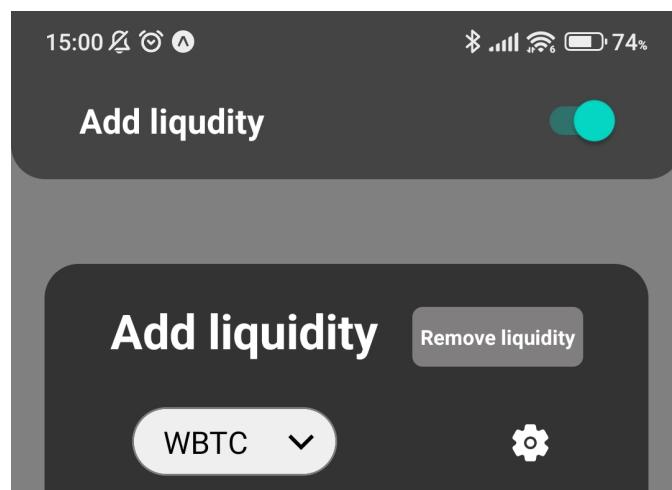
*ábra 28 - Add liquidity oldal: minimum likviditási érték*

Ha nem elegendő likviditást szeretnénk hozzá adni egy pool-hoz, akkor azt a rendszer jelezni fogja, ahogy az ábrán is látható (ábra 28). Ez abban mutatkozik meg, hogy egy hibaüzenetet találunk, ahol a rendszer jelzi, hogy a minimum likviditás legalább 1100 kell legyen, illetve inaktívvá válik az Add liquidity gomb. Fontos megjegyezni, hogy minden mező értéke nagyobb vagy egyenlő kell legyen, mint 1100. Ezért, ha a rendszer egy kisebb értéket számol ki, mint ez a szám, akkor változtatnunk kell a bemenetünkön, máskülönben nem lesz lehetőségünk likviditást hozzá adni. Ha sikeresen kiválasztottuk a számunkra és a rendszernek is megfelelő értékeket, akkor az Add liquidity gomb újra aktívvá válik, amelyre, ha rányomunk, akkor hasonlóan a Swap-hoz az alkalmazás megkér, hogy válasszunk egy wallet-ot. Ha ez

megtörtént, akkor a wallet alkalmazásban el kell fogadjuk a tranzakciót, majd ennek a tranzakciónak a hash értéke meg fog jelenni a saját alkalmazásunkban.

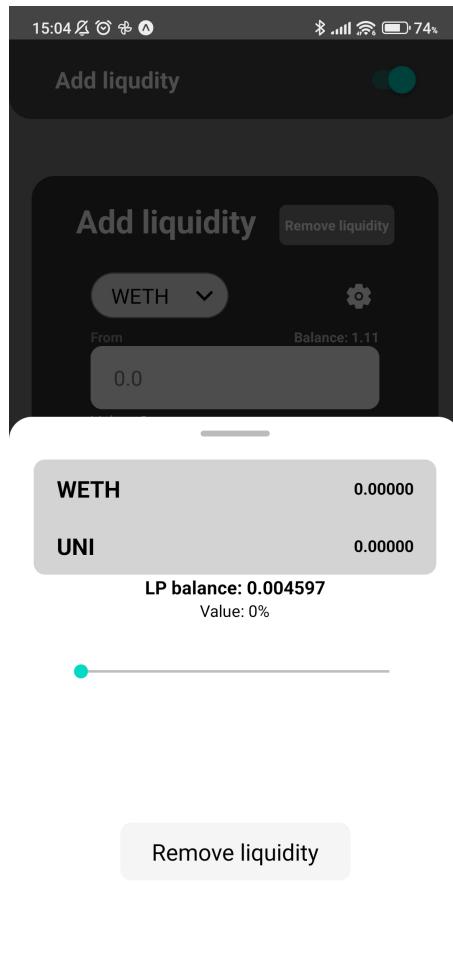
### 6.2.3. Remove liquidity

Ahhoz, hogy likviditást tudjunk eltávolítani elsősorban szükséges, hogy kiválasszunk két token-t, majd fontos, hogy az add liquidity kapcsoló be legyen kapcsolva, majd rá kell nyomnunk a Remove liquidity gombra (ábra 29).



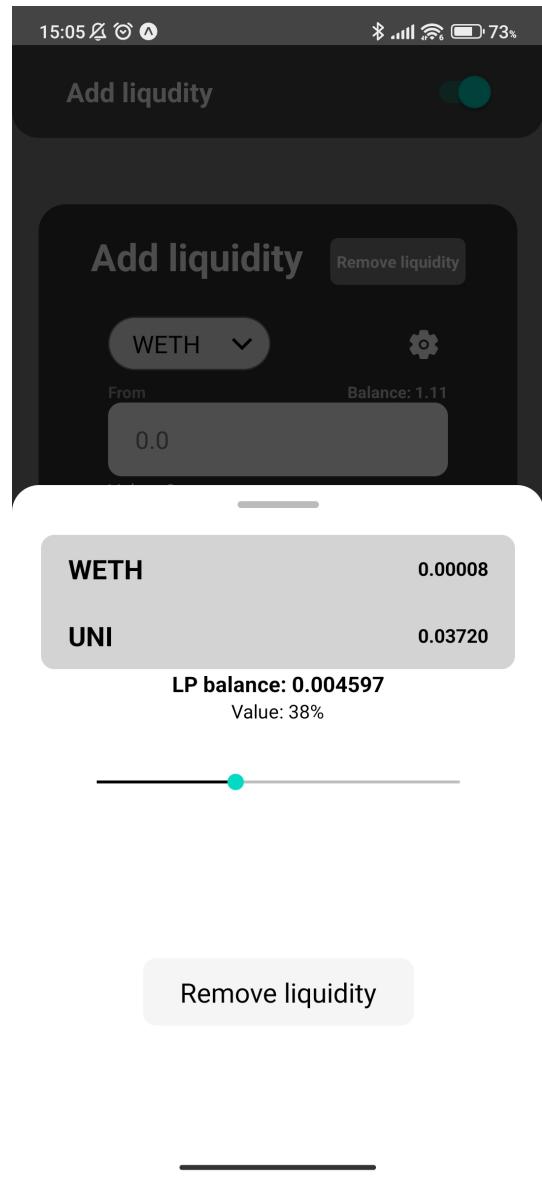
ábra 29 - Remove liquidity gomb

Fontos megjegyezni, hogyha egy olyan token párt választunk ki, amely még nem volt approve-olva, akkor az alkalmazás kérni fogja, hogy először approve-oljuk, annak érdekében, hogy a Router smart contract hozzáférjen a kriptovaluta párunkhöz. Ha nem adunk először approve-ot, vagyis nem engedélyezzük ezt a műveletet a smart contract-nak, akkor nem lesz lehetőségünk likviditást eltávolítani a pool-ból. Ezt a műveletet szükséges hasonlóan megtegyük swap esetén, olyankor amikor a kiválasztott tokenek-hez a router smart contract nem fér hozzá. Ezt hasonlóan a Remove liquidity-nél egy Approve gomb-al kell megtegyük.



*ábra 30 - Remove liquidity modal*

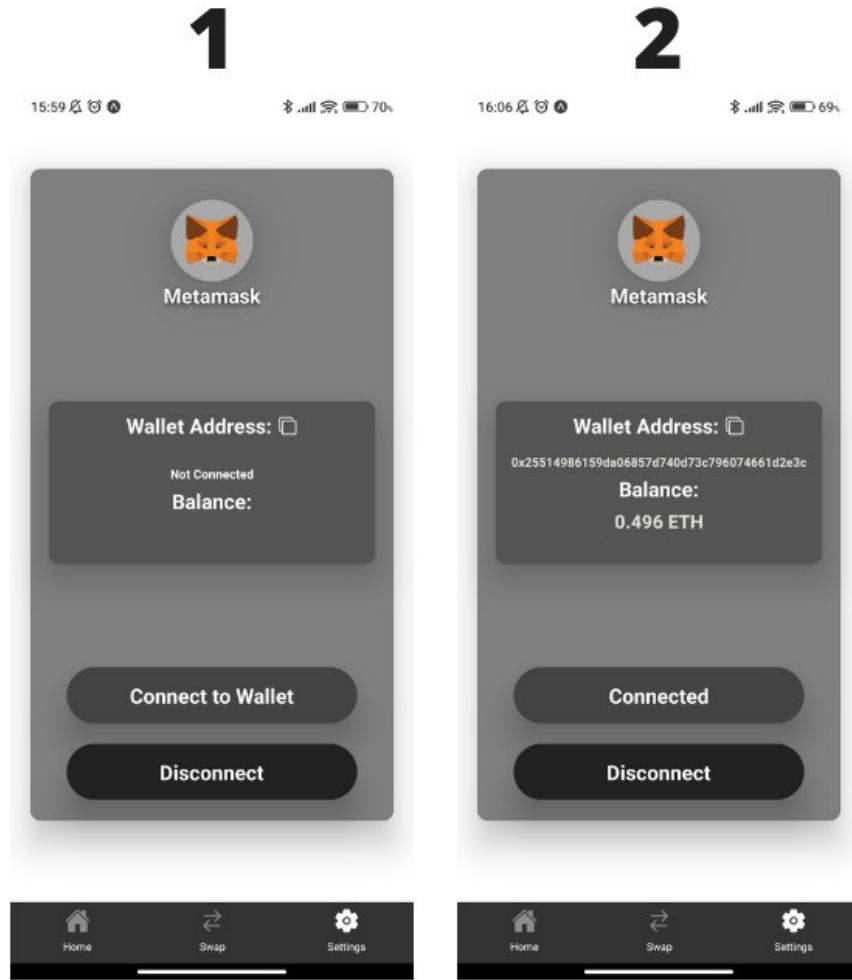
Ezen az oldalon láthatunk egy modal-t, amelyen a token párjainkat láthatjuk (ábra 30). A token párral párhuzamosan láthatjuk az egyenleg értéküket, amelyek a pool-ban vannak. Ez alatt láthatjuk az LP balance-t, amely képviseli a hozzájárult értékünket (nyereségünket) az adott pool-hoz. Ez pool-onként változik. Alatta láthatunk egy value értéket, amely a csúszka értékét jelzi. Ezzel a csúszkával (slider) tudjuk megmondani, hogy a pool-nak hány százalékát szeretnénk kivenni. Amiután állítottunk ezt az értéket, akkor a rendszer kiszámolja a két új egyenleget a két token párra. Ez látható a következő ábrán (ábra 31). Ha sikeresen beállítottuk a kívánt értéket, akkor a Remove liquidity gombra rákattintva megkér, hogy válasszunk egy wallet-ot, majd fogadjuk el a tranzakciót. Ha ez megtörtént, akkor a tranzakció hash értéke fog látszani az Add liquidity oldalon.



ábra 31- Remove liquidity token egyenleg számolás

### 6.3. Beállítások oldal

A beállítások oldal célja, hogy a saját pénztárcánkat tudjuk hozzácsatolni a rendszerhez (ábra 32, első kép). Ezen az oldalon láthatjuk majd a wallet címünket, amelyet kitudunk másolni, illetve az Ethereum egyenlegünket.



ábra 32 - Beállítások oldal

Ha rányomunk a Connect to Wallet gombra, akkor az alkalmazás megkér, hogy válasszunk egy wallet-et, az előző oldalakhoz hasonlóan. Ha csatlakoztatjuk a pénztárcánkat, akkor láthatjuk a Wallet Address címszó alatt (ábra 32, második kép) a saját address-ünket, illetve a Balance kulcsszó alatt az Ethereum egyenlegünket. A disconnect gombra kattintva a rendszer leválasztja a wallet-ünket, visszahozva az ábra 32., első képén látható állapotot.

## 7. Következtetések

A dolgozatom elkészítése során megismerkedtem a kriptovilág különböző részeivel.

Megismertem különböző technológiákat, amelyekre épül az egész kripto piac, illetve olyan platformokat, amelyek részét képezik az alkalmazásomnak, mint amilyen a Uniswap. Fontos volt ennek mélyre ható tanulmányozása és a Goerli Testnet kínálta lehetőséget használata, hiszen ennek az alkalmazásnak a tesztelése, kipróbálása nagyban hozzájárult, hogy egy hasonló alkalmazást hozzak létre. Emellett a Metamask és a WalletConnect rendszer dokumentációival is több időt kellett eltölteni ahhoz, hogy jól megismerjem a működésüket. Mivel egy cross-platform mobilalkalmazást fejlesztettem ezért magával a React Native nyelvvel is jobban megismerkedtem, amely szímpatikussá is vált számomra. Kihívást jelentett a nyelv bizonyos szintű egyedisége, ezért bizonyos részekre saját megoldást kellett kitalálni, mint amilyen a swap logika felépítése (például, ha a user egy input mezőbe ír, akkor egyidejűleg milyen események történjenek). Rengeteg kutatás és informálódás eredményeképpen sikerült jobban megértenem a kripto tőzsdék működését, vagyis a DEX-eket. A megértésük által a dolgozatom keretén belül sikerült lefejleszteni azokat a funkcionálitásokat, amelyek nélkül nem létezhetne egy decentralizált tőzsde. Összeségében egy működő decentralizált tőzsde mobil alkalmazást hoztam létre, amely működik úgy iOS-en, mint Android-on.

## 8. Összefoglalás

### 8.1. Megvalósítások

Az alkalmazást sikerült megvalósítani egy olyan környezetben, amely nem annyira elterjedt. Itt konkréten a mobil alkalmazásra gondolok, hiszen nem sok decentralizált tőzsde mobil alkalmazás van, amely cross platform-al is rendelkezik. Például, ha a Uniswap-ot vesszük alapul, akkor ez az alkalmazás csak web, illetve iOS platformokra érhető el. Általában ezek az alkalmazások inkább weboldalként érhetőek el. Éppen ezért sikerként könyvelhető el, hogy egy cross platformra készült mobil alkalmazást sikerült lefejleszteni, amely blokklánc technológiára épül. Nagy szerepet játszott ebben a React Native, illetve az Expo keretrendszer, hiszen ezek nélkül nem lehetett volna ezt megvalósítani. Az alkalmazás egy egyszerű, viszont könnyen használható felületet kínál, ahol a főbb kereskedési műveletek sikeresen eltudjuk végezni.

### 8.2. Főbb funkciók és összehasonlítás hasonló alkalmazásokkal

Az alkalmazás tudja azokat a funkciókat, amelyek szükségesek egy decentralizált tőzsde

működéséhez. Ha a Uniswap-ra gondolunk, akkor a mi alkalmazásunk is hasonlóan működik, ugyanazokat az elveket követi. Lehetőségünk van a főbb funkciók használatára, mint például a Swap, Add liquidity, Remove liquidity, illetve a wallet hozzácsatolás és leválasztás. Ezek a funkciók, amelyek egy decentralizált tőzsdtet azzá tesznek, amivé kell. Plusz funkcióként szerepel a főbb kriptovaluták kilistázása, illetve részletes nézete. Emellett lehetőségünk van testre szabni egy tranzakciót, illetve képesek vagyunk a saját pooljainkat megnézni.

Végső soron elmondhatjuk, hogy az alkalmazás tudja azokat a főbb funkciókat, mint a Uniswap, PancakeSwap vagy más decentralizált tőzsde.

### 8.3. Továbbfejlesztési lehetőségek

Ugyanúgy, ahogy minden alkalmazást, ezt is tovább lehet fejleszteni a különböző plusz funkciókkal vagy a UI szébbé tételevel. Ezek a következők lennének:

- Dockerizálni a rendszert és cloud-ban futtatni, hogy ne legyen szükséges a szerver lokális futtatására
- Reszponzívabbá tenni az alkalmazást
- Mivel hamarosan lejár a WalletConnect könyvtár támogatottsága (június 28), így keresni más megoldást ennek a helyettesítésére
- Flash Swap
- Pool részletesebb megtekintése funkció
- Analytic funkció, chart-ok bevitel, saját valuták részletesebb nézete
- Aggregátor implementálás
- Az adatbázis automatizálása (új pool-ok és token-ek automatikus mentése)
- Több blokklánc támogatása
- Saját token bevezetése, ami által lehet támogatni a projektet

## 9. Bibliográfia

- [1] Wikipedia - Satoshi Nakamoto. (2023, Április 20). Retrieved from Wikipedia: [https://en.wikipedia.org/wiki/Satoshi\\_Nakamoto](https://en.wikipedia.org/wiki/Satoshi_Nakamoto), [Hozzáférés dátuma: 2023, június 9.]
- [2] What is Ethereum?, W. I. (n.d.). *The foundation for our digital future*. Retrieved from Ethereum.org: <https://ethereum.org/en/what-is-ethereum/> [Hozzáférés dátuma: 2023, június 9.]
- [3] What is a CEX? Retrieved from Bitcoin.com: <https://www.bitcoin.com/get-started/what-is-a-cex/> [Hozzáférés dátuma: 2023, június 9.]
- [4] Centralized Exchange (CEX) Vs. Decentralized Exchange (DEX). (2022, Szeptember 8). Retrieved from Blockchain Council : <https://www.blockchain-council.org/cryptocurrency/centralized-exchange-vs-decentralized-exchange/> [Hozzáférés dátuma: 2023, június 9.]
- [5] Uniswap - How Uniswap works. Protocol Overview – Retreived from <https://docs.uniswap.org/contracts/v2/concepts/protocol-overview/how-uniswap-works> [Hozzáférés dátuma: 2023, június 11.]
- [6] Uniswap. (n.d.). Smart contracts. Retrieved from Uniswap V2 binary smart contract system:<https://docs.uniswap.org/contracts/v2/concepts/protocol-overview/smart-contracts> [Hozzáférés dátuma: 2023, június 11.]
- [7] Uniswap. (n.d.). Glossary . Retrieved from Uniswap Docs: <https://docs.uniswap.org/contracts/v2/concepts/protocol-overview/glossary> [Hozzáférés dátuma: 2023, június 11.]

- [8] Uniswap. (n.d.). *Swaps*. Retrieved from Uniswap Docs: <https://docs.uniswap.org/contracts/v2/concepts/core-concepts/swaps> [Hozzáférés dátuma: 2023, június 11.]
- [9] Uniswap. (n.d.). *Pools*. Retrieved from Uniswap Docs: <https://docs.uniswap.org/contracts/v2/concepts/core-concepts/pools> [Hozzáférés dátuma: 2023, június 11.]
- [10] Heller, M. *InfoWorld - Technology insight for the enterprise*. Retrieved from: <https://www.infoworld.com/article/3526447/typescript-vs-javascript-understand-the-differences.html> [Hozzáférés dátuma: 2023, Június 1.]
- [11] Bećirović, N. (2023, Június 2). *React Native — Platform Specific Code*. Retrieved from Medium.com: <https://medium.com/maestral-solutions/react-native-platform-specific-code-e217db5778f> [Hozzáférés dátuma: 2023, Június 1.]
- [12] Bulat, R. (2020. Augusztus 11). *Uniswap V2: Everything New with the Decentralised Exchange*. Forrás: Medium: <https://rossbulat.medium.com/uniswap-v2-everything-new-with-the-decentralised-exchange-52b4bb2093ab> [Hozzáférés dátuma: 2023, Június 15.]
- [13] *Uniswap/v2-core is licensed under the GNU General Public License v3.0*. Retrieved from <https://github.com/Uniswap/v2-core/blob/master/LICENSE> [Hozzáférés dátuma: 2023, Június 15.]
- [14] Csaba, K. *Github - Decentralized Exchange*. Forrás: <https://github.com/kormocicsaba/decentralized-exchange/tree/mobile-app> [Hozzáférés dátuma: 2023, június 16.]

- [15] Csaba, K. *Google Drive.* Forrás: MongoDB connection string:  
<https://drive.google.com/file/d/1HJieSXW32JHtaXU6V-dKEsFjlFE7KPVC/view?usp=sharing> [Hozzáférés dátuma: 2023, június 17.]
- [16] Agbo, J. (2022, November 25). *What Is Fully Diluted Valuation (FDV) In Crypto?* Retrieved from CoinGecko: <https://www.coingecko.com/learn/what-is-fully-diluted-valuation-fdv-in-crypto> [Hozzáférés dátuma: 2023, június 17.]