Szakdolgozat

Miskolci Egyetem

Web3 webalkalmazás fejlesztése

Készítette:

Barta Balázs

Programtervező informatikus

Témavezető:

Dr. Hornyák Olivér

Miskolc, 2024

Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematikai Intézeti Tanszék **Szám:**

Szakdolgozat Feladat

Barta Balázs (S90NXK) programtervező informatikus jelölt részére.

A szakdolgozat tárgyköre: Webfejlesztés

A szakdolgozat címe: Web3 webalkalmazás fejlesztése

A feladat részletezése:

1. Fejlesszen webalkalmazást, amely a kriptovaluta menedzsment funkciókat biztosít!
2. Mutassa be az alkalmazás elméleti hátterét, a használt technológiákat, a fejlesztési módszertant, beleértve a választott programnyelveket és a fejlesztői környezetet!
3. Implementálja a kriptopénz tárcával való autentikációt és jogosultságkezelést, a blokklánc alapú pénztárca portfólió lekérdezését és a biztonságos tranzakciók végrehajtását!
4. Mutassa be az alkalmazás működését és a felhasználói felületet!

Témavezető: Dr. Hornyák Olivér

A feladat kiadásának ideje: 2024. 09. 11.

. . . . . .
szakfelelős

Eredetiségi Nyilatkozat

Alulírott **Barta Balázs**; Neptun-kód: `S9ONXK` a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős Programtervező informatikus szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy *Web3 webalkalmazás fejlesztése* című szakdolgozatom saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, ...év ..... .hó ..... .nap

.....

Hallgató

1.

szükséges (módosítás külön lapon)

A szakdolgozat feladat módosítása

nem szükséges

....

dátum témavezető(k)

2. A feladat kidolgozását ellenőriztem:

témavezető (dátum, aláírás): konzulens (dátum, aláírás):

....

....

....

3. A szakdolgozat beadható:

....

dátum témavezető(k)

4. A szakdolgozat ..... szövegoldalt

..... program protokollt (listát, felhasználói leírást)

..... elektronikus adathordozót (részletezve)

.....

..... egyéb mellékletet (részletezve)

.....

tartalmaz.

....

dátum témavezető(k)

5.

bocsátható

A szakdolgozat bírálatra

nem bocsátható

A bíráló neve: .....

....

dátum szakfelelős

6. A szakdolgozat osztályzata

a témavezető javaslata: .....

a bíráló javaslata: .....

a szakdolgozat végleges eredménye: .....

Miskolc, ...

a Záróvizsga Bizottság Elnöke

Tartalomjegyzék

1. fejezet

Bevezetés

A Web3.0 technológiák megjelenésével, az internetes világ fejlődése egy új irányba terelődött a korábbi időszakokhoz képest. Előtérbe került az online szolgáltatások decentralizálása és az adatok fokozott biztonságának garantálása. Az újabb alkalmazások lehetővé teszik, hogy a felhasználók teljes ellenőrzést gyakoroljanak a digitális vagyonuk és az egyéb adataik felett, figyelmen kívül hagyva a közvetítőket így létre hozva egy decentralizált, felhasználó-központú digitális világot. Ezen rendszerek terjedésével egyidejűleg egyre inkább felértékelődtek a kripto valuták és a blokklánc alapú eszközök.

A dolgozatom célja, hogy egy olyan modern webalkalmazást fejlesszek, amely kizárólag Web3.0 technológiákat alkalmaz. A kidolgozandó alkalmazás lehetőséget biztosít kriptovaluta pénztárcával való autentikációra, jogosultság kezelésre és blokklánc alapú pénztárca portfólió lekérésére, és egy biztonságos tranzakcióra. Az alkalmazásom nem csak a decentralizált gazdasági tevékenységek támogatását célozza, emellett még felhasználó barát és skálázható megoldást is nyújt.

A dolgozat további részében a következő fejezetek kerülnek bemutatásra:

- **Irodalmi áttekintés** Az alkalmazásom elméleti hátterének ismertetése, amely tartalmazza a Web3.0 technológiák bemutatását, a blokklánc alapú autentikációt és a decentralizált rendszerek működését.
- **A fejlesztési bemutatása:** A választott programnyelvek és a fejlesztői környezet bemutatása. A tervezés lépéseinek részletezése.
- **Az alkalmazás működésének bemutatása:** Az egyes funkciók részletes bemutatása legfontosabb kód részletekkel, ideértve a kriptopénztárca alapú autentikációt és a blokklánc alapú portfólió lekérdezést is.
- **A felhasználói felület bemutatása:** Minden funkció részletes bemutatása a felhasználó felé, illetve az oldal funkcionalitásának és teljesítményének elemzése.

2. fejezet

Irodalmi áttekintés

2.1. Web3.0 Technológiák bemutatása

A Web3.0 egy olyan fogalom, amelyet az internet fejlődésének következő szakaszának leírására használnak. A fogalmat 2014-ben vezette be Gavin Wood, az Ethereum nevű, blokklánc-alapú szoftverplatform egyik társalapítója. A Web3.0-at egy decentralizált, jövőbeni internetverzióként írta le, amely csökkenti a jelenlegi Web2.0-t uraló vállalatok, mint a Meta, az Amazon és a Google befolyását. Ebben az internetverzióban a felhasználók nagyobb kontrollt gyakorolhatnak adataik felett, ahelyett, hogy azok olyan vállalatokkal lennének megosztva, amelyek személyes információkat hasznosítanak ki. Az eredmény egy olyan interaktív internetes élmény, amelyben a személyes adatok tulajdonlása és a magánélet védelme erőteljesebben érvényesül. [11]

2.1.1. A web eddigi története

Ahhoz, hogy megértsük, hogyan működik a Web3.0, és miért fogja megváltoztatni azt, ahogyan az emberek az interneten interakcióba lépnek, hasznos megérteni a web történelmét. A webet két különböző időszakra oszthatjuk, Web1.0-ra és Web2.0-ra. [11]

2.1.2. Web1.0

A Web 1.0 az 1990-es évek elejére nyúlik vissza. Ez a verzió statikus weboldalakat tartalmazott, amelyeket HTML-ben kódoltak. Egy nyílt és decentralizált helyet hozott létre, ahol a felhasználók böngészők, például Netscape és Internet Explorer segítségével kereshettek és találhattak információkat. Az internet ezen verziója hozta létre az olyan weboldalakat, mint a WebMD, ahol orvosi állapotokat és tüneteket lehetett keresni, valamint a GeoCities, ahol a felhasználók saját weboldalakat hozhattak létre online közösségekben. [11]

2.1.3. Web2.0

Az évtized végére az emberek már nemcsak egy online enciklopédiaként tekintettek az internetre. A blogok, fórumok és wikipédiák elterjedése lehetővé tette az emberek számára, hogy megosszák tevékenységeiket és összekapcsolódjanak egymással. Ezzel kezdődött a Web2.0, amely az 2000-es évek elejére és közepére vált ismertté. A technológiai cégek kihasználták a közösségi média erejét, hogy olyan infrastruktúrát fejlesszenek ki,

amely lehetővé tette, hogy néhány cég nagy hatalomra tegyen szert. Ezek közé a cégek közé tartozik a Meta, a Twitter, a Google és az Amazon. [11]

2.1.4. Web3.0 működése

A Web3.0 a Web1.0 elosztott természetét ötvözi a Web2.0 interaktív jellegével egy könnyen használható környezetben. Ideálisan lehetővé teszi, hogy az egyes felhasználók nagyobb kontrollt gyakoroljanak online élményeik felett, és növeli a biztonságot a blokklánc technológia révén (a decentralizált tranzakciós napló, amely a felhasználók számára látható blokkokban tárolja az adatokat). A Web2.0 arra kényszerít, hogy a nagy technológiai cégek technológiájára és biztonságára támaszkodjunk. A Web3.0 a kezedbe adja az irányítást – és minden más felhasználó kezébe. A Web3.0 fejlesztésében részt vevő felhasználók tokeneket kaphatnak a részvételükért. [11]

2.1.5. A centralizált hálózat

A centralizált hálózatok olyan rendszerek, amelyekben egyetlen csomópont, amelyet gyakran központi szerverként vagy központi csomópontként emlegetnek, irányítja az egész hálózatot. A hálózat többi csomópontja mind ehhez a központi csomóponthoz csatlakozik. [7]

2.1.6. A centralizált hálózatok működése

A centralizált hálózatok egyetlen központi hatóság vagy szerver irányítása alatt működnek. Ez a szerver jogosult a hálózaton belüli adatok kezelése és elosztása felett. Ez a központi egység hozza meg az összes döntést arról, hogy hogyan történik az adatok továbbítása, tárolása és feldolgozása. Ebben a típusú hálózatban minden felhasználói közlésnek át kell haladnia a központi szerveren, amely képes nyomon követni és irányítani az összes tevékenységet.

Ez a központi hatóság felelős a hálózat biztonságáért és karbantartásáért. A centralizált struktúra lehetővé teszi a hatékony irányítást és a gördülékeny kommunikációt. Ugyanakkor potenciális kockázatokat is felvet, mint például az egyetlen hibahely és a kiberbiztonsági támadásokkal szembeni sebezhetőség. Azonban megfelelő biztonsági intézkedésekkel mérsékelhetők ezek a kockázatok. [7]

2.1.7. A decentralizált hálózat

Ellentétben a centralizált hálózatokkal, az decentralizált hálózatok a kontrollt több csomópont között osztják meg. Ebben az elrendezésben minden egyes csomópont függetlenül működik a többitől, és a hálózat továbbra is működik, még akkor is, ha az egyik csomópont meghibásodik. [7]

2.1.8. A decentralizált hálózatok működése

A decentralizált hálózatok egyetlen irányító hatóság nélkül működnek. Ehelyett az összes csomópont vagy előfizető egyenlő hatalommal és kontrollal rendelkezik az adatok továbbításának irányításában. Ezen modell alkalmazásával ezek a hálózatok magas szintű biztonságot és adatvédelmet biztosítanak. Egy elosztott hálózatban az adatokat több csomóponton tárolják, így gyakorlatilag lehetetlen, hogy egyetlen entitás irányítsa

vagy manipulálja az adatokat. Ez az elosztott jelleg hozzájárul a hálózat robusztusságához is, mivel egy csomópont meghibásodása nem befolyásolja a rendszer teljes működését. Ezen kívül a decentralizált hálózatok peer-to-peer (olyan hálózati architektúra, ahol a résztvevő csomópontok közvetlenül kommunikálnak egymással, anélkül, hogy szükség lenne központi szerverre vagy közvetítő eszközre.) struktúrája demokratikusabb és átláthatóbb digitális környezetet teremt, ahol a felhasználók teljes mértékben irányítják saját adataikat. [7]

2.1.9. Különbségek a centralizált és decentralizált hálózatok között

Aspektus	**Centralizált hálózatok**	**Decentralizált hálózatok**
Irányítás eloszlása	Egyetlen csomópont irányít	Irányítás elosztott több csomópont között
Hiba hatása	Egyetlen hiba pontja a központi csomópont	Ellenálló a csomópont meghibásodásokkal szemben
Skálázhatóság	Skálázhatósági problémák lehetnek a központi csomópont terheltsége miatt	Jobban skálázható, mivel a terhelés eloszlik
Biztonsági szempontok	Sérülékeny a biztonsági támadásokkal szemben, mivel minden adat egy helyen tárolódik	Az adatok elosztása csökkenti az adatlopás kockázatát

2.1.10. A centralizált és decentralizált hálózatok előnyei

Centralizált hálózatok:

- Hatékonyság: A centralizált hálózatokat a központi csomópont határozza meg, ami magas hatékonyságot eredményez.
- Kezelés egyszerűsége: Az irányítás és a karbantartás központosított, így egyszerűsíti a kontrollt és a karbantartást.
- Költséghatékonyság: A centralizált hálózatok gyakran kevesebb hardvert igényelnek, így olcsóbbak.

Decentralizált hálózatok:

- Ellenálló képesség: A decentralizált hálózatok képesek ellenállni a gyakori hálózati problémáknak és más meghibásodásoknak, mivel nincs egyetlen hiba pontjuk.
- Fokozott adatvédelem és biztonság: Az adatok elosztása csökkenti az adatvédelmi és biztonsági kockázatokat.
- Jobb skálázhatóság: A terhelés elosztása lehetővé teszi a decentralizált hálózatok számára, hogy jobban skálázhatók legyenek. [7]

2.2. A blokklánc technológia

A blokklánc technológia egy adatbázis, amely rekordokat tárol egy elosztott és decentralizált főkönyvben. Átlátható és megbízható, mivel miután az adatokat rögzítik és tárolják a blokklánc hálózaton, általában lehetetlen azokat megváltoztatni vagy módosítani. A tranzakciók duplikálódnak és eloszlanak a blokklánc hálózaton minden felhasználó számára, aki csatlakozik a blokklánchoz. A tranzakció eredeti tulajdonosa egy digitális aláírást ad hozzá az úgynevezett főkönyvhöz. Ez hitelesíti az információt és védi a manipulációval szemben. [9]

2.2.1. A blokklánc technológia fontossága

A blokklánc alapvető fontosságú, mivel a vállalkozások az információkra építenek, amelyeket több forrásból kapnak. Ahhoz, hogy a legjobban tájékozott döntéseket hozhassák, a vállalkozásoknak gyorsan, biztonságosan és pontosan kell megkapniuk az információkat. A blokklánc biztosítja az információk azonnali átvitelét, amelyet a vállalkozások keresnek, egy megosztott és módosíthatatlan főkönyv segítségével. Az adatokat a blokklánc-on pontosan abban a sorrendben helyezik el, ahogyan megkapták, biztosítva ezzel azok biztonságát. Bármilyen változás egy blokkban nem maradna észrevétlen, mivel ez egy sor változást indítana el az összes hozzá kapcsolódó blokkban. [9]
A blokklánc főkönyv teljes átláthatóságot és bizalmat biztosít, mivel minden felhasználónak ugyanazt az információs láncot mutatja végig. A vállalkozások új hatékonyságokat érhetnek el működésükben, amelyek potenciálisan új lehetőségeket teremthetnek a sikerhez. A blokkláncot sokféle célra használhatjuk, többek között fizetésre, gyártásra, rendelések nyomon követésére és még sok másra.

2.2.2. A blokklánc technológia működése

A blokklánc technológia három alapvető ötletre vagy fogalomra épül: blokkok, bányászok és csomópontok.

Blokkok: A blokk tartalmazza egy tranzakció adatait, amelyek megmutatják egy eszköz útvonalát vagy mozgását. A körülményektől függően ezek a blokkok tartalmazhatják, hogy ki hajtja végre a tranzakciót, mi a tranzakció tartalma és mikor történt.

Bányászok: A blokklánc hálózat bányászai bányászat segítségével új blokkokat hoznak létre a lánc számára. Ezt úgy teszik, hogy fejlett szoftverek segítségével megoldásokat keresnek bonyolult matematikai problémákra, amelyek egy hasht generálnak. A hash egy adott bithosszúságú szám (a jelenlegi implementációkban ez 256 bites), amely tartósan kapcsolódik egy nonce-hoz, egy 32 bites egész számhoz, amelyet minden blokkhoz hozzárendelnek. Az angol nyelvben a „nonce” kifejezés eredetileg azt jelentette, hogy „valami, ami csak egy adott alkalomra van”. A kriptográfia és számítástechnika világában ezt a jelentést használják ki, mivel a nonce egy egyszeri felhasználású értéket jelent, amelyet egy adott művelet vagy tranzakció biztonsága érdekében alkalmaznak. Miután a bányászok sikeresen megtalálták az elfogadott nonce-ot és hasht, a blokk hozzáadódik a nagyobb blokklánc hálózathoz.

Csomópontok: A blokklánc technológia csomópontja bármilyen elektronikus eszközként megjelenhet, amely képes fenntartani a hálózat működését és megőrizni a láncot

úgy, hogy másolja azt. Minden csomópontnak el kell fogadnia és jóvá kell hagynia az új blokkokat ahhoz, hogy az egész blokklánc frissüljön. [9]

2.2.3. A blokklánc előnye a személyazonosság-kezelésben

A blokklánc első és legfontosabb előnye a személyazonosság-kezelésben, hogy megszabadít a terhes papíralapú azonosítási folyamatoktól. A blokklánc alapú személyazonosság-kezelési megoldások képesek egy másolatot létrehozni a személyazonosság igazolásáról, ami segíthet azoknak a felhasználóknak, akik elvesztették eredeti dokumentumaikat. Azok a felhasználók, akik elvesztik vagy elkeverik az eredeti személyazonosító igazolásaikat, visszakaphatják azokat, bár bizonyos folyamatokon kell keresztülmenniük a blokklánc alapú személyazonosság-kezelés révén. A blokklánc alapú személyazonosság-kezelés segít a szervezeteknek biztosítani, hogy a nyilvántartások állandóak és manipulálhatatlanok maradjanak. A kormányzati szervek a személyazonosító dokumentumokat a blokkláncon tárolhatják, ezzel biztosítva a biztonságot és a megbízhatóságot. Ezenkívül a blokklánc biztonsága garantálja, hogy a személyazonosító igazolás nyilvántartása állandó. [8]

2.2.4. Az azonosítási folyamat egyszerűsítése

A személyazonosság-kezelési megoldások gyakran bonyolultak a különálló rendszerek és a meglévő manuális folyamatok miatt. Ezeknek a korlátoknak a leküzdésére a blokklánc technológia hatékonyabbá teszi a személyazonosság-kezelési megoldásokat. A blokklánc alapú személyazonosság-kezelés egyértelmű és látható előnyei megmutatkoznak olyan megoldásokban, mint a Blockpass. Ez egy blokklánc alapú KYC, azaz "Ismerd meg ügyfeled" portál, amely azonosítási ellenőrzési portálként szolgál. [8]

2.2.5. A kriptopénz tárca

A kriptopénz tárcák arra szolgálnak, hogy tárolják a privát kulcsodat, biztosítva, hogy a kriptopénzed mindig elérhető legyen. Emellett lehetővé teszik a kriptopénzek, mint például a Bitcoin és az Ethereum küldését, fogadását és felhasználását. [4]

2.2.6. MetaMask

A MetaMask lehetővé teszi a felhasználók számára, hogy közvetlenül kezeljék kriptovalutáikat és csatlakozzanak a decentralizált alkalmazásokhoz. A pénztárca titkosított kulcsokat használ a tranzakciók biztonságos feldolgozására, biztosítva a felhasználók eszközeinek védelmét. Emellett a pénztárca könnyen integrálható más blokkláncokkal és kriptovaluta platformokkal, így sokféle felhasználási esethez alkalmazható eszközzé válik. Támogatja a hardveres pénztárcákat is, amelyek további biztonsági réteget biztosítanak azoknak a felhasználóknak, akik offline szeretnék tárolni a pénzüket. A MetaMask emellett felhasználóbarát felülettel rendelkezik, így azok számára is elérhető, akik újonnan ismerkednek a kriptovalutákkal. A folyamatos frissítések és a közösség áramvonalas fejlesztése révén a MetaMask továbbra is fejleszti funkcióit és javítja a felhasználói élményt.

2.2.7. A pénztárca-alapú hitelesítés

A Web3.0 hitelesítés, más néven pénztárca alapú hitelesítés, egyre inkább elterjedő megoldásaként jelenik meg arra, hogyan autentikáljuk magunkat weboldalakra és alkalmazásokra. Lényegében lehetővé teszem, hogy rákattintsak egy "bejelentkezés a pénztárcámmal" gombra, amely egy párbeszédablakot indít el, és ezáltal jóváhagyhatom a bejelentkezést a digitális pénztárcámban. Mivel nyilvános/privát kulcs rendszert használok, biztonságosabbnak tekinthetem ezt, mint a hagyományos jelszó alapú módszereket, és kevésbé vagyok kitéve phishing támadásoknak. A pénztárca alapú hitelesítés igazi varázsa, hogy nemcsak hitelesítésre használhatom. A pénztárcámmal nemcsak egy eszközt kapok a bejelentkezéshez, hanem egy eszközt is a fizetéshez és az adatok tárolásához a blokkláncon. A pénztárca alapú hitelesítés lehetőséget ad számomra a „mély" hitelesítésre és új alkalmazásokra a weboldalon kívül. A pénztárca alapú hitelesítés egy kicsit olyan, mint a Google vagy Facebook fiókba való bejelentkezés. A folyamat azzal kezdődik, hogy te, mint végfelhasználó, rákattintasz a „bejelentkezés a pénztárcáddal" gombra. Miután ezt megtetted (és attól függően, hogy asztali gépen vagy mobilon vagy), egy felugró ablak jelenik meg, amely arra kér, hogy „írj alá egy üzenetet". Ez az üzenet érvényesíti, hogy te vagy a pénztárcád tulajdonosa, és megerősíti a weboldal számára, hogy te vagy az, aki azt állítod magadról. [12]

2.3. Decentralizált Pénzügyi Szolgáltatások

2.3.1. A DeFi, és működése

A Decentralizált Pénzügyi Szolgáltatások, vagyis DeFi, nyílt és decentralizált pénzügyi szolgáltatásokat kínál a blokklánc-technológia segítségével. A DeFi nyilvános blokkláncházatokat használ működéséhez, ahol az Ethereum a legjelentősebb platform. Ez a struktúra eltér a jelenlegi pénzügyi rendszerektől, amelyek erősen függenek a közvetítőktől, például a bankoktól. A DeFi központjában az okosszerződések állnak, amelyek az automatizálás és a biztonság érdekében működnek, hogy biztosítsák a zökkenőmentes pénzügyi tranzakciókat, valamint a résztvevők által meghatározott szabályok és feltételek betartását. Az okosszerződések a blokklánc kódjában találhatók, és önmagukat végrehajtó megállapodásokként működnek. Az okosszerződések használata a DeFi-ben megszünteti a közvetítő hagyományos szerepét a pénzügyi rendszerben. Ebben az új rendszerben a résztvevők közvetlenül egymással bonyolítanak le pénzügyi tevékenységeket. A peer-to-peer interaktivitás lehetővé teszi, hogy kölcsönözzünk, hitelezünk, kereskedjünk, derivatívákat vásároljunk és még sok egyéb tevékenységet végezzünk. [10]

2.3.2. A DeFi felhasználói és célja

Bárki, aki a hagyományos bankrendszeren kívül szeretne kölcsönözni, hitelezni, befektetni vagy valutákkal kereskedni, használhatja a DeFi-t. A DeFi célja, hogy a blokklánc átláthatóságát, megváltoztathatatlanságát és biztonságát kihasználva egy befogadóbb, hozzáférhetőbb és hatékonyabb pénzügyi ökoszisztémát hozzon létre. Lehetővé teszi, hogy azok az emberek is kontrollt szerezzenek az eszközeik felett és hozzáférjenek alapvető pénzügyi szolgáltatásokhoz, akik számára nehézséget jelenthet a jelenlegi pénzügyi rendszerben való részvétel. [10]

2.3.3. A DeFi előnyei

A DeFi számos előnyt kínál a hagyományos pénzügyi rendszerekkel szemben. Néhány kulcsfontosságú előny a következő:

- Hozzáférhetőség: Bárhol is legyél a világon, amennyiben van internetkapcsolatod, a DeFi lehetővé teszi, hogy hozzáférj alapvető pénzügyi szolgáltatásokhoz. Ezáltal több millió ember, aki jelenleg nem tud banki szolgáltatásokat igénybe venni, élvezheti azok előnyeit.
- Átláthatóság: A nyilvános blokkláncok rögzítik és tárolják az összes DeFi-n keresztül végbemenő tranzakciót. Ez azt jelenti, hogy bárki hozzáférhet és megtekintheti a tranzakciókat, ami teljes átláthatóságot biztosít.
- Biztonság: A DeFi protokollok a blokklánc kriptográfiai jellemzőire támaszkodnak, ami ellenállóbbá teszi őket a csalásokkal, a cenzúrával és a hackelési kísérletekkel szemben. A felhasználók általában maguk birtokolják privát kulcsaikat, így csökken annak kockázata, hogy a központi letétkezelők kezeljék az eszközeiket.
- Interoperabilitás: A DeFi projektek gyakran törekednek az interoperabilitásra, amely lehetővé teszi a zökkenőmentes eszközátvitelt és kölcsönhatást különböző platformok között. Ez az interoperabilitás ösztönzi az innovációt és bővíti az elérhető pénzügyi szolgáltatások körét.
- Hozamlehetőségek: A DeFi új módokat kínál arra, hogy passzív jövedelemforrást hozz létre. Ezt különféle hitelnyújtási lehetőségeken és stakingen keresztül érheted el. További bevételi forrást biztosíthat a likviditási poolokban való részvétel és a yield farming. Ezek a lehetőségek vonzó hozamot kínálnak a hagyományos megtakarítási vagy befektetési lehetőségekhez képest, azonban a piac volatilis, és a hozamok nem garantáltak. [10]

3. fejezet

A fejlesztés bemutatása

3.1. A választott programnyelvek bemutatása

Webalkalmazás fejlesztésénél elengedhetetlen a HTML5 jelölőnyelv és a CSS3 stíluslapnyelv ismerete így én is ezeknek a nyelveknek a megismerésével kezdtem a felkészülést. A továbbiakban a JavaScript programozási nyelv megismerésével foglalkoztam, majd a webalkalmazásom megírására a React mellett döntöttem. A React egy nyílt forráskódú frontend JavaScript-könyvtár. Ezen programozási nyelveket használtam az alkalmazás elkészültéhez, ezeket fogom most röviden bemutatni.

3.1.1. HTML

A HTML (HyperText Markup Language) egy leíró nyelv, amelyet kifejezetten a weboldalakhoz terveztek és szabványosítottak. Fontos megjegyezni, hogy ez egy leíró nyelv, nem pedig programozási nyelv. Gyakran programozási nyelvként emlegetik, de ez nem helyes, mert a HTML segítségével csak egy weboldal struktúráját (elrendezését) hozhatjuk létre, és nem tudjuk használni arra, hogy intelligensebben viselkedjen. [14]

3.1.2. CSS

„A CSS az angol „cascading style sheets" kifejezés rövidítése, ami magyarul „egymásba ágyazott stíluslapokat" jelent. A hangsúly a „stíluson" van – míg a HTML a weblap szerkezetét határozza meg (főcímek, bekezdések, stb.), és lehetővé teszi, hogy különböző elemeket (képek, videók) ágyazz webes dokumentumodba, addig a CSS a weblap vizuális stílusáért felel – az oldal elrendezéséért, a színekért, a betűkészletekért, azok méretéért, és így tovább." [3]

A CSS különféle tulajdonságokat és értékeket használ, amelyeket a HTML elemekre alkalmazhatunk. Ezek a tulajdonságok meghatározzák a szöveg formázását, a betűtípusokat, a színeket, a hátteret, a margókat és egyéb vizuális elemeket. A külső stíluslapok fájljait .css kiterjesztéssel kell elmenteni. Az ilyen stíluslapok tartalmát az egész webhely összes oldalára alkalmazhatjuk anélkül, hogy minden HTML fájl [head] szakaszába be kellene írni. Ez javítja az oldalak forráskódjának átláthatóságát. Emellett elegendő egyetlen fájlt módosítani, és a változtatások megjelennek minden oldalon, amely ugyanazt a CSS fájlt használja. A belső CSS használata akkor válhat szükségessé, ha csak egy oldal stílusán szeretnénk változtatni anélkül, hogy a többi oldalt érintenénk. Például, ha az összes weboldalad egy közös külső CSS fájlra hivatkozik, de van egy

oldal, amelynek saját, egyedi stílusa kell, akkor a belső CSS használata a megfelelő választás. [3]

3.1.3. JavaScript

A JavaScript egy kliensoldali programozási nyelv, amelyet alkalmazások, játékok és webfejlesztés során dinamikus interakciók létrehozására használnak. Gyakran JS-ként emlegetik, és a HTML-lel és CSS-sel együtt a web egyik alapvető technológiájának számít. A fejlesztők a JavaScriptet arra használják, hogy életre keltsenek egy weboldalt, mivel ez átalakítja a statikus oldalakat olyanná, amely reagál a felhasználói interakciókra, mozgással teszi élénkebbé az oldalt, és valós idejű, változó, személyre szabott tartalmat jelenít meg. [15]

3.1.4. React

A mai technológiai környezetben a React a legnépszerűbb JavaScript front-end keretrendszerek között szerepel. Izgalmas látni, hogyan épült be zökkenőmentesen mind a jól bejáratott vállalatok, mind a feltörekvő startupok fejlesztési gyakorlatába. A Reactot eredetileg a Facebook mutatta be 2013-ban, és azóta alapvető technológiává vált, amely számos alkalmazás gerincét alkotja, beleértve azokat is, amelyek azon a platformon futnak, ahonnan származik.

- JSX - A JSX a JavaScript XML rövidítése. Ez egy JavaScriptben használt HTML / XML kiterjesztés, ami megkönnyíti és egyszerűsíti a React komponensek létrehozását.
- Virtuális DOM - A Virtuális DOM egy reprezentációja a DOM objektumnak. Gyakorlatilag ez egy másolata az eredeti DOM-nak. Bármely webalkalmazás módosításakor a teljes felhasználói felületet a Virtuális DOM újra rajzolja. Ezt követően összevetik az eredeti és a Virtuális DOM közötti eltéréseket, és a módosításokat az eredeti DOM állapota szerint alkalmazzák.
- Kiemelkedő teljesítmény - Olyan funkciókkal, mint a Virtuális DOM, a JSX és a komponensek, a React jóval gyorsabb a többi keretrendszernél.
- Android / iOS alkalmazások fejlesztése - A React Native segítségével JavaScript és ReactJS ismeretek birtokában egyszerűen készíthetők Android vagy iOS alapú alkalmazások.

3.1.5. NodeJS

A Node.js egy nyílt forráskódú és keresztplatformos JavaScript futtató környezet. Ez egy népszerű eszköz szinte bármilyen projekt számára. A Node.js a V8 JavaScript motort futtatja, a Google Chrome magját, a böngészőn kívül. Ez teszi a Node.js-t nagy teljesítményűvé. Egy Node.js alkalmazás egyetlen folyamatban fut, anélkül, hogy minden kéréshez új szálat hozna létre. A Node.js egy sor aszinkron I/O primitívet kínál a standard könyvtárában, amely megakadályozza a JavaScript kód blokkolását, és általában a Node.js könyvtárakat nem blokkoló paradigmák használatával írják, tehát a blokkoló viselkedés kivétel, nem pedig a norma. A Node sok tulajdonsága közül számomra fontos tulajdonság lesz az NPM azaz a Node Package Manager. Ennek segítségével tudunk telepíteni különböző könyvtárakat az alkalmazásunkhoz. [6]

3.1.6. Axios

Az Axios egy HTTP kliens a node.js-hez és a böngészőhöz. Izomorf, vagyis ugyanazt a kódbázist használhatja a böngészőben és a node.js környezetben. A szerveroldalon a natív node.js http modult használja, míg a kliensoldalon az XMLHttpRequest-eket alkalmazza.

Jellemzők:

- XMLHttpRequest-ek készítése böngészőből,
- HTTP-kérések küldése node.js-ből,
- ígéret (Promise) API támogatása,
- kérések és válaszok elfogása és módosítása,
- kérések és válaszok adatainak átalakítása,
- kérések megszakítása,
- időkorlátok beállítása,
- lekérdezési paraméterek sorosítása beágyazott elemek támogatásával,
- kérésadatok automatikus sorosítása:
    - JSON (application/json),
    - többrészes/FormData (multipart/form-data),
    - URL kódolt űrlap (application/x-www-form-urlencoded).
- HTML űrlapok JSON-ként való küldése,
- automatikus JSON adatkezelés a válaszban,
- folyamatkövetés böngészőkben és node.js-ben extra információkkal (sebesség, hátralévő idő),
- sávszélesség korlátozás beállítása node.js-ben,
- spec-kompatibilis FormData és Blob támogatás (beleértve a node.js-t),
- kliensoldali támogatás az XSRF elleni védelemhez. [2]

3.1.7. A fejlesztői környezet bemutatása

Egyetemi tanulmányaim során számos programnyelvvel megismerkedtem, ezek alkalmazására viszont szükségem volt egy sokoldalú fejlesztői környezetre. Mint a legtöbb alkalommal, a szakdolgozatom megírásánál is a Visual Studio Code mellett döntöttem.

3.1.8. Visual Studio Code

A Visual Studio Code, gyakran VS Code-ként van említve, a Microsoft népszerű, ingyenes és nyílt forráskódú kódszerkesztője. Sokoldalúságáról, könnyedségéről és számos funkciójáról ismert, ami miatt a fejlesztők körében kedvelt eszközzé vált. A VS Code támogat több programozási nyelvet, mint például a Python, JavaScript, C++ és még sok más, így bármilyen fejlesztési projekthez ideális szerkesztő.

A Visual Studio Code kulcsfontosságú jellemzői:

- Intelligens kiegészítés: A VS Code intelligens javaslatokat, automatikus kiegészítést és gyors javításokat kínál, ami gyorsabbá és hatékonyabbá teszi a kódírást.
- Beépített hibakeresés: Egy erős hibakereső eszköz, amely segít a hibák diagnosztizálásában és megoldásában közvetlenül a szerkesztőben, így a fejlesztők hibamentes kódot írhatnak.
- Kiterjedt bővítmény-piac: A VS Code több ezer nyelvi, keretrendszeri és eszköz bővítménnyel rendelkezik, amelyekkel a fejlesztők személyre szabhatják a munkakörnyezetüket.
- Verziókezelés integráció: Zökkenőmentesen integrálódik a Gittel és más verziókezelő rendszerekkel, megkönnyítve a projektek kezelését, nyomon követését és együttműködését.
- Testreszabás és témázás: A fejlesztők személyre szabhatják a VS Code kinézetét és hangulatát különféle témákkal, ikonokkal és beállításokkal.

A VS Code platformfüggetlen, tehát Windows, MacOS és Linux rendszereken egyaránt működik, biztosítva az egységes élményt, függetlenül attól, milyen operációs rendszert használunk. Ez a rugalmasság és a fejlett funkciók kombinációja miatt mind kezdők, mind pedig profi fejlesztők kedvelt választása. [13]

3.2. Az alkalmazás tervezés lépései

3.2.1. Célok meghatározása

Első lépésként meghatároztam a webalkalmazásom fejlesztésének céljait. Célom volt egy olyan alkalmazás fejlesztése, amely a legmodernebb technológiákat használja. Felhasználóbarát felülettel rendelkezik, átlátható és könnyen használható. Megvalósítja a kriptopénz tárcával való autentikációt és jogosultság kezelést. Blokklánc alapú pénztárca portfólió megjelenítését. Egy teszt szerveren működő, de valós és biztonságos tranzakció megvalósítását.

A célok megértése és pontos meghatározása segített irányt adni a további tervezési folyamatok megvalósításában.

3.2.2. Kutatás és elemzés

Ebben a szakaszban piackutatást és a hasonló tevékenységet végző weboldalak elemzését végeztem. Számos weboldalt megvizsgáltam a felhasználók igényei és szokásai szerint. Megismertem az alkalmazás célcsoportját és így sikerült felismernem milyen megoldások működnek és mik azok, amiket inkább figyelmen kívül kell hagyjak.

3.2.3. Funkcionalitási követelmények meghatározása

Részletesen meghatároztam az alkalmazás funkcióit, milyen alapvető funkciókat fogok biztosítani a felhasználónak és hogy hogyan fogja ezeket elérni. Fontosnak gondoltam, hogy ezek a funkciók a felhasználók igényeinek megfelelően igazodjanak.

3.2.4. Felhasználó felület tervezés

A felhasználói felület tervezéséhez szükségem volt a piackutatásra és a hasonló felszereltséggel rendelkező weboldalak megvizsgálására. A tervezést az elrendezés megtervezésével kezdtem. Az elrendezést szerettem volna kicsit a saját igényeimnek is megfeleltetni. Unalmasnak találom például a sok weboldal által alkalmazott fenti navigációs sávot, ezért ezt én a bal oldalra helyeztem el. Az oldalak felső sávján így kisebb információk kaphattak helyet. A fő tartalmat pedig mind a két oldalam esetén középre tettem, így láthatóan ez az oldal fő eleme.

A színek kiválasztására is nagy hangsúlyt fektetem. Nem szerettem volna túl nagy kontrasztot tartani egyes elemek között, hiszen ez zavaró lehet a szemünknek. Egy alapvetően sötét témát hoztam létre, én nagyon szeretem ezeket a megvalósításokat, gyakran használok ehhez hasonló megoldásokat, ha van rá lehetőség. Nem zsúfoltam tele az alkalmazásom mindenféle fölösleges információval így a felhasználó könnyen megtalál minden funkciót. A felhasználói felületet egy későbbi fejezetben részletesen bemutatom.

3.2.5. Backend tervezés

Gyakran ehhez a lépéshez tartozik még az adatbázis tervezés is, ami az én alkalmazásom esetén nem volt szükséges, hiszen a felhasználó, illetve a tranzakció adatait mind a MetaMask tárolja, így ezt a lépést kihagytam. Így backend téren csupán a saját API-m szerverének a megvalósítását kellett megterveznem. Ebben a lépésben határoztam meg, hogy hogyan fog kommunikálni egymással a backend és a frontend ezáltal hogyan biztosítja az adatbiztonságot.

3.2.6. Fejlesztési technológiák kiválasztása

A fejlesztési folyamat során ki kell választani a megfelelő technológiákat és eszközöket. A frontend fejlesztéshez HTML, CSS, JavaScript és a React keretrendszert használtam, míg a backend fejlesztéshez a Node.js-t választottam. Ezeket a korábbiakban már röviden bemutattam.

3.2.7. Fejlesztés

Miután minden tervezési szempontot figyelembe vettem, elkezdtem a munkát. A fejlesztési fázis során a frontend-el kezdtem mivel az alkalmazásom ezen az oldalon valósít meg több funkciót. A backend-et akkor készítettem el, amikor az oldal azon funkciójához értem, amihez szükséges volt. Csak akkor léptem tovább egyik lépésről a másikra, ha az adott lépést már késznek tekintettem, így garantálva az összes funkció működését.

3.2.8. Prototípus készítés

Fontosnak tartottam a fejlesztés során a prototípus készítést, az átlátható fejlesztés miatt. Mindegy egyes funkció megvalósításához külön projektet hoztam létre, a számomra legkényelmesebb sorrendbe. Amint minden projekt elkészült, a végén már csak össze kellett fűzni az elkészült részeket. Ez a módszer nagyon megkönnyítette a munkámat és rengetek időt megspóroltam vele.

3.2.9. Tesztelés

Az alkalmazás tesztelése során ellenőrizni kell a funkcionalitást, a sebességet, a biztonságot és a felhasználói élményt. Természetesen a munkám során folyamatosan teszteltem az éppen aktuális fejlesztés alatt álló részt, így a hibák könnyebben megtalálhatóak voltak és javíthatóvá váltak.

4. fejezet

Az alkalmazás működésének bemutatása

Ebben a részben elkészült alkalmazásom működését mutatom be, valamint minden olyan technológiát, amit használtam. Az alkalmazást fullstack elemekkel valósítottam meg tehát van frontend és backend része is a projektnek. Adatbázis létrehozására nem volt szükség, hiszen a bejelentkezést MetaMask-os kriptopénz-tárcával, a top 100 token megjelenítését pedig egy külső API segítségével jelenítem meg. Olyan sorrendbe fogom bemutatni az programkódom részleteit, ahogyan én is haladtam a fejlesztés során.

Mire a munkám végére értem a kezdeti alap React projekthez használatos fájlok száma jócskán megnövekedett, a könnyebb átláthatóság és az egyes elemek újrahasználhatósága miatt.

Ha létrehozunk egy új React appot a használandó fájlok az src mappába kerülnek, ami a következőket tartalmazza:

- App.css,
- App.js,
- App.text.js,
- index.css,
- index.js,
- logo.svg,
- reportWebVitals.js,
- setupTests.js.

Ezek közül a fájlok közül számomra az App.css az App.js és az index.js-re volt szükségem.

Az általam létrehozott fájlok:

- costumButton.js,
- Donation.js,

- navbar.js,
- portfolio.js,
- priceing_api.js,
- profile.js,
- table.js,
- walletConnect.js.

Ezeket fogom bemutatni a továbbiakban, kódrészletekkel, illetve a szerkesztett App.js és App.css fájlokat.

Először is bemutatnám azokat a fájlokat, amelyek az átláthatóság és az újra használhatóság miatt hoztam létre:

4.1. costumButton.js

Ez a fájl egy egyszerű újra használható <button> elemet tartalmaz, amit bármelyik fájlba bele lehet importálni és bármennyiszer újra lehet használni.

```
1  const CustomButton = ({ className, text, onClick }) => {
2    return <button className={className} onClick={onClick}>{text}</button>;
3  };
4
5  export default CustomButton;
```

4.1. ábra. customButton kódrészlet

Az újra meghívott gomb 3 adatot kér, a stíluslap osztályának nevét (mindegyik gombra van létrehozva stíluslapon belüli osztály) a gombra írandó szöveget, illetve a gomb akciójához szükséges onClick függvény nevét.

4.2. priceing_api.js

A priceing_api fájlban egyetlen dolog történik, a Coinpaprika API-jának a meghívása axios segítségével. Sokáig keresgettem olyan ingyenesen elérhető API-kat, amik tartalmazzák a számomra szükséges adatokat a kriptovalutákról. Erre a feladatra tökéletesnek bizonyult ez az oldal, hiszen náluk ingyenesen elérhető az a mennyiségű adat, amit a fejlesztés során felhasználtam. Az általam használt adatok: a token neve, jelenlegi ára, az árának változása az utóbbi 1 órában, az árának változása az utóbbi 24 órában, az árának változása az utóbbi 1 hétben, a legutóbbi 24 óra forgalma, illetve a piaci kapitalizáció. Minden pénzügyi adat amerikai dollárban jelenik meg.

Az API hívás a következőképpen néz ki:

```
import axios from 'axios';

const fetchCoinData = async () => {
    try {
        const { data } = await axios.get('https://api.coinpaprika.com/v1/tickers');
        return data.slice(0, 100);
    } catch (error) {
        console.error('Error fetching data:', error.response ? error.response.data : error.message);
        throw error;
    }
};

export default fetchCoinData;
```

4.2. ábra. API hívás

4.3. Coinpaprika

CoinPaprika egy kriptovaluta piaci adatokat szolgáltató platform, amely átfogó információkat kínál a kriptovalutákról. Fő célja, hogy pontos és naprakész adatokat nyújtson a kriptopiaci szereplőkről, ezzel segítve a felhasználókat a megalapozott befektetési döntések meghozatalában.

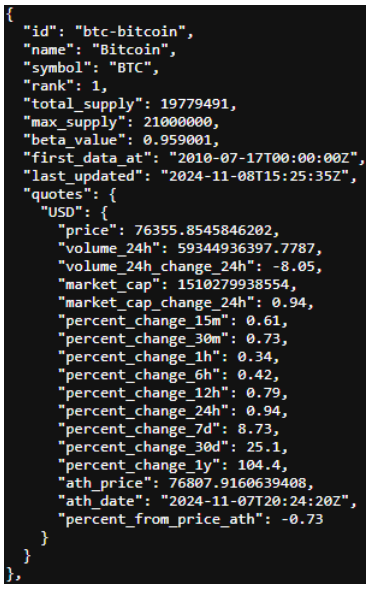
4.3.1. CoinPaprika főbb jellemzői

- Piaci adatok: Több ezer kriptovalutát részletez, beleértve az aktuális árakat, piaci kapitalizációkat, kereskedési volumeneket és kínálatot.
- Tokenek és kriptovaluták listája: A CoinPaprika széles választékot kínál tokenekből és kriptovalutákból, beleértve a legnagyobb piaci kapitalizációjú érméket, mint a BTC, ETH és sok más altcoin.
- Történelmi adatok: A CoinPaprika hozzáférést biztosít történelmi ár adatokhoz és grafikonokhoz, segítve a felhasználókat a trendek hatékony elemzésében.
- API integráció: A CoinPaprika API segítségével a fejlesztők hozzáférhetnek az összes piaci adathoz, megkönnyítve ezzel ezen adatok beépítését különböző alkalmazásokba vagy weboldalakba.
- Piacelemzés és kutatás: A platform különféle piacelemző eszközöket, kutatási jelentéseket és piaci betekintéseket kínál, segítve a felhasználókat a mélyebb megértés megszerzésében.

A CoinPaprika remek forrás mindazok számára, akik átfogó és megbízható kriptovaluta adatokra vágynak. Felhasználóbarát felülete és pontos adat szolgáltatásai különösen hasznosak a kezdő és tapasztalt kriptobefektetők számára egyaránt.

Összességében a CoinPaprika egy sokoldalú és megbízható platform, amely lehetővé teszi a kriptopiacok részletes elemzését, és segíti a felhasználókat a megalapozott befektetési döntések meghozatalában. [5]

A meghívott API első eleme a Bitcoin adatait mutatja, a következőképpen nézett ki 2024.11.08-án:



4.3. ábra. API adatok

Legfontosabb adatok:

- id: az adott token azonosítója,
- name: az adott token neve,
- rank: a listában szereplő helye,
- price: az adott token jelenlegi ára,
- percent_change_1h: az árának változása az utóbbi 1 órában,
- percent_change_24h: az árának változása az utóbbi 24 órában,
- percent_change_7d: az árának változása az utóbbi 1 hétben,
- volume_24h: a legutóbbi 24 óra forgalma,
- market_cap: a piaci kapitalizáció.

A https://api.coinpaprika.com/v1/tickers API-jában 2000 token található, ebből jelenítettem meg a top 100-at.

4.4. App.js

Az alkalmazás főoldalát az App.js fájlban valósítottam meg ami számos importot tartalmaz különböző komponensei miatt. Található ezen az oldalon egy navigációs sáv, ahol a legfontosabb funkciókat elérhetjük. Az oldalon megjelenítem a két leghíresebb kriptovalutának az aktuális árfolyamát melyhez szükségem volt egy API-ra, amit a priceing_api.js fájlban hívok meg. Az oldal fő komponense egy táblázat, ahol felsorolásra

került a TOP 100 aktuális kriptovaluta különböző adatai, amit szintén a priceing_api segítségével jelenítek meg, rendezési funkciókkal. Emellett a Főoldalról képesek vagyunk a navigációs sáv segítségével átlépni a Profile oldalra.

A képernyő felső sávján megjelenített Bitcoin és Ethereum árának megjelenítéséhez szükséges kódrészlet:

```
const [bitcoinPrice, setBitcoinPrice] = useState(null);
const [ethereumPrice, setEthereumPrice] = useState(null);

useEffect(() => {
  const fetchPrices = async () => {
    const coinData = await fetchCoinData();
    const bitcoin = coinData.find(coin => coin.name === 'Bitcoin');
    const ethereum = coinData.find(coin => coin.name === 'Ethereum');

    if (bitcoin && ethereum) {
      setBitcoinPrice(bitcoin.quotes.USD.price);
      setEthereumPrice(ethereum.quotes.USD.price);
    }
  };

  fetchPrices();
}, []);


<h5>Bitcoin: {bitcoinPrice !== null ? `${bitcoinPrice.toFixed(2)}$` : 'Loading...'}</h5>
<h5>Ethereum: {ethereumPrice !== null ? `${ethereumPrice.toFixed(2)}$` : 'Loading...'}</h5>
```

4.4. ábra. Bitcoin és Ethereum árának megjelenítéséhez

Amíg az API-tól lekért adatok megérkeznek a helyükön a „Loading...” szöveg látható. Ezen a megjelenítésen kívül az oldalon a TOP 100 token adatait felsoroló táblázat található. Ezt a táblázatot a könnyebb kezelhetőség miatt a table.js fájlba helyeztem.

4.5. table.js

A table.js fájl dolgozza fel és jeleníti meg a képernyőn azokat az adatokat, amiket a priceing_api sikeresen lekér nekünk az API hívás segítségével. Az elem létrehoz egy táblázatot, aminek az első sora fix:

- Name (a token neve),
- Priceing (jelenlegi ára),
- 1 h (az árának változása az utóbbi 1 órában),
- 24 h (az árának változása az utóbbi 24 órában),
- 7 d (az árának változása az utóbbi 1 hétben),
- 24 h traffic (a legutóbbi 24 óra forgalma),
- Market capitalization (a piaci kapitalizáció).

Az oszlopok nevei rendező gombok is egyben. Ezekkel a gombokkal lehet abc sorrendbe állítani az adott listát a neveik alapján, illetve a szám adatok alapján csökkenő vagy növekvő sorrendbe.

A táblázat többi sora az egyes tokenek adatait tartalmazza. A megjelenítés azonosító alapján zajlik viszont ez nem szerepel a képernyőn. A Priceing, 24 h traffic, és a Market capitalization oszlopaiban szereplő adatok mögött megjelenik az „USD" felirat, a 1 h, 24 h és 7 d oszlop adatai pedig százalékosan vannak megjelenítve a könnyebb láthatóság miatt. A táblázat sora attól függ, hogy a priceing_api fájlban mennyi tokent állítunk be az API lekéréshez.

A rendező funkciót a következőképpen oldottam meg:

```
const sortData = (key) => {
    let direction = 'ascending';
    if (sortConfig.key === key) {
        if (sortConfig.direction === 'ascending') {
            direction = 'descending';
        } else if (sortConfig.direction === 'descending') {
            direction = 'none';
        }
    }

    if (direction === 'none') {
        setData(originalData);
    } else {
        const sortedData = [...data].sort((a, b) => {
            if (typeof a.quotes.USD[key] === 'number' && typeof b.quotes.USD[key] === 'number') {
                return direction === 'ascending' ? a.quotes.USD[key] - b.quotes.USD[key] : b.quotes.USD[key] - a.quotes.USD[key];
            }
            return direction === 'ascending' ? a[key].localeCompare(b[key]) : b[key].localeCompare(a[key]);
        });
        setData(sortedData);
    }

    setSortConfig({ key, direction });
};
```

4.5. ábra. Az adatok rendezése

4.6. navbar.js

Ez az elem az alkalmazásom egyik legfontosabb eleme, hiszen egyszerre kell két oldalon is jelen lennie, hogy megvalósuljon a kettő között az átlépés. Ez az elem tartalmazza az alkalmazásom logóját és nevét, itt lehet belépni a MetaMaskos kriptopénz tárcánkkal és még a kripto donation funkcióhoz szükséges elemek is ezen a sávon találhatóak.

A MetaMasknak van saját felülete a belépéshez. Ehhez le kellett töltenem a böngészőmhöz a MetaMask saját bővítményét, amelyet a későbbiekben használok. A belépési felület integrálásához létre hozta a walletConnect.js fájlt az átláthatóság miatt. A belépést követően az oldal megjeleníti a belépési gombon a bejelentkezett felhasználó azonosítóját így meggyőződhetünk róla, hogy valóban be vagyunk jelentkezve. A biztonság érdekében az azonosítóból csak az első 8 karaktert jeleníti meg az oldal:

4.7. walletConnect.js

Ebben a fájlban több elem is található, de elsősorban a belépési funkció megvalósítására készítettem. Mivel ebben a fájlban kapom meg a felhasználói fiók adatait, így ebben a fájlban oldottam meg a fiók azonosítójának a tovább küldését a portfólio.js részére, ez elengedhetetlen a felhasználó tokenjeinek a megjelenítéséhez. Enélkül külön a Profil oldalon kellene megadnunk azt az azonosítót, aminek az adatait le szeretnénk kérni:

```
const connectWalletHandler = async () => {
  if (window.ethereum && window.ethereum.isMetaMask) {
    try {
      const accounts = await window.ethereum.request({ method: 'eth_requestAccounts' });
      const displayAddress = `${accounts[0].slice(0, 8)}...`;
      setUserAddress(displayAddress);
      setConnButtonText(`Your address: ${displayAddress}`);
    } catch (error) {
      setErrorMessage(error.message);
    }
  } else {
    setErrorMessage('Please install the MetaMask browser extension to interact');
  }
};
```

4.6. ábra. Bejelentkezés, azonosító megjelenítés

```
const fetchTokens = async (address) => {
  const chains = ['ethereum', 'avalanche', 'arbitrum', 'base', 'linea', 'optimism', 'scroll'];
  const fetchedTokenData = {};

  try {
    for (const chain of chains) {
      const response = await fetch(`http://localhost:3500/${chain}?address=${address}`);
      if (!response.ok) {
        throw new Error(`Error fetching ${chain} chain: HTTP ${response.status}`);
      }
      const data = await response.json();
      fetchedTokenData[chain] = data;
    }
    setTokenData(fetchedTokenData);
  } catch (error) {
    console.error("Error fetching tokens:", error);
    setErrorMessage("Failed to fetch token data.");
  }
};
```

4.7. ábra. Azonosító továbbítása

4.8. profile.js

A profile.js fájlban található a Profil oldal, ahova a navigációs sávon levő gomb segítségével lehet eljutni. A felület számos szempontból hasonlít a Főoldalra, itt is megtalálható ugyan az a Bitcoin és Ethereum aktuális árfolyam megjelenítés az oldal tetején, illetve ezen az oldalon is megtalálható a navigációs sáv. Ennél a pontnál azt is meg kellett oldanom, hogy miután a felhasználó bejelentkezett, de oldalt vált a Főoldal, illetve a Profil oldal között, a „Your address…” gombon az azonosító mind két oldalon megjelenjen frissítés nélkül. Így a navigációs sáv nem lehetett egy újra felhasználható elem, hiszen, ha így oldottam volna meg, akkor újra és újra rá kellene kattintanunk a „Login with MetaMask wallet” feliratú gomra, hogy a gombon az azonosítónk jelenjen meg. Így amellett döntöttem, hogy a Főoldalon hívom meg a navbar.js fájlt és továbbítom a portfolio.js-nek, hogy a gombon levő felirat ne változzon oldalváltáskor.

Az App.js fájlban a navigációs sáv továbbítása így néz ki:

A Profil oldal legfőbb eleme viszont nem a navigációs sáv, hanem a portfólió, amely 7db láncon jeleníti meg a felhasználó összes tokenjét. Ahhoz, hogy ezt megjelenítsem szükségem volt egy backendre, illetve egy saját API-ra. Saját API létrehozásához a Moralist választottam.

```
const App = () => {
  return (
    <Router>
      <Navbar />
      <Routes>
        <Route path="/" element={<BurschCoin />} />
        <Route path="/profile" element={<Profile />} />
      </Routes>
    </Router>
  );
};
```

4.8. ábra. A navigációs sáv továbbítása

4.9. Moralis

A Moralis egy fejlett Web3.0 fejlesztési platform, amely lehetővé teszi, hogy gyorsan és egyszerűen építs blokklánc alkalmazásokat. A Moralis célja, hogy jelentősen csökkentse a Web3.0 alkalmazások fejlesztési idejét felhasználóbarát API-k, SDK-k és különféle fejlesztői eszközök biztosításával.

A Moralis főbb jellemzői:

- Web3.0 API: A Moralis Web3.0 API hozzáférést biztosít különféle blokklánc adatokhoz, beleértve a tranzakciókat, felhasználói tokeneket, NFT-ket, okosszerződéseket stb., könnyű integrációval bármilyen alkalmazásba.
- Moralis SDK: Az SDK segítségével a fejlesztők könnyedén hozzáférhetnek Web3.0 funkciókhoz, például a felhasználók MetaMask vagy WalletConnect segítségével történő hitelesítéséhez.
- Adatbázis: A Moralis lehetővé teszi a fejlesztők számára, hogy valós idejű adatokat tároljanak egy skálázható adatbázisban, amely automatikusan szinkronizál a blokklánccal.
- Moralis Streams: Ez a funkció lehetővé teszi blokklánc események, például tranzakciók vagy okosszerződés változások valós idejű továbbítását.
- NFT és Token támogatás: A Moralis számos eszközt kínál NFT fejlesztéshez, például NFT metaadatok lekéréséhez és megjelenítéséhez.
- Több láncot támogató kompatibilitás: A Moralis több blokklánc hálózatot támogat, beleértve az Ethereumot, a Binance Smart Chaint, a Polygont és az Avalanche-et, lehetővé téve alkalmazások fejlesztését több láncon keresztül.

A blokklánc alapú alkalmazások fejlesztési folyamatának egyszerűsítésével a Moralis nagyban megkönnyíti a Web3.0 fejlesztők munkáját. Egyedi backend fejlesztése helyett a Moralis biztosítja az infrastruktúrát, amely lehetővé teszi a dApp-ok gyorsabb és hatékonyabb létrehozását.

4.10. API kulcs

Az API kulcs egy egyedi kód, amelyet egy API (alkalmazásprogramozási interfész) használ az alkalmazás vagy felhasználó azonosítására, aki meghívja azt. Az API kulcsok

segítenek nyomon követni és monitorozni, ki használja az API-t és hogyan, továbbá az alkalmazások hitelesítésére és engedélyezésére is használhatók – hasonlóan ahhoz, ahogy a felhasználónevek és jelszavak működnek. Egy API kulcs lehet egyetlen kulcs vagy több kulcsból álló készlet. A felhasználókat arra ösztönzik, hogy kövessék a legjobb gyakorlatokat az API kulcsok jogosulatlan használatának elkerülése érdekében, ezáltal növelve a biztonságot az API kulcsok lopása ellen. [1]

Az API kulcs létrehozásához regisztráltam az oldalra és az ingyenes csomagjukban meg is kaptam az egy darab saját API kulcsomat.

4.11. Backend

A Moralis segítségével létrehoztam egy saját szervert, amit a http://localhost:3500/ porton futtatok. Itt történik meg a felhasználó azonosítója segítségével 7db láncon az adatok lekérése. Legelőször lekérem a felhasználó teljes natív balanszát, ami a blokklánc összes vagyona. Ezt bontom szét a különböző blokkláncok token balanszára, amivel már tudok dolgozni. Az Ethereum lánc adatainak felvétele így néz ki:

```
async function getDemoData(address, chain) {
    const nativeBalance = await Moralis.EvmApi.balance.getNativeBalance({
        address,
        chain,
    });
    const native = nativeBalance.result.balance.ether;

    const tokenBalances = await Moralis.EvmApi.token.getWalletTokenBalances({
        address,
        chain,
    });

    const tokens = tokenBalances.result.map((token) => token.display());

    return { native, tokens };
}

app.get("/ethereum", async (req, res) => {
    const address = req.query.address;
    if (!address) {
        return res.status(400).json({ error: "Address is required" });
    }

    try {
        const data = await getDemoData(address, EvmChain.ETHEREUM);
        res.status(200).json(data);
    } catch (error) {
        console.error(error);
        res.status(500).json({ error: error.message });
    }
});
```

4.9. ábra. Ethereum lánc

A többi lánc adatainak a felvételéhez is a getDemoData függvényt használom.

4.12. portfolio.js

Ebben a fájlban valósul meg a felhasználó tokenjeinek a kiíratása. Mivel a walletConnect.js fájlban tovább küldtük a felhasználó azonosítóját így ezt már csak kezelnünk kell. Miután átvettük az azonosítót, hozzá köthetjük a meglévő token láncainkkal a szerver API-jához:

```
useEffect(() => {
  if (address) {
    const fetchBalance = async () => {
      const chains = ['ethereum', 'avalanche', 'arbitrum', 'base', 'linea', 'optimism', 'scroll'];

      try {
        const newBalances = {};

        for (const chain of chains) {
          const response = await fetch(`http://localhost:3500/${chain}?address=${address}`);
          if (!response.ok) {
            const errorDetails = await response.text();
            throw new Error(`HTTP error! status: ${response.status}, details: ${errorDetails}`);
          }
          const data = await response.json();

          newBalances[chain] = {
            balance: (parseFloat(data.native) || 0).toFixed(5),
            tokens: data.tokens.map(token => {
              const [value, name] = token.split(' ');
              return { name: name || 'Token', amount: (parseFloat(value) || 0).toFixed(5) };
            }),
          };
        }

        setBalances(newBalances);
      } catch (error) {
        console.error(`Error fetching data:`, error);
        setError(`Error loading data: ${error.message}`);
      }
    };
    fetchBalance();
  }
}, [address]);
```

4.10. ábra. Az azonosító és a láncok összekötése

A lekért token láncok: Ethereum, Avalanche, Arbitrum, Base, Linea, Optimism, Scroll

Ebben a fájlban történik a kiíratása is a felhasználói adatoknak hasonlóképpen a table.js fájlhoz itt is táblázatban jelennek meg az adatok. A táblázatnak annyi sora van ahány tokenje az adott láncon a felhasználónak.

4.13. Donation.js

Ebben a fájlban valósítom meg az adomány funkcióját az oldalnak. A MetaMask oldalán létezik módszer tranzakció integrálására, így ezt vettem alapul fejlesztés közben. Annyi csupán a különbség, hogy abban a megoldásban nekünk kell külön megadnunk a feladó azonosítót és a cél azonosítót. Mivel a bejelentkezett felhasználó azonosítóját már a bejelentkezést követően továbbítottam, így nem volt más dolgom, mint ebben a

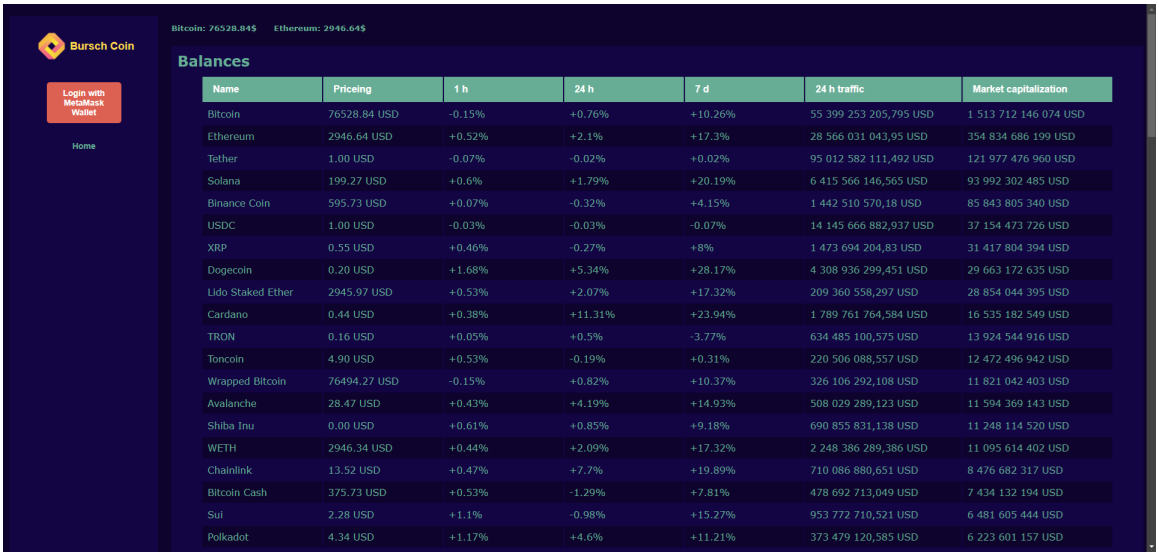
fájlban is kezelni azt. Egy olyan megoldást találtam ki a tranzakcióhoz, ami könnyen megvalósítható és ez az adományozás. Így nem kell külön megadnunk a kiinduló fiók azonosítóját és a cél azonosítóját sem. A tranzakciót indító fiók az lesz, amivel bejelentkeztünk a fogadó pedig az, amit megadunk a programkódban, ezesetben az én saját fiókom azonosítója. Az adományozó funkció során a bejelentkezett felhasználó egy inputba a saját pénztárcája keretein belül megadhat egy bizonyos összeget, amit az oldal elutal teszt szerveren keresztül a kódban levő azonosítóra. A fejlesztés során sokszor bele futottam a tranzakciós költségbe. A probléma fő oka a kripto valuták aktuális árának folyamatos változása, ezzel egyetemben a tranzakciós költség is folyamatosan változik. Ennek a kezelését is sikerült megoldanom, így a felhasználó által megadott átküldendő összegből előre levonjuk a tranzakciós költséget az aktuális árhoz képest. A tranzakció a MetaMask saját teszt szerverén megy végbe.

5. fejezet

A felhasználói felület bemutatása

5.1. Főoldal

Amint megnyitjuk az oldalt először a Főoldalt fogjuk látni:



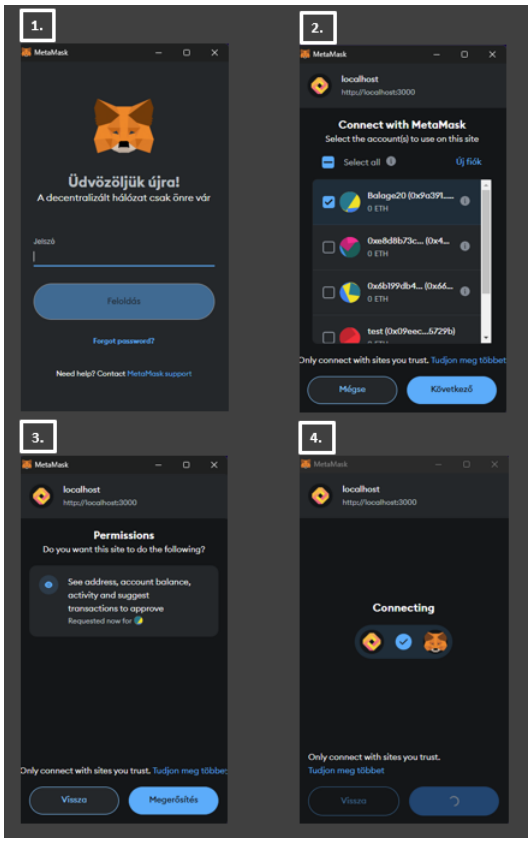
5.1. ábra. Főoldal

A Főoldalon látható bal oldalon a navigációs sáv, mellette legfelül a Bitcoin és az Ethereum tokenek jelenlegi árfolyama, alatta pedig az a táblázat, ami felsorolja nekünk a TOP 100 token jelenlegi adatait. A táblázat legfelső sorain gombok helyezkednek el, amit egyszeri lenyomásra abc sorrendbe (a név esetén) vagy csökkenő sorrendbe (az összes többi oszlop esetén) tudjuk rendezni a táblázat adatait. A navigációs sávon 3 elem látható:

- Az oldal logója és mellette a neve. Ez egy gomb, amire, ha rákattintunk az oldal a lap tetejére ugrik bárhol is legyünk az oldalon.
- A „Login with MetaMask Wallet” gomb. Erre kattintva tudunk bejelentkezni a MetaMask fiókunba.
- A „Home” gomb, ezzel a Főoldalra tudunk ugrani.

5.2. Bejelentkezés

Ha rákattintunk a „Login with MetaMask Wallet” gombra fel fog ugrani a MetaMask bejelentkezési felülete, ahol be tudunk jelentkezni. Miután megadtuk a jelszót az oldal megkérdezi, hogy a fiókhoz tartozó melyik azonosítókat szeretnénk az oldalhoz csatlakoztatni (itt annyi jelenik meg amennyi a fiókunkhoz tartozik). Majd az oldal megkérdezi tőlünk, hogy hozzá járulunk-e, hogy használja a fiók adatainkat (azonosító, egyenleg stb. . .):

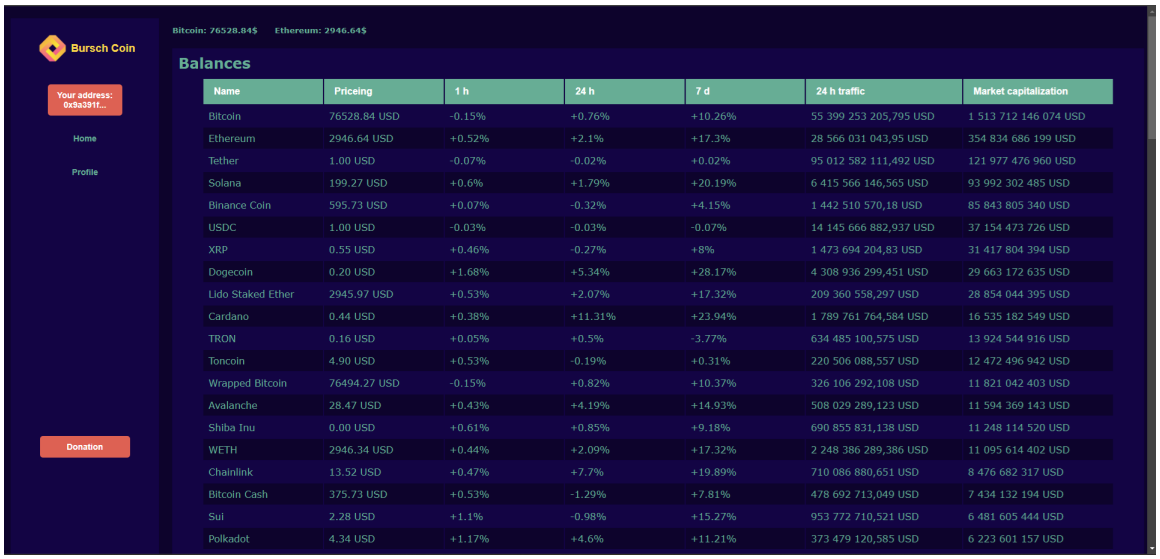


5.2. ábra. Bejelentkezési felület

1. A Felugró ablak, ahol meg tudjuk adni a jelszónkat a bejelentkezéshez.
2. Itt tudjuk kiválasztani, hogy melyik azonosítónkat csatlakoztassuk az oldalhoz.
3. Itt tudjuk megengedni az oldalnak, hogy hozzáférjen a fiókunk adataihoz a „Megerősítés” gombra kattintva.
4. Ezt a felületet fogjuk látni amíg az oldal hozzákapcsolódik a kriptopénz tárcánkhoz.

5.3. Főoldal a bejelentkezés után

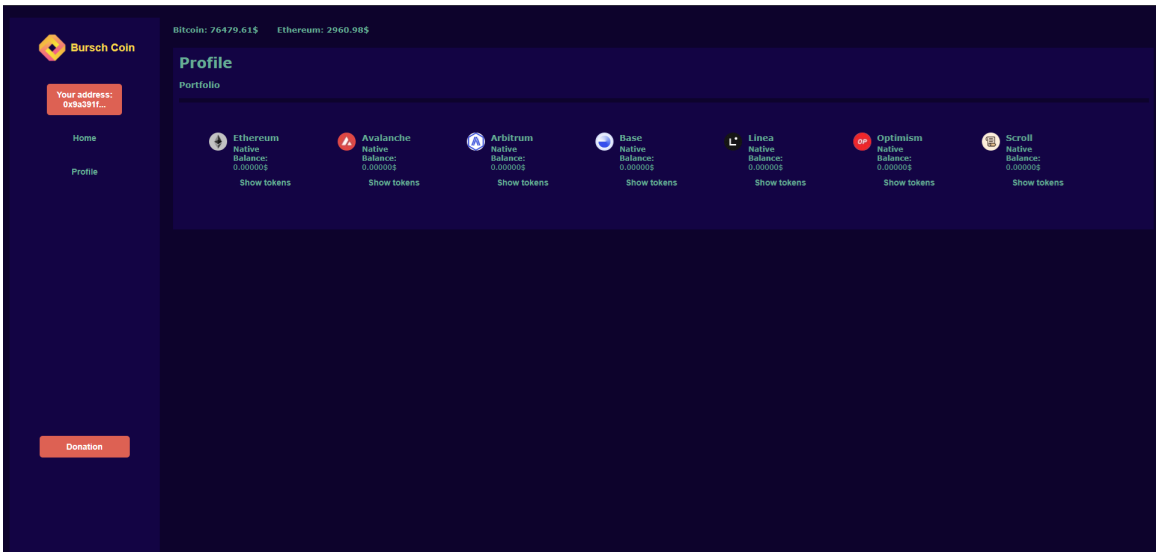
Miután megtörtént a bejelentkezés, a Főoldalunkon történni fog egy kis változás, meg fog jelenni pár új elem a navigációs sávon, illetve a „Login with MetaMask Wallet” gomb feliratára a „Your address: . . . .” felírat került így láthatjuk is, hogy bejelentkeztünk:



5.3. ábra. Főoldal bejelentkezés után

5.4. Profil oldal

Ha rákattintunk a „Profile” gombra az oldal át navigál minket a Profil oldalra, ahol már a bejelentkezett felhasználói adatainkat láthatjuk. A navigációs sáv megtartotta a kinézetét, illetve a lap tetején itt is látható a Bitcoin és az Ethereum jelenlegi árfolyama. Ellenben itt a Főoldali táblázat helyett azokat a token láncokat láthatjuk, amelyeken az oldal képes megjeleníteni a felhasználói fiók adatait:

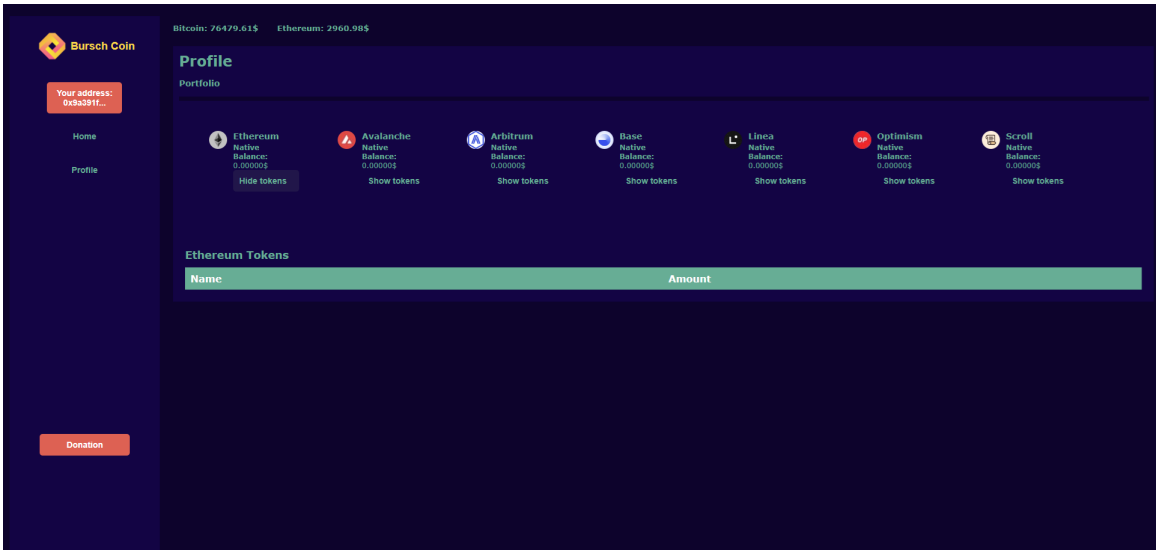


5.4. ábra. Profil felület

5.5. Tokenek megtekintése

Láthatjuk a 7db tokenláncot amin megtekinthetjük a fiókunk adatait. Ha bármelyik tokennél rákattintunk a „Show tokens” gombra, lentebb megjelenik egy táblázatba a tokenjeink neve és a birtokunkban levő tokenek darabszáma és a gomb szövege „Hide

tokens”-re vált. Amint rákattintunk egy másik token „Show tokens” gombjára az előbb megnyitott eltűnik és az aktuális fog látszani:



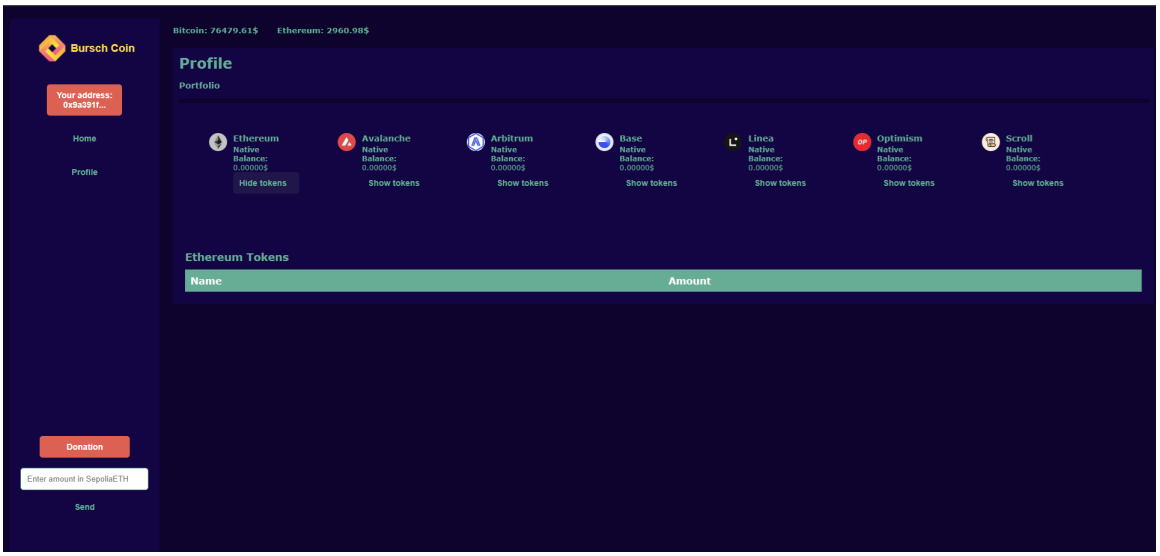
5.5. ábra. Tokenek megjelenítése

Egy látványosabb portfólió, itt látható a tokenen láncon kenti vagyonunk és az éppen megnyitott tokenek:

5.6. ábra. Megnyitott tokenek

5.6. Adakozás funkció

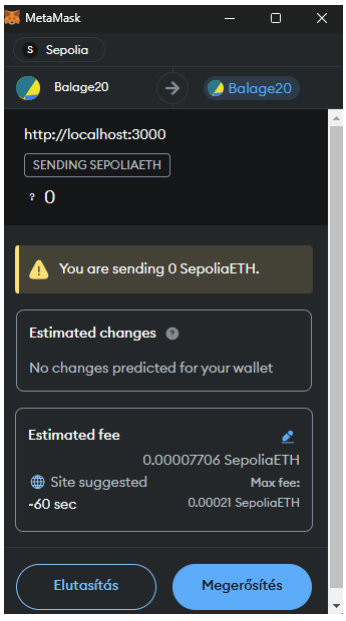
Ha a navigációs sávra tekintünk még nem beszéltünk a bejelentkezés után megjelent „Donation” gombról. Ha erre rákattintunk, alatta megjelenik 2 új elem. Az első elem egy imput sáv, ahol meg tudjuk adni milyen összeggel szeretnénk adakozni, a második pedig egy „Send” azaz küldés gomb:



5.7. ábra. Adakozás funkció

5.7. A tranzakció megerősítése

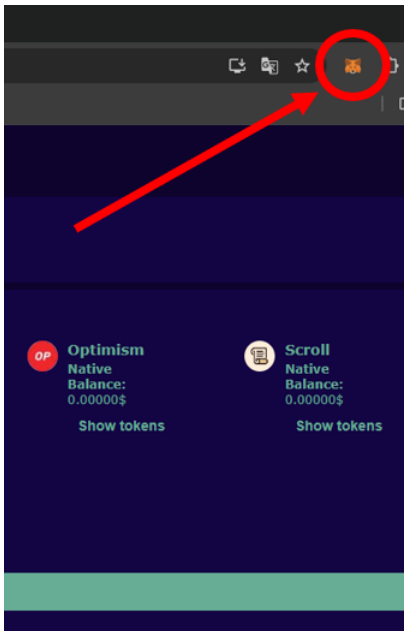
Ha beírtuk az átküldeni kívánt összeget kattintsunk a „Send” gombra, ekkor a MetaMask felugró ablaka fel fogja ajánlani az átküldeni kívánt összeggel a tranzakció lehetőségét:



5.8. ábra. A tranzakció megerősítése

5.8. A tranzakció elindítása

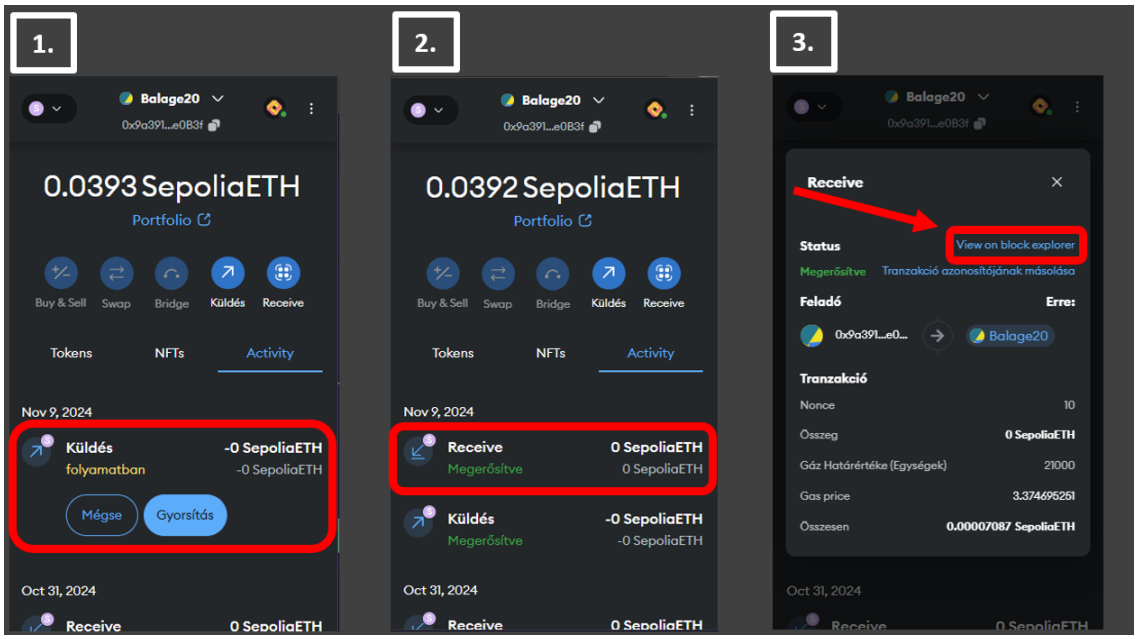
Ha a „Megerősítés” gombra kattintunk a kis ablak el fog tűnni és a tranzakció elindul, ha az „Elutasítás” gombra kattintunk az oldal értesít minket, hogy a tranzakció nem zajlott le. Annak érdekében, hogy nyomon követhessük a tranzakciónkat kattintsunk a böngészőnk jobb felső sarkában található a MetaMask bővítményre:



5.9. ábra. MetaMask bővítmény elérése

5.9. A tranzakció részleteinek megtekintése

Ha rákattintottunk a bővítményben megjelenik a megindított tranzakció. Amint várunk egy kicsit, a státusza „folyamatban”-ról „Megerősítve”-re vált. Sajnos itt a „Gyorsítás”-gombbal nem történik időbeli különbség. Ha „Megerősítve” státuszt látunk, kattintsunk a tranzakcióra, hogy megtekintsük annak részleteit. Először egy kisebb áttekintőt fogunk látni a bővítményen belül. Ha az értékelőn belül a „View on block explorer” gombra kattintunk egy új oldal nyílik meg ahol részletesebb áttekintést fogunk látni a tranzakciónkról.

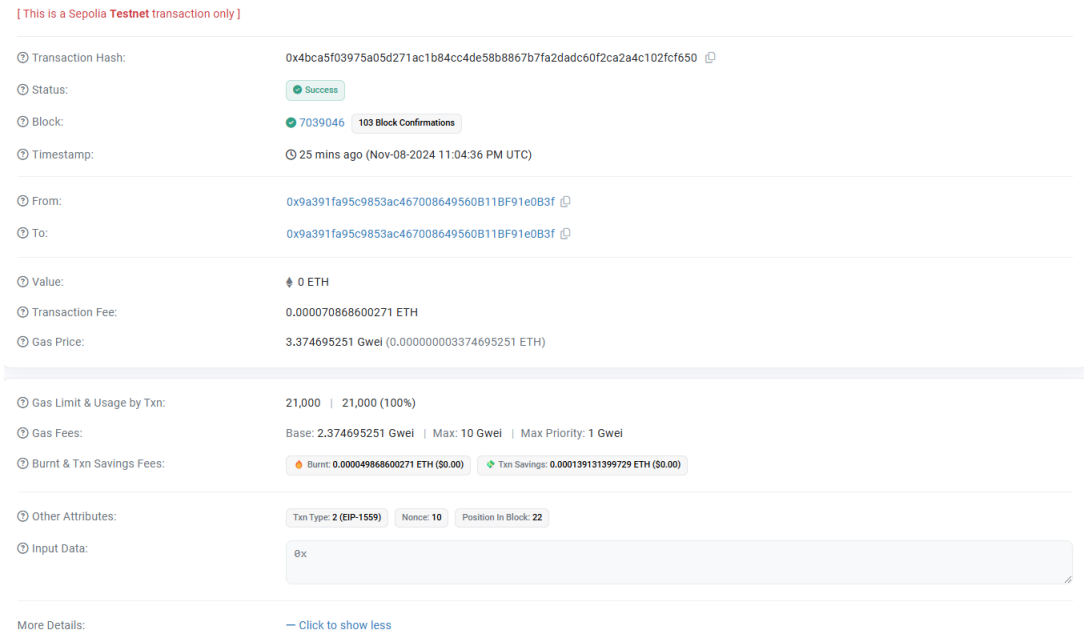


5.10. ábra. A tranzakció részletes megtekintése

1. Megjelent a tranzakció a bővítményben tehát elküldtük az összeget.
2. A státusz „Megerősítve” rendben elutaltuk az összeget, kattintsunk a tranzakcióra.
3. Megjelenik a kisebb áttekintő kitől kihez ment az összeg stb... kattintsunk a „View on block explorer” gombra a részletesebb adatokért.

5.10. A tranzakció részleteinek megtekintése a MetaMask oldalán

Miután rákattintottunk a „View on block explorer” gombra a következő oldal fogad minket:



5.11. ábra. Tranzakciós adatok a MetaMask oldalán

Ezen a felületen láthatjuk a részletesebb adatokat:

- Pontosan mikor zajlott az utalás.
- Kitől kihez ment az összeg, a fogadó és a feladó fél teljes azonosítója.
- Mennyi volt a pontos összeg, ami átutalásra került az átutalási költség nélkül.
- Milyen valutában zajlott a tranzakció.

6. fejezet

Keresőmotor optimalizálás

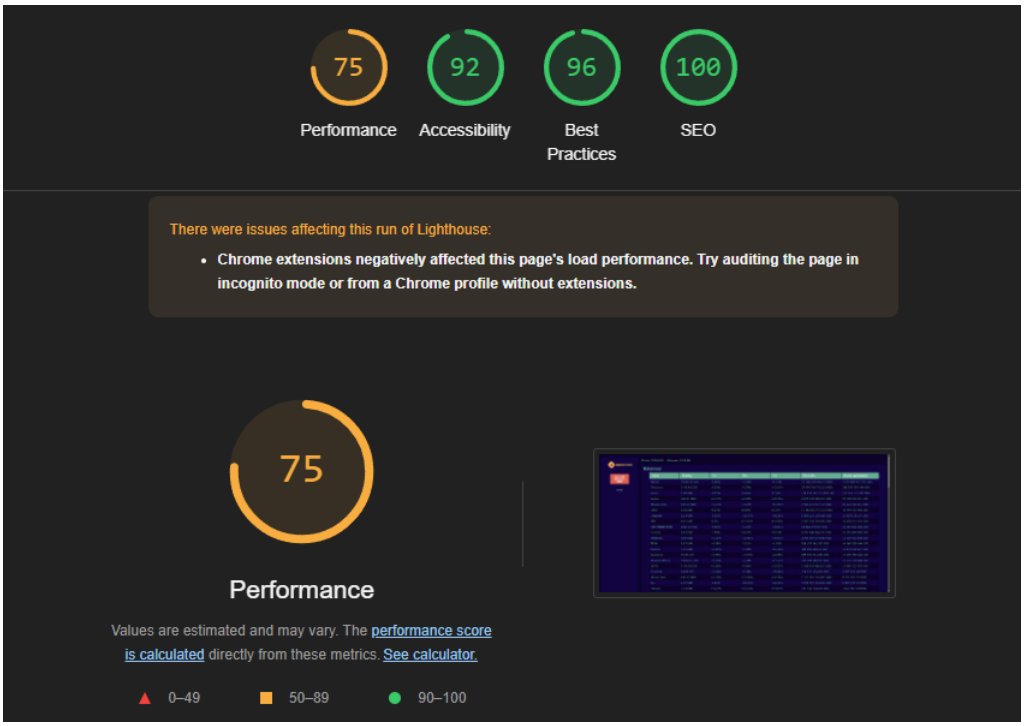
A keresőoptimalizálás (SEO) egy olyan folyamat, amelynek célja a weboldal nem fizetett keresőmotoros rangsorolásának javítása. Minél magasabb a helyezés a keresési eredmények között, annál több látogató érkezik az oldalra. Az egyik legegyszerűbb módszer annak ellenőrzésére, hogy az oldalunk mennyire optimalizált és a különböző keresőmotorok milyenre értékelik a Google Chrome - Developer Tools - Lighthouse segítségével lehet. Bármelyik weboldalon megnyomva az F12 gomb-ot megjelenik a DevTools felület és a felső menüsávból ki kell választani a Lighthouse fület. A jobb oldalon található „Analyze page load" gombra kattintva elkezdi kielemezni a kívánt weboldalt. A négy alábbi kategóriát szoktuk figyelembe venni általában egy weboldalnál mikor ellenőrizzük a munkánkat:

- Performance: Azaz, a weboldal sebessége. Ez több fontos elemből is áll, mint például, az, hogy mennyi idő után jelenik meg teljes egészében a weboldal vagy mennyi idő elteltével lesz használható. Az rossz eredmények oka legtöbbször az alábbiak:
  - Túl nagy képek használata,
  - A JavaScript fájlok helytelen vagy rossz helyen való betöltése,
  - Olyan CSS formázások, illetve JavaScript funkciók betöltése, amik nincsenek használva,
  - Ha a CSS és JavaScript fájlok nincsenek letömörítve.
- Accessibility: Magyarul megközelíthetőség. Ez az oldal olvashatósága, illetve használhatósága. Ennél a pontnál az alábbi hibákat emelném ki egy weboldal esetében:
  - A képek „alt" attribútuma nincs megadva. Ez azért fontos, mert ezek alapján könnyebben találja meg a weboldal képeit a Google képkereső, így akár egy kép alapján is érkezhet felhasználó az oldalra. Továbbá a nem tudja valamiért betölteni a képet az oldal, akkor legalább az itt megadott szöveg megjelenik.
  - A linkek, gombok és egyéb elemek kattinthatósága.
  - Olvashatatlan szövegeket. Például világos szürke háttéren fehér szöveg.
- Best Practices: Itt leginkább a weboldal kódolását, illetve az abból eredő hibákat elemzi. Itt a legkritikusabb hibák az alábbiak:

  - HTTPS hiánya.
  - Elavult JavaScript könyvtárak használata, amik biztonsági rést jelenthetnek.
  - Torzított kép megjelenítés, tehát amiknek az arányai nem egyeznek a képével.

- SEO: Ez a rész kizárólag a weboldal elemeit veszi figyelembe. Fontos megemlíteni, hogy nem elég, ha betartjuk a SEO-hoz szükséges konvenciókat, hanem azok tartalma is fontos. Ez az elemzés csak annyit vesz figyelembe, hogy kód szinten szerepel-e minden, aminek kell, hogy az oldal kereső optimalizálása elvégezhető legyen.

Miután megtörtént a kielemzés a lenti képen láthatjuk a szoftver eredményeit. Az egyes hibákra kattintva a többet is olvashatunk a hibákról és megoldási javaslatokat is kapunk a javításra.



6.1. ábra. Az alkalmazás eredményei

7. fejezet

Összefoglalás

A feladatom egy olyan modern webalkalmazást fejlesztése volt, amely kizárólag Web3.0 technológiákat alkalmaz. Az alkalmazás lehetőséget biztosít kriptovaluta pénztárcával való autentikációra és jogosultság kezelésre. Blokklánc alapú pénztárca portfólió lekérésére, és egy biztonságos adományozó funkcióra. Fontosnak tartottam a komponensek újra felhasználhatóságát, későbbi fejlesztések leegyszerűsítése és gyorsítása érdekében. A végső cél egy mindennapi használatra megfelelő alkalmazás fejlesztése volt, kriptopénz tárcával rendelkező felhasználók számára. Az alapos előzetes felkészülésnek köszönhetően minden kitűzött célt sikerült elérnem. Ám ez koránt sem jelenti azt, hogy az alkalmazás elkészült. Több lehetőség is megfordult a fejemben, amiket szeretnék a későbbiekben megvalósítani (pl,: több nyelven való megjelenítés, sötét/világos téma választhatósága). Hozzátenném, hogy munkám során számos új ismerettel gazdagodtam, amiket a későbbiekben hasznosítani fogok tudni. Nem dolgoztam még ekkora alkalmazáson, izgalmasnak és tanulságosnak találtam. A fejlesztés ideje alatt, alkalmam volt olyan emberekkel is konzultálni, akik nálam sokkal jártasabban az informatika világában, tőlük rengeteg haszon tippet kaptam akár a tervezéshez, akár a fejlesztéshez. A szakdolgozatom végéhez érve úgy érzem nem csak az alkalmazást fejlesztettem, hanem ezzel egyidőben magamat is.

8. fejezet

Summary

My task was to develop a modern web application that uses only Web3.0 technologies and provides cryptocurrency wallet authentication and privilege management functions. Blockchain based wallet portfolio retrieval and a secure donation function. I considered it important to reuse components to simplify and speed up future development. The ultimate goal was to develop an application suitable for everyday use by users with a cryptocurrency wallet. Thanks to the thorough preparation beforehand, I was able to achieve all the goals I defined. But this does not mean that the application is ready. I had a number of potential functions that I would like to implement in the future (e.g. multi-language display, dark/light theme switch). I would add, that my work has given me a lot of new knowledge that I will be able to use in the future. I have not worked on an application of this size before and found it exciting and instructive. During development, I also had the opportunity to consult with people who are more knowledgeable than I am in the world of IT, and they gave me a lot of useful tips for both design and development. By the end of my thesis, I feel that I have not only improved the application, but also myself at the same time.

Irodalomjegyzék

[1] Binance academy. Mi az az api-kulcs, és hogyan használható biztonságosan? https://academy.binance.com/hu/articles/what-is-an-api-key-and-how-to-use-it-securely, 2023.

[2] Axios. What is axios? https://axios-http.com/docs/intro, 2024.

[3] Máté Balázs. Mi az a css? https://matebalazs.hu/css.html, 2024.

[4] Coinbase. What is a crypto wallet? https://www.coinbase.com/learn/crypto-basics/what-is-a-crypto-wallet, 2024.

[5] Coinpaprika. The only cryptocurrency api solution you will ever need. https://coinpaprika.com, 2024.

[6] NodeJS. Run javascript everywhere. https://nodejs.org/en, 2024.

[7] Chiara Quiocho. What are centralized and decentralized networks? https://www.ninjaone.com/it-hub/endpoint-management/centralized-and-decentralized-networks/, 2024.

[8] Sennovate. Blockchain for identity and access management. https://sennovate.com/blockchain-for-identity-and-access-management/, 2024.

[9] Coursera Staff. Understanding blockchain technology: How it works, uses, and benefits. https://www.coursera.org/articles/blockchain-technology?utm_medium=sem&utm_source=gg&utm_campaign=B2C_EMEA__coursera_FTCOF_career-academy_pmax-multiple-audiences-country-multi-set2&campaignid=20882109092&adgroupid=&device=c&keyword=&matchtype=&network=x&d, 2024.

[10] Coursera Staff. What is defi? a guide to decentralized finance. https://www.coursera.org/articles/what-is-defi?utm_medium=sem&utm_source=gg&utm_campaign=B2C_EMEA__coursera_FTCOF_career-academy_pmax-multiple-audiences-country-multi-set2&campaignid=20882109092&adgroupid=&device=c&keyword=&matchtype=&network=x&devicemode, 2024.

[11] Coursera Staff. What is web3? (+ how does it work?). https://www.coursera.org/articles/web-three?utm_medium=sem&utm_source=gg&utm_campaign=B2C_EMEA__coursera_FTCOF_career-academy_pmax-multiple-audiences-country-multi-set2&campaignid=20882109092&adgroupid=&device=c&keyword=&matchtype=&network=x&devicemodel=&, 2024.

[12] Itai Turbahn. What is wallet-based authentication? https://www.dynamic.xyz/blog/web3-auth-101, 2023.

[13] VSC. Using react in visual studio code. https://code.visualstudio.com/docs/nodejs/reactjs-tutorial, 2024.

[14] webdesignsuli.hu. Html 5 alapismeretek. https://labatlani-aranyisi.hu/wp-content/uploads/2019/10/mindent-a-html-r%C5%91l.pdf#:~:text=Mi%20is%20az%20a%20HTML%3F%20HTML%20%28HyperText%20Markup,megeml%C3%ADteni%2C%20hogy%20le%C3%ADr%C3%B3%20%C3%A9s%20nem%20pedig%20programoz%C3%A1si%20nyelv, 2014.

[15] WixEncyclopediat. What is javascript? https://www.wix.com/encyclopedia/definition/javascript?utm_source=google&utm_medium=cpc&utm_campaign=21361713794^163727006776^search%20-%20dsa&experiment_id=p1617977759^&gad_source=1&gclid=CjwKCAiAxKy5BhBbEiwAYiW--yXmO1Mcuw_DPV0NjjeS8yJQ_4ArqLx52oVyBZT, 2024.

CD Használati útmutató

Az adathordozó tartalma a szakdolgozatom teljes forráskódját, illetve a kész .pdf fájlt, „dolgozat.pdf” néven.

Az alkalmazás futtatása a következő lépéseket igényli:

- Csomagolja ki a .zip fájlt!
- Telepítse a Visual Studio Code legfrissebb verzióját, és a Node.js legfrissebb verzióját
- Töltse le a használt böngészőhöz a MetaMask bővítmény legfrissebb verzióját
- Ha feltelepült parancssorban a fontend mappában belüli bursch _coin mappába navigálva futtatjuk az „npm install” parancsot, ha feltelepült, a backend mappához navigálunk és ismét az „npm install” paranccsal telepítjük a megfelelő csomagokat.
- Ezután a backend mappában maradva „npm start” paranccsal elindítjuk a szervert. Ha ez megtörtént ismét a fontend mappában belüli bursch_coin mappába navigálunk és futtatjuk az „npm start” paranccsal a felületet.