

Algorand útikalauz stopposoknak

Írta Lipovszki Gábor

2024. dec. 05. 11:54

Tartalomjegyzék

1. Bevezetés	4
1.1. Néhány szó az Algorandról	4
1.2. Erőforrások az interneten	5
1.2.1. Youtube videók fejlesztőknek	5
1.3. Algorand pénztárcák	6
1.4. Az Algorand blokklánc csomópontjai	6
1.5. Interakció az Algorand blokklánccal	7
1.6. SDK-k	8
1.7. Az Algorand fejlesztő eszközei	8
2. AlgoKit installálás	10
2.1. AlgoKit installálás <code>codespaces</code> alatt	10
2.2. AlgoKit installálás Windows alatt	11
2.2.1. Windows verzió ellenőrzés	11
2.2.2. VS Code installálás	11
2.2.3. Git installálás	11
2.2.4. WSL installálás	11
2.2.5. Docker Desktop installálás	12
2.2.6. Node.js installálás	12
2.2.7. Python installálás	13
2.2.8. <code>pipx</code> installálás	13
2.2.9. AlgoKit installálás	13
2.3. Az AlgoKit használata	13
3. Néhány példa az Algorand parancsok használatára	16
4. Néhány példa a Python SDK használatára	19
4.1. Előkészületi lépések	19
4.2. Algorand számla létrehozása	20
4.3. Számla egyenleg kiírása	21
4.4. Fizetési tranzakció kiadása	22
4.5. Számlaegyenlegek kiírása	24
4.6. Pénzeszköz létrehozása	26

4.7.	Pénzeszközbe történő „benevezés”, opt-in	29
4.8.	Csere atomi tranzakciós csoporttal	30
5.	Fejlesztés a Javascript SDK-val	36
5.1.	Előkészületi lépések	36
5.2.	Algorand számlaszám létrehozása	36
5.3.	Számla egyenleg kiírása	37
5.4.	Fizetési tranzakció kiadása	39
5.5.	Számlaegyenlegek kiírása	41
5.6.	Pénzeszköz létrehozása	44
5.7.	Pénzeszközbe történő „benevezés”, opt-in	47
5.8.	Csere atomi tranzakciós csoporttal	50
5.9.	Node.js Web-es alkalmazások	55
5.9.1.	Pera WalletConnect mintapélda	55
5.10.	Közvetlenül a Web böngészőre támaszkodó alkalmazásfejlesztés	64
5.10.1.	A @perawallet/connect összepakolása	64
5.10.2.	Csocsó eredmények rögzítése az Algorand Testnet blokkláncon	67
5.11.	Az első rész összefoglalása	73
6.	A TEAL	74
6.1.	A TEAL program elemei	74
6.2.	Az AVM felépítése (architektúrája)	74
6.3.	AVM adattípusok	75
6.3.1.	Alulcsordulás és túlcsordulás kezelés	75
6.4.	Algorand Smart Signatures	77
6.5.	Példa az ASS használatára	77
6.6.	Algorand alkalmazások	77
6.6.1.	A bytekód maximális mérete	77
6.6.2.	A bytekód maximális végrehajtási költsége	77
6.7.	Példa az Algorand alkalmazások használatára: “Hello, World”	77
6.7.1.	A backend	80
6.7.2.	A frontend	85
6.7.3.	Az ABI	86
6.7.4.	“HelloWorld” mintaprogram ABI használata nélkül	86
6.7.5.	“HelloWorld” mintaprogram ABI használatával	87
7.	Esettanulmány: tulajdonrész opciós vételi jog	91
7.1.	A feladat leírása	91
7.2.	A feladat elvi megoldása	91
7.3.	Az okos szerződés magyarázata	92
7.3.1.	Az okos szerződés osztályának definiálása	92
7.3.2.	A globális állapotváltozók definiálása	93
7.3.3.	createApplication – az okos szerződés létrehozása után meghívva	93
7.3.4.	bootstrap – kezdeti paraméterek beállítása	93
7.3.5.	A globális állapotváltozók visszaolvasása	95
7.3.6.	buyAsset – zseton vétele	96

7.3.7.	sendAlgosToCreator – zseton vételár visszaküldés	98
7.3.8.	clawback – zseton visszavétele	98
7.3.9.	clawbackNoIncAmount – zseton bevonása	99
7.3.10.	deleteAsset – az ASA törlése	99
7.3.11.	deleteApplication – az okos szerződés törlése előtt meghívva	99
7.4.	Az okos szerződés tesztelése	100
7.4.1.	Az egyes Jest tesztek előtti beállítások	101
7.4.2.	bootstrap teszt	103
7.4.3.	getAppVersion teszt	103
7.4.4.	getAppCreatorAddress teszt	103
7.4.5.	getAssetAmountInitial teszt	104
7.4.6.	getAssetAmount teszt	104
7.4.7.	getAssetPrice teszt	104
7.4.8.	getAssetId teszt	104
7.4.9.	getSellPeriodEnd teszt	104
7.4.10.	getGlobalState teszt	105
7.4.11.	opt in to asset teszt	106
7.4.12.	buyAsset teszt	106
7.4.13.	buyAsset 2nd time teszt	108
7.4.14.	sendAlgosToCreator teszt	109
7.4.15.	clawback teszt	109
7.4.16.	buyAsset after clawback teszt	109
7.4.17.	clawback again teszt	110
7.4.18.	'opt out buyer from asset' teszt	110
7.4.19.	'deleteAsset' teszt	110
7.4.20.	'deleteApplication' teszt	110
7.5.	A tesztek futtatása	111
7.6.	A frontend	119
7.6.1.	A frontend oldali npm scriptek	119
7.6.2.	A frontend fejlesztés menete	120
7.6.3.	Algorand specifikus részek	121
8.	Algorit update, 2024. október 29.	125
8.1.	git update	125
8.2.	Docker Desktop update	125
8.3.	node.js update	126
8.4.	Python update	127
8.5.	pipx update	127
8.6.	algorit újrainstallálás	127
8.7.	Utómunkák	128

1. Bevezetés

Ez az Útikalauz segít eligazodni a „stopposoknak”, azaz a fejlesztőknek az Algorand blokklánc kacs-karingós, zegzugos világában. Nem csak elméleti áttekintést nyújt, hanem példákkal is szemlélteti az egyes jelenségeket.

Ne ess pánikba! – ahogy azt az eredeti Útikalauz mondja.

Messze kerüld el! – a feleségem véleménye az Algorandról.

Aki tudja, csinálja, aki nem tudja, tanítja – magyar közmondás

1.1. Néhány szó az Algorandról

Az Algorand-ot Silvio Micali alkotta. Az motiválta, hogy a blokklánc új blokkjai minimális erőforrás-használattal jöjjenek létre, ugyanakkor a biztonság se szenvedjen csorbát. Ennek érdekében egy új kriptográfiai primitívet alkotott, ez a Verifiable Random Function (VRF). Ezzel a konszenzus úgy teremthető meg, hogy az egyes fázisokban a bizottsági tagokat a VRF segítségével, véletlenszerű sorsolással választják ki. Minden lekötött, „staked” Algó egy szavazatot jelent a sorsolás során. Ugyanakkor a VRF-fel utólag ellenőrizhető, hogy jogos volt-e az a fajta véletlen választás, ami bekövetkezett.

Az Algorand elnevezés is Micali professzornak köszönhető: az algorithmic és a random szavak összetételéből született. Összesen 10 milliárd Algó van, ezek a hálózat létrehozásakor egyszerre mind létrejöttek. A kisbefektetők a különféle tőzsdéken tudnak Algót venni. Jelenleg kb. 8,3 milliárd Algó van forgalomban, a maradék Algorand pénzmennyiségből az Algorand Foundation különféle programokat finanszíroz – pl. az öngazgatást (governance), fejlesztéseket, DeFi (Decentralised Finance) támogatást stb.

Az Algorand hálózat 2019-ben kezdte meg a működését. Jelenleg (2024 októberében) kb. 7500 tranzakció/sec az elérhető maximális sebessége, és kb. 2,9 sec alatt válik egy tranzakció véglegessé. A blokklánc elágazás valószínűsége rendkívül kicsi, 10^{-18} , vagyis elágazás évmilliárdok alatt fordulhat csupán elő. A tranzakciós díj jelenleg 0,001 Algó, ami a jelenlegi Algó árfolyam mellett kb. 10 fillér.

A kiváló műszaki paraméterek ellenére a CoinCodex szerint az Algorand a tőkésítettségi mutató szempontjából jelenleg (2024 októberében) a hetvenkettedik helyen áll a digitális pénzek rangsorában. A közösség nem díjazta, hogy az Algorand relé hálózatát lényegében „csókosok” üzemeltetik a jelen pillanatban is, és akiknek az Algorand Foundation évente súlyos dollármilliókat fizet.

Az Algorand Foundation főmérnöke az idén, 2024-ben oly módon szeretne ezen változtatni, hogy az új blokkok előállításáért pénz (Algó) járjon, vagyis egyfajta „bányászat” legyen bevezetve a hálózatban. Azt reméli, hogy a relé hálózat fokozatosan leállítható lesz, és az Algorand önfenntartó hálózattá válik. Mivel a bányászat az Algó lekötésével végezhető („proof of stake”, „kockázati bizonyíték” alapján), a blokkok előállításának jutalmazásától azt várja, hogy a jelenlegi 15% körüli tőkelekötés legalább 25%-ra nő az idén, továbbá azt, hogy az Algorand bekerül az első 20 digitális pénz közé.

Mivel a tranzakciós díjak rendkívül alacsonyak, és ezt szeretnék a jövőben is alacsonyan tartani, az első 2–3 évben a tranzakciós díjából adódó bányászati jutalmat az Algorand Foundation egészítené ki. A főmérnök reményei szerint az Algorand árfolyamnövekedése, a hálózat tranzakciószámának

növekedése, valamint a tranzakciós díj észszerű növelése a bányász folyamatot az első 2–3 év után önfenntartóvá fogja tenni.

Az Algorand Foundation idei, 2024-es fejlesztési tervei (roadmap) a következőket tartalmazzák:

- dinamikus blokk idő bevezetése, vagyis a bizottsági tagok leglassabb 5%-ának kihagyása a konszenzus teremtés során, ami további blokk idő csökkentést tesz lehetővé,
- az AlgoKit 2.0-ban az Algorand szerződéseket natív Python nyelven lehet megírni, ami megszünteti majd a fejlesztői szűk keresztmetszetet,
- a nem archív relé csomópontok bevezetésével a tárolt adatmennyiség csökkenthető,
- új blokkok előállításnak jutalmazásával, a „bányászattal” a lekötött Algó mennyisége növelhető,
- a relé típusú hálózati topológiáról a P2P (peer to peer) gossip hálózati topológiára történő fokozatos átállással a Bitcoinhoz és a többi digitális pénzhez hasonló rendszer kialakítása.

1.2. Erőforrások az interneten

- Az Algorand Technologies (korábban Algorand, Inc.) honlapja.
- Az Algorand Foundation honlapja
- Az Algorand Fórum
- Meghívó az Algorand Discord szerverére
- Algorand linkgyűjtemény: Awesome Algo
- Algorand blokklánc vizsgálók: lora, allo', pera
- Magyar honlap: Algorand.hu

1.2.1. Youtube videók fejlesztőknek

A fejlesztőknek szóló Youtube videók a <https://youtube.com/@algodevs> címen érhetőek el. A „Lejátszási listák” fülre kattintva megjelennek az egyes témákhoz tartozó videók. Néhány ezek közül:

- Algorand Development Environment Setup (fejlesztői környezet létrehozása)
- AVM Explained (az Algorand Virtuális Gép ismertetése)
- PyTeal Tutorial for Beginners (Full Course) (PyTeal tanfolyam)
- Beaker for Beginners (Full Course) (Beaker tanfolyam)
- TealScript for Beginners (Full Course) (TealScript tanfolyam)
- The Differences Between Ethereum & Algorand (Az Ethereum és az Algorand közötti különbségek)
- Bonus Content: Algorand Foundation CTO John Woods (Bónusz: Interjúk az Algorand Foundation főmérnökével, John Woods-szal)
- Beginner Algorand Bootcamp [April 2023] (Algorand képzés kezdőknek, 2023. április)

1.3. Algorand pénztárcák

A felhasználó az Algorand pénztárcán keresztül kerül közvetlen kapcsolatba az Algorand blokkláncal. A felhasználói élményt alapvetően meghatározza a pénztárca minősége, használhatósága.

A pénztárca szoftver vagy hardver pénztárca lehet. A hardver pénztárcák jellemzője, hogy a titkos kulcsot egy kriptográfiailag biztonságos memóriában tárolják, amelyből az adat közvetlenül nem olvasható ki. A hardver pénztárcák a tranzakciók aláírását egy külön célhardverrel végzik, amelyen a felhasználó explicit beavatkozása (gombnyomás) szükséges egy tranzakció aláírásához.

Az egyik legismertebb hardver pénztárca gyártó a Ledger. A Ledger Live alkalmazás a Ledger hardver pénztárcához tartozó kezelőfelület. A Ledger hardver pénztárcával sokféle digitális pénz kezelhető, többek között az Algorand is.

A hivatalos Algorand pénztárca a Pera pénztárca. A Pera kétféle pénztárcát ajánl: web-es felületű pénztárcát és iOS ill. Android alkalmazással megvalósított pénztárcát. A web-es pénztárca biztonsági okok miatt kivezetésre került, jelenleg már csak számlák egyenlege figyelhető meg vele. Az iOS és az Android alkalmazások az App Store-ból vagy a Google Play-ről tölthetők le.

Az Algorand blokkláncnak számos olyan jellemzője van, melyet a hivatalos Pera pénztárca még nem támogat, vagy csak részben támogat. Néhány ilyen jellemző:

- a Ledger Nano X hardver pénztárca csak USB-vel nem használható,
- a Pera egyelőre nem támogatja multisig számlákat,
- a Pera pénztárcában nem kezelhetők közvetlenül az Algorand alkalmazások,
- a Pera pénztárca mentése nem túl kényelmes.

Egy másik pénztárca az A-Wallet, amely egy web-es pénztárca. Az A-Wallet támogatja a Ledger HW pénztárcák használatát. Sok érdekes jellemzője van, pl. a multisig (többszörös aláírás) támogatása, a Google Authenticator használata a kétfaktoros aláírás támogatásához, a payment gateway támogatása, szavazás támogatása a blokkláncban stb.

1.4. Az Algorand blokklánc csomópontjai

Ha szeretnénk egy olyan alkalmazást fejleszteni, amely az Algorand blokkláncot használja, akkor ehhez az alkalmazásból el kell tudnunk érni az Algorand valamelyik publikus hálózatát, vagy saját (privát) hálózati csomópontot kell létrehozunk. Az Algorand blokkláncban többféle hálózati csomópont lehetséges:

- relé csomópontok, melyek gyors hálózati forgalmat biztosítják
- archív csomópontok, amelyek a teljes Algorand blokklánc történetét tárolják. A blokkláncban való gyors kereshetőséget egy Indexer-nek nevezett PostgreSQL adatbázis kezelő biztosítja.
- nem archív csomópontok, amelyek helytakarékoság miatt mindegyik Algorand számlára vonatkozóan csak az utolsó 1000 tranzakciót tárolják.

Az archív csomópontok elérése szabványos REST API felületen keresztül lehetséges. Például a nodely.io ingyenesen használható Algorand végpontjai a playground linken keresztül biztosítják, hogy kipróbálhassuk a REST API parancsainak a működését. Az alacsony szintű curl parancsok

használata helyett a fejlesztők SDK-kon (System Development Kit, Rendszerfejlesztő csomag) keresztül tudják kényelmesen elérni az egyes REST API funkciókat.

A fejlesztő dönthet úgy, hogy saját hálózati csomópontot üzemeltet. A saját csomópont archív vagy nem archív csomópont lehet. A csomópontok üzemeltetésével kapcsolatos HW előfeltételek leírása az előbbi hiperhivatkozásnál található meg.

Saját csomópont Linux vagy Docker alatt telepíthető. Linux alatt vannak bináris telepítő készletek, és csomagba szervezett telepítő készletek.

A telepített Algorand csomópont alkotó elemei:

- `algod`, Algorand démon, amely a blokklánc kezelését végzi
- `kmd`, Key Management démon, amely a pénztárca kezelését végzi
- `indexer`, Indexer, a blokklánc adatok gyors elérését biztosító adatbázis
- `conduit`, az Algorand blokklánc adatainak eljuttatása külső alkalmazások számára
- parancssorban használható programok, pl. `goal`, `kmd`

Az Algorand Foundation a fejlesztők munkájának megkönnyítése érdekében kifejlesztette az AlgoKit-et, amely Docker felület alá telepít egy Algorand csomópontot, és a fejlesztést elősegítő környezetet.

A fejlesztés során használható Algorand blokklánc fajták:

- `privát` blokklánc, egyetlen csomópontja a fejlesztői gépen van.
- `betanet`, teszt blokklánc a legelső próbákhoz. Saját betanet teszt Algó-t használ, mely ingyenesen letölthető a <https://bank.betanet.algodev.network/> címről.
- `testnet`, teszt blokklánc, amely nagyon hasonlít az éles blokkláncra. Saját teszt Algó-t használ, amely ingyenesen letölthető a <https://bank.testnet.algorand.network/> címről.
- `mainnet`, éles blokklánc

1.5. Interakció az Algorand blokkláncsal

- Parancsnyelv: Ha saját csomópontot üzemeltetünk, akkor az `algod`, `kmd` és `indexer` processzek kezelése shell parancsokkal lehetséges, pl. a `goal` és a `kmd` parancsok a leginkább használtak.
- REST API: A saját csomópontunk szolgáltatásai REST API felületen keresztül is elérhetőek: a `goal`, `kmd` és `indexer` processzek egy-egy REST API felületet is biztosítanak. Ezen szolgáltatások eléréshez API kulcsokra van szükség. Ha publikáljuk ezeket a kulcsokat, akkor a csomópontunkat mások is használhatják.
- SDK-k: A REST API felületeket az Algorand SDK-k „csomagolják be”, vagyis a különböző nyelvi környezetekből (JavaScript, PHP, Java, Rust stb.) kényelmes hozzáférést biztosítanak a processzek REST API felületéhez.

Saját csomópont üzemeltetése többféleképpen lehetséges:

- Linux alatt Algorand blokklánc csomópont üzembe helyezésével, lásd `Install a node`, ezen belül az `update script` használatával lásd `Installing on Linux`

- Linux alatt privát Algorand blokklánc csomópont üzembe helyezésével, lásd [Create a private network](#)
- Docker (és Docker Compose) alatt: Algorand Sandbox üzembe helyezésével, lásd [Algorand Sandbox](#)

1.6. SDK-k

Az Algorand blokklánc a különféle nyelvi környezetekből az SDK-k segítségével érhető el. Az Algorand által biztosított „hivatalos” SDK-k a következők:

- Javascript Algorand SDK
- Python Algorand SDK
- Go Algorand SDK
- Java Algorand SDK

A fejlesztői közösség által készített további SDK-k a következők:

- PHP Algorand SDK
- .NET Algorand SDK
- Rust SDK
- Swift SDK
- Unity SDK

1.7. Az Algorand fejlesztő eszközei

Az Algorand fejlesztő eszközei a hagyományokhoz hasonlóan számos rétegben érhetők el. Ennek részben történeti okai vannak. Az egyes szintek a következők:

- bináris szint, az Algorand Virtuális Gép (Algorand Virtual Machine, a továbbiakban AVM) utasításainak a szintje. Az Algorand fejlesztői dokumentáció részletesen leírja az AVM architektúráját és az egyes utasítások működését, lásd [v10 opkódok](#).
- assembly szint: a TEAL assembly szintű programozási nyelv. (A TEAL a Transaction Execution and Approval Language rövidítése.) A TEAL nyelven megírt assembly programot a TEAL compiler fordítja le az Algorand Virtuális Gép bináris kódjává. A TEAL compiler mind a parancsnyelvi felületen, mind az SDK-kon keresztül elérhető.
- compiler szint: a PyTeal compiler egy Python nyelvben megírt „pszeudo”-Python kódot fordít TEAL-re. A használhatóságát megnehezíti, hogy keveredik benne a normál Python kód és PyTeal fordítónak szóló „pszeudo”-Python kód. A TEAL-hez képest sokkal áttekinthetőbb kód írható benne a vezérlési struktúrák használhatósága miatt.
- compiler szint: a Beaker compiler a PyTeal továbbfejlesztett változata, amely Python objektumként tesztelhető az Algorand blokkláncban használható alkalmazást. Az Algorand blokkláncban az alkalmazás nagyjából az Ethereum szerződésnek felel meg, de számos korlátozással biztosították a blokklánc fejlesztők, hogy az Algorand blokklánc működése gyors maradjon. Mivel az Algorand blokklánc alkalmazások csak 1-2 évvel az „alap” AVM után jelentek

meg, így a Beaker is szükségszerűen későbbi „találmány”.

- compiler szint: a PuyaPy Python -> TEAL compiler, amely tisztán Python nyelvről fordít az Algorand Virtuális Gép assembly nyelvi szintjére.
- compiler szint: a TealScript fordítóprogram a TypeScript nyelv egy részhalmazából fordít TEAL szintre.

A fejlesztő eszközök keretrendszerekbe vannak szervezve. Az AlgoKit az egyik legismertebb. Az AlgoKit biztosítja a fejlesztő számára:

- az Algorand blokklánc elérését. A blokklánc lehet privát, vagy a Betanet, Testnet, Mainnet valamelyike. A blokklánc elérése Docker és Docker Compose segítségével, az Algorand Sandbox segítségével történik.
- a PyTeal, Beaker, PuyaPy fejlesztő eszközök elérését. A fejlesztő eszközök számára virtuális környezetet teremt, és a függőségeket is kezeli.
- a TealScript fejlesztő eszköz elérését
- mintapéldákat
- automatikus felhasználói felület generálást az Algorand alkalmazás alapján

Blokklánc vizsgálóként az AlgoKit mellett a LORA ajánlott, mert képes nem csak a Betanet, Testnet, Mainnet tranzakciónak megjelenítésére, hanem az Algorand Sandbox által kezelt privát blokklánc tranzakcióinak megjelenítésére is.

Egy másik keretrendszer a TealCraft keretrendszer, amely lehetővé teszi a TealScript böngészőből történő közvetlen használatát.

2. AlgoKit installálás

Az AlgoKit az Algorand Foundation által kifejlesztett keretrendszer, amellyel biztosítható:

- egy lokális (privát) hálózat felállítása, indítása, leállítása, vagy kapcsolódás az Algorand Beta-net, Testnet vagy Mainnet hálózatához, lásd `algokit local` parancsok
- fejlesztői keretrendszerek inicializálása, példák betöltése, lásd `algokit init` parancs
- virtuális környezet megteremtése a fejlesztéshez, lásd `algokit bootstrap all` parancs

Az AlgoKit installálásának előfeltételei Windows 10 vagy Windows 11 operációs rendszer alatt:

- VS Code (Visual Studio Code) installálása
- Git installálása
- WSL2 (Windows Subsystem for Linux) installálása
- Docker Desktop installálása
- Python installálása
- pipx installálása

Az installálás lépéseiről egy jó összefoglaló található a YouTube-on, lásd Ryan Fox Setup Your Algorand Development Environment on Windows in 10 Minutes (Hozd létre Windows-on az Algorand fejlesztői környezetet 10 perc alatt) című videóját.

A Github alatt regisztrált felhasználók ingyen havi 120 óra virtualizált környezethez jutnak a codespaces alatt. Ez a környezet web böngészőből használható. Az AlgoKit ebben a környezetben minimális számú lépéssel installálható, mivel a `codespaces` automatikusan biztosítja a következő komponensek használatát:

- `vscode` szövegszerkesztő
- `git` verziókezelő
- Docker és Docker Composer
- Node.js

2.1. AlgoKit installálás codespaces alatt

A szükséges lépések:

- Lépünk be a github.com alatti számlánkba
- Készítünk egy új repository-t
- Menjünk az új repository-ba. Nyomjuk meg a `Code` gombot, válasszuk a `Codespaces` tab-ot.
- Nyomjuk meg a “Create codespace on main” gombot. Létrejön egy új fejlesztői környezet, VS Code szerkesztővel.
- Ha nem jelenik meg terminál fül, akkor kattintsunk a `≡ -ra`, majd válasszuk a `Terminal | New terminal` menüpontokat.

- A terminál fülön adjuk ki: `pipx install algokit`
- A terminál fülön adjuk ki: `algokit localnet start`

Egy-két perc alatt installálódik docker alatt egy Algorand lokális hálózati csomópont.

Készen is vagyunk! Ugorjunk a következő részhez, amely néhány példával az AlgoKit használatát mutatja be!

2.2. AlgoKit installálás Windows alatt

2.2.1. Windows verzió ellenőrzés

Menjünk a [Vezérlőpult | Rendszer](#) lapra. Ellenőrizzük az OS verziószámát. A minimális támogatott verzió: 19045.¹

2.2.2. VS Code installálás

Menjünk a VS Code Download weblapra. Nyomjuk meg a “Download, Windows 10, Windows 11” gombot. Indítsuk el a letöltött fájlt. Indítsuk el a VS Code-ot.

2.2.3. Git installálás

A VS Code-ban a 'View | Terminal' menüelemekre kattintva indítsunk egy terminál ablakot. A terminál ablakban adjuk ki:

```
git --version
```

Ha hibaüzenetet kapunk, akkor adjuk ki:

```
winget install --exact --id Git.Git
```

Ellenőrizzük, hogy sikerült-e az installálás:

```
git --version
# => git version 2.44.0.windows.1
```

2.2.4. WSL installálás

A Windows Subsystem for Linux installálására azért van szükség, mert a Docker Desktop a Linux container-eket ennek a segítségével futtatja.

A VS Code terminál ablakában adjuk ki:

```
wsl -l -v
```

Ha hibaüzenetet kapunk, akkor adjuk ki:

```
wsl --install
```

Amikor megjelenik egy új terminál ablak, adjuk meg az új UNIX felhasználónevünket és jelszavunkat:

```
1 Enter new UNIX username: fel használ ónév
2 New password: j el szó
```

¹ A docker.desktop 4.35.0 által megkövetelt minimális Windows verzió 19045 vagy kés bbi

3 Retype password: `j el szó új ra`

Írjuk be a start sorba: **Ubuntu**, és indítsuk el az Ubuntu alkalmazást.

VS Code terminál ablakban adjuk ki ismét:

```
wsl -l -v
```

A következő fog megjelenni:

```
1  NAME      STATE      VERSION
2  * Ubuntu   Running    2
```

2.2.5. Docker Desktop installálás

A VS Code terminál ablakában adjuk ki:

```
docker version
```

Ha hibaüzenetet kapunk, akkor adjuk ki:

```
winget install --exact --id Docker.DockerDesktop
```

Az installálás befejeződése után indítsuk újra a Windows-t.

A Docker Desktop ikonra kattintva indítsuk el a Docker-t. Fogadjuk el a feltételeket (“Accept”). Megjegyzés: lehet, hogy az “Accept” az indító ablak mögött van „elbújva”.

A VS Code terminálablakban adjuk ki:

```
docker version
```

Most már nem kapunk hibaüzenetet.

2.2.6. Node.js installálás

A Node.js installálása a TealScript és a React frontend működéséhez szükséges. A VS Code terminálablakában adjuk ki:

```
node --version
```

Ha hibaüzenetet kapunk, akkor adjuk ki:

```
winget install --exact --id OpenJS.NodeJS
```

Ellenőrzésképpen adjuk ki a következő parancsokat:

```
node --version
```

```
# => v21.7.3
```

```
npm --version
```

```
# => 10.5.0
```

2.2.7. Python installálás

A Python installálása a PuyaPy fordítóprogram működéséhez szükséges, de a korábbi PyTeal és Beaker is Python kódot használ. A VS Code terminál ablakában adjuk ki:

```
python --version
```

Ha hibaüzenetet kapunk, akkor adjuk ki:

```
winget install python.python.3.12
```

Megjegyzés: érdemes a legújabb Python-t installálni, de minimum Python 3.11-et.

2.2.8. pipx installálás

Csukjuk be a VS Code-ot, és indítsuk el újra, hogy a VS Code lássa a path-on a python parancsot. A VS Code terminál ablakban adjuk ki:

```
pipx --version
```

Ha hibaüzenetet kapunk, akkor adjuk ki:

```
python -m pip install pipx
```

A VS Code terminál ablakban adjuk ki ismét:

```
pipx --version
```

```
# => 1.5.0
```

Most már nem kapunk hibaüzenetet.

2.2.9. AlgoKit installálás

A VS Code terminál ablakában adjuk ki:

```
pipx install algokit
```

2.3. Az AlgoKit használata

Ellenőrizzük, hogy sikerült-e az installálás. A codespaces vagy a VS Code terminál ablakában adjuk ki:

```
algokit --version
```

```
# => algokit, version 2.0.3
```

Indítsuk el az Algorand lokális hálózati környezetet:

```
algokit localnet start
```

Ellenőrizzük, hogy sikeresen elindultak-e a Docker környezet alatt a container-ek. A Windows alatt a Docker Desktop ablakban láthatók a konténer-ek és az elindított image-ek. A codespaces környezetben a

```
docker container ls
```

parancsot kell kiadnunk.

Négy image-et kell látnunk: `algorand/indexer:latest`, `algorand/conduit:latest`, `postgres:13-alpine` és `algorand/algod:latest`

Ezek a konténerek futtatják az algokit, indexer-t, a conduit-ot (ez egyfajta eseménykezelő), a postgres adatbáziskezelőt, és az algod Algorand démont.

A Docker image-ek helyes működése a következő paranccsal is ellenőrizhető:

```
algokit localnet status
```

Példa:

```
1 @A-Maugli → /workspaces (main) $ pipx install algokit
2   installed package algokit 1.13.1, installed using Python 3.10.13
3   These apps are now globally available
4     - algokit
5 done! ★ ★
6 @A-Maugli → /workspaces (main) $ algokit localnet start
7 Starting AlgoKit LocalNet now...
8 docker: conduit Pulling
9 docker: algod Pulling
10 docker: indexer-db Pulling
11 docker: indexer Pulling
12 docker: indexer Pulled
13 docker: conduit Pulled
14 docker: indexer-db Pulled
15 docker: algod Pulled
16 docker: Network algokit_sandbox_default Creating
17 docker: Network algokit_sandbox_default Created
18 docker: Container algokit_sandbox_algod Creating
19 docker: Container algokit_sandbox_postgres Creating
20 docker: Container algokit_sandbox_algod Created
21 docker: Container algokit_sandbox_postgres Created
22 docker: Container algokit_sandbox_conduit Creating
23 docker: Container algokit_sandbox_conduit Created
24 docker: Container algokit_sandbox_indexer Creating
25 docker: Container algokit_sandbox_indexer Created
26 docker: Container algokit_sandbox_algod Starting
27 docker: Container algokit_sandbox_postgres Starting
28 docker: Container algokit_sandbox_postgres Started
29 docker: Container algokit_sandbox_algod Started
30 docker: Container algokit_sandbox_conduit Starting
31 docker: Container algokit_sandbox_conduit Started
32 docker: Container algokit_sandbox_indexer Starting
33 docker: Container algokit_sandbox_indexer Started
34 docker: Container algokit_sandbox_postgres Waiting
35 docker: Container algokit_sandbox_indexer Waiting
36 docker: Container algokit_sandbox_algod Waiting
37 docker: Container algokit_sandbox_conduit Waiting
38 docker: Container algokit_sandbox_algod Healthy
39 docker: Container algokit_sandbox_postgres Healthy
40 docker: Container algokit_sandbox_indexer Healthy
41 docker: Container algokit_sandbox_conduit Healthy
42 Started; execute algokit explore to explore LocalNet in a web user interface.
43 @A-Maugli → /workspaces (main) $ docker ps
```

```

44 CONTAINER ID   IMAGE                                COMMAND                                CREATED        STATUS
    ↪ PORTS
    ↪ NAMES
45 0c7f8f8fa8ef   algorand/indexer:latest             "docker-entrypoint.s..." 5 minutes ago  Up 5 minutes
    ↪ 0.0.0.0:8980->8980/tcp, :::8980->8980/tcp
    ↪ al goki t_sandbox_i ndexer
46 2c45e5017538   algorand/condui t:latest            "docker-entrypoint.sh"     5 minutes ago  Up 5 minutes
    ↪
    ↪ al goki t_sandbox_condui t
47 0b590c17237f   postgres:13-alpi ne                 "docker-entrypoint.s..." 5 minutes ago  Up 5 minutes
    ↪ 0.0.0.0:5443->5432/tcp, :::5443->5432/tcp
    ↪ al goki t_sandbox_postgres
48 ea05aa8270b7   algorand/al god:latest              "/node/run/run.sh"         5 minutes ago  Up 5 minutes
    ↪ 4160/tcp, 9100/tcp, 0.0.0.0:9392->9392/tcp, :::9392->9392/tcp, 0.0.0.0:4002->7833/tcp,
    ↪ :::4002->7833/tcp, 0.0.0.0:4001->8080/tcp, :::4001->8080/tcp  al goki t_sandbox_al god
49 @A-Maugli → /workspaces (main) $ al goki t explore
50 Opening local net in https://app.dappflow.org using default browser
51 @A-Maugli → /workspaces (main) $ al goki t local net status
52 # al god status
53 Status: Running
54 Port: 4001
55 Last round: 0
56 Time since last round: 0.0s
57 Genesis ID: dockernet-v1
58 Genesis hash: sl0ctEZ9Qm2gqYLLQnz9+KTTi hiOgwFXCLCXtsEUobo=
59 Versi on: 3.23.1
60 # condui t status
61 Status: Running
62 # i ndexer-db status
63 Status: Running
64 # i ndexer status
65 Status: Running
66 Port: 8980
67 Last round: 0
68 Versi on: 3.4.0
69 @A-Maugli → /workspaces (main) $

```

A lokális Algorand csomópont működését egy böngészőben a `algorit explore` paranccsal tudjuk ellenőrizni. A `codespaces` környezetben ennek kiadása előtt menjünk a `Ports` fülre, és a `Visibility` oszlopban kattintsunk a jobb egér gombbal a 4001, 4002 és 8980 portoknak megfelelő helyekre, majd válasszuk a `Port visibility | Public` menüpontokat.

Ezt követően válasszuk a `Terminal` fület, és adjuk ki a `algorit explore` parancsot. A böngészőben felpattanó ablakban nyomjuk meg az OK gombot!

3. Néhány példa az Algorand parancsok használatára

A parancsnyelvi környezet a következő paranccsal érhető el:

```
algokit localnet console
```

codespaces használatakor a következő parancsok előzetes kiadására van szükség:

```
pipx install algokit
```

```
algokit localnet start
```

```
algokit localnet stop
```

```
sudo chown -R codespace:codespace /.config/algokit
```

Példa:

```
1 @A-Maugli → /workspaces (main) $ algokit localnet console
2 Opening Bash console on the algod node; execute exit to return to original console
3 root@ea05aa8270b7: ~# goal wallet list
4 #####
5 Wallet: unencrypted-default-wallet
6 ID:      b971fb4cc5463c57a8563eed3413c0de
7 #####
8 root@ea05aa8270b7: ~# goal account list
9 [online]      NMA5HTMA53EGFBS5523NGEZ3TGSGLB3VGYMP4YH22VGFEPHL7HUNI OZQM
10 → NMA5HTMA53EGFBS5523NGEZ3TGSGLB3VGYMP4YH22VGFEPHL7HUNI OZQM      4000000000000000 mi croAl gos
11 [online]      3B2D7UNBANXVZWPTI FWQZ7AC7MPPLBKCHAL5MNJPGMAGEZQQA7TDMG57E
12 → 3B2D7UNBANXVZWPTI FWQZ7AC7MPPLBKCHAL5MNJPGMAGEZQQA7TDMG57E      4000000000000000 mi croAl gos
13 [online]      40FWMGCQX6VS65NNWQAVRVL3M502HN3BPUUXTE2CR3SYZLI FVCH3K5TM4
14 → 40FWMGCQX6VS65NNWQAVRVL3M502HN3BPUUXTE2CR3SYZLI FVCH3K5TM4      2000000000000000 mi croAl gos
15 root@ea05aa8270b7: ~# goal wallet new w1
16 Please choose a password for wallet 'w1':
17 Please confirm the password:
18 Creating wallet...
19 Created wallet 'w1'
20 Your new wallet has a backup phrase that can be used for recovery.
21 Keeping this backup phrase safe is extremely important.
22 Would you like to see it now? (Y/n): y
23 Your backup phrase is printed below.
24 Keep this information safe -- never share it with anyone!
25
26 ordinary usage hockey nurse shop rebel picnic female element guitar furnace rain enforce drum
27 → metal ostrich arrow safe immune melody dog mule organ abandon cannon
28 root@ea05aa8270b7: ~# goal account new
29 Created new account with address 6BW3G56B6PNWB2CVXQNKYTDW5R5SLVEDMVEM7I2FBI2ZXRTEH7EACLAOY
30 root@ea05aa8270b7: ~# goal account list
31 [online]      NMA5HTMA53EGFBS5523NGEZ3TGSGLB3VGYMP4YH22VGFEPHL7HUNI OZQM
32 → NMA5HTMA53EGFBS5523NGEZ3TGSGLB3VGYMP4YH22VGFEPHL7HUNI OZQM      4000000000000000 mi croAl gos
33 [online]      3B2D7UNBANXVZWPTI FWQZ7AC7MPPLBKCHAL5MNJPGMAGEZQQA7TDMG57E
34 → 3B2D7UNBANXVZWPTI FWQZ7AC7MPPLBKCHAL5MNJPGMAGEZQQA7TDMG57E      4000000000000000 mi croAl gos
35 [online]      40FWMGCQX6VS65NNWQAVRVL3M502HN3BPUUXTE2CR3SYZLI FVCH3K5TM4
36 → 40FWMGCQX6VS65NNWQAVRVL3M502HN3BPUUXTE2CR3SYZLI FVCH3K5TM4      2000000000000000 mi croAl gos
37 [offline]     Unnamed-0      6BW3G56B6PNWB2CVXQNKYTDW5R5SLVEDMVEM7I2FBI2ZXRTEH7EACLAOY      0
38 → mi croAl gos      *Default
39 root@ea05aa8270b7: ~# goal account new --wallet w1
40 Please enter the password for wallet 'w1':
```



```

33 Created new account with address RTKECLZXEQLG2DRSSG2KWWHNE53UVH5F2MYLUENKKQUEOUI 4HPJQTJGXPM
34 root@ea05aa8270b7: ~# goal account list
35 [online]      NMA5HTMA53EGFBS5523NGEZ3TGSGLB3VGYMP4YH22VGFEPHL7HUNI OZQM
   ↳ NMA5HTMA53EGFBS5523NGEZ3TGSGLB3VGYMP4YH22VGFEPHL7HUNI OZQM      4000000000000000 mi croAl gos
36 [online]      3B2D7UNBANXVZWP7I FWQZ7AC7MPPLBKCWHAL5MNJPGMAGEZQQA7TDMG57E
   ↳ 3B2D7UNBANXVZWP7I FWQZ7AC7MPPLBKCWHAL5MNJPGMAGEZQQA7TDMG57E      4000000000000000 mi croAl gos
37 [online]      40FWMGCQX6VS65NNWQAVRVL3M502HN3BPUUXTE2CR3SYZLI FVCH3K5TM4
   ↳ 40FWMGCQX6VS65NNWQAVRVL3M502HN3BPUUXTE2CR3SYZLI FVCH3K5TM4      2000000000000000 mi croAl gos
38 [offline]     Unnamed-0      6BW3G56B6PNWB2CVXQNKYTDW5R5SLVEDMVE7I 2FBI Z2XRTEH7EACLAOY      0
   ↳ mi croAl gos      *Default
39 root@ea05aa8270b7: ~# goal account list --wallet w1
40 [offline]     Unnamed-1      RTKECLZXEQLG2DRSSG2KWWHNE53UVH5F2MYLUENKKQUEOUI 4HPJQTJGXPM      0
   ↳ mi croAl gos
41 root@ea05aa8270b7: ~# goal clerk send --from
   ↳ NMA5HTMA53EGFBS5523NGEZ3TGSGLB3VGYMP4YH22VGFEPHL7HUNI OZQM --to
   ↳ RTKECLZXEQLG2DRSSG2KWWHNE53UVH5F2MYLUENKKQUEOUI 4HPJQTJGXPM --amount 1000000
42 Sent 1000000 Mi croAl gos from account NMA5HTMA53EGFBS5523NGEZ3TGSGLB3VGYMP4YH22VGFEPHL7HUNI OZQM to
   ↳ address RTKECLZXEQLG2DRSSG2KWWHNE53UVH5F2MYLUENKKQUEOUI 4HPJQTJGXPM, transaction ID:
   ↳ IKASGOOYTBMHI XOL2GTWG5QWUOAXUHOFPTZRRBC43FANR3M3DZTQ. Fee set to 1000
43 Transaction IKASGOOYTBMHI XOL2GTWG5QWUOAXUHOFPTZRRBC43FANR3M3DZTQ committed in round 1
44 root@ea05aa8270b7: ~# goal account balance --address
   ↳ RTKECLZXEQLG2DRSSG2KWWHNE53UVH5F2MYLUENKKQUEOUI 4HPJQTJGXPM
45 1000000 mi croAl gos
46 root@ea05aa8270b7: ~# exit
47 exit
48 @A-Maugli → /workspaces (main) $

```

A példa magyarázata:

- 3. sor: a `goal wallet list` kilistázza a kiindulási pénztárcát (default wallet)
- 8. sor: a `goal account list` kilistázza a kiindulási pénztárcába felvett számlákat
- 12. sor: új pénztárcát a `goal wallet new` paranccsal tudunk készíteni. A pénztárca neve `w1`
- 24. sor: új számlát a `goal account new` paranccsal tudunk létrehozni. Látszik, hogy a számla a default pénztárcában jött létre.
- 31. sor: új számlát a `w1` pénztárcában a `goal account new --wallet w1` paranccsal tudunk felvenni
- 34. sor: a számlák kilistázása a kiindulási pénztárcából
- 39. sor: a számlák kilistázása a `w1` pénztárcából
- 41. sor: Algót küldeni az egyik számláról a másikra a `goal clerk send` paranccsal tudunk. Meg kell adnunk a forrás számla címét a `--from` után, a cél számla címét a `--to` után, és az elküldeni kívánt összeget, mikro-Algókban: `--amount 1000000`. Itt tehát 1 Algót küldtünk a `NMA5H...IOZQM` számláról a `RTKEC...JGXPM` számlára.
- 44. sor: egy számla egyenlegét a `goal account balance` paranccsal tudjuk megállapítani. Meg kell adnunk a számlához tartozó azonosítót is, a `--address` után.

Egy érdekesség a Bitcoin, Ethereum világából érkezőknek: a `w1` nevű új pénztárca létrehozásakor kiírtuk a "backup phrase"-t, „visszaállító kifejezést”. Figyeljük meg, hogy ez 25 szóból áll. A 25 szó egy 256 bites entrópiát kódol, és ellenőrző összeggel védi a visszaállító kifejezést. Ez a 256 bites ent-

rópia megfelel a BIP39-es szabványban szereplő 256 bites entrópiának, de **nem úgy van kódolva**, ahogyan azt a BIP39 szabvány előírja, hanem az Algorand 11 bites tömbformátumának megfelelően. A részleteket lásd a *What's the rationale behind the bespoke 25-word mnemonic standard?* (Milyen megfontolás van az állítólagos 25-szavas Algorand mnemonik szabvány mögött?) című fórum cikkben.

Természetesen semmi akadály, hogy az Algorand 25 szavas mnemonikját (jelmondatát) átalakítsuk a BIP39-ben szereplő 24 szavas mnemonikra (jelmondatra), és fordítva. A hierarchikus címkézés ebből az entrópiából történik, lásd pl. a BIP39 weblapot. Itt tehát a BIP39 szabványnak megfelelő 24 szó vagy az Algorandban használt 25 szó nem privát címet kódol, hanem 256 bites entrópiát.

Az esetleges zavart tovább fokozza, hogy lehetőség van az egyes számlákhoz tartozó titkos kulcs kiexportálására is. Ezek is 25 szavas mnemonik formájában íródnak ki. A pénztárcák esetén a 25 szó az entrópiát kódolja, és a segítségével tetszőleges számú számla állítható elő (publikus kulcs, titkos kulcs).

Igazán ínyenceknek azt is elárulhatjuk, hogy a Ledger HW pénztárca esetében viszont úgy valósították meg a HW pénztárcán futó Algorand App-ot, hogy az entrópia kódolására csak 24 szót használnak, és egy egészen más titkosító algoritmust használnak. A `algorand.oortnet.com` címen található egy helyes kis segédprogram, amely a Ledger HW pénztárca mnemonikja, azaz visszaállító kifejezése alapján képes előállítani az egyes, hierarchikus Algorand számlákhoz tartozó címeket, kulcsokat, és mnemonikokat. Ez azt jelenti, hogy ha nem férünk hozzá a Ledger HW pénztárcánkhoz (mert elromlott, ellopták stb.), akkor is hozzá tudunk férni az általa kódolt Algorand számlákhoz.

Megjegyzés: A BIP39 honlapon az Algorand pénznem nem választható ki, de a <https://github.com/Coinomi/bip39-coinomi/releases> címről letölthető egy olyan standalone (önállóan futtatható) változat, amely az Algorand esetén is működőképes.

4. Néhány példa a Python SDK használatára

A Python SDK használatát Ryan Fox “Algorand Bootcamp for Beginners” előadásain használt példákkal mutatom be. A Python mintapéldák megtalálhatók a <https://github.com/A-Maugli/akt02> repository-ban, a hellow/playground/python_api könyvtárban.

4.1. Előkészületi lépések

Indítsuk el a codespaces munkaterületünket. Mikor a munkaterület betöltődött, indítsuk el a privát blokkláncot:

```
1 @A-Maugli → /workspaces/akt02 (main) $ algokit --version
2 algokit, version 1.13.0
3 @A-Maugli → /workspaces/akt02 (main) $ algokit localnet stop
4 Stopping AlgoKit LocalNet now...
5 docker: Container algokit_sandbox_indexer Stopping
6 docker: Container algokit_sandbox_indexer Stopped
7 docker: Container algokit_sandbox_conduit Stopping
8 docker: Container algokit_sandbox_conduit Stopped
9 LocalNet Stopped; execute algokit localnet start to start it again.
10 @A-Maugli → /workspaces/akt02 (main) $ algokit localnet start
11 algokit has a new version available, run algokit localnet reset --update to get the latest version
12 Starting AlgoKit LocalNet now...
13 docker: Container algokit_sandbox_algod Creating
14 docker: Container algokit_sandbox_postgres Creating
15 docker: Container algokit_sandbox_algod Created
16 docker: Container algokit_sandbox_postgres Created
17 docker: Container algokit_sandbox_conduit Created
18 docker: Container algokit_sandbox_indexer Created
19 docker: Container algokit_sandbox_algod Starting
20 docker: Container algokit_sandbox_postgres Starting
21 docker: Container algokit_sandbox_postgres Started
22 docker: Container algokit_sandbox_algod Started
23 docker: Container algokit_sandbox_conduit Starting
24 docker: Container algokit_sandbox_conduit Started
25 docker: Container algokit_sandbox_indexer Starting
26 docker: Container algokit_sandbox_indexer Started
27 docker: Container algokit_sandbox_algod Waiting
28 docker: Container algokit_sandbox_conduit Waiting
29 docker: Container algokit_sandbox_postgres Waiting
30 docker: Container algokit_sandbox_indexer Waiting
31 docker: Container algokit_sandbox_algod Healthy
32 docker: Container algokit_sandbox_postgres Healthy
33 docker: Container algokit_sandbox_conduit Healthy
34 docker: Container algokit_sandbox_indexer Healthy
35 Started; execute algokit explore to explore LocalNet in a web user interface.
```

Installáljuk a projekthez tartozó függőségeket az `algokit bootstrap all` paranccsal:

```
1 @A-Maugli → /workspaces/akt02 (main) $ algokit bootstrap all
2 Poetry not found; attempting to install it...
3 ? We couldn't find poetry ; can we install it for you via pipx so we can install Python
  → dependencies? Yes
4 Installing Python dependencies and setting up Python virtual environment via Poetry
5 poetry: Creating virtualenv playground in /workspaces/akt02/hellow/.venv
```

```

6 poetry: Installing dependencies from lock file
7 poetry:
8 poetry: Package operations: 25 installs, 0 updates, 0 removals
9 poetry:
10 poetry: - Installing exceptiongroup (1.2.0)
11 poetry: - Installing idna (3.6)
12 poetry: - Installing pycparser (2.21)
13 poetry: - Installing sniffio (1.3.0)
14 poetry: - Installing typing-extensions (4.9.0)
15 poetry: - Installing anyio (4.3.0)
16 poetry: - Installing certifi (2024.2.2)
17 poetry: - Installing cffi (1.16.0)
18 poetry: - Installing h11 (0.14.0)
19 poetry: - Installing httpcore (0.16.3)
20 poetry: - Installing msgpack (1.0.7)
21 poetry: - Installing pycryptodomex (3.20.0)
22 poetry: - Installing pynacl (1.5.0)
23 poetry: - Installing rfc3986 (1.5.0)
24 poetry: - Installing wrapt (1.16.0)
25 poetry: - Installing deprecated (1.2.14)
26 poetry: - Installing docstring-parser (0.14.1)
27 poetry: - Installing executing (1.2.0)
28 poetry: - Installing httpx (0.23.3)
29 poetry: - Installing py-algorand-sdk (2.5.0)
30 poetry: - Installing semantic-version (2.10.0)
31 poetry: - Installing tabulate (0.9.0)
32 poetry: - Installing algokit-utils (2.2.1)
33 poetry: - Installing pyteal (0.24.1)
34 poetry: - Installing beaker-pyteal (1.1.1)
35 poetry:
36 poetry: Installing the current project: playground (0.1.0)
37 Finished bootstrapping /workspaces/akt02

```

A Ports fülön tegyük globálissá a portokat. Ehhez a lakat szimbólumra kell kattintanunk az egérrel. A lakat ennek hatására kinyílik.

Aktiváljuk a virtuális környezetet. Készítsünk a playground könyvtáron belül egy python_api könyvtárat a grafikus felület segítségével, majd menjünk ebbe a könyvtárba.

```

1 @A-Maugli → /workspaces/akt02 (main) $ cd hello
2 @A-Maugli → /workspaces/akt02/hello (main) $ source .venv/bin/activate
3 (playground-py3.10) @A-Maugli → /workspaces/akt02/hello (main) $ cd playground
4 (playground-py3.10) @A-Maugli → ../akt02/hello/playground (main) $ mkdir python_api
5 (playground-py3.10) @A-Maugli → ../akt02/hello/playground (main) $ cd python_api
6 (playground-py3.10) @A-Maugli → ../akt02/hello/playground/python_api (main) $

```

4.2. Algorand számla létrehozása

A fájl explorer-rel vagy a terminálban a nano 01-account_generation.py segítségével hozzuk létre a következő fájlt:

```

1 from algosdk import account, mnemonic
2
3 def generate_algorand_keypair():
4     private_key, address = account.generate_account()

```

```

5     print("My address: {}".format(address))
6     print("My private key: {}".format(private_key))
7     print("My passphrase: {}".format(mnemonic.from_private_key(private_key)))
8
9     generate_algorand_keypair()

```

Futtassuk a `01-make_account.py` fájlt:

```

1 (pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/python_api (mai n) $ python
   → 01-account_generation.py
2 My address: 3API2XI BSH71 KMSZ3SNRGO1 NI SSNXVAYT20V6BLL4GBZRAVWTMYAWCXMPI
3 My private key:
   → y1mWx1K1ZMGbHojwGOSWkTMTs0XuCI KOATJ9PFc2AyrYHo1dAZH+hTJZ3JsTOQ1EpNvUGJ6dXwVr4YOYgrabMA==
4 My passphrase: defense floor glow festival siren utility visit marine lawn away enroll crawl
   → chicken holiday impulse angry space alert october sick purpose snow exotic ability rather

```

Gratulálok! Sikerült egy Python programból egy számlaszámot és a hozzá tartozó privát kulcsot létrehozni, valamint a privát (titkos) kulcsot 25 szavas mnemonikus formában kiírni!

4.3. Számla egyenleg kiírása

A fájl explorer-rel vagy a terminálban a `nano 02-account_balance.py` segítségével hozzuk létre a következő fájlt:

```

1 from algosdk import kmd
2 from algosdk.wallet import Wallet
3 from algosdk.v2client import Algod
4 import json
5
6 # define sandbox values for kmd client
7 kmd_address = "http://localhost:4002"
8 kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
9
10 # define sandbox values for algod client
11 algod_address = "http://localhost:4001"
12 algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
13
14 def main() :
15     # create KMDClient
16     kmd_client = kmd.KMDClient(kmd_token, kmd_address)
17
18     # connect to default wallet
19     wallet = Wallet("unencrypted-default-wallet", "", kmd_client)
20
21     # gather the three default accounts
22     wallet_addresses = wallet.list_keys()
23     addr1 = wallet_addresses[0]
24     addr2 = wallet_addresses[1]
25     addr3 = wallet_addresses[2]
26
27     # create algod client
28     algod_client = algod.AlgodClient(algod_token, algod_address)
29
30     # check account balance
31     account_info = algod_client.account_info(addr1)

```

```

32     print("{} balance: {}".format(account_info.get('address'), account_info.get('amount'))) + "\n"
33     account_info = algod_client.account_info(addr2)
34     print("{} balance: {}".format(account_info.get('address'), account_info.get('amount'))) + "\n"
35     account_info = algod_client.account_info(addr3)
36     print("{} balance: {}".format(account_info.get('address'), account_info.get('amount'))) + "\n"
37
38 main()

```

Futtassuk le a 02-account_balance.py fájlt:

```

1 (pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/python_api (main) $ python
   ↳ 02-account_balance.py
2 DP07AIGM60H2UREMX2ZPS6SBMRAVEGAHV43BCTX4SI AOKQBWFFJRZY0E balance: 4000000000000000 microAlgo
3
4 DUWWUSWZSLBPGVY6GJ7QM5RTUC54EPJ4273FN50TFELSV0CAQGT62WRUQ balance: 2000000000000000 microAlgo
5
6 ICNK7LCUJZEULNYG3SEKZ5LGM5J3H2TTI BKYAAUPXACUEKHW33DGVQVA74 balance: 4000000000000000 microAlgo

```

Látható, hogy a pénztárca kezdetben három számlaszámot tartalmaz, és ezekben összesen 10^{16} mikro-Algó van, azaz 10 milliárd Algó. (10 milliárd Algó = 10 000 millió Algo = 10^{10} Algo)

4.4. Fizetési tranzakció kiadása

A fájl explorer-rel vagy a terminálban a nano 03-payment_transaction.py segítségével hozzuk létre a következő fájlt:

```

1 from algosdk import transaction
2 import json
3 import base64
4
5 from algosdk import kmd
6 from algosdk.wallet import Wallet
7 from algosdk.v2client import Algod
8
9 # define sandbox values for kmd client
10 kmd_address = "http://localhost:4002"
11 kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
12
13 # define sandbox values for algod client
14 algod_address = "http://localhost:4001"
15 algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
16
17 def main():
18     # create KMDClient
19     kmd_client = kmd.KMDClient(kmd_token, kmd_address)
20
21     # connect to default wallet
22     wallet = Wallet("unencrypted-default-wallet", "", kmd_client)
23
24     # gather the three default accounts
25     wallet_addresses = wallet.list_keys()
26     addr1 = wallet_addresses[0]

```

```

27     addr2 = wallet_addresses[1]
28     addr3 = wallet_addresses[2]
29
30     # create algod client
31     algod_client = algod.AlgodClient(algod_token, algod_address)
32
33     # build unsigned transaction
34     params = algod_client.suggested_params()
35     receiver = addr2
36     note = "Hello World".encode()
37     amount = 1000000
38     unsigned_txn = transaction.PaymentTxn(addr1, params, receiver, amount, None, note)
39
40     # sign transaction
41     signed_txn = unsigned_txn.sign(wallet.export_key(addr1))
42
43     # submit transaction
44     txid = algod_client.send_transaction(signed_txn)
45     print("Successfully sent transaction with txID: {}".format(txid))
46
47     # wait for confirmation
48     try:
49         confirmed_txn = transaction.wait_for_confirmation(algod_client, txid, 4)
50     except Exception as err:
51         print(err)
52         return
53
54     print("Transaction information: {}".format(
55         json.dumps(confirmed_txn, indent=4)))
56     print("Decoded note: {}".format(base64.b64decode(
57         confirmed_txn["txn"]["txn"]["note"].decode())))
58
59 main()

```

Futtassuk le a 03-payment_transaction.py fájlt:

```

1 (pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/python_api (mai n) $ python
↳ 03-payment_transaction.py
2 Successfully sent transaction with txID: 21CYN4NT6052XTN3YNKM5RCDBHSMEIA5PK7LA66TJN3M3KZF4HSQ
3 Transaction information: {
4     "confirmed-round": 1,
5     "pool-error": "",
6     "txn": {
7         "sig": "ZDb6f6E30kybPW6KBF7gaosEQBpAZCXIMLgeYSG23Bbg/Y0nUJz8ZsFz8/R7nTsYf1im1H509umuKjUxocJ
↳ OkAA==",
8         "txn": {
9             "amt": 1000000,
10            "fee": 1000,
11            "gen": "dockernet-v1",
12            "gh": "d2lGnqyAVFyDA61oZLE5NmjU64lg2vQr1jG3C1nnQ6l=",
13            "lv": 1000,
14            "note": "SGVsbG8gV29ybGQ=",
15            "rcv": "DUWUUSWZSLBPGVY6GJ7QM5RTUC54EPJ4273FN50TFELSVOCAQGTf62WRUQ",
16            "snd": "DP07AIGM60H2UREMX2ZPS6SBMRAVEGAHV43BCTX4SI AOKQBWFFJRZYOE",
17            "type": "pay"
18        }

```

```

19     }
20 }
21 Decoded note: Hello World

```

Látható, hogy sikerült a tranzakció, a tranzakció azonosító 2ICYN...F4HSQ. Kírártuk magát a tranzakciós rekordot is. (txn: tranzakció, amt: amount, fee, fv: first block value, gen: genezis, gh: genezis hash, lv: last block value, note, rcv: receiver, snd: sender, pay: payment txn)

A `note` mezőt dekódolt formában is kírártuk.

4.5. Számlaegyenlegek kíírása

A fájl explorer-rel vagy a terminálban a `nano 04-account_info.py` segítségével hozzuk létre a következő fájlt:

```

1  from algosdk import kmd
2  from algosdk.wallet import Wallet
3  from algosdk.v2client import algod
4  import json
5
6  # define sandbox values for kmd client
7  kmd_address = "http://localhost:4002"
8  kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
9
10 # define sandbox values for algod client
11 algod_address = "http://localhost:4001"
12 algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
13
14 def main() :
15     # create KMDClient
16     kmd_client = kmd.KMDClient(kmd_token, kmd_address)
17
18     # connect to default wallet
19     wallet = Wallet("unencrypted-default-wallet", "", kmd_client)
20
21     # gather the three default accounts
22     wallet_addresses = wallet.list_keys()
23     addr1 = wallet_addresses[0]
24     addr2 = wallet_addresses[1]
25     addr3 = wallet_addresses[2]
26
27     # create algod client
28     algod_client = algod.AlgodClient(algod_token, algod_address)
29
30     # check account details
31     account_info = algod_client.account_info(addr3)
32     print("Account information: {}".format(
33         json.dumps(account_info, indent=4)))
34     account_info = algod_client.account_info(addr2)
35     print("Account information: {}".format(
36         json.dumps(account_info, indent=4)))
37     account_info = algod_client.account_info(addr1)
38     print("Account information: {}".format(
39         json.dumps(account_info, indent=4)))
40 main()

```


Futtassuk le a 04-account_info.py fájl:

```
1 (playground-py3.10) @A-Maugli → .../akt02/hello/placeholder/python_api (main) $ python
   ↪ 04-account_info.py
2 Account information: {
3   "address": "1CNK7LCUJZEULNYG3SEKZ5LGM5J3H2TT1BKYAAUPXACUEKHW33DGXV0A74",
4   "amount": 4000000000000000,
5   "amount-without-pending-rewards": 4000000000000000,
6   "apps-local-state": [],
7   "apps-total-schema": {
8     "num-byte-slice": 0,
9     "num-uint": 0
10  },
11  "assets": [],
12  "created-apps": [],
13  "created-assets": [],
14  "min-balance": 100000,
15  "participation": {
16    "selection-participation-key": "2m3w5viSs8ZXPhW0+Mns+z8oX3dkJEH3M+DbJQWtN/U=",
17    "state-proof-key": "e9zi9f7feDLyGhG4RQoljddri eRHGpRMqZQ+7WeFuY0FAI ca07snXb68Tdl5b+Fz2hGNI8j
   ↪ hE8qbAQSsNhopp/Q==",
18    "vote-first-validated": 0,
19    "vote-key-dilution": 10000,
20    "vote-last-validated": 30000,
21    "vote-participation-key": "ERZwnlHHIhb37ERB93xwo6ejSbA3TyAtggzegnylS8="
22  },
23  "pending-rewards": 0,
24  "reward-base": 0,
25  "rewards": 0,
26  "round": 1,
27  "status": "Online",
28  "total-apps-opted-in": 0,
29  "total-assets-opted-in": 0,
30  "total-created-apps": 0,
31  "total-created-assets": 0
32 }
33 Account information: {
34   "address": "DUWWUSWZSLBPGVY6GJ7QM5RTUC54EPJ4273FN5QTFELSVOCAQGTf62WRUQ",
35   "amount": 2000000001000000,
36   "amount-without-pending-rewards": 2000000001000000,
37   "apps-local-state": [],
38   "apps-total-schema": {
39     "num-byte-slice": 0,
40     "num-uint": 0
41  },
42  "assets": [],
43  "created-apps": [],
44  "created-assets": [],
45  "min-balance": 100000,
46  "participation": {
47    "selection-participation-key": "JB7ZLgYDgt54LEZ4wpWEpKZ0swglTFkLI mXprW0yi3A=",
48    "state-proof-key": "/Kp6qQ9X2DBrBTI00BhCzmuDqmdEap/HDPAd9dxI8YUR5+HYgyQvn8456ImNm7vMMT28XgJ
   ↪ BSY4em3HI4b0Ii4A==",
49    "vote-first-validated": 0,
50    "vote-key-dilution": 10000,
51    "vote-last-validated": 30000,
52    "vote-participation-key": "k1S30rIKTr8D9CoB154NuK0hbiKOSYspUPNdGEIwcmY="
```

```

53     },
54     "pending-rewards": 0,
55     "reward-base": 0,
56     "rewards": 0,
57     "round": 1,
58     "status": "Online",
59     "total-apps-opted-in": 0,
60     "total-assets-opted-in": 0,
61     "total-created-apps": 0,
62     "total-created-assets": 0
63   }
64   Account information: {
65     "address": "DPO7AI GM60H2UREMX2ZPS6SBMRAVEGAAHV43BCTX4SI AOKOBWFFJRZYOE",
66     "amount": 3999999998999000,
67     "amount-without-pending-rewards": 3999999998999000,
68     "apps-local-state": [],
69     "apps-total-schema": {
70       "num-byte-slice": 0,
71       "num-uint": 0
72     },
73     "assets": [],
74     "created-apps": [],
75     "created-assets": [],
76     "min-balance": 100000,
77     "participation": {
78       "selection-participation-key": "tSvVZrUkGrQbS4YTJ7//K/ew56tJGd9r0DmXFSK01To=",
79       "state-proof-key": "uGQ3C1j Rdub2CgTsqTNakL4fvDBI i Di wgaUB39NDurQev601wo5U9cPx6t+tBK7i wmNgL8
↵ 0l aMHP3eUki sdIp0==",
80       "vote-first-validated": 0,
81       "vote-key-dilution": 10000,
82       "vote-last-validated": 30000,
83       "vote-participation-key": "EPG0+Y9oj VI gfStV72u8U4sCPLYLgqhys9XV0I zudrM="
84     },
85     "pending-rewards": 0,
86     "reward-base": 0,
87     "rewards": 0,
88     "round": 1,
89     "status": "Online",
90     "total-apps-opted-in": 0,
91     "total-assets-opted-in": 0,
92     "total-created-apps": 0,
93     "total-created-assets": 0
94   }

```

A három számlára vonatkozóan bőséges információt kaptunk. Az “assets”, azaz “pénzeszközök” mező még üres. Az ASA, azaz az Algorand Standard Assets a blokkláncon létrehozott, nem Algorand token-eket jelenti. Az ASA segítségével tetszőleges pénzeszközt létre lehet hozni, akár még „Fiatkát” is.

4.6. Pénzeszköz létrehozása

A fájl explorer-rel vagy a terminálban a `nano 05-asset_create.py` segítségével hozzuk létre a következő fájlt:

```

1  from algosdk import kmd, transaction
2  from algosdk.wallet import Wallet
3  from algosdk.v2client import algod
4
5  import json
6  import base64
7
8  # define sandbox values for kmd client
9  kmd_address = "http://localhost:4002"
10 kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
11
12 # define sandbox values for algod client
13 algod_address = "http://localhost:4001"
14 algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
15
16 def main() :
17     # create KMDClient
18     kmd_client = kmd.KMDClient(kmd_token, kmd_address)
19
20     # connect to default wallet
21     wallet = Wallet("unencrypted-default-wallet", "", kmd_client)
22
23     # gather the three default accounts and corresponding mnemonic passphrase
24     wallet_addresses = wallet.list_keys()
25     addr1 = wallet_addresses[0]
26     addr2 = wallet_addresses[1]
27     addr3 = wallet_addresses[2]
28
29     # create algod client
30     algod_client = algod.AlgodClient(algod_token, algod_address)
31
32     # build unsigned transaction
33     params = algod_client.suggested_params()
34     unsigned_txn = transaction.AssetConfigTxn(sender=addr1,
35         sp=params,
36         total=10000, # Fungible tokens have total issuance greater than 1
37         decimals=2, # Fungible tokens typically have decimals greater than 0
38         default_frozen=False,
39         unit_name="FUNTOK",
40         asset_name="Fun Token",
41         manager=addr1,
42         strict_empty_address_check=False,
43         reserve="",
44         freeze="",
45         clawback="",
46         url="https://path/to/my/fungible/asset/metadata.json",
47         metadata_hash="", # Typically include hash of metadata.json (bytes)
48     )
49
50     # sign transaction
51     signed_txn = unsigned_txn.sign(wallet.export_key(addr1))
52
53     #submit transaction
54     txid = algod_client.send_transaction(signed_txn)
55     print("Successfully sent transaction with txID: {}".format(txid))
56

```

```

57 # wait for confirmation
58 try:
59     confirmed_txn = transaction.wait_for_confirmation(algod_client, txid, 4)
60 except Exception as err:
61     print(err)
62     return
63
64 print("Transaction information: {}".format(
65     json.dumps(confirmed_txn, indent=4)))
66
67 # write the asset index to an environment file
68 f = open('asset.index', 'w+')
69 f.write(f'{{confirmed_txn["asset-index"]}}')
70 f.close()
71
72 main()

```

Az új pénzeszköz neve a “Fun Token”, („Vicces zseton”), az elemi egység neve FUNTOK, összesen 10000 db elemi egységet hozunk kezdetben létre, és 100 elemi egység felel meg egy egységnek, a decimals = 2 beállítás miatt, vagyis ténylegesen 100,00 FUNTOK zsetont hozunk létre.

Futtassuk le a 05-asset_create.py fájlt:

```

1 Transaction information: {
2   "asset-index": 1002,
3   "confirmed-round": 2,
4   "pool-error": "",
5   "txn": {
6     "sig": "9KHcv6feb9b1fkX6Wg6oRvlqj1/ttyaDjX95X00Wmv2ITe/c6zt3KHFPMBgbNZ9BmaoNkFIY0yr0a6XF5nJ
↵ e+Aw==",
7     "txn": {
8       "apar": {
9         "an": "Fun Token",
10        "au": "https://path/to/my/fungible/asset/metadata.json",
11        "dc": 2,
12        "m": "DP07AIGM60H2UREMX2ZPS6SBMRAVEGAAHV43BCTX4SIAOKQBFFFJRZYOE",
13        "t": 10000,
14        "un": "FUNTOK"
15      },
16      "fee": 1000,
17      "fv": 1,
18      "gen": "dockernet-v1",
19      "gh": "d2IGnqyAVFyDA61oZLE5NmjU64I g2v0r1jG3C1nn06I=",
20      "lv": 1001,
21      "snd": "DP07AIGM60H2UREMX2ZPS6SBMRAVEGAAHV43BCTX4SIAOKQBFFFJRZYOE",
22      "type": "acfg"
23    }
24  }
25 }

```

Létrejött az új “Fun Token” asset. Látható, hogy a létrehozott pénzeszköz indexe, az “asset-index” értéke 1002. Az “apar” a paramétereket tartalmazza: an: asset name, au: asset url, dc: decimals, m: manager address, t: total units, un: unit name, acfg: asset configuration (txn type)

4.7. Pénzeszközbe történő „benevezés”, opt-in

Az Algorandban mindaddig nem lehet egy számlára ASA-t küldeni, amíg az adott számla ki nem fejezi a hajlandóságát a pénzeszköz fogadására. Ezt a hajlandóságot angolul opt-in -nek hívjuk, ami azt jelenti, hogy „benevezünk” valamire, itt nevezetesen az adott ASA-ra.

Az opt-in úgy történik, hogy 0 egységet küldünk annak a címnek az adott ASA-ból, amelyik be szeretne nevezni rá. Ezt a mechanizmust az AssetOptInTxn függvény név tökéletesen elfedi.

A fájl explorer-rel vagy a terminálban a nano 06-asset_opt_in.py segítségével hozzuk létre a következő fájlt:

```
1 from algosdk import kmd, transaction
2 from algosdk.wallet import Wallet
3 from algosdk.v2client import algod
4
5 import json
6 import base64
7
8 # define sandbox values for kmd client
9 kmd_address = "http://localhost:4002"
10 kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
11
12 # define sandbox values for algod client
13 algod_address = "http://localhost:4001"
14 algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
15
16 def get_asset_index(default_index = 1010):
17     # try to read the asset index from our environment file
18     try:
19         index = int(open('asset.index', 'r').readline())
20     # otherwise return the default index
21     except:
22         index = default_index
23     return index
24
25 def main() :
26     # create KMDClient
27     kmd_client = kmd.KMDClient(kmd_token, kmd_address)
28
29     # connect to default wallet
30     wallet = Wallet("unencrypted-default-wallet", "", kmd_client)
31
32     # gather the three default accounts and corresponding mnemonic passphrase
33     wallet_addresses = wallet.list_keys()
34     addr1 = wallet_addresses[0]
35     addr2 = wallet_addresses[1]
36     addr3 = wallet_addresses[2]
37
38     # create algod client
39     algod_client = algod.AlgodClient(algod_token, algod_address)
40
41     # build unsigned transaction
42     params = algod_client.suggested_params()
43     sender = addr2
```

```

44 index = get_asset_index(default_index = 2) # ensure this matches the asset-index returned by
↳ asset_create.py
45 unsigned_txn = transaction.AssetOptInTxn(sender, params, index)
46
47 # sign transaction
48 signed_txn = unsigned_txn.sign(wallet.export_key(addr2))
49
50 # submit transaction
51 txid = algod_client.send_transaction(signed_txn)
52 print("Successfully sent transaction with txID: {}".format(txid))
53
54 # wait for confirmation
55 try:
56     confirmed_txn = transaction.wait_for_confirmation(algod_client, txid, 4)
57 except Exception as err:
58     print(err)
59     return
60
61 print("Transaction information: {}".format(
62     json.dumps(confirmed_txn, indent=4)))
63
64 main()

```

Futtassuk le a 06-asset_opt_in.py fájlt:

```

1 (playground-py3.10) @A-Mauggi → .../akt02/hello/playground/python_api (main) $ python
↳ 06-asset_opt_in.py
2 Successfully sent transaction with txID: 7DCSMMAVZNVHZ02AZMON5GKIEE4LJ4PDGWGAQFQWX6Y76POGWTC
3 Transaction information: {
4     "confirmed-round": 3,
5     "pool-error": "",
6     "txn": {
7         "sig": "DtNu3ZMV1jd+3EQHUeSbVa3oqP9FCJavz5kk5TsBoml CJZ40Z/KYubPT40B186gasqHXnRoK3mkeex4s+L
↳ 7PAw==",
8         "txn": {
9             "arcv": "DUWWUSWZSLBPGVY6GJ7QM5RTUC54EPJ4273FN50TFELSVOCAQGTf62WRUQ",
10            "fee": 1000,
11            "fv": 2,
12            "gen": "dockernet-v1",
13            "gh": "d2lGnqyAVFyDA61oZLE5NmjU64l g2v0r1j G3C1nnQ6l =",
14            "lv": 1002,
15            "snd": "DUWWUSWZSLBPGVY6GJ7QM5RTUC54EPJ4273FN50TFELSVOCAQGTf62WRUQ",
16            "type": "axfer",
17            "xaid": 1002
18        }
19    }
20 }

```

A tranzakció típus: "axfer", "asset transfer", és a "xaid" a "transfer asset id", és az "arcv", az "address receiver" az a cím, amelyik az ASA-ra benevezett.

4.8. Csere atomi tranzakciós csoporttal

Az Algorandban az ASA sok mindent képviselhet. Tegyük fel, hogy példánkban egy másik pénzeszközt jelent. Ahhoz, hogy az ASA-t el tudjuk adni Algorand-ért, biztosítani kell, hogy két tranzakció

menjen egyszerre végbe, vagy ne menjen végbe egy sem:

- a vevő Algorand-dal fizet az eladónak az ASA-ért
- az eladó elküldi az ASA-t a vevőnek

Az Algorandban a „tranzakció csoport” fogalma biztosítja a fenti cserét.

A fájl explorer-rel vagy a terminálban a `07-atomic_transaction.py` segítségével hozzuk létre a következő fájlt:

```
1 from algosdk import transaction
2 import json
3 import base64
4
5 from algosdk import kmd
6 from algosdk.wallet import Wallet
7 from algosdk.v2client import Algod
8
9 # define sandbox values for kmd client
10 kmd_address = "http://localhost:4002"
11 kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
12
13 # define sandbox values for algod client
14 algod_address = "http://localhost:4001"
15 algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
16
17 def get_asset_index(default_index = 1010):
18     # try to read the asset index from our environment file
19     try:
20         index = int(open('asset.index', 'r').readline())
21     # otherwise return the default index
22     except:
23         index = default_index
24     return index
25
26 def main() :
27     # create KMDClient
28     kmd_client = kmd.KMDClient(kmd_token, kmd_address)
29
30     # connect to default wallet
31     wallet = Wallet("unencrypted-default-wallet", "", kmd_client)
32
33     # gather the three default accounts and corresponding mnemonic passphrase
34     wallet_addresses = wallet.list_keys()
35     addr1 = wallet_addresses[0]
36     addr2 = wallet_addresses[1]
37     addr3 = wallet_addresses[2]
38
39     # create algod client
40     algod_client = algod.AlgodClient(algod_token, algod_address)
41
42     # build unsigned payment transaction
43     params = algod_client.suggested_params()
44     sender = addr2
45     receiver = addr1
46     amount = 1000000
```

```

47     txn_1 = transaction.PaymentTxn(sender, params, receiver, amount)
48
49     # build unsigned asset transfer transaction
50     sender = addr1
51     receiver = addr2
52     amount = 100 # remember this ASA has 2 decimal places, so this is 1.00 FUNTOK
53     index = get_asset_index(default_index = 1010) # ensure this matches the asset-index returned
54     ↪ by asset_create.py
55     txn_2 = transaction.AssetTransferTxn(sender, params, receiver, amount, index)
56
57     # group transactions
58     gid = transaction.calculate_group_id([txn_1, txn_2])
59     txn_1.group = gid
60     txn_2.group = gid
61
62     # sign transaction
63     stxn_1 = txn_1.sign(wallet.export_key(addr2))
64     stxn_2 = txn_2.sign(wallet.export_key(addr1))
65
66     # assemble transaction group
67     signed_group = [stxn_1, stxn_2]
68
69     #submit atomic transaction group
70     txid = algod_client.send_transactions(signed_group)
71     print("Successfully sent transaction with txID: {}".format(txid))
72
73     # wait for confirmation
74     try:
75         confirmed_txn = transaction.wait_for_confirmation(algod_client, txid, 4)
76     except Exception as err:
77         print(err)
78         return
79
80     print("Transaction information: {}".format(
81         json.dumps(confirmed_txn, indent=4)))
82
83 main()

```

Futtassuk le a 07-atomic_transaction.py fájlt:

```

1 (pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/python_api (mai n) $ python
2 ↪ 07-atomic_transaction.py
3 Successfully sent transaction with txID: 6AW7M7GDZKS2YA575LEOSRD3NKR7LF7RWTGY7I EWSL56B67MPY4A
4 Transaction information: {
5     "confirmed-round": 4,
6     "pool-error": "",
7     "txn": {
8         "sig": "WJz54q0qdzBtHhN417rmeKTxfqI BPKAI Ig+XytRtZrV42VAUswRcemomUEbV3vKm4I aZI DEVJr4sI Qi fM3j
9         ↪ XBCQ==",
10        "txn": {
11            "amt": 1000000,
12            "fee": 1000,
13            "fv": 3,
14            "gen": "dockernet-v1",
15            "gh": "d2I GnyAVFyDA61oZLE5Nmj U64I g2v0r1j G3C1nn06I =",
16            "grp": "SBs5j PxxYFk7y/WhsI LxTXn7QI BByaETrD5xTHECPI =",

```



```

15         "lv": 1003,
16         "rcv": "DP07AI GM60H2UREMX2ZPS6SBMRAVEGAHV43BCTX4SI AOKQBWFFJRZYOE",
17         "snd": "DUWWUSWZSLBPGVY6GJ7QM5RTUC54EPJ4273FN50TFELSVOCAQGTf62WRUQ",
18         "type": "pay"
19     }
20 }
21 }

```

Az tranzakció egyik felét írtuk ki, a “pay” “payment” txn-t, azaz a fizetési tranzakciót. De van egy másik tranzakció is a tranzakciós csoportban: az “axtr”, azaz az “asset transfer” tranzakció.

Az 04-account_info.py jön a segítségünkre:

```

1 (pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/python_api (mai n) $ python
↪ 04-account_info.py
2 Account information: {
3     "address": "ICNK7LCUJZEULNYG3SEKZ5LGM5J3H2TTI BKYAAUPXACUEKHW33DGXV0A74",
4     "amount": 4000000000000000,
5     "amount-without-pending-rewards": 4000000000000000,
6     "apps-local-state": [],
7     "apps-total-schema": {
8         "num-byte-slice": 0,
9         "num-uint": 0
10    },
11    "assets": [],
12    "created-apps": [],
13    "created-assets": [],
14    "min-balance": 100000,
15    "participation": {
16        "selection-participation-key": "2m3w5vi Ss8ZXPW0+Mns+z8oX3dkJEH3M+DbJQWtN/U=",
17        "state-proof-key": "e9zi 9f7feDLyGhG4RQoIj ddri eRHGpRMqZQ+7WeFuY0FAI ca07snXb68Tdl 5b+Fz2hGNI 8j
↪ hE8qbAQSSnhopp/Q=",
18        "vote-first-valid": 0,
19        "vote-key-dilution": 10000,
20        "vote-last-valid": 30000,
21        "vote-participation-key": "ERZwnI HHI hb37ERB93xwo6ej SbA3TyAtggzegnryl S8="
22    },
23    "pending-rewards": 0,
24    "reward-base": 0,
25    "rewards": 0,
26    "round": 4,
27    "status": "Online",
28    "total-apps-opted-in": 0,
29    "total-assets-opted-in": 0,
30    "total-created-apps": 0,
31    "total-created-assets": 0
32 }
33 Account information: {
34     "address": "DUWWUSWZSLBPGVY6GJ7QM5RTUC54EPJ4273FN50TFELSVOCAQGTf62WRUQ",
35     "amount": 1999999999998000,
36     "amount-without-pending-rewards": 1999999999998000,
37     "apps-local-state": [],
38     "apps-total-schema": {
39         "num-byte-slice": 0,
40         "num-uint": 0
41    },

```

```

42 "assets": [
43   {
44     "amount": 100,
45     "asset-id": 1002,
46     "is-frozen": false
47   }
48 ],
49 "created-apps": [],
50 "created-assets": [],
51 "min-balance": 200000,
52 "participation": {
53   "selection-participation-key": "JB7ZLgYDgt54LEZ4wpWEpKZ0swglTFkLI mXprW0yi 3A=",
54   "state-proof-key": "/Kp6q09X2DBrBTI Q0BhCzmuDqmdEap/HDPAd9dxI 8YUR5+HYgy0vn8456I mNm7vMMT28XgJ
↵ BSY4em3HI 4b0I i 4A==",
55   "vote-first-valid": 0,
56   "vote-key-dilution": 10000,
57   "vote-last-valid": 30000,
58   "vote-participation-key": "k1S30rI KTr8D9CoB154NuK0hbi kQSYspUPNdGEI WcMY="
59 },
60 "pending-rewards": 0,
61 "reward-base": 0,
62 "rewards": 0,
63 "round": 4,
64 "status": "Online",
65 "total-apps-opted-in": 0,
66 "total-assets-opted-in": 1,
67 "total-created-apps": 0,
68 "total-created-assets": 0
69 }
70 Account information: {
71   "address": "DP07AIGM60H2UREMX2ZPS6SBMRAVEGAAHV43BCTX4SI AOKQBWFFJRZYOE",
72   "amount": 399999999997000,
73   "amount-without-pending-rewards": 399999999997000,
74   "apps-local-state": [],
75   "apps-total-schema": {
76     "num-byte-slice": 0,
77     "num-uint": 0
78   },
79   "assets": [
80     {
81       "amount": 9900,
82       "asset-id": 1002,
83       "is-frozen": false
84     }
85   ],
86   "created-apps": [],
87   "created-assets": [
88     {
89       "index": 1002,
90       "params": {
91         "creator": "DP07AIGM60H2UREMX2ZPS6SBMRAVEGAAHV43BCTX4SI AOKQBWFFJRZYOE",
92         "decimals": 2,
93         "default-frozen": false,
94         "manager": "DP07AIGM60H2UREMX2ZPS6SBMRAVEGAAHV43BCTX4SI AOKQBWFFJRZYOE",
95         "name": "Fun Token",
96         "name-b64": "RnVulFRva2Vu",
97         "total": 10000,

```

```

98         "unit-name": "FUNTOK",
99         "unit-name-b64": "RI VOVE9L",
100        "url": "https://path/to/my/fungi/asset/metadata.json",
101        "url-b64": "aHROcHM6Ly9wYXRoL3RvL215L2Z1bmdpYmxlL2Fzc2V0L211dGFkYXRhLmpzb24="
102    }
103 }
104 ],
105 "min-balance": 200000,
106 "participation": {
107     "selection-participation-key": "tSvVZrUkGrQbS4YTJ7//K/ew56tJGd9rODmXFSK01To=",
108     "state-proof-key": "uG03C1j Rdub2CgTsqTNakL4fvDBI i Di wgaUB39NDurQev601wo5U9cPx6t+tbK7i wmNgL8_j
↪ 0I aMHP3eUki sdI p0=",
109     "vote-first-valuation": 0,
110     "vote-key-dilution": 10000,
111     "vote-last-valuation": 30000,
112     "vote-participation-key": "EPGO+Y9oj VI gfstV72u8U4sCPLYLgqhys9XV0I zudrM="
113 },
114 "pending-rewards": 0,
115 "reward-base": 0,
116 "rewards": 0,
117 "round": 4,
118 "status": "Online",
119 "total-apps-opted-in": 0,
120 "total-assets-opted-in": 1,
121 "total-created-apps": 0,
122 "total-created-assets": 1
123 }
124 (pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/python_api (main) $

```

A példa 42..48. sorában látszik, hogy a DUWU...2WRUQ címre az 1002-es "asset-id"-ből 1,00 FUNTOK megérkezett, ami a 2 tizedesjegy miatt 100-nak látszik. Ugyanakkor a 79..85. sorban az is látszik, hogy az ASÁ-t kibocsájtó DPO7A...ZYYOE" számla egyenlege 100-zal csökkent, és már csak 9900 egység van a "FUNTOK"-ból, ami a 2 tizedes miatt 99,00-nek felel meg.

5. Fejlesztés a Javascript SDK-val

Az Algorand Javascript SDK segítségével többféle környezetben is lehetséges a fejlesztés:

- Node.js környezetben, a konzol használatával
- Node.js környezetben, Web böngésző használatával
- közvetlenül a Web böngészőben, „összepakolt” JS könyvtár(ak) használatával

A következő részben az előzőleg már látott példákat mutatjuk be újra, ezúttal az Algorand JavaScript SDK használatával, Node.js környezetben, a konzol használatával. A JavaScript mintapéldák megtalálhatók a <https://github.com/A-Maugli/akt02> repository-ban, a `hellow/playground/js_api` könyvtárban.

5.1. Előkészületi lépések

Indítsuk el az utoljára használt Codespaces munkaterületünket. Mikor a munkaterület betöltődött, indítsuk el a privát blokkláncot:

```
1 @A-Maugli → /workspaces/akt02 (main) $ algokit --version
2 algokit, version 1.13.0
3 @A-Maugli → /workspaces/akt02 (main) $ algokit localnet stop
4 Stopping Algorand LocalNet now...
5 docker: Container algokit_sandbox_indexer Stopping
6 docker: Container algokit_sandbox_indexer Stopped
7 docker: Container algokit_sandbox_conduit Stopping
8 docker: Container algokit_sandbox_conduit Stopped
9 LocalNet Stopped; execute algokit localnet start to start it again.
10 @A-Maugli → /workspaces/akt02 (main) $ algokit localnet start
```

A Ports fülön tegyük globálissá a portokat. Ehhez a lakat szimbólumra kell kattintanunk az egérrel. A lakat ennek hatására kinyílik.

Hozzunk létre egy könyvtárat a `/workspace/akt02/hellow/playground` alatt, `js_api` névvel. Lépünk be ebbe a könyvtárba:

```
cd /workspace/akt02/hellow/playground/js_api
```

Hozzunk létre egy `package.json` csomagot:

```
npm init
```

Adjuk hozzá a `packages.json` leíró fájljához az `algsdk` könyvtárat:

```
npm install algsdk
```

5.2. Algorand számlaszám létrehozása

A fájl explorer-rel vagy egy terminálban a nano szövegszerkesztővel hozzuk létre a következő fájlt: `01-account_generation.js`

```
1 const algsdk = require('algsdk');
2 const DEBUG = 0;
3
```

```

4  const createAccount = function() {
5      try {
6          const myAccount = algosdk.generateAccount();
7          console.log("Account address:", myAccount.addr);
8          if (DEBUG) console.log("Private key:", myAccount.sk);
9          let b64_sk = Buffer.from(myAccount.sk).toString('base64');
10         console.log("Private key:", b64_sk);
11         const accountMnemonic = algosdk.secretKeyToMnemonic(myAccount.sk);
12         console.log("Account mnemonic:", accountMnemonic);
13         return myAccount;
14     }
15     catch (err) {
16         console.log("err:", err);
17     }
18 }
19
20 createAccount();

```

Installáljuk a függőségeket az `algokit bootstrap all` paranccsal:

```

1  (playground-py3.10) @A-Maugli → .../akt02/hello/placeholder/js_api (main) $ algokit bootstrap all
2  Installing npm dependencies
3  npm:
4  npm: added 13 packages, and audited 14 packages in 1s
5  npm:
6  npm: 3 packages are looking for funding
7  npm: run npm fund for details
8  npm:
9  npm: found 0 vulnerabilities
10 npm: npm notice
11 npm: npm notice New minor version of npm available! 10.2.4 -> 10.5.0
12 npm: npm notice Changelog: <https://github.com/npm/cli/releases/tag/v10.5.0>
13 npm: npm notice Run npm install -g npm@10.5.0 to update!
14 npm: npm notice
15 Finished bootstrapping /workspaces/akt02/hello/placeholder/js_api

```

Futtassuk a `01-account_generation.js` fájlt:

```

1  (playground-py3.10) @A-Maugli → .../akt02/hello/placeholder/js_api (main) $ node
   → 01-account_generation.js
2  Account address: MLLDAU2SRDZBTT5KCMNPK6B406N6ZYSUEX32AEPQH7Q25EWCK7MJCVK3BY
3  Private key:
   → WGOJV00y9vZR33nzi zAfpNhJNEdSxUoae/nzgr8T8A9i 1jBTUojyGc+qExr1eDx3m+ziVCX3oBHwP+GuksJX2A==
4  Account mnemonic: prison certain present comfort uniform laundry whisper keep lazy scene auto
   → medal neck snack mutual shed enable unaware witness connect lecture agent legend abandon
   → crumble
5  (playground-py3.10) @A-Maugli → .../akt02/hello/placeholder/js_api (main) $

```

5.3. Számla egyenleg kiírása

A fájl explorer-rel vagy a terminálban a `nano 02-account_balance.js` segítségével hozzuk létre a következő fájlt:

```

1  algosdk = require('algosdk');
2

```

```

3  const DEBUG=0;
4
5  // define sandbox values for kmd client
6  const kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
7  const kmd_server = "http://localhost";
8  const kmd_server_port = 4002;
9
10 // define sandbox values for algod client
11 const algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
12 const algod_server = "http://localhost";
13 const algod_server_port = 4001;
14
15 async function main() {
16   // create kmd client
17   const kmd_client = new algosdk.Kmd(kmd_token, kmd_server, kmd_server_port);
18
19   // list wallets
20   wallets = await kmd_client.listWallets();
21   if(DEBUG) console.log('wallets:', wallets);
22
23   // get wallet index for default wallet
24   wallets.wallets.forEach(item => {
25     if (item.name === 'unencrypted-default-wallet') {
26       wallet_id = item.id;
27     }
28   })
29   // get wallet_handle for default wallet
30   wallet_handle = await kmd_client.initWalletHandle(wallet_id, '');
31   if(DEBUG) console.log('wallet_handle:', wallet_handle);
32
33   // get accounts (addresses) from default wallet
34   wallet_addresses = await kmd_client.listKeys(wallet_handle.wallet_handle_token);
35   if(DEBUG) console.log('wallet_addresses:', wallet_addresses);
36
37   // create algod client
38   const algod_client = new algosdk.Algodv2(algod_token, algod_server, algod_server_port);
39
40   // check account balance for each account
41   wallet_addresses.addresses.forEach(async (addr) => {
42     if(DEBUG) console.log('addr', addr);
43     account_info = await algod_client.accountInformation(addr).do();
44     if(DEBUG) console.log('account_info', account_info);
45     console.log('%s balance: %s microAlgos', account_info.address, account_info.amount);
46
47   });
48
49   // release wallet handle
50   let hr = await kmd_client.releaseWalletHandle(wallet_handle['wallet_handle_token']);
51   if (DEBUG) console.log('wallet handle released, hr:', hr)
52 }
53
54 main();

```

Futtassuk le a 02-account_balance.js fájlt:

```

1 (pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/js_api (mai n) $ node
  → 02-account_balance.js
2 DP07AIGM60H2UREMX2ZPS6SBMRAVEGAAHV43BCTX4SI AOKQBWFFJRZY0E balance: 399999999997000 mi croAl gos
3 DUWWUSWZSLBPGVY6GJ7QM5RTUC54EPJ4273FN50TFELSVOCAQGTf62WRU0 balance: 199999999998000 mi croAl gos
4 I CNK7LCUJZEULNYG3SEKZ5LGM5J3H2TTI BKYAAUPXACUEKHW3DGVQA74 balance: 4000000000000000 mi croAl gos
5 (pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/js_api (mai n) $

```

Látható, hogy a pénztárca kezdetben három számlaszámot tartalmaz, és ezekben összesen 10^{16} mikro-Algó van, azaz 10 milliárd Algó. (10 milliárd Algó = 10 000 millió Algó = 10^{10} Algo)

5.4. Fizetési tranzakció kiadása

A fájl explorer-rel vagy a terminálban a `nano 03-payment_transaction.js` segítségével hozzuk létre a következő fájlt:

```

1 const algosdk = require('algosdk');
2
3 const DEBUG=0;
4
5 // define sandbox values for kmd client
6 const kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
7 const kmd_server = "http://localhost";
8 const kmd_server_port = 4002;
9
10 // define sandbox values for algod client
11 const algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
12 const algod_server = "http://localhost";
13 const algod_server_port = 4001;
14
15 async function getWalletId(
16     kmdClient,
17     walletName) {
18
19     // list wallets
20     wallets = await kmdClient.listWallets();
21     if(DEBUG) console.log('wallets:', wallets);
22
23     // get wallet index for default wallet
24     let walletId='';
25     wallets.wallets.forEach(item => {
26         if (item.name === walletName) {
27             walletId = item.id;
28         }
29     })
30     return walletId;
31 }
32
33 async function main() {
34     // create kmd client
35     kmd_client = new algosdk.Kmd(kmd_token, kmd_server, kmd_server_port);
36
37     // get wallet id for default wallet
38     wallet_id = await getWalletId(kmd_client, 'unencrypted-default-wallet');
39     if(DEBUG) console.log('wallet_id:', wallet_id);

```

```

40 // get wallet_handle for default wallet
41 wallet_handle = await kmd_client.initWalletHandle(wallet_id, '');
42 if(DEBUG) console.log('wallet_handle:', wallet_handle);
43
44
45 // get accounts (addresses) from default wallet
46 wallet_addresses = await kmd_client.listKeys(wallet_handle.wallet_handle_token);
47 if(DEBUG) console.log('wallet_addresses:', wallet_addresses);
48 addr1 = wallet_addresses.addresses[0];
49 addr2 = wallet_addresses.addresses[1];
50 addr3 = wallet_addresses.addresses[2];
51
52 // create algod client
53 algod_client = new algosdk.Algodv2(algod_token, algod_server, algod_server_port);
54
55 // build unsigned transaction
56 params = await algod_client.getTransactionParams().do();
57 params['fee'] = 0;
58 if (DEBUG) console.log('params:', params);
59 unsigned_txn = algosdk.makePaymentTxnWithSuggestedParamsFromObject({
60   from: addr1,
61   to: addr2,
62   amount: algosdk.algosToMicroalgos(0.15),
63   note: algosdk.encodeObj("Hello World"),
64   suggestedParams: params
65 });
66 if (DEBUG) console.log('unsigned_txn:', unsigned_txn);
67
68 // sign transaction
69 addr1_sk = await kmd_client.exportKey(wallet_handle.wallet_handle_token, "", addr1);
70 if (DEBUG) console.log('addr1_sk:', addr1_sk);
71 signed_txn = unsigned_txn.signTxn(addr1_sk.private_key);
72 if (DEBUG) console.log('signed_txn:', signed_txn);
73
74 // submit transaction
75 tx_id = await algod_client.sendRawTransaction(signed_txn).do();
76 console.log("Successfully sent transaction with tx_id: %s", tx_id);
77 console.log('tx_id["txId"]:', tx_id['txId']);
78
79 // wait for confirmation
80 console.log('Awaiting confirmation (this may take several seconds)...');
81 const roundTimeout = 7;
82 const confirmed_txn = await algosdk.waitForConfirmation(
83   algod_client,
84   tx_id['txId'],
85   roundTimeout
86 );
87 console.log('Transaction is successfully completed');
88
89 // log confirmed transaction
90 console.log('confirmed_txn:', confirmed_txn);
91
92 // log decoded note
93 let decoded_note = algosdk.decodeObj(confirmed_txn['txn']['txn']['note']);
94 console.log('decoded_note:', decoded_note);
95
96 // release wallet handle

```



```

97     let hr = await kmd_client.releaseWalletHandle(wallet_handle['wallet_handle_token']);
98     if (DEBUG) console.log('wallet handle released, hr:', hr)
99 }
100
101 main();

```

Futtassuk le a `03-payment_transaction.js` fájlt:

```

1 (playground-py3.10) @A-Maugli → .../akt02/hello/playground/js_api (main) $ node
  ↳ 03-payment_transaction.js
2 Successfully sent transaction with tx_id: { txId:
  ↳ 'MIPIZFQWBSQJGMSZ5DR4BP5WK5OWWT7KN7YTOAHDM4FM6TCTNUPQ' }
3 tx_id["txId"]: MIPIZFQWBSQJGMSZ5DR4BP5WK5OWWT7KN7YTOAHDM4FM6TCTNUPQ
4 Awaiting confirmation (this may take several seconds)...
5 Transaction is successfully completed
6 confirmed_txn: {
7   'confirmed-round': 5,
8   'pool-error': '',
9   txn: {
10    sig: Uint8Array(64) [
11      135, 55, 63, 198, 207, 182, 235, 66, 129, 201, 133,
12      65, 195, 236, 195, 242, 242, 95, 182, 253, 181, 189,
13      151, 179, 127, 251, 8, 21, 20, 37, 8, 91, 177,
14      93, 49, 14, 226, 71, 67, 126, 128, 238, 73, 29,
15      82, 105, 21, 67, 195, 245, 68, 25, 127, 26, 84,
16      110, 62, 218, 168, 41, 84, 190, 50, 4
17    ],
18    txn: {
19      amt: 150000,
20      fee: 1000,
21      fv: 4,
22      gen: 'dockernet-v1',
23      gh: [Uint8Array],
24      lv: 1004,
25      note: [Uint8Array],
26      rcv: [Uint8Array],
27      snd: [Uint8Array],
28      type: 'pay'
29    }
30  }
31 }
32 decoded_note: Hello World
33 (playground-py3.10) @A-Maugli → .../akt02/hello/playground/js_api (main) $

```

Látható, hogy sikerült a tranzakció, a tranzakció azonosító MIPIZ...TNUPQ. Kírártuk magát a tranzakciós rekordot is. (txn: tranzakció, amt: amount, fee, fv: first block value, gen: genesis, gh: genesis hash, lv: last block value, note, rcv: receiver, snd: sender, pay: payment txn)

A `note` mezőt dekódolt formában is kírártuk.

5.5. Számlaegyenlegek kíírása

A fájl explorer-rel vagy a terminálban a `nano 04-account_info.js` segítségével hozzuk létre a következő fájlt:

```

1  var algodk = require('algosdk');
2
3  const DEBUG=0;
4
5  // define sandbox values for kmd client
6  const kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
7  const kmd_server = "http://localhost";
8  const kmd_server_port = 4002;
9
10 // define sandbox values for algod client
11 const algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
12 const algod_server = "http://localhost";
13 const algod_server_port = 4001;
14
15 async function main() {
16     // create kmd client
17     const kmdClient = new algodk.Kmd(kmd_token, kmd_server, kmd_server_port);
18
19     // list wallets
20     wallets = await kmdClient.listWallets();
21     if(DEBUG) console.log('wallets:', wallets);
22
23     // get wallet index for default wallet
24     let wallet_id=''
25     wallets.wallets.forEach(item => {
26         if (item.name === 'unencrypted-default-wallet') {
27             wallet_id = item.id;
28         }
29     })
30     // get wallet_handle for default wallet
31     wallet_handle = await kmdClient.initWalletHandle(wallet_id, '');
32     if(DEBUG) console.log('wallet_handle:', wallet_handle);
33
34     // get accounts (addresses) from default wallet
35     wallet_addresses = await kmdClient.listKeys(wallet_handle.wallet_handle_token);
36     if(DEBUG) console.log('wallet_addresses:', wallet_addresses);
37
38     // create algod client
39     const algodClient = new algodk.Algodv2(algod_token, algod_server, algod_server_port);
40
41     // check account balance for each account
42     wallet_addresses.addresses.forEach(async (addr) => {
43         if(DEBUG) console.log('addr', addr);
44         account_info = await algodClient.accountInformation(addr).do();
45         console.log('account_info', account_info);
46     });
47
48     // release wallet handle
49     let hr = await kmdClient.releaseWalletHandle(wallet_handle['wallet_handle_token']);
50     if (DEBUG) console.log('wallet handle released, hr:', hr)
51 }
52
53 main();

```

Futtassuk le a 04-account_info.js fájl:

```

1 (pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/js_api (mai n) $ node
2 → 04-account_info.js
3 account_info {
4   address: 'DP07AI GM60H2UREMX2ZPS6SBMRAVEGAAHV43BCTX4SI AOKQBWFFJRZYOE',
5   amount: 399999999846000,
6   'amount-without-pending-rewards': 399999999846000,
7   'apps-local-state': [],
8   'apps-total-schema': { 'num-byte-slice': 0, 'num-uint': 0 },
9   assets: [ { amount: 9900, 'asset-id': 1002, 'is-frozen': false } ],
10  'created-apps': [],
11  'created-assets': [ { index: 1002, params: [Object] } ],
12  'min-balance': 200000,
13  participation: {
14    'selection-participation-key': 'tSvVZrUkGrObS4YTJ7//K/ew56tJGd9r0DmXFSK01To=',
15    'state-proof-key':
16      → 'uG03C1j Rdbu2CgTsqTNakL4fvDBI i Di wgaUB39NDur0ev601wo5U9cPx6t+tBK7i wmNgL80I aMHP3eUki sdi pQ==',
17    'vote-first-valid': 0,
18    'vote-key-dilution': 10000,
19    'vote-last-valid': 30000,
20    'vote-participation-key': 'EPG0+Y9oj VI gfstV72u8U4sCPLYLgqhys9XV0I zudrM='
21  },
22  'pending-rewards': 0,
23  'reward-base': 0,
24  rewards: 0,
25  round: 5,
26  status: 'Online',
27  'total-apps-opted-in': 0,
28  'total-assets-opted-in': 1,
29  'total-created-apps': 0,
30  'total-created-assets': 1
31 }
32 account_info {
33   address: 'DUWWUSWZSLBPGVY6GJ7QM5RTUC54EPJ4273FN50TFELSV0CAQGTf62WRUQ',
34   amount: 2000000000148000,
35   'amount-without-pending-rewards': 2000000000148000,
36   'apps-local-state': [],
37   'apps-total-schema': { 'num-byte-slice': 0, 'num-uint': 0 },
38   assets: [ { amount: 100, 'asset-id': 1002, 'is-frozen': false } ],
39   'created-apps': [],
40   'created-assets': [],
41   'min-balance': 200000,
42   participation: {
43     'selection-participation-key': 'JB7ZLgYDgt54LEZ4wpWEpKZ0swgl TFKLI mXprW0yi 3A=',
44     'state-proof-key':
45       → '/Kp6q09X2DBrBTI Q0BhCzmuDqmdEap/HDPAd9dxI 8YUR5+HYgy0vn8456I mNm7vMMT28XgBSY4em3HI 4b0I i 4A==',
46     'vote-first-valid': 0,
47     'vote-key-dilution': 10000,
48     'vote-last-valid': 30000,
49     'vote-participation-key': 'k1S30rI KTr8D9CoB154NuK0hbi kQSYspUPNdGEI WcMY='
50   },
51   'pending-rewards': 0,
52   'reward-base': 0,
53   rewards: 0,
54   round: 5,
55   status: 'Online',
56   'total-apps-opted-in': 0,

```

```

54   'total-assets-opted-in': 1,
55   'total-created-apps': 0,
56   'total-created-assets': 0
57 }
58 account_info {
59   address: '1CNK7LCUJZEULNYG3SEKZ5LGM5J3H2TTI BKYAAUPXACUEKHW33DGXVQA74',
60   amount: 4000000000000000,
61   'amount-without-pending-rewards': 4000000000000000,
62   'apps-local-state': [],
63   'apps-total-schema': { 'num-byte-slice': 0, 'num-uint': 0 },
64   assets: [],
65   'created-apps': [],
66   'created-assets': [],
67   'min-balance': 100000,
68   participation: {
69     'selection-participation-key': '2m3w5vi Ss8ZXPW0+Mns+z8oX3dkJEH3M+DbJQWtN/U=',
70     'state-proof-key':
71     ↪ 'e9zi 9f7feDLyGhG4RQoIj ddri eRHGpRMqZQ+7WeFuY0FAI ca07snXb68TdI 5b+Fz2hGNI 8hE8qbAQSSnhopp/Q==',
72     'vote-first-valid': 0,
73     'vote-key-dilution': 10000,
74     'vote-last-valid': 30000,
75     'vote-participation-key': 'ERZwnIHHI hb37ERB93xwo6ej SbA3TyAtggzegnryl S8='
76   },
77   'pending-rewards': 0,
78   'reward-base': 0,
79   rewards: 0,
80   round: 5,
81   status: 'Online',
82   'total-apps-opted-in': 0,
83   'total-assets-opted-in': 0,
84   'total-created-apps': 0,
85   'total-created-assets': 0
86 }
(pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/js_api (main) $

```

A három számlára vonatkozóan bőséges információt kaptunk. A 8. és 36. sorban láthatóak az előzőleg, a `python_api` segítségével létrehozott “assets”, azaz “pénzeszközök”. Az ASA, azaz az Algorand Standard Assets a blokkláncon létrehozott, nem Algorand token-eket jelenti. Az ASA segítségével tetszőleges pénzeszköz létrehozható, akár még a „Fibatka” is.

5.6. Pénzeszköz létrehozása

A fájl explorer-rel vagy a terminálban a `nano 05-asset_create.js` segítségével hozzuk létre a következő fájlt:

```

1  const algosdk = require('algorithmsdk');
2  const fs = require('fs');
3  const DEBUG=0;
4
5  // define sandbox values for kmd client
6  const kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
7  const kmd_server = "http://localhost";
8  const kmd_server_port = 4002;
9
10 // define sandbox values for algod client

```

```

11 const algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
12 const algod_server = "http://localhost";
13 const algod_server_port = 4001;
14
15 async function getWalletId(
16     kmdClient,
17     walletName) {
18
19     // list wallets
20     wallets = await kmdClient.listWallets();
21     if(DEBUG) console.log('wallets:', wallets);
22
23     // get wallet index for default wallet
24     let walletId='';
25     wallets.wallets.forEach(item => {
26         if (item.name === walletName) {
27             walletId = item.id;
28         }
29     })
30     return walletId;
31 }
32
33 async function main() {
34     // create kmd client
35     kmd_client = new algosdk.Kmd(kmd_token, kmd_server, kmd_server_port);
36
37     // connect to default wallet
38     wallet_id = await getWalletId(kmd_client, 'unencrypted-default-wallet');
39     wallet_handle = await kmd_client.initWalletHandle(wallet_id, '');
40
41     // gather the first three accounts from the wallet
42     wallet_addresses = await kmd_client.listKeys(wallet_handle.wallet_handle_token);
43     addr1 = wallet_addresses.addresses[0];
44     addr2 = wallet_addresses.addresses[1];
45     addr3 = wallet_addresses.addresses[2];
46
47     // create algod client
48     algod_client = new algosdk.Algodv2(algod_token, algod_server, algod_server_port);
49
50     // get params
51     params = await algod_client.getTransactionParams().do();
52     if (DEBUG) console.log('params:', params);
53
54     // build asset create txn
55     unsigned_txn = algosdk.makeAssetCreateTxnWithSuggestedParamsFromObject({
56         from: addr1,
57         suggestedParams: params,
58         total: 10000, // Fungible token, number of total coins: 10000 / 100, because decimals is
59             ↳ 2
60         decimals: 2,
61         defaultFrozen: false,
62         unitName: "FUNTOK",
63         assetName: "Fun Token",
64         manager: addr1,
65         reserve: undefined,
66         freeze: undefined,
67         clawback: undefined,

```

```

67     assetURL: "https://path/to/my/fungible/asset/metadata.json",
68     assetMetadataHash: undefined
69   })
70   if (DEBUG) console.log('unsigned_txn:', unsigned_txn);
71
72   // sign transaction
73   addr1_sk = await kmd_client.exportKey(wallet_handle.wallet_handle_token, '', addr1);
74   if (DEBUG) console.log('addr1_sk:', addr1_sk);
75   signed_txn = unsigned_txn.signTxn(addr1_sk.private_key);
76   if (DEBUG) console.log('signed_txn:', signed_txn);
77
78   // submit transaction
79   tx_id = await algod_client.sendRawTransaction(signed_txn).do();
80   console.log("Successfully sent transaction with tx_id: %s", tx_id);
81   console.log('tx_id["txId"]:', tx_id['txId']);
82
83   // wait for confirmation
84   console.log('Awaiting confirmation (this may take several seconds)...');
85   const roundTimeout = 7;
86   const confirmed_txn = await algosdk.waitForConfirmation(
87     algod_client,
88     tx_id['txId'],
89     roundTimeout
90   );
91   console.log('Transaction is successfully completed');
92
93   // log confirmed transaction
94   console.log('confirmed_txn:', confirmed_txn);
95
96   // write asset id to an environment file
97   asset_index = (confirmed_txn['asset-index']).toString();
98   fs.writeFile('5_asset_index.txt', asset_index, err => {
99     if (err) {
100       console.log(err);
101     }
102     else {
103       console.log('File 5_asset_index.txt written successfully')
104     }
105   });
106
107   // release wallet handle
108   let hr = await kmd_client.releaseWalletHandle(wallet_handle['wallet_handle_token']);
109   if (DEBUG) console.log('wallet handle released, hr:', hr)
110 }
111
112 main();

```

Az új pénzeszköz neve a “Fun Token”, („Vicces zseton”), az elemi egység neve FUNTOK, összesen 10000 db elemi egységet hozunk kezdetben létre, és 100 elemi egység felel meg egy egységnek, a decimals = 2 beállítás miatt, vagyis ténylegesen 100,00 FUNTOK zsetont hozunk létre.

Futtassuk le a `05-asset_create.js` fájlt:

```

1 (playground-py3.10) @A-Maugli → .../akt02/hello/playground/js_api (main) $ node
  → 05-asset_create.js

```

```

2 Successfully sent transaction with tx_id: { txId:
  ↳ '6MRNWTG0ORZF7KESPQIS2TTWVIGXJSMMMIQJWKZ5UKGDTWCAJCXAX' }
3 tx_id["txId"]: 6MRNWTG0ORZF7KESPQIS2TTWVIGXJSMMMIQJWKZ5UKGDTWCAJCXAX
4 Awaiting confirmation (this may take several seconds)...
5 Transaction is successfully completed
6 confirmed_txn: {
7   'asset-index': 1007,
8   'confirmed-round': 6,
9   'pool-error': '',
10  txn: {
11    sig: Uint8Array(64) [
12      89, 214, 148, 184, 195, 166, 94, 96, 97, 177, 83,
13      5, 197, 148, 243, 68, 131, 166, 73, 135, 164, 255,
14      221, 67, 198, 251, 3, 104, 127, 90, 2, 136, 81,
15      56, 33, 250, 248, 179, 151, 226, 162, 146, 192, 46,
16      32, 105, 237, 69, 133, 79, 139, 118, 75, 207, 170,
17      47, 7, 121, 75, 48, 55, 22, 128, 2
18    ],
19    txn: {
20      apar: [Object],
21      fee: 1000,
22      fv: 5,
23      gen: 'dockernet-v1',
24      gh: [Uint8Array],
25      lv: 1005,
26      snd: [Uint8Array],
27      type: 'acfg'
28    }
29  }
30 }
31 File 5_asset_index.txt written successfully
32 (playground-py3.10) @A-Maugli → .../akt02/hello/playground/js_api (main) $

```

Létrejött az új “Fun Token” asset. Látható, hogy a létrehozott pénzeszköz indexe, az “asset-index” értéke 1007. Az “apar” a paramétereket tartalmazza, de a paraméterek ezek a listán nem látszanak, an: asset name, au: asset url, dc: decimals, m: manager address, t: total units, un: unit name, acfg: asset configuration (txn type)

5.7. Pénzeszközbe történő „benevezés”, opt-in

Az Algorandban mindaddig nem lehet egy számlára ASA-t küldeni, amíg az adott számla ki nem fejezi a hajlandóságát a pénzeszköz fogadására. Ezt a hajlandóságot angolul opt-in -nek hívjuk, ami azt jelenti, hogy „benevezünk” valamire, itt nevezetesen az adott ASA-ra.

Az opt-in úgy történik, hogy 0 egységet küldünk annak a címnek az adott ASA-ból, amelyik szeretne benevezni rá, lásd `makeAssetTransferTxnWithSuggestedParamsFromObject`.

A fájl explorer-rel vagy a terminálban a `nano 06-asset_opt_in.js` segítségével hozzuk létre a következő fájlt:

```

1 const algosdk = require('algorithms');
2 const fs = require('fs');
3 const DEBUG=0;
4

```

```

5 // define sandbox values for kmd client
6 const kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
7 const kmd_server = "http://localhost";
8 const kmd_server_port = 4002;
9
10 // define sandbox values for algod client
11 const algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
12 const algod_server = "http://localhost";
13 const algod_server_port = 4001;
14
15 async function getWalletId(
16     kmdClient,
17     walletName) {
18
19     // list wallets
20     wallets = await kmdClient.listWallets();
21     if(DEBUG) console.log('wallets:', wallets);
22
23     // get wallet index for default wallet
24     let walletId='';
25     wallets.wallets.forEach(item => {
26         if (item.name === walletName) {
27             walletId = item.id;
28         }
29     })
30     return walletId;
31 }
32
33 function getAssetIndex(fname) {
34     try {
35         var data = fs.readFileSync(fname, 'utf8');
36         var asset_index = Number(data);
37         if (DEBUG) console.log(asset_index);
38     } catch (err) {
39         console.error(err);
40     }
41     return asset_index;
42 }
43
44 async function main() {
45     // create kmd client
46     kmd_client = new algodk.Kmd(kmd_token, kmd_server, kmd_server_port);
47
48     // connect to default wallet
49     const wallet_name = 'unencrypted-default-wallet';
50     const wallet_pw = '';
51     wallet_id = await getWalletId(kmd_client, wallet_name);
52     wallet_handle = await kmd_client.initWalletHandle(wallet_id, wallet_pw);
53
54     // gather the first three accounts from the wallet
55     wallet_addresses = await kmd_client.listKeys(wallet_handle.wallet_handle_token);
56     addr1 = wallet_addresses.addresses[0];
57     addr2 = wallet_addresses.addresses[1];
58     addr3 = wallet_addresses.addresses[2];
59
60     // create algod client
61     algod_client = new algodk.Algodv2(algod_token, algod_server, algod_server_port);

```



```

62
63 // get asset index from file
64 asset_index = getAssetIndex('5_asset_index.txt');
65 if (DEBUG) console.log('asset_index:', asset_index);
66
67 // get params
68 params = await algod_client.getTransactionParams().do();
69 if (DEBUG) console.log('params:', params);
70
71 // build asset optin transaction
72 unsigned_txn = algosdk.makeAssetTransferTxnWithSuggestedParamsFromObject({
73   suggestedParams: params,
74   from: addr2,
75   to: addr2,
76   assetIndex: asset_index,
77   amount: 0
78 });
79 if (DEBUG) console.log('unsigned_txn:', unsigned_txn);
80
81 // sign transaction
82 addr2_sk = await kmd_client.exportKey(wallet_handle.wallet_handle_token, '', addr2);
83 if (DEBUG) console.log('addr1_sk:', addr2_sk);
84 signed_txn = unsigned_txn.signTxn(addr2_sk.private_key);
85 if (DEBUG) console.log('signed_txn:', signed_txn);
86
87 // submit transaction
88 tx_id = await algod_client.sendRawTransaction(signed_txn).do();
89 console.log("Successfully sent transaction with tx_id: %s", tx_id);
90 console.log('tx_id["txId"]:', tx_id['txId']);
91
92 // wait for confirmation
93 console.log('Awaiting confirmation (this may take several seconds)...');
94 const roundTimeout = 7;
95 const confirmed_txn = await algosdk.waitForConfirmation(
96   algod_client,
97   tx_id['txId'],
98   roundTimeout
99 );
100 console.log('Transaction is successfully completed');
101
102 // log confirmed transaction
103 console.log('confirmed_txn:', confirmed_txn);
104
105 // release wallet handle
106 let hr = await kmd_client.releaseWalletHandle(wallet_handle['wallet_handle_token']);
107 if (DEBUG) console.log('wallet handle released, hr:', hr)
108 }
109
110 main();

```

Futtassuk le a 06-asset_opt_in.js fájlt:

```

1 (playground-py3.10) @A-Maugli → .../akt02/hello/playground/js_api (main) $ node
  → 06-asset_opt_in.js
2 Successfully sent transaction with tx_id: { txId:
  → 'MC4NDEDTI W7TDRI Z2HETCADG4JF7WR4RJCGUVUSSHDVHNYLFWHLA' }

```

```

3 tx_id["txId"]: MC4NDEDTIW7TDRI Z2HETCADG4JF7WR4RJCGUVUSSHDVHNYLFWHLA
4 Awaiting confirmation (this may take several seconds)...
5 Transaction is successfully completed
6 confirmed_txn: {
7   'confirmed-round': 7,
8   'pool-error': '',
9   txn: {
10    sig: Uint8Array(64) [
11      21, 72, 32, 49, 110, 132, 206, 97, 160, 12, 242,
12      37, 177, 24, 0, 138, 8, 106, 193, 4, 98, 70,
13      202, 194, 177, 178, 169, 3, 42, 32, 187, 228, 235,
14      79, 210, 248, 246, 23, 112, 177, 111, 164, 166, 59,
15      177, 71, 147, 120, 56, 122, 228, 224, 40, 186, 53,
16      70, 152, 218, 47, 242, 249, 78, 62, 9
17    ],
18    txn: {
19      arcv: [Uint8Array],
20      fee: 1000,
21      fv: 6,
22      gen: 'dockernet-v1',
23      gh: [Uint8Array],
24      lv: 1006,
25      snd: [Uint8Array],
26      type: 'axfer',
27      xaid: 1007
28    }
29  }
30 }
31 (pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/js_api (main) $

```

A tranzakció típus: “axfer”, “asset transfer”, és a “xaid” a “transfer asset id”, és az “arcv”, az “address receiver” az a cím, amelyik az ASA-ra benevezett.

5.8. Csere atomi tranzakciós csoporttal

Az Algorandban az ASA sok mindent képviselhet. Tegyük fel, hogy példánkban egy másik pénzeszközt jelent. Ahhoz, hogy az ASA-t el tudjuk adni Algorand-ért, biztosítani kell, hogy két tranzakció menjen egyszerre végbe, vagy ne menjen végbe egy sem:

- a vevő Algorand-dal fizet az eladónak az ASA-ért
- az eladó elküldi az ASA-t a vevőnek

Az Algorandban a „tranzakció csoport” fogalma biztosítja a fenti cserét.

A fájl explorer-rel vagy a terminálban a `nano 07-atomic_transaction.js` segítségével hozzuk létre a következő fájlt:

```

1 const algosdk = require('algosdk');
2 const fs = require('fs');
3 const DEBUG=0;
4
5 // define sandbox values for kmd client
6 const kmd_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
7 const kmd_server = "http://localhost";
8 const kmd_server_port = 4002;

```

```

9
10 // define sandbox values for algod client
11 const algod_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
12 const algod_server = "http://localhost";
13 const algod_server_port = 4001;
14
15 async function getWalletId(
16     kmdClient,
17     walletName) {
18
19     // list wallets
20     wallets = await kmdClient.listWallets();
21     if(DEBUG) console.log('wallets:', wallets);
22
23     // get wallet index for default wallet
24     let walletId='';
25     wallets.wallets.forEach(item => {
26         if (item.name === walletName) {
27             walletId = item.id;
28         }
29     })
30     return walletId;
31 }
32
33 function getAssetIndex(fname) {
34     var asset_index = undefined;
35     try {
36         var data = fs.readFileSync(fname, 'utf8');
37         asset_index = Number(data);
38         if (DEBUG) console.log(asset_index);
39     } catch (err) {
40         console.error(err);
41     }
42     return asset_index;
43 }
44
45 async function main() {
46     // create kmd client
47     kmd_client = new algosdk.Kmd(kmd_token, kmd_server, kmd_server_port);
48
49     // connect to wallet
50     const wallet_name = 'unencrypted-default-wallet';
51     const wallet_pw = '';
52     wallet_id = await getWalletId(kmd_client, wallet_name);
53     wallet_handle = await kmd_client.initWalletHandle(wallet_id, wallet_pw);
54
55     // gather the first three accounts from the wallet
56     wallet_addresses = await kmd_client.listKeys(wallet_handle.wallet_handle_token);
57     addr1 = wallet_addresses.addresses[0];
58     addr2 = wallet_addresses.addresses[1];
59     addr3 = wallet_addresses.addresses[2];
60
61     // create algod client
62     algod_client = new algosdk.Algodv2(algod_token, algod_server, algod_server_port);
63
64     // get asset index from file
65     asset_index = getAssetIndex('5_asset_index.txt');

```

```

66     if (DEBUG) console.log('asset_index:', asset_index);
67
68     // get params
69     params = await algod_client.getTransactionParams().do();
70     if (DEBUG) console.log('params:', params);
71
72     // build unsigned payment transaction
73     txn_1 = algodsdk.makePaymentTxnWithSuggestedParamsFromObject({
74         suggestedParams: params,
75         from: addr2,
76         to: addr1,
77         amount: algodsdk.algosToMicroalgos(1.0)
78     });
79
80     // build unsigned asset transfer transaction
81     txn_2 = algodsdk.makeAssetTransferTxnWithSuggestedParamsFromObject({
82         suggestedParams: params,
83         from: addr1,
84         to: addr2,
85         assetIndex: asset_index,
86         amount: 100 // this ASA has 2 decimal places, so this is 1.00 FUNTOK
87     });
88
89     // compute group id for transactions
90     gid = algodsdk.computeGroupID([txn_1, txn_2]);
91     txn_1.group = gid;
92     txn_2.group = gid;
93
94     // sign transactions
95     addr2_sk = await kmd_client.exportKey(wallet_handle.wallet_handle_token, wallet_pw, addr2);
96     if (DEBUG) console.log('addr2_sk:', addr2_sk);
97     stxn_1 = await txn_1.signTxn(addr2_sk.private_key);
98
99     addr1_sk = await kmd_client.exportKey(wallet_handle.wallet_handle_token, wallet_pw, addr1);
100    if (DEBUG) console.log('addr1_sk:', addr1_sk);
101    stxn_2 = await txn_2.signTxn(addr1_sk.private_key);
102
103    // assemble transaction group
104    signed_group = [stxn_1, stxn_2];
105
106    // submit atomic transaction group
107    tx_id = await algod_client.sendRawTransaction(signed_group).do();
108    console.log("Successfully sent transaction group with tx_id: %s", tx_id);
109    console.log('tx_id["txId"]:', tx_id['txId']);
110
111    // wait for confirmation
112    console.log('Awaiting confirmation (this may take several seconds)...');
113    const roundTimeout = 7;
114    const confirmed_txn = await algodsdk.waitForConfirmation(
115        algod_client,
116        tx_id['txId'],
117        roundTimeout
118    );
119    console.log('Transaction group is successfully completed');
120
121    // log confirmed transaction
122    if (DEBUG) console.log('confirmed_txn:', confirmed_txn);

```

```

123 // release wallet handle
124 hr = await kmd_client.releaseWalletHandle(wallet_handle['wallet_handle_token']);
125 if (DEBUG) console.log('wallet handle released, hr:', hr)
126 }
127
128
129 main();

```

Futtassuk le a `07-atomic_transaction.js` fájlt:

```

1 (pl playground-py3.10) @A-Maugli → .../akt02/hello/pl playground/js_api (main) $ node
  ↪ 07-atomic_transaction.js
2 Successfully sent transaction group with tx_id: { txId:
  ↪ 'EEFGBYXM3AI CQCPRUAXOJN3CPXI M6GJQ7G5NHI QUTPWKFUS70J7A' }
3 tx_id["txId"]: EEFGBYXM3AI CQCPRUAXOJN3CPXI M6GJQ7G5NHI QUTPWKFUS70J7A
4 Awaiting confirmation (this may take several seconds)...
5 Transaction group is successfully completed
6 (pl playground-py3.10) @A-Maugli → .../akt02/hello/pl playground/js_api (main) $

```

Az `04-account_info.js` segítségével kiíráthatjuk, hogy sikerült-e a csere:

```

1 (pl playground-py3.10) @A-Maugli → .../akt02/hello/pl playground/js_api (main) $ node
  ↪ 04-account_info.js
2 account_info {
3   address: 'DUWWUSWZSLBPGVY6GJ70M5RTUC54EPJ4273FN5QTFELSVCAOQGF62WRUQ',
4   amount: 199999999146000,
5   'amount-without-pending-rewards': 199999999146000,
6   'apps-local-state': [],
7   'apps-total-schema': { 'num-byte-slice': 0, 'num-uint': 0 },
8   assets: [
9     { amount: 100, 'asset-id': 1002, 'is-frozen': false },
10    { amount: 100, 'asset-id': 1007, 'is-frozen': false }
11  ],
12  'created-apps': [],
13  'created-assets': [],
14  'min-balance': 300000,
15  participation: {
16    'selection-participation-key': 'JB7ZLgYDgt54LEZ4wpWEpKZ0swglTFkLI mXprW0yi 3A=',
17    'state-proof-key':
18      ↪ '/Kp6qQ9X2DBrBTI 00BhCzmuDqmdEap/HDPAd9dxI 8YUR5+HYgyQvn8456I mNm7vMMT28XgBSY4em3HI 4b0I i 4A==',
19    'vote-first-valid': 0,
20    'vote-key-dilution': 10000,
21    'vote-last-valid': 30000,
22    'vote-participation-key': 'k1S30rI KTr8D9CoB154NuK0hbi kQSYspUPNdGEI WcMY='
23  },
24  'pending-rewards': 0,
25  'reward-base': 0,
26  rewards: 0,
27  round: 8,
28  status: 'Online',
29  'total-apps-opted-in': 0,
30  'total-assets-opted-in': 2,
31  'total-created-apps': 0,
32  'total-created-assets': 0
33 }
account_info {

```

```

34 address: 'DP07AI GM60H2UREMX2ZPS6SBMRAVEGAAHV43BCTX4SI AOKQBWFFJRZY0E',
35 amount: 4000000000844000,
36 'amount-without-pending-rewards': 4000000000844000,
37 'apps-local-state': [],
38 'apps-total-schema': { 'num-byte-slice': 0, 'num-uint': 0 },
39 assets: [
40   { amount: 9900, 'asset-id': 1002, 'is-frozen': false },
41   { amount: 9900, 'asset-id': 1007, 'is-frozen': false }
42 ],
43 'created-apps': [],
44 'created-assets': [
45   { index: 1002, params: [Object] },
46   { index: 1007, params: [Object] }
47 ],
48 'min-balance': 300000,
49 participation: {
50   'selection-participation-key': 'tSvZrUkGrObS4YTJ7//K/ew56tJGd9r0DmXFSK01To=',
51   'state-proof-key':
52     ↪ 'uG03C1jRdub2CgTsqTNakL4fvDBI i Di wgaUB39NDurOev601wo5U9cPx6t+tBK7i wmgL80I aMHP3eUki sdl p0==',
53   'vote-first-valid': 0,
54   'vote-key-dilution': 10000,
55   'vote-last-valid': 30000,
56   'vote-participation-key': 'EPG0+Y9oj VI gfstV72u8U4sCPLYLgqhys9XV0I zudrM='
57 },
58 'pending-rewards': 0,
59 'reward-base': 0,
60 rewards: 0,
61 round: 8,
62 status: 'Online',
63 'total-apps-opted-in': 0,
64 'total-assets-opted-in': 2,
65 'total-created-apps': 0,
66 'total-created-assets': 2
67 }
68 account_info {
69   address: 'ICNK7LCUJZEULNYG3SEKZ5LGM5J3H2TTI BKYAAUPXACUEKHW33DGXVQA74',
70   amount: 4000000000000000,
71   'amount-without-pending-rewards': 4000000000000000,
72   'apps-local-state': [],
73   'apps-total-schema': { 'num-byte-slice': 0, 'num-uint': 0 },
74   assets: [],
75   'created-apps': [],
76   'created-assets': [],
77   'min-balance': 100000,
78   participation: {
79     'selection-participation-key': '2m3w5vi Ss8ZXPhW0+Mns+z8oX3dkJEH3M+DbJOWtN/U=',
80     'state-proof-key':
81       ↪ 'e9zi 9f7feDLyGhG4R0oI j ddri eRHGpRMqZQ+7WeFuY0FAI ca07snXb68TdI 5b+Fz2hGNI 8hE8qbA0SsNhopp/Q==',
82     'vote-first-valid': 0,
83     'vote-key-dilution': 10000,
84     'vote-last-valid': 30000,
85     'vote-participation-key': 'ERZwnl HHI hb37ERB93xwo6ej SbA3TyAtggzegnyl S8='
86   },
87   'pending-rewards': 0,
88   'reward-base': 0,
89   rewards: 0,
90   round: 8,

```

```

89   status: 'Online',
90   'total-apps-opted-in': 0,
91   'total-assets-opted-in': 0,
92   'total-created-apps': 0,
93   'total-created-assets': 0
94 }
95 (pl ayground-py3.10) @A-Maugli → .../akt02/hello/pl ayground/js_api (main) $

```

A 41. sorokban látszik, hogy a DP07A...YY0E címre az 1007-es “asset-id”-ből 1,00 FUNTOK megérkezett, ami a 2 tizedesjegy miatt 100-nak látszik.

5.9. Node.js Web-es alkalmazások

A Web-es fejlesztés történhet egyszerű HTML használatával, vagy különféle keretrendszerek (pl. React, Angular, Vue stb.) használatával.

Akár egyszerű HTML-ben történik a fejlesztés, akár keretrendszerrel, a közös ezekben a webfejlesztésekben az, hogy a sok-sok Node.js modult valamilyen „web packer” program segítségével néhány nagyobb fájlba pakolják össze, és ezeket az összepakolt fájlokat egy webszerver program segítségével teszik elérhetővé a böngésző számára.

Fejlesztői üzemmódban a fájlok összepakolása és Web szerveren keresztül történő elérése „azonnal”, a forrás minden egyes módosítása után megtörténik. A végleges változat elkészítéséhez a „web packer” alaposabb elemzést végez, és optimalizálja az összepakolt könyvtárak méretét.

5.9.1. Pera WalletConnect mintapélda

A WalletConnect Node.js modul feladata, hogy egy alkalmazás kapcsolatba tudjon lépni a felhasználó pénztárcájával, és a felhasználó engedélyével különféle műveleteket végezhesen.

Az Algorand Pera pénztárcához elkészített Pera Connect SDK segítségével különféle műveleteket lehet kiadni: pl. connect, disconnect, reconnect, payment transaction stb. A Pera Connect dokumentációja a <https://docs.perawallet.app/references/pera-connect> címen található. A Pera mintapéldákat is adott a Pera Connect-hez. A plain vanilla Javascript mintapélda továbbfejlesztésével született az alábbi kis minta, amely a https://github.com/A-Maugli/pera-connect-vanillajs-demo_mod5 repository-ban érhető el.

A fejlesztés Node.js alatt történt. A package.json fájl tartalma:

```

1 | i pi@i pi -Virtual Box: ~/Downloads/pera-connect-vanillajs-demo_mod5$ cat package.json
2 | {
3 |   "name": "perawallet-connect-vanillajs-demo",
4 |   "version": "1.0.0",
5 |   "description": "Pera WalletConnect demo on Algorand Testnet",
6 |   "scripts": {
7 |     "start": "parcel serve ./src/index.html --open --dist-dir ./dist/bundled_noopt",
8 |     "build": "parcel build ./src/index.html --no-scope-hoist --dist-dir ./dist/bundled && serve
   |     ↪ ./dist/bundled",
9 |     "clean": "rm -rf ./parcel-cache ./dist ./node_modules ./package-lock.json"
10 |   },
11 |   "dependencies": {
12 |     "@perawallet/connect": "^1.3.4",
13 |     "algorand": "^2.7.0",

```

```

14   "jquery": "^3.7.1",
15   "parcel": "^2.11.0"
16 },
17 "keywords": [
18   "@perawallet/connect",
19   "Algorand Testnet",
20   "parcel v2"
21 ],
22 "devDependencies": {
23   "process": "^0.11.10"
24 }
25 }

```

A használt NPM modulok: @perawallet/connect, algosdk és jquery. A használt Web packer a parcel csomag 2.11.0 verziója.

A megvalósított npm script-ek:

1. `npm run start` – interaktív web packer + web szerver indítás
2. `npm run build` – web packer optimalizált build indítás, külön web szerver indítás
3. `npm run clean` – a nem feltétlenül szükséges fájlok törlése

Egy lehetséges munkamenet:

```

1  | l i p i @ l i p i - V i r t u a l   B o x :   ~ / D o w n l o a d s / p e r a - c o n n e c t - v a n i l l a j s - d e m o _ m o d 5 $   n p m   r u n   c l e a n
2
3  | >   p e r a w a l l e t - c o n n e c t - v a n i l l a j s - d e m o @ 1 . 0 . 0   c l e a n
4  | >   r m   - r f   . / .   p a r c e l - c a c h e   . / d i s t   . / n o d e _ m o d u l e s   . / p a c k a g e - l o c k . j s o n
5
6  | n p m   n o t i c e
7  | n p m   n o t i c e   N e w   m i n o r   v e r s i o n   o f   n p m   a v a i l a b l e !   1 0 . 2 . 4   - >   1 0 . 4 . 0
8  | n p m   n o t i c e   C h a n g e l o g :   h t t p s : / / g i t h u b . c o m / n p m / c l i / r e l e a s e s / t a g / v 1 0 . 4 . 0
9  | n p m   n o t i c e   R u n   n p m   i n s t a l l   - g   n p m @ 1 0 . 4 . 0   t o   u p d a t e !
10 | n p m   n o t i c e
11 | l i p i @ l i p i - V i r t u a l   B o x :   ~ / D o w n l o a d s / p e r a - c o n n e c t - v a n i l l a j s - d e m o _ m o d 5 $   n p m   i n s t a l l
12 | n p m   W A R N   d e p r e c a t e d   @ w a l l e t c o n n e c t / t y p e s @ 1 . 8 . 0 :   W a l l e t C o n n e c t ' s   v 1   S D K s   a r e   n o w   d e p r e c a t e d .   P l e a s e
   | ↪   u p g r a d e   t o   a   v 2   S D K .   F o r   d e t a i l s   s e e :   h t t p s : / / d o c s . w a l l e t c o n n e c t . c o m /
13 | n p m   W A R N   d e p r e c a t e d   s t a b l e @ 0 . 1 . 8 :   M o d e r n   J S   a l r e a d y   g u a r a n t e e s   A r r a y # s o r t ( )   i s   a   s t a b l e   s o r t ,   s o
   | ↪   t h i s   l i b r a r y   i s   d e p r e c a t e d .   S e e   t h e   c o m p a t i b i l i t y   t a b l e   o n   M D N :   h t t p s : / / d e v e l o p e r . m o z i l l a . o r g / j
   | ↪   e n - U S / d o c s / W e b / J a v a S c r i p t / R e f e r e n c e / G l o b a l _ O b j e c t s / A r r a y / s o r t # b r o w s e r _ c o m p a t i b i l i t y
14 | n p m   W A R N   d e p r e c a t e d   @ w a l l e t c o n n e c t / c l i e n t @ 1 . 8 . 0 :   W a l l e t C o n n e c t ' s   v 1   S D K s   a r e   n o w   d e p r e c a t e d .
   | ↪   P l e a s e   u p g r a d e   t o   a   v 2   S D K .   F o r   d e t a i l s   s e e :   h t t p s : / / d o c s . w a l l e t c o n n e c t . c o m /
15
16 | a d d e d   2 5 6   p a c k a g e s ,   a n d   a u d i t e d   2 5 7   p a c k a g e s   i n   4 0 s
17
18 | 9 1   p a c k a g e s   a r e   l o o k i n g   f o r   f u n d i n g
19 |   r u n   n p m   f u n d   f o r   d e t a i l s
20
21 | f o u n d   0   v u l n e r a b i l i t i e s
22 | l i p i @ l i p i - V i r t u a l   B o x :   ~ / D o w n l o a d s / p e r a - c o n n e c t - v a n i l l a j s - d e m o _ m o d 5 $   d u   - s h
23 | 3 9 1 M
24 | l i p i @ l i p i - V i r t u a l   B o x :   ~ / D o w n l o a d s / p e r a - c o n n e c t - v a n i l l a j s - d e m o _ m o d 5 $   n p m   r u n   b u i l d
25
26 | >   p e r a w a l l e t - c o n n e c t - v a n i l l a j s - d e m o @ 1 . 0 . 0   b u i l d
27 | >   p a r c e l   b u i l d   . / s r c / i n d e x . h t m l   -- n o - s c o p e - h o i s t   -- d i s t - d i r   . / d i s t / b u n d l e d   & &   s e r v e   . / d i s t / b u n d l e d

```



```

28
29 ★ Built in 5.73s
30
31 dist/bundled/index.html          1.66 KB    1.23s
32 dist/bundled/index.0ed18bef.js   667.56 KB  1.41s
33 dist/bundled/App-94e9365e.cefc48d1.js 320.25 KB  1.66s
34 dist/bundled/index.runtime.7745c7dc.js 2.21 KB    592ms
35
36
37
38
39
40 - Local:    http://localhost:3000
41 - Network:  http://10.0.2.15:3000
42
43 Copied local address to clipboard!
44
45
46
47 HTTP 2/28/2024 3:55:57 PM 127.0.0.1 GET /
48 HTTP 2/28/2024 3:55:57 PM 127.0.0.1 Returned 304 in 47 ms
49 HTTP 2/28/2024 3:55:57 PM 127.0.0.1 GET /index.runtime.7745c7dc.js
50 HTTP 2/28/2024 3:55:57 PM 127.0.0.1 Returned 304 in 3 ms
51 HTTP 2/28/2024 3:55:57 PM 127.0.0.1 GET /index.0ed18bef.js
52 HTTP 2/28/2024 3:55:57 PM 127.0.0.1 Returned 304 in 8 ms
53 HTTP 2/28/2024 3:55:57 PM 127.0.0.1 GET /App-94e9365e.cefc48d1.js
54 HTTP 2/28/2024 3:55:57 PM 127.0.0.1 Returned 304 in 4 ms
55 ^C
56 INFO Gracefully shutting down. Please wait...

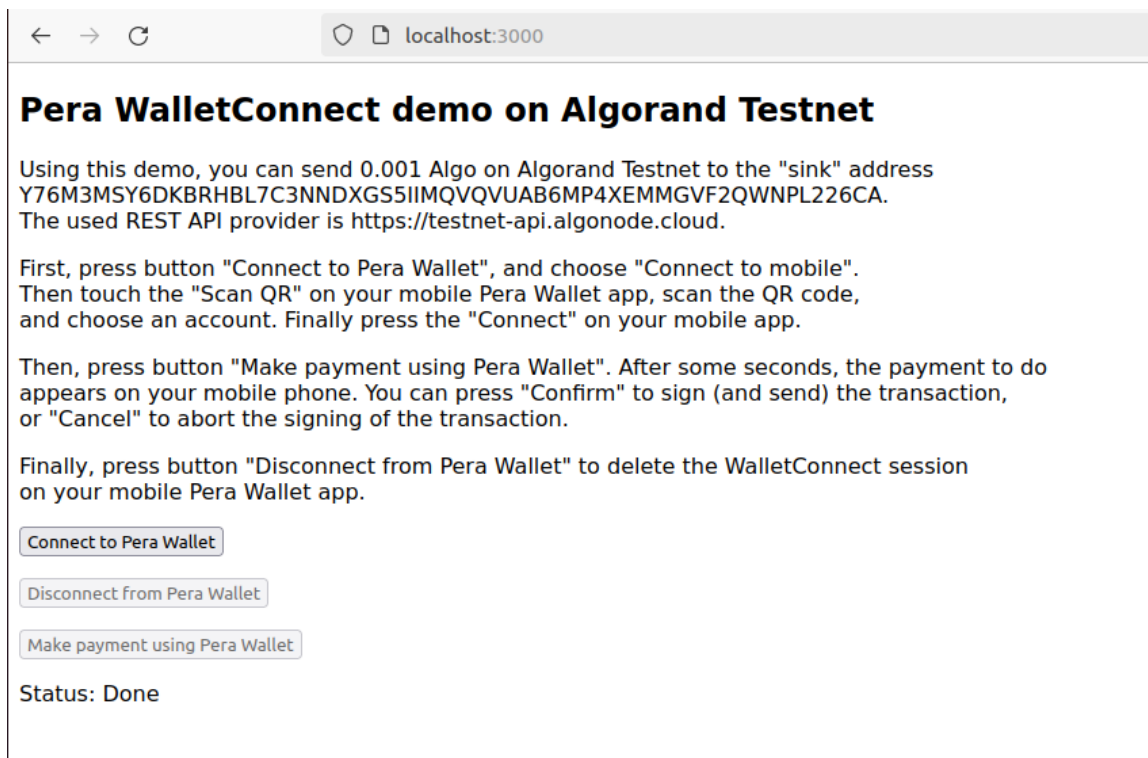
```

A munkamenet magyarázata:

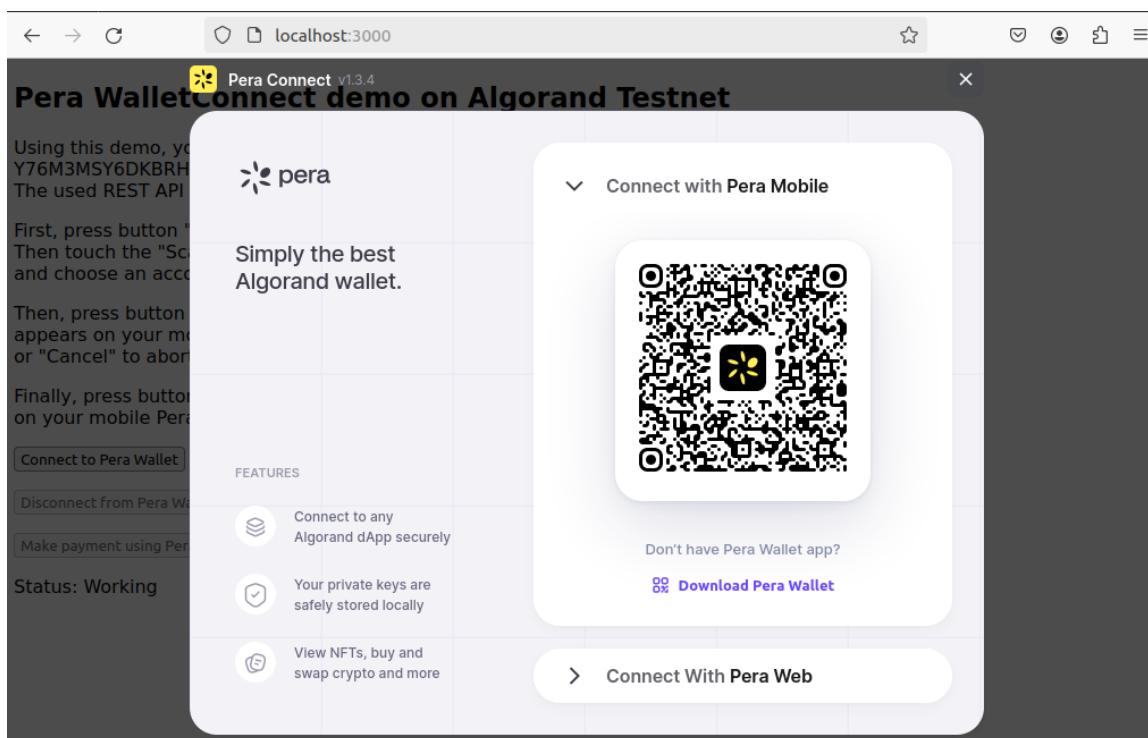
- 1. sor: törölünk mindet, amit csak lehet
- 11. sor: installáljuk a `node_modules` könyvtárba a Node.js modulokat
- 22. sor: kiíratjuk, hogy mennyi diszk területet használ a projektünk. Apait-anyait magához rántott, mert 391 Mbyte lett az összesített mérete.
- 24. sor: felépítjük az optimalizált, összepakolt alkalmazást, és elindítunk egy web szervert

A böngészőben elindított demó alkalmazás az 1. ábrán látható. Funkciói:

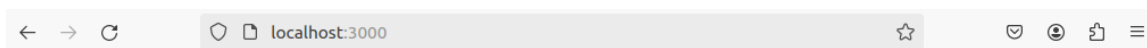
- “Connect to Pera Wallet” – a demó alkalmazás összekapcsolása a Pera pénztárcával. A gomb megnyomása után választani tudunk, hogy a demó alkalmazás egy Web-es pénztárcával vagy egy iOS/Android pénztárcával kapcsolódjon össze. Válasszuk a “Connect to mobile” („Kapcsolódás mobil pénztárcához”) lehetőséget. Ekkor a képernyőn megjelenik egy QR kód, lásd 2. ábra. A mobilunkon nyissuk meg a Pera pénztárcát, és pásztázzuk be a QR kódot. Válasszunk ki a mobilon egy számlát, amit szeretnénk összekapcsolni a Pera WalletConnect demó alkalmazással.
- “Make payment using PeraWallet” – fizetési művelet végzése, 0,001 Algo küldése az Algorand „mindent elnyelő” címére, lásd 3. ábra. A gomb megnyomása után pár másodperccel megjelenik egy fizetési tranzakciós képernyő a mobilunkon. A tranzakció a “Confirm” („Megerősítés”) gomb megnyomásával írható alá és küldhető el, a “Cancel” („Elhalasztás”) gomb megnyomásával pedig



1. ábra. A Pera WalletConnect demó alkalmazás induláskor



2. ábra. Kapcsolódás a mobil Pera pénztárcához



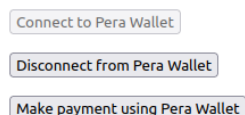
Pera WalletConnect demo on Algorand Testnet

Using this demo, you can send 0.001 Algo on Algorand Testnet to the "sink" address Y76M3MSY6DKBRHBL7C3NNDXG55IIMQVQVUAB6MP4XEMMGVF2QWNPL226CA. The used REST API provider is <https://testnet-api.algonode.cloud>.

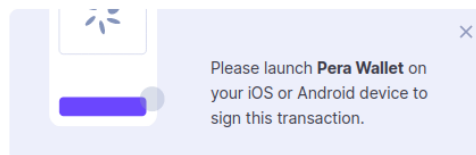
First, press button "Connect to Pera Wallet", and choose "Connect to mobile". Then touch the "Scan QR" on your mobile Pera Wallet app, scan the QR code, and choose an account. Finally press the "Connect" on your mobile app.

Then, press button "Make payment using Pera Wallet". After some seconds, the payment to do appears on your mobile phone. You can press "Confirm" to sign (and send) the transaction, or "Cancel" to abort the signing of the transaction.

Finally, press button "Disconnect from Pera Wallet" to delete the WalletConnect session on your mobile Pera Wallet app.



Status: Working



3. ábra. Fizetési tranzakció kezdeményezése

törölhető.

- “Disconnect from Pera Wallet” – a WalletConnect munkamenet törlését végzi. A munkamenet a mobilon is törölhető, a Settings | WalletConnect Sessions részben.

Az demó alkalmazás forrása a `src` könyvtárban van. Az `src/index.html` tartalma:

```
1 | l i p i @ l i p i - V i r t u a l B o x : - / D o w n l o a d s / p e r a - c o n n e c t - v a n i l l a j s - d e m o _ m o d 5 / s r c $ c a t i n d e x . h t m l
2 | < ! D O C T Y P E h t m l >
3 | < h t m l >
4 |
5 | < h e a d >
6 |   < t i t l e > P e r a W a l l e t C o n n e c t d e m o o n A l g o r a n d T e s t n e t < / t i t l e >
7 |   < m e t a c h a r s e t = " U T F - 8 " >
8 |   < s t y l e >
9 |     d i v . c 1 {
10 |       m a r g i n - t o p : 1 e m ;
11 |       m a r g i n - b o t t o m : 1 e m ;
12 |     }
13 |
14 |     b o d y {
15 |       f o n t - f a m i l y : s a n s - s e r i f ;
16 |     }
17 |   < / s t y l e >
18 |   < s c r i p t t y p e = " m o d u l e " s r c = ". / p e r a _ w a l l e t c o n n e c t _ d e m o . j s " > < / s c r i p t >
19 | < / h e a d >
20 |
21 | < b o d y >
22 |   < h 2 > P e r a W a l l e t C o n n e c t d e m o o n A l g o r a n d T e s t n e t < / h 2 >
23 |   < p >
24 |     U s i n g t h i s d e m o , y o u c a n s e n d 0 . 0 0 1 A l g o o n A l g o r a n d T e s t n e t t o t h e " s i n k " a d d r e s s < b r / >
25 |     Y 7 6 M 3 M S Y 6 D K B R H B L 7 C 3 N N D X G S 5 I I M O V Q V U A B 6 M P 4 X E M M G V F 2 0 W N P L 2 2 6 C A . < b r / >
26 |     T h e u s e d R E S T A P I p r o v i d e r i s https://testnet-api.algonode.cloud.
27 |   < / p >
28 |   < p >
29 |     F i r s t , p r e s s b u t t o n " C o n n e c t t o P e r a W a l l e t " , a n d c h o o s e " C o n n e c t t o m o b i l e " . < b r / >
30 |     T h e n t o u c h t h e " S c a n Q R " o n y o u r m o b i l e P e r a W a l l e t a p p , s c a n t h e Q R c o d e , < b r / >
31 |     a n d c h o o s e a n a c c o u n t . F i n a l l y p r e s s t h e " C o n n e c t " o n y o u r m o b i l e a p p .
32 |   < / p >
33 |   < p >
34 |     T h e n , p r e s s b u t t o n " M a k e p a y m e n t u s i n g P e r a W a l l e t " . A f t e r s o m e s e c o n d s , t h e p a y m e n t t o d o
35 |     ↪ < b r / >
36 |     a p p e a r s o n y o u r m o b i l e p h o n e . Y o u c a n p r e s s " C o n f i r m " t o s i g n ( a n d s e n d ) t h e t r a n s a c t i o n , < b r / >
37 |     o r " C a n c e l " t o a b o r t t h e s i g n i n g o f t h e t r a n s a c t i o n .
38 |   < / p >
39 |   < p >
40 |     F i n a l l y , p r e s s b u t t o n " D i s c o n n e c t f r o m P e r a W a l l e t " t o d e l e t e t h e W a l l e t C o n n e c t s e s s i o n < b r / >
41 |     o n y o u r m o b i l e P e r a W a l l e t a p p .
42 |   < / p >
43 |   < d i v c l a s s = " c 1 " >
44 |     < b u t t o n i d = " c o n n e c t _ p e r a _ w a l l e t " > C o n n e c t t o P e r a W a l l e t < / b u t t o n >
45 |   < / d i v >
46 |   < d i v c l a s s = " c 1 " >
47 |     < b u t t o n i d = " d i s c o n n e c t _ p e r a _ w a l l e t " > D i s c o n n e c t f r o m P e r a W a l l e t < / b u t t o n >
48 |   < / d i v >
49 |   < d i v c l a s s = " c 1 " >
50 |     < b u t t o n i d = " m a k e _ p a y m e n t _ p e r a _ w a l l e t " > M a k e p a y m e n t u s i n g P e r a W a l l e t < / b u t t o n >
51 |   < / d i v >
52 |   < p a r i a - l a b e l = " S t a t u s o f l a s t c o m m a n d " i d = " w a l l e t _ c o n n e c t _ s t a t u s " > S t a t u s : < / p >
53 |   < p a r i a - l a b e l = " O p t i o n a l e r r o r m e s s a g e " i d = " w a l l e t _ c o n n e c t _ e r r o r " > E r r o r : < / p >
54 | < / b o d y >
```

55 </html >

Az `index.html` a weblapot írja le. Az ismertető szövege `<p>` tagok között van, ezt követi a funkciókhoz tartozó három nyomógomb létrehozása a `<button>` segítségével. A böngésző által megjelenített weblapot az 1. ábra mutatja.

A nyomógombokat kezelő Javascript a `src/pera_walletconnect_demo.js` fájlban található.

```
1 | ipi@ipi -Virtual Box: ~/Downloads/pera-connect-vanilla/js-demo_mod5/src$ cat
   | → pera_walletconnect_demo.js
2 | "use strict";
3 |
4 | import algosdk from "algosdk";
5 | //import { PeraWalletConnect } from "@perawallet/connect";
6 | import * as pwcsdk from "@perawallet/connect";
7 | import $ from "jquery";
8 |
9 | const working = "Status: Working";
10 | const done = "Status: Done";
11 |
12 | const peraWallet = new pwcsdk.PeraWalletConnect();
13 |
14 | const token = "";
15 | const server = "https://testnet-api.algonode.cloud";
16 | const port = 443;
17 | const client = new algosdk.Algodv2(token, server, port);
18 |
19 | let accountAddress = "";
20 | const sinkAddress = "Y76M3MSY6DKBRHBL7C3NNDXGS5IIMOVQVUAB6MP4XEMMGVF2QWNPL226CA";
21 |
22 |
23 | $(window.document).ready(function () {
24 |   console.log("document ready");
25 |   addEventListeners();
26 |   reconnectSession();
27 | });
28 |
29 | function addEventListeners() {
30 |   $("#connect_pera_wallet").on("click", function () {
31 |     handleConnectWallet();
32 |   });
33 |
34 |   $("#disconnect_pera_wallet").on("click", function () {
35 |     handleDisconnectWallet();
36 |   });
37 |
38 |   $("#make_payment_pera_wallet").on("click", function () {
39 |     handlePayment();
40 |   })
41 | }
42 |
43 | function reconnectSession() {
44 |   setStatus(working);
45 |   clearError();
46 |   // Reconnect to the session when the component is mounted
47 |   peraWallet
```

```

48 .reconnectSession()
49 .then((accounts) => {
50   if (peraWallet.connector !== null) {
51     peraWallet.connector.on("disconnect", handleDisconnectWallet);
52   }
53   if (accounts.length) {
54     accountAddress = accounts[0];
55     setButtonState(true);
56   }
57   else {
58     setButtonState(false);
59   }
60   setStatus(done);
61 })
62 .catch((error) => {
63   console.log('Error in reconnectSession: ' + error.name + ' ' + error.message);
64   setStatus(done);
65   setError(error);
66 });
67 }
68
69 function setButtonState(connected) {
70   if (connected) {
71     $("#connect_perawallet").prop('disabled', true);
72     $("#disconnect_perawallet").prop('disabled', false);
73     $("#make_payment_perawallet").prop('disabled', false);
74   }
75   else {
76     $("#connect_perawallet").prop('disabled', false);
77     $("#disconnect_perawallet").prop('disabled', true);
78     $("#make_payment_perawallet").prop('disabled', true);
79   }
80 }
81
82 function setStatus(msg) {
83   $("#wallet_connect_status").text(msg);
84 }
85
86 function setError(error) {
87   const error_name = error.name;
88   const error_message = error.message.replace(/\n/g, '<br/>');
89   if (error_message !== "") {
90     $("#wallet_connect_error").html('Error: ' + error_name + ' ' + error_message);
91     $("#wallet_connect_error").show(0);
92   }
93   else {
94     $("#wallet_connect_error").text('No errors');
95     $("#wallet_connect_error").hide();
96   }
97 }
98
99 function clearError() {
100   setError({ name: '', message: '' })
101 }
102
103 function handleConnectWallet() {
104   setStatus(working);

```

```

105 clearError();
106 peraWallet
107   .connect()
108   .then((newAccounts) => {
109     peraWallet.connector.on("disconnect", handleDisconnectWallet);
110     accountAddress = newAccounts[0];
111     setButtonState(true);
112     setStatus(done);
113   })
114   .catch((error) => {
115     if (error?.data?.type !== "CONNECT_MODAL_CLOSED") {
116       console.log('Error in handleConnectWallet: ' + error.name + ' ' + error.message);
117       setStatus(done);
118       setError(error);
119     }
120   });
121 }
122
123 function handleDisconnectWallet() {
124   setStatus(working);
125   clearError();
126   peraWallet
127     .disconnect()
128     .then(() => {
129       setButtonState(false);
130       setStatus(done);
131     })
132     .catch((error) => {
133       console.log('Error in handleDisconnectWallet: ' + error.name + ' ' + error.message);
134       setStatus(done);
135       setError(error);
136     });
137   accountAddress = "";
138 }
139
140 async function handlePayment() {
141   setStatus(working);
142   clearError();
143   const params = await client.getTransactionParams().do();
144   const txn = algosdk.makePaymentTxnWithSuggestedParamsFromObject({
145     from: accountAddress,
146     to: sinkAddress,
147     amount: algosdk.algosToMicroalgos(0.001),
148     suggestedParams: params
149   });
150   let txGroup = [{ txn, signers: [accountAddress] }];
151   try {
152     const signedTxnGroup = await peraWallet.signTransaction([txGroup]);
153     const { txId } = await client.sendRawTransaction(signedTxnGroup).do();
154     console.log('Info in handlePayment: ', 'Payment txn sent to Algorand network');
155     console.log('Info in handlePayment: ', 'txId: ', txId);
156     setStatus(done);
157   } catch (error) {
158     console.log('Error in handlePayment: ' + error.name + ' ' + error.message);
159     setStatus(done);
160     setError(error);
161   }

```

Mikor DOM felépül, akkor megtörténik a nyomógombok eseménykezelőinek beállítása (25. sor, 29..41. sor) és a legutolsó WalletConnect munkamenet visszaállítása (26. sor, 43..67. sor). A gombok engedélyezése/tiltása a 69..80. sorokban történik, aszerint, hogy volt-e már sikeres kapcsolódás a pénztárcával. A státusz üzenet kiírása a 82..84. sorokban történik, az esetleges hibaüzenet kiírása a 86..97. sorokban. A pénztárcával való összekapcsolódás kezelése a 103..121. sorokban van, a szétkapcsolás kezelése a 123..138. sorokban. A fizetési tranzakció kezelése a 140..162. sorokban látható.

5.10. Közvetlenül a Web böngészőre támaszkodó alkalmazásfejlesztés

Ez a fajta alkalmazásfejlesztés kihagyja a Node.js szintet. A HTML, CSS és Javascript fájlokat a fejlesztő közvetlenül a Web böngészőben futtathatja. A használt Javascript könyvtárakat az alkalmazás összepakolt formájukban hivatkozza meg. A belövéskor a böngésző fejlesztői környezete használható (F12).

Az algosdk JavaScript könyvtár összepakolt formában pl. CDN-ekről is elérhető. A teljes könyvtár kb. 300 Kbyte méretű. Azt, hogy hogyan lehet a @perawallet/connect Node.js modulból könyvtárat készíteni, a következő rész mutatja be.

5.10.1. A @perawallet/connect összepakolása

A @perawallet/connect csak a Node.js alatt érhető el, és nincs meg „összepakolt”, a web böngészők által is fogyasztható formában. A Web böngészőre támaszkodó fejlesztéshez ezt is össze kell pakolni valamelyik ismert „web packer”, pl. a webpack használatával.

A <https://github.com/A-Maugli/pera-connect-vanillajs-demo-webpack> repository-ban található egyfajta megoldás erre a feladatra.

A @perawallet/connect Node.js modult a src/pwc.js fájl hivatkozza meg:

```

1 //import {PeraWalletConnect} from "@perawallet/connect";
2 const pwcsdk = require("@perawallet/connect");
3 exports.pwcsdk = pwcsdk;
4
5 const algosdk = require("algosdk");
6 exports.algosdk = algosdk;
```

A fájl kiexportálja mind a Pera wallet connect SDK-t, lásd pwcsdk, mind az Algorand JS SDK-t, lásd algosdk.

A webpack konfigurációs állománya, a webpack.config.js a következő:

```

1 const path = require('path');
2
3 module.exports = {
4   mode: 'production',
5   entry: './src/pwc.js',
6   output: {
7     filename: 'perawalletconnect.min.js',
8     path: path.resolve(__dirname, 'dist/browser'),
9     library: {
```



```

10     type: 'umd',
11     name: 'pwc',
12   },
13 },
14 devtool: 'source-map',
15 resolve: {
16   // Add '.ts' as resolvable extensions
17   extensions: ['.ts', '.js'],
18 },
19 module: {
20   rules: [
21     // All files with a '.ts' extension will be handled by 'ts-loader'.
22     {
23       test: /\.ts$/,
24       loader: 'ts-loader',
25       options: {
26         configFile: path.resolve(__dirname, 'tsconfig-browser.json'),
27       },
28     },
29
30     // All output '.js' files will have any sourcemaps re-processed by 'source-map-loader'.
31     { test: /\.js$/, loader: 'source-map-loader' },
32   ],
33   // Don't parse tweetnacl module -
34   // ↪ https://github.com/dchest/tweetnacl-js/wiki/Using-with-Webpack
35   noParse: [/[\\/]tweetnacl[\\/]$/, /[\\/]tweetnacl-auth[\\/]$/],
36 };

```

Az 'entry' belépési pontot a könyvtárba a `./src/pwc.js` fájl definiálja (5. sor). A könyvtár típusa 'umd' lesz (univerzálisan elérhető modul), a neve pedig 'pwc', ami a Pera wallet connect rövidítéséből adódott (10. és 11. sor). Az eredmény a `dist/browser` könyvtárban áll elő (8. sor).

Az összepakolt könyvtár a `public/test.html` segítségével tesztelhető:

```

1  <!DOCTYPE html >
2  <html >
3
4  <head>
5    <title>Test for @perawallet/connect packed JavaScript library</title>
6    <meta charset="UTF-8">
7    <!--
8    <script type="module" src="./browser/algosdk.min.js"></script>
9    -->
10   <!-- Note: perawalletconnect.min.js contains both pwcsdk and algosdk -->
11   <!--       but algosdk version is v2.1 -->
12   <script type="module" src="./jslib/perawalletconnect.min.js"></script>
13 </head>
14
15 <body>
16 <h2>Test for @perawallet/connect packed JavaScript library</h2>
17 <p>Use F12 to see Console</p>
18 <p>Expected: </p>
19 <p>success: Pera wallet connection created</p>
20 <p>success: algosdk client created</p>
21 <script>

```

```

22 window.onload = function(e) {
23   const pwcSdk = pwc.pwcSdk;
24   const algoSdk = pwc.algoSdk;
25
26   // Create pera wallet connection
27   try {
28     const peraWallet = new pwcSdk.PeraWalletConnect();
29     console.log(' success: Pera wallet connection created');
30   }
31   catch (e) {
32     console.log(' error: ', e);
33   }
34
35   // Create algo client
36   const token = "";
37   const server = "https://testnet-api.algonode.cloud";
38   const port = 443;
39   try {
40     const client = new algoSdk.Algodv2(token, server, port);
41     console.log(' success: algo client created');
42   }
43   catch (e) {
44     console.log(' error: ', e);
45   }
46 }
47 </script>
48 </body>
49 </html >

```

Az összepakolt könyvtárra történő hivatkozás a böngészőkben megszokott módon, a `<script>...</script>` paranccsal történik (12. sor). A könyvtár tesztelését a 21..47. sorok közötti Javascript végzi.

Megvárjuk, amíg a DOM előáll, vagyis a `window.onload` eseménykezelőbe tesszük a további teszt sorokat. Itt előállítjuk a `pwc` objektumból a `pwcSdk` és `algoSdk` objektumot (23. és 24. sor), majd kiadunk egy `PeraWalletConnect()` hívást (28. sor), és naplózzuk a siker vagy hiba tényét a konzolon.

A `pwc`-be az Algorand JS SDK 2.1 változata is belekerült a `@perawallet/connect` függőségei miatt. A teszt azt is bemutatja, hogy az összepakolt könyvtárral valóban létre tudunk hozni egy Algorand klienst is (40. sor).

Egy másik példa a teljes előző pera wallet connect demót mutatja be újra. A hozzá tartozó két fájl: a `public/index.html` és a `public/pera_walletconnect_demo.js` fájlok. A `public/index.html` az összepakolt könyvtárat a következőképpen hivatkozza:

```

1 <script type="module" src="./js/lib/perawalletconnect.min.js"
2   integrity="sha384-1/BfpY6oNI kbLhI Q2HqXVz2NzZb2zw5D6i Dz6Qkdi 1E6dCxZyJm1+NN9SMLaml Xm"
3   crossorigin="anonymous"></script>

```

Itt már megadtuk a könyvtár épségének az ellenőrzésére a könyvtár sha384-es ellenőrző összegét is.

A `public/pera_walletconnect_demo.js` fájlnak csak az eleje más a Node.js változathoz képest:

```

1 "use strict";
2
3 //import algosdk from "algosdk";
4 ///import { PeraWalletConnect } from "@perawallet/connect";
5 //import * as pwcsdk from "@perawallet/connect";
6 //import $ from "jquery";
7
8 const working = "Status: Working";
9 const done = "Status: Done";
10
11 const pwcsdk = pwc.pwcsdk;
12 const perawallet = new pwcsdk.PeraWalletConnect();
13 ...

```

5.10.2. Csocsó eredmények rögzítése az Algorand Testnet blokkláncon

Az Algorand tranzakciók **note** mezőjében max. 1 Kbyte információ adható meg. Egy Algorand tranzakció költsége 0,001 Algó, ami a jelenlegi Algó árfolyam mellett kb. 10 filléres díjnak felel meg.² A tranzakciók költsége csak akkor nagyobb, ha hálózati torlódás van. Az Algorand hálózat jelenleg 7500 tps sebességű, ahol a tps a tranzakció per sec rövidítése. Mivel a hálózati forgalom jelenleg 50..100 tps, nagyon valószínűtlen a 0,001 Algónál nagyobb díj. Egyébként is, az alkalmazás a Testnet blokkláncon fut, ahol ingyen lehet teszt Algókhoz jutni.

A demó alkalmazás a <https://github.com/A-Maugli/csocso-vanillajs-pwc> repository-ban van, a **src** könyvtárban. Az **index.html** meghivatkozza a használt Javascript könyvtárakat, ezek az **algosdk.min.js**, a **jquery-3.7.1.min.js**, és a **perawalletconnect.min.js** könyvtárak. A HTML fájl két része: a **Store Game Result** (23..35. sor) és a **Read Game Results** (41..47. sor).

A HTML felületen megadható a játékosok neve és az eredmény. A **Store game result** gomb megnyomásakor az alkalmazás a **JW6L2Z...FNKILM** számláról a **UU0B7Z...C7Q62E** „univerzális nyelű” számlára elküld 0,001 Algót, és a tranzakció **note** mezőjébe JSON formátumban beírja a mérkőzés eredményét. A fizetési tranzakció a Pera wallet connect segítségével történik, biztonságos módon, így a demó alkalmazás sehol sem tartalmazza a **JW6L2Z...FNKILM** forrás számla privát kulcsát.

A **Read game results** gomb megnyomásakor egy keresési műveletet küldünk az indexernek, és visszaolvassuk azokat a tranzakciókat, amelyekben a forrás a **JW6L2Z...FNKILM** számla volt. Szűrést végzünk, és csak azokat a tranzakciókat írjuk ki, amelyekben a **note** mezőben lévő JSON struktúra a megfelelő formátumú.

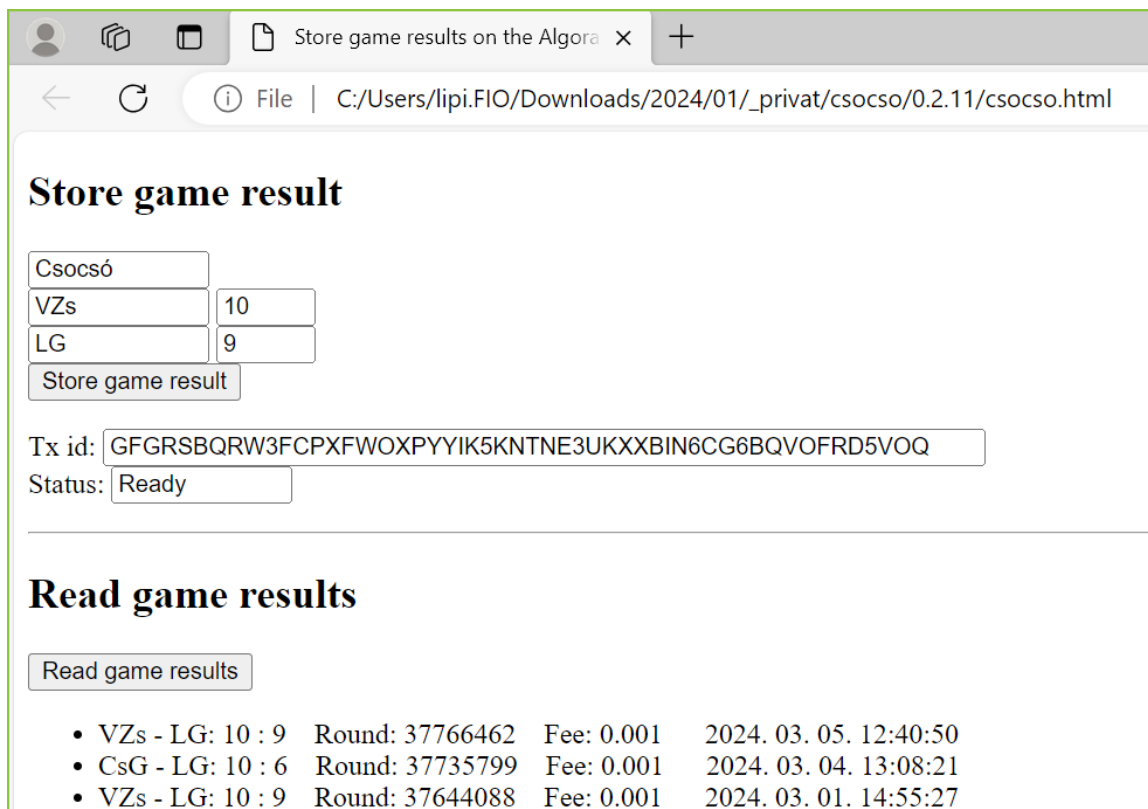
A demót megvalósító **csocso.js** tartalma:

```

1 "use strict";
2
3 //const algod = require('algosdk');
4 //const pwcsdk = require('@perawallet/connect');
5 //const $ = require('jquery');
6
7 const ver = "0.2.12"; // Use PeraWalletConnect();
8

```

² A tranzakciós díj a jövőben változhat. Érdemes az SDK konstansokat használni, pl. a JS SDK-ban ezt: `const minFee = algosdk.ALGORAND_MIN_TX_FEE`



4. ábra. A Csocsó demó alkalmazás felhasználói felülete

```

9  /* Algonode.io API */
10 const baseServer = "https://testnet-api.algonode.cloud";
11 const baseServerIdx = "https://testnet-idx.algonode.cloud";
12 const port = 443;
13 const token = "";
14
15 const working = "Working";
16 const ready = "Ready";
17 var addr_src, addr_dst;
18 var account_addr = ''; // address selected on PeraWallet
19
20 const testing = false;
21 if (testing) {
22     debugger;
23     addr_src = 'CDRLCYZICK7XEBZ7M4HKYWNB7ZZL04BF005RGZNMH05ZH7EZWUDF06Z32U'; // Test account 1
24     addr_dst = 'UU0B7ZC2IEE4A7J04WY4TXKXWDFNATM43TL73IZRAF1FF0E6ORPKC7Q62E';
25 }
26 else {
27     addr_src = 'JW6L2ZCQT3UIQH5AFM3CVW3C7M3QFYHXA3EU4WHRE0ATRMMXP6MBFNK1LM'; // Csocsó account
28     addr_dst = 'UU0B7ZC2IEE4A7J04WY4TXKXWDFNATM43TL73IZRAF1FF0E6ORPKC7Q62E';
29 }
30
31 const algod_client = new algosdk.Algodv2(token, baseServer, port);
32 console.log('algod_client created');
33 const indexer_client = new algosdk.Indexer(token, baseServerIdx, port);
34 console.log('indexer_client created');

```

```

35
36 const pwcSdk = pwc.pwcSdk;
37 const peraWallet = new pwcSdk.PeraWalletConnect();
38 console.log('peraWallet created');
39
40 async function reconnectSessionA() {
41   let accounts = await peraWallet.reconnectSession();
42   if (peraWallet.connector !== null) {
43     peraWallet.connector.on("disconnect", handleDisconnectWalletA);
44   }
45   return accounts;
46 }
47
48 async function handleConnectWalletA() {
49   let newAccounts = await peraWallet.connect();
50   peraWallet.connector.on("disconnect", handleDisconnectWalletA);
51   return newAccounts;
52 }
53
54 async function handleDisconnectWalletA() {
55   await peraWallet.disconnect();
56   console.log('Info in handleDisconnectWallet: ', 'disconnected');
57   let accountAddress = "";
58   return accountAddress;
59 }
60
61 function algo_send_tx(algodClient, note) {
62   (async () => {
63     // Reconnect/connect wallet
64     let accounts = await reconnectSessionA();
65     if (accounts.length == 0) {
66       accounts = await handleConnectWalletA();
67     }
68     if (accounts.length == 0) {
69       throw ('Wallet connect error');
70     }
71     if (accounts[0] !== addr_src) {
72       await handleDisconnectWalletA();
73       throw ('Please connect to ' + addr_src);
74     }
75     else {
76       account_addr = accounts[0];
77     }
78
79     // get params from algod
80     let params = await algodClient.getTransactionParams().do();
81
82     let obj = {
83       "from": account_addr,
84       "to": addr_dst,
85       "amount": 1,
86       "note": algosdk.encodeObj(note),
87       "suggestedParams": params
88     };
89     let txn = algosdk.makePaymentTxnWithSuggestedParamsFromObject(obj);
90     // sign transaction
91     let txGroup = [{ txn, signers: [account_addr] }];

```



```

149     list += "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;";
150     list += "\t Fee: " + tx.transactions[i].fee / 1000000.0;
151     list += "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;";
152     if (typeof note1.date !== "undefined") {
153         list += "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;";
154         let date = new Date(note1.date);
155         let date1 = date.toLocaleDateString("hu-HU");
156         let time1 = date.toLocaleTimeString("hu-HU");
157         list += date1 + " " + time1;
158     }
159     list += "</li>";
160 }
161 }
162 return list
163 }
164
165
166 function algo_get_tx() {
167     if ($('#get_tx_status').val() !== working) {
168         (async () => {
169             $('#get_tx_status').val(working);
170             $('#game_list').empty();
171
172             let next_token = "";
173             let tx_limit = 50;
174             let num_tx = 1;
175             let list = "";
176
177             while (num_tx > 0) {
178                 let tx = await indexer_client
179                     .lookupAccountTransactions(addr_src)
180                     .limit(tx_limit)
181                     .nextToken(next_token).do();
182                 num_tx = tx.transactions.length;
183                 list += make_list_from_tx(tx);
184                 next_token = tx['next-token'];
185             }
186             $('#get_tx_status').val(ready);
187             $('#game_list').append(list);
188             $('#game_list').show();
189         })().catch(e => {
190             console.log(e);
191         });
192     }
193 }
194
195 $(window.document).ready(function () {
196     console.log("document ready");
197     const game_name = "Csocsó";
198     const user1 = "CsG";
199     const user2 = "LG";
200     $('#game').val(game_name);
201     $('#user1').val(user1);
202     $('#user2').val(user2);
203     $('#goal1').val('0');
204     $('#goal2').val('0');
205     $('#tx_id').val('');

```

```

206     $('#send_tx_status').val('');
207
208     $('#game_list').empty();
209     $('#get_tx_status').val('');
210
211     $('#store_game_result').on("click", function () {
212         console.log("Handler for Store game result is called.");
213         algo_send_tx_outer();
214     });
215
216     $('#read_game_results').on("click", function () {
217         console.log("Handler for Read game results is called.");
218         algo_get_tx();
219     });
220
221 });

```

Az Algod és Indexer szolgáltatás REST végpontjának megadása a 10. és 11. sorban történt. Az `algod_client` és `indexer_client` létrehozása a 31. és 33. sorban látható. A `peraWallet` létrehozása (37. sor) után a `$(document).ready` sorban (195. sor) folytatódik a végrehajtás. A HTML mezők kezdeti feltöltése után (197..209. sor) a gombokhoz tartozó eseménykezelők beállítása történik: az `id="store_game_result"` azonosítójú gombé a 211..214. sorban, az `id="read_game_results"` azonosítójú gombé a 216..219. sorban. A gombnyomáskor hívott függvények a `algo_send_tx_outer()` és a `algo_get_tx()`.

Lássuk ezek után a függvényeket. A Pera pénztárca kapcsolatot kezelő függvények (Pera wallet connect) a következők: a `reconnectSessionA` aszinkron függvény (40..46. sor) egy pénztárca kapcsolatot próbál újraépíteni. Ha sikeres a futása, akkor az `accounts` tömbben adja vissza a pénztárca műveletekben használható számlaszámokat. A `handleConnectWalletA` aszinkron függvény (48..52. sor) egy pénztárca kapcsolatot hoz létre. Ha sikeres a futása, akkor a `newAccounts` tömbben adja vissza a pénztárca műveletekben használható számlaszámokat. A `handleDisconnectWalletA` aszinkron függvény egy pénztárca kapcsolatot szüntet meg.

Az `algo_send_tx` függvény a `note` tartalmát küldi el a blokkláncra. A függvény megpróbál egy létező pénztárca kapcsolatot használni (64. sor). Ha ez a kapcsolat nem ajánlja fel azt a címet, amin a tranzakciót küldeni szeretnénk, akkor hibát dob (71..73. sor), egyébként pedig az `account_addr` változóba teszi a számlaszámot (76. sor). Ezután beolvassa az Algorand hálózatról a paramétereit (pl. kezdő blokk, genesis blokk, stb., lásd 80. sor), és létrehoz egy fizetési tranzakciót (82..89. sor). A tranzakciót a Pera pénztárcával aláírja (91..92. sor), majd az aláírt tranzakciót elküldi az Algorand hálózatnak (94. sor). Kíírja a tranzakció azonosítót a HTML formra (96. sor), és várakozni kezd mindaddig, de max. 5 körön keresztül, amíg be nem kerül a blokkláncba a tranzakció (98. sor). Ha a tranzakció beíródott a blokkláncba, akkor kiírja a formra, hogy a tx elküldése redben megtörtént (99. sor), egyébként pedig a kivétel kezelésével kiírja a hibát a formra (102. sor).

A `algo_send_tx_outer()` függvény beolvassa a formról a paramétereit (109..114. sor), összeállítja a `note` rekordot (115..122. sor), és meghívja az előzőleg ismertetett `algo_send_tx` függvényt.

A `algo_get_tx` függvény – az `indexer` segítségével – ciklusban visszaolvassa azokat a tranzakciókat a blokkláncról, melyekben a forrás cím az általunk megadott cím, lásd 177..185. sor. A kiolvasás részletekben történik, egyszerre `tx_limit` (50) tranzakciót olvas vissza. A visszaolvasott tranzak-

ciókat a `make_list_from_tx` függvénnyel alakítja át megjeleníthető formátumúra (hívása a 183. sorban, a függvény a 128..163. sorban látható.)

Szeretném arra a „furasaságra” felhívni a figyelmet, hogy a hálózattal való kommunikációt a `.do()` hatására végzi el a kliens JS API függvény, lásd pl. a 80., 94. és 181. sort.

5.11. Az első rész összefoglalása

Az első részben bemutattuk:

- az AlgoKit installálását
- az Algorand `goal` parancs használatát
- a Python API használatát
- a JS API használatát
- a Pera pénztárca kezelését a `walletconnect` interfész segítségével
- a Node.js alatti modulok összepakolását Web böngésző számára
- a `note` mező használatát nyilvántartási feladatokra

A következő részben az Algorand blokklánc programozhatóságáról fogunk szólni. Bemutatjuk majd:

- az AVM néhány utasítását
- a TEAL használatát parancsnyelvi környezetben
- a PyTeal használatát smart signature-ok vagy Algorand szerződések írására
- a LORA blokklánc vizsgáló használatát az Algorand szerződések tesztelésére
- a Beaker használatát Algorand szerződések írására, telepítésére, tesztelésére
- a TealScript használatát Algorand szerződések írására, telepítésére, tesztelésére
- a TEAL kód debug-olását
- a PuyaPy natív Python fordítóprogramot, amellyel Python-ban írt programokat lehet közvetlenül TEAL szerződéssé lefordítani

6. A TEAL

A TEAL az Algorand Virtuális Munkagép (AVM) „assembly nyelve”. A lefordított program a byte-kód, melyet az AVM közvetlenül értelmezni képes.

6.1. A TEAL program elemei

A TEAL program elemei:

- `#pragma version`, amely a használni kívánt AVM verzióját adja meg. Jelenleg 1 és 11 közötti verzió adható meg.
- címkék, amelyekre az ugró utasításokban lehet hivatkozni, pl. `l10:`, `subr_hello:`
- szimbolikus műveleti kódok, amelyek a nevükkel a végrehajtott műveletre emlékeztetnek, pl. `dup`, `callsub`, `return`
- a szimbolikus műveleti kódhoz tartozó paraméterek, pl. az ugró utasítások esetén egy címke, vagy paraméter betöltéskor a betöltendő paraméter neve, pl.
`asset_param_get AssetUnitName`
- megjegyzések, pl.
`// ez itt egy megjegyzés`

A TEAL az AVM egyes műveleteihez egy-egy szimbolikus kódot rendel. A műveletek paraméterei lehetnek magában a bytekódban tárolva, vagy az AVM architektúrájának megfelelő egyéb tárterületeken, pl. a stack-en, az átmeneti tároló területen stb.

Pl. a `dup` utasítás AVM kódja `0x49`. Ez az utasítás a stack tetején lévő értéket újra feldobja a stack-re.

6.2. Az AVM felépítése (architektúrája)

Az AVM következő végrehajtható utasítását a PC, a program számláló mutatja. Az egyes utasításokkal a következő adatok érhetők el:

- a stack, amely 1000 elemű, és egy eleme képes egy 64 bites előjel nélküli egész szám vagy egy max. 4096 hosszú `byte []` string tárolására.
- egy átmeneti tároló terület, amely 256 rekeszből áll, és mindegyik rekesz képes egy 64 bites előjel nélküli egész szám vagy egy max. 4096 hosszú `byte []` string tárolására.
- a tranzakcióban szereplő számlák vagy ASA-k mezői, pl. a küldő fél Algorand címe, a fogadó fél Algorand címe, a küldött összeg stb.
- globális paraméterek, pl. `MinTxFee` minimális tranzakciós díj, vagy a hálózat azonosítására szolgáló `GenesisHash` stb.
- az okos szerződést létrehozó számlához tartozó globális kulcs-érték párok, max. 64 db
- az okos szerződésbe benevező számlá(k)hoz tartozó lokális kulcs-érték párok, max. 16 db
- mailbox-ok, melyek max. 32K hosszúak lehetnek, de korlátlan számú lehet belőlük
- az okos szerződés vagy okos aláírás hívásakor megadott paraméterek

6.3. AVM adattípusok

A stack-en csak `uint64` és `[]byte` tárolható, ahol a `[]byte` vektor max. hossza 4096. Egyes AVM utasítások ennél korlátozottabb értéktartománnyal rendelkeznek. Pl. a `bigint` műveletekben értelem-szerűen csak a `bigint` adattípus használható.

Néhány ilyen AVM adattípus:

- Előjeletlen egész szám: `uint64` x , $0 \leq x \leq 2^{64} - 1$. A stack-en `uint64` adattípusként tárolódik.
- Byte vektor (string): `[]byte` x , ahol x hossza nem nagyobb 4096-nál. A stack-en `[]byte` adattípusként tárolódik.
- `bigint`: min. 1 byte, max. 64 byte hosszúságú nem negatív egész szám. A stacken `[]byte` adattípusként tárolódik.
- `address`, 32 byte-os `bigint`. A stack-en `[32]byte` adattípusként tárolódik.

Megjegyzés: Az Algorand címek 58 karakterből állnak. Ezek a karakterek 32-es alapú számrendszerben kódolják a számlához tartozó nyilvános kulcsot és egy 4 byte-os ellenőrző kódot. A TEAL az `addr` pszeudo utasítás után álló 58 karakteres Algorand címet 32 byte-os byte tömbbé alakítja át, amely a számla nyilvános kulcsának felel meg.

- `method`: 4 byte-os metódus kiválasztó érték, lásd ARC-4. A stack-en `[4]byte` adattípusként tárolódik.

Referenciák:

The Algorand Virtual Machine (AVM) and TEAL.

ARC-4: Application Binary Interface (ABI)

TEAL specification

go-algorand/data/transactions/logic/opcodes.go

6.3.1. Alulcsordulás és túlcsordulás kezelés

Ha egy `uint64` típusú számból kivonunk egy nála nagyobb `uint64` típusú számot, akkor az AVM bytekód végrehajtása hibajelzéssel azonnal megszakad.

Ha két `uint64` típusú szám közötti művelet végrehajtása során az eredmény $2^{64} - 1$ -nél nagyobb, akkor hasonló módon a bytekód végrehajtása hibajelzéssel azonnal megszakad.

1. TEAL példa: adattípusok

```
1 | i pi@l i pi -Virtual Box: ~/n_beta1$ nano ex1.teal
2 | #pragma version 10 // max. TEAL version on mainnet
3 | int 123456789 // uint64
4 | int 0x1234567812345678 // uint64
5 | byte "Hello" // [5]char
6 | byte "world"
7 | byte 0x123456789ABCDEF1234567 // bigint
8 | addr RGW4Q2Q2SKKTGZJ2XG3GTHHTTLX07I JAXKEMOCAFOI I6DTGSHNZTMGVXI // address
9 | == // compare 11 byte bigint with 32 byte address, expected: 0
10 | pop // pop result from stack
11 | == // compare "Hello" with "world", expected: 0
```

```

12 pop          // pop result from stack
13 ==          // compare 12345678 with 0x1234567812345678, expected: 0
14 CTRL/X
15 lipi@lipi -Virtual Box: ~/n_beta1$ ./goal clerk compile ex1.teal
16 ex1.teal: 2V4EUCTZJY7KB663Z5EEYJPDRVR3KAGPOJK7PPC4D7OXXGKOP304UFBYZ3Y
17 lipi@lipi -Virtual Box: ~/n_beta1$ hexdump -C ex1.teal.tok
18 00000000 0a 81 95 9a ef 3a 81 f8 ac d1 91 81 cf 95 9a 12 |.....|
19 00000010 80 05 48 65 6c 6c 6f 80 05 77 6f 72 6c 64 80 0b |..Hello..world..|
20 00000020 12 34 56 78 9a bc de f1 23 45 67 80 20 89 ad c8 |.4Vx...#Eg. ...|
21 00000030 6a 1a 92 95 33 65 3a b9 b6 69 9e 73 9c d7 77 7d |j...3e:..i..s..w}|
22 00000040 09 05 d4 46 38 40 2b 90 8f 0e 66 91 db 12 48 12 |...F8@+...f...H.|
23 00000050 48 12 |H.|
24 00000052
25 lipi@lipi -Virtual Box: ~/n_beta1$ ./goal clerk compile -D ex1.teal.tok
26 #pragma version 10
27 pushint 123456789
28 pushint 1311768465173141112
29 pushbytes 0x48656c6c6f // "Hello"
30 pushbytes 0x776f726c64 // "world"
31 pushbytes 0x123456789abcdef1234567 // 0x123456789abcdef1234567
32 pushbytes 0x89adc86a1a929533653ab9b6699e739cd777d0905d44638402b908f0e6691db // addr
   ↪  RGW4Q202SKKTGZJ2XG3GTHTTTTLX07I JAXKEMOCAFOI I6DTGSHNZTMGVXI
33 ==
34 pop
35 ==
36 pop
37 ==
38 lipi@lipi -Virtual Box: ~/n_beta1$

```

Az 2. sorban lévő `#pragma version` a használni kívánt AVM változatot adja meg. Értéke jelenleg 1 és 11 között lehet. A 3. sorban egy `uint64` típusú egész szerepel decimális alakban megadva. A 4. sor azt mutatja, hogy a `0x` prefix használatával az `uint64` hexadecimális alakban is megadható. Az 5. és 6. sorban egy-egy byte vektor (string) lett megadva. A 7. sorban egy 11 byte-os bigint szerepel, végül a 8. sorban egy Algorand cím lett megadva az `addr` pseudo-utasítás segítségével.

A TEAL program a `goal clerk compile` segítségével fordítható le bytekódra, lásd 15. sor. A lefordított program a `.tok` kiterjesztést kapja. A lefordított program hexadecimális dump-ja a 18..24. sorban látható.

Végül a 25. sorban a `-D` kapcsolóval disassembláltuk a bytekódot. Látható, hogy a TEAL fordító a 10-es AVM változatban a `pushint` és `pushbytes` AVM kódot használja a konstansok stack-en történő tárolásához. A 32. sorban látható, hogy az Algorand címből valóban 32 byte-os byte tömb keletkezett.

A bytekód végrehajtásakor a 33. sorban lévő `==` leemeli a stack tetején lévő két értéket, jelen esetben egy címet (32. sor) és egy bigint-et (31. sor), és összehasonlítja őket. Mivel nem egyenlők, egy 0 értéket tesz a stackre. Ezt az értéket a `pop` (34. sor) „eldobja”. A 35. sorban lévő `==` leemeli a stack tetején lévő két értéket, jelen esetben egy byte tömböt (30. sor) és egy másik byte tömböt (29. sor), és összehasonlítja őket. Mivel nem egyenlők, egy 0 értéket tesz a stackre. Ezt a `pop` (36. sor) „eldobja”. Végül a 37. sorban álló `==` leemeli a stack tetején lévő két értéket, jelen esetben egy `uint64` típusú értéket (28. sor) és egy másik `uint64` típusú értéket (27. sor). Mivel nem egyenlők, egy 0 értéket tesz a stackre.

@todo Jó lenne, ha egy szimulátorban futtatva láthatnánk a bytekód működését. Ahhoz, hogy egy bytekódot meghívhassuk, be kell ágyazni a kódot egy okos aláírásba vagy egy okos szerződésbe. A következő rész ennek a módját ismerteti.

6.4. Algorand Smart Signatures

@todo

6.5. Példa az ASS használatára

@todo

6.6. Algorand alkalmazások

6.6.1. A bytekód maximális mérete

A bytekód okos aláírások (smart signatures) esetén max. 1000 byte hosszú lehet. A paraméterek megadása tovább csökkenti ezt a méretet. Okos szerződések (smart contracts) esetén az engedélyező (approval) és a feltétlen kiszállást végző (clear) programok együttes mérete alapéretelmezésben 2 Kbyte lehet. A méret 2 Kbytes-os lépésekben növelhető, és mind az engedélyező, mind a feltétlen kiszállást végző program max. 8 Kbyte méretű lehet.

6.6.2. A bytekód maximális végrehajtási költsége

A bytekód végrehajtása során az egyes műveletek költségének (cost) összege okos aláírások esetén max. 20 000 egység lehet. A legtöbb bytekód művelet költsége 1. Vannak olyan műveletek, melyeknek lényegesen nagyobb a költsége. Pl. az `sha256` művelet költsége 35 egység, az `ed25519verify` költsége 1900 egység. Egy adott művelet költsége az egyes műveletek leírásánál található meg: v10 műveleti kódok

Okos szerződések esetén az engedélyező és a feltétlen kiszállást végző programban lévő műveletek költségének összege tranzakciónként max. 700 egység lehet. Ezt a maximum értéket tranzakciós csoportok és belső tranzakciók használatával jelentősen megnövelhetjük. Pl. ha a tranzakciós csoportban 10 tranzakció van, akkor a maximum érték $10 \cdot 700 = 7000$ egységre módosul. Hasonló módon, minden egyes belső tranzakció 700-zal növeli a maximum értékét. Mivel a tranzakciós csoportban max. 16 tranzakció lehet, a belső tranzakciók száma pedig max. 256 lehet, a műveletek költsége maximum $700 \cdot (16 + 256) = 190\,400$ lehet. Ilyenkor a tranzakció végrehajtási költsége (fee) is megnő, flat fee esetén max. $(16 + 256) \cdot 0,001$ Algo = 0,272 Algo lesz.³

Hivatkozás: Az opkódok végrehajtási költsége

Hivatkozás: Minimális tx díjként ne adj meg fixen 1000 microAlgó-t

6.7. Példa az Algorand alkalmazások használatára: “Hello, World”

Az Algorand “HelloWorld” alkalmazása a legegyszerűbben az `algokit init` parancs segítségével hozható létre. A szükséges lépések:

³ A tranzakciós díj a jövőben változhat. Érdemes az SDK konstansokat használni, pl. a JS SDK-ban ezt: `const minFee = algodk.ALGORAND_MIN_TX_FEE`

- Lépj be a <https://github.com> alatti számládra
- Hozz létre egy új repository-t, pl. `asc`
- Hozz létre a repository-ban egy README.MD fájlt
- Nyomd meg a “Code” gombot. Menj a “Codespaces” fülre. Indítsd el a “Codespaces”-t.
- A megjelenő VS Code böngésző ablakban installáld az algokit-et: `pipx install algokit`
- Indítsd el a localnet-et: `algokit localnet start`
- A `docker ps` paranccsal vizsgáld meg, hogy elindult és fut a négy darab container
- A `ports` fülön tedd publikussá a 4001, 4002 és 8980 portokat
- Futtasd az `algokit init` parancsot a következőképpen:
 - válaszd ki: **Smart Contracts & DApp Frontend**
 - válaszd ki: **TypeScript**
 - könyvtárnévnek add meg: **ex**
 - válaszd ki a következő mintát: **Starter**
 - Hagyd meg az alapértéket a smart contract névnek: **HelloWorld**
 - Arra a kérdésre, hogy “Futtassa az ‘algokit project bootstrap’parancsot?” **Yes**

```

1 @A-Maugli → /workspaces/asc (main) $ algokit init
2 ? Which of these options best describes the project you want to build? Smart Contracts & DApp
   ↳ Frontend
3 ? Which language would you like to use for the smart contract? TypeScript
4 ? Name of project / directory to create the project in: ex
5 Starting template copy and render at /workspaces/asc/ex...
6   Name of the template preset to use.
7   Starter - for a simpler starting point ideal for prototyping
8   Name of the default smart contract app.
9   HelloWorld
10 ==== Checking compatibility with the cli ====
11 ==== 1/4 - Initializing base template ====
12 Starting template copy and render at /workspaces/asc/ex...
13 Template render complete!
14   Project initialized at ex ! For template specific next steps, consult the documentation of
   ↳ your selected template
15 Your selected template comes from:
16 → https://github.com/algorithmicfoundations/algokit-base-template
17 Your template includes a README.md file, you might want to review that as a next step.
18 ==== 2/4 - Initializing frontend template ====
19 Starting template copy and render at /workspaces/asc/ex/projects/ex-frontend...
20 Template render complete!
21   Project initialized at ex-frontend ! For template specific next steps, consult the
   ↳ documentation of your selected template
22 Your selected template comes from:
23 → https://github.com/algorithmicfoundations/algokit-react-frontend-template
24 Your template includes a README.md file, you might want to review that as a next step.
25 ==== 3/4 - Initializing backend template ====
26 Starting template copy and render at /workspaces/asc/ex/projects/ex-contracts...
27 Template render complete!

```

```

28   Project initialized at ex-contracts ! For template specific next steps, consult the
    ↳ documentation of your selected template
29 Your selected template comes from:
30 → https://github.com/algorand-devrel/teal-script-algokit-template
31 Your template includes a README.md file, you might want to review that as a next step.
32 ==== 4/4 - Finalizing setup ====
33 Template render complete!
34 ? Do you want to run algokit project bootstrap for this new project? This will install and
    ↳ configure dependencies allowing it to be run immediately. Yes
35 Installing npm dependencies
36 npm:
37 npm: added 568 packages, and audited 569 packages in 49s
38 npm:
39 npm: 137 packages are looking for funding
40 npm: run npm fund for details
41 npm:
42 npm: found 0 vulnerabilities
43 npm: npm notice
44 npm: npm notice New minor version of npm available! 10.5.0 -> 10.7.0
45 npm: npm notice Changelog: <https://github.com/npm/cli/releases/tag/v10.7.0>
46 npm: npm notice Run npm install -g npm@10.7.0 to update!
47 npm: npm notice
48 Copying /workspaces/asc/ex/projects/ex-frontend/.env.template to
    ↳ /workspaces/asc/ex/projects/ex-frontend/.env and prompting for empty values
49 Installing npm dependencies
50 npm: npm WARN deprecated @walletconnect/types@1.8.0: WalletConnect's v1 SDKs are now deprecated.
    ↳ Please upgrade to a v2 SDK. For details see: https://docs.walletconnect.com/
51 npm: npm WARN deprecated @walletconnect/client@1.8.0: WalletConnect's v1 SDKs are now deprecated.
    ↳ Please upgrade to a v2 SDK. For details see: https://docs.walletconnect.com/
52 npm: npm WARN deprecated @motionone/vue@10.16.4: Motion One for Vue is deprecated. Use Oku Motion
    ↳ instead https://oku-ui.com/motion
53 npm:
54 npm: added 367 packages, and audited 369 packages in 1m
55 npm:
56 npm: 41 packages are looking for funding
57 npm: run npm fund for details
58 npm:
59 npm: found 0 vulnerabilities
60   Project initialized at ex ! For template specific next steps, consult the documentation of
    ↳ your selected template
61 Your selected template comes from:
62 → https://github.com/algorandfoundation/algokit-fullstack-template
63 Directory is already under git revision control, skipping git setup
64 VSCode configuration detected in project directory, and 'code' command is available on path,
    ↳ attempting to launch VSCode
65 @A-Maugli → /workspaces/asc (main) $

```

Az `algokit init` parancs a következő könyvtárszerkezetet hozza létre:

```

1  asc
2    + ex
3      + projects
4        + ex-contracts
5          + __test__
6            HelloWorld.test.ts
7          + contracts

```

```

8         + clients
9         + artifacts
10        HelloWorld.algo.ts
11        package.json
12    + ex-frontend
13        + src
14            + assets
15            + components
16            + contracts
17            + interfaces
18            + styles
19            + utils
20                App.tsx
21                Home.tsx
22                main.tsx
23        .env
24        .env.template
25        package.json

```

6.7.1. A backend

Az okos szerződés

Az `algokit init` parancs létrehozza egy Typescript-ben megírt Algorand mintaszerződés kódját az `ex-contracts/contracts/HelloWorld.algo.ts` fájlban:

```

1  import { Contract } from '@algorandfoundation/tealscript';
2
3  export class HelloWorld extends Contract {
4      /**
5       * Calculates the sum of two numbers
6       *
7       * @param a
8       * @param b
9       * @returns The sum of a and b
10     */
11     private getSum(a: uint64, b: uint64): uint64 {
12         return a + b;
13     }
14
15     /**
16     * Calculates the difference between two numbers
17     *
18     * @param a
19     * @param b
20     * @returns The difference between a and b.
21     */
22     private getDifference(a: uint64, b: uint64): uint64 {
23         return a >= b ? a - b : b - a;
24     }
25
26     /**
27     * A method that takes two numbers and does either addition or subtraction
28     *
29     * @param a The first uint64
30     * @param b The second uint64

```



```

31  * @param operation The operation to perform. Can be either 'sum' or 'difference'
32  *
33  * @returns The result of the operation
34  */
35  doMath(a: uint64, b: uint64, operation: string): uint64 {
36      let result: uint64;
37
38      if (operation === 'sum') {
39          result = this.getSum(a, b);
40      } else if (operation === 'difference') {
41          result = this.getDifference(a, b);
42      } else throw Error('Invalid operation');
43
44      return result;
45  }
46
47  /**
48   * A demonstration method used in the AlgoKit fullstack template.
49   * Greets the user by name.
50   *
51   * @param name The name of the user to greet.
52   * @returns A greeting message to the user.
53   */
54  hello(name: string): string {
55      return 'Hello, ' + name;
56  }
57  }

```

Az szerződés két meghívható metódusa a `doMath` és a `hello`. A szerződés a következőképpen fordítható le TEAL kódra:

```

cd ex/projects/ex-contracts
npm run build

```

A fordítás során keletkező fájlok

Az eredmény fájlok a `contracts/artifacts` könyvtárban képződnek:

- `HelloWorld.approval.teal`, az okos szerződést létrehozó TEAL kód + az okos szerződés metódusai
- `HelloWorld.arc4.json`, az alkalmazási hívási felület leíró JSON fájl (ABI fájl, Application Binary Interface)
- `HelloWorld.arc32.json`, az alkalmazást leíró JSON fájl, tartalmazza az ARC4 ABI leírást is
- `HelloWorld.arc56_draft.json`, az alkalmazást még részletesebben leíró JSON file. Tartalmazza pl. a source map-et JSON formátumban. Az ARC56_draft még nincs elfogadva, Joe Polny github repójában található meg.
- `HelloWorld.clear.teal`, az opt-out feltétel nélküli végrehatásához tartozó TEAL kód
- `HelloWorld.src_map.json`, a Typscript források, a TEAL források és az AVM PC-k összerendelését végző map fájl

Az Algorand okosszerződést leíró TypeScript file a `contracts/clients` könyvtárban képződik:

- `HelloWorldClient.ts`, az alkalmazás hívását segítő TypeScript wrapper

A `HelloWorldClient.ts` TypeScript Wrapper feladata

A TypeScript Wrapper magas szintű interfészt biztosít egy Algorand okosszerződéshez. Az automatikusan generált kód elősegíti, hogy az okosszerződés műveleteit könnyen használhassuk a TypeScript alkalmazásokban.

A TypeScript Wrapper automatikusan elkészíti az okosszerződés hívásához szükséges tranzakciókat és segítséget nyújt a komplex adatstruktúrák felhasználó-barát kezelésében:

- absztrakciót nyújt: Nem szükséges mélyrehatóan ismerni a TEAL-t, a kliens magasabb szintű interfészt biztosít.
- automatizálja a tranzakciókat: automatikusan kezeli a tranzakciók összeállítását, az aláírásokat és a hálózati kommunikációt.
- típusbiztonságot biztosít: a generált kód TypeScript típusokat használ, ami segíti a hibák elkerülését a fejlesztés során.

A TypeScript Wrapper részei:

- Okosszerződés specifikáció: Az `APP_SPEC` objektum tartalmazza az okosszerződés specifikációját, beleértve a szerződésben elérhető különböző metódusokat (`doMath`, `hello`, `createApplication`) és azok paramétereit, viselkedését. Ez az információ alapvető a kliens működéséhez, mivel meghatározza, hogy milyen műveletek végezhetők az okosszerződésen.
- `HelloWorldClient` osztály: a szerződés metódusainak hívásait magasabb szinten kezelő osztály. Az okosszerződés különböző funkcióit egyszerű TypeScript függvényhívásként érhetjük el, mint például a `doMath()` vagy a `hello()` függvények.
- `HelloWorldCallFactory` osztály: ez az osztály előre definiált függvényeket tartalmaz, amelyek tranzakciókat hoznak létre az okosszerződés meghívásához. Például a `doMath()` metódus egy tranzakciót készít, amely két szám összeadását vagy kivonását végzi el a szerződés segítségével.
- `OnCompletion` típusok és állapot kezelés: az alkalmazás hívásához szükséges típusok definiálása, valamint a szerződés (globális) állapotában tárolt adatok kezelésére szolgáló `IntegerState` és `BinaryState` típusok.
- `deploy`: A `deploy()` metódus segítségével az okosszerződést idempotens módon telepíthetjük. Az idempotens viselkedés azt jelenti, hogy ugyanazt az eredményt kapjuk, ha a telepítést többször hajtjuk végre.

A TypeScript Wrapper egyszerűsíti az Algorand okosszerződések meghívását. Elrejti az alacsony szintű működést (tranzakciók, tranzakció csoportok, aláírások kezelése), és lehetővé teszi decentralizált alkalmazások egyszerű és gyors fejlesztését.

A szerződés tesztelése

A szerződés tesztelése a Jest keretrendszerre támaszkodik. A `jest` és a `ts-jest` npm modulok installálása után a TypeScript-ben megírt tesztek közvetlenül a Jest keretrendszerben végrehajthatók. A

Jest részletesebb leírása a Nathan Rozentals *Mastering TypeScript* című könyvének 10. fejezetében található: *Test-Driven Development*.

A szerződéshöz tartozó Jest test az `ex-contracts/__test__/HelloWorld.test.ts` fájlban található:

```
1 import { describe, test, expect, beforeEach, beforeAll } from '@jest/global';
2 import { algorandFixture } from '@algorandfoundation/algorand-utils/testing';
3 import * as algokit from '@algorandfoundation/algorand-utils';
4 import { HelloWorldClient } from '../contracts/clients/HelloWorldClient';
5
6 const fixture = algorandFixture();
7 algokit.Config.configure({ populateAppCallResources: true });
8
9 let appClient: HelloWorldClient;
10
11 describe('HelloWorld', () => {
12   beforeEach(fixture.beforeEach);
13
14   beforeAll(async () => {
15     await fixture.beforeEach();
16     const { testAccount } = fixture.context;
17     const { algorand } = fixture;
18
19     appClient = new HelloWorldClient(
20       {
21         sender: testAccount,
22         resolveBy: 'id',
23         id: 0,
24       },
25       algorand.client.algod
26     );
27
28     await appClient.create.createApplication({});
29   });
30
31   test('sum', async () => {
32     const a = 13;
33     const b = 37;
34     const sum = await appClient.doMath({ a, b, operation: 'sum' });
35     expect(sum?.return?.valueOf()).toBe(BigInt(a + b));
36   });
37
38   test('difference', async () => {
39     const a = 13;
40     const b = 37;
41     const diff = await appClient.doMath({ a, b, operation: 'difference' });
42     expect(diff?.return?.valueOf()).toBe(BigInt(a >= b ? a - b : b - a));
43   });
44
45   test('hello', async () => {
46     const diff = await appClient.hello({ name: 'world!' });
47     expect(diff?.return?.valueOf()).toBe('Hello, world!');
48   });
49 });
```

Magyarázata:

- az 1. sorban beimportált Jest globálisok feladata:
 - describe, a Jest teszt csoportok leírására szolgál
 - test, az egyes Jest tesztek leírására szolgál
 - expect, a tesztek eredményének vizsgálatára szolgál
 - beforeAll, az egyes teszt csoportok előtt futtatott teszt setup
 - beforeEach, az egyes tesztek előtt futtatott teszt setup
- a 2. sorban importált algorandFixture állítja elő a 6. sorban a legfontosabb Algorand paramétereket, pl. a 17. sorban található algorandClient típusú algorand változót, amely az algorand.client.algod property-ben tartalmazza az algod (algorand démon) hivatkozását.
- a 11. sorban történik a teszt csoport megadása
- a 12. sor írja elő, hogy minden egyes teszt előtt le kell futtatni a fixture.beforeEach függvényt
- a 14. sor írja elő a teszt csoportra vonatkozó, egyszer futtatandó teszt beállításokat:
 - 15. sor: vár az Algorand fixture-re
 - 16. sor: kiveszi a testAccount-ot a fixture-ből
 - 17. sor: kiveszi az algorand: algorandClient property-t a fixture-ből
 - 19..26. sor: létrehoz egy új appClient-et
 - 28. sor: létrehoz egy új app-ot, azaz okosszerződést a blokkláncon. Megjegyzés: Az app, application és smart contract (app, alkalmazás, okosszerződés) egymás szinonimái.
- 31..36. sor: egy teszt definiálása
 - 34. sor: a blokkláncon élő szerződés doMath metódusának meghívása, az appClient-en keresztül. Az appClient intézi az app-ot hívó tranzakció összeállítását, a tranzakció aláírását, elküldését, és az eredmény struktúrált formában történő előállítását.
 - 35. sor: annak a vizsgálata, hogy az eredmény egyezik-e $a + b$ -vel.
- 38..43. sor: egy újabb teszt definiálása, amely az app doMath() metódusát hívja, majd ellenőrzi, hogy az app metódusa a megfelelő eredmény szolgáltatata-e
- 45..48. sor: egy újabb teszt definiálása, amely az app hello() metódusát hívja, majd ellenőrzi az app-pal végeztetett string összefűzés eredményét

`npm run test`

paranccsal futtathatók. A futtatás eredménye:

```
1 PASS   __test__/HelloWorld.test.ts (45.45 s)
2   HelloWorld
3     ✓ sum (9982 ms)
4     ✓ difference (10250 ms)
5     ✓ hello (10407 ms)
6
7 Test Suites: 1 passed, 1 total
```

```
8 Tests:      3 passed, 3 total
9 Snapshots:  0 total
10 Time:       45.863 s
11 Ran all test suites.
```

Ha a Jest TypeScript kód debug-olására van szükség, akkor a VS Code "Run and Debug" gombjának megnyomása után indítsunk el egy debug terminált. A debug terminálban a `npm run test` paranccsal indítsuk el a futást. A megadott töréspontokon megáll a futás, és lépésenként folytathatjuk a végrehajtást.

6.7.2. A frontend

Az `ex-frontend` egy React frontend-et tartalmaz. Menjünk a `ex-frontend` könyvtárba, és futtassuk a kliens generálást és a packer programot:

```
cd ex/projects/ex-frontend
npm run dev
```

A `package.json` fájlból látszik, hogy mi megy ekkor végbe:

```
1 cat package.json
2 ...
3 "scripts": {
4   "generate:app-clients": "algokit project link --all",
5   "dev": "npm run generate:app-clients && vite",
6   "build": "npm run generate:app-clients && tsc && vite build",
7   "preview": "vite preview"
8 },
9 ...
```

Az `algokit project link --all` parancs hatására „legenerálódik” az összes szerződéshez tartozó typed client, a `ex-frontend/src/contract` könyvtárba.

A `vite` parancs elindítja a vite packer-t dev módban (gyors, szinte azonnali fordítás), és egy hyperlink-en keresztül elérhetővé teszi e React alkalmazást.

Ha a `codespaces`-t használjuk, akkor módosítani kell a `.env` fájlt is, a következőképpen: a Ports fülről ki kell másolni a 4001 portnál szereplő URL-t, és be kell írni a `localhost` helyébe. A 4001 portot pedig 443-ra kell átírni:

```
régi:
VITE_ALGOD_SERVER=http://localhost
VITE_ALGOD_PORT=4001

új:
VITE_ALGOD_SERVER=https://(a ports fülről átmásolt érték)
VITE_ALGOD_PORT=443
```

Hasonló módon, a `VITE_KMD_SERVER`, `VITE_KMD_PORT` és a `VITE_INDEXER_SERVER`, `VITE_INDEXER_PORT` soroknál is módosítani kell a `.env` fájlt.

Ezután a következő parancsokat kell kiadnunk az `ex-frontend` könyvtárban:

```

1 # vite.env fájl létrehozás, node_modules felöltés érdekében
2 algokit project bootstrap all
3 # lokális Algorand hálózati csomópont indítása
4 algokit localnet start
5 # A 4 db container vizsgálata
6 docker ps
7 # többször érdemes kiadni a docker ps parancsot, mert jelenleg a conduit hajlamos az elszállásra.
  ↳ Ilyenkor próbálkozni lehet a
8 algokit localnet reset
9 # paranccsal.
10
11 # A 4001, 4002, és 8980 portok public-ká tétele a ports fülön
12
13 # Végül
14 npm run dev
15 # és CTRL + kattintás segítségével elindítani a böngészőben a React alkalmazást.
16   ↳ Local: http://localhost:5173/
17 # A böngészőben F12-vel fejlesztői környezet indítása.

```

6.7.3. Az ABI

Az ABI (Application Binary Interface) azt írja le, hogy milyen adattípusok használhatók egy okos szerződés valamelyik metódusának meghívásakor. A használható adattípusok definiálása lehetővé teszi, hogy az okos szerződéseket erősen típusos (strongly typed) paraméterekkel lehessen meghívni.

Eredetileg az Ethereum-ban definiálták először az ABI-t.

Az Algorandban az arc-0004 leírás definiálja az ABI-t, azaz az Algorand fejlesztői környezetben használható adattípusokat. A read-only paraméterek definícióját az arc-0022, a napló eseményekét az arc-0028 tartalmazza.

Az ABI réteg és az TEAL (AVM) közötti típuskonverziót a különféle fejlesztő rendszerek – TypeScript, PuyaPy – automatikusan elvégzik. Ennek megfelelően az ABI-t használó okos szerződések hosszabbak, bonyolultabbak a szükséges adatkonverziók miatt. Például a `string` ABI adattípus esetén az első két byte-on a string hossza tárolódik, ezt követi maga a string. Az Algorand `[]byte` adattípusa viszont implicit módon tartalmazza a string hosszúságát, a stack-re feltett adatbyte-ok száma révén.

6.7.4. “HelloWorld” mintaprogram ABI használata nélkül

Egy “HelloWorld” mintaprogram ABI használata nélkül, PuyaPy Pythonban a következő:

```

contract.py
1 from algopy import Contract, Txn, Log
2
3 class HelloWorldContract(Contract):
4     def approval_program(self) -> bool:
5         name = Txn.application_args(0)
6         log(b"Hello, " + name)
7         return True
8
9     def clear_state_program(self) -> bool:

```

10

return True

A Python kód magyarázata: A 3. sorban a Contract osztályból származtatjuk a HelloWorldContract szerződést. Az `approval_program` metódus a tranzakció végrehajtását engedélyezi vagy tiltja, 1 vagy 0 visszatérési érték esetén. A `clear_state_program` metódus a szerződésbe való benevezés esetén feltétlen kiszállást tesz lehetővé. Az 5. sor beolvassa az alkalmazás meghívására használt tranzakció első paraméterét (a paraméterek számozása a TEAL-ben 0-tól kezdődik), a 6. sor pedig naplózza a "Hello, "+name eredményét.

A PuyaPy compiler hívásakor a két függvény külön TEAL állományba kerül. A fenti Python kód lefordításával előálló engedélyező (approval) TEAL program a következő lesz:

HelloWorldContract.approval.teal

```

1 #pragma version 10
2
3 examples.hello_world.contract.HelloWorldContract.approval_program:
4     byte "Hello, "
5     txna ApplicationArgs 0
6     concat
7     log
8     int 1
9     return

```

Magyarázat: A TEAL program a 4. sorban a "Hello, " stringet teszi fel az AVM stack-re, az 5. sorban pedig az alkalmazás első paraméterét. Figyelem, az paraméterek indexelése 0-tól kezdődik. A 6. sorban összefűzi a két stringet, a 7. sorban pedig az eredményül kapott stringet leveszi a stack-ről, és naplóbejegyzésként rögzíti a blokkláncon. Végül a 8. sorban az 1 értéket teszi fel a stack-re. Ez lesz az engedélyező program visszatérési értéke, az 1 érték azt jelenti, hogy engedélyezzük a tranzakció végrehajtását.

6.7.5. "HelloWorld" mintaprogram ABI használatával

Egy "HelloWorld" mintaprogram ABI használatával, PuyaPy Pythonban megírva a következő:

contract.py

```

1 from algopy import ARC4Contract, String, arc4
2
3 class HelloWorldContract(ARC4Contract):
4     @arc4.abi_method
5     def hello(self, name: String) -> String:
6         return "Hello, " + name

```

Az engedélyező (approval) TEAL program a következő:

HelloWorldContract.approval.teal

```

1 #pragma version 10
2
3 examples.hello_world_arc4.contract.HelloWorldContract.approval_program:
4     txn NumAppArgs
5     bz main_bare_routing@5

```

```

6     method "hello(string)string"
7     txna ApplicationArgs 0
8     match main_hello_route@2
9     err // reject transaction
10
11    main_hello_route@2:
12        txn OnCompletion
13        !
14        assert // OnCompletion is NoOp
15        txn ApplicationID
16        assert // is not creating
17        txna ApplicationArgs 1
18        extract 2 0
19        callsub hello
20        dup
21        len
22        itob
23        extract 6 2
24        swap
25        concat
26        byte 0x151f7c75
27        swap
28        concat
29        log
30        int 1
31        return
32
33    main_bare_routing@5:
34        txn OnCompletion
35        !
36        assert // reject transaction
37        txn ApplicationID
38        !
39        assert // is creating
40        int 1
41        return
42
43
44    // examples.hello_world_arc4.contract.HelloWorldContract.hello(name: bytes) -> bytes:
45    hello:
46        proto 1 1
47        byte "Hello, "
48        frame_dig -1
49        concat
50        retsub

```

A program magyarázata: A TEAL program 4. sora a paraméterek számát teszi fel az AVM stack-re. Az 5. sor a `main_bare_routing@5` címűre ugrik, ha nem adtak meg az alkalmazás hívásakor egyetlen paramétert sem, lásd 33. sor. A tranzakció `OnCompletion` mezője a következő értékeket veheti fel:

- `NoOp (0)`
 - ha az az `AppId` nulla, akkor most jött létre az alkalmazás.
 - ha az `AppId` nem nulla, akkor az alkalmazás hívása.

- OptIn (1)
 - ha az az AppId nulla, akkor érvénytelen
 - ha az AppId nem nulla, akkor benevezés (opt-in) az alkalmazásba.
- CloseOut (2)
 - ha az az AppId nulla, akkor érvénytelen
 - ha az AppId nem nulla, akkor kiszállás (opt-out) az alkalmazásból.
- ClearState (3) - nem fordulhat elő az approval programban
- UpdateApplication (4)
 - ha az az AppId nulla, akkor érvénytelen
 - ha az AppId nem nulla, akkor az alkalmazás módosítása
- DeleteApplication (5)
 - ha az az AppId nulla, akkor érvénytelen
 - ha az AppId nem nulla, akkor az alkalmazás törlése

A 34. sor az OnCompletion értékét teszi a stack-re. A 35. sor egy negálás, vagyis 0 értéket tesz a stack-re, ha a stack-en lévő érték nem nulla, és 1 értéket tesz a stack-re, ha a stack-en lévő érték nulla. A 36. sorban lévő `assert` azonnal megszakítja a program futását, ha a stack tetején lévő érték nulla. Vagyis: a program végrehajtása megszakad, ha az OnCompletion értéke nem NoOp-pal egyenlő. Hasonló módon, ha az ApplicationID értéke nem nulla, akkor a program futása azonnal megszakad (39. sor). Vagyis a két `assert` biztosítja, hogy most jött létre az alkalmazás, és NoOp a kód értéke. A 40. sorban lévő 1 érték stack-re helyezésevel engedélyezzük az alkalmazás létrehozását. Minden más esetben – vagyis ha a completion kód nem NoOp vagy az alkalmazás nem éppen most jön létre – hibával megszakítjuk a hívást.

A 6. sorban a `method "hello(string)string"` a hello metódus 4 byte-os HASH lenyomatát hozza létre és helyezi a stack-re, az ABI szabványnak megfelelően. A 7. sor az alkalmazás 1. paraméterét teszi a stack-re. Ha a két érték egyezik, akkor a `main_hello_route@2` címkénél folytatódik a végrehajtás, egyébként pedig a végrehajtás hibajelzéssel azonnal megszakad.

A 12..16. sor ellenőrzi, hogy az Oncompletion == NoOp és az ApplicationID != 0 feltételek teljesülnek-e. Ha nem, akkor a végrehajtás hibával megszakad.

A 17. sorban feltesszük az AVM stackre a második hívási paramétert, ami a `name` ABI string. A 18. sorban levágjuk róla a string hosszát kódoló 2 byte-ot, ezzel megkapjuk az ABI string típusból az AVM elemi stringet. A 19. sorban meghívjuk a `hello` szubrutint, amelyik a "Hello, " stringet összefűzi a call frame-en lévő AVM stringgel.

Innentől az eredmény ABI szabványnak megfelelő visszakódolása történik, vagyis az AVM eredménystringből ABI eredménystringet állít elő az okos szerződés, a következőképpen:

A 20. sor megduplázza az AVM eredmény stringet. A 21. sor megállapítja az eredmény string hosszát. A 22. sorban lévő `itob` parancs az uint64 típusú 8 byte-os egész számból, ami a string hossza, egy 8 byte-os stringet készít. A 23. sor ennek az utolsó 2 byte-ját szedi ki. A 24. sorban lévő

`swap` megcseréli a stack 2 főső elemét, vagyis a stack tetején az AVM eredmény string lesz, eggyel beljebb a string hossza, 2 byte-on ábrázolva. Pl. ha az eredmény hossza 17, akkor ez a 2 byte a 0x31 0x37 lesz. A 25. sor összefűzi a 2 byte-os hosszt és az eredmény stringet.

Ezután az eredmény előállítás történik, az arc-0028-nak megfelelően. A 4 byte-os prefix, amit az eredmény elé kell tenni, a következőképpen állítható elő:

```
1 const { sha512_256 } = require("js-sha512");
2 sig = "return";
3 //sig = "Swapped(uint64, uint64)";
4 hash = sha512_256(sig);
5 prefix = hash.slice(0, 8);
6 console.log("Prefix: ", prefix);
```

Egészen ellentétes az intuícióval, hogy az event log-hoz a 4 byte-os prefix-et nem a "hello(string)" hash-elésével, hanem ehelyett egyszerűen a "return" hash-elésével kell előállítani. A 26. sor az eredmény hash-ének első 4 byte-ját tartalmazza, 0x151f7c75. A 27. és 28. sor ezt a hash értéket előlről hozzáfűzi az eddigi stringhez, ami eddig 2 byte-on a hosszt tartalmazza, majd magát az AVM eredménystringet. A 29. sor ezt az eredményt naplózza a blokkláncban. Végül a 30. sor az 1 érték stack-re helyezésével engedélyezi a tranzakciót, és a 31. sorban véget ér ennek az ágnak a végrehajtása.

Látható, hogy mennyivel bonyolultabb lett a "HelloWorld" szerződés az ABI használatával, mint anélkül.

Hivatkozások:

[Contract ABI Specification - Solidity 0.8.26 documentation](#)

[ABI details - Algorand Developer Portal](#)

[arc-0004, Algorand Transaction Calling Conventions](#)

[arc-0022, Add 'read-only' annotation to methods](#)

[arc-0028, Algorand Event Log Spec](#)

[arc-0032, Application Specification](#)

[TEAL V10 opcodes](#)

Könyvek:

[Nathan Rozentals *Mastering TypeScript*, 4. kiadás](#)

7. Esettanulmány: tulajdonrész opciós vételi jog

7.1. A feladat leírása

A „Bizalmi Kör” nevű társaság tíz vevő számára értékesíteni szeretné a tulajdonrészének $10 \cdot 0,1\%$ -át. Ehhez a társaság egy blokkláncon 10 db zsetont bocsát ki, és egy okos szerződés segítségével értékesíti a zsetonokat. A zsetonok a kibocsátást követő 30 nap során vásárolhatók meg. A zsetonok mindegyike egy-egy opciós vételi jogot testesít meg. Egy vevő csak egy zsetont vásárolhat meg.

Ahhoz, hogy a zseton tulajdonosa élhessen az opciós vételi jogával, meg kell jelennie a Bizalmi Kör titkárságán, és ott be kell mutatnia

- a pénztárcájában lévő, megvásárolt zsetont,
- a személyi igazolványát,
- valamint egy igazolást arról, hogy a Bizalmi Kör bankszámlájára átutalta a $0,1\%$ tulajdonrész vételárát.

Ekkor megkapja a tulajdonrész-szerzést igazoló hivatalos iratokat, és bevonásra kerül tőle a zseton.

Amennyiben a zseton tulajdonosa a zseton megvásárlástól számított 4 napon belül nem él az opciós vételi jogával, akkor a zseton automatikusan visszazáll az okos szerződés tulajdonába, és ismét eladható.

Megjegyzés: a folyamat jelentősen egyszerűsödne, ha a zseton nem csupán opciós vételi jogot biztosítana, hanem magát a tulajdonrésztestesítene meg. De a jelenlegi jogi szabályozás miatt ekkor is szükség lenne a személyes adatok megadására.

7.2. A feladat elvi megoldása

A „Bizalmi Kör” tulajdonrész opciós vételi jogot megtestesítő zsetonok az Algorand blokkláncon egy ASA, azaz Algorand Standard Asset formájában állíthatók elő. Az ASA paraméterei:

- Rövid név: BKTOVJ
- Hosszú név: Bizalmi Kör Zseton
- Darabszám: 10 db
- Web lap címe: <https://algorand.hu/bk/bktovj.html>

Az ASA egy okos szerződéssel értékesíthető. Az okos szerződés feladatai:

- ASA eladása, “first come, first served” módon, fix áron. Az okos szerződés biztosítja, hogy a zseton vételárának kiegyenlítése után a zseton a vevő pénztárcájába kerüljön. A zseton birtoklása révén a vevő opciós vételi joghoz jut.
- „lejárt” ASA automatikus visszavétele: amennyiben a zseton tulajdonosa adott időn belül nem él az opciós vételi jogával, a zseton az okos szerződés számlájára kerül vissza. Ehhez a szerződés “clawback” műveletet hajt végre, és eggyel növeli az eladható ASÁ-k számát.
- opciós vételi jog érvényesítése: ha a vevő a „Bizalmi Kör” titkárságán bemutatta a zsetont és az egyéb szükséges igazolásokat, akkor a „Bizalmi Kör” ügyintézője „bevonja” a zsetont. Ehhez

a szerződés “clawback” műveletet hajt végre, de nem növeli az eladható ASÁ-k számát.

Számla címek:

- az okos szerződés létrehozójának Algorand címe
- az okos szerződéshez tartozó Algorand cím
- a vevő számlájának Algorand címe

Az ASA vételt a vevő kezdeményezi oly módon, hogy egy web lapon keresztül meghívja az okos szerződés ASA eladást végző metódusát. Ekkor egy WalletConnect felület összekapcsolja a vevő pénztárcáját az okos szerződéssel, és a vevő az okos szerződés által előállított tranzakció aláírásával képes megvenni a zsetont.

Ahhoz, hogy egyszerű legyen a “clawback” feltétel vizsgálata, az okos szerződés a zseton eladásakor „befagyasztja” a zsetont a vevő pénztárcájába. Így elég az aktuális idő és a vétel időpontjának különbségét figyelni, és ha az nagyobb egy adott időnél, jelen esetben 4 napnál, akkor végrehajtható a “clawback”.

Megjegyzés: a szerződés nem tud „magától” tranzakciókat kezdeményezni. A “clawback” funkció megvalósításához egy ütemezett folyamatnak naponta ki kell gyűjtenie a lejárt értéket birtokló számlákat, és meg kell hívnia a szerződés “clawback” funkcióját.

7.3. Az okos szerződés magyarázata

A feladatot megoldó Algorand alkalmazás vagy Algorand okos szerződés a BizKor.algo.ts fájlban található. Az okos szerződés TypeScript-ben készült, és a TealScript fordítóprogram segítségével fordítható le TEAL kóddá.

Hivatkozások:

- TEALScript forráskód
- TEALScript dokumentáció
- TEALScript példa programok

7.3.1. Az okos szerződés osztályának definiálása

```
1 import { Contract } from '@algorandfoundation/teal script';  
2 // verzió történet ...  
  
31 // eslint-disable-next-line no-unused-vars  
32 class BizKor extends Contract {  
33 // BizKor állapotváltozók és metódusok...  
34 }
```

A 1. sorban történik a `Contract` osztály beimportálása. A 32. sorban ebből származtatjuk a `BizKor` okos szerződés osztályt.

7.3.2. A globális állapotváltozók definiálása

```
31 // eslint-disable-next-line no-unused-vars
32 class Bizkor extends Contract {
33   appVersion = GlobalStateKey<string>({ key: 'apv' });
34
35   appCreatorAddress = GlobalStateKey<Address>({ key: 'apca' });
36
37   assetAmountInitial = GlobalStateKey<uint64>({ key: 'asa_total' });
38
39   assetAmount = GlobalStateKey<uint64>({ key: 'asa_amt' });
40
41   assetPrice = GlobalStateKey<uint64>({ key: 'asa_price' });
42
43   asset = GlobalStateKey<AssetID>({ key: 'asa_id' });
44
45   sellPeriodEnd = GlobalStateKey<uint64>({ key: 'end' });
46
47   assetValidityPeriod = GlobalStateKey<uint64>({ key: 'asa_v' });
```

A 33.47. sorokban a globális állapotváltozókat definiáljuk. Ezek kulcs-érték párok, és egy okos szerződésben 64 darab lehet belőlük.

A `GlobalStateKey` után meg kell adni a kulcs típusát: `string`, `uint64`, `Address`, `AssetID` stb. Ezt követően opcionálisan egy `{key: 'rövid-key-name'}` adható meg. Ennek a szerepe csupán a helytakarékosság, t.i. az okos szerződésen belül, az AVM kód ezzel a kulcsnévvel hivatkozik a kulcs-érték párra. Egy globális állapotnak a következőképpen tudunk értéket adni: `this.kulcsnév.value = érték`. Egy globális változó értékére a `this.kulcsnév.value` kifejezéssel tudunk hivatkozni.

7.3.3. `createApplication` – az okos szerződés létrehozása után meghívva

```
49 /**
50  * Init the values of global keys
51  */
52 createApplication(): void {
53   this.appVersion.value = 'v1.3';
54   this.appCreatorAddress.value = globals.creatorAddress;
55   this.assetAmountInitial.value = 0;
56   this.assetAmount.value = 0;
57   this.assetPrice.value = 0;
58   this.asset.value = AssetID.zeroIndex;
59   this.sellPeriodEnd.value = 0;
60   this.assetValidityPeriod.value = 0;
61 }
```

Az okos szerződés létrehozásakor a `createApplication` metódus is lefut. Ennek 53..60. soraiban történik meg a globális állapotváltozók inicializálása.

7.3.4. `bootstrap` – kezdeti paraméterek beállítása

```
63 /**
64  * create ASA, set global key values
65  * @param assetPrice ASA price in microAlgos
```

```

66  * @param assetAmount ASA initial amount in pieces
67  * @param sellPeriodLength sell period length in secs
68  * @param assetValidityPeriod asset validity in secs, after that time it can be clawbacked
69  */
70  bootstrap(assetPrice: uint64, assetAmount: uint64, sellPeriodLength: uint64,
↳   assetValidityPeriod: uint64) {
71    /// allow only the app creator to call this method
72    verifyAppCallTxn(this.txn, { sender: globals.creatorAddress });
73
74    /// assert bootstrap hasn't been called yet
75    assert(this.assetAmountInitial.value === 0);
76
77    // create asset
78    const asset = sendAssetCreation({
79      configAssetTotal: assetAmount,
80      configAssetDecimals: 0,
81      configAssetName: 'Bizalmi Kör Zseton',
82      configAssetUnitName: 'BKT0VJ1',
83      configAssetURL: 'https://algorand.hu/bk/bktovj.html',
84      configAssetDefaultFrozen: 0,
85      configAssetManager: globals.currentApplicationAddress,
86      configAssetReserve: globals.currentApplicationAddress,
87      configAssetFreeze: globals.currentApplicationAddress,
88      configAssetClawback: globals.currentApplicationAddress,
89    });
90
91    // set global values
92    this.assetAmountInitial.value = assetAmount;
93    this.assetAmount.value = assetAmount;
94    this.assetPrice.value = assetPrice;
95    this.asset.value = asset;
96    this.sellPeriodEnd.value = globals.latestTimestamp + sellPeriodLength;
97    this.assetValidityPeriod.value = assetValidityPeriod;
98  }

```

Az okos szerződés `bootstrap` metódusa felel az okos szerződés kezdeti paramétereinek beállításáért. A 70. sorban látható, hogy a metódus paramétereit szabályozzák a zseton árát, a létrehozandó zsetonok darabszámát, az értékesítési periódus hosszát másodpercekben megadva, és a zseton lejáratási periódust, szintén másodpercekben megadva.

A 72. sorban látható ellenőrzés biztosítja, hogy csak a szerződés létrehozója hívhassa meg ezt az eljárást.

A 75. sorban végzett ellenőrzés biztosítja, hogy a `bootstrap` metódust ne lehessen kétszer meghívni. Talán helyesebb lett volna egy külön csak erre a célra szolgáló, `already_bootstrapped` globális állapotváltozót felvenni erre a célra.

Az ASA létrehozása a 78..89. sorokban történik, egy belső tranzakció használatával. A 87. és 88. sorokban megadtuk a befagyasztási (“freeze”) és visszavonási (“clawback”) művelet végzésére szolgáló címet is, ami jelen esetben nem más, mint az okos szerződés Algorand címe.

A globális állapotváltozók beállítása a 92..97. sorokban történik. A `this.asset.value = asset` értékadásban nem csupán az asset azonosítóját, hanem az asset egyéb jellemzőit is tároljuk.

7.3.5. A globális állapotváltozók visszaolvasása

```
100  /**
101   * get app creator address
102   * @returns app creator address
103   */
104  getAppCreatorAddress(): Address {
105      return this.appCreatorAddress.value;
106  }
107
108  /**
109   * get app version
110   * @returns app version
111   */
112  getAppVersion(): string {
113      return this.appVersion.value;
114  }
115
116  /**
117   * get asa initial amount
118   * @returns asa amount minted initially
119   */
120  getAssetAmountInitial(): uint64 {
121      return this.assetAmountInitial.value;
122  }
123
124  /**
125   * get asa amount
126   * @returns sellable asa amount
127   */
128  getAssetAmount(): uint64 {
129      return this.assetAmountInitial.value;
130  }
131
132  /**
133   * get asa price
134   * @returns asa price in microAlgos
135   */
136  getAssetPrice(): uint64 {
137      return this.assetPrice.value;
138  }
139
140  /**
141   * get asa id
142   * @returns asa id
143   */
144  getAssetID(): AssetID {
145      return this.asset.value;
146  }
147
148  /**
149   * get end of sell period
150   * @returns end of sell period as absolute time in sec, from 01-Jan-1970
151   */
152  getSellPeriodEnd(): uint64 {
153      return this.sellPeriodEnd.value;
```

}

A globális állapotváltozók visszaolvasására külön metódusok szolgálnak, melyek az ABI szabályainak megfelelően elvégzik az adatkonverziót is. Ugyanakkor a `appClient.getGlobalState()` hívással „nyers” formában az összes állapotváltozó egyszerre beolvasható az app külön hívása nélkül is, vagyis a `getGlobalState()` hívás „sokkal olcsóbb”. Az `assetValidityPeriod` lekérdezésére szolgáló getter függvény véletlenül kimaradt a sorból.

7.3.6. buyAsset – zseton vétele

```

156  /**
157   * Buy 1 piece of the asset
158   * @param payment txn, where amount is equal to assetPrice, receiver is app address
159   */
160  buyAsset(payment: PayTxn): void {
161    /// Ensure asset selling period hasn't ended yet
162    assert(global.s.latestTimestamp <= this.sellPeriodEnd.value, 'Sell period ended');
163
164    /// Ensure that buyer hasn't bought earlier this asset
165    assert(this.txn.sender.assetBalance(this.asset.value) === 0, 'Asset already bought');
166
167    /// Verify payment transaction: receiver is the app, amount is the asset price
168    verifyPayTxn(payment, {
169      sender: this.txn.sender,
170      receiver: global.s.currentApplicationAddress,
171      amount: { greaterThanEqualTo: this.assetPrice.value, lessThanEqualTo: this.assetPrice.value
172        ↵ },
173    });
174
175    /// Is there still an asset to sell? (this can be optimized away)
176    assert(this.assetAmount.value > 0, 'No more ASA to sell');
177
178    /// Opt into asset, unconditionally
179    sendAssetTransfer({
180      xferAsset: this.asset.value,
181      assetAmount: 0,
182      assetReceiver: this.app.address,
183    });
184
185    /// Unfreeze asset
186    sendAssetFreeze({
187      freezeAsset: this.asset.value,
188      freezeAssetAccount: this.txn.sender,
189      freezeAssetFrozen: false,
190    });
191
192    /// Send asset to the buyer
193    sendAssetTransfer({
194      xferAsset: this.asset.value,
195      assetReceiver: this.txn.sender,
196      assetAmount: 1,
197    });
198
199    /// Freeze the asset at the buyer's address (this can be optimized away)
200    sendAssetFreeze({

```



```

200     freezeAsset: this.asset.value,
201     freezeAssetAccount: this.txn.sender,
202     freezeAssetFrozen: true,
203   });
204
205   // Decrease asset amount (this can be optimized away)
206   this.assetAmount.value = this.assetAmount.value - 1;
207 }

```

A zseton vételt a `buyAsset` metódus valósítja meg. Ennek egyetlen paramétere van, egy fizetési tranzakció, amely a zseton megvételére szolgál. A generált TEAL kód vizsgálatából látszik, hogy erre a tranzakcióra a metóduson belül mint egy tranzakciós csoport tranzakciójára történik hivatkozás. A tranzakciós csoport végrehajtásakor vagy az összes tranzakció végbemegy atomi módon, vagy egyetlen egy sem megy végbe. Az Algorand szerződés így biztosítja, hogy ha nem lép fel semmilyen hiba, akkor a fizetési tranzakció ellenében a zsetont megkapja a vevő, de ha valamilyen hiba történt, akkor nem megy végbe a fizetési tranzakció.

A `buyAsset` metódusban számos ellenőrzés történik:

- a 162. sor azt ellenőrzi, hogy a vétel időpontja az értékesítési periódus vége előtti van-e. Ha a feltétel nem áll fenn, akkor a metódus végrehajtása hibajelzéssel megszakad.
- a 165. sor azt ellenőrzi, hogy a vevőnek az eladandó zsetonból még egy sincs a birtokában (azaz „0 db van a birtokában”).
- a 168..172. sorok az ellenőrzik, hogy
 - a fizetési tranzakció küldője ugyanaz a cím-e, mint az okos szerződést meghívó tranzakció küldőjének a címe, lásd 169. sor,
 - a fizetési tranzakció címzettje megegyezik-e az okos szerződés Algorand címével, lásd 170. sor,
 - a fizetési tranzakcióban küldött összeg nem kisebb-e vagy nagyobb-e, mint a globális állapotváltozóban szereplő ár, lásd 171. sor
- a 175. sor azt ellenőrzi, hogy van-e még eladható zseton

Ezt követően a 178..182. sorokban a metódus „benevez” (opt-in) az ASÁ-ba egy belső tranzakció segítségével. Nem sikerült megoldani, hogy ne legyen szükség a metódus meghívása előtt külön benevezésre, „xxx” hibát kaptam, ezért ezek a sorok fölöslegesek. (Talán a 165. sorban lévő assertion miatt az opt-in műveletet az elé kellene rakni.)

A 185..189. sorok először feloldják a zseton befagyasztását, majd a 192..196. sorok elküldik a vevőnek a zsetont, végül a 199..203. sorok befagyasztják a vevő címén lévő zsetont.

Joggal kérdezheti az Olvasó, hogy mi szükség van a zseton befagyasztásának a feloldására, ha a vevő még nem is rendelkezik a zsetonnal? Erre a később ismertetett visszavonási folyamat miatt van szükség: ha a vevő vett egy zsetont, majd azt a lejáratát miatt visszavontuk, akkor a vevő címe „befagyasztott” állapotban marad, befagyasztott címre pedig nem lehet zsetont küldeni.

A `buyAsset` végül a 206. sorban csökkenti az eladható zsetonok számát nyilvántartó globális állapotváltozó értékét.

Megjegyzés: mivel a `buyAsset` négy belső tranzakciót használ, a metódus meghívásakor ezeknek a belső tranzakcióknak is ki kell fizetni a díját, vagyis a `buyAsset` metódust a minimális tranzakciós díj ötszörösével kell meghívni.

7.3.7. `sendAlgosToCreator` – zseton vételár visszaküldés

```
209  /**
210   * Send Algos from the app address to the app creator address
211   */
212  sendAlgosToCreator(): void {
213    /// Allow only the creator to call this method
214    verifyAppCallTxn(this.txn, { sender: global.s.creatorAddress });
215
216    /// Send back all the Algos above minAmount to the app creator
217    const minAmount = 600_000; // uAlgos
218    const balance = global.s.currentApplicationAddress.balance;
219    if (balance > minAmount) {
220      sendPayment({
221        receiver: global.s.creatorAddress,
222        amount: balance - minAmount,
223      });
224    }
225  }
```

A zsetonok vételárát a `sendAlgosToCreator` metódus hívásával lehet elküldeni az okos szerződésből az okos szerződés létrehozójának. Mint a `bootstrap` metódusnál, itt is ellenőrzi a metódus, hogy ki a hívó (lásd 214. sor), és csak az okos szerződés létrehozójának engedi meg a metódus futtatását.

Az okos szerződés címén szeretnénk, ha ott maradna 0.6 Algó, és csak az ezt meghaladó összeget utaljuk át az okos szerződés létrehozójának, lásd 219..223. sorok.

7.3.8. `clawback` – zseton visszavétele

```
227  /**
228   * Clawback asset to app & inc amount
229   * @param addr address from which to clawback asset
230   */
231  clawback(addr: Address): void {
232    /// Allow only the app creator to call this method
233    verifyAppCallTxn(this.txn, { sender: global.s.creatorAddress });
234
235    /// Clawback assets to app
236    sendAssetTransfer({
237      xferAsset: this.asset.value,
238      assetAmount: 1,
239      assetSender: addr,
240      assetReceiver: global.s.currentApplicationAddress,
241    });
242
243    /// Inc asset amount
244    this.assetAmount.value = this.assetAmount.value + 1;
245  }
```

Egy zseton visszavétele a `clawback` metódus hívásával valósítható meg. Paraméterként meg kell

adni, hogy melyik címről szeretnénk visszavenni a zsetont. A 236..241. sorokban történik meg a visszavétel: egy belső tranzakcióval elküldjük a zsetont a megadott címről a szerződésnek. Erre azért van lehetőségünk, mert a szerződés címét adtuk meg a zseton létrehozáskor a visszavonási címnek. A 244. sorban növeljük az eladható zsetonok számát.

Megjegyzés: egy külső, időzített eljárás ellenőrzi, hogy melyek a „lejárt” zsetonok, és a `clawback` metódus segítségével visszaszedi őket. Célszerű lenne a `clawback`-et kiegészíteni egy olyan ellenőrzéssel, hogy valóban lejárt-e a zseton, azaz az utolsó művelet (vásárlás) és a mostani művelet között eltelt-e már az `assetValidityPeriod` értéknek megfelelő időtartam.

7.3.9. `clawbackNoIncAmount` – zseton bevonása

Egy zseton bevonása a tulajdonrész sikeres megszerzésekor a `clawbackNoIncAmount` metódus meghívásával lehetséges. Egyezik a `clawback` metódussal, de ez nem növeli az eladható zsetonok számát.

7.3.10. `deleteAsset` – az ASA törlése

```
265  /**
266   * Delete asset with in app
267   */
268  deleteAsset(): void {
269    /// Allow only the app creator to call this method
270    verifyAppCallTxn(this.txn, { sender: global.s.creatorAddress });
271    // assert(this.txn.sender === this.app.creator, 'Allow only the app creator to call this
    ↪ method');
272
273    /// Delete asset
274    sendAssetConfig({
275      configAsset: this.asset.value,
276    });
277  }
```

Az okos szerződés törlésének előfeltétele, hogy az okos szerződés már ne rendelkezzen a létrehozott ASÁ-val. A `deleteAsset` metódus törli az ASÁ-t, a 274..276. sorokban látható belső tranzakcióval. Igazán érdekes, hogy épp ilyen `sendAssetConfig` hívással törölhető az ASA.

7.3.11. `deleteApplication` – az okos szerződés törlése előtt meghívva

```
279  /**
280   * Delete app with ABI method
281   */
282  deleteApplication(): void {
283    /// Allow only the app creator to call this method
284    verifyAppCallTxn(this.txn, { sender: global.s.creatorAddress });
285
286    /// Send back Algos to app creator account
287    sendPayment({
288      receiver: global.s.creatorAddress,
289      amount: 0,
290      closeRemainderTo: global.s.creatorAddress,
291    });
292  }
```

A szerződés törlése során a `deleteApplication` metódus kerül meghívásra. Ahhoz, hogy ez hibátlanul lefusson, a `deleteAsset` előzetes meghívása szükséges, a `deleteAsset` hibátlan lefutásának előfeltétele pedig az, hogy az okos szerződés címén legyen az összes zseton. A `deleteApplication` 287..291. sorában látható belső tranzakció elküldi a szerződés létrehozójának a címére a maradék Algorand-ot.

7.4. Az okos szerződés tesztelése

Az okos szerződés tesztelését végző Jest fájl a `BizKor.test.ts`.

Az Algokit Typescript segédprogramok között található `algorandFixture.ts` return értékei:

```
126 return {
127   get context() {
128     return context
129   },
130   get algorand() {
131     return algorandClient
132   },
133   beforeEach,
134 }
```

ahol a `context` a következő:

```
110 context = {
111   algod: transactionLoggerAlgod,
112   indexer: indexer,
113   kmd: kmd,
114   testAccount,
115   generateAccount: async (params: GetTestAccountParams) => {
116     const account = await getTestAccount(params, transactionLoggerAlgod, kmd)
117     algorandClient.setSignerFromAccount(account)
118     return { ...account, signer: algorandClient.account.getSigner(account.addr) }
119   },
120   transactionLogger: transactionLogger,
121   waitForIndexer: () => transactionLogger.waitForIndexer(indexer),
122   waitForIndexerTransaction: (transactionId: string) => runWhenIndexerCaughtUp(() =>
123     ↳ lookupTransactionById(transactionId, indexer)),
124 }
```

Ezt azt jelenti, hogy a `context`-ben megtalálható:

- egy `algod` algorand kliens
- egy `kmd` key management démon kliens
- egy `testAccount` teszt számlaszám
- egy `generateAccount` függvény
- egy `transactionLogger` függvény stb., stb.

Hivatkozások:

- Jest dokumentáció
- Algokit Typescript segédprogramok

- `algorandFixture.ts`

7.4.1. Az egyes Jest tesztek előtti beállítások

```

1  /* eslint-disable no-console */
2  import { describe, test, expect, beforeAll, beforeEach } from '@jest/globals';
3  import { algorandFixture } from '@algorandfoundation/algorand-utils/testing';
4  import * as algokit from '@algorandfoundation/algorand-utils';
5  import algosdk, { Transaction } from 'algosdk';
6  import { TransactionSignerAccount } from '@algorandfoundation/algorand-utils/types/account';
7  import { BizKorClient } from '../contracts/clients/BizKorClient';
8
9  const fixture = algorandFixture();
10 algokit.Config.configure({ populateAppCallResources: true });
11
12 let appClient: BizKorClient;
13
14 describe('BizKor', () => {
15   const log = false; // skip console.log() calls
16   const paramAppVersion = 'v1.3'; // app version
17   const paramAssetPrice = 1_000_000; // microAlgos
18   const paramAssetAmountInitial = 10; // pieces
19   const paramSellPeriodLength = 1000; // sec
20   const paramAssetValidityPeriod = 100; // sec
21
22   let acc1: algosdk.Account;
23   let signer1: TransactionSignerAccount;
24   let acc2: algosdk.Account;
25
26   beforeEach(fixture.beforeEach);
27
28   beforeAll(async () => {
29     await fixture.beforeEach();
30     const { algod, kmd } = fixture.context;
31
32     acc1 = await algokit.getOrCreateKmdWalletAccount(
33       {
34         name: 'Buyer of Biz. Kör. token',
35         fundWith: algokit.algos(100),
36       },
37       algod,
38       kmd
39     );
40     if (log) console.log('acc1.addr (token buyer):', acc1.addr);
41     // signer1 = algosdk.makeBasicAccountTransactionSigner(sender1);
42     signer1 = {
43       addr: acc1.addr,
44       // eslint-disable-next-line no-unused-vars
45       signer: async (txnGroup: Transaction[], indexesToSign: number[]) => {
46         return txnGroup.map((tx) => tx.signTxn(acc1.sk));
47       },
48     };
49
50     acc2 = await algokit.getOrCreateKmdWalletAccount(
51       {
52         name: 'App creator',

```

```

53     fundWith: algoKit.algos(100),
54   },
55   algo,
56   kmd
57 );
58 if (log) console.log('acc2. addr (app creator):', acc2.addr);
59
60 appClient = new BizKorClient(
61   {
62     sender: acc2,
63     resolveBy: 'id',
64     id: 0,
65   },
66   algo
67 );
68
69 await appClient.create.createApplication({});
70 });

```

A 2. sor a Jest teszteléshez szükséges függvények importját tartalmazza.

A 3. sor a Jest teszteléshez szükséges `algo` és `kmd` klienseket előállító `algorandFixture` importját tartalmazza.

A 4. sor az Algorand segédrutinokat tartalmazó `algoKit` importját tartalmazza.

Az 5. sor az `algosdk` API-t importálja.

A 6. sor a `TransactionSignerAccount` típust importálja.

A 7. sor a `BizKorClient`-et importálja, amely a `BizKor` okos szerződéshez definiálja a TypeScript típusokat.

A 9. sor a Jest fixture-t mint az `algorandFixture()`-t definiálja.

A 10. sor egy konfigurációs beállítás az okos szerződések hívásához.

A 12. sor definiálja az okos szerződés klienst, ami egy típusos kliens, a TEALScript fordítóprogram hozza létre az okos szerződés forrásából. A típusos kliens sokkal könnyebbé teszi az okos szerződés kezelését, módszerainak hívását, mert a TypeScript nyelv típusdefinícióival írja le az okos szerződést.

A 14. sorban kezdődik a tesztalalmaz leírása. A tesztek futtatása során a naplózás (`console.log` hívások) a 15. sorban tiltható vagy engedélyezhető. A 16..20. sorban az okos szerződés paramétereiket adjuk meg. A 22. sorban szereplő `acc1` lesz a vevő számlája, a 24. sorban szereplő `acc2` pedig az a számla, amely az okos szerződést létrehozza. A 23. sorban szereplő `signer1` az `acc1`-hez tartozó, aláírást végző szerkezet.

A 26. sorban `beforeEach(fixture.beforeEach)` azt jelenti, hogy *minden egyes teszt* előtt le kell futtatni a `fixture`-t, ami az `algorandFixture` futtatását jelenti.

A Jest tesztek előtt *csak egyetlen egyszer* fut le a 28..70. sor közötti rész. A `fixture.beforeEach()` (29. sor) előállítja a `context`-ben az `algo`, `kmd` stb. változókat.

A 30. sorban a `fixture.context`-ből visszanyerjük az `algo` `algo` kliens és `kmd` `kmd` kliens változókat.

A 32..39. sorban létrehozunk vagy visszanyerünk egy kmd számlát, és feltöltjük 100 Algóval. Ez az `acc1` számla lesz a vevőnek a számlája, aki a zsetont megveszi.

A 42..48. sorban készítünk egy `signer1` tranzakció aláíró a vevő számára. A `signer1` tranzakció aláíró tartalmazza a `acc1` számla címét (`acc1.addr`), valamint egy `signer` aláíró függvényt, mellyel az `acc1` nevében tranzakció csoportok írhatók alá.

Az 50..57. sorokban létrehozunk vagy visszanyerünk egy kmd számlát, és feltöltjük 100 Algóval. Ez az `acc2` számla lesz az a számla, amely az okos szerződést létrehozza.

A 60..67. sorokban létrehozuk az okos szerződéshez tartozó klienst. Mivel a megadott `id` paraméter 0, mindig új `appClient` jön létre.

Végül a 69. sorban létrehozuk az okos szerződést a blokkláncon, és a szerződés létrehozása után meghívjuk a `createApplication` metódust.

7.4.2. bootstrap teszt

```
72 test('bootstrap', async () => {
73   await appClient.fundAppAccount(algokit.microworld(600_000));
74   const assetPrice = paramAssetPrice;
75   const assetAmount = paramAssetAmountInitial;
76   const sellPeriodLength = paramSellPeriodLength;
77   const assetValidityPeriod = paramAssetValidityPeriod;
78   // fee must be paid for 2 transactions, due to the inner transaction
79   await appClient.bootstrap(
80     { assetPrice, assetAmount, sellPeriodLength, assetValidityPeriod },
81     { sendParams: { fee: algokit.transactionFees(2) } }
82   );
83   const globalState = await appClient.getGlobalState();
84   expect(globalState.asa_total?.asNumber()).toBe(assetAmount);
85   expect(globalState.asa_amt?.asNumber()).toBe(assetAmount);
86   expect(globalState.asa_price?.asNumber()).toBe(assetPrice);
87 });
```

A 79. sorban történik az alkalmazás/okos szerződés `bootstrap` metódusának hívása. Az első `{ }` között a metódus paramétereit kell megadni, a második `{ }` között pedig az okos szerződést hívó tranzakció paramétereit, itt a tranzakciós díjat. A tranzakciós díj a szokásos díj kétszerese, a metódusban lévő belső tranzakció miatt.

7.4.3. getAppVersion teszt

```
89 test('getAppVersion', async () => {
90   const version = await appClient.getAppVersion({});
91   expect(version.return).toBe(paramAppVersion);
92 });
```

7.4.4. getAppCreatorAddress teszt

```
94 test('getAppCreatorAddress', async () => {
95   const appCreatorAddress = await appClient.getAppCreatorAddress({});
96   expect(appCreatorAddress.return).toBe(acc2.addr);
```

```
97 });
```

7.4.5. getAssetAmountInitial teszt

```
99 test('getAssetAmountInitial', async () => {
100   const assetAmountInitial = await appClient.getAssetAmountInitial({});
101   expect(assetAmountInitial.return).toBe(BigInt(paramAssetAmountInitial));
102 });
```

7.4.6. getAssetAmount teszt

```
104 test('getAssetAmount', async () => {
105   const assetAmountInitial = await appClient.getAssetAmount({});
106   expect(assetAmountInitial.return).toBe(BigInt(paramAssetAmountInitial));
107 });
```

7.4.7. getAssetPrice teszt

```
109 test('getAssetPrice', async () => {
110   const assetPrice = await appClient.getAssetPrice({});
111   expect(assetPrice.return).toBe(BigInt(paramAssetPrice));
112 });
```

7.4.8. getAssetId teszt

```
114 test('getAssetId', async () => {
115   const assetId = await appClient.getAssetId({});
116   expect(assetId.return).toBeGreaterThan(BigInt(1_000));
117 });
```

7.4.9. getSellPeriodEnd teszt

```
119 test('getSellPeriodEnd', async () => {
120   const sellPeriodEnd = await appClient.getSellPeriodEnd({});
121   // get date/time
122   const now = new Date();
123   // get msec since 1970
124   const millisecondsSinceEpoch = now.getTime();
125   // get sec from msec
126   const secondsSinceEpoch = Math.floor(millisecondsSinceEpoch / 1000);
127   // check sellPeriodEnd
128   if (log) console.log('sellPeriodEnd: ', sellPeriodEnd.return);
129   expect(sellPeriodEnd.return).toBeGreaterThan(BigInt(secondsSinceEpoch)); // "al goki t local net
    ↪ reset" may be required
130   expect(sellPeriodEnd.return).toBeLessThan(BigInt(secondsSinceEpoch + paramSellPeriodLength));
131 });
```


7.4.10. getGlobalState teszt

```
133 test('getGlobalState', async () => {
134   const globalState = await appClient.getGlobalState();
135   const apv = globalState.apv!.asByteArray();
136   const apca = globalState.apca?.asByteArray();
137   const asaTotal = globalState.asa_total?.asNumber();
138   const asaAmt = globalState.asa_amt?.asNumber();
139   const asaPrice = globalState.asa_price?.asNumber();
140   const asalId = globalState.asa_id?.asNumber();
141   const end = globalState.end?.asNumber();
142   const asaV = globalState.asa_v?.asNumber();
143   // console.log('globalState:', globalState);
144
145   // get apvGood, i.e. without the length (first 2 bytes)
146   const apvGood = Buffer.from(apv).slice(2).toString('utf-8'); // get rid of length
147   if (log) console.log('apvGood: ', apvGood);
148   expect(apvGood).toBe(paramAppVersion);
149   // get apcaGood, i.e. encode 32 byte Algorand address as string
150   const bufferApca = Buffer.from(apca!);
151   if (log) console.log('bufferApca: ', bufferApca);
152   if (log) console.log('bufferApca.length: ', bufferApca.length);
153   const apcaGood = algosdk.encodeAddress(bufferApca); // encode as Algorand address
154   if (log) console.log('apcaGood: ', apcaGood);
155   expect(apcaGood).toBe(acc2.addr);
156
157   if (log) console.log('getGlobalState apv (appVersion):', apv);
158   if (log) console.log('getGlobalState apca (appCreatorAddress):', apca);
159   if (log) console.log('getGlobalState asa_total (assetAmountInitial):', asaTotal);
160   expect(asaTotal).toBe(paramAssetAmountInitial);
161   if (log) console.log('getGlobalState asa_amt (assetAmount):', asaAmt);
162   expect(asaAmt).toBe(paramAssetAmountInitial);
163   if (log) console.log('getGlobalState asa_price (assetPrice):', asaPrice);
164   expect(asaPrice).toBe(paramAssetPrice);
165   console.log('getGlobalState asa_id (asset):', asalId);
166   if (log) console.log('getGlobalState end (sellPeriodEnd):', end);
167   if (log) console.log('getGlobalState asa_v (assetValidityPeriod):', asaV);
168   expect(asaV).toBe(paramAssetValidityPeriod);
169 });
```

A 134. sorban a `getGlobalState()` hívással megtörténik az összes állapotváltozó beolvasása.

A 135. sor az `apv` (`appVersion`) értékét állítja elő, byte tömbként. A tömb elején lévő két byte az ABI string hosszát tárolja. Ettől a 146. sorban szabadulunk meg, és a 148. sorban ellenőrizzük, hogy azt kaptuk-e, amit a tesztek elején lévő paraméterek tartalmaznak.

A 136. sor az `apca` (`appCreatorAddress`) értékét állítja elő, byte tömbként. Az ebben lévő 32 byte az Algorand számlához tartozó publikus kulcs. Ebből az ASCII alakú cím kifejezését a 153. sor végzi.

A 137..142. sor a többi globális állapotváltozó értékét állítja elő, az `asNumber()` getter rutin segítségével. Ezekben az esetekben nincs átforgalmazási feladat.

7.4.11. opt in to asset teszt

```
171 test('opt in to asset', async () => {
172   const { algod } = fixture.context;
173   const params = await algod.getTransactionParams().do();
174   const globalState = await appClient.getGlobalState();
175   const asset = globalState.asa_id!.asNumber();
176   if (log) console.log('Try to opt in to asset: ', asset, acc1.addr);
177   const txn1 = algod.makeAssetTransferTxnWithSuggestedParamsFromObject({
178     from: acc1.addr,
179     to: acc1.addr,
180     amount: 0,
181     assetIndex: asset,
182     suggestedParams: params,
183   });
184   const stxn1 = txn1.signTxn(acc1.sk);
185   const txn2 = await algod.sendRawTransaction(stxn1).do();
186   await algod.waitForConfirmation(algod, txn2.txId, 4);
187 });
```

A 177..183. sorokban történik meg az `acc1.addr` cím benevezése (opt in) a zsetonba (ASA). A zseton azonosítóját a 175. sorban, a globális állapotból állítjuk elő.

7.4.12. buyAsset teszt

```
189 test('buyAsset', async () => {
190   const { algod, testAccount } = fixture.context;
191   const params = await algod.getTransactionParams().do();
192   // Make a payment tx, to buy asset
193   const appRef = await appClient.appClient.getAppReference();
194   // const appAddress = await algod.getApplicati onAddress(appRef.appId);
195   if (log) console.log('buyAsset: testAccount.addr ', testAccount.addr);
196   if (log) console.log('buyAsset: appRef.appAddress ', appRef.appAddress);
197   if (log) console.log('buyAsset: appCreatorAddr ', acc2.addr);
198   const tx1 = algod.makePaymentTxnWithSuggestedParamsFromObject({
199     from: acc1.addr,
200     to: appRef.appAddress,
201     amount: paramAssetPrice,
202     suggestedParams: params,
203   });
204
205   // Buy asset
206   const globalState = await appClient.getGlobalState();
207   const asset = globalState.asa_id!.asNumber();
208   const compose = appClient.compose().buyAsset(
209     {
210       payment: tx1,
211     },
212     {
213       sender: signer1,
214       sendParams: {
215         fee: algod.transactionFees(5),
216       },
217       assets: [Number(asset)],
218     }
219   );
```

```

219 );
220 // atc, build group, sign, send
221 const atc = await compose.atc();
222 const txs = atc.buildGroup().map((tx) => tx.txn);
223 const signed = await signer1.signer(
224   txs,
225   Array.from(Array(txs.length), (_, i) => i)
226 );
227 const txg = await algod.sendRawTransaction(signed).do();
228 await algodk.waitForConfirmation(algod, txg.txid, 4);
229 });

```

190. sor, `algod` betöltése, 191. sor, `params` tranzakciós paraméterek betöltése, 193. sor, `appRef` betöltése. 198..203. sor, fizetési tranzakció előállítás, a küldő cím az `acc1.addr`, a fogadó cím az `appRef.appAddress`.

206. sor, globális állapot beolvasása. 207. sor, a globális állapotból az `asa_id` előállítása. 208..219. sor, a `buyAsset` metódus hívás a `compose-on` keresztül. Figyelje meg a 215. sorban az 5-szörös tranzakciós díjat, és a 217. sorban az `assets` tömbben megadott `asset_id`-t. Ez utóbbi megadása nélkül "Unavailable asset" hibaüzenetet kapnánk. A 221. sor a tranzakciókat nyeri vissza, a 222. sor csoport azonosítóval látja el a tranzakciókat. A 223..226. sor aláírja a tranzakciókat.

A 225. sorban lévő konstrukció magyarázata, a ChatGPT4 szerint:

Az `Array.from(Array(txs.length), (_, i) => i)` egy tömböt hoz létre, amely a tranzakciók indexeit tartalmazza. Ez azt jelzi, hogy melyik tranzakciókat kell aláírni.

Az `Array(txs.length)` létrehoz egy új tömböt, amelynek hossza megegyezik a `txs` tömb hosszával. Kezdetben minden eleme `undefined`. Az `Array.from()` egy új tömböt hoz létre az adott tömbből. Ebben az esetben az első paraméter a létrehozott `undefined` elemekkel rendelkező tömb. A második paraméter egy függvény, amely minden elemre (itt `undefined`) és az indexére (`i`) meghívásra kerül. Ebben az esetben a függvény egyszerűen visszaadja az indexet (`i`), így a végeredmény egy tömb lesz, amely 0-tól kezdődően tartalmazza a tranzakciók indexeit.

Megjegyzés: a `buyAsset` hívása egyszerűsíthető, ha az `execute` függvénnyel végeztetjük el a tranzakciós csoport létrehozását, aláírását és elküldését. Ekkor a `buyAsset` teszt vége megváltozik:

```

208   const result = await appClient.compose().buyAsset(
209     { payment: tx1, },
210     {
211       sender: signer1,
212       sendParams: {
213         fee: algodk.transactionFees(5),
214       },
215       assets: [Number(asset)],
216     }
217   )
218   .execute();
219   await algodk.waitForConfirmation(result.txids[0], 4, algod);
220 });

```

Figyelje meg az Olvasó a 218. sorban lévő `execute()` függvényt.

A 213. sorban összesen 5 tranzakció díját kellett megfizetnünk az `algokit.transactionFees` függvénnyel: az `app.buyAsset` metódusának hívása 1 tranzakció, a `buyAsset` metóduson belül pedig további 4 belső tranzakció van.

7.4.13. buyAsset 2nd time teszt

```
231 test('buyAsset 2nd time', async () => {
232   const { algod, testAccount } = fixture.context;
233   const params = await algod.getTransactionParams().do();
234
235   // Make a payment tx, to buy asset
236   const appRef = await appClient.appClient.getAppReference();
237   // const appAddress = await algodk.getAppAddress(appRef.appId);
238   if (log) console.log('buyAsset: testAccount.addr ', testAccount.addr);
239   if (log) console.log('buyAsset: appRef.appAddress ', appRef.appAddress);
240   if (log) console.log('buyAsset: appCreatorAddr ', acc2.addr);
241   const tx1 = algodk.makePaymentTxnWithSuggestedParamsFromObject({
242     from: acc1.addr,
243     to: appRef.appAddress,
244     amount: paramAssetPrice,
245     suggestedParams: params,
246   });
247
248   // Buy asset
249   const globalState = await appClient.getGlobalState();
250   const asset = globalState.asa_id!.asNumber();
251   const compose = appClient.compose().buyAsset(
252     {
253       payment: tx1,
254     },
255     {
256       sender: signer1,
257       sendParams: {
258         fee: algokit.transactionFees(5),
259       },
260       assets: [Number(asset)],
261     }
262   );
263
264   const atc = await compose.atc();
265   const txs = atc.buildGroup().map((tx) => tx.txn);
266   const signed = await signer1.signer(
267     txs,
268     Array.from(Array(txs.length), (_, i) => i)
269   );
270   try {
271     await algod.sendRawTransaction(signed).do();
272   } catch (err) {
273     console.log('this test should fail, as the buyer already has a coin', err); //
274     ↪ err.response.body.data.pc);
275   }
276 });
```

Egyezik a `buyAsset` tesztrel, de itt az `app` hibát ad, mert a vevő már rendelkezik egy zsetonnal. A hiba kezelése a 272..274. sorban történik.

7.4.14. sendAlgosToCreator teszt

```
277 test('sendAlgosToCreator', async () => {
278   await appClient.sendAlgosToCreator({}, { sendParams: { fee: algokit.transactionFees(2) } });
279 });
```

7.4.15. clawback teszt

```
281 test('clawback', async () => {
282   await appClient.clawback({ addr: acc1.addr }, { sendParams: { fee: algokit.transactionFees(2)
283     ↪ } });
283 });
```

7.4.16. buyAsset after clawback teszt

```
285 test('buyAsset after clawback', async () => {
286   const { algod, testAccount } = fixture.context;
287   const params = await algod.getTransactionParams().do();
288   // Make a payment tx, to buy asset
289   const appRef = await appClient.getAppReference();
290   // const appAddress = await algosdk.getApplicationAddress(appRef.appId);
291   if (log) console.log('buyAsset: testAccount.addr ', testAccount.addr);
292   if (log) console.log('buyAsset: appRef.appAddress ', appRef.appAddress);
293   if (log) console.log('buyAsset: appCreatorAddr ', acc2.addr);
294   const tx1 = algosdk.makePaymentTxnWithSuggestedParamsFromObject({
295     from: acc1.addr,
296     to: appRef.appAddress,
297     amount: paramAssetPrice,
298     suggestedParams: params,
299   });
300
301   // Buy asset
302   const globalState = await appClient.getGlobalState();
303   const asset = globalState.asa_id!.asNumber();
304   const compose = appClient.compose().buyAsset(
305     {
306       payment: tx1,
307     },
308     {
309       sender: signer1,
310       sendParams: {
311         fee: algokit.transactionFees(5),
312       },
313       assets: [Number(asset)],
314     }
315   );
316   // atc, build group, sign, send
317   const atc = await compose.atc();
318   const txs = atc.buildGroup().map((tx) => tx.txn);
319   const signed = await signer1.signer(
320     txs,
321     Array.from(Array(txs.length), (_, i) => i)
322   );
323   const txg = await algod.sendRawTransaction(signed).do();
```

```
324     await algodk.waitForConfirmation(algod, txg.txId, 4);
325   });
```

7.4.17. clawback again teszt

```
327   test('clawback again', async () => {
328     await appClient.clawback({ addr: acc1.addr }, { sendParams: { fee: algodk.transactionFees(2)
329     ← } });
329   });
```

7.4.18. 'opt out buyer from asset' teszt

```
331   test('opt out buyer from asset', async () => {
332     const { algod } = fixture.context;
333     const params = await algod.getTransactionParams().do();
334     const globalState = await appClient.getGlobalState();
335     const asset = globalState.asa_id!.asNumber();
336     const appRef = await appClient.getAppReference();
337     if (log) console.log('Try to opt out from asset: ', acc1.addr);
338     const txn1 = algodk.makeAssetTransferTxnWithSuggestedParamsFromObject({
339       from: acc1.addr,
340       to: appRef.appAddress,
341       closeRemainderTo: appRef.appAddress,
342       amount: 0,
343       assetIndex: asset,
344       suggestedParams: params,
345     });
346     const stxn1 = txn1.signTxn(acc1.sk);
347     const txn2 = await algod.sendRawTransaction(stxn1).do();
348     await algodk.waitForConfirmation(algod, txn2.txId, 4);
349   });
```

A 338..345. sorban elküldi a vevő összes zsetonját az okos szerződésnek, úgy, hogy a `closeRemainderTo` mezőbe is az okos szerződés címét teszi. Ez egyenértékű azzal, hogy a vevő „kiszáll” (“opt out”) az adott zsetonból.

7.4.19. 'deleteAsset' teszt

```
351   test('deleteAsset', async () => {
352     await appClient.deleteAsset({}, { sendParams: { fee: algodk.transactionFees(2) } });
353   });
```

7.4.20. 'deleteApplication' teszt

```
355   test('deleteApplication', async () => {
356     await appClient.deleteApplication({}, { sendParams: { fee: algodk.transactionFees(2) }
357     ← });
357   });
```

7.5. A tesztek futtatása

A tesztek Codespaces alatt a következő parancsokkal futtathatók:

```
1  algokit --version
2  cd biz_kor/projects/biz_kor-contracts
3  npm install
4  npm audit
5  npm audit fix
6  npm run build
7  algokit localnet start
8  algokit localnet status
9  npm run build
10 # edit BizKor.algo.ts, see clawbackNolncAmount
11 # edit BizKor.algo.ts, see line 229, 250
12 npm run build
13 npm run test
```

A parancsok eredménye a következő:

```
1  Welcome to Codespaces! You are on our default image.
2  - It includes runtimes and tools for Python, Node.js, Docker, and more. See the full list here:
3  ↪ https://aka.ms/ghcs-default-image
4  - Want to use a custom image instead? Learn more here: https://aka.ms/configure-codespace
5
6  To explore VS Code to its fullest, search using the Command Palette (Cmd/Ctrl + Shift + P or F1).
7
8  Edit away, run your app as usual, and we'll automatically make it available for you to access.
9
10 @A-Maugli → /workspaces/akt02 (main) $ algokit --version
11 algokit, version 2.4.2
12 @A-Maugli → /workspaces/akt02 (main) $ ls
13 README.md  biz_kor  hellow
14 @A-Maugli → /workspaces/akt02/biz_kor (main) $ ls
15 README.md  biz_kor.code-workspace  projects
16 @A-Maugli → /workspaces/akt02/biz_kor (main) $ cd projects
17 @A-Maugli → /workspaces/akt02/biz_kor/projects (main) $ ls
18 biz_kor-contracts  biz_kor-frontend
19 @A-Maugli → /workspaces/akt02/biz_kor/projects (main) $ cd biz_kor_contracts
20 bash: cd: biz_kor_contracts: No such file or directory
21 @A-Maugli → /workspaces/akt02/biz_kor/projects (main) $ cd biz_kor-contracts/
22 @A-Maugli → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ ls
23 README.md  __test__  contracts  jest.config.js  package-lock.json  package.json  tsconfig.json
24 @A-Maugli → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ npm install
25
26 added 567 packages, and audited 568 packages in 13s
27
28 137 packages are looking for funding
29   run npm fund for details
30
31 2 vulnerabilities (1 moderate, 1 high)
32
33 To address all issues, run:
34   npm audit fix
35
```

```

36 Run npm audit for details.
37 npm notice
38 npm notice New minor version of npm available! 10.8.2 -> 10.9.0
39 npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.9.0
40 npm notice To update run: npm install -g npm@10.9.0
41 npm notice
42 @A-Maugli → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ npm audit
43 # npm audit report
44
45 braces <3.0.3
46 Severity: high
47 Uncontrolled resource consumption in braces - https://github.com/advisories/GHSA-grv7-fg5c-xmjj
48 fix available via npm audit fix
49 node_modules/braces
50
51 micromatch <4.0.8
52 Severity: moderate
53 Regular Expression Denial of Service (ReDoS) in micromatch -
54 → https://github.com/advisories/GHSA-952p-6rrq-rcjv
55 fix available via npm audit fix
56 node_modules/micromatch
57
58 2 vulnerabilities (1 moderate, 1 high)
59
60 To address all issues, run:
61   npm audit fix
62 @A-Maugli → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ npm audit fix
63
64 changed 3 packages, and audited 568 packages in 2s
65
66 137 packages are looking for funding
67   run npm fund for details
68
69 found 0 vulnerabilities
70 @A-Maugli → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ npm run compile-contract
71
72 > biz_kor-contracts@0.0.0 compile-contract
73 > tealscript contracts/*.algo.ts contracts/artifacts
74
75 Error when parsing tsdoc comment for clawback: Error: address is not an argument of clawback
76 Error when parsing tsdoc comment for clawbackNoIncAmount: Error: address is not an argument of
77 → clawbackNoIncAmount
78
79 /workspaces/akt02/biz_kor/projects/biz_kor-contracts/node_modules/node-fetch/lib/index.js:1501
80     reject(new FetchError(request to ${request.url} failed, reason:
81       → ${err.message}, 'system', err));
82         ^
83
84 FetchError: request to http://localhost:4001/v2/teal/compile?sourcemap=true failed, reason:
85   at ClientRequest.<anonymous> (/workspaces/akt02/biz_kor/projects/biz_kor-contracts/node_modules/
86     → s/node-fetch/lib/index.js:1501:11)
87   at ClientRequest.emit (node:events:519:28)
88   at emitErrorEvent (node:_http_client:108:11)
89   at Socket.socketErrorListener (node:_http_client:511:5)
90   at Socket.emit (node:events:519:28)
91   at emitErrorNT (node:internal/streams/destroy:169:8)
92   at emitErrorCloseNT (node:internal/streams/destroy:128:3)
93   at process.processTicksAndRejections (node:internal/process/task_queues:82:21) {

```



```

89     type: 'system',
90     errno: 'ECONNREFUSED',
91     code: 'ECONNREFUSED'
92   }
93
94 Node.js v20.17.0
95 @A-Maugli → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ npm run compile-contract
96
97 > biz_kor-contracts@0.0.0 compile-contract
98 > tealscript contracts/*.algo.ts contracts/artifacts
99
100 Error when parsing tsdoc comment for clawback: Error: Addr is not an argument of clawback
101 Error when parsing tsdoc comment for clawbackNoIncAmount: Error: Addr is not an argument of
102   ↳ clawbackNoIncAmount
103
104 /workspaces/akt02/biz_kor/projects/biz_kor-contracts/node_modules/node-fetch/lib/index.js:1501
105     reject(new FetchError(request to ${request.url} failed, reason:
106       ↳ ${err.message}, 'system', err));
107           ^
108
109 FetchError: request to http://localhost:4001/v2/teal/compile?sourcemap=true failed, reason:
110   at ClientRequest.<anonymous> (/workspaces/akt02/biz_kor/projects/biz_kor-contracts/node_modules_
111     ↳ s/node-fetch/lib/index.js:1501:11)
112   at ClientRequest.emit (node:events:519:28)
113   at emitErrorEvent (node:_http_client:108:11)
114   at Socket.socketErrorListener (node:_http_client:511:5)
115   at Socket.emit (node:events:519:28)
116   at emitErrorNT (node:internal/streams/destroy:169:8)
117   at emitErrorCloseNT (node:internal/streams/destroy:128:3)
118   at process.processTicksAndRejections (node:internal/process/task_queues:82:21) {
119     type: 'system',
120     errno: 'ECONNREFUSED',
121     code: 'ECONNREFUSED'
122   }
123
124 Node.js v20.17.0
125 @A-Maugli → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ algokit localnet status
126 # container engine
127 Name: docker (change with algokit config container-engine)
128 Error: LocalNet has not been initialized yet, please run 'algokit localnet start'
129 @A-Maugli → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ algokit localnet start
130 indexer has a new version available, run algokit localnet reset --update to get the latest
131   ↳ version
132 algod has a new version available, run algokit localnet reset --update to get the latest version
133 Starting AlgoKit LocalNet now...
134 docker: algod Pulling
135 docker: conduit Pulling
136 docker: indexer-db Pulling
137 docker: indexer Pulling
138 docker: proxy Pulling
139 docker: indexer Pulled
140 docker: conduit Pulled
141 docker: proxy Pulled
142 docker: algod Pulled
143 docker: indexer-db Pulled
144 docker: Network algokit_sandbox_default Creating
145 docker: Network algokit_sandbox_default Created
146 docker: Container algokit_sandbox_postgres Creating

```

```

142 docker: Container algokit_sandbox_algod Creating
143 docker: Container algokit_sandbox_algod Created
144 docker: Container algokit_sandbox_postgres Created
145 docker: Container algokit_sandbox_conduit Creating
146 docker: Container algokit_sandbox_conduit Created
147 docker: Container algokit_sandbox_indexer Creating
148 docker: Container algokit_sandbox_indexer Created
149 docker: Container algokit_sandbox_proxy Creating
150 docker: Container algokit_sandbox_proxy Created
151 docker: Container algokit_sandbox_algod Starting
152 docker: Container algokit_sandbox_postgres Starting
153 docker: Container algokit_sandbox_algod Started
154 docker: Container algokit_sandbox_postgres Started
155 docker: Container algokit_sandbox_conduit Starting
156 docker: Container algokit_sandbox_conduit Started
157 docker: Container algokit_sandbox_indexer Starting
158 docker: Container algokit_sandbox_indexer Started
159 docker: Container algokit_sandbox_proxy Starting
160 docker: Container algokit_sandbox_proxy Started
161 docker: Container algokit_sandbox_algod Waiting
162 docker: Container algokit_sandbox_conduit Waiting
163 docker: Container algokit_sandbox_postgres Waiting
164 docker: Container algokit_sandbox_indexer Waiting
165 docker: Container algokit_sandbox_proxy Waiting
166 docker: Container algokit_sandbox_proxy Healthy
167 docker: Container algokit_sandbox_postgres Healthy
168 docker: Container algokit_sandbox_algod Healthy
169 docker: Container algokit_sandbox_conduit Healthy
170 docker: Container algokit_sandbox_indexer Healthy
171 Started; execute algokit explore to explore LocalNet in a web user interface.
172 @A-Maugli → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ algokit localnet status
173 # container engine
174 Name: docker (change with algokit config container-engine)
175 # algod status
176 Status: Running
177 Port: 4001
178 Last round: 0
179 Time since last round: 0.0s
180 Genesis ID: dockernet-v1
181 Genesis hash: Uwfl iOj 9XZXEKs3GGWBebEtbE00zZPxGNVOXGC5za1w=
182 Version: 3.26.0
183 # conduit status
184 Status: Not running
185 # indexer-db status
186 Status: Running
187 # indexer status
188 Status: Not running
189 # proxy status
190 Status: Running
191 Error: At least one container isn't running; execute algokit localnet start to start the LocalNet
192 @A-Maugli → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ algokit localnet status
193 # container engine
194 Name: docker (change with algokit config container-engine)
195 # algod status
196 Status: Running
197 Port: 4001
198 Last round: 0

```

```

199 Time since last round: 0.0s
200 Genesis ID: dockernet-v1
201 Genesis hash: Uwfl iOj 9XZXEKs3GGWBebEtbE00zZPxGNVOXGC5za1w=
202 Version: 3.26.0
203 # conduit status
204 Status: Running
205 # indexer-db status
206 Status: Running
207 # indexer status
208 Status: Running
209 Port: 8980
210 Last round: 0
211 Version: 3.5.0
212 # proxy status
213 Status: Running
214 @A-Maugli i → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ npm run compile-contract
215
216 > biz_kor-contracts@0.0.0 compile-contract
217 > tealscript contracts/*.algo.ts contracts/artifacts
218
219 Error when parsing tsdoc comment for clawback: Error: Addr is not an argument of clawback
220 Error when parsing tsdoc comment for clawbackNoIncAmount: Error: Addr is not an argument of
↳ clawbackNoIncAmount
221 @A-Maugli i → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ npm run compile-contract
222
223 > biz_kor-contracts@0.0.0 compile-contract
224 > tealscript contracts/*.algo.ts contracts/artifacts
225
226 @A-Maugli i → .../akt02/biz_kor/projects/biz_kor-contracts (main) $ npm run test
227
228 > biz_kor-contracts@0.0.0 test
229 > npm run build && jest
230
231
232 > biz_kor-contracts@0.0.0 build
233 > npm run compile-contract && npm run generate-client
234
235
236 > biz_kor-contracts@0.0.0 compile-contract
237 > tealscript contracts/*.algo.ts contracts/artifacts
238
239
240 > biz_kor-contracts@0.0.0 generate-client
241 > algokit generate client contracts/artifacts/ --language typescript --output
↳ contracts/clients/{contract_name}Client.ts
242
243 Generating TypeScript client code for application specified in
↳ /workspaces/akt02/biz_kor/projects/biz_kor-contracts/contracts/artifacts/BizKor.arc32.json and
↳ writing to contracts/clients/BizKorClient.ts
244 Reading application.json file from path
↳ /workspaces/akt02/biz_kor/projects/biz_kor-contracts/contracts/artifacts/BizKor.arc32.json
245 Generating TS client for BizKor
246 Writing TS client to
↳ /workspaces/akt02/biz_kor/projects/biz_kor-contracts/contracts/clients/BizKorClient.ts
247 Operation completed successfully
248
249 console.info

```

```

250 LocalNet account 'Buyer of Biz.Kör. token' doesn't yet exist; created account
    ↳ ZKDY63B3QNQPZ6ZHD3HWTCG2MAYS00QRWVY6CLOQSXXPZY533YYWRCDAY with keys stored in KMD and
    ↳ funding with 100 ALGOs
251
252   at Object.getOrCreateKmdWalletAccount
    ↳ (node_modules/@algorandfoundation/src/localnet/get-or-create-kmd-wallet-account.ts: 51: 17)
253
254 console.debug
255 Transferring 100000000µALGOs from LM7ZQJLPE6QJ4NYBNPMZHEZXFLCNSY326650TX64MF2VKWJJKGYBTEIDJ4
    ↳ to ZKDY63B3QNQPZ6ZHD3HWTCG2MAYS00QRWVY6CLOQSXXPZY533YYWRCDAY
256
257   at Object.transferAlgos
    ↳ (node_modules/@algorandfoundation/src/transfer/transfer-algos.ts: 38: 46)
258
259 console.info
260 LocalNet account 'App creator' doesn't yet exist; created account
    ↳ J3U6032GZ66RNMP7IP50QCYPEHTYYS6YHHTXCHIKWUEDAIFVWRW7DDCL5U with keys stored in KMD and
    ↳ funding with 100 ALGOs
261
262   at Object.getOrCreateKmdWalletAccount
    ↳ (node_modules/@algorandfoundation/src/localnet/get-or-create-kmd-wallet-account.ts: 51: 17)
263
264 console.debug
265 Transferring 100000000µALGOs from LM7ZQJLPE6QJ4NYBNPMZHEZXFLCNSY326650TX64MF2VKWJJKGYBTEIDJ4
    ↳ to J3U6032GZ66RNMP7IP50QCYPEHTYYS6YHHTXCHIKWUEDAIFVWRW7DDCL5U
266
267   at Object.transferAlgos
    ↳ (node_modules/@algorandfoundation/src/transfer/transfer-algos.ts: 38: 46)
268
269 console.debug
270 Created app 1004 from creator J3U6032GZ66RNMP7IP50QCYPEHTYYS6YHHTXCHIKWUEDAIFVWRW7DDCL5U
271
272   at Object.createApp (node_modules/@algorandfoundation/src/app.ts: 115: 48)
273
274 console.debug
275 Transferring 600000µALGOs from J3U6032GZ66RNMP7IP50QCYPEHTYYS6YHHTXCHIKWUEDAIFVWRW7DDCL5U to
    ↳ SW4BTZGCCNMSDYSANRLKF00ZNYRKH4B7J6DNSEUAYPSRGOYZRGKQEAW5EY
276
277   at Object.transferAlgos
    ↳ (node_modules/@algorandfoundation/src/transfer/transfer-algos.ts: 38: 46)
278
279 console.log
280 getGlobalState asa_id (asset): 1008
281
282   at __test__/BizKor.test.ts: 165: 13
283
284 console.log
285 this test should fail, as the buyer already has a coin URLTokenBaseHTTPError: Network request
    ↳ error. Received status 400 (Bad Request): TransactionPool.Remember: transaction
    ↳ PCY7D6URKBX5QS7JCN7T6K6NDIJGFZFACRPMVYN5FI6M030SOPNQ: Logic eval error: assert failed
    ↳ pc=486. Details: app=1004, pc=486, opcodes=intc_1 // 0; ==; assert
286   at Function.checkHttpError (/workspaces/akt02/biz_kor/projects/biz_kor-contracts/node_modules/
    ↳ @algorandfoundation/src/types/urlTokenBaseHTTPClient.ts: 129: 11)
287   at processTicksAndRejections (node:internal/process/task_queues: 95: 5)
288   at Function.formatFetchResponse (/workspaces/akt02/biz_kor/projects/biz_kor-contracts/node_
    ↳ _modules/@algorandfoundation/src/types/urlTokenBaseHTTPClient.ts: 137: 5)

```

```

289   at AlgoHttpClientWithRetry.callWithRetry (/workspaces/akt02/bizkor/projects/bizkor-contract
290   ↪ acts/node_modules/@algorandfoundation/src/types/algo-http-client-with-retry.ts:30:20)
291   at AlgoHttpClientWithRetry.post (/workspaces/akt02/bizkor/projects/bizkor-contracts/node_
292   ↪ _modules/@algorandfoundation/src/types/algo-http-client-with-retry.ts:68:12)
293   at HTTPClient.post (/workspaces/akt02/bizkor/projects/bizkor-contracts/node_modules/algo
294   ↪ sdk/src/client/client.ts:269:19)
295   at SendRawTransaction.do (/workspaces/akt02/bizkor/projects/bizkor-contracts/node_modul_e_
296   ↪ s/algosdk/src/client/v2/algo/sendRawTransaction.ts:53:17)
297   ↪ {
298 response: {
299   body: {
300     data: [Object],
301     message: 'TransactionPool.Remember: transaction
302     ↪ PCY7D6URK BX5QS7JCN7T6K6NDI JGFZFACRPMVYN5FI6M030SOPNQ: logic eval error: assert
303     ↪ failed pc=486. Details: app=1004, pc=486, opcodes=intc_1 // 0; ==; assert'
304   },
305   status: 400,
306   headers: {
307     '291': 'content-length',
308     'keep-alive': 'connection',
309     'application/json; charset=UTF-8': 'content-type',
310     'Thu, 17 Oct 2024 17:11:57 GMT': 'date',
311     Origin: 'vary'
312   },
313   text: '{"data":{"app-index":1004,"eval-states":[{"stack":[0,0]}],"group-index":1,"pc":4
314   ↪ 86},"message":"TransactionPool.Remember: transaction
315   ↪ PCY7D6URK BX5QS7JCN7T6K6NDI JGFZFACRPMVYN5FI6M030SOPNQ: logic eval error: assert failed
316   ↪ pc=486. Details: app=1004, pc=486, opcodes=intc_1 // 0; ==; assert"}\n',
317   ok: false
318 },
319 status: 400
320 }
321
322   at __test__/BizKor.test.ts:273:15
323   at Generator.throw (<anonymous>)
324
325 PASS __test__/BizKor.test.ts (42.277 s)
326   BizKor
327     ✓ bootstrap (1498 ms)
328     ✓ getAppVersion (1445 ms)
329     ✓ getAppCreatorAddress (1380 ms)
330     ✓ getAssetAmountInitial (1529 ms)
331     ✓ getAssetAmount (1541 ms)
332     ✓ getAssetPrice (1410 ms)
333     ✓ getAssetId (1442 ms)
334     ✓ getSellPeriodEnd (1400 ms)
335     ✓ getGlobalState (1351 ms)
336     ✓ opt in to asset (1304 ms)
337     ✓ buyAsset (1835 ms)
338     ✓ buyAsset 2nd time (1381 ms)
339     ✓ sendAlgoToCreator (1139 ms)
340     ✓ clawback (1398 ms)
341     ✓ buyAsset after clawback (1518 ms)
342     ✓ clawback again (1408 ms)
343     ✓ opt out buyer from asset (1364 ms)
344     ✓ deleteAsset (1369 ms)
345     ✓ deleteApplication (1414 ms)

```

```
336
337 Test Suites: 1 passed, 1 total
338 Tests:      19 passed, 19 total
339 Snapshots:  0 total
340 Time:        42.389 s
341 Ran all test suites.
342 @A-Maugli → ... /akt02/biz_kor/projects/biz_kor-contracts (main) $
```

7.6. A frontend

Az `algokit` által generált frontend alkalmazás a React keretrendszer használatát feltételezi.

A generált React alkalmazás tartalmazza pl. a különféle pénztárcák kezelését végző modult (walletconnect interface-t használva), illetve példát ad egy Algorand app létrehozására, ill. metódusainak meghívására. Nem végzi el a fejlesztő helyett a grafikus felület megtervezését, és a grafikus felületen belül az Algorand app-ok megfelelő metódusainak helyes meghívását sem.

Szerencsére a `contract` kifejlesztése során megírt tesztek jól használhatók a React komponensek kifejlesztése során is, mert a megírandó kódot már nagyrészt tartalmazzák. Mi több, a TealScript `contract` fejlesztése során lehetőség van a frontend React komponensek vázának automatikus generálására is, az `npm run generate-components` parancs futtatásával. Megjegyzés: Ezt a generátort a jövőben nem kívánják tovább fejleszteni, kivezetésre fog kerülni.

Természetesen előfordul, hogy a tesztekhez képest a kód átszervezésére vagy jelentős bővítésére van szükség. A tesztek során pl. külön történt az Algorand app létrehozása (`createApplication`) és felparaméterezése (`bootstrap`), míg a React alkalmazásban mindez egy lépésbe vonható össze. Egy másik példa: a Biz. Kör. érme visszahívásakor (`clawback`) a tesztek során csak egy érmét kezeltünk. A React frontend viszont végignézi, hogy (1) milyen számlaszámok rendelkeznek az érmevel, és (2) ezen számlaszámok között van-e olyan, amelyre már lejárt a türelmi idő. Végül egy ciklusban visszahívja azokat az érméket, melyek túl régóta nem lettek felhasználva a tulajdonrész megvételére.

A React frontend megértéséhez szükség van a React alapjainak ismeretére. A React keretrendszerről a fejlesztők többek között a következő forrásokból tudnak tájékozódni:

- Nathan Rozentals: *Mastering TypeScript. Build enterprise-ready, modular web applications using TypeScript 4 and modern frameworks.* Packt, 4th edition, Chapter 12: React
- Adam Boduch, Roy Derks: *React and React Native. A complete hands-on guide to modern web and mobile development with React.js,* Packt, 3rd edition
- Űdemy, *Complete React Developer (w/ Redux, Hooks, GraphQL)*, 36 rész, 42 óra videó, lásd [udemy.com](https://www.udemy.com)

7.6.1. A frontend oldali npm scriptek

A `package.json` fájlban található script-ek:

```
1 $ cd bi_z_kor/projects/bi_z_kor-frontend
2 $ cat package.json
3 ...
4 "scripts": {
5   "generate:app-clients": "algokit project link --all",
6   "dev": "npm run generate:app-clients && vite",
7   "build": "npm run generate:app-clients && tsc && vite build",
8   "test": "jest --coverage --passWithNoTests",
9   "playwright:test": "playwright test",
10  "lint": "eslint src --ext ts,tsx --report-unused-disable-directives --max-warnings 0",
11  "lint:fix": "eslint src --ext ts,tsx --report-unused-disable-directives --max-warnings 0
12  ↪ --fix",
13  "preview": "vite preview"
14 }
```

A nekik megfelelő npm parancsok:

- `npm run generate:app-clients`, ami a `contract`-okhoz tartozó “typed client wrapper” fájlokat generálja le, és teszi a `src/contracts` könyvtárba. Emlékeztetjük rá az Olvasót, hogy ez egy TypeScript-ben megírt wrapper réteg, amely sokkal könnyebbé teszi az Algorand okos szerződések kezelését: az app létrehozását, módosítását, a metódusok meghívását, a tranzakciók vagy tranzakció csoportok létrehozását, aláírását stb. Mindezt már láttuk az okos szerződés tesztelésekor.
- `npm run dev`, ami gerálja a “typed client wrapper” fájlokat, majd elindítja a vite frontend fejlesztői rendszert. Ebben a „fejlesztői módban” a React alkalmazáshoz tartozó React forrás fájlok változásakor azonnal megtörténik az alkalmazás újrafordítása, és a React alkalmazást a vite szerver a `http://localhost:5137` url-en teszi elérhetővé a böngészőből.
- `npm run build`, ami gerálja a “typed client wrapper” fájlokat, majd elidítja a vite build parancsát, ami a rollup.js web packer segítségével a kész React alkalmazás kismillió fájlját néhány nagyobb js és css fájlba pakolja össze. Az eredmény a `dist` könyvtárba kerül.
- `npm run test`, ami a *.spec.tx kiterjesztésű Jest teszt fájlokat futtatja. Egy ilyen van pl. a `biz_kor-frontend/src/utis` könyvtárban, az `ellipseAddress.ts` tesztelésére, mintaként.
- `npm run playwright.test`, ami a böngésző szintjét teszteli az alkalmazást. A minta teszt file a `biz_kor-frontend/test/example.spec.ts` fájlban található. A futtatáshoz Python környezet is szükséges. Jelenleg ennek hiánya miatt elszáll.
- `npm run lint`, ami a formázási és szintaktikai hibákat ellenőrzi.
- `npm run lint:fix`, ami a formázási és szintaktikai hibákat ellenőrzi és javítja.
- `npm run preview`, ami elindítja a vite web szerveret oly módon, hogy a `dist` könyvtárban lévő, összekapolt React alkalmazást teszi elérhetővé a böngésző számára a `http://localhost:4173/` url-en.

7.6.2. A frontend fejlesztés menete

A `bizkor-frontend` fejlesztésekor mind a `contract`, mind a `frontend` használja a generált “typed client” fájlt. A wrapper generálása a `frontend` oldalon minden script parancsba be van építve:

```

1 "scripts": {
2   "generate:app-clients": "algokit project link --all",
3   "dev": "npm run generate:app-clients && vite",
4   "build": "npm run generate:app-clients && tsc && vite build",
5   "preview": "vite preview"
6 }

```

A `generate:app-clients` hatására történik meg a wrapper átmásolása a `frontend/src/contract` könyvtárba. A többi parancs is meghívja ezt a parancsot.

A TealScript `contract`-ok fejlesztésekor lehetőség van a `frontend` React komponensek automatikus generálására is, az `npm run generate-components` parancs futtatásával. Megjegyzés: Ezt a

generátort a jövőben nem kívánják tovább fejleszteni, kivezetésre fog kerülni.

7.6.3. Algorand specifikus részek

- pénztárca kezelés
- számlaszám kezelés
- algod, kmd, indexer kezelés

A fejlesztés során felmerült problémák és kérdések:

- Verziókezelés: Az `@algorandfoundation/tealscript` csomag "latest" verzióval szerepel a `package.json` fájlban. Csak a `package-lock.json` fájl tartalmazza a konkrétan használt verziót. Vigyázat: ha a `package-lock.json` fájlt töröljük, akkor előfordulhat, hogy az `npm install` már nem ugyanazt a verziójú TealScript-et installálja. Ez gondokat okozhat, ha pl. a hibakezelés a pc alapján szeretne felhasználóbarát hibaiüzeneteket kiírni, hiszen a TealScript újabb verziója másképpen generálhatja a TEAL kódot. Megoldás: (1) A `package-lock.json` file megtartása, vagy (2) az adott esetben használt TealScript verzió beírása a `package.json` fájlba.
- Generátor formázási probléma: Ha az Algorand TealScript szerződés ABC alakú, csupa nagybetűt tartalmazó név, akkor a generált typed client fájlneve `ABCClient.ts` lesz, de beimportálni `AbcClient` névvel tudjuk.
- Generátor formázási probléma: Az Algorand TealScript app-ban használt `abc_def` alakú, aláhúzás karaktert tartalmazó (globális) változók a `frontend` typed clientben `abcDef` változókként hivatkozhatók.
- React komponens tervezési kérdés: A "call bootstrap" gombra ill. komponensre külön nincs szükség, a "create DAO" komponensbe be lehet tenni a "bootstrap" metódus hívását is.
- React specifikus feladat: A "call bootstrap" hívása után frissíteni kell az "egy zseton ára", ill. "zsetonok száma" állapotváltozókat, amik a szülő komponensben vannak tárolva. Megoldása: a szülő komponens event handler-jét lett props-ként átadtam a gyermek komponensnek, majd meghívtam a gyermek komponensből. A szülő komponens event handler-jében az állapot megfrissíthető.

```
1 // src/components/BizkorBootstrap.tsx
2 type Props = {
3   buttonClass: string
4   //...
5   onClick?: React.MouseEventHandler<HTMLButtonElement>
6 }
7 //...
8   if (props.onClick) {
9     props.onClick(event);
10  }
```

```
1 // src/Home.tsx
2 // Get the price of tokens from app and store in state
3 const getPrice = async () => {
4   try {
5     const state = await typedClient.getGlobalState()
6     setPrice(state.asaPrice!.asNumber())
```

```

7     } catch (e: any) {
8         if (e.message !== "Couldn't find global state") {
9             console.warn(e)
10        }
11        setPrice(0)
12    }
13 }
14
15 const handleBootstrapButtonClick = async () => {
16     console.log('handleCallbackButtonClick is called')
17     await getPrice();
18 };
19
20 //...
21 <BizkorBootstrap
22     //...
23     onClick={handleBootstrapButtonClick}

```

- Algorand keretrendszer specifikus kérdés: hogyan lehet egy algod kliens hivatkozást szerezni?

Megoldás:

```

1 import { getAlgodConfigFromViteEnvironment } from '../utils/network/getAlgoClientConfigs'
2 import * as algokit from '@algorandfoundation/algokit-utils'
3
4 const algodConfig = getAlgodConfigFromViteEnvironment()
5 const algod = algokit.getAlgoClient({
6     server: algodConfig.server,
7     port: algodConfig.port,
8     token: algodConfig.token,
9 })

```

- Algorand keretrendszer specifikus kérdés: Hogyan lehet az optIn tranzakciót elküldeni?

Megoldás: a globális állapotból megállapítjuk az asset ID-t (4.sor), előállítunk egy olyan tranzakciót, amely 0 egységet küld az adott asset-ből (5..11. sor), és a sendTransaction-nal aláírjuk és elküldjük a tranzakciót (12. sor).

```

1 console.log( Opt in to asset )
2 const params = await algod.getTransactionParams().do();
3 const globalState = await props.TypedClient.getGlobalState();
4 const asset = globalState.asaId!.asNumber();
5 const txn1 = algod.makeAssetTransferTxnWithSuggestedParamsFromObject({
6     from: activeAddress!,
7     to: activeAddress!,
8     amount: 0,
9     assetIndex: asset,
10    suggestedParams: params,
11 });
12 algokit.sendTransaction({transaction: txn1, from: sender }, algod);
13 //await algokit.waitForConfirmation(txn1.txID(), 4, algod);

```

Megjegyzés: az, hogy az `algokit.sendTransaction` várakozik-e a txn elküldése után, vagy sem, az algokit beállításoktól függ.

- Algorand kerendszer specifikus kérdés: Hogyan lehet a buyAsset txn-t elküldeni?

Megoldás: a typed client “compose” metódusának meghívása szükséges, mert egy *tranzakciós csoport* előállítására van szükség. Ebben az első tranzakció a **payment** tranzakció, a második maga az Algorand okoszerződés **buyAsset** metódusának a hívása (1..12. sor). Az **execute** metódus (12. sor) aláírja a tranzakciós csoportban lévő tranzakciókat. Megjegyzés: meg kell adnunk a foreign assets tömbben (11. sor, **assets**: [...]) a Biz.Kör. érme asset ID-jét is, különben a hívás hibaüzenettel elszáll: "Asset not found".

```

1      const result = await props.typeClient.compose().buyAsset(
2        {
3          payment: tx1,
4        },
5        {
6          sender: sender,
7          sendParams: {
8            fee: algokit.transactionFees(5),
9          },
10         assets: [Number(asset)],
11       }
12     ).execute();
13     const waitRoundsToConfirm = 4;
14     await algokit.waitForConfirmation(result.txIds[0], waitRoundsToConfirm, algod);

```

- React specifikus kérdés: Hogyan adhatunk át egy payment txn-t prop-ként?

Megoldás: @todo

- Algorand keretrendszer specifikus kérdés: Hogyan lehet az appCreatorAddress-t visszanyerni?

Megoldás:

```

1      const appRef = await props.typeClient.appClient.getAppReference();
2      const appAddr = appRef.appAddress;

```

- Algorand keretrendszer specifikus kérdés: Hogyan lehet a price-t visszanyerni?

Megoldás: a globális állapotból:

```

1      const price = globalState.asaPrice!.asNumber();

```

- Algorand keretrendszer specifikus probléma: Hogyan lehet felhasználóbarát hibaüzeneteket küldeni?

Megoldás: A map fájlból kikereshető, hogy melyik **assert**-hez milyen TEAL pc tartozik. Hiba felléptekor a pc értéke alapján felhasználóbarát hibaüzenet íratható ki:

```

1      try {
2        enqueueSnackbar('A vételi tranzakció elküldése...', { variant: 'info' })
3        const result = await props.typeClient.compose().buyAsset(
4          //...
5        ).execute();
6        const waitRoundsToConfirm = 4;
7        await algokit.waitForConfirmation(result.txIds[0], waitRoundsToConfirm, algod);

```

```

8     enqueueSnackbar( A vételi tranzakció elküldve: ${result.txIds[0]} , { variant:
      ↪ 'success' })
9   } catch(e: any) {
10    const msg='Nem sikerült a tranzakció elküldése';
11    if (e.response.body.data.pc === 460) {
12      enqueueSnackbar( ${msg}, mert a tranzakció típusa nem fizetési tranzakció , {
        ↪ variant: 'error' })
13    }
14    else if (e.response.body.data.pc === 475) {
15      enqueueSnackbar( ${msg}, mert véget ért az értékesítési időszak , { variant: 'error'
        ↪ })
16    }
17    else if (e.response.body.data.pc === 486) {
18      enqueueSnackbar( ${msg}, mert Ön már rendelkezik ezzel a zsetonnal , { variant:
        ↪ 'error' })
19    }
20    else if (e.response.body.data.pc === 494) {
21      //...

```

- elvi probléma: a .env file-ban az admin_mode és az app_id változók szolgáltak arra, hogy szabályozzák, hogy (1) milyen szerepkörben vagyunk: a szerződés létrehozója esetén az admin_mode="true", míg a normál felhasználó esetén "false"; (2) létre kell-e hozni a szerződést (app_id=0), vagy a szerződés már korábban létre lett hozva (app_id=nnn).

Sajnos, a kész alkalmazásba ez a fájl is beépül, emiatt ez a megoldás nem használható.

Megoldás: processz env változók használata?

8. Algorit update, 2024. október 29.

8.1. git update

A `winget list` paranccsal állapítható meg, hogy van-e frissebb változat a korábban installált programokból, esetünkben a git-ből.

A git update megkezdése előtt a VS Code-ban be kell csukni a nyitott munkaterületet. Az update során a VS Code termináljában használt parancsok:

```
winget list --id Git.Git  
  
winget upgrade --id Git.Git  
  
winget list --id Git.Git  
  
git --version
```

Az update naplója:

```
1 PS C:\Users\l\ipi.F10> winget list --id Git.Git  
2 Name Id      Version Available Source  
3 -----  
4 Git  Git.Git 2.44.0 2.47.0 winget  
5 PS C:\Users\l\ipi.F10> winget upgrade --id Git.Git  
6 Found Git [Git.Git] Version 2.47.0  
7 This application is licensed to you by its owner.  
8 Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.  
9 Successfully verified installer hash  
10 Starting package install...  
11 The installer will request to run as administrator, expect a prompt.  
12 Successfully installed  
13 PS C:\Users\l\ipi.F10> winget list --id Git.Git  
14 Name Id      Version Source  
15 -----  
16 Git  Git.Git 2.47.0 winget  
17 PS C:\Users\l\ipi.F10> git --version  
18 git version 2.47.0.windows.1  
19 PS C:\Users\l\ipi.F10>
```

8.2. Docker Desktop update

A Docker Desktop verziójának megállapítása érdekében először el kell indítani a Docker Desktopot. Ezt a Docker Desktop ikonra történő dupla kattintással tehetjük meg.

Az update-hez használt VS Code terminál parancsok:

```
winget list --id Docker.DockerDesktop  
  
winget update --exact --id Docker.DockerDesktop
```

A Docker Desktop update 3–4 percig tart. Ezt követően a Windows újraindítása szükséges.

A Docker Desktop update naplója:

```
1 PS C:\Users\l\ipi.F10> winget list --id Docker.DockerDesktop
```



```

12 Starting package install...
13 The installer will request to run as administrator, expect a prompt.
14 Successfully installed
15 PS C:\Users\Lipi.FIO\Downloads> node --version
16 v23.1.0
17 PS C:\Users\Lipi.FIO\Downloads> npm --version
18 10.9.0

```

8.4. Python update

Állapítsuk meg az installált Python verziószámát, a `winget list python.python.3` paranccsal. Ez a Python 3.12.3 volt.

Nézzük meg a <https://www.python.org/downloads/> webhelyen, hogy mi az aktuálisan elérhető Python verziószáma. 2024. október 29-én a Python 3.13 volt elérhető.

Az update során használt parancsok:

```
winget uninstall python.python.3.12
```

```
winget install python.python.3.13
```

A VS Code újraindítása után:

```
python --version
```

=> 3.13.0

Megjegyzés: Hiba esetén érdemes a Path környezeti változót is ellenőrizni, mind a Felhasználó, mind a Rendszer részben, hogy nem maradt-e ott valahol az előző Python verzió útvonala.

8.5. pipx update

```
python -m pip uninstall pipx
```

A c:\Users\Lipi.FIO\pipx könyvtár törlése, majd:

```
python -m pip install pipx
```

8.6. algokit újrainstallálás

```
pipx install algokit
```

=> installed package algokit 2.4.3, installed using Python 3.13.0

=> These apps are now globally available

=> - algokit.exe

Ezt követően némelyik vírusellenőrző program, pl. a Bitdefender bizonyos file-okat fertőzöttnek vél, és karanténba helyez, pl.

```

1 File C:\Users\Lipi.FIO\pipx\venvs\algokit\libsite-packages\pywin32_system32\pythoncom313.dll file
  ↳ is infected, Gen: Variant.Tedy.659017
2 Bitdefender also quarantened the pywin32_system32\pythoncom313.dll file.

```

A karanténból ezeket a fájlokat vissza kell tenni az eredeti helyükre. Bitdefender esetén például ez a következőképpen tehető meg:

„Védelem” | „Beállítások” | „Karanténba helyezett fenyegetések”

A „Karantén kezelése” gomb megnyomásakor a Bitdefender kiírta, hogy milyen fájlok vannak karanténban.

Két file-t állítottam vissza: `pythoncom313.dll` és `pywintypes313.dll`.

A virustotal web helyen is ellenőrizhetők ezek a fájlok. A biztonsági cégeknek csupán töredéke (9/71) véli ezeket a fájlokat fertőzöttnek.

8.7. Utómunkák

1. Az `algokit init` a Python upgrade után nem találja a `python3.EXE` fájlt:

```
1 algokit init
2 ...
3 Unhandled CalledProcessError: Command
  ↳ '"C:\Users\Iipifl0\AppData\Local\Microsoft\WindowsApps\python3.EXE" pre_init.py' returned
  ↳ non-zero exit status 9009.
```

A helyes működéshez egy Admin módban elindított PowerShell-ben ki kell adnunk a következő parancsot:

```
1 New-Item -ItemType SymbolicLink -Path "C:\Windows\System32\python3.exe" -Target
  ↳ "C:\Users\Iipifl0\AppData\Local\Programs\Python\Python313\python.exe"
```

2. Az `algokit init` által generált frontend első futtatása során a következő hibát kapjuk:

```
1 npm run dev
2 ...
3 file:///C:/Users/Iipifl0/Downloads/2024/10/ALGO/Hackaton_FR/circle_of_trust/ct/projects/ct-fronte_
  ↳ nd/tailwind.config.js:2
4 module.exports = {
5   ^
6
7 ReferenceError: module is not defined
```

A megoldás az, hogy a `tailwind.config.js` fájlt átnevezzük `tailwind.config.cjs` fájlá.

3. Az update után célszerű a `AppData\Local\Temp` könyvtárban lévő fájlok törlése. Ezzel kb. 3,5 GByte hely szabadítható fel.