STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18-20-M/01: Informační technologie

Interakce s bitcoinovým protokolem

Mário Daniel Doskočil
Moravskoslezský kraj

Karviná 2022

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18-20-M/01: Informační technologie

Interakce s bitcoinovým protokolem

Interactions with the Bitcoin protocol

Autoři: Mário Daniel Doskočil
Škola: Střední škola teleinformatiky, Ostrava, příspěvková organizace Opavská 1119/12 708 61 Ostrava-Poruba
Kraj: Moravskoslezský kraj
Konzultant: Bez konzultanta

Karviná 2022

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Karviné dne 31.12.2022 ……

Mário Daniel Doskočil

Poděkování

Děkuji třídní učitelce Mgr. Petře Johnové za představení SOČ. Češtinářce Mgr. Ivaně Štěpánové a Ing. Adéle Swaczynové za kontrolu gramatických chyb a formální úpravy. Matce Marii Doskočilové za podporu a rovněž kontrolu gramatických chyb. Ing. Davidu Vašíčkovi za kontrolu formální úpravy. Škole za zapůjčení techniky na provoz lightningového uzlu, konkrétně učiteli Mgr. Pavlu Dědičíkovi.

Anotace

Práce popisuje základní kryptografické a algoritmické koncepty, které jsou stěžejní pro síť Bitcoin. V druhé části práce jsou tyto informace využity pro tvorbu webových aplikací, interagujících přímo s bitcoinovým uzlem, či úzce souvisejících s technologiemi Bitcoinu.

Klíčová slova

Bitcoin; Lightning Network; LN; Blockchain; Krypto;

Annotation

The paper describes the basic cryptographic and algorithmic concepts that are fundamental to the Bitcoin network. In the second part of the paper, information is used for a web application that interacts with a Bitcoin node.

Keywords

Bitcoin; Lightning Network; LN; Blockchain; Crypto;

Obsah

1 Úvod

Bitcoin, kryptoměna, blockchain, jsou pojmy, které dnes zná i širší veřejnost. Málokdo však ví, co tyto slova vlastě představují a jaká úchvatná technologie se za nimi skrývá. Práce popisuje základní koncepty Bitcoinu a možnosti interakce s ním prostřednictvím vlastních aplikací.

Práce se dělí na teoretickou a praktickou část. V teoretické části jsou popsány základní principy důležité pro pochopení Bitcoinu. V praktické části se pak tyto principy uplatňují v ukázkových webových aplikacích. Čtenář znalosti z teoretické části a příklady z praktické části, může využít k vývoji vlastní aplikace.

Pro testování aplikací byl sestaven lightningový uzel. Výstupem práce jsou tři aplikace, každá určená pro jinou skupinu uživatelů, ale všechny úzce související s Bitcoinem.

Aplikace byly zveřejněny na serveru github.com. Teoretická část je k dispozici ve formě markdownu, díky nástroji gitbook[1].

[1] https://www.gitbook.com

2 TEORETICKÁ ČÁST

Tato kapitola popisuje základní kryptografické a algoritmické koncepty, na kterých závisí technologie Bitcoinu.

2.1 Kryptografie

„U kryptografie je zapotřebí pochopit, že neexistuje žádné množství násilí, které vyřeší matematický problém." (Jacob Appelbaum, přeloženo autorem[2])

Kryptografie, také šifrování, je nauka o utajování dat. Zabývá se širokou škálou algoritmů pro utajení obsahu zprávy, ověření odesílatele, či samotného obsahu.

Obecně tyto algoritmy závisí na obtížnosti získání hodnoty, která může zvrátit funkci. Pro dešifrování informace zašifrované pomocí algoritmu spadajícího do **symetrické kryptografie** je zapotřebí znát stejný klíč, jenž byl použit k zašifrování informace. Algoritmus z **asymetrické kryptografie** vyžaduje pro dešifrování privátní klíč, který byl vygenerován v páru s použitým veřejným klíčem pro šifrování zprávy. Pro neprolomenou kryptografickou hashovací funkci neexistuje inverzní funkce.

2.1.1 Hashovací funkce

Hashovací funkce je matematickou funkcí, která bere libovolný datový řetězec a vrací jiný přesně stanovené délky – hash(otisk). Funkce musí mít minimální pravděpodobnost kolize, tedy že pro dva různé vstupy vrací stejný výstup. Obecně platí, že malou změnou ve vstupních datech je dosaženo velké změny na výstupu.

Kryptografická hashovací funkce musí navíc splňovat jednosměrnost – z výstupu nelze vypočíst vstup.

[2] Původní citát „One must acknowledge with cryptography no amount of violence will ever solve a math problem"

Matematicky je možné zapsat kryptografickou hashovací funkci jako $H: X \rightarrow Y$, kde X je nekonečná množina[3], Y je konečnou množinou. Pokud splňuje tyto podmínky:

$$\forall x \in X \text{ je snadné najít } H(x)$$

$$\forall y \in Y \text{ je obtížné najít } x \in X : H(x) = y$$

$$\forall x \in X \text{ je obtížné najít } x' \in X, x' \neq x : H(x) = H(x')$$

$$\text{je obtížné najít } x, x' \in X, x \neq x' : H(x) = H(x')$$

Příklad použití hashovací funkce SHA256 (Secure Hash Algorithm):

$$SHA256(Bitcoin) = B4056DF6691F8DC72E56302DDAD345D65FEAD3EAD9299609A826E2344EB63AA4$$

$$SHA256(bitcoin) = 6B88C087247AA2F07EE1C5956B8E1A9F4C7F892A70E324F1BB3D161E05CA107B$$

Kryptografická hashovací funkce tedy nalezne využití v případě potřeby uložení informace v nečitelné podobě, kterou je však snadné ověřit se znalostí původní hodnoty.

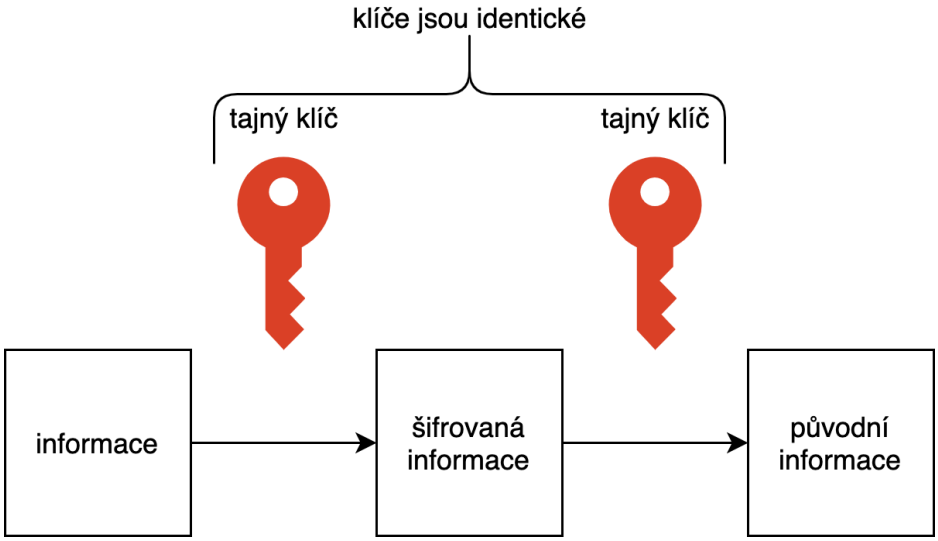
Uplatnění v Bitcoinové síti najde hashovací funkce zejména v procesu těžby. Zde je využita funkce SHA256d, která je rovna otisku z výstupu hashovací funkce SHA256.

$$SHA256d(x) = SHA256(SHA256(x))$$

2.1.2 Symetrická kryptografie

Symetrická kryptografie používá tentýž klíč k šifrování i dešifrování dat. Příjemce tedy musí znát klíč odesílatele, což je největší nevýhoda této šifry, jelikož při přenosu klíče může dojít k jeho odhalení.[1]

[3] Množina X je nekonečná, avšak množina Y konečná, tudíž funkce musí mít kolizní vstupy, avšak jsou extrémně obtížné k nalezení



Obrázek 1 - Šifrování informace symetrickou šifrou

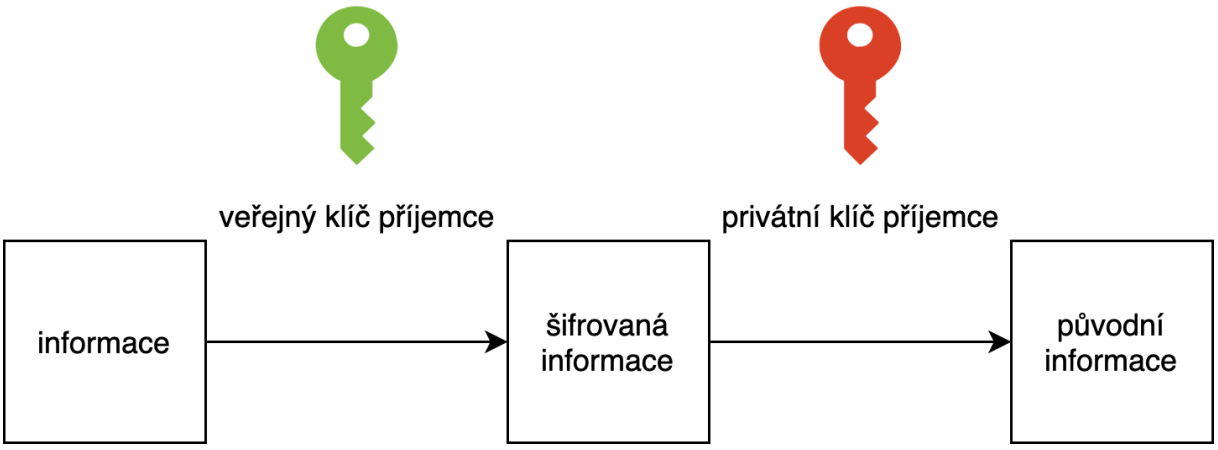
Šifru c je tedy možné vypočíst šifrovací funkcí $f: M, K \rightarrow C$, kde M je zpráva a K šifrovací klíč. Získat M je pak možné inverzní funkcí $f^{-1}: C, K \rightarrow M$.

$$c = f(m, k)$$

$$m = f^{-1}(c, k)$$

2.1.3 Asymetrická kryptografie

Asymetrická kryptografie využívá páru klíčů k zašifrování a rozšifrování dat. Jeden z klíčů je veřejný a druhý soukromý(privátní). Odesílatel šifruje data veřejným klíčem příjemce. Příjemce pak dešifruje data svým privátním klíčem.



Obrázek 2 - Šifrování informace asymetrickou šifrou

Šifru informace c- šifru je možné vypočítat pomocí funkce $f: M, K_v \rightarrow C$, kde M je zpráva a K_v je veřejný klíč příjemce. Inverzní funkcí je pak $f^{-1}: C, K_s \rightarrow M$, kde K_s je soukromý klíč příjemce.

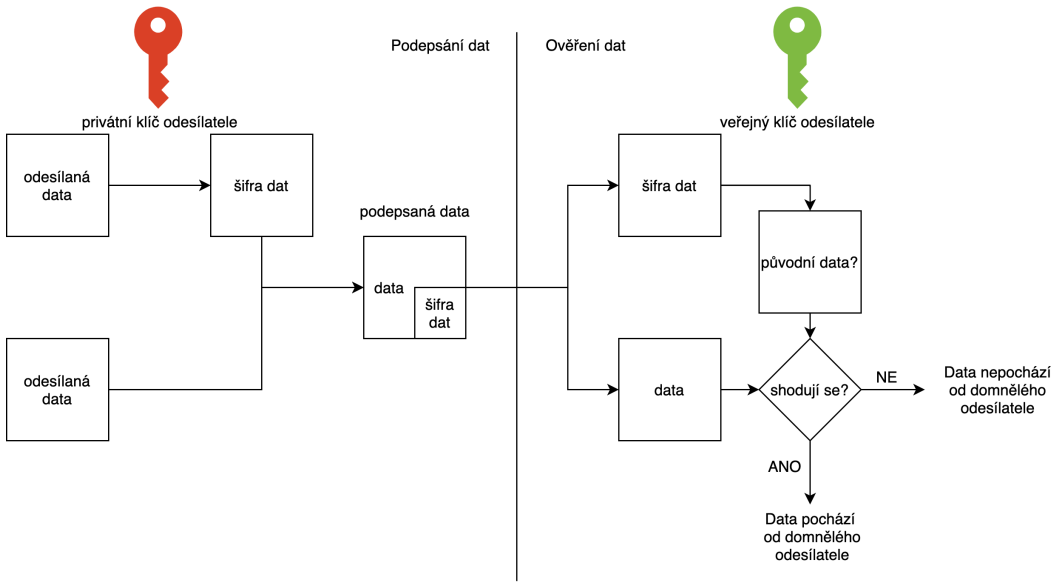
$$c = f(m, k_v)$$

$$m = f^{-1}(c, k_s)$$

Příkladem asymetrického šifrovacího algoritmu může být ECDH (Elliptic Curve Diffie-Hellman), který je založený na eliptických křivkách.

2.1.4 Digitální podpis

Digitální podpis umožňuje ověření původu dat. Využívá k tomu asymetrické kryptografie. Principem podpisu je zašifrování odesílaných dat soukromým klíčem odesílatele. Příjemce může ověřit pravost dat rozšifrováním přijatých dat pomocí veřejného klíče odesílatele.



Obrázek 3 - Ověření, zda data pochází od domnělého odesílatele

Každá platná transakce v bitcoinové síti musí být podepsána soukromým klíčem odesílatele.

2.2 Bitcoin

„Jestli tomu nevěříš nebo ti to nedochází, nemám čas, zkoušet tě o tom přesvědčovat, promiň.“ (Satoshi Nakamoto[4])

Bitcoin je decentralizovaná platební síť, bitcoin je pak její účetní jednotkou. Klíčová technologie tohoto konceptu se nazývá blockchain, který slouží jako nezměnitelná databáze transakcí. Algoritmus zabezpečující blockchain se pak nazývá Proof-of-Work. Obecná pravda v síti je určena podle objemu práce vynaložené na určení této pravdy. Tímto mechanismem je zajištěn konsensus všech uzlů v síti.

Jelikož je síť decentralizovaná – nemá centrální bod, neexistuje žádná konkrétní entita, která by ji řídila. Množství bitcoinů v síti je konečné a přesně určené ve zdrojovém kódu. Transakce nezvratitelné, necenzurovatelné a jejich pravdivost jednoduše ověřitelná. Prostředky jsou nezabavitelné. Podvod je prakticky nemožný vzhledem k fyzikálním vlastnostem výpočetních technologií. Pokus o podvod vyústí ve zbytečně vynaloženou práci, a tedy ke ztrátě prostředků.

Bitcoin umožňuje posílání transakcí bez prostředníka, v krátké době, s relativně nízkými poplatky[5], kdekoliv, kdykoliv a komukoliv na světě. Bitcoin představuje nový ekonomický koncept – peníze nezávislé na státě a jiných institucích, peníze na jejichž fungování mají vliv jen sami uživatelé. Tržní hodnota virtuálních prostředků se neodvíjí od ničeho jiného než od nabídky a poptávky. Hodnotu nemá samotná účetní jednotka – bitcoin, nýbrž celá platební síť, jedná se totiž o nejbezpečnější síť vůbec.[2]

Příjemce i odesílatel mohou zůstat neznámí. Jelikož jsou však transakce zapisovány do blockchainu, Bitcoin je označován jako pseudoanonymní. Všechny platby i vlastnictví je založeno na kryptografických důkazech. Je využíváno asymetrické kryptografie i kryptografické hashovací funkce.

[4] Původní citát: „If you don’t believe it or don’t get it, I don’t have the time to try to convince you, sorry.”

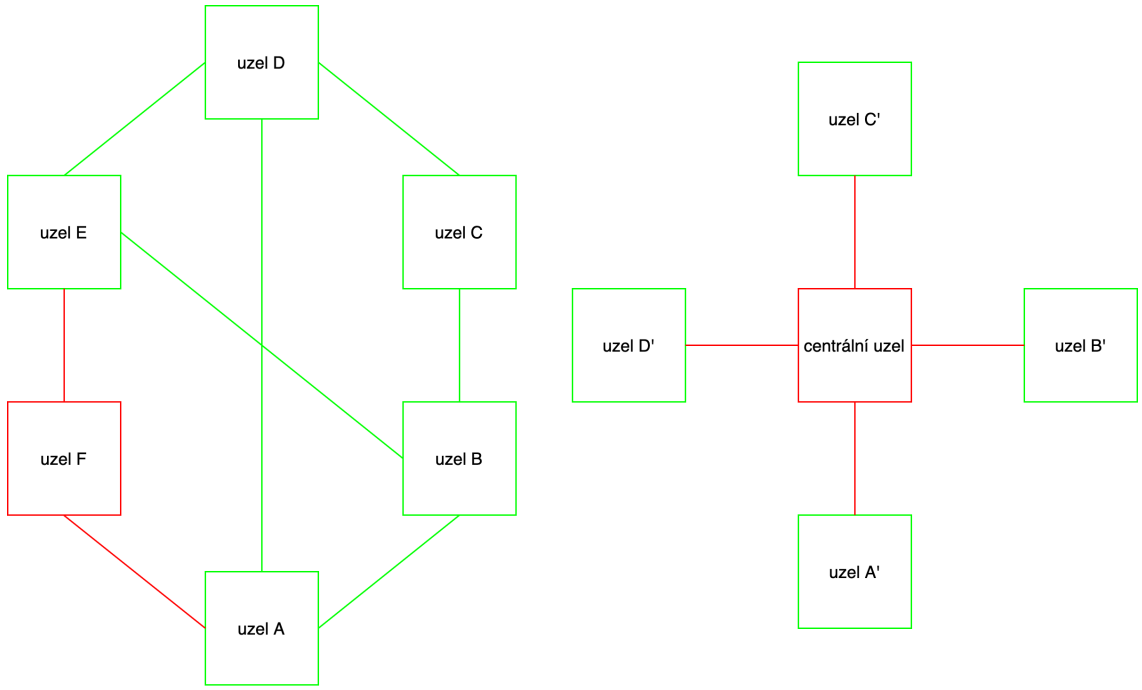
[5] Při využití off-chain řešení

2.2.1 Peer-to-Peer síť

Peer-to-peer (p2p, klient klientovi) síť je takový síťový model, ve kterém jsou si všechny uzly rovny právy a důležitostí. V p2p síti mezi sebou komunikují počítače navzájem bez zprostředkovatele komunikace. Neexistuje v ní žádný centrální uzel řídící síť, a tudíž je decentralizovaná.

Jednotlivé uzly se nazývají „peery“. Pro fungování sítě není zapotřebí přímé propojení všech uzlů. V centrálním síťovém modelu platí, že ztráta provozuschopnosti u centrálního uzlu vyústí v nefunkčnost celé sítě. Oproti tomu v p2p síti se mohou uzly libovolně připojovat a odpojovat, aniž by to mělo vliv na její provoz.

Na obrázku 4 je vlevo vyobrazena p2p síť, která je funkční i přes odpojení uzlu F ze sítě. Komunikace mezi uzly a a E je stále možná, stejně jako komunikace mezi kterýmikoli jinými uzly. Na obrázku 4 vpravo se nachází diagram centralizované sítě, která není schopná provozu z důvodu odpojení centrálního uzlu.

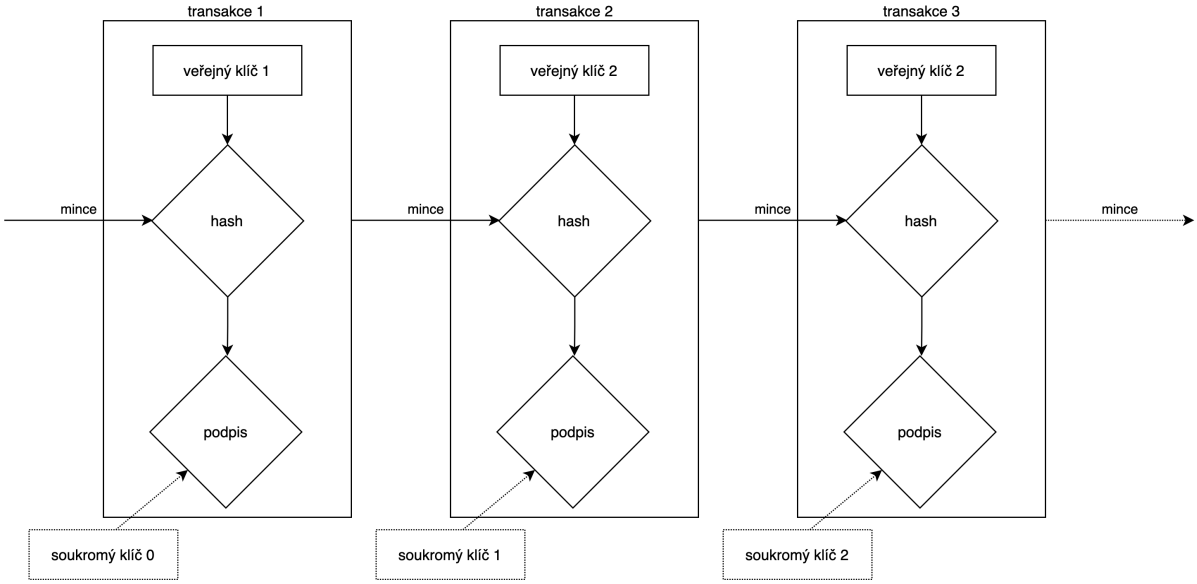


Obrázek 4 - Nalevo p2p síť, kde uzel F je odpojen. napravo centralizovaná síť, kde centrální uzel je odpojen

Pro připojení nového uzlu do sítě je zapotřebí, aby uzel odeslal požadavek na „DNS seed server," který poskytne IP adresy uzlů v síti. Adresy těchto DNS serverů jsou zapsány ve zdrojovém kódu.[3] Konkrétně v bitcoinové síti mezi sebou komunikují jednotlivé uzly za pomoci protokolu Tor[6], který zajišťuje anonymitu.[4]

2.2.2 Transakce

Transakce umožňuje přesun prostředků. K vytvoření transakce v síti je zapotřebí veřejný klíč příjemce a soukromý klíč odesílatele. Bitcoinová mince je definována jako řetězec digitálních podpisů. Každá transakce je podepsání hashe mince s veřejným klíčem příjemce, privátním klíčem odesílatele.



Obrázek 5 - Proces podepsání transakce

Po vytvoření hashe transakce je nutné vyslat jej do sítě. Podepsaný hash se přes libovolný uzel v síti, posílá do mempoolu.

Mempool[7] je označení pro „místo" kde čekají nevyřízené transakce. Těžaři z těchto transakcí poté sestavují bloky, podle velikosti poplatků.

Účetní jednotku bitcoin je možné dělit na stomiliontiny, kdy se jedna stomiliontina nazývá satoshi [sat].

---

[6] Oficiální stránky projektu: https://www.torproject.org

[7] Mempool explorer https://mempool.space/cs/

Transakční poplatek je určen velikostí transakce. Velikost transakce se uvádí ve veličině virtuální velikosti s jednotkou vbyte [VB]. Každá transakce se skládá ze vstupů a výstupů. Obojí je označováno jako neutracený transakční vstup tzv. UTXO. Transakční poplatek se vypočítá následovně:

T_s- počet segwitových bajtů
T_n- počet nesegwitových bajtů
T_{WU}- váha transakce
T_{VB}- virtuální velikost transakce
T_p- poplatek v sat na vbyte

$$T_{WU} = T_n \cdot 4 + T_s$$
$$T_{VB} = \frac{T_{WU}}{4}$$
$$T_p = \frac{\text{rozdíl vstupů a výstupů transakce}}{T_{VB}}$$

Celkový poplatek v satoshi je tedy roven $T_p \cdot T_{VB}$, což odpovídá rozdílu mezi vstupy a výstupy transakce. Hodnotu poplatku v sat/vbyte si určuje odesílatel.

2.2.3 Adresy

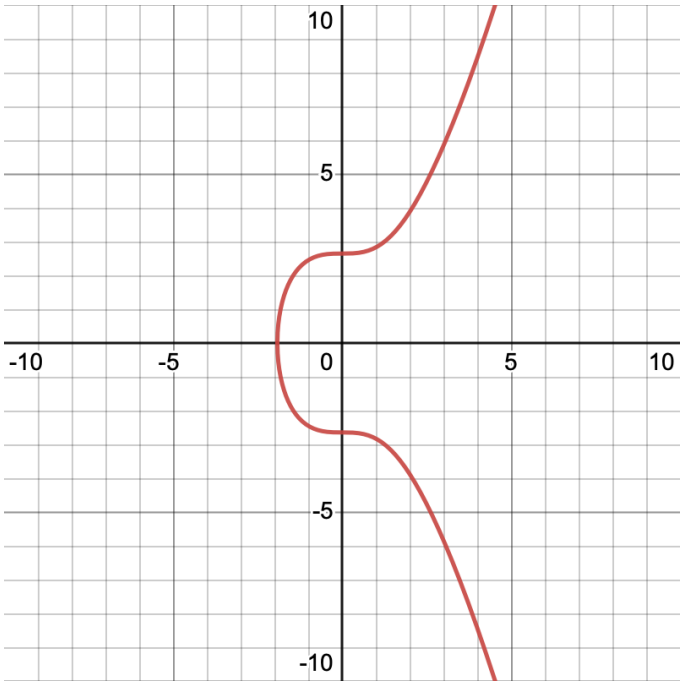
Adresa slouží jako alternativa za veřejný klíč v transakcích. Pro každou transakci je možné, a žádoucí generace nové adresy. Následující tabulka uvádí nejrozšířenější typy adres.

Tabulka 1 - Typy bitcoinových adres[5]

Typ	Začátek adresy	Označení
P2PKH	1	LEGACY
P2SH	3	SEGWIT
P2WPKH	bc1q	NATIVE SEGWIT
P2TP	bc1p	TAPROOT

Segwitové transakce, zabírají méně prostoru v bloku, jsou proto levnější.[6]

Adresa je generována z veřejného klíče, který je generován ze soukromého klíče. Bitcoin pro tento účel využívá eliptické křivky secp256k1 definované jako $y^2 = x^3 + 7$ přes pole celých čísel $\mathbb{Z}_p$, kde $p = 2^{256} - 2^{32} - 977$, tedy největší prvočíslo do 2^{256}. Počáteční bod křivky G, je definován v hexadecimálním tvaru ve specifikacích křivky.[7][8]



Obrázek 6 - Eliptická křivka secp256k1 definovaná přes pole reálných čísel

Tabulka 2 - Operace na eliptické křivce definované přes konečné pole[7]

Sčítání bodů ECAdd	Zdvojení bodu ECDouble
$\lambda = (y_G - y)modinv(x_G - x)(mod\ p)$ $x_R = \lambda^2 - x - x_G(mod\ p)$ $y_R = \lambda(x - x_R) - y(mod\ p)$	$\lambda = (3x^2)modinv(2y)(mod\ p)$ $x_R = \lambda^2 - 2x(mod\ p)$ $y_R = \lambda(x - x_R) - y(mod\ p)$

Využitím těchto dvou operací je dosaženo operace „násobení“ bodů.

Postup generace adresy:[7][9]

1. Vygenerování náhodného celého čísla $n \in \mathbb{Z} - ((-\infty; 1) \cup (10^{77} - 1;\ +\infty))$
2. n je privátním klíčem
3. Výpočtu souřadnic $[x, y]$ dosáhneme násobením n s G
4. Veřejný klíč (V_k) je výsledkem operace řetězení souřadnic výsledného bodu: V_k = x || y
5. Veřejný klíč nekomprimovaný (V_{kn}) získáme následovně:
   5.1. Verze veřejného klíče = $04||V_k$
   5.2. $C_1 = SHA256$(verze veřejného klíče)
   5.3. $C_2 = RIPEMD160$[8](C_1)
   5.4. Otisk verze klíče je roven 0x00$||C_2$
   5.5. $C_3 = SHA256$(otisk verze klíče)
   5.6. $C_4 = SHA256(C_3)$
   5.7. Kontrolní součet otisku tvoří prvních osm znaků C_4
   5.8. Kontrolní součet klíče je roven řetězení Otisk verze klíče || Kontrolní součet otisku
   5.9. $V_{kn} = Base58$[9](kontrolní součet klíče)
6. Veřejný klíč komprimovaný (V_{kk}) získáme:
   6.1. Pokud je souřadnice y sudá, přepona je rovna 02, jinak 03
   6.2. Verze veřejného klíče = přepona || souřadnice x v hexadecimálním tvaru
   6.3. $D_1 = SHA256$(verze veřejného klíče)
   6.4. $D_2 = RIPEMD160(D_1)$
   6.5. Otisk verze klíče je roven 0x00$||D_2$
   6.6. $D_3 = SHA256$(otisk verze klíče)
   6.7. $D_4 = SHA256(D_3)$
   6.8. Kontrolní součet otisku tvoří prvních osm znaků D_4
   6.9. Kontrolní součet klíče je roven řetězení Otisk verze klíče || Kontrolní součet otisku

[8] https://en.bitcoin.it/wiki/RIPEMD-160

[9] wikipedia.org/wiki/Binary-to-text_encoding#Base58

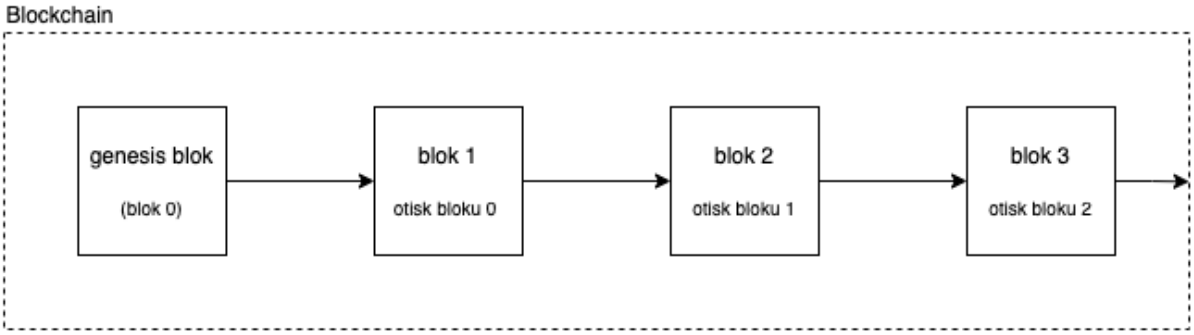
6.10. $V_{kn} = Base58$(kontrolní součet klíče)

7. V_{kn} i V_{kk} jsou nyní adresy typu P2PKH (pay to public key hash)

Dalšími úpravami je možné z veřejného klíče získat jiný typ adresy.

2.2.4 Blockchain

Blockchain (česky bločenka[10]) je datová struktura sloužící k uchovaní a čtení dat. Ke změně již zapsaných dat je však zapotřebí vynaložit velký výpočetní výkon. Blockchain je složen z jednotlivých "bloků" obsahujících transakce. Všechny uzly (full node) v síti mají uloženou jeho kopii a vidí tedy všechny dosavadní transakce. Důležitou vlastností je tedy neměnnost. Každý blok má přesně daný obsah a pořadí.[10][11]



Obrázek 7 - Vizualizace principu blockchainu

Na bitcoinovém blockchainu je možné provádět softforky, tedy zpětně kompatibilní změny protokolu. Což je rozdíl oproti hardforkům, kdy dochází ke vzniku nové kryptoměny.

První blok v blockchainu (blok 0) se nazývá "genesis blok" a jako jediný neobsahuje otisk předchozího bloku, místo něj je dosazen řetězec 64 nul.[12]

Blok je složen z hlavičky (header) a transakcí. Hlavička bloku obsahuje otisk bloku předchozího. Její velikost je 640 bitů = 80 bytů. Maximální teoretická velikost transakcí v bloku je omezena na 4 MB.[16] Hexadecimální reprezentaci celé hlavičky bloku je možné popsat následovně:[13][14]

04e0ff3feb36c62f0471cee034811019e43b14f459b50e00cea30a000000000000
00000659cecf4a06ed500031b741384e87d40ce5c16c3ec8c09b09ffe4b863c218d
1f282d3c61e4480f17d767c2ab

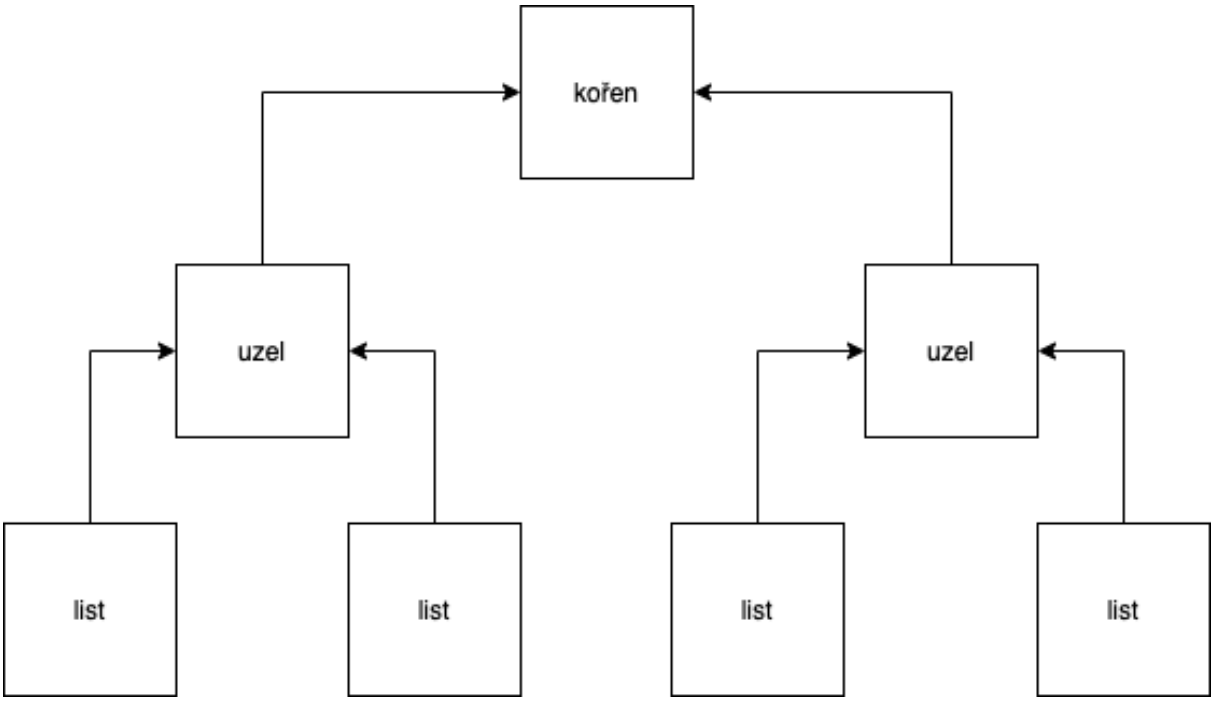
[10] „bločenka" https://web.archive.org/web/20210116033215/https://www.blockchain.cz/

Tabulka 3 - Popis hlavičky bloku

Komponent	Význam	Délka v bitech	Délka v hex reprezentaci
Verze bloku	Podpora softforků[15]	32 bitů	8 znaků
Hash předchozího bloku	Řetězení/návaznost bloků	256 bitů	64 znaků
Hash kořene Merkelova stromu	Důkaz o transakcích	256 bitů	64 znaků
Časové razítko (timestamp)	Razítko pro zařazení	32 bitů	8 znaků
Cíl těžby (target)	Určuje obtížnost	32 bitů	8 znaků
Nonce	Mění se pro získání správného otisku	32 bitů	8 znaků

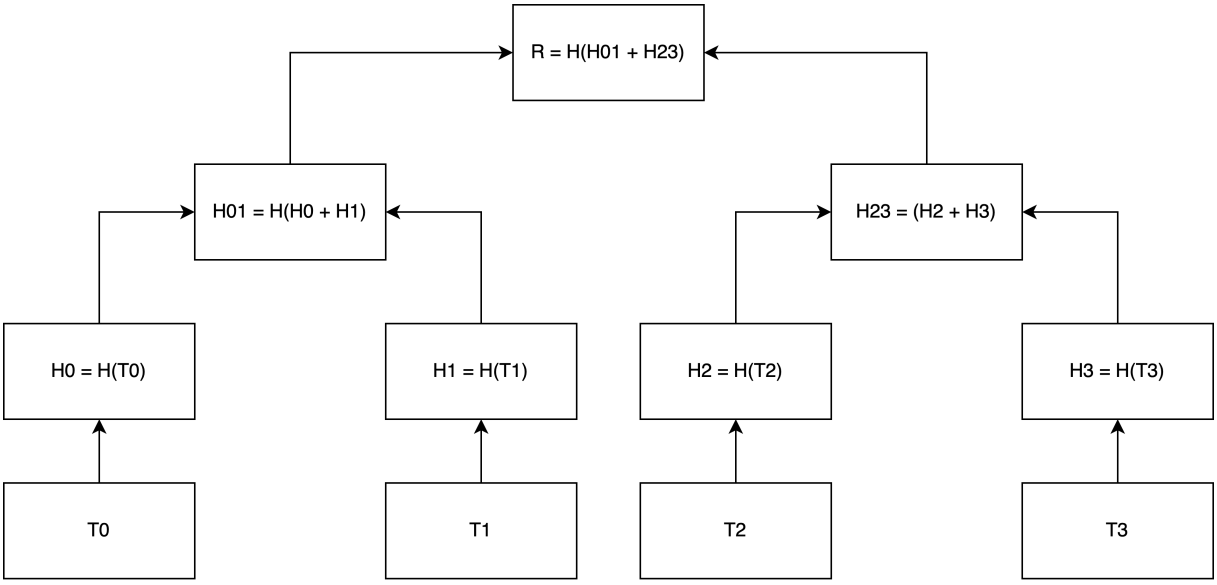
2.2.4.1 Hashové stromy

Merkleův strom, nebo také hashový strom, je typ binárního stromu. Datová struktura slouží k takovému uložení dat, aby jejich integritu bylo možné ověřit pomocí co nejmenšího počtu poskytnutých dat ve formě hashů. Strom se skládá z kořene, uzlů a listů.[17]



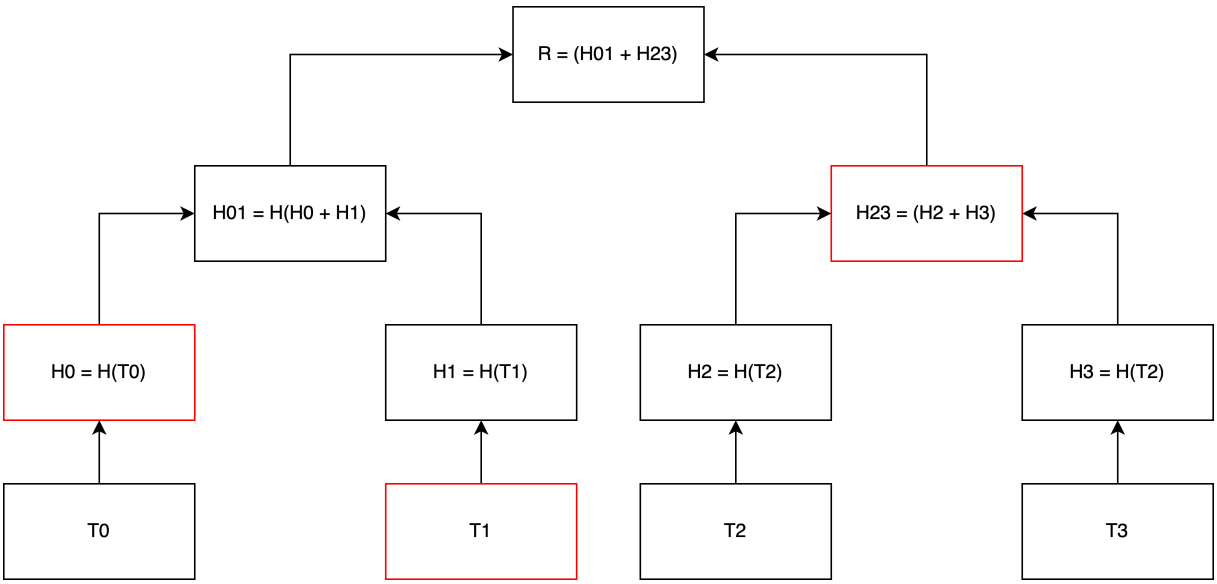
Obrázek 8 – Struktura hashového stromu

V "listech" jsou uložena požadovaná data. Každý uzel obsahuje hash součtu otisků předchozích uzlů nebo listů. Kořen je tedy hash všech prvků ve stromě. Každý uzel musí mít dva vstupy, tedy listy nebo jiné uzly. V případě lichého počtu listů/uzlů se poslední list/uzel páruje sám se sebou.[18]



Obrázek 9 - Příklad hashového stromu s transakcemi

Na obrázku 9 se nachází příklad hashového stromu se čtyřmi listy, T0 až T3. Pro ověření integrity dat je potřeba znát jen určité hashe. Na obrázku 10 jsou vyobrazeny červenou barvou. Jejich počet H je roven $H = log_2(L)$, kde L je počet listů.

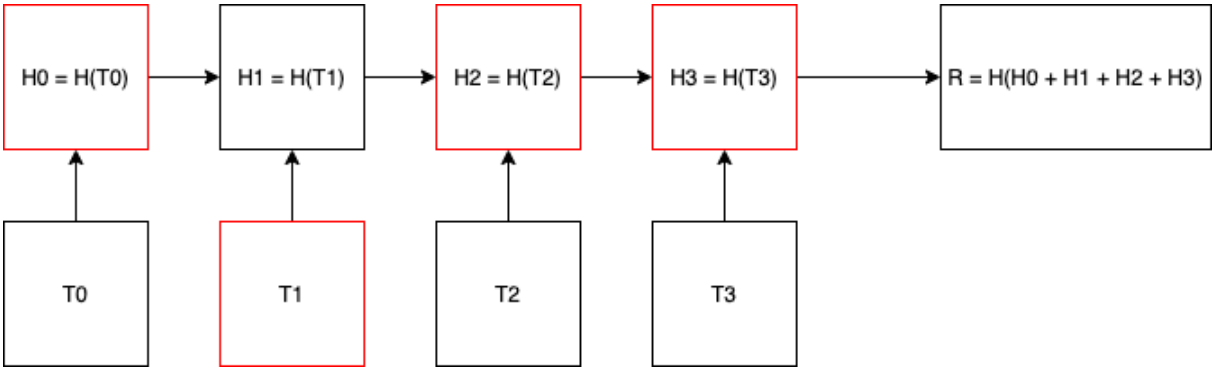


Obrázek 10 - Ověření integrity dat T1 v hashovém stromu

Pro ověření kořene stromu se čtyřmi listy je tedy zapotřebí znalosti jen dvou hashů H0 a H23. Data k ověření jsou T1 jejich hash tedy lze vypočítat.

$$H = log_2(4)$$

$$H = 2$$



Obrázek 11 - Ověření integrity dat T1 zapsaných v poli

Množství dat ve struktuře na obrázku 11 je stejné jako v předchozím příkladu, ale k jejich ověření je potřebná znalost alespoň tří hashů, jsou vyznačeny červeně. Platí tedy, že počet nutných hashů H je roven počtu listů $L - 1$. Hash dat T1 není potřebný, protože je možné jej dopočítat z této hodnoty. Pro ověření integrity dat je tedy zapotřebí znát je všechny.

V následující tabulce lze vidět, že se zvyšujícím se množstvím dat je tento rozdíl stále patrnější:

Tabulka 4 - Srovnání datových struktur

Pro data zapsaná v hashovém stromě		Pro data zapsaná v poli	
$H = log_2 (L)$		$H = L–1$	
Počet listů (L)	Počet nutných hashů (H)	Počet listů (L)	Počet nutných hashů (H)
2	1	2	1
4	2	4	3
8	3	8	7
16	4	16	15
1024	10	1024	1023

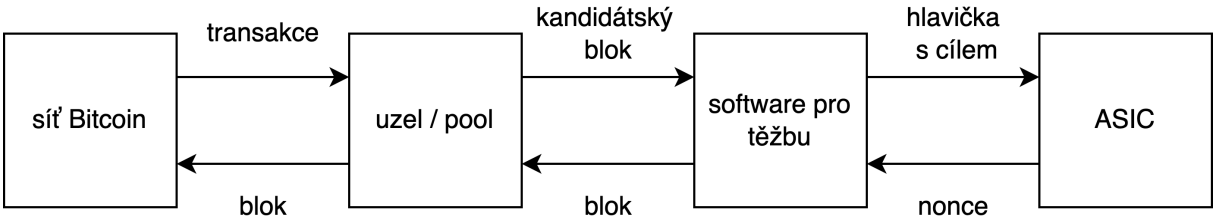
V blockchainu Bitcoinu, každý blok obsahuje v hlavičce otisk kořene Merkleova stromu, ve kterém listy tvoří transakce. Transakcí v bloku mohou být tisíce. Díky tomuto kořenu je pak možné ověřit, zda jsou v bloku opravdu zaneseny správné transakce. Pokud by se někdo pokusil změnit, některou z transakcí výsledný hash kořene by byl jiný, a blok tudíž neplatný.

2.2.5 Těžba

Proces, při kterém se do blockchainu zapisují nová data se nazývá těžba (mining). Těžaři hledají číslo (nonce), které přidají do hlavičky bloku tak, aby její otisk byl menší než cíl těžby.[19]

Počet operací, dosazení různých noncí, které může těžař provést, se udává v hashích za sekundu [H/s].

Těžař, získá data z uzlu a následně z nich sestaví kandidátský blok. Vybírá transakce, které mají nastavený nejvyšší transakční poplatek, který připadne těžaři.[20]



Obrázek 12 - Průběh těžby[21]

Cílem těžaře je vytěžit nový blok, za nějž obdrží odměnu ve formě mincotvorné (Coinbase) transakce.[22] Změnou posledních 4 bytů hlavičky bloku, tedy nonce se snaží docílit zmenšení otisku bloku pod cíl těžby. Každou sekundu je možné vyzkoušet 2^{32} čísel, to je maximální číslo, které je možné vepsat do 4 bytů. 4 byty = 32 bitů. $2^{32} = 4\ 294\ 967\ 296$. Po sekundě se však mění časové razítko v hlavičce bloku a je možné vyzkoušet stejný počet noncí znovu. Pokud má těžař výkon větší než 2^{32} H/s může paralelně pracovat na více verzích bloku s jinými transakcemi.

Množství bitcoinů v mincotvorné transakci se každých 210 000 bloků sníží na polovinu. Tento proces se označuje jako halving. Konečný počet halvingů je 32. Počáteční odměna v Coinbase transakci byla 50 bitcoinů. Konečný počet bitcoinů je tedy možné vypočítat následovně:

$$\sum_{x=0}^{32} 210000 \frac{50}{2^x} \approx 20999999.9769^{11} \approx 21000000$$

Konečný počet bitcoinů tedy nikdy nedosáhne hranice 21 milionů, jen se jí velmi přiblíží.

[11] Tento výsledek se liší od skutečného matematického výsledku, jelikož musíme provést celočíselné dělení v jednotkách satoshi. Bitcoin je totiž dělitelný jen na osm desetinných míst.

Protože pravděpodobnost nalezení nového bloku je velmi malá, menší těžaři se sdružují to tzv. poolů. Principem poolu je spojení více těžařů, kteří se v případě nálezu nonce o odměnu z bloku rozdělí.

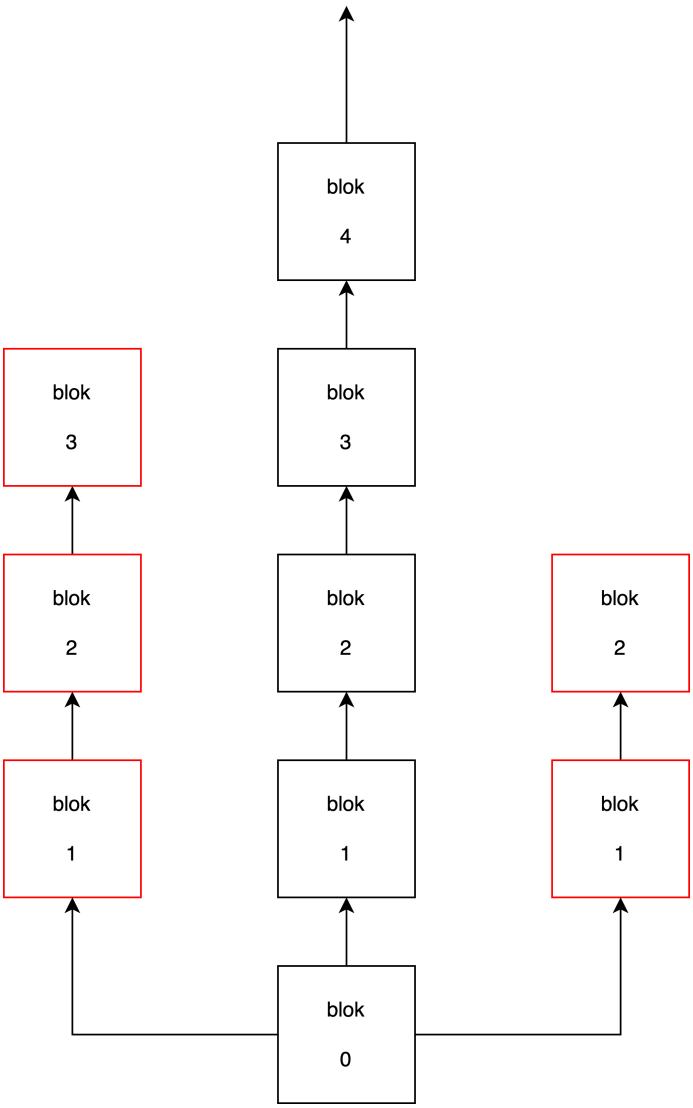
Výška bloku udává jeho pořadí v blockchainu. Čím je blok „hlouběji“, tím obtížnější je jeho změnění. Po šesti blocích už je změna prakticky nemožná, proto se pro důležité transakce doporučuje vyčkat na vytěžení dalších pěti bloků.[23] Tato praktika slouží zejména jako prevence proti 51% útoku.

2.2.5.1 Proof-of-Work

Důkaz o vykonané práci (Proof of Work, PoW) je konsensuální algoritmus. Určuje, který blockchain je ten pravý.

Těžaři využívají zákaznický integrovaný obvod (Application-specific integrated circuit, ASIC), tedy obvod určen jen pro jediný účel. V případě Bitcoinu se tedy jedná o obvod navržený k řešení kryptografické hashovací funkce SHA256.

Jev kdy se blockchain rozděluje se nazývá fork (vidlička) a je naprosto běžný. Vzniká v případě, že dva těžaři vytěží blok se stejnou výškou, ale s pozměněnými transakcemi, nebo totožný blok s jinou hlavičkou.



Obrázek 13 - Fork bitcoinového blockchainu

Na zápis každého bloku je potřeba vynaložit práci. Řetězec bloků, na který byla vynaložena největší práce, síť příjme za správný. Všechny kratší řetězce bloků zaniknou. I pokud získá jedna entita většinu výpočetního výkonu sítě, nemůže zapisovat neplatné transakce. Uzly v síti takový blok nepřímou. Existuje však riziko 51% útoku.

2.2.5.2 Složitost

Obtížnost se mění v závislost na rozdílu v čase vytěžených bloků, tak aby i přes vzrůstající hashrate sítě průměrná doba zápisu bloku byla stále 10 minut. Každých 2 016 bloků probíhá úprava složitosti (difficulty adjustment) – cíle těžby. Ideální časový rozdíl mezi 2 016 bloky je 1 209 600 sekund. Pokud je čas vyšší než ideální čas, obtížnost klesne, je-li tomu naopak, obtížnost stoupá.

V praxi se mění cíl těžby (target) v hlavičce bloku. Pro zvýšení obtížnosti, bude target požadovat větší množství nul ve výsledném otisku bloku. Pro snížení obtížnosti se zase maximální velikost otisku zvedá.

2.2.5.3 51% útok

51% útok (51% attack), je sice nepravděpodobný, ale možný. Pokud se jedné entitě podaří ovládnout velkou část výpočetního výkonu sítě, zvyšuje se pravděpodobnost, že dokáže vytvořit delší verzi blockchainu než celý zbytek sítě dohromady. V takovém případě by tato entita byla teoreticky schopna realizovat tento typ útoku. Jako prevence útoku slouží vyčkání na zatěžení transakce.

Realizace 51% útoku probíhá následovně:

1. Síť pracuje na veřejné verzi blockchainu
2. Útočník pracuje na „tajné verzi“ blockchainu
3. Útočník na nejdelší veřejné verzi blockchainu provede transakci, za zboží, nebo službu
4. Pokud se útočníkovi povede tajně vytvořit delší verzi blockchainu, zveřejní ji, jinak opakuje 2. krok
5. Útočník získal zboží/službu, ale bitcoiny zůstaly na jeho adrese

Útok je však vzhledem k velkému hashratu sítě velmi těžko proveditelný. Ke dni 14.7.2022 udával web https://www.crypto51.app cenu 51% útoku na bitcoinovou síť 608 627$ za hodinu, to dle tehdejšího kurzu odpovídá 14 697 916 Kč. Jedná se však o teoretickou cenu, web počítá s cenou daného výkonu na trhu, ale neexistuje žádný dodavatel tak velkého množství výkonu.[24] Navíc ani s většinou výkonu sítě, není zaručen úspěšný útok. Pravděpodobnost úspěšného útoku s 50 % hashratu sítě je rovna $\left(\frac{1}{2}\right)^6 = 0{,}015625 = 1{,}5625\%$.

Tabulka 5 - Pravděpodobnosti úspěšného 51% útoku

Podíl hashratu sítě v %	Pravděpodobnost vytěžení tří po sobě jdoucích bloků v %	Pravděpodobnost vytěžení šesti po sobě jdoucích bloků v %
1	0,0001	0,0000000001
10	0.1	0,0001
25	1,5625	0,0244140625
50	12,5	1,5625
75	42.1875	17,7978515625

2.3 Lightning network

Lightning network je druhá vrstva bitcoinové sítě stojící nad blockchainem. Umožňuje navýšení platební kapacity sítě a rychlosti jednotlivých plateb.**Chyba! Nenalezen zdroj odkazů.**

V této druhé vrstvě figurují lightningové uzly, které mezi sebou otevírají platební kanály. Propojováním těchto uzlů je pak možné přesouvat likviditu mezi jednotlivými platebními kanály. Na rozdíl od základní vrstvy se v lightningové síti na blockchain nezapisují jednotlivé transakce, ale jen konečné vypořádání platebního kanálu.

Prostředky se zde nepřesouvají pomocí transakcí na veřejný klíč (adresu), nýbrž pomocí tzv. lightning invoice (faktura). Dále existují protokoly, jako LNURL nebo lightning address[12], které velmi zpříjemňují uživatelský zážitek.

LNURLp (pay) umožňuje dynamicky generovat faktury. Server LNURL démonem přijímá http požadavky na generování invoice. Jednu LNURLp je tedy možné veřejně sdílet a každý na ní může opakovaně provádět platby s libovolnou výší. LNURLw (withdraw) Naopak požaduje v těle požadavku lightningovou fakturu k proplacení. Pokud je tedy LNURLw zveřejněna, kdokoliv může vybrat (opakovaně, či jednorázově) částku z předem stanoveného rozsahu.

Lightning address je jen další vrstvou abstrakce nad LNURL. Stejnou informaci, jakou nese LNURL, je možné zkomprimovat do kratšího tvaru, díky domluvenému standartu lightning address serveru. (více v kapitole 3.4.2)

[12] https://lightningaddress.com

3 PRAKTICKÁ ČÁST

Následující část popisuje funkcionalitu některých částí webových aplikací. Cílem této části je přiblížit čtenáři konkrétní způsoby interakce s abstraktními koncepty z teoretické části.

3.1 Použité technologie

Všechny následující aplikace jsou psány v HTML, CSS a JavaScriptu. Pro zajištění reaktivity stránky je použit framework Svelte. Pro spuštění JavaScriptu na straně serveru je využit runtime Node.js. Pro optimalizaci CSS je použit Framework Tailwind.

3.1.1 Node.js

Node.js (https://nodejs.org) je nejrozšířenější javascriptový runtime. Umožňuje spouštět JavaScript mimo prohlížeč. K jazyku navíc přidává ekvivalent standardních knihoven.

```
const fs = require('fs');
const file = fs.readFileSync('file.txt', 'utf8');
```

Ukázka kódu 1 - Příklad využití knihovny fs[13] pro načtení souboru file.txt

3.1.2 Svelte

Svelte (https://svelte.dev) je javascriptový framework, který svou rychlostí překonává konkurenci jako React, Angular nebo Vue. Svelte se v bulit-timu překládá do nativního javascriptu. Aplikace jsou tedy malé a rychlé.

```
<script>
  let name = "world";
</script>

<h1>Hello {name}!</h1>
```

Ukázka kódu 2 - Svelte umožňuje přistoupit k proměnným, funkcím a konstantám kdekoliv v HTML

3.1.3 Express

Express (https://expressjs.com) je rychlý a minimalistický backednový framework pro Node.js. Umožňuje vytvoření webového serveru, REST API nebo obecně všeho, co je na backendu

[13] File Systém (fs) - https://nodejs.org/api/fs.html

potřeba. V následujících aplikacích je použit pro přístup k databázi, získávání dat z API skrze tajný API klíč, uložený na serveru, a samotnému servingu[14] statických stránek.

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
    res.send('SOČ!');
})

app.listen(port, () => {
    console.log(`Ukázková aplikace běží na portu ${port}`);
})
```

Ukázka kódu 3 - Příklad serveru, který na portu 3000 pošle odpověď „SOČ!“

3.1.4 Tailwind CSS

Tailwind CSS (https://tailwindcss.com) je CSS framework. Jeho účelem je eliminovat redundantní CSS. Framework obsahuje CSS soubor se všemi myslitelnými třídami. Následně se tyto třídy přiřazují komponentům. V build-timu se pak všechny nepotřebné třídy smažou a zůstává jen optimální CSS.

```
<div class="bg-slate-800 w-20 h-72 grid grid-rows-1 items-center">
  <h1 class="text-white text-5xl text-center">SOČ!</h1>
</div>
```

Ukázka kódu 4 - Příklad použití tailwindu

[14] Vysvětlení termínu - https://expressjs.com/en/starter/static-files.html

Obrázek 14 – Výstup z ukázky kódu 4

3.1.5 PWA

PWA (https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps) – Progessive Web Apps jsou webové aplikace, které je možné instalovat podobně jako ty nativní. Při správné konfiguraci jsou schopny fungovat off-line.

3.1.6 Umbrel

Umbrel (https://umbrel.com) je operační systém pro domácí servery. Tento systém slouží zejména pro snadné spouštění kontejnerů. Zároveň poskytuje velmi přívětivé grafické rozhraní a snadnou vzdálenou správu přes Tor.

Pro testování aplikací bylo žádoucí sestrojení vlastního lightningového uzlu. K tomuto účelu byl použit jednodeskový počítač Raspberry Pi 4[15] s již zmíněným operačním systémem Umbrel. Pro uložení kopie blockchainu bylo použito 1TB Disku od firmy SanDisk. Samotný operační systém je pak uložen na 32 GB microSD kartě, rovněž od firmy SanDisk.

Obrázek 15 - Komponenty pro sestavení lightningového uzlu

[15] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

3.2 Node & market data Dashboard

Aplikace zobrazuje data z libovolného bitcoinového/lightningového uzlu. Defaultně je aplikace určená k použití s Umbrel nodem. Data se zobrazí v dlaždicovém uspořádání.

Jednotlivé dlaždice jsou svelte komponenty. Se základní znalostí HTML je možné změnit jejich uspořádání dle vlastních potřeb. Stejně tak je možné přidávat vlastní komponenty.

3.2.1 Získávání dat

Jak bylo uvedeno v teoretické části, většina uzlů není dostupná z clearnetu, ale za pomocí protokolu Tor (případně alternativ). Aplikace proto musí umět poslat požadavky i přes Tor.

```
const fetchTor = async (url) => {
    try {
        return await (await fetch(url, {
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json',
            },
            agent: new SocksProxyAgent('socks5h://127.0.0.1:9050'),
        })).json()
    } catch (e) {
        return { error: e }
    }
}
```

Ukázka kódu 5 - Kód pro „bezpečné “ získávání dat přes Tor

Požadavky jsou směrovány přes socks5 proxy. Protokol socks5 má defaultní port 9050. Pro správnou funkčnost je tedy nutné spustit tento proxy server lokálně (například za použití Dockeru[16]) nebo vzdáleně, přičemž je nutné upravit konfiguraci aplikace. Pro posílání požadavků přes proxy je použita knihovna socks-proxy-agent[17]

[16] Docker - https://www.docker.com

[17] Knihovna - https://www.npmjs.com/package/socks-proxy-agent

```
app.get("/api/fees", async (req, res) => {
    res.json({
        fees: await
fetchTor("http://kx72vljgxk2ao7wqlrjghix6jecbhhb6rsvbscvbqilxekyjmmham2ad.onion/api
/mempool/fees"),
        rewards: await
fetchTor("http://nqieh33bhvzhm35l4qe4ifzbrbpxk4szwlu42wonqycvmqurlzptigyd.onion/api
/v1/mining/reward-stats/144"),
    });
});
```

Ukázka kódu 6 - Zaslání požadavku na server za Torem

Po načtení dat z uzlu přes Tor je express server pošle přes http protokol na frontend, kde budou vykreslena.

3.2.2 Vykreslení dat

Data se zobrazují na stránce jako čistý text, nebo formou grafu. Grafy jsou vykreslovány za pomocí knihovny *Chart.js*[18].

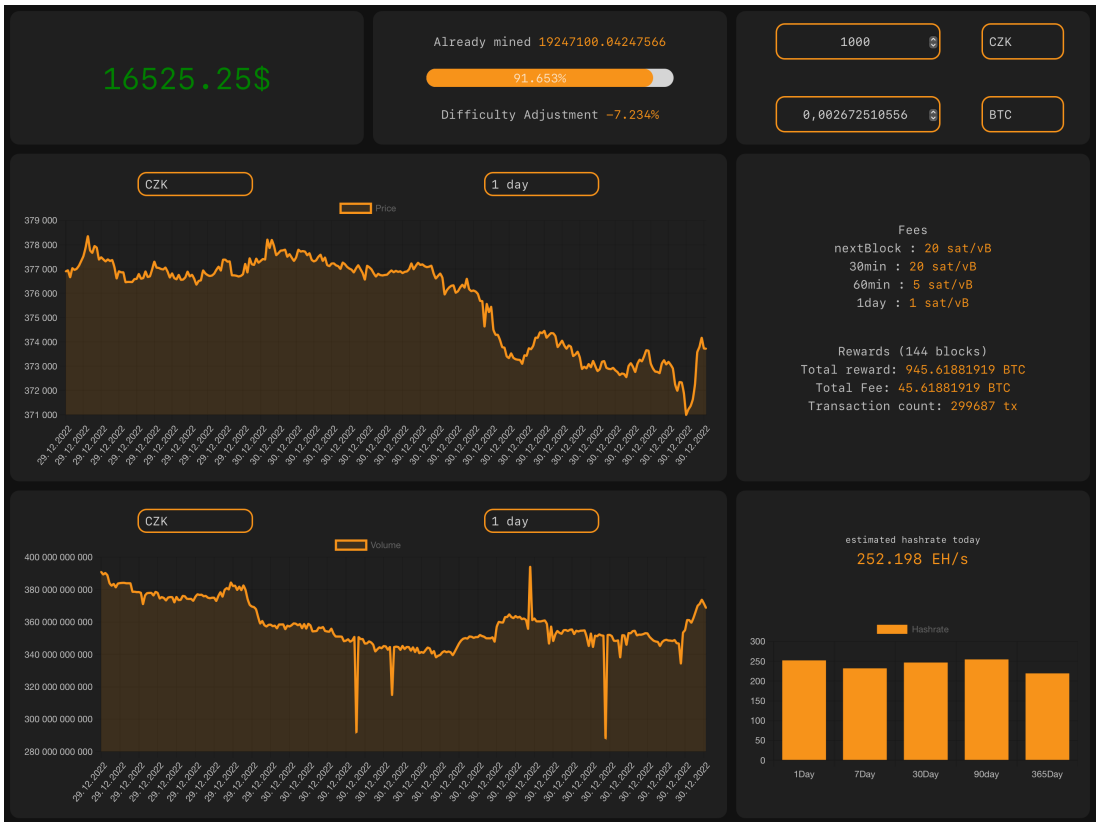
```
const drawMinersChart = (miners) => {
    const ctx = document.getElementById('minersChart').getContext('2d');
    new Chart(ctx, {
        type: 'doughnut',
        data: {
            labels: Object.keys(miners.miners),
            datasets: [{
                data: Object.values(miners.miners).map(x => x.blocks.length),
                backgroundColor: [
                    '#f7931a',
                ],
            }
            ]
        },
    });
}
```

Ukázka kódu 7 - Příklad vykreslení grafu

[18] Chart.js - https://www.chartjs.org

3.2.3 Jednotlivé komponenty

Krátké ukázky kódu jednotlivých komponent slouží jako představa o praktickém fungování komponentů, a zejména jako ukázka využití konkrétních technologií.



Obrázek 16 - Vzhled nástěnky

3.2.3.1 Cena

Zřejmě nejjednodušší komponent je jednoduché zobrazení aktuální ceny Bitcoinu. Stejně jako graf ceny a graf objemu nenačítá data z uzlu, který tato data mít nemůže, ale z veřejných API burz. Komponent využívá technologii web WebSocket. Cena je díky tomu vždy aktuální.

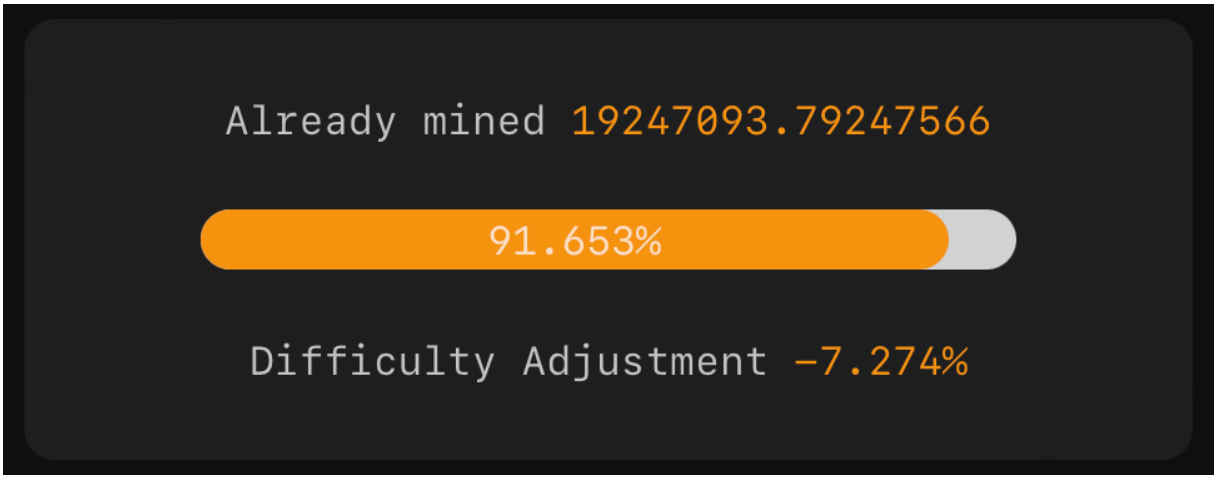
```
const socket = new WebSocket(config.priceWS.url);
let price = 0;
socket.onmessage = (event) => {
    data = Number((JSON.parse(event.data)).k.c).toFixed(2);
    document.getElementById("price").style.color = data >= price ? "green" : "red";
    price = data;
}
```

Ukázka kódu 8 - Využití websocket pro získání aktuální ceny Bitcoinu

URL defaultního websocket serveru je *wss://stream.binance.com:9443/ws/btcbusd*. Jelikož nepotřebuje získávat data přes Tor, celý kód tohoto komponentu je klientský.

3.2.3.2 Obtížnost a množství vytěžených bitcoinů

Tento komponent zobrazuje množství již vytěžených bitcoinů a jejich poměr ke konečnému množství bitcoinů. a také předpokládanou úpravu složitosti na další období.



Obrázek 17 - Vzhled zobrazení již vytěžených bitcoinů

3.2.3.3 Převodník

Převodník umožňuje konverzi měn zvolených v konfiguračním souboru (defaultně CZK, USD, EUR, PLN, GBP) do Bitcoinu nebo jeho řádově nižších jednotek (SAT, mSAT)

```
<input type="number" min="0" bind:value={before}
    on:input={()=>{if(after !== before / rates[currency]){after =
parseFloat((before / rates[currency] * Number(to)).toFixed(12))}}}>
```

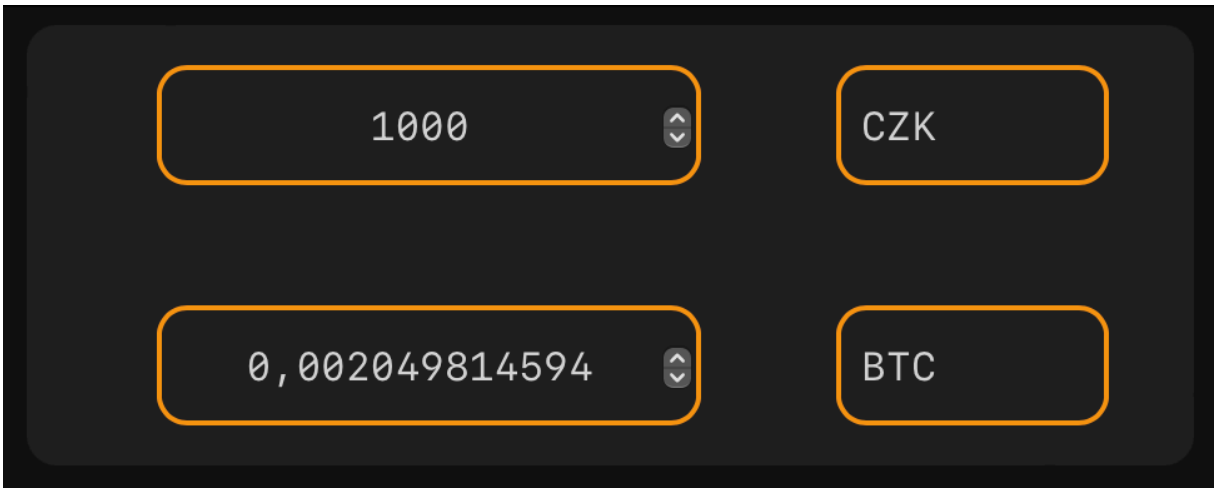
Ukázka kódu 9 - Využití reaktivity svetle pro převodník

Díky frameworku Svelte je možné velmi jednoduše vytvořit oboustrannou reaktivitu mezi vstupy. Bind:value automaticky změní hodnotu výstupního pole při změně hodnoty vstupního pole. *On:input* je event listener, kombinací těchto dvou metod na obou vstupech je dosaženo reaktivity s velmi krátkým kódem. Vzhled je pak řešen výhradně v atributu class, díky Tailwind CSS.

Možnost výběru měn, které mohou být v konfiguračním souboru měněny, musí být dynamicky aktualizovány. Je k tomu použit Svelte cyklus *each*, jehož syntaxe ve Svelte vypadá následovně:

```
<select bind:value={currency} on:change={()=>{after = parseFloat((before /
rates[currency] * Number(to)).toFixed(16))}}>
    {#each config.convertor.currencies as option}
        <option value={option}>{option.toUpperCase()}</option>
    {/each}
</select>
```

Ukázka kódu 10 - Využití reaktivity při změně jednotky



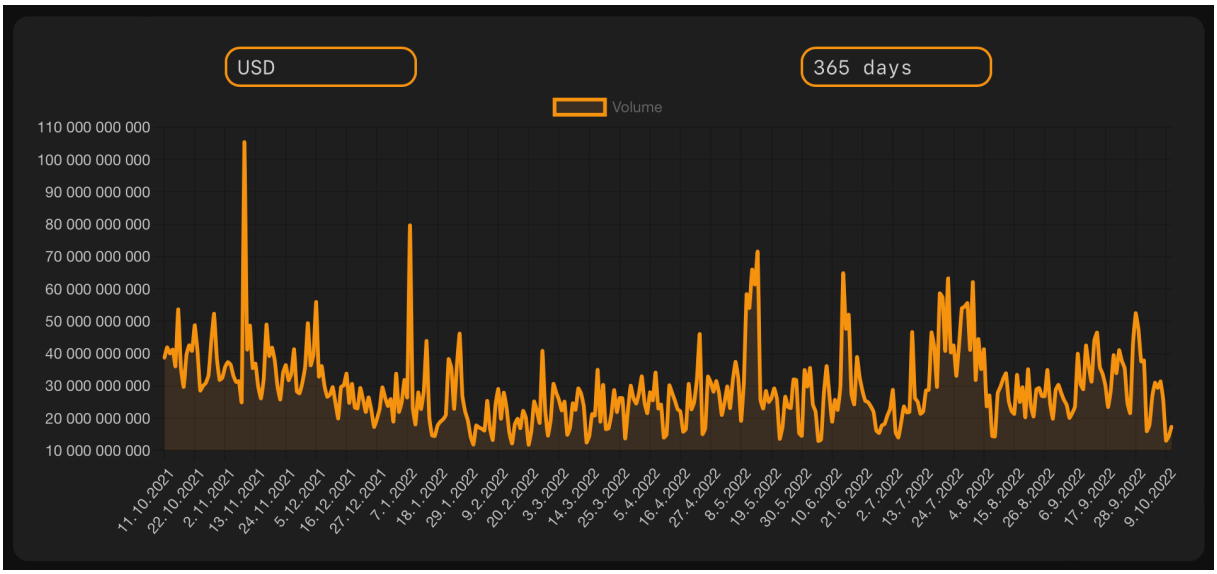
Obrázek 18 - Vzhled převodníku

3.2.3.4 Graf ceny

Zobrazení ceny ve vybrané měně pro vybrané časové období. Měny se dají upravit v konfiguračním souboru. Vykreslen pomocí knihovny *Chart.js*.

Obrázek 19 - Vzhled grafu ceny

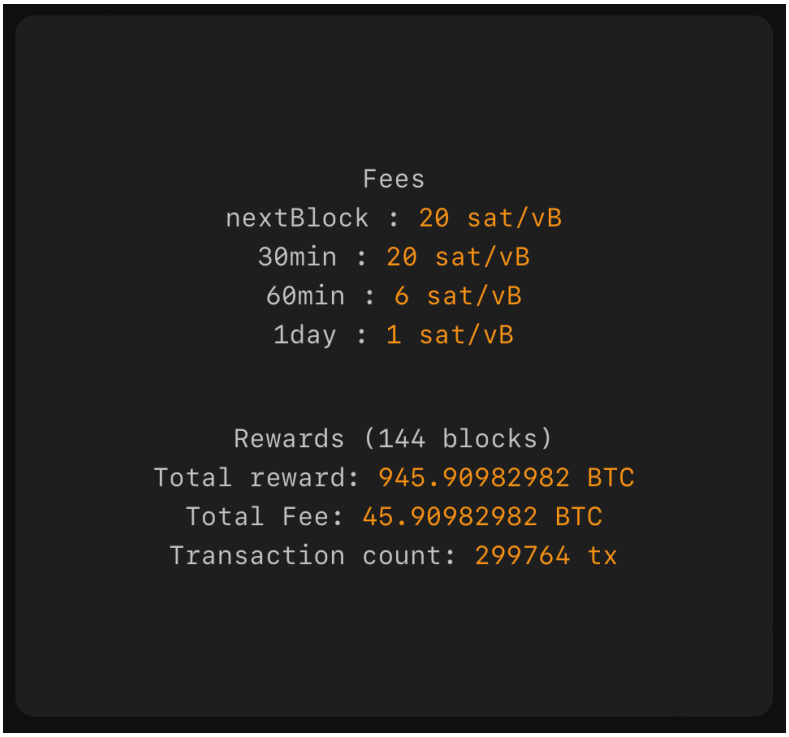
3.2.3.5 Graf objemu



Obrázek 20 - Vzhled grafu objemu

3.2.3.6 Poplatky a odměny

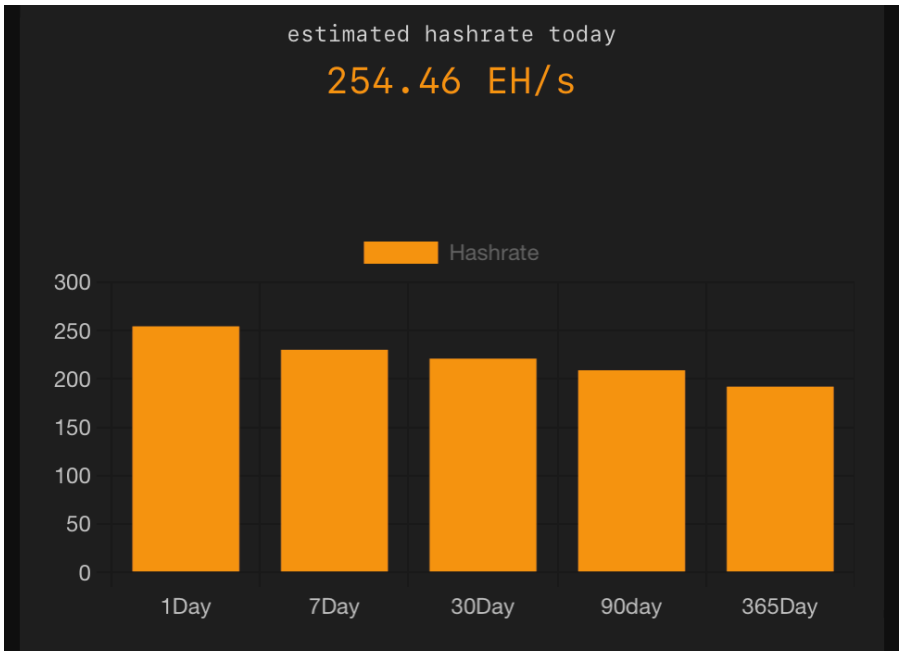
Tento komponent zobrazuje, jaký poplatek je třeba nastavit transakci, aby byla zapsána do blockchainu v určitém časovém intervale. Druhá část pak zobrazuje součet odměn za posledních 144 bloků a počet transakcí v těchto blocích.



Obrázek 21 - Vzhled zobrazení poplatků

3.2.3.7 Hashrate

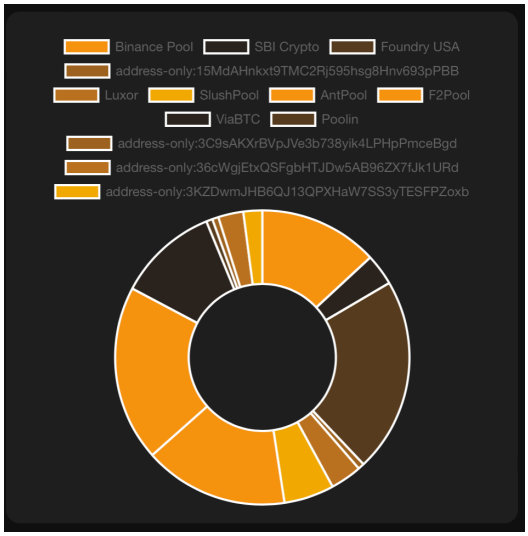
Hashrate komponent zobrazuje sloupcový graf s denním, týdenním, měsíčním a ročním průměrným hashratem.



Obrázek 22 - Vzhled grafu hashratu

3.2.3.8 Distribuce vytěžených bloků

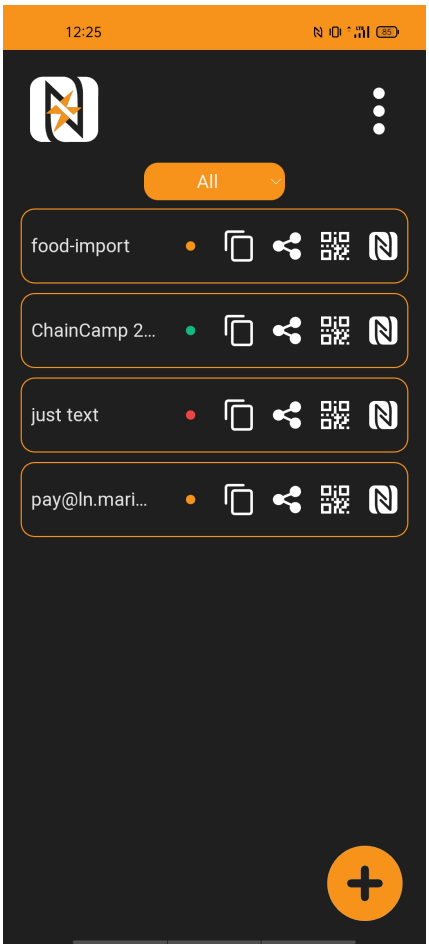
Koláčový graf distribuce vyobrazuje množství bloků vytěžených určitým těžařem. Vzhledem k důvodům popsaným v kapitole a by měla být distribuce vytěžených bloků rovnoměrná, pro vyloučení 51% útoku. Na grafu se zobrazí i adresa tzv. nezávislých těžařů, těžařů bez poolu.



Obrázek 23 - Vzhled grafu distribuce bloků vzhledem k poolům

3.3 lnNote – aplikace pro správu lnurl

Aplikace umožňuje vytváření a správu LNURL a lightningových adres. Hlavní funkcí je zápis dat na NFC tagy. Může být tedy využita například pro vytváření lightningových platebních karet.



Obrázek 24 - Hlavní stránka aplikace

Data jsou organizována ve formě jednotlivých záznamů, přičemž jsou roztříděny podle typu. Barevné indikátory u každého záznamu, značí typ tohoto záznamu.

1. LNURLw vygenerované v aplikaci (zelená) – tento typ záznamu může být doplněn prostředky prostřednictvím lightningové transakce.

2. LNURLp nebo lightningové adresa (zlatá) – aplikace umí dekódovat tento typ záznamu a následně získat a zobrazit data, například o maximální délce popisu.

3. Textové záznamy (červená) – aplikace nemůže o těchto záznamech načíst žádná data, ale stále je možný zápis na NFC nebo jiná forma exportu.

Hlavní cílem aplikace je tedy snadná organizace více záznamů, a zejména jejich export. Avšak aplikace obsahuje více nástrojů, například pro vytváření LNURL.

3.3.1 NFC rozhraní

Interakce s NFC jsou prováděny za pomocí Web NFC API[19], které umožňuje používání NDEF protokolu.

```
writeTag = async () => {
    if ("NDEFReader" in window) {
        ndef = new window.NDEFReader();
        try {
            await ndef.write("text");
        } catch (err) {
            alert(err);
        }
    }
};
```

Ukázka kódu 11 - Příklad kódu pro zapsání textových dat na NFC tag

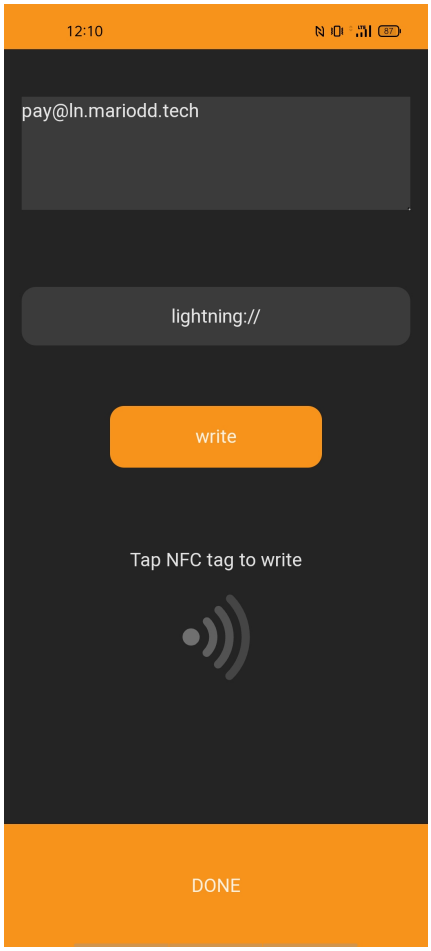
```
let newRec;
read = async () => {
    newRec = "";
    if ("NDEFReader" in window) {
        const ndef = new window.NDEFReader();
        try {
            await ndef.scan();
            ndef.onreading = (event) => {
                const decoder = new TextDecoder();
                let tmp = "";
                for (const record of event.message.records) {
                    tmp = decoder.decode(record.data);
                }
                newRec = tmp;
            }
        } catch (err) {
            alert(err);
        }
    }
}
```

Ukázka kódu 12 - Příklad pro čtení dat z NFC tagu

[19] https://developer.mozilla.org/en-US/docs/Web/API/Web_NFC_API

V ukázce kódu 12 se do proměnné *newRec* vždy ukládá poslední nalezená hodnota. Čtení se tedy neukončí, dokud si uživatel nepřeje.

V aplikace je možné všechny záznamy zapsat na NFC tag, včetně prefixu pro daný protokol. Uživatel tedy může zapsat svoji lightningovou adresu s prefixem *lightning://* na tag a po načtení se otevře lightningová peněženka.



Obrázek 25 - Rozhraní pro zápis dat s prefixem

3.3.2 QR

QR kódy jsou další formou exportu a importu záznamů. Jsou využívány taky pro zobrazení faktury sloužící k zaslání prostředků k naplnění LNURLw. Čtení je zajištěno knihovnou *Html5Qrcode*[20]. Generování QR kódů je pak řešeno knihovnou *qrcode*[21].

```
let newRec;
const read = () => {
    newRec = "";
    html5QrCode = new Html5Qrcode("reader");

    const qrCodeSuccessCallback = (decodedText) => {
        newRec = decodedText;
    };

    const config = { fps: 30, qrbox: { width: 200, height: 200 } };

    html5QrCode.start(
        { facingMode: "environment" },
        config,
        qrCodeSuccessCallback
    );
};
```

Ukázka kódu 13 - Čtení QR kódů

```
const gen = () => {
    QRCode.toCanvas(
        document.getElementById("canvas"),
        "text",
        {
            width: 250,
            errorCorrectionLevel: "H",
        },
        (err) => {
            if (err) { alert(error) };
        }
    );
}
```

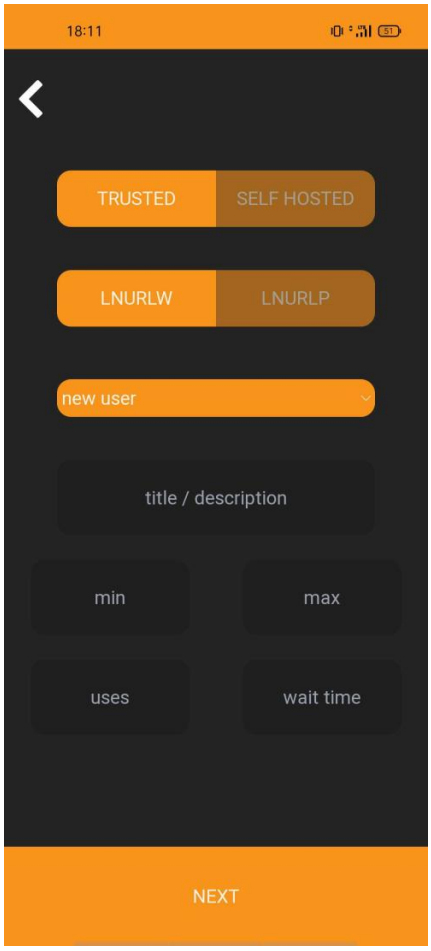
Ukázka kódu 14 - Generování QR kódu

[20] https://github.com/mebjas/html5-qrcode

[21] https://github.com/soldair/node-qrcode

3.3.3 LNURL

LNURL p i w je možné vygenerovat přímo v aplikaci, prostřednictvím LNbits[22] API. Pro vytvoření LNURL tedy uživatel vůbec nemusí opustit aplikaci. Je možné vytvořit LNURL přes vlastní LNbits server (potřeba zadat API klíč) nebo na defaultním serveru.

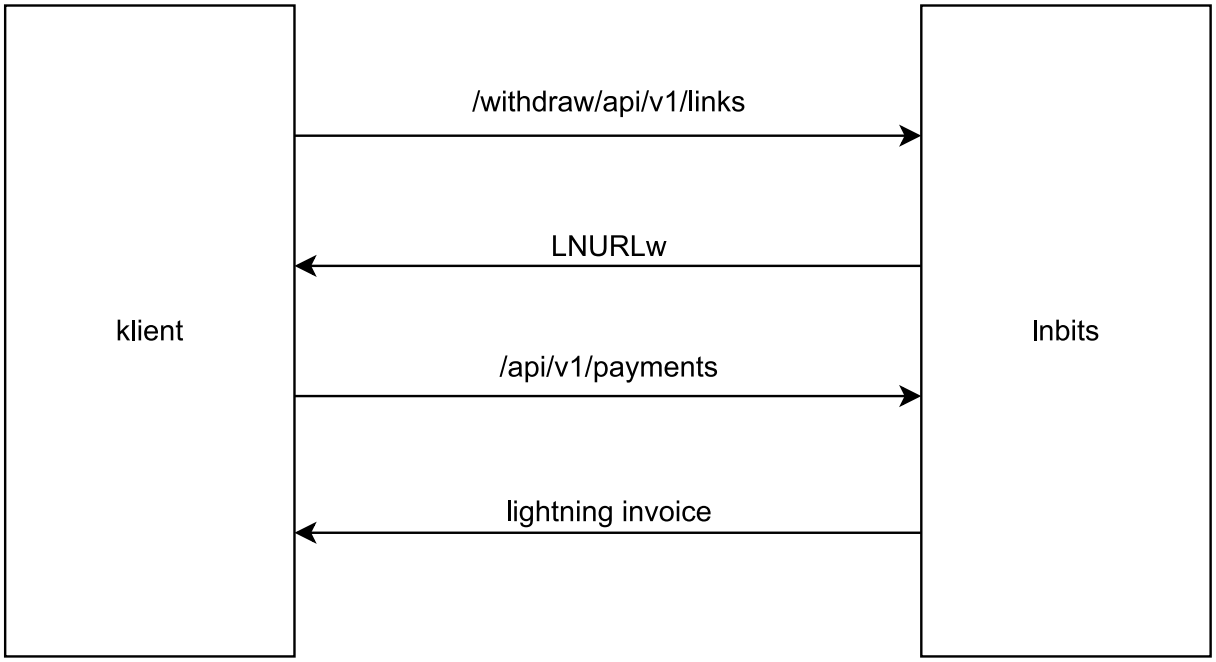


Obrázek 26 - Rozhraní pro vytvoření LNURL

3.3.3.1 Dobíjení LNURLw

LNURL typu w je možné dobíjet pomocí lightning invoice, pokud je generovaná přímo v aplikaci. Aplikace pošle API požadavek na defaultní nebo uživatelem zvolený server s LNbits. Takovýto server, kromě endpointu pro vytvoření LNURL, disponuje také endpointem pro vytvoření lightning invoice.

[22] https://lnbits.com – Stránky projektu



Obrázek 27 - Generování LNURLw a lightning invoice pro její naplnění

Výše, na kterou je vystavena lightning faktura, je počet použití krát maximální výběr plus fixní poplatek, avšak jedná se o doporučenou částku. Uživatel si částku může zvolit sám.

Obrázek 28 - Ukázka stránky vyzývající k naplnění LNURLw

3.3.4 LNURL Dekodér

Aplikace disponuje dekodérem LNURL. LNURL je jen URL zakódována pomocí bech32 s prefixem *lnurl://*.

```
const decode = (lnurl) => {
    const d = bech32.decode(lnurl, 1500)
    const b = bech32.fromWords(d.words)
    return Buffer.from(b).toString()
}
```

Ukázka kódu 15 - Dekódování LNURL

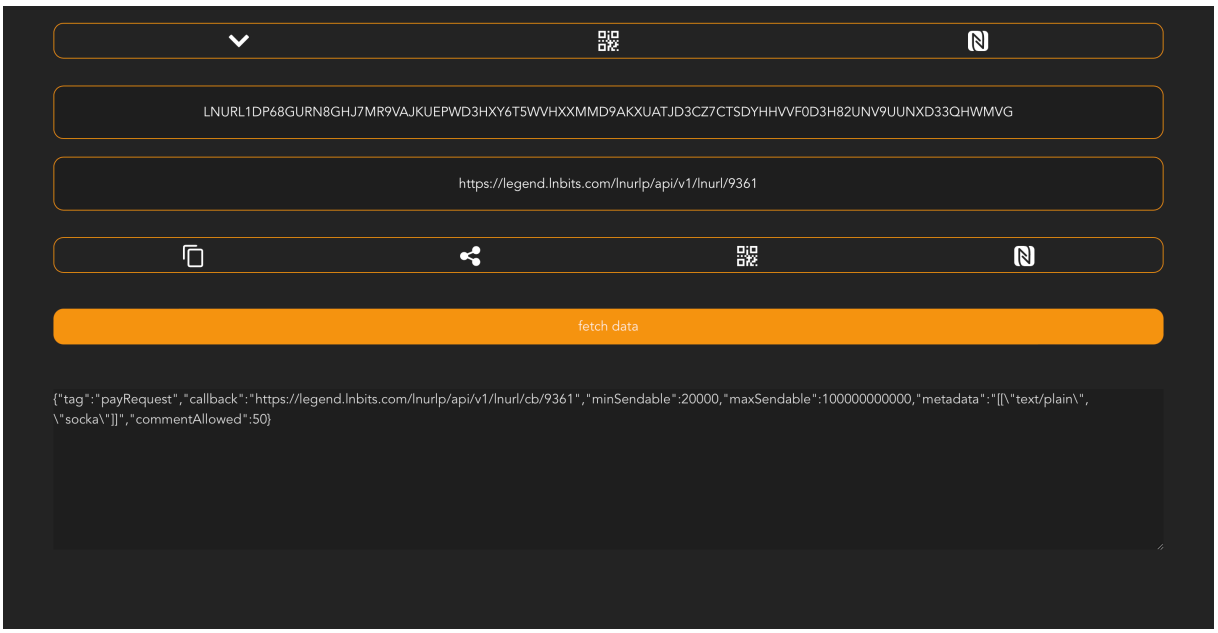
```
const encode = (url) => {
    const words = bech32.toWords(Buffer.from(url, 'utf8'))
    return bech32.encode('lnurl', words, 1500)
}
```

Ukázka kódu 16 - Enkódování LNURL

Dále je zde podpora pro lightning adresy.

```
const decodeLightningAddress = (address) => {
    return `https://${address.split("@")[1]}/.well-
known/lnurlp/${address.split("@")[0]}`;
}
```

Ukázka kódu 17 - Dekódování Lightningové adresy

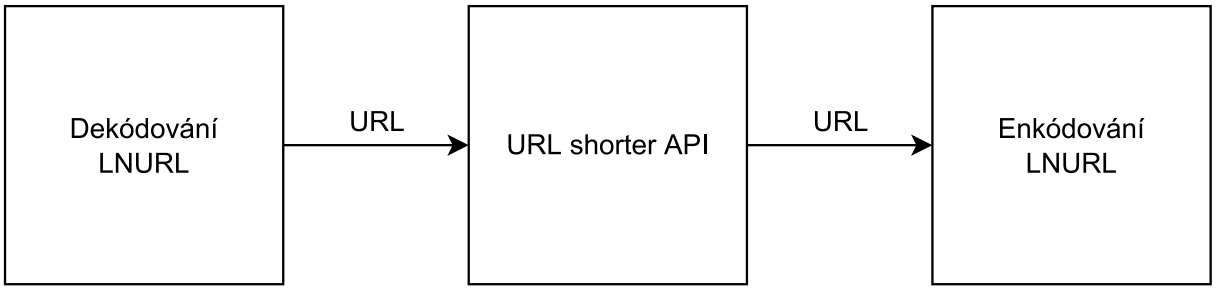


Obrázek 29 - Ukázka dekódování LNURL

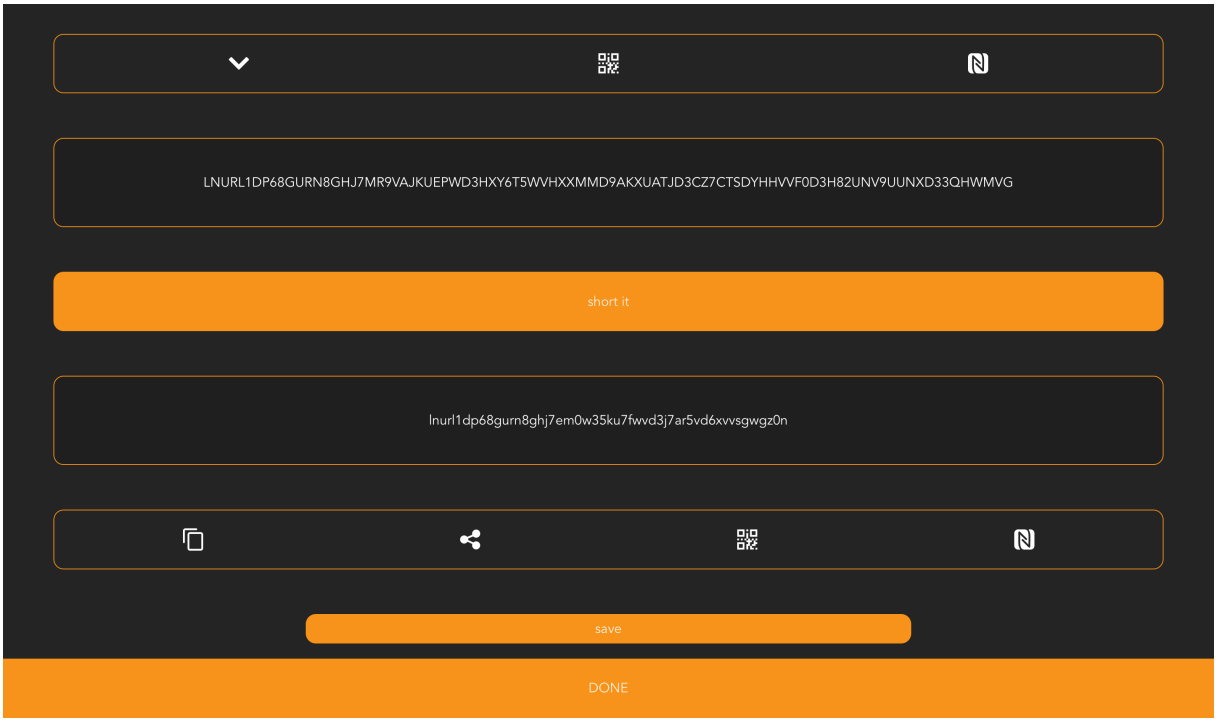
Každý záznam ve formě LNURL je tedy možné dekódovat a pokud se jedná o validní URL, aplikace z ní umožňuje zobrazit data ve formě JSONu.

3.3.5 LNURL zkracovač

Existují LNURL, které jsou příliš dlouhé, aby byly zapsány do některých NFC tagů. Do aplikace je proto integrován zkracovač těchto LNURL. Principem je dekódování LNURL, odeslání dekódovaná URL přes API k vybranému poskytovateli zkracovací služby, zkrácená URL se opět zakóduje do LNURL a uloží jako záznam.



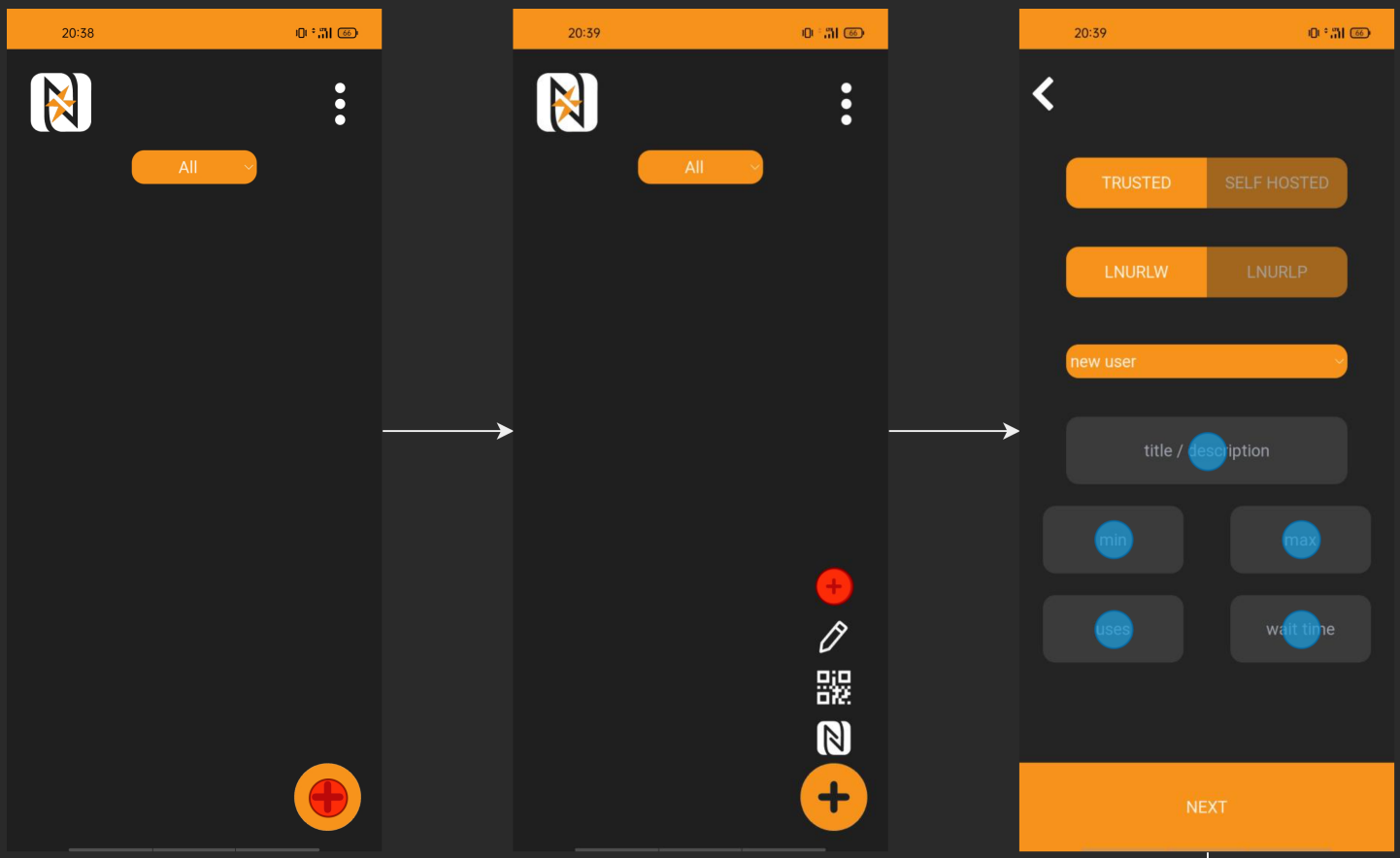
Obrázek 30 - Princip zkracování LNURL

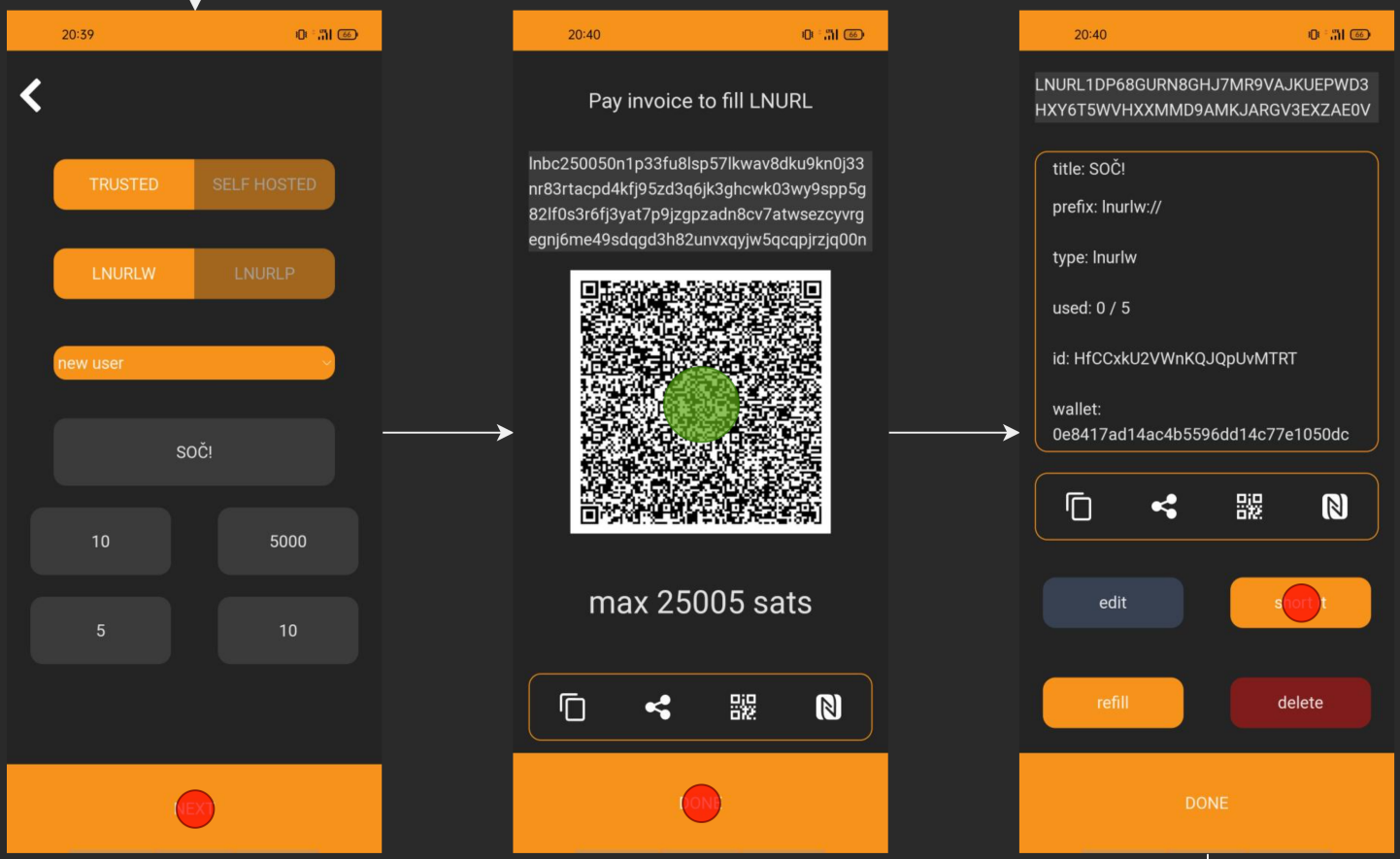


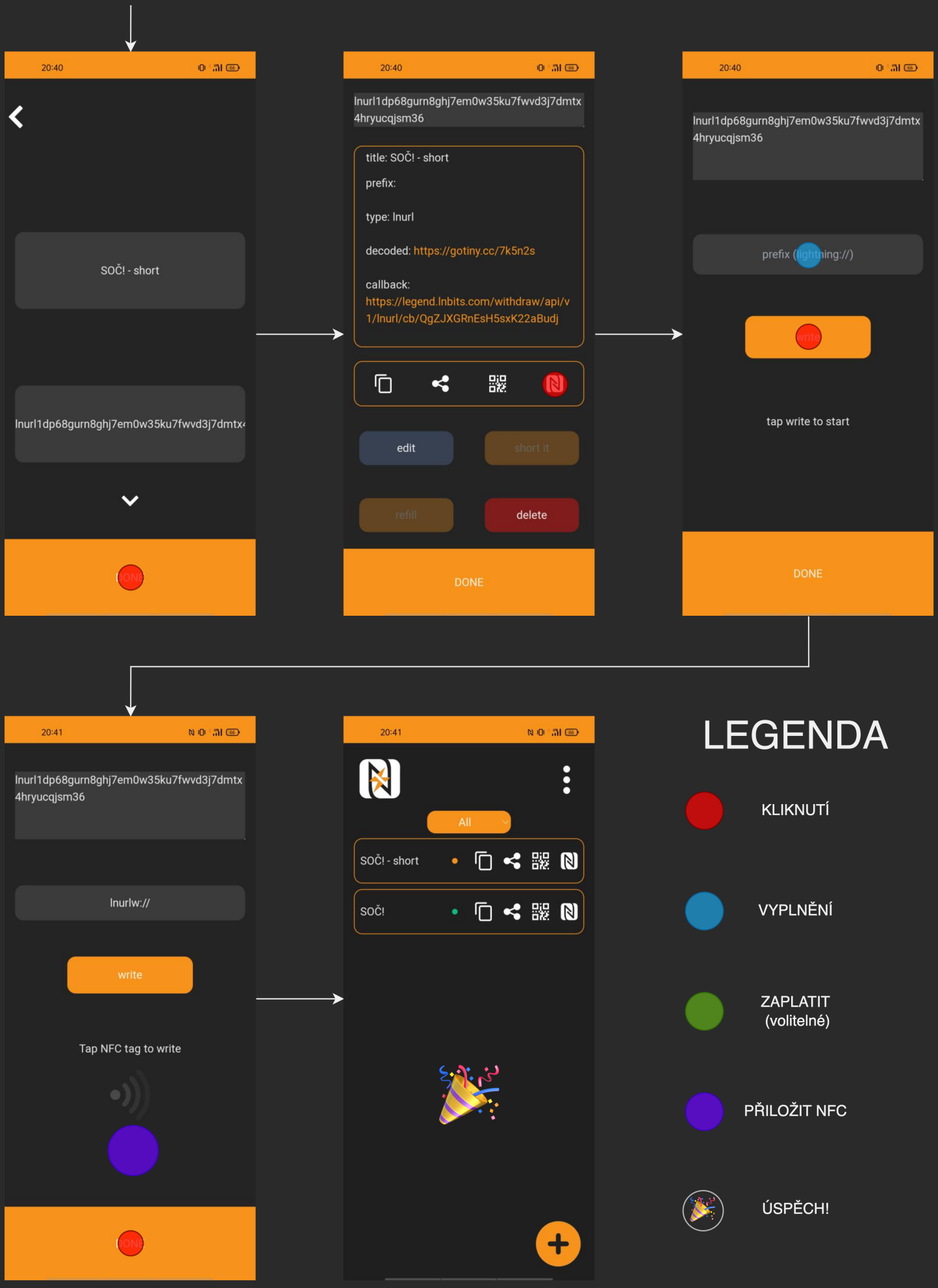
Obrázek 31 - Příklad zkrácení LNURL

3.3.6 Návod na vytvoření NFC platební karty

Následují strany ukazují návod, jak vytvořit funkční lightningovou NFC platební kartu bez opuštění aplikace. Tento proces ukazuje mnoho funkcí aplikace v praxi.


20:38
All
20:39
All
20:39
TRUSTED
SELF HOSTED
LNURLW
LNURLP
new user
title / description
min
max
uses
wait time
NEXT

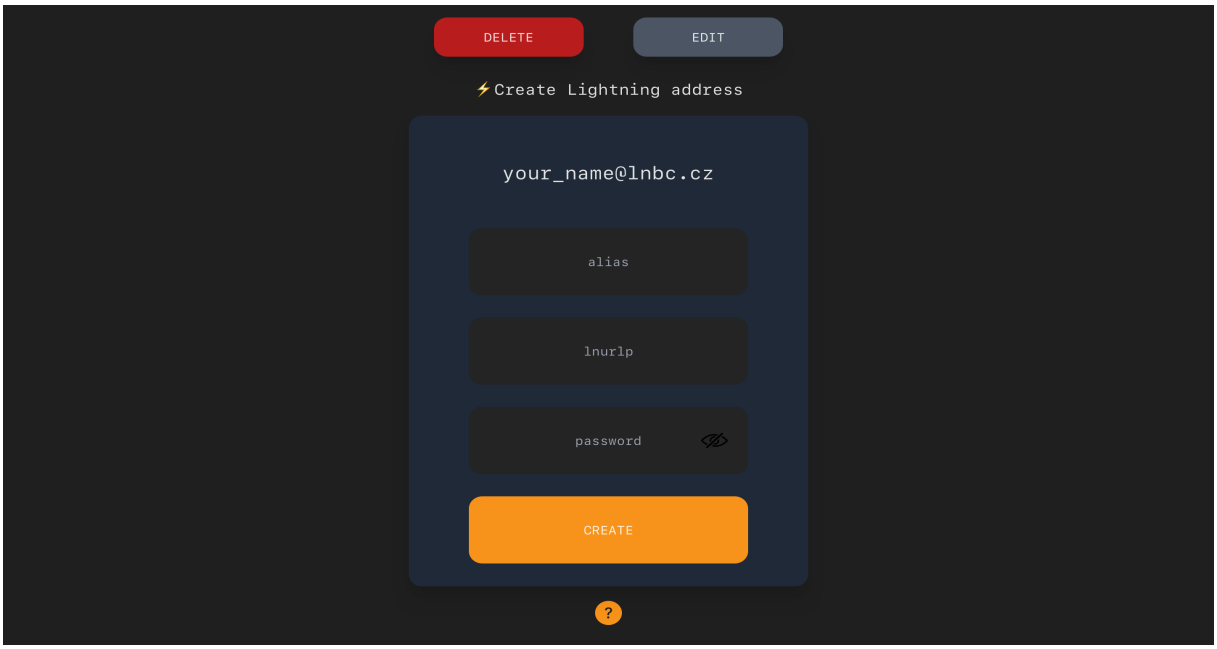

20:39
TRUSTED
SELF HOSTED
LNURLW
LNURLP
new user
SOČ!
10
5000
5
10
20:40
Pay invoice to fill LNURL
lnbc250050n1p33fu8lsp57lkwav8dku9kn0j33
nr83rtacpd4kfj95zd3q6jk3ghcwk03wy9spp5g
82lf0s3r6fj3yat7p9jzgpzadn8cv7atwsezcyvrg
egnj6me49sdqgd3h82unvxqyjw5qcqpjrzjq00n
max 25005 sats
20:40
LNURL1DP68GURN8GHJ7MR9VAJKUEPWD3
HXY6T5WVHXXMMD9AMKJARGV3EXZAE0V
title: SOČ!
prefix: lnurlw://
type: lnurlw
used: 0 / 5
id: HfCCxkU2VWnKQJQpUvMTRT
wallet:
0e8417ad14ac4b5596dd14c77e1050dc
edit
refill
delete
DONE


20:40
SOČ! - short
lnurl1dp68gurn8ghj7em0w35ku7fwvd3j7dmtx
DONE
20:40
lnurl1dp68gurn8ghj7em0w35ku7fwvd3j7dmtx
4hryucqjsm36
title: SOČ! - short
prefix:
type: lnurl
decoded: https://gotiny.cc/7k5n2s
callback:
https://legend.lnbits.com/withdraw/api/v
1/lnurl/cb/QgZJXGRnEsH5sxK22aBudj
edit
short it
refill
delete
DONE
20:40
lnurl1dp68gurn8ghj7em0w35ku7fwvd3j7dmtx
4hryucqjsm36
prefix (lightning://)
write
tap write to start
DONE
20:41
lnurl1dp68gurn8ghj7em0w35ku7fwvd3j7dmtx
4hryucqjsm36
lnurlw://
write
Tap NFC tag to write
DONE
20:41
All
SOČ! - short
SOČ!
LEGENDA
KLIKNUTÍ
VYPLNĚNÍ
ZAPLATIT
(volitelné)
PŘILOŽIT NFC
ÚSPĚCH!

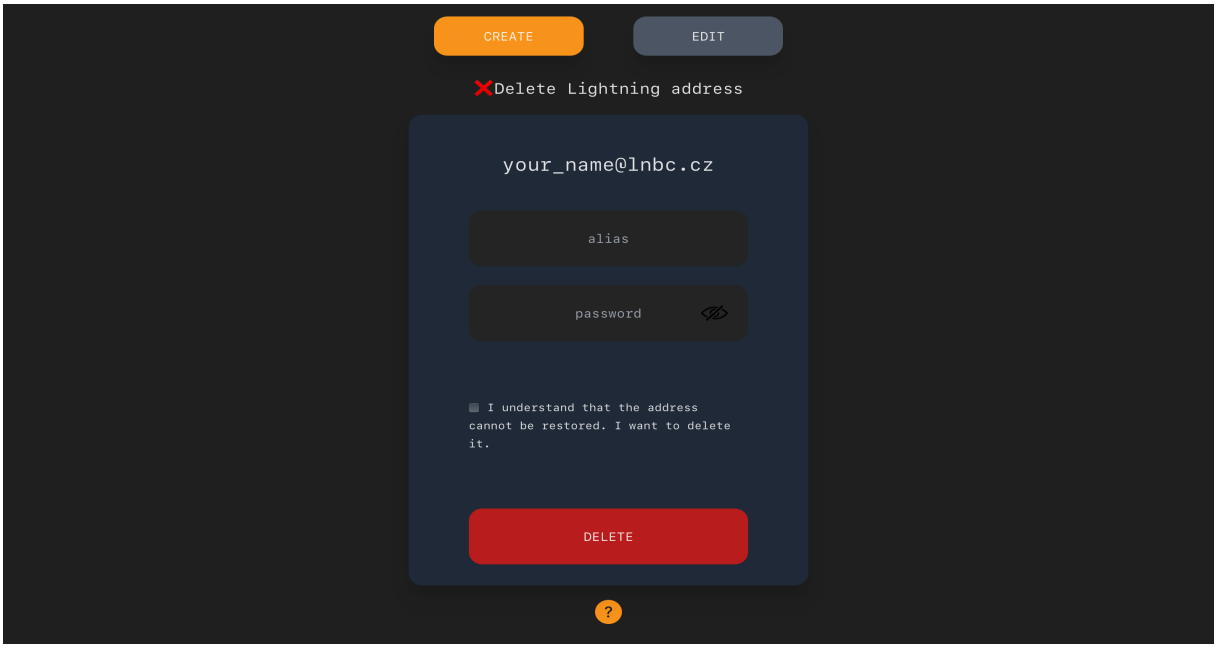
3.4 lnAddresses

Aplikace lnAddresses je webová aplikace, která umožňuje uživateli vytvořit lightningovou adresu pouze na základě LNURLp. Na rozdíl od oficiálního lightning address serveru aplikace nevyžaduje žádnou interakci uživatele s konkrétní peněženkou.

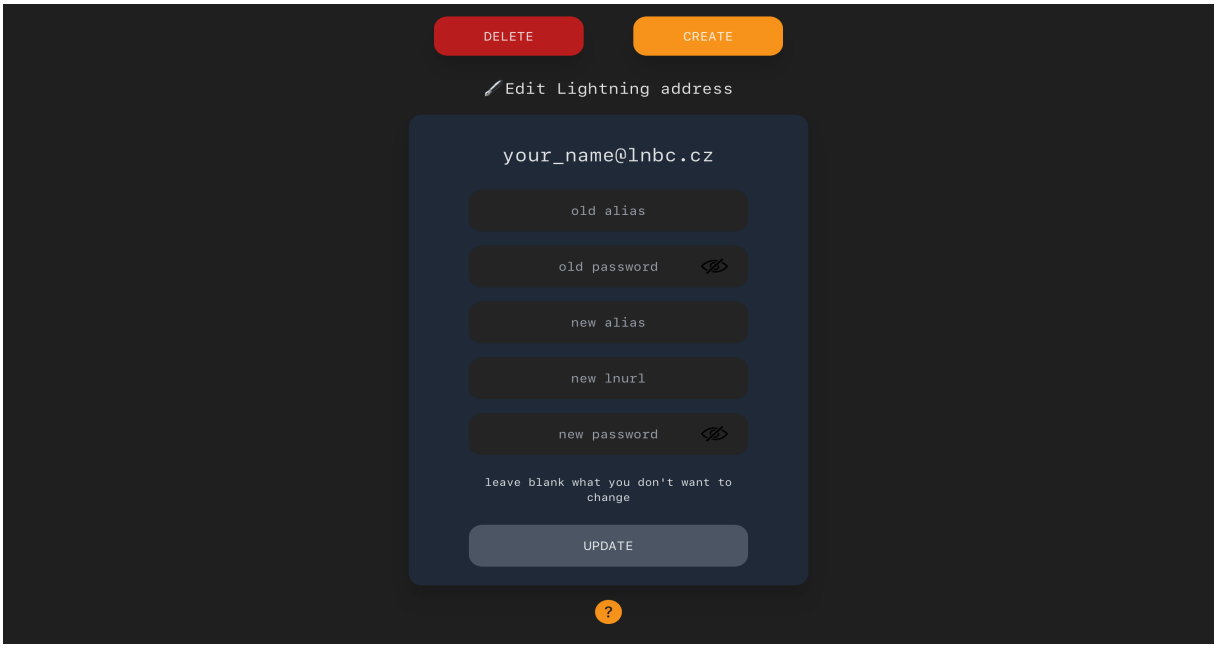
3.4.1 Frontend



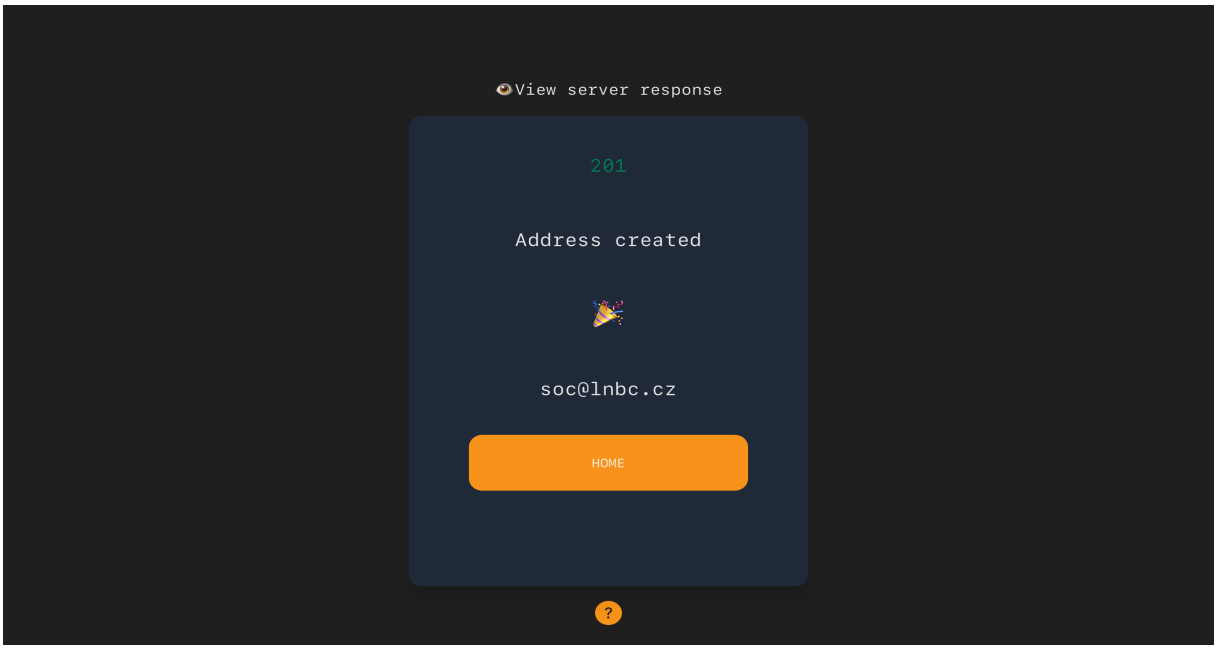
Obrázek 32 - Rozhraní pro vytvoření adresy



Obrázek 33 - Rozhraní pro smazání adresy



Obrázek 34 - Rozhraní pro editaci adresy



Obrázek 35 - Úspěšné vytvoření adresy

Aplikace je plně responsivní. Technologie používané na frontendu jsou Svelte a Tailwind CSS. Po změně akce (například z vytvoření adresy na smazání adresy) se aplikace překreslí pomocí JavaScriptu bez opětovaného připojení k serveru, jedná se tedy o takzvanou single-page aplikaci.

3.4.2 Backend

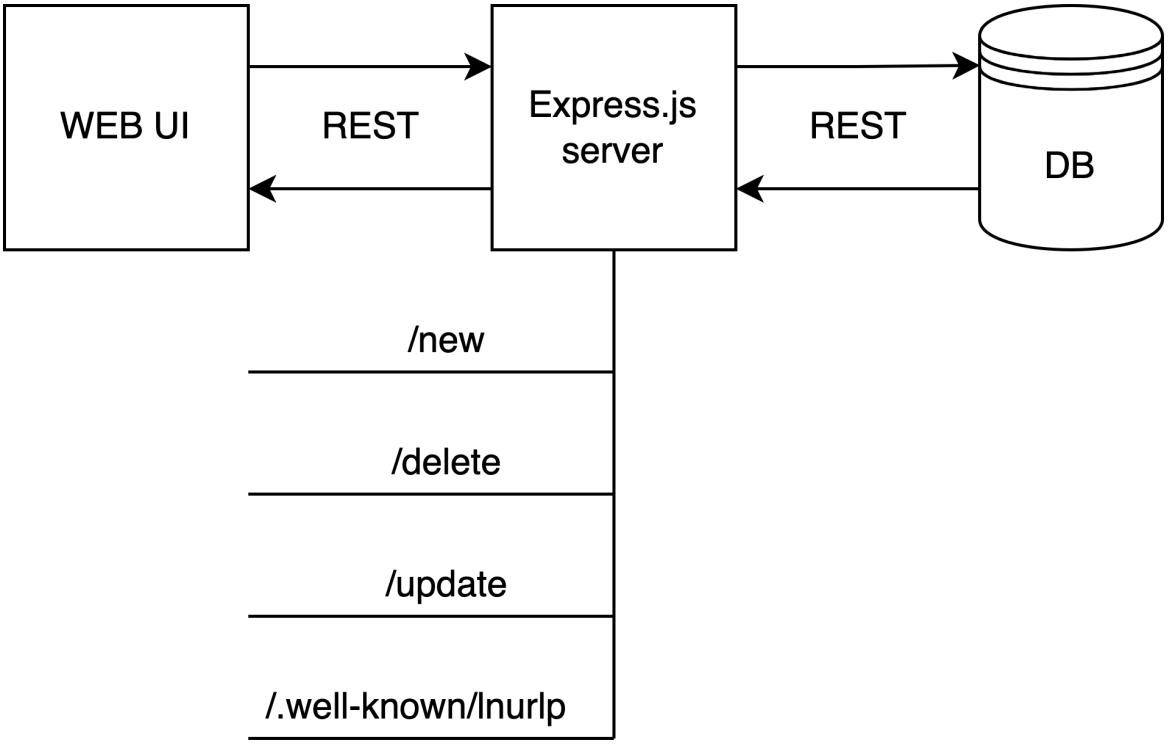
Na straně serveru se nachází dynamická cesta pojmenovaná po aliasu adresy. Ve chvíli, kdy na konkrétní endpoint přijde požadavek, aplikace porovná název endpointu se záznamy v databázi. Pokud jej nalezne, získá LNURL zadanou při vytváření adresy. Tuto LNURL následně dekóduje a na získanou URL pošle dotaz. Odpověď v JSON formátu od získané URL poté pošle jako vlastní odpověď původnímu žadateli.

```
app.get('/.well-known/lnurlp/:alias', async (req, res) => {
    const {alias} = req.params;
    const user = (await db.fetch({"alias": alias})).items[0];
    if (user) {
        try {
            res.json(await (await fetch(decodeLnurl(user.lnurl))).json());
        } catch {
            res.status(400).json({"message": "Could not reach LNURLp server"});
        }
    } else {
        res.status(404).json({"message": "user not found"});
    }
});
```

Ukázka kódu 18 - Kód pro vytvoření endpointu podle standartu lightning adres

Konkrétní endpointy mají tvar https://lnbc.cz/.well-known/lnurlp/[alias]. Tento formát není náhodný. Ve standardizaci protokolu lightningových adres[23] je uvedena tato konkrétní cesta. Pokud by tento standart nebyl dodržen, peněženky by nemohly určit endpoint, jelikož tato informace není v adrese obsažena.

[23] https://github.com/andrerfneves/lightning-address/blob/master/README.md



Obrázek 36 - Schéma aplikace s vyobrazenými endpointy serveru

Na backendu je použit Node.js s frameworkem Express. Aplikace používá autentizaci uživatelů na základě aliasu adresy a uživatelem zvoleného hesla. Hesla jsou zahashovaná s použitím soli a uložena v databázi. Uživatel tedy může jednoduše upravovat údaje spojené s konkrétní adresou.

```
const hash = bcrypt.hashSync(secret, bcrypt.genSaltSync(10));
```

Ukázka kódu 19 - Hashovaní s použitím soli

```
if (user && bcrypt.compareSync(secret, user.hash)) {
    await db.delete(user.key);
    res.status(200).json({ "message": "deleted" })
} else {
    res.status(401).json({ "message": "unauthorized" });
}
```

Ukázka kódu 20 - Logika autorizace

3.5 Gitbook

Teoretická část této práce byla zveřejněna v online podobě ve formě markdownu[24]. Jedná se o obecný popis a rychlý přehled technických informací o Bitcoinu. Bylo použito služby Gitbook. Cílem je zvětšit dosah informací shrnutých v teoretické části této práce. Text je k nalezení zde: https://mario-doskocil.gitbook.io/bitcoin/

[24] https://cs.wikipedia.org/wiki/Markdown

4 ZÁVĚR

Práce představila teoretické koncepty, na nichž Bitcoin stojí, a dále poskytla praktické ukázky interakce s Bitcoinem z pohledu vývojáře aplikací. Aplikace, které byly uvedeny jako příklad, mají praktické využití. lnAddresses a lnNote mají i svou uživatelskou základnu. Čtenář by nyní měl mít konkrétní představu o fungování Bitcoinu a jeho integraci do vlastní aplikace.

5 POUŽITÁ LITERATURA

[1] Symetrická kryptografie – Wikisofia. [online]. Copyright © 2013 ISSN [cit. 03.12.2021]. Dostupné z: https://wikisofia.cz/wiki/Symetrická_kryptografie

[2] Bitcoin — The most secure Network worldwide | by Max Mittelstaedt I Bitcoin for Governments | Medium. Medium – Where good ideas find you. [online]. Dostupné z: https://medium.com/@Max_BTC/bitcoin-the-most-secure-network-worldwide-7f4fa2e402dc

[3] P2P Network — Bitcoin. Getting Started — Bitcoin [online]. Copyright © Copyright Bitcoin Project 2009 [cit. 19.12.2021]. Dostupné z: https://developer.bitcoin.org/devguide/p2p_network.html

[4] Tor - Bitcoin Wiki. [online]. Dostupné z: https://en.bitcoin.it/wiki/Tor

[5] List of address prefixes - Bitcoin Wiki. [online]. Dostupné z: https://en.bitcoin.it/wiki/List_of_address_prefixes

[6] What is SegWit? - Bitcoin Magazine - Bitcoin News, Articles and Expert Insights. Bitcoin Magazine - Bitcoin News, Articles and Expert Insights [online]. Copyright © 2022 [cit. 14.07.2022]. Dostupné z: https://bitcoinmagazine.com/guides/what-is-segwit

[7] Mobilefish.com - Cryptocurrency address generator and validator (v1.1). Mobilefish.com - The web development, programming, internet of things, blockchain and other technologies resource. [online]. Dostupné z: https://www.mobilefish.com/services/cryptocurrency/cryptocurrency.html

[8] Secp256k1 - Bitcoin Wiki. [online]. Dostupné z: https://en.bitcoin.it/wiki/Secp256k1

[9] Graphical Address Generator. The Royal Fork [online]. Dostupné z: https://www.royalfork.org/2014/08/11/graphical-address-generator/#learnmeabitcoin

[10] Developer Guides — Bitcoin. Getting Started — Bitcoin [online]. Copyright © Copyright Bitcoin Project 2009 [cit. 14.07.2022]. Dostupné z: https://developer.bitcoin.org/devguide/index.html

[11] Block Chain — Bitcoin. Getting Started — Bitcoin [online]. Copyright © Copyright Bitcoin Project 2009 [cit. 14.07.2022]. Dostupné z: https://developer.bitcoin.org/devguide/block_chain.html

[12] Genesis block - Bitcoin Wiki. [online]. Dostupné z: https://en.bitcoin.it/wiki/Genesis_block

[13] Block Header. How Does Bitcoin Work? [online]. Dostupné z: https://learnmeabitcoin.com/technical/block-header

[14] Anatomie bloku a těžby. Bitcoinový slovník naučný [online]. Dostupné z: https://btc-slovnik.cz/pages/block.html

[15] bips/bip-0009.mediawiki at master · bitcoin/bips · GitHub. GitHub: Where the world builds software · GitHub [online]. Copyright © 2022 GitHub, Inc. [cit. 14.07.2022]. Dostupné z: https://github.com/bitcoin/bips/blob/master/bip-0009.mediawiki

[16] bips/bip-0141.mediawiki at master · bitcoin/bips · GitHub. GitHub: Where the world builds software · GitHub [online]. Copyright © 2022 GitHub, Inc. [cit. 14.07.2022]. Dostupné z: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki#other-consensus-critical-limits

[17] Merkle Tree | Brilliant Math & Science Wiki. Brilliant | Learn interactively [online]. Dostupné z: https://brilliant.org/wiki/merkle-tree/

[18] Páruje se sá se sebou

[19] Everything you need to know about Bitcoin mining. Everything you need to know about Bitcoin mining [online]. Copyright © 2010 [cit. 14.07.2022]. Dostupné z: https://www.bitcoinmining.com

[20] Mining - Bitcoin Wiki. [online]. Dostupné z: https://en.bitcoin.it/wiki/Mining

[21] Mining — Bitcoin. Getting Started — Bitcoin [online]. Copyright © Copyright Bitcoin Project 2009 [cit. 14.07.2022]. Dostupné z: https://developer.bitcoin.org/devguide/mining.html

[22] Coinbase Transaction. How Does Bitcoin Work? [online]. Dostupné z: https://learnmeabitcoin.com/technical/coinbase-transaction

[23] Some things you need to know - Bitcoin. 302 Found [online]. Copyright © Bitcoin Project 2009 [cit. 14.07.2022]. Dostupné z: https://bitcoin.org/en/you-need-to-know

[24] How Much Would it Cost to 51% Attack Bitcoin? | Braiins. Braiins | Bitcoin mining company [online]. Dostupné z: https://cs.braiins.com/blog/how-much-would-it-cost-to-51-attack-bitcoin

[25] Genesis block - Bitcoin Wiki. [online]. Dostupné z: https://en.bitcoin.it/wiki/Genesis_block

[26] Lightning Network. Lightning Network [online]. Dostupné z: https://lightning.network

[27] PRITZKER, Yan. Vynález jménem bitcoin. Přeložil Tereza WONGOVÁ. [Praha]: Braiins Publishing, 2020. ISBN 978-80-907975-0-5.

6 SEZNAM OBRÁZKŮ A TABULEK

7 SEZNAM ZKRATEK

- API – Application Programming Interface
- ASIC - Application-specific integrated circuit
- BTC – Bitcoin
- CSS - Cascading Style Sheets
- ECC – Eliptic-curve cryptography
- ECDH – Elliptic-curve Diffie–Hellman
- ECDSA – Elliptic Curve Digital Signature Algorithm
- GB – Gigabyte
- HTML – Hypertext Markup Language
- HTTP – Hypertext Transfer Protocol
- JSON – JavaScript Object Notation
- LNURL – Lightning URL
- LNURLp – LNURL pay
- LNURLw – LNURL withdrawal
- NDEF - NFC Data Exchange Format
- NFC – Near Field Communication
- QR - Quick Response (code)
- REST - Representational State Transfer
- SAT – Satoshi
- SD – Secure Digital
- SHA – Secure Hash Algorithm
- TB – Terabyte
- Tor – The Onion Router
- URL - Uniform Resource Locator

8 ZDROJOVÉ KÓDY

Všechny aplikace popsané v praktické části je možné nalézt na serveru github.com.

- BTC dashboard - https://github.com/MarioDoDo/BTCdashboard
- LNnote – https://github.com/MarioDoDo/lnNote
- Lightning addresses – https://github.com/MarioDoDo/lnAddresses