Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Použitelnost decentralizovaných technologií pro vývoj webových aplikací (Web 3.0)

Diplomová práce

Autor: Daniel Schmid
Studijní obor: Aplikovaná informatika

Vedoucí práce: prof. RNDr. PhDr. Antonín Slabý, CSc.

Hradec Králové Květen 2023

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 20.4.2023 Daniel Schmid

Poděkování:
Děkuji vedoucímu diplomové práce prof. RNDr. PhDr. Antonínu Slabému, CSc. za odborné vedení, vstřícnost a cenné rady při zpracování této práce.

Anotace

Práce se zabývá výzkumem exitujících technologií pro vývoj decentralizovaných aplikací, které mají být součásti konceptu webu 3.0. Popisuje vlastnosti a fungování jednotlivých technologií a zkoumá, které z nich jdou nejlépe aplikovat na vývoj uživatelsky přívětivé aplikace s ohledem na bezpečnost ukládání uživatelských dat. Z těchto všech možností je následně vybrána technologie, případně kombinace technologií, jejichž praktické použití je součástí ukázek v praktické části práce. Výsledkem práce je též závěr, zda jsou již existující řešení připravena na vývoj i jiných než finančních decentralizovaných aplikací (DEFI), jsou probrány výhody a nevýhody těchto řešení a navržen způsob, který by tyto nevýhody nebo chybějící technologie mohl vyřešit.

Annotation

Title: Applicability of decentralized technologies for web application development (Web 3.0)

The work deals with the research of existing technologies for the development of decentralized applications, which are part of the Web 3.0 concept. It describes the properties and functioning of individual technologies and examines which of them can best be applied to the development of a user-friendly application regarding the security of user data storage. From these options, a technology, or a combination of technologies is subsequently selected, the practical use of which is part of the demonstration in the practical part of the work. The result of the work is also a conclusion as to whether already existing solutions are ready for the development of non-financial decentralized applications (DEFI), the advantages and disadvantages of these solutions are discussed, and a method that could solve these disadvantages or missing technologies is proposed.

Obsah

1 Úvod

Internet lze považovat za nezbytný prostředek pro fungování dnešního světa. Jeho prostřednictvím zůstávají lidé v kontaktu, ať už se nachází kdekoliv. Velký význam má i pro firmy. Z korporací jako Google, Facebook, Amazon, Apple se stávají giganti sbírající více a více uživatelských dat, které využívají pro zvýšení svých zisků. Běžným uživatelům pak nezbývá než tento fakt akceptovat pro potřebu využívání služeb těchto korporací. Využívání osobních dat se však je však kritizováno regulačními orgány jak ve Spojených státech Amerických, tak i v Evropské unii, kde je výsledkem např. vznik nařízení GDPR (nařízení o ochraně osobních údajů). V konečném důsledku to však tyto giganty příliš neomezuje.

Tento trend se ale nezamlouvá i několika významným lidem. Příkladem může být Sir Timothy John Berners-Lee, který je tvůrcem World Wide Webu a ředitel konsorcia W3C, které se stará o pokračující vývoj webu. Inrupt, což je společnost, kde je spoluzakladatelem, v roce 2018 reaguje spuštěním projektu Solid, který má dát uživatelům moc nad vlastními osobními daty. Tyto data se mají ukládat v individuálních datových trezorech (na serverech) a společnosti mohou k těmto datům získat přístup jen se svolením uživatele prostřednictvím zabezpečeného odkazu. Je však reálné donutit tyto společnosti, aby se vzdaly moci nad daty svých uživatelů?

V posledních letech se zvyšuje zájem o decentralizované aplikace neboli DApp. Ty jsou však nejvíce využívány u aplikací finančních právě díky nepřítomnosti centrální autority a výborného zabezpečení. Decentralizované aplikace, jak již název napovídá, nevyužívají centrálního prvku, tedy serveru (datového uložiště), kde jsou data hromadně ukládány a zpracovávány. Místo toho jsou data uložena na zařízeních klientů používajících tyto decentralizované aplikace. Tato myšlenka ukládání dat nadchla některé natolik, že by tyto aplikace měly být součástí konceptu webu 3.0.

2 Motivace a cíl práce

Web 3.0 ale není přesně definován a není tedy jasné, co všechno by měl umět. Hlavním cílem je získat bezpečné a svobodné místo, které bude postavené na decentralizovaných systémech. To by mělo zamezit vydělávání na uživatelských datech. Mezi znaky webu 3.0 patří sémantický web, decentralizované aplikace (DApp), sdílené aplikace (GoogleDocs), dotazování v přirozeném jazyce, internet věcí (IoT) či případně částečná umělá inteligence webu. [1]

Jedním z hlavních pilířů webu 3.0 je snaha o decentralizované systémy a distribuovaná data. Distribuovaná data se vyznačují svou efektivností přistupovat k datům skrze nejbližší zdroj. Tímto zdrojem pak nebude nic jiného než ostatní uživatelé daného systému. To znamená, že data budou ukládána na zařízení všech uživatelů. Je ale tento přístup bezpečnější? Aplikace pro kryptoměny se považují za bezpečné, avšak žádná z nich neukládá osobní data. Identitu konkrétní osoby ve světě kryptoměn zastupuje tzv. kryptopeněženka. Ta není nijak svázaná s identitou reálného člověka a uživatel si jich může vytvořit nespočet. Stejně tak lze tuto peněženku považovat za přenosnou z osoby na osobu. Pokud uživatel bude chtít, může předat přístup ke své peněžence komukoliv jinému. Bezpečnost tedy spočívá hlavně v tom, že přes nesouhlas uživatele se ke správě jeho peněženky nikdo jiný nedostane. Bezpečné jsou i transakce mezi těmito peněženkami, které musí být schváleny a zfalšování dat je obtížné až dokonce nemožné.

Jak již bylo zmíněno, tato kryptopeněženka není svázaná se skutečnou identitou uživatele. To mnohdy vede i k nelegálním obchodům, jelikož (ne)výhodou kryptoměn je i nedohledatelnost uživatele vzhledem ke skutečnosti, že tyto aplikace právě nepracují s osobními daty uživatelů.

Lze tedy ukládat data uživatelů decentralizovaně a zároveň bezpečně, aby k nim nemohli ostatní uživatelé získat bez povolení přístup? A nebude pro běžného uživatele používání decentralizovaných aplikací nebo webu 3.0 složitější než používání aplikací, na které jsou doteď zvyklí?

Odpověďmi na tyto otázky se bude zabývat tato diplomová práce, která bude zkoumat existující řešení pro vývoj DApp a webu 3.0 a zohledňovat bezpečnost a složitost používaní těchto technologií pro běžného uživatele.

3 Historický vývoj webu

Za vynálezce webu je nejčastěji považován Sir Tim Burner-Lee (1989). Postupný vývoj webu lze rozdělit do několika etap, a to web dokumentů (web 1.0), web uživatelů (web 2.0) a web dat (web 3.0). [2]

3.1 Web 1.0

Web 1.0 sloužil převážně jako zdroj různorodých, často komplexních a multimediálních informací, které byly propojeny pomocí hypermediálních odkazů [2]. Weby byly vytvářeny pouze několika málo autory a zbytek uživatelů mohl obsah těchto webů pouze číst. Proto bývá tato verze označována jako „read-only web“. Tyto weby neobsahovaly totiž žádné formuláře, ovládací prvky a obecně postrádaly prvky interaktivity s uživatelem, které jsou dnes naprosto běžné. Uživatel kromě procházení obsahů těchto webů, neměl žádnou možnost, jak s ostatními uživateli komunikovat, nebo případně reagovat na konkrétní obsah. [1–3]

Tyto stránky nazýváme statickými stránkami, jelikož postrádají jakoukoliv dynamičnost. Veškerý obsah pocházel pouze ze souborového systému daného serveru. Informace nebyly ukládány do databází, jako tomu bývá dnes, ale vše bylo staticky obsaženo v HTML souborech. Hlavním cílem tedy bylo vytvoření společného informačního prostoru pro uživatele. Období této verze trvalo přibližně od roku 1989 do roku 2005. [1–3]

3.2 Web 2.0

Termín web 2.0 byl oficiálně zaveden v říjnu roku 2004 pány Dalem Doughertem a Craigem Clinem prostřednictvím konferenčního brainstormingu. Nejedná se však o další vývoj webu 1.0, ale spíše o rozšíření původních myšlenek a principů předchozí verze. [2, 3]

Web 2.0 přidává dynamičnost, kdy spolu uživatelé přicházejí do interakce, ať už na sociálních sítích, tak v různých webových fórech zabývající se konkrétními tématy, případně skrze chatovací aplikace. Obsah nemusí být staticky uložen v souboru, ale je načítán například z relačních databází, kdy se dokáže přizpůsobit

požadavkům uživatele. Různým uživatelům se tedy může načítat různý obsah. Díky této dynamičnosti uživatelé mohou obsah, jak prohlížet, tak i vytvářet. [1, 2]

Zároveň s neustálým vývojem HTML konsorciem W3C vznikají další nové možnosti při vytváření webů. Těmi je například snazší práce s přehráváním videí, přístup k hardwaru zařízení uživatele jako GPS sensor, kamera či mikrofon.

Pro získávání dat z webů nebo pro výměnu dat mezi webovými službami lze vytvářet aplikační programovací rozhraní (API), například REST, SOAP, GraphQL, atd. [4]

Díky těmto vlastnostem bývá web 2.0 často nazýván jako web znalostní, zamřený na sociální interakci mezi lidmi či web pro čtení a zápis. [2]

3.3 Web 3.0

Termín web 3.0 byl poprvé použit roku 2006 Johnem Markoffem z New York Times. Prvotní myšlenkou bylo hlavně sémantické označování obsahu, tj. sémantický web, s kterou přišel již Tim Berners Lee (vynálezce webu). Pomocí metadat jsou informace převedeny na smysluplné informace, které jsou lépe zpracovatelné inteligentními softwarovými agenty. [2, 3]. Web 3.0 má nestrukturované informace na webu dávat do souvislostí na základě významu kontextu, ve kterém se informace nachází. Stroj má datům porozumět a katalogizovat je podobným způsobem jako člověk. Takto shromážděná data mají být kategorizována hierarchickým způsobem pro snadné a specifické vyhledávání. [3]

S technologickým vývojem však přibývají další názory, co všechno by měl web 3.0 umět. Koncept sémantického webu jde spíše do pozadí a do popředí se dostává problematika poněkud odlišná. Jak již bylo zmíněno, vize webu 3.0 si momentálně dává za cíl i decentralizaci a distribuované ukládání dat uživatelů, kdy jim bude ponechána vláda nad jejich osobními daty. [1]

Součástí této vize má být i umělá inteligence a strojové učení. Díky tomu dokáže stroj lépe zpracovat přirozený jazyk a může lépe reagovat na potřeby uživatelů. [1] Nové informace by pak měly být vytvářeny spíše počítači než lidmi. [4] Příkladem může být ChatGPT, který v závislosti na vstupu od uživatele dokáže snadno

nabídnout hledanou informaci, případně vysvětlit danou problematiku nebo reagovat v kontextu na předchozí komunikaci - to vše v rodném jazyce uživatele.

Velmi populární je též myšlenka webu 3D, kde jsou známé služby jako Second life či Red Light. Web3D umožňuje lidem žít ve virtuálním světe skrze svého avatara, kde se můžou setkávat s ostatními lidmi, případně se společně účastnit skupinových aktivit. Vedle reálného života pak může mít člověk i život virtuální. [5]

Koncept webu 3.0 se tedy během let výrazně změnil. Nejedná se pouze o web sémantický, jak původně bylo zamýšleno, ale možnosti, které by měl nabízet, se velmi rozšířily.

3.4 Web 4.0 a Web 5.0

O termínu web 4.0 vznikaly články již od roku 2012, kdy bývá označován jako web symbiotický. Obecně se od této vize očekává, že lidé budou v symbióze se stroji [6, 7]. To může znamenat, že by umělá inteligence byla natolik vyspělá, aby pomocí našich myšlenek mohla procházet web. [6]

Jako u webu 3.0 se jedná spíše o návrhy, co všechno by měl tento koncept zvládat a stejně tak se mohou s technologickým vývojem požadavky postupem času měnit. Některé zdroje dokonce připisují webu 4.0 koncepty z webu 3.0 – například digitální alter ego, což souvisí s web3D [8].

Existují i zdroje, které se snaží definovat web 5.0, přestože ani web 3.0 a ani web 4.0 nemají jasně určeno, co všechno mají umět. Navíc koncept webu 5.0 je velmi podobný konceptu webu 3.0, kde jde hlavně o decentralizaci. Jack Doersey, který se snaží koncept webu 5.0 prosadit, poukazuje na to, že decentralizace webu 3.0 je založena hlavně na blockchainu [9–11], i když definice toho, co všechno má web 3.0 zvládat, není přesně dána, a tudíž není ani nijak specifikováno pomocí jakých technologií toho má dosahovat. Proto nelze říci, že web 3.0 musí pro decentralizaci dat používat striktně blockchain. Je pravdou, že tato technologie se decentralizovaným a distribuovaným systémům přibližuje nejvíce. Avšak vznikají i jiné technologie, které blockchain nevyužívají vůbec nebo pouze okrajově, například pro zajištění integrity dat. Technologie probrané v dalších kapitolách jsou příkladem toho, že decentralizovaného přístupu lze dosáhnout i bez použití blockchainu.

3.5 Současná verze webu

Jaká je tedy aktuální verze webu? To nelze jednoznačně určit. Je zřejmé, že dnešní weby podporují požadavky webu 1.0 a naprostá většina i webu 2.0, jelikož podstatná část webů umožňuje interakci mezi uživateli, například formou komentářů u produktů nebo článků. Stejně tak jsou již zaběhnuté některé funkce webu 3.0. Poměrně rychlým tempem se rozrůstají nástroje pracující s umělou inteligencí upravující fotografie, zvuk nebo video na základě strojově natrénovaného modelu. Pokud jde o sémantický web, vyhledávače poskytují mnohem lepší výsledky, než tomu bylo před několika léty. Nicméně sémantický web za použití OWL (Ontology Web Language), RDF (Resource Description Framework) a SPARQL (dotazovací jazyk nad daty), které velmi často autoři [3, 4, 6] ve starších článcích (2012 – 2016) zmiňovali, příliš používané nejsou, většinou pouze pro specifické odvětví.

Decentralizované aplikace jsou rovněž součástí dnešního světa, spíše však v podobně aplikací finančních (DEFI), případně aplikací, kde chytré kontrakty spolu s blockchainem umožňují uzavírat smlouvy bez zprostředkovatele. To tedy naznačuje, že funkcionality, které by měl web 3.0 umožňovat, jsou již zcela nebo alespoň částečně používány. Pokud by se ale měřilo, jakých funkcionalit využívají uživatelé nejvíce, jistě by stále vyhrál web 2.0.

Je tedy zřejmé, že tímto způsobem nelze určit, jaká verze webu je aktuální. Tyto verze nemají sloužit pro orientační zjištění, v jaké fázi vývoje se v současné době web nachází, ale zejména ukazují, kam by mohl budoucí vývoj webu směřovat. Určení současné verze není tedy pro další vývoj webu příliš důležité.

4 Decentralizované aplikace a Blockchain

4.1 Rozdělení systémů

Tato část by měla zodpovědět na následující otázky. Jaký je rozdíl mezi decentralizovaným, distribuovaným a centralizovaným systémem? Jedná se v případě decentralizovaných a distribuovaných systému o totéž? Je možné, aby systém splňoval všechny tyto vlastnosti?

Centralizované systémy jsou momentálně nejpoužívanějším modelem pro softwarové aplikace. Tyto systémy řídí tok informací a případně ostatní zařízení z jednoho centra. Na tomto centru je vše závislé, jak odesílání či přijímání informací, tak i celková dostupnost systému. Pokud centrální bod selže, může selhat celý systém. Tyto systémy požívají i ty největší firmy jako Meta Platforms (Facebook), Amazon či Google. [12]

U distribuovaných systému je výpočet distribuován mezi více uzlů namísto jednoho. Například Google kromě centrální architektury začal používat i interně distribuovanou architekturu, aby urychlil výpočetní i datovou latenci. Takto může být systém zároveň centralizovaný i distribuovaný. [12]

Decentralizovaný systém oproti centralizovanému nezadává žádnému jinému uzlu příkazy, co má vykonat. I tento systém však může být zkombinován s distribuovaným systémem. Například blockchain, který je probrán v kapitole 4.3, je distribuován, jelikož jeho blokový řetězec se nachází na více počítačích. Zároveň je i decentralizovaný, protože pokud jeden uzel selže, síť dokáže bez problému fungovat dál. V praxi to tedy znamená, že každou aplikaci, která používá blockchain, lze distribuovat a decentralizovat. [12]

4.2 Požadavky na DApp

Decentralizovaný systém by tedy neměl mít nějaký centrální prvek, který by řídil ostatní. [1] Co to ale v praxi znamená? Nemůže mít třeba alespoň pomocný centrální prvek, který by například byl jen prostředníkem v komunikaci mezi uzly?

Komunitě vadí hlavně centrálně spravovaná data. Proto je důležité, aby data byla distribuována mezi více uzlů. Není ani zcela nutné vynechání centrálního prvku. Jde například o potřebu navázání spojení s dalšími uzly v síti, kde se používá pouze protokol HTTP. Existují protokoly jako BitTorrent, kde některé jeho implementace využívají centrálního trackeru, ale data jako taková jsou distribuována mezi uzly. Tracker se pouze stará o vzájemnou viditelnost mezi klienty. Na druhé straně toto může připomínat i distribuovaný centralizovaný systém, který používají velké korporace. Zde však data nejsou mezi uzly uživatelů jako v případě BitTorrentu, ale mezi servery jedné společnosti, což porušuje hlavní myšlenku DApp, aby data byla ve správě uživatelů.

Navíc systém, kde existuje centrální bod selhání, nelze už jen z jeho podstaty považovat za decentralizovaný. Není ale vyloučeno, aby server provozovatele byl součástí sítě jako jeden z uzlů a měl funkci záložního uzlu. Pokud by nastala situace, že je v síti malé množství aktivních klientů, tak by záložní uzel provozovatele dokázal systém udržet dostupný.

Dalším požadavkem na decentralizovanou aplikaci je její otevřený zdrojový kód. [1] Je logické, že bez přístupného zdrojové kódu, nemusí uživatelé dané aplikaci uvěřit, že je opravdu decentralizovaná a data například neposílá někam na své servery, kde je dále zpracovává. Open-source kód ale přináší vždy i nějaké nevýhody. Pokud udělá vývojář nějakou chybu, veřejně ji vystaví a jakýkoliv uživatel se zlými úmysly ji může zneužít. Vývojáři pak musí být obratní a chyby řešit velmi rychle, případně se spoléhat na komunitu, která je nahlásí a nebude jich zneužívat. Dalším problémem může být slabé zabezpečení. Existují systémy využívající architektury security by obscurity, která spoléhá na to, že uživatelé neví, jak vlastně vnitřně systém funguje a jak je reálně zabezpečen. Faktem však zůstává, že tento typ zabezpečení není příliš doporučován, jelikož utajení vnitřní implementace systému

může být prolomeno. Je tedy lepší využívat prověřených a bezpečných algoritmů v open-source kódu, a tím dát uživatelům najevo, jak dobře je o jejich data postaráno.

Velmi diskutabilní je i nepřítomnost cenzury v DApp. Je logické, že při nepřítomnosti centrální správy dat, nebudou data nijak cenzurována . Jenže ne vždy je cenzura špatná. Například cenzura, kterou aplikují centralizované systémy, se týká prodeje nelegálního zboží jako drog a zbraní, nebo pornografie či dokonce dětské pornografie. Tento typ dat není například cenzurován na deep webu, což může být jeden z důvodů, proč není příliš používán širokou veřejností. Málokdo by totiž chtěl přicházet do styku s tímto obsahem. Nepřítomnost cenzury by tedy klidně mohla vést k tomu, že by se decentralizované aplikace používaly pouze pro nelegální záležitosti, a tím pádem si nezískaly důvěru u širší veřejnosti.

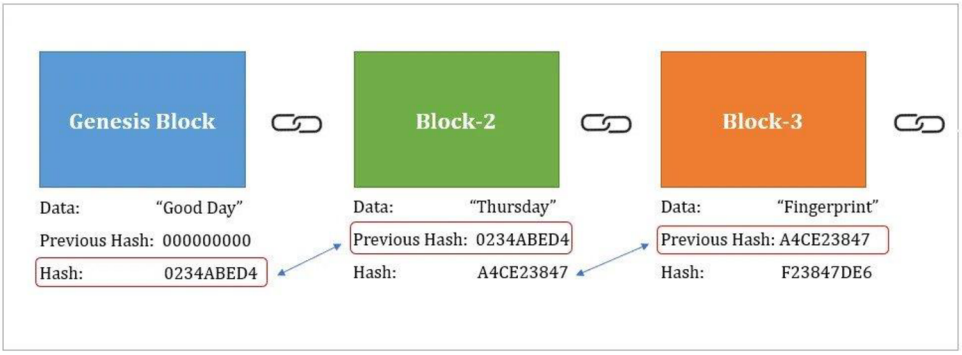
Ani zde tedy není přesně definováno, jak má být decentralizovaná aplikace implementována. Vždy svým způsobem závisí hlavně na vývojářích, jakým směrem se rozhodnou vydat. Nejdůležitější však je možnost uživatelů kontrolovat svá vlastní data a ověřit si, jak systém s jejich daty nakládá (open-source). Cenzura jako taková je velmi citlivé téma, které se špatně řeší i v systémech centralizovaných, natož v systémech decentralizovaných, kde není zatím moc možností, jak nelegální obsah cenzurovat.

4.3 Blockchain

Blockchain je v současné době velmi často zmiňovanou technologií, od které se očekává další rozvoj u decentralizovaných aplikací v rámci konceptu webu 3.0. Je velmi využívaný v decentralizovaných financích, právě díky své bezpečnosti. Některé technologie pro vývoj DApp zmíněné v dalších kapitolách jsou na blockchainu postaveny, a proto je zde pro lepší porozumění problematika blockchainu mírně přiblížena.

Blockchain je distribuovaná a decentralizovaná databáze, která je postavena na nedůvěryhodných transakcích. Tato databáze bývá velmi často označována jako účetní kniha. Je veřejně přístupná všem, a i přesto je velmi odolná proti neoprávněné manipulaci s daty. Této odolnosti je dosaženo díky decentralizovanému konsenzu, kdy je potřeba každou transakci ověřit mezi uzly. Důležité je zmínit, že data již obsažená v blockchainu jsou neměnná. Kopii celého neměnného blockchainu mají všechny uzly uloženou u sebe. [12]

Kromě dat, které identifikují obě strany transakce (příjemce a odesílatele) a částku, může obsahovat i další dodatečná data (v závislosti na potřebě dané aplikace). Důležitým prvkem je hash, který se vytvoří na základě dat v transakci a tento hash je pak součástí dalšího záznamu. Každý záznam tedy obsahuje hash záznamu předchozího, a právě tento hash je i součástí výpočtu hashe aktuálního záznamu. Tento způsob zaručuje nemožnost neoprávněně měnit data, jelikož by došlo ke změně hashe, který by pak nesouhlasil s hashem následujícího záznamu, a ten s dalším záznamem atd. Na základě toho, by pak kopie blockchainu tohoto uzlu byla označená jako neplatná. [12]



Obrázek 1 Blockchain hash
Zdroj: [13]

4.3.1 Transakce

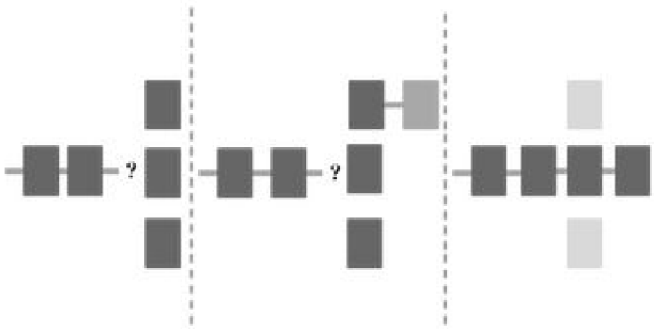
Jelikož blockchain pracuje s nedůvěryhodnými transakcemi, musí každý uzel ověřit, že požadavek na transakci pochází z autentického zdroje. Toho je dosaženo pomocí digitálního podpisu, který je vytvořen za použití požadavku na zprávu a soukromého klíče žadatele. Tento digitální podpis je pak používán uzly v síti, které ověřují autentičnost požadavku na základě veřejného klíče žadatele. Takto je zajištěna integrita zprávy, protože ji nelze během předávání změnit, tak aby nedošlo k znehodnocení digitálního podpisu. Výhodou závislosti podpisu na zprávě je, že soukromý klíč nemůže být znám, a tedy nemůže být použit někým jiným pro jinou transakci. Digitální podpis pro zajištění pravosti bez ohrožení bezpečnosti je vypočten pomocí algoritmu eliptické křivky ECDSA. [14]

Po ověření transakce veřejným klíčem, je přidána s dalšími transakcemi do bloku. Aby mohl být takto přidaný blok propojen s blokem předchozím, je potřeba jej zkontrolovat a potvrdit. K potvrzení se používá kolektivního výpočetního výkonu těžařů (uzlů) v síti, kteří řeší složitý matematický problém. Tato hádanka se řeší do té doby, dokud některý z uzlů nenajde takové řešení, které spolu s daty bloku generuje hash, který odpovídá podmínkám sítě pro uzamčení bloku. Jakmile je správně řešení nalezeno, je blok ověřen a propojen s posledním ověřeným blokem v blockchainu. Tím se aktualizuje sekvenční účetní kniha, jejíž kopie je distribuována mezi ostatní uzly v síti. [14]

Seskupování transakcí do bloků řeší problém dvojího utrácení. Jelikož blockchain je nedůvěryhodná síť, nevěří se, že někdo nevydal dvě transakce pro stejný zdroj. Takto v bloku seskupené transakce se považují za takové, jako by nastaly ve stejnou dobu. Pokud je tedy vydána další transakce ke zdvojnásobení útraty a je ověřena jiným uzlem, který neobdržel transakci první, a tudíž ji neověřil, je i tato druhá transakce autentizována. Nicméně transakce se neprovádí, dokud nejsou přidány do sítě, a když nestane čas pro potvrzení zablokování prostředků, bude sítí odmítnuta, protože byl tento zdroj již použit. [14]

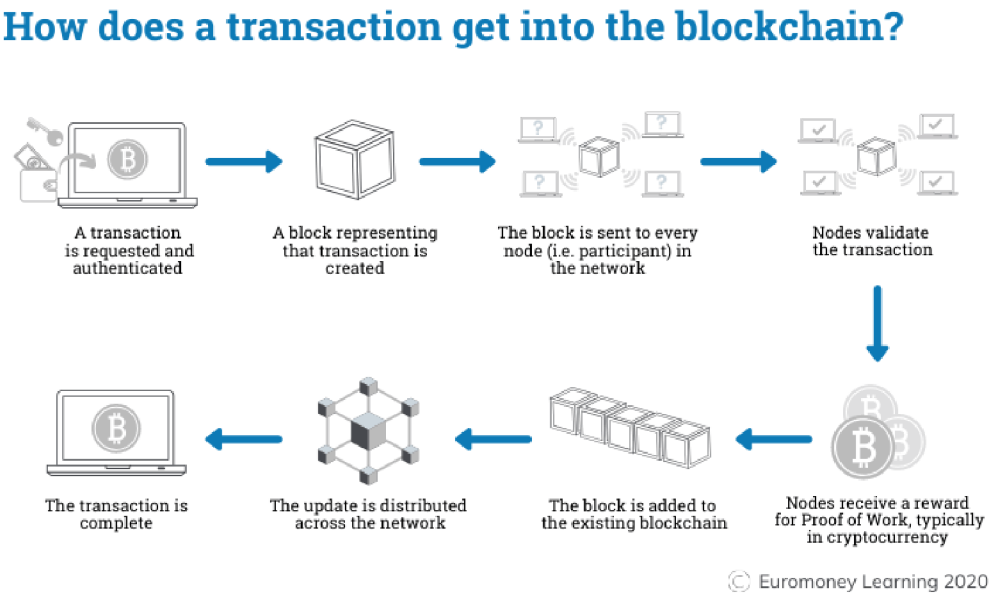
Toto řešení ovšem generuje další problém, kdy stejný blok vyřeší více těžařů v jednom okamžiku. Takto potvrzené bloky se vysílají do zbytku sítě. Uzly v síti pak staví na bloku, který obdrželi jako první. Pokud by každý uzel přidal do sítě jiný blok,

nebylo by možné jednoznačně určit konec blockchainu. I na to však existuje řešení. Blockchainová síť totiž vyžaduje, aby každý uzel přijal ten nejdelší řetězec. Takže uzel, který jako první vyřeší následující blok a propojí ho se sítí, vytvoří nejdelší řetězec, který se jako jediný použije. Ostatní (kratší) řetězce budou uzly, které je používaly, zahozeny a tyto uzly aktualizují svou verzi blockchainu na novější verzi s nejdelším řetězcem. Díky tomu bude blockchain stabilizovaný. [14]



Obrázek 2 Problém nejednoznačnosti konce řetězce
Zdroj: [14]

Potřeba velkého výkonu a závod v řešení těchto matematických problémů má pro blockchainovou síť velký přínos. Jelikož pravděpodobnost současného řešení bloků je nízká, není velká šance, aby se více bloků řešilo současně znovu a znovu. Tento závod má navíc velký přínos do bezpečnosti blockchainu. Pokud by někdo chtěl nelegálně přidat transakci, potřeboval by alespoň 51 % výpočetního výkonu této sítě a musel by soutěžit s ostatními těžaři o nejrychlejší vyřešení nejdelšího řetězce. Toto sice není vyloučeno, ale vzhledem k velikosti výkonu dosavadních blockchainových sítí, je to velmi nepravděpodobné. [14]



Obrázek 3 Přidání transakce do blockchainu
Zdroj: [15]

4.3.2 Chytré kontrakty

Chytrým kontraktem neboli chytrou smlouvou se rozumí uzavření libovolného kontraktu bez potřeby zprostředkovatele. Není tedy potřeba důvěřovat třetí straně (např. bance). Nejedná se o nic jiného než o kus kódu, který existuje v blockchainu. Ten je vykonán při validaci transakce. [12]

Každá platforma může mít vlastní skriptovací jazyk pro vytváření kontraktů. Ty pak mohou mít různé funkcionality nebo omezení (více v kapitole 4.3.5). Stejně tak mohou používat různé mechanismy schvalování, například proof of work nebo proof of stake, které jsou rozebrány v následujících kapitolách.

4.3.3 Proof of Work

Proof of Work neboli důkaz o práci je způsob, jakým jsou transakce ověřovány. Velmi často bývá tento proces označován jako těžení. Jeho hlavním cílem je zajištění

bezpečnosti sítě. Princip spočívá v nalezení složitého řešení, které je ale lehce ověřitelné. [16]

Jak již bylo zmíněno v kapitole o transakcích, součástí jednoho bloku je více transakcí, respektive všechny transakce, které vznikly od ověření posledního bloku. Úkolem těžaře (uzlu) je tento blok uzamknout (ověřit), aby se mohl stát součástí blockchainu a on tak mohl získat odměnu formou kryptoměny. [16]

Každý těžař má tento blok k dispozici spolu s hashem posledního bloku v blockchainu. Jeho úkolem je zkoušet náhodné číslo, označované jako nounce, které je součástí vstupu spolu s hashem posledního bloku a všech transakcí do hashovací funkce. Takto vzniklý hash musí být co nejmenší, tzn. musí začínat sítí určeným počtem nul. Počet nul může být měněn každých 2016 bloků v závislosti na tom, jak se těžařům daří a kolik jich momentálně je. Pokud je tedy těžařů málo, počet potřebných nul se může snížit a v opačném případě naopak zvýšit. Tímto způsobem se dynamicky udržuje obtížnost pro řešení bloku. Například bitcoinová síť se snaží, aby ověření bloku trvalo kolem 10 minut. [16]

Těžař, který dokáže zkoušením velkého množství nouncí zjistit takovou, jež by poskytla dostatečně malý hash, který bude sítí schválen, může daný blok uzamknout. Aby však nedošlo k tomu, že řešitel do tohoto bloku přidal vlastní neověřenou transakci, je toto řešení, které se nazývá golden nounce, poskytnuto ostatním uzlům, kteří provedou stejný výpočet pro kontrolu hashe. Pokud by měl řešitel u sebe nelegálně přidanou, případně upravenou transakci, hash by ostatní uzlům nevyšel stejně, a tudíž by toto řešení nebylo sítí přijato. [16]

Nalezení řešení vyžaduje obrovské množství vyzkoušených nouncí, a tedy obrovské množství vypočtených hashů. Každý těžař chce být co nejúspěšnější a uzamknout co nejvíce bloků, aby získal co největší odměnu. Z tohoto důvodu potřebuje co nejlepší výkon. Dříve bylo možné kryptoměnu Bitcoin těžit na běžném PC, jelikož pro výpočty se používalo CPU počítače. Postupem času se však zjistilo, že grafická karta, respektive GPU má mnohem větší výkon a její instrukční sada byla dostačující na to, aby šla použít pro výpočet hashů. Momentálně je však nejvyužívanějším pro těžbu bitcoinů ASIC neboli integrovaný obvod specifický pro aplikaci. [17] Díky tomu, že mnoho těžařů má výkonné stroje s několika GPU, nebo několik ASIC řešení, nelze jim dnes na běžném PC konkurovat a šance na to,

aby stolní počítač našel řešení dříve než k tomuto účelu specificky navržený hardware, je velmi malá a v praxi by tento počítač pouze spotřebovával elektrickou energii.

Takto výkonné stroje řešením těchto hádanek spotřebují obrovské množství elektrické energie. Tato skutečnost má nevýhodu z globální hlediska v celkové spotřebě energie. Výhodou pak je, že spotřeba energie, může sloužit jako motivace k tomu, aby těžař nepodváděl při schvalování transakcí, jelikož by tyto zdroje spotřeboval zbytečně. Toto řešení tedy vychází z teorie her a je nastaveno tak, aby se uzlům vyplatilo nepodvádět.

4.3.4 Proof of Stake

Jelikož důkaz o práci má za následek velkou spotřebu elektrického proudu, díky provádění mnoha zbytečných výpočtů, které jsou častěji zahozeny, než použity, vznikl návrh na nové ověřování transakcí nazývané proof of stake neboli důkaz o podílu. [16]

Tento způsob nevyužívá k ověření bloku všechny uzly sítě, ale pouze jeden vybraný uzel, kterému se říká validátor. Validátor již neprovádí hromadu výpočtů pro uzamčení bloku, ale pouze ověřuje všechny transakce v rámci bloku a v případě, že jsou validní, může blok uzamknout a přidat do blockchainu. [16]

Brání ale něco uzlu v tom, aby nějakou transakci zfalšoval a tím se obohatil? Jako v proof of work byla vkladem vynaložená elektrická energie i zde musí mít každý validátor nějaký vklad, kterým ručí. Vkladem nebývá nic jiného než kryptoměna, která tento mechanismus využívá, a to v nemalé částce. [18]

Například v síti Ethereum musí uzel zastavit až 32 ETH, což při kurzu, kdy 1 ETH odpovídá 36 206 Kč činí zástavu přes 1 milion Kč. Validátorem se tedy nemůže stát každý, kdo si pořídil výkonný stroj, jako tomu bylo v případě PoW, ale pouze ti, kteří mají dostatek prostředků, což se dá označit za velkou nevýhodu PoS. Tím pádem se ostatní uzly, které nedokáží zastavit takovou částku, nemohou stát validátory, a tak dostávat odměnu za uzamčení bloku.

Pokud by tento validátor blok upravil pro svůj prospěch, přijde jak o svou odměnu, tak případně i o svou zástavu. Validátoři se tedy mohou kontrolovat mezi

sebou jako tomu bylo i v případě PoW a i zde je velká motivace, aby validátor nepodváděl. Navíc odměna za validaci není zdaleka tak vysoká jako zástava, kterou validátor poskytl. [17]

4.3.5 Bitcoin vs. Ethereum

Blockchain je tedy velmi důležitým prvkem k provedení operace, která by v centralizovaném řešení vyžadovala nějakou centrální autoritu. Blockchain tohoto dosahuje právě pomocí kontraktů. Každá platforma, respektive kryptoměna má nějaké odlišnosti v tom, jak vlastní blockchain implementuje.

Velmi známou kryptoměnou je Bitcoin, který pro ověřování transakcí používá proof of work. Co se týče kontraktů jejich škálovatelnost je velmi malá a v podstatě slouží pouze pro schvalování transakcí. Bitcoin má tedy poměrně omezený skriptovací jazyk. [18]

Výhodou však je skutečnost, že čím méně toho kód v rámci kontaktu dokáže, tím více může být bezpečnější. Při vývoji mnoha funkcí mohou vývojáři snadno udělat chybu, která by mohla ovlivnit budoucí rozvoj sítě.

Na rozdíl od Bitcoinu, Ethereum se snaží, aby chytré kontrakty dokázaly více než být prostředkem pro platby digitální měnou. Ethereum je navrženo jako univerzální programovatelný blockchain, který provozuje virtuální stroj zvaný Ethereum Virtual Machine (EVM), jenž dokáže spouštět libovolně složitý kód. Oproti bitcoinovému blockchainu je ethereový Turingovsky kompletní, takže může fungovat jako počítač pro všeobecné použití. [18]

Ethereum ve verzi 1.0 používal pro schvalování transakcí též proof of work, avšak s příchodem nové verze 2.0 přešel na ekologičtější variantu proof of stake.

4.4 DApp na blockchainu

4.4.1 DEFI

Pojem decentralizované finance, označuje finanční aplikace postavené na blockchain technologii. Tyto aplikace poskytují různé finanční služby jako jsou půjčky, vkládání a výběr peněz, obchodování s kryptoměnami atd. DEFI nemají žádnou centrální autoritu a požadavky jsou zpracovány pomocí chytrých kontraktů na blockchainu. [19]

DEFI zastupuje obrovskou škálu různých aplikací. Tyto aplikace mohou být pak dále kategorizovány jako:

- *DEX (decentralizovaná burza)* – umožňuje obchodovat s kryptoměnami a digitálními aktivy
- *tokenizace aktiv* – uživatelé mohou vlastnit podíl v reálných aktivách
- *pojišťovny* – možnost uzavírat pojištění pro vlastní kryptoměny

Nejedná se však o konečný výčet. Těchto kategorií je více a mohou stále vznikat nové s ohledem na budoucí rozvoj této technologie. [19]

4.4.2 NFT

NFT neboli Non-Fungible Token, je digitální token reprezentující jedinečné digitální aktivum. Na rozdíl od fungibilních tokenů, což jsou kryptoměny, které jsou navzájem zaměnitelné (1 BTC = 1 BTC), reprezentují NFT nenahraditelné aktivum jako je například digitální umělecké dílo či unikátní sběratelskou kartu.

Informace o těchto tokenech jsou ukládány do blockchainu, kde jsou data o tom, kdo je jeho vlastníkem či jaké aktivum tento konkrétní token představuje. Díky jedinečnosti NFT je pak hodnota tohoto tokenu určována poptávkou na digitálním trhu. I z tohoto důvodu je NFT velmi oblíbeným v uměleckém světe, kde umělci prodávají svá digitální díla. [20]

5 Decentralizovaná uložiště souborů

5.1 BitTorent

BitTorrent je protokol starající se o sdílení souborů v síti peer-to-peer. Obecně řeší problém pomalého stahování z jednoho serveru skrze protokol HTTP, kdy servery sice mívají vysokou rychlost uploadu, ale potřebují obstarat mnoho uzlů najednou. To znamená, že při vyšší zátěži je upload pomalejší, a tedy i stahování uživatele. Další nevýhodou stahování z jednoho uzlu může být jeho vzdálenost od uživatele, což může mít též negativní dopad na rychlost stahování. [21]

BitTorrent tyto nedostatky řeší tak, že pokud je to možné, data stahuje od nejbližších uzlů. Každý soubor je ale na rozdíl od běžného stahování rozdělen do několika menších částí. Tyto části pak posílají stahujícím uzlům ostatní uzly v síti. Díky tomu lze najednou stahovat z několika uzlů (od každého jinou část) a dosahovat celkově velké přenosové rychlosti i přes to, že běžní uživatelé nemají tak vysokou rychlost nahrávání jako servery pro tento účel zřízené. [21]

DHT

BitTorrent aplikace velmi často vyžadují přítomnost centrálního uzlu, tzv. trackeru, který slouží pro výměnu informací mezi uzly, respektive vrací seznam uzlů, které mohou poskytnout daný soubor. Samotné stahování souboru ale probíhá pouze mezi uzly a centrální prvek již nijak nezasahuje. [21]

Existují však implementace protokolu, které využívají DHT (distributed hash table). To umožňuje vyhledávání dat mezi uzly bez potřeby centrálního uzlu. [21]

Části dat jsou identifkovány za použití hashovací funkce. Takto vzniklé identifikátory jsou rozmístěny do uzlů podle hodnoty hashe. Uzly, které obsahují daný hash, mají zodpovědnost za ukládání a vyhledávání dat s tímto identifikátorem. [21]

Ten, který pak data potřebuje, si pomocí hashovací funkce zjistí identifikátor a dotáže se uzlu, který je za data s tímto identifikátorem zodpovědný. Pokud tento uzel data nemá, hledá, dokud nenajde jiný, který tyto data obsahuje. [21, 22]

Existuje více implementací DHT a každá torrentová aplikace může využívat různých algoritmů. Velmi známým algoritmem pro DHT je například Kademlia. [22]

Výhody BitTorrentu

Výhodou protokolu je již zmíněná rychlost při stahování velkých souborů. Některé implementace dokonce podporují streamování videa. Torrenty používající DTH, které neběží na soukromém trackeru mohou mít velmi dobrou dostupnost i rychlost.

V případě použití BitTorrent protokolu u vlastních aplikací lze odlehčit souborovému serveru, z kterého se stane pouze jeden z mnoha uzlů a nebude komunikace záviset pouze na něm. Není tedy potřeba velká rychlost uploadu ani příliš výkonný server. Výhodou můžou být nižší pořizovací náklady na server, a to pouze v případě, kdy má tento server sloužit jako záloha, pokud není dostupný žádný jiný uzel. V opačném případě by nebyl server potřeba vůbec.

Nevýhody BitTorrentu

BitTorrent není příliš vhodný pro přenos malých souborů. Samotná inicializace stahování vzhledem k navozování spojení s více uzly nějaký čas trvá. Pro soubory jako fotografie a textové soubory není tedy příliš vhodný a samotné stahování spíše zpomaluje.

Nevýhodou může být i pomalé stahování nově přidaného souboru. Tento soubor se pak stahuje pouze z jednoho uzlu (který soubor přidal) a rychlost stahování je limitována rychlostí nahrávání tohoto uzlu.

V případě, že je konkrétní soubor dostupný pouze skrze soukromý tracker, který nevyužívá DHT a tento tracker přestane fungovat, nelze soubor stáhnout i navzdory skutečnosti, že je u jiných uzlů soubor dostupný, avšak bez trackeru neviditelný pro ostatní.

I když soubor může být uložen na více uzlech, není zárukou, že bude vždy dostupný. Některé uzly soubory pouze stáhnou a již dále neposkytují uzlům dalším. Případně mohou být všechny uzly mající tento soubor offline. Pokud je soubor starší,

může se stát, že jeho kopii již žádný uzel nemá a nebude jej tedy možné stáhnout. Tento scénář je ovšem možný i u standardního stahování z jednoho serveru, kdy daný server přestane soubor poskytovat. Tento problém je spíše k vidění u soukromých trackerů, které nemají mnohdy mnoho uživatelů, kteří by tento obsah udržovali, a tak méně stahované nebo starší soubory přestanou být postupně dostupné.

Pojem torrent se stal synonymem pro šíření nelegálního obsahu, jelikož jeho prostřednictvím dochází ke sdílení filmů a softwaru podléhajícímu autorskému právu. Jelikož obsah těchto souborů není uložen na jednom serveru, nelze za šíření nelegálního obsahu potrestat konkrétní osoby.

Nicméně, existují i společnosti, které svůj software sdílí takto dobrovolně, hlavně kvůli zmíněným výhodám tohoto protokolu. Legálně si lze například stáhnout různé linuxové distribuce, které jsou dostupné zdarma. [22]

5.2 IPFS

IPFS (InterPlanetary File System) je distribuovaný systém pro ukládání a sdílení souborů v rámci peer-to-peer sítě. Cílem IPFS je vytvoření jednotného globálního systému pro přístup k datům, který by dokázal nahradit současné centralizované webové služby. [21–23]

Klíčovou vlastností tohoto systému je právě sdílení souborů mezi uživateli, kdy je obsah stahován přímo od uživatelů, kteří soubory nahráli nebo si jej už jednou stáhli. Princip je tedy podobný BitTorrent protokolu, avšak IPFS výhradně využívá pro přístup k datům DHT [23].

Pomocí IPFS lze ukládat například statické webové stránky, videa, obrázky, dokumenty atd. Mezi vlastnosti IPFS patří odolnost vůči cenzuře a ověřování integrity dat. [21]

Soubory uložené na IPFS jsou chráněny pomocí hashů. Respektive hash slouží jako jedinečný identifikátor (CID – Content Identifier), který je vypočítán na základě obsahu souboru. Pro přístup k těmto souborům je tedy nutné znát jejich jedinečný identifikátor. [21, 23]

Pokud by tato ochrana nestačila, je samozřejmě možné soubory šifrovat pomocí různých algoritmů. Je však třeba myslet na to, že IPFS se stará hlavně o sdílení a ukládání dat. Bezpečnostní rizika si pak musí každý systém používající IPFS vyřešit sám podle své potřeby.

IPNS

Přistup k souborům na základě hashe, který slouží jako identifikátor a je vypočten podle obsahu souboru, generuje problém při úpravě dat. Při změně dat totiž dojde i ke změně hashe, takže by si uživatelé při úpravě souboru museli neustále posílat novou adresu, což není příliš efektivní.

Tento nedostatek má řešit právě IPNS (InterPlanetary Name System), který má na starosti aktualizaci adres na IPFS síti, takže umožňuje uživatelům udržovat trvalejší, a tím pádem přehlednější odkazy na soubory v síti IPFS. [21, 22, 24]

IPNS funguje na základě vytváření odkazů na soubory pomocí hashe, který si mohou vytvořit sami uživatelé. Tento hash je spojen s aktuální adresou CID souboru. Pokud je CID souboru změněno, IPNS zaktualizuje odkaz na novou správnou hodnotu CID.

Výhody IPFS

Mezi velkou výhodu IPFS patří zajištění snadného přístupu k souborům pomocí jedinečných odkazů (CID), čímž se nemusí řešit problémy s duplicitními názvy, případně s duplicitními adresami. Jelikož hash obecně vypadá jako dlouhá směs náhodných znaků, lze říci, že ten, kdo nebude figurovat odkazem na tento soubor, k němu nedokáže získat přístup.

Navíc ostatní technologie, které jsou zde rozebrány se starají hlavně o distribuci dat. Nedokážou pracovat se soubory bez toho, aniž by se musely ukládat binárně do databází, což není moc praktické právě kvůli velikost těchto databází. Tyto technologie pro ukládání dat je tedy vhodné kombinovat právě s IPFS, kdy se do databáze ukládá pouze adresa na soubory.

Nevýhody IPFS

Jak již bylo zmíněno, IPFS se příliš nezabývá bezpečností ukládání dat. I když šifrování souborů podporuje za použití privátních klíčů, musí si ho každý uživatel zajistit převážně sám v závislosti na jeho potřebách.

Při stahování velkých souborů může být rychlost stahování poměrně pomalá, obzvláště jsou-li uloženy na mnoha různých uzlech. Soubor totiž není rozložen na více částí, jako tomu je v BitTorrent protokolu, a proto se celý stahuje z jednoho uzlu.

I přes to, že je IPFS navržen tak, aby byl odolný proti výpadkům, nelze zajistit dostupnost souborů. Ta je dána tím, zda jsou uzly, na kterých jsou tyto soubory uloženy, dostupné. V případě, že by byl soubor uložen na malém množství uzlů, mohlo by se snadno stát, že nemusí být vždy k dispozici. Tento problém řeší Filecoin, o němž se píše v následující kapitole.

5.3 Filecoin

FileCoin je decentralizovaná síť pro uložiště souborů, která využívá IPFS a blockchain. Blockchain je používán pro evidenci transakcí a IPFS pro ukládání souborů. Cílem Filecoinu je vytvořit trh s uložištěm, kde uživatelé mohou nabízet a poptávat úložné kapacity. FileCoin jakožto kryptoměna (token) je pak odměnou pro uživatele nabízející úložný prostor, což celkově slouží jako motivace pro obětování svého uložiště ostatním uživatelům. [21]

Výhody Filecoinu

Tato síť může díky motivaci uživatelů poskytnout vlastní kapacity, vyřešit problém s nedostupností souborů v rámci IPFS. Díky finančnímu ohodnocení je motivace dostupnosti uložiště poskytovatele dostatečně velká, aby tuto dostupnost neomezoval například vypnutím svého PC nebo serveru.

Filecoin se navíc stará i o bezpečnost a šifrování dat, které samotné IPFS tolik neřeší.

Nevýhody Filecoinu

Co je pro poskytovatele úložného prostoru výhodou, může být nevýhodou pro poptávajícího. Aby byla zajištěna dostupnost a dostatečná kapacita uložiště, je třeba, aby poptávající motivoval poskytovatele finančně. Tato služba tedy není zdarma, což lze odvodit i z toho, že síť využívá blockchain.

Nicméně, z tohoto důvodu síť vznikla, a i v případě zřizovaní centralizovaných systémů, je třeba platit, buď za hostingové / cloudové služby nebo za nákup a udržování vlastního serveru.

Je však důležité zmínit cenu. Jelikož se jedná o kryptoměnu (FIL), nedá se vyjádřit přesnou částkou. Stejně tak záleží na typech a velikosti ukládaných souborů. Avšak z dat za rok 2021 lze odvodit, že cena za gigabajt uložených dat se pohybovala od 0,03 do 0,07 USD za měsíc, což je nesrovnatelné s cloudovými službami, kde se částky za měsíc úložného prostoru pohybují v úplně jiných řádech. Navíc v případě cloudových či hostingových služeb je třeba zvolit vhodný tarif na základě potřeby velikosti uložiště. V síti Filecoin se platí za aktuálně používané uložiště, což je rozhodně flexibilnější a efektivnější přístup. Na druhou stranu, Filecoin pouze ukládá soubory anelze na něm spouštět aplikace jako v případě cloudových služeb.

5.4 Storj

Storj je decentralizovaná síť pro ukládání souborů. Cílem je motivovat uživatelé k poskytnutí svého nevyužitého uložiště a zároveň zajistit bezpečné a spolehlivé ukládání dat v rámci peer-to-peer sítě. [25]

Storj využívá blockchain a pro šifrování metodu AES. Data, která mají být nahrána do sítě, jsou ještě před tím zašifrována a části těchto zašifrovaných dat jsou uloženy na různých uzlech v síti. Data jsou tedy chráněna před neoprávněným přístupem. [25]

Díky tomu, že jsou data rozdělena na části a každá tato část se nachází na více uzlech, jsou data velmi dostupná i v případě výpadku některých uzlů. Navíc lze části stahovat ze všech uzlů najednou, čímž je zajištěna velká rychlost stahování, stejně jako u protokolu BitTorrent.

Uživatelé jsou motivování formou tokenu Storjcoin X (SJCX), který lze získat za poskytování svého úložného prostoru. Blockchain navíc slouží pro zjišťování, zda nebylo s daty manipulováno a pro ukládání záznamů o smlouvách ohledně poskytování uložiště.

Výhody Storj

Díky redundanci dat se jedná o velmi spolehlivou síť, jelikož se snižuje závislost mezi jednotlivými uzly a riziko ztráty dat. Storj se automaticky stará o šifrování dat ještě před jejich nahráním do sítě, což z něj dělá poměrně bezpečnou síť. Data jsou rozložena na různé časti, které jsou rozmístěny mezi uzly. To rovněž zvyšuje jeho bezpečnost, jelikož potencionální útočník, který má fyzicky přístup k těmto datům, nebude mít data kompletní. Storj je navíc poměrně snadné na konfiguraci.

Nevýhody Storj

Jelikož využívá blockchain, jedná se opět o placenou síť. Jak však bylo zmíněno u jiných technologií, nákladům se nelze vyhnout ani v centralizované síti.

Storj není příliš velká síť a má velmi málo uživatelů poskytujících uložiště. To se může postupem času změnit, avšak momentálně může mít tato skutečnost negativní vliv na rychlost sítě a celkově na její dostupnost. Soubor má omezenou maximální velikost, což nemusí být v některých případech žádoucí.

5.5 Sia

Sia je decentralizovaná síť pro sdílení a ukládání souborů. Soubory jsou ukládány na síť za pomoci kryptografických klíčů a blockchainu. [24] Pro motivaci uživatelů poskytnout vlastní úložný prostor slouží kryptoměna Siacoin.

Sia data rozděluje na menší segmenty, které následně zašifruje a šíří do uzlů po celé síti. Tento způsob zajišťuje opět vyšší úroveň bezpečnosti a zároveň slouží jako prevence před ztrátou dat. [21]

Výhody Sia

I tato síť je velmi bezpečnou díky šifrování a segmentaci dat na menší části. Redundance dat navíc poskytuje lepší dostupnost těchto dat a uzly na sebe nejsou navzájem příliš závislé.

Nevýhody Sia

Sia je stále poměrně malá síť, což může mít negativní dopad na dostupnost dostatečně velkého úložného prostoru pro ukládání větších souborů.

Opět pro potřeby vyšší bezpečnosti a zápisu transakcí využívá blockchain, a stejně tak i tokeny pro odměňování uživatelů za poskytování uložiště. Stejně jako předchozí technologie není zdarma.

Oproti Storj je Sia náročnější na konfiguraci, která je spíše vhodná pro technicky zdatnější uživatelé se zkušenostmi s kryptografií.

6 Decentralizovaná uložiště dat

6.1 Úvod do problematiky

CAP teorém

CAP teorém popisuje, že u distribuovaných systémů, nelze mít zároveň data plně konzistentní (consistency), dostupná (availability) a odolná vůči výpadkům sítě (partition tolerance). Vždy jde vyhovět pouze dvěma z těchto tří vlastností. [26]

Konzistence si dává za cíl mít na všech uzlech stejná data. Dostupnost pak, že data v případě potřeby musí být stále dostupná a odolnost vůči výpadku, že systém musí být schopný provozu i v případě výpadku nebo zpoždění komunikace. [26]

V praxi to znamená, že je nutné rozhodnout, které vlastnosti jsou důležitější. Například pokud je vyžadována hlavně dostupnost, může to být na úkor konzistence dat. Data tedy budou dostupná, ale každý uzel může mít data mírně odlišná. Pokud například bude důležitější konzistence, může dojít ke zpoždění v dostupnosti, kdy se musí uzly neustále synchronizovat, aby měly správná data.

Tento problém se tedy týká všech těchto distribuovaných technologií a každá technologie může mít jinak nastavené priority v závislosti na tom, jak je navržena.

CRDTS

CRDTS neboli Conflict-free Replicated Data Types jsou datové struktury, které jsou využívány pro synchronizaci dat mezi uzly v distribuovaných sítích. Tyto struktury nabízí rychlou a efektivní synchronizaci dat a zároveň umožňují minimalizovat konflikty mezi daty uloženými na uzlech. [27]

Pokud jsou na různých uzlech upravena stejná data, musí být výsledná hodnota dat na všech uzlech stejná. Strategií, jak řešit konflikty, existuje více a vždy záleží na konkrétní implementaci. Někdy mohou být změny spojeny (*merge*), někdy založeny na prioritě (*priority-based*).. Nebo naopak existují strategie jako *last-write-wins* či *first-write-wins*, kdy se propíše ta změna, která byla jako první / poslední. [27] Tyto strategie však vedou k tomu, že některé změny jsou vždy

zahozeny, což nemusí být vždy úplně žádoucí. Záleží tedy na konkrétním aplikaci, zda potřebuje uchovávat všechny změny, nebo jsou pro ně důležité jen některé.

Tyto datové struktury níže zmíněné technologie využívají, například Gun JS, OrbitDB a ThreadDB.

6.2 Ethereum

Co se týče blockchainu, nejlepší volbou pro vývoj decentralizované aplikace, je jednoznačně Ethereum díky turningovské kompletnosti jazyka Solidity pro vytváření chytrých kontraktů. Díky tomu není omezen pouze na schvalování obyčejných transakcí, ale dokáže vykonávat libovolný kód. [18]

Solidity

Solidity je programovací jazyk určený pro vytváření chytrých kontraktů. Má syntaxi podobnou JavaScriptu. Podporuje statické typy, využívání knihoven a vytváření uživatelsky definovaných typů. Je zkompilován do Ethereum Virtual Machine (EVM), což ho činí spustitelným na všech uzlech Ethereum sítě. Nejedná se o jediný jazyk pro psaní chytrých kontraktů v rámci sítě Ethereum, ale je komunitou nejpoužívanější, hlavně díky svým rozsáhlým možnostem. Tento jazyk je založen na IFTTT logice, tzn. IF-THIS-THEN-THAT. [28]

EVM je běhové prostředí s omezenými oprávněními. Díky těmto omezením nemá přístup k okolnímu prostředí jako je síť, systém souborů nebo jiné procesy. [28]

Plyn a odměna

Plyn je využíván k měření počtu kroků, které bude transakce na EVM vyžadovat. Princip je velmi podobný jako u řízení procesů ve spojitosti s časem CPU. Čím je transakce složitější, tím spotřebuje více výpočetních zdrojů a tím je větší konečná cena za spotřebovaný plyn. Každý kód tedy vyžaduje nějaký poplatek za spotřebu tohoto virtuálního plynu. [28]

Tento systém motivuje těžaře k provádění transakcí a chytrých smluv za využití vlastního času a zdrojů. Uživatel může prioritizovat svojí transakci tím, že bude ochoten zaplatit vyšší cenu za plyn. Díky vyšší ceně bude uzel motivován vyšší odměnou k rychlejšímu vyřízení transakce. Obdobně může uživatel nastavit i maximální hranici poplatku, riskuje však, že uzel nebude motivován k vyřízení transakce, nebo toto maximum nebude na vyřízení transakce ani stačit. Pokud by plyn během vykonávání transakce došel, bude vyvolána výjimka a veškeré transakcí provedené změny, budou vráceny. Uzel však vykonal práci na provedení transakce, a tudíž mu odměna zůstane a tvůrce transakce tak o své prostředky přijde. Pokud této maximální hranice nebylo dosaženo, je přeplatek vrácen zpět uživateli. [28]

Tento způsob je i motivací pro nevytváření škodlivého kódu, který u Turingovsky kompletního stroje generuje problém zastavení, kdy nelze určit, zda program přestane běžet, či zda bude běžet navždy. Kód by tak mohl skončit zacyklením, což by v rámci této sítě mohlo být bráno jako DDoS útok. Útočník by tak přišel o své prostředky a transakce by skončila po vyčerpání plynu. [28]

V Ethereum síti se ale platí tokenem (kryptoměnou) ETH neboli éterem. Proč tedy existuje plyn a nepočítá se poplatek rovnou v ETH? Důvodem je kolísavost hodnoty ETH. Plyn zajišťuje, že každý stejně náročný operační kód bude mít v průběhu času stejnou hodnotu. Pokud by se tedy místo plynu používal rovnou éter, kód transakce, který by se vykonal například o pár dní dříve, by mohl mít výraznou změnu v ceně. Cenu za plyn pak nastavuje uživatel a odměna za vyřízení transakce se na základě výpočtu transakčního poplatku platí těžaři v ETH. [28]

Výhody Ethereum

Blockchain je zatím bezkonkurenčně nejlepší pro vyřizování požadavků v rámci peer-to-peer sítě, která není postavena na důvěře. Konkrétně Ethereum blockchain je využíván i dalšími technologiemi právě díky škálovatelnosti jeho chytrých kontraktů.

Nejvhodnější je jeho použití právě na DEFI aplikace, kde se vzhledem k zaměření těchto aplikací s poplatky počítá. Uživatelskou výhodou je hlavně absence centrální autority.

Nevýhody Ethereum

Jak bylo v kapitolách výše již několikrát zmíněno, blockchain obecně funguje na motivaci zisku ze schvalování transakcí. To znamená, že uživatelé, kteří by chtěli nové transakce přidávat, například v rámci aplikace poslat někomu zprávu nebo přidat někam příspěvek, bude ho to stát vždy nějaká étera. Uživatel by byl nucen nakoupit si kryptoměnu, aby mohl aplikaci používat. Bohužel se jedná o nemalé poplatky. Pokud by šlo o nějakou chatovací aplikaci, musel by uživatel posílat zprávy velmi rozvážně a používání SMS by bylo mnohem levnější. V průměru se schválení transakce v rámci Ethereum sítě může vyšplhat na 3–6 dolarů (ke dni 28. 2. 2023) v závislosti na kurzu kryptoměny ETH a v závislosti na aktuální vytíženosti sítě.

6.3 Gun JS

GUN je malý, jednoduchý a rychlý protokol sloužící k synchronizaci dat. V podstatě se jedná o nástroj pro tvorbu decentralizovaných databází v peer-to-peer síti. Jeho výhodou je, že data lze ukládat v libovolné formě v závislosti na potřebě vývojáře. Jedná se o grafovou databázi, která umí ukládat data jako klíč – hodnota, relačně, nebo dokumenty podobné JSON. Umožňuje ukládat i soubory, živě streamovat video a pracovat s relačními a hypergrafickými daty. [29]

Pro synchronizaci a řešení konfliktů využívá systému CRDT. Gun se vyznačuje i velkou dostupností dat, dokonce i v offline režimu.

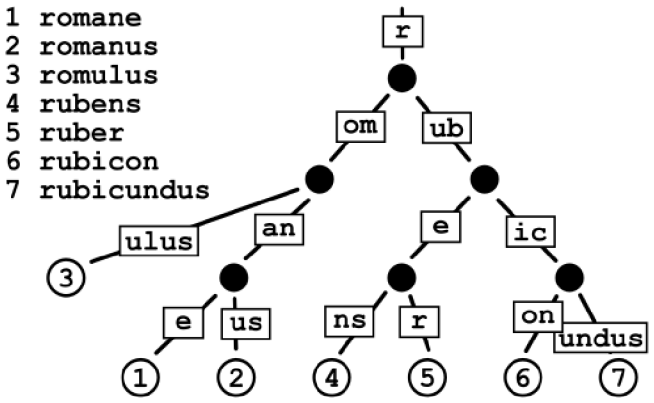
Gun JS rozšiřují další API nástroje. Zřejmě nejpoužívanějším je SEA API (Security, Encryption & Authorization), které se používá pro zabezpečení. Gun obecně používá tři prostory pro ukládání dat a to veřejný, uživatelský a neměnný (frozen). Tyto prostory pak fungují v závislosti na tom, zda je využíváno SEA. Jak fungují tyto oprávnění lze vidět na obrázku č. 4. [29]

	WITHOUT SEA	WITH SEA	EXAMPLE
PUBLIC SPACE	-everybody can write -everybody can read -everybody can see content	-everybody can overwrite -everybody can read -only user with the right key can see content	Wiki
USER SPACE	-only owner can write -everybody can read -everybody can see	-only owner can write -everybody can read -only user with the right key can see content	Profile
FROZEN SPACE	-one time write only -everybody can read -everybody can see	-one time write only -everybody can read -only user with the right key can see content	

Obrázek 4 Oprávnění uživatelů v jednotlivých prostorech s a bez SEA
Zdroj: [29]

SEA API nabízí hlavně funkce pro šifrování, dešifrování a podepisování obsahu. Nejvíce je využíváno právě pro práci s user API, kdy nabízí uživatelům možnost svá data šifrovat. Implementuje dokonce i funkce jako sdílení svých dat jiným uživatelům, případně umožňuje zpřístupnění části své databáze uživatelům, kterým byl vystaven certifikát. [29]

GUN JS má k dispozici i několik adaptérů pro ukládání dat. Defaultním je radix, který ukládá data do lokálního uložiště prohlížeče za použití radix stromu, jenž ukládá data s ohledem na optimalizaci pro úsporu místa, viz obrázek 5. Dalšími adaptéry jsou pak IPFS, SQLite, MongoDB, MySQL, ElasticSearch, atd. Při použití těchto adapterů, ale nejsou data ukládána mezi uzly, ale centralizovaně. [29]



Obrázek 5 Radix strom
Zdroj: [30]

Pro zavedení GUN aplikace je zapotřebí existence tzv. superuzlů, které mají za úkol pomáhat najít novým uzlům ostatní uzly v síti. Nemají však přístup k datům v rámci GUN sítě a pouze pomáhají při vytváření sítě nové. Když je síť dostatečně velká, je možné tyto uzly odstranit bez ohrožení chodu sítě. Gun podporuje i tzv. relay uzly, které slouží jako prostředník mezi uzly, kteří nemohou přímo komunikovat. Tyto uzly mají za úkol překonávat překážky jako firewall či NAT a jsou vytvářeny jako websockety. [29]

Výhody GUN

GUN je založen na důvěře mezi uzly, ale snahou autorů tohoto projektu, je data co nejlépe zabezpečit pomocí nejmodernějších šifrovacích algoritmů, avšak implementace těchto funkcí závisí hlavně na vývojáři. Další výhodou v síti s důvěrou je její rychlost a flexibilita. Není totiž nutné o každém zápisu hlasovat, nebo si ověřovat data od dalších uzlů, což může aplikace v nedůvěryhodné síti zpomalovat.

Výhodou je určitě i přítomnost oddělených prostředí pro ukládání dat, kdy jedno je vhodné pro veřejná data, další pro privátní a poslední pro data neměnná. Velmi přínosné je, že dokáže pracovat i v offline režimu.

Nevýhody GUN

Nevýhodou této technologie může být, že se jedná o síť s důvěrou mezi uzly. To je nežádoucí v případě, že této důvěry bude chtít někdo zneužít a síť ohrozit. Proto není tato technologie vhodná pro práci s bankovními nebo zdravotními daty, kde je nutná maximální ochrana. Nicméně GUN nabízí spoustu funkcí jako digitální podpisy nebo šifrování, díky kterým lze zajistit, ověření, od koho data pochází a nemožnost nepovolaným uživatelům číst obsah v čitelné podobně nebo jej měnit.

Za nevýhodu lze i považovat potřebu bootsrap či relay serverů. To však není chybou GUN JS, ale tím, jak pracují internetové protokoly, které nejsou na přímou komunikace mezi uzly přizpůsobené.

6.4 OrbitDB

OrbitDB je open-source databáze, která je založená na IPFS umožňující ukládání a synchronizaci dat v peer-to-peer síti. Využívá konceptu CRDT jako GUN pro řešení konfliktů, které mohou nastat v situacích, kdy přistupuje ke stejným datům více uzlů současně. Každý uzel v této síti má vlastní instanci OrbitDB, která je synchronizována s ostatními uzly pomocí IPFS. [31]

Data umožnuje ukládat v několika typech databází, například klíč-hodnota, dokumentové databáze a časové řady. [31]

Velmi zajímavé je v OrbitDB i zabezpečení. Pokud je potřeba nějaká data sdílet, lze vytvořit tzv. access controller řídící přístup k těmto datům. Jedná se o speciální objekt, který je součástí každého záznamu, a díky tomu, lze pro různé uživatele vytvářet různé úrovně přístupu k datům. Lze například omezovat zápis nebo čtení i na základě dalších kritérií, jako je typ transakce nebo čas. [31]

Výhody OrbitDB

Výhodou OrbitDB je její funkčnost fungovat offline. Toho je docíleno držením kopie OrbitDB na každém uzlu. Uživatelé tedy mohou pracovat s daty, i když nemají přístup k internetu. Je poměrně snadná na implementaci a nabízí více typů databází, s kterými umí pracovat. Mezi výhody určitě patří i zmíněný access controller, s jehož pomocí lze vyřešit přístupová práva různých uživatelů k různým datům.

Nevýhody OrbitDB

Nevýhodou může být pomalejší rychlost přenosu dat, jelikož každý uzel musí synchronizovat kopii dat s ostatními uzly v síti. I přesto, že podporuje více typů databází, neumožňuje vytvářet SQL databáze, které umožňují lepší dotazování nad daty.

Tento projekt je stále ve fázi vývoje, a tudíž nemusí být všechny funkce, které momentálně nabízí, v dalších verzích dostupné nebo mohou být pozměněny. [31]

6.5 ThreadDB

ThreadDB je též decentralizovaná open-source databáze, která se stará o ukládání a správu dat v decentralizovaných aplikacích. Dokáže data ukládat, indexovat a vyhledávat díky rozsáhlému dotazovacímu jazyku Query API. Dotazy lze provádět podobně jako pomocí SQL, avšak s tím rozdílem, že data jsou uložena jako objety JSON. ThreadDB tedy spojuje výhody práce s SQL databázemi a objektovými databázemi. [32]

Výhody ThreadDB

Velkou výhodou ThreadDB je především indexace a možnost dotazování nad daty. Je možné zajistit integraci s IPFS a další výhodou je skutečnost, že využívá CRDTs pro řešení konfliktů a zajištění konzistence dat. [32]

Nevýhody ThreadDB

I přes to, že je založen na CRDTs, může nastat situace, že bude některé konflikty třeba řešit manuálně. Práce s ThreadDB je spíše vhodná pro technicky zdatné uživatele, kteří mají alespoň nějaké základní znalosti programování a databázových technologií.

Jedná se o poměrně novou technologii, která je stále vyvíjena, takže se mohou objevovat chyby, které budou opraveny až později.

6.6 BigchainDB

BigchainDB je distribuovaný open-source databázový systém, který spojuje funkce blockchainu a tradičních databází. Snahou BigchainDB je vytvořit rychlou, bezpečnou a velmi škálovatelnou platformu, a to hlavně po správu digitálních aktiv. [33]

Data jsou ukládána přímo do bloků, které obsahují transakce. Bloky jsou pak obdobně jako v blockchainu ověřovány pomocí kryptografických

funkcí, které zajišťují autentičnost a integritu těchto dat. BigChainDB dokonce umožňuje ukládat různé typy dat jako obrázky či zvukové soubory. [33]

Výhody BigChainDB

I přes to, že je částečně založen na principu blockchainu, je navržen tak, aby zpracovával velké množství dat rychle a efektivně. Data uložená do BigchainDB jsou obdobně jako v blockchainu chráněna před neoprávněnou manipulací, a to i díky jejich nemožnosti je změnit.

Mezi výhody určitě patří i možnost ukládat různé typy dat.

Nevýhody BigChainDB

BigchainDB je navržen hlavně pro ukládání digitálních aktiv a transakcí, což nemusí být pro některé aplikace nebo druhy dat vhodné. Obdobné to je i pro neměnnost dat, kdy některé aplikace vyžadují, aby data bylo možné měnit.

Data jako taková jsou ukládána decentralizovaně, avšak BigchainDB využívá tzv. validátory, kteří transakce ověřují a pro ověření musí být transakce schváleny a podepsány většinou validátorů. Ti však nefungují stejně jako v případě blockchainu a může se stát, že je většina validátorů ovládána jedním subjektem. Kvůli tomu lze BigchainDB považovat za částečně centralizovaný.

7 Zabezpečení a kryptografie

Protože jsou v případě distribuovaných systémů data ukládána na zařízení uživatelů, jedinou možností, jak citlivá data chránit, je jejich šifrování. V případě blockchainových technologií je další přidanou hodnotou řešení nedůvěry mezi uzly a zajištění integrity dat, díky nemožnosti tyto data měnit, což má z hlediska zabezpečení též významnou roli.

Některá decentralizovaná uložiště souborů se starají o zabezpečení dat samy (FileCoin, Storj, Sia), jiné nechávají tento problém na vývojářích (IPFS, BitTorrent).

Obdobné to je i s decentralizovanými uložišti dat, avšak s rozdílem, že technologie využívající blockchain nejsou primárně určeny pro ukládání citlivých dat, a proto se touto problematikou příliš nezabývají. Zbylé technologie Gun JS, OrbitDB a ThreadDB pak nabízejí různé funkcionality, kterými lze data zabezpečit, většinou za použití asymetrického a symetrického šifrování. Data lze tedy zašifrovat, aby k nim měli přístup pouze konkrétní uživatelé a i podepsat pro účely ověření autora obsahu. Dokonce nabízí i další pokročilejší funkce, kdy lze určit, kam má mít konkrétní uživatel přístup nebo jaké má mít oprávnění [29, 31, 32].

Obecně, ať už se jedná o soubory či data, používají tyto technologie běžné zabezpečovací standardy. Například pro vytváření hashů nejčastěji používají šifrovací algoritmus SHA-256, případně SHA-3. Pro symetrické šifrování bývají využívány různé implementace algoritmu AES-256 a pro asymetrické je nejčastěji využíváno šifrovací algoritmus ECDSA s odlišnými typy eliptických křivek. [18, 29, 31, 32]

Dá se říci, že všechny tyto technologie obsahují formu nějakého zabezpečení, avšak vždy bude záležet hlavně na tom, s jakými daty se pracuje a jak vývojáři s jejich zabezpečením naloží. Z hlediska zabezpečení jsou použitelné pro ukládání citlivějších informací uživatelů. Je však potřeba poznamenat, že ne vždy lze pro libovolný systém použít libovolnou technologii. Například aplikace starající se o finance by rozhodně měly figurovat v nedůvěryhodné síti, která umí například řešit problém dvojího utracení. Naopak u aplikací, kde je práce s daty citlivým tématem i v centralizovaných systémech, což jsou například systémy pracující

se zdravotními daty pacientů nebo energetické systémy, není vhodné používat decentralizované technologie vůbec.

Centralizované technologie mají tu výhodu, že ukládají data na zabezpečené servery, ke kterým má mnohdy fyzický přístup jen několik málo pověřených osob. To může dělat ukládání dat o něco bezpečnější než u decentralizovaných technologií. Nevýhodou však může být, že v případě úniku těchto dat, nemusí být dostatečně zabezpečená. Většinou jsou chráněny hlavně hesla uživatelů, u kterých bývá ukládán pouze jejich otisk (hash), a pokud není aplikována nějaká forma anonymizace těchto dat, mohou se útočnici dostat i k citlivým informacím uživatelů.

V případě, kdy jsou data distribuovaná, se počítá s tím, že jsou fyzicky přístupná i neoprávněným osobám, a tudíž se citlivější data rovnou zabezpečí formou šifrování. U decentralizovaných technologií je možné šifrovat většinu ukládaných dat, jelikož dešifrování probíhá na zařízeních uživatelů, a tím je rozložena zátěž, která by v případě centralizovaných technologií mohla být obrovská, jelikož by se o dešifrování staraly pouze servery poskytovatele. V případě centrální architektury je tedy nutné šifrovat s rozmyslem, aby se zbytečně nezatěžovaly servery, avšak tak, aby byla data dostatečně zabezpečená i v případě úniku.

Není však vyloučeno tyto přístupy kombinovat a na server poskytovatele ukládat zašifrovaná data, která si uživatel dešifruje sám až na svém zařízení. Takto může například docházet k výměně dat mezi uživateli. Data jsou zabezpečena formou end-to-end šifrování, kdy na server chodí data v nečitelném stavu a je možné je dešifrovat jen na straně uživatele.

8 Praktická část

Pro praktickou část budou použity technologie GUN JS a IPFS. Volba těchto technologií vyplývá ze zadání, kdy cílem je použít technologie, které budou co nejsnáze ovladatelné pro uživatele. Většina ostatních technologií využívá blockchain, který bývá zatížen poplatky a některé z nich vyžadují vytvoření kryptopeněženky, což není uživatelsky přívětivé.

Právě tento důvod vedl k tomu, proč bude v praktické části použit GUN JS. Navíc použití blockchainu, je potřeba hlavně tam, kde je nutnost zajistit integritu dat, případně zajistit důvěru v nedůvěryhodné síti. Pokud nic z toho není vyžadováno, stačí se spolehnout na technologie, kde není důvěra zcela nutná.

Při tvorbě aplikací, které jsou uživatelsky nejzajímavější, což jsou zejména aplikace se sociálním zaměřením, stačí zajistit data formou šifrování, aby nebyla čitelná pro ostatní uživatele a digitálním podpisem pro potřeby ověření původce dat. Toto vše nabízí právě GUN JS.

8.1 Prvotní inicializace

Vytvoření vlastního GUN relay serveru

GUN JS ukládá data distribuovaně, avšak k tomu, aby se uzly mezi sebou dokázaly propojit potřebují tzv. superuzly. Z historických důvodů nelze totiž spojit uzly přímo mezi sebou, zvláště pokud nemají veřejnou IP adresu, nebo IPv6 adresu. Pro komunikaci mezi uzly je využíváno Web RTC, které používá veřejné signalizační servery starající se o koordinaci komunikace mezi uzly, kde není možné vzájemné propojení (nemají IPV6 adresu, jsou chráněny firewallem, nebo jejich IPv4 adresa není veřejná). [29]

Toto lze označit zatím za nevýhodu, jelikož potřeba superuzlů může působit centralizovaně. Autoři však slibují nápravu formou vývoje vlastního protokolu AXE (Advanced eXchange Equation).

K vytvoření superuzlů lze použít Node.js, který umožňuje psát javascriptový kód na backendu. Ukázka níže používá express, což je backendový framework právě pro Node.js

```
    const express = require('express');
    const Gun = require('gun');
    const app = express();
    const port = 3030;

    app.use(Gun.serve);

    const server = app.listen(port, () => {
        console.log(`App is listening at
http://localhost:${port}`);
    })

    Gun({web: server});
```

Ukázka 1 Vytvoření superuzlu v Node.js
Zdroj: autor

Inicializace GUN

Před použitím GUN je nejdříve nutné naimportovat knihovny. Lze využít buď CDN, nebo knihovny stáhnout a naimportovat z lokálního zdroje.

```
<head>
<script
src="https://cdn.jsdelivr.net/npm/gun/examples/jquery.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/gun/gun.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/gun/sea.js"></script>
</head>
```

Ukázka 2 Import důležitých knihoven
Zdroj: autor

Po importu se vytvoří instance do proměnných potřebných v rámci všech ukázek, viz ukázka 3. Proměnná peers obsahuje seznam superuzlů a relay uzlů, které se starají o komunikaci mezi klienty. V tomto případě bude uzlem lokální superuzel, který byl vytvořen v ukázce 1. Následně dojde k vytvoření instance Gun, kde lze v parametru předat seznam superuzlů a relay uzlů. Poté stačí inicializovat sea pro

práci se šifrováním a user pro práci s uživatelským účtem. Veškerý kód se umístí do asynchronního bloku.

```
<script>
    const peers = ['http://localhost:3030/gun'];
    const gun = new Gun(peers);
    const sea = Gun.SEA;
    const user = gun.user();
;(async() => {
    //v asynchorním bloku bude veškerý kód používaný v ukázkách
})();
</script>
```

Ukázka 3 Inicializace GUN JS
Zdroj: autor

8.2 Uživatelé

GUN JS má již v základu implementované funkce pro práci s uživateli. Nejzákladnější funkce slouží pro vytvoření, autentizaci a odhlášení uživatele.

Vytvoření uživatele

Vytvoření uživatele je velmi jednoduché. Vstupem do metody *create()* je alias (uživatelské jméno) a heslo. Dále je možné přidat libovolnou callback funkci, která je zavolána po vytvoření uživatele a option object s jehož pomocí lze definovat další dodatečné možnosti.

Nevýhodou je nemožnost vynutit jedinečnost aliasu z důvodu, že databáze dokáže fungovat i offline. Ověřit existenci uživatele s konkrétním aliasem možné je, ale v případě, že je uzel offline a nemá aktuální data, bude mít ověření nekorektní výsledek.

Autentizace a odhlášení

Možnosti autentizace uživatele jsou dvě. O autentizaci se stará funkce *auth()*, která přijímá alias uživatele, jeho heslo a případně opět callback a option objekt. Tato metoda je přetížená, takže alias a heslo lze nahradit použitím klíčů uživatele.

Zmíněné klíče je možné získat po prvotním vytvoření, případně po přihlášení uživatele. Přihlášení pomocí klíčů se může využít například při zapomenutí hesla.

V decentralizované síti je obnova hesla poněkud složitá, někdy až nemožná. Gun JS nabízí několik možností, jak svůj účet obnovit, například pomocí třech přátel (které si musel uživatel předem vybrat), případně nastavením bezpečnostních otázek. Je však možno nechat uživatele jeho klíče zálohovat a pokud zapomene heslo, může se přihlásit právě pomocí těchto klíčů. Je však potřeba zmínit, že zálohované klíče je vhodné též zašifrovat pomocí uživatelem zvoleného hesla, aby nezískal přístup k účtu každý, kdo dokáže získat uživatelovy klíče (například neopatrným zacházením s klíči).

Pro odhlášení lze použít metodu *leave()*. Ta nepřijímá žádné parametry a vyžaduje, aby byl uživatel přihlášen. Tato metoda bohužel neumožňuje přidat callback pro zpracování výsledku odhlášení. Je tedy nutné po zavolání metody zkontrolovat, zda opravdu došlo k odhlášení, například kontrolou přítomností klíčů v proměnné uživatele.

Další metody pro práci s uživatelem

Existuje několik dalších metod či vlastností pro práci s uživateli. Pro kontrolu, zda je uživatel přihlášen se použije vlastnost *is*. V případě, že není žádný uživatel autentizován, je jejím výsledem „undefined“.

Velmi užitečnou metodou může být i *recall()*, kde lze například nastavit, aby byl uživatel přihlášen po celou platnost sezení. Dokud má tedy uživatel otevřenou v prohlížeči nějakou záložku, kde je přihlášen, zůstane přihlášen i ve všech ostatních.

```
//vytvoření uživatele s aliasem alice, heslem password123 a vypsáním
výsledku vytvoření do konzole
const alice = user.create("alice", "password123", console.log);

//přihlášení uživatele alice, vypsání výsledků a klíčů uživatele do
konzole
const  aUser = user.auth("alice", "password123", (ack) => {
    console.log(ack);
    console.log(JSON.stringify(ack.sea));
})

//získané klíče, lze přeparsovat zpět do JSON a prihlásit se
const  pair = JSON.parse('{"pub":"lb4Dbv1PtUnn-
rUcbafIS90M4dLSPCGzrqkBzr9kBGQ.D0taZkYkQYjH-n8tfQTSHJqczu-JacA-_-
ed1B3oxOU","epub":"Brq5JZhgMWhhKI5yenrDTCUubTV_r-cXE5ROwJM71e4.0P8QW-
QyK6CrdeHuAfUlzZHzGKjJsBDTuzZeYnHIYoE","priv":"ZLF4Dx4lw0jbr3Yp2JBvcppyc
sru08WvAvMsCspK6hg","epriv":"u_h4kk935i_7YWFXE-
u2xDzUcbn2QBaAETOHSfwaxgg"}');

//přihlášení uživatele pomocí klíčů a vypsání informací o účtu
const  bUser = user.auth(pair, console.log);

//odhlášení uživatele
bUser.leave();

//kontrola, zda je uživatel přihlášen
if(bUser.is){}

//uchování přihlášení v session storage
bUser.recall({sessionStorage: true})
```

Ukázka 4 Práce s uživateli v GUN JS
Zdroj: autor

8.3 Ukládání dat

Jak bylo zmíněno v teoretické části, Gun JS definuje tři prostory. Uživatelský, veřejný a neměnný. Do svého uživatelského prostoru smí data ukládat pouze uživatelé, kteří jsou přihlášeni, tzn. mají přístup ke klíčům daného účtu. Obecně pak platí, že každý může číst obsah z jakéhokoliv prostoru. Nicméně za použití SEA lze tento obsah šifrovat a ostatní (pokud nemají přístup) nedokáží zobrazit obsah původní.

Obecně se pro přístup k datům ve všech prostorech využívá metody *get()* a pro zápis metod *put()* a *set()*. Metoda put vkládá data bez ověření jejich existence a pokud pod daným klíčem data existují, tak jej přepíše. Set je o něco komplexnější,

ověřuje existenci a pokud již uzel existuje, data nepřepisuje. Zásadní rozdíl je i v tom, že *put()* data vkládá jako textový řetězec, pokud je ale třeba vložit nějaký objekt, musí se využít funkce *set()*. Tyto metody se v rámci prostorů liší pouze argumenty, které od sebe prostory rozlišují formou speciálních znaků (více v dalších kapitolách). Na obrázku 6 je možné vidět, jak jsou data ukládána do lokálního uložiště prohlížeče.

Obrázek 6 Data uložená do lokálního uložiště prohlížeče
Zdroj: autor

Veřejný prostor

Jelikož je veřejný prostor nejpřístupnějším prostorem, není potřeba žádného speciálního znaku v argumentu. Zápis do veřejného prostoru demonstruje následující ukázka.

```
//vloží pod klíč greetings objekt hello a do něj hodnotu world
gun.get("greetings").put({hello : "world"});

//obodbný zápis
gun.get("greetings2").get("hello").put("world");

//vkládání objektů za použití set
gun.get('messages').set({
    name: "Alice",
    message: "Lorem ipsum",
    createdAt: Date.now()
})
//výpis objektů, nejdříve je potřeba data pomocí get získat a
následně namapovat
const messages = gun.get("messages");
messages.map().once(m => { console.log(m); })
```

Ukázka 5 Práce s daty ve veřejném prostoru
Zdroj: autor

Uživatelský prostor

Standardně uživatel ukládá osobní data pouze do svého prostoru. Pokud se uživatel autentizuje, nahrají se potřebné informace o něm do proměnné. Skrze ní lze pak číst či zapisovat data do uživatelského prostoru. Je však možné udělit přístup do svého prostoru jinému uživateli (více v kapitole o sdílení dat). Pokud má pak uživatel nastavený přístup, přistupuje k datům jiného uživatele podobně jako v prostoru veřejném. Pro přístup do prostoru konkrétního uživatele, je nutné znát jeho alias, respektive veřejný klíč a před něj přidat znak ~, který určuje, že se přistupuje k uživatelskému prostoru.

Jelikož mohou být cizí uživatelská data uložena na uzlech jiných uživatelů, je nutné citlivější data chránit formou šifrování. Šifrování nabízí právě SEA, které definuje několik užitečných metod. Pro obyčejné zašifrování vlastních dat, aby se k nim nedostal někdo jiný, postačí metoda *encrypt()*, kde je vstupem řetězec, který se má zašifrovat a klíče přihlášeného uživatele. Obdobné to je i s metodou pro dešifrování *decrypt()*, pouze s tím rozdílem, že prvním parametrem je zašifrovaný objekt.

```
//přihlášení a nahrání instance uživatele do proměnné alice
const alice = user.auth("alice", "password123");

//uložení klíče name do uživatelského prostoru přes proměnnou
alice.get('name').put('Alice Novaková');

//pokud je potřeba lze tato data zašifrovat pomocí klíčů uživatele
const safeName = await sea.encrypt('Alice Novaková', alice._.sea);
alice.get('name').put(safeName);

//před použitím je potřeba data rozšifrovat
const decryptName = await sea.decrypt(safeName, alice._.sea)

 //získání dat alice skrze její veřejný klíč
const aData = gun.get('~' + Alice.pub).get("name");
```

Ukázka 6 Práce s daty v uživatelském prostoru
Zdroj: autor

Neměnný prostor

Pro přístup k neměnnému prostoru, se používá znak #. Jelikož data ukládaná do tohoto prostoru jsou neměnná, lze pro ně vytvořit hash, který se použije jako klíč k těmto datům.

```
const data = "tyto data nelze měnit";
//metoda work pro získání hashe (data, pair, callback, opt)
//pair a callback jsou null, jelikož nejsou třeba a do optional lze
určit použitou metodu SHA-256
const hash = await SEA.work(data, null, null, {name: "SHA-256"});
gun.get('#').get(hash).put(data);
```

Ukázka 7 Ukládání dat do neměnného prostoru
Zdroj: autor

8.4 Mazání dat

Mazání dat není v distribuovaném systému jednoduchou záležitostí. Data totiž nelze smazat ze všech uzlů, jelikož ne vždy budou všechny uzly držící tyto data aktivní. Pokud by nastal pokyn smazat data, všechny aktivní uzly by tak učinily. Bohužel v situaci, kdy by se připojil uzel, který má starší kopii dat, obsahující smazanou informaci, byla by delegována ostatním uzlům, jelikož nemají nikde uložen příznak o tom, že byla již smazána.

Mazaní dat se tedy v tomto případě řeší přepsáním na novou hodnotu, jako je null či prázdný řetězec. V konečném důsledku bude pouze změněna reference, ale uzel s daty bude existovat dále.

```
//smazání hodnoty name přihlášené uživatelky alice
alice.get("name").put(null);
```

Ukázka 8 Mazání dat
Zdroj: autor

8.5 Sdílení dat

Sdílení dat je v GUN JS velmi jednoduché. Všechna data jsou totiž přístupná všem, ať už se jedná o data z kteréhokoliv prostoru. Jenže tento fakt nemusí být někdy žádoucí, například v případě, že se ukládají citlivější data. Těmi můžou být nějaké osobní informace nebo chat mezi uživateli. K tomuto účelu právě slouží šifrování a podepisování dat pomocí klíčů uživatele. Navíc data je možné uložit do vlastního uživatelského prostoru a pomocí metody *certify()* nastavit jinému uživateli práva pro úpravu těchto dat. Pokud je třeba obsah zašifrovat ale zároveň musí být informace dostupné i jinému konkrétnímu uživateli, stačí při šifrování použít svůj klíč a veřejný klíč jiného uživatele. Takto zašifrovaná data si dokáže druhý uživatel snadno dešifrovat. Naopak pro ostatní uživatele jsou data nečitelná a rozluštění obsahu by vyžadovalo útok hrubou silou, který je velmi časově náročný a útočník by takto zašifrovaná data nebyl zřejmě schopný získat.

Někdy může být třeba data sdílet mezi více uživateli. Příkladem může být skupinový chat. V tomto případě lze zajistit šifrování opět skrze utajovaný klíč. Tvůrce skupinového chatu může vytvořit libovolný řetězec nebo si klíč nechat vygenerovat za použití SEA a metody *pair()*. Tento klíč lze předat ostatním uživatelům zašifrováním tohoto klíče způsobem, který byl zmíněn výše v prvním odstavci, tedy za použití svých klíčů a veřejného klíče uživatele, pro kterého se obsah šifruje. To znamená, že pro X uživatelů, bude nutné vytvořit X zašifrovaných klíčů. Tento vytvořený klíč, pak mohou ostatní uživatelé používat k šifrování zasílaných zpráv a každý uživatel vlastnící klíč bude moci zprávu přečíst.

Aby bylo možné ověřit, kým jsou data vytvořena, lze data podepsat svým klíčem. Uživatelům, kteří chtějí ověřit původce zprávy, stačí pak znát veřejný klíč uživatele, od kterého má obsah pocházet a na základě toho zjistit, zda je autorem opravdu on.

```
//pomocí pair() lze vygenerovat klíče a tím simulovat existujícího
uživatele
const Alice = await SEA.pair()
const Bob = await SEA.pair()

// Alice uděluje přístup Bobovi pro zápis do její části grafu inbox
a stories
//parametry metody jsou: who - cizí klíč, případně seznam klíčů [],
//policy - lze měnit části podgrafu inbox a vytvářet nové poduzly a
lze měnit části podgrafu stories
//authority - vystavující autoritou je alice, cb - nullový a v
optional objektu definice doby platnosti
const certificate = await SEA.certify(Bob.pub, [{"*": "inbox", "+":
"*"}, {"*": "stories"}], Alice, null, {expiry:
Gun.state()+(60*60*24*1000)})

// Bob může psát do grafu Alice za použití optional objektu, kde se
skrze cert předá vystavený certifikát
gun.get('~'+Alice.pub).get('inbox').get('bob'+Bob.pub).put('ahoj
alice', null, {opt: {cert: certificate}})
```

Ukázka 9 Použití metody certify
Zdroj: autor

```
//STRANA ALICE
//získání informací o Bobovi na základě jeho veřejného klíče
const bob = gun.user(Bob.pub);
//vytvoření secretu za použití Bobova veřejného šifrovacího klíče a
klíčů přihlášené alice
const secret = await SEA.secret(bob.epub, alice._.sea)
//zašifrování zprávy pro boba na základě vytvořeného secretu
const encryptedData = await SEA.encrypt('zpráva pro Boba', secret);
//zašifrovaná data lze ještě podepsat
const signedData = await SEA.sign(encryptedData , alice._.sea);


//STRANA BOBA
const alice = gun.user(Alice.pub);
const secret = await SEA.secret(alice.epub, bob._.sea)
//ověření původce zprávy, tato metoda vrátí objekt bez podpisu
const verifyData = await SEA.verify(signedData, alice.pub)
//dešifrování zprávy
if(verifyData){
  const decryptedData = await SEA.decrypt(verifyData, secret);
}
```

Ukázka 10 Použití koncového šifrování a podepisování
Zdroj: autor

8.6 Publikování na IPFS

Pokud je webová stránka statická využívající dynamičnost pouze formou Javascriptu na klientské části, lze ji a veškerý dodatečný obsah (například obrázky, videa, …) publikovat na IPFS.

Možností, jak publikovat na IPFS, je několik. První z nich je vytvoření vlastního IPFS uzlu. Přes tento uzel se pak nahrávají a získávají veškerá data. Avšak tento uzel by měl být, pro co nejlepší dostupnost souborů stále online, jelikož dostupnost je daná množstvím ostatních online uzlů, na které nelze spoléhat.

Další možností je využití služeb k tomuto účelu navržených, například Pinata umožňující aktualizaci souborů díky propojení s Githubem a publikování souborů skrze API. Některé služby jsou zdarma, avšak za cenu nejisté dostupnosti, jiné jsou placené, ale se zaručenou dostupností uzlu.

Jelikož web může být občas změněn, je otázkou, zda nevyužít IPNS, u kterého by nebylo nutné manuálně měnit odkazy v rámci webu. Bohužel IPNS trpí poměrně dlouhou odezvou při překladu, která může trvat od několika vteřin až po několik minut, v závislosti na množství dat, které je potřeba prohledat. Z tohoto důvodu nebude IPNS v praktické části použito, jelikož v případě webových stránek je rychlost načtení velmi důležitá pro udržení návštěvníka.

Vytvoření vlastního uzlu

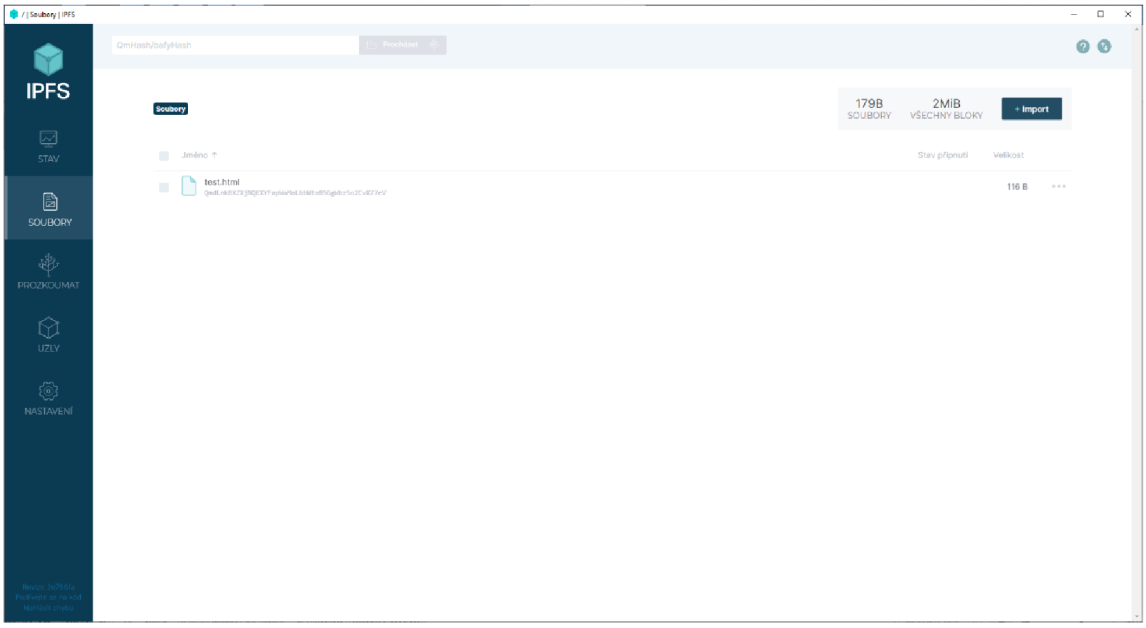
Pro vytvoření IPFS uzlu je zapotřebí stažení IPFS desktopové aplikace, jež je dostupná pro Windows, Mac i Linuxové distribuce. Tu lze stáhnout z repositáře na github.com[1]. Po instalaci a spuštění se spustí i IPFS daemon a zařízení se automaticky připojí k IPFS síti.

Nyní je vytvořen funkční lokální uzel IPFS. Aby uživatelé data z tohoto uzlu byli schopni přijímat, museli by mít vlastní IPFS uzel. To nebude nutné v případě, že prohlížeče v budoucnu zahrnou podporu pro distribuované protokoly. Momentálně je ale možné využít tzv. IPFS veřejné brány. Jednou z nich je například ipfs.io. Konkrétní soubor lze pak získat skrze url: *ipfs.io/ipfs/<CID_SOUBORU>.*

---

[1] Z odkazu: https://github.com/ipfs/ipfs-desktop

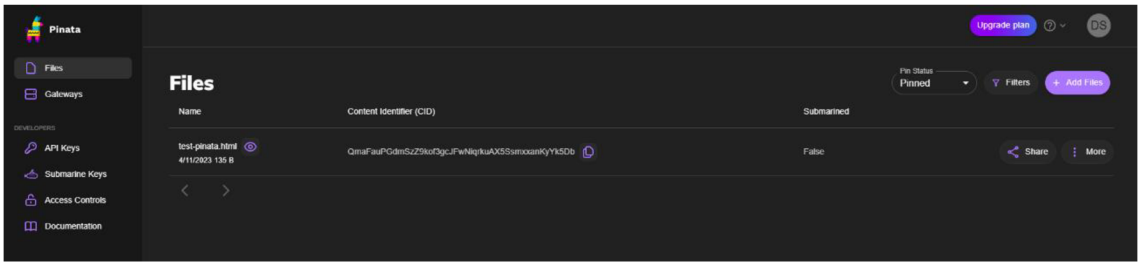
Ovládání této aplikace je velmi intuitivní, viz obrázek 7. Pro práci se soubory stačí navštívit záložku *Soubory*, kde je možné nové soubory importovat, publikovat do IPNS či mazat.



Obrázek 7 Rozhraní IPFS desktopové aplikace
Zdroj: autor

Použití IPFS služeb

V případě, že vytváření vlastního uzlu nepřipadá v úvahu, stačí založit účet na pinata.cloud. Ta má též velmi jednoduché ovládání, viz obrázek 8. V neplacené variantě není příliš mnoho možností, avšak v případě vývoje větší webové aplikace, bude jistě praktičtější využití placené verze oproti používání vlastního uzlu. Tato služba je totiž přímo stavěna pro vývoj a nabízí tak snazší údržbu souborů v případě nasazení nové verze než při publikování na vlastním uzlu.



Obrázek 8 Webové rozhraní Pinata
Zdroj: autor

Webová stránka na vlastní doméně

Jelikož odkazy na veřejné IPFS brány nejsou zrovna nejpřehlednější a pro člověka jsou těžko zapamatovatelné, je třeba poskytnout webovým stránkám vlastní doménu. Po registraci libovolné domény se u DNS změní A záznam na IP adresu na některou z veřejných IPFS bran. To zajistí, že uživatel při zadání domény bude nasměrován na konkrétní IPFS bránu. Nyní je třeba určit, jaký obsah se má uživateli zobrazit. K tomu postačí znát CID souboru (hash) a přidat nový textový (TXT) DNS záznam. Ten musí být pojmenován *_dnslink*. Pro doménu *priklad.cz*, by tedy takovýto záznam vypadal takto *_dnslink.priklad.cz*. Hodnotou v tomto záznamu pak bude *dnslink=/ipfs/<CID_souboru>.*

8.7 Porovnání rychlosti IPFS a webhostingů

Pro ukázku byl jeden totožný soubor nahrán na webhostingové servery a IPFS. Soubor byl ve všech případech přístupný z internetu (netestovalo se lokálně). U každé varianty bylo 20x otestováno, jak rychle je soubor načten. V případě webhostingu byla jedna z variant zdarma a druhá placená a jednalo se o servery s umístěním v České republice. Obě varianty jsou formou sdíleného hostingu, což je ovšem jedna z nejčastěji používaných variant pro menší weby.

V případě IPFS byla stránka načtena v průměru za *98,94 ms* s mediánem *87 ms*. U webhostingů byly výsledky o něco horší. U neplacené varianty se stránka načítala v průměru za *353,16 ms* s mediánem *331 ms* a u placené varianty byla doba průměrného načtení *269,79 ms* a medián *269 ms*.

To tedy poukazuje na to, že IPFS může být několikanásobně rychlejší než webhosting. U webhostingu samozřejmě závisí na poskytovateli a na zvolené variantě. Výsledky u nesdíleného hostingu by jistě dopadly odlišně, protože by se uživatelé nemuseli mezi sebou dělit o zdroje.

8.8 Porovnání s centralizovanými technologiemi

Pro některé decentralizované technologie je velmi klíčové, kolik vývojářů (uživatelů) jich využívá. Množství uživatelů je zřejmě nejpodstatnější pro IPFS, u kterého ovlivňuje celkovou dostupnost souborů sdílených na IPFS síti. Po spuštění vlastního IPFS uzlu se v průměru připojí k dalším 486 uzlům. Tato hodnota byla vypočtena z měření od 15. 1. 2023 do 15. 2. 2023, kdy bylo během tohoto období provedeno 30 připojení k IPFS síti. Vzhledem k tomu, že se má jednat o celosvětový distribuovaný souborový systém, není množství těchto uzlů velké.

Nicméně, pro ostatní technologie je též podstatná velikost její komunity, která určuje celkový zájem o danou technologii, a tedy i její budoucí vývoj a udržení na trhu. Z tohoto důvodu byly vybrány technologie popsané v této práci, které mají dostupný veřejný repositář na github.com. Celkem se jedná o 9 decentralizovaných technologií, ke kterým bylo vybráno 9 nejznámějších centralizovaných technologií (též dostupných z github.com), s kterými jsou porovnávány. Jelikož jsou decentralizované technologie různého charakteru, jsou vybrány i technologie centralizované pro různá odvětví, konkrétně technologie pro vývoj frontendu, backendu a ukládání dat.

Je sledován počet hvězdiček, který určuje oblíbenost této technologie, dále forks (počet zkopírování repositáře) a contributors (přispěvatelé), což jsou uživatelé, kteří nějakým způsobem k danému projektu přispěli, například vytvořením nějaké funkcionality nebo dokumentace. Tyto údaje jsou dostupné v tabulce 1 a je z nich vypočteno celkové zastoupení decentralizovaných technologií u každé zmíněné kategorie, které je možné vidět na grafu 1.

V počtu hvězdiček mají decentralizované technologie zastoupení pouze 12 % ze všech udělených hvězd. Co se týče kopií repositáře, zde je zastoupení ještě menší, a to pouze 4 %. U přispěvatelů se pak jedná o 9% zastoupení.

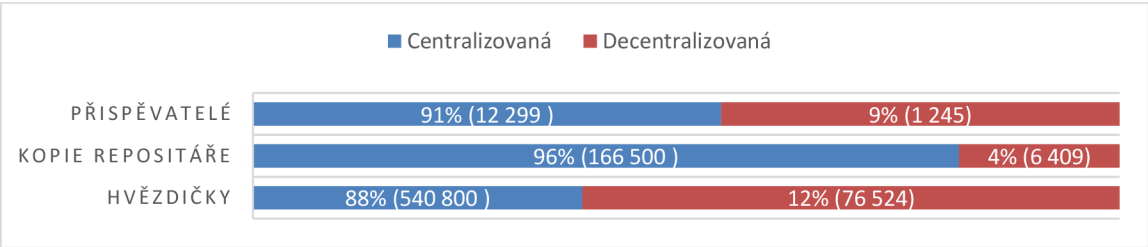
Nejlepší zastoupení mají decentralizované technologie v počtu udělených hvězd, avšak tyto hvězdy může udělit i ten, kdo tuto technologii nikdy nepoužil a pouze tím dává najevo, že mu daný projekt přijde zajímavý. Větší výpovědní hodnotu má určitě počet kopií repositáře, ale jedná se o údaj, který se může od skutečnosti lišit, vzhledem k faktu, že existuje více možností a zdrojů, jak mohou

uživatelé daný projekt stáhnout (použít). Nejdůležitějším z těchto ukazatelů je tedy množství přispěvatelů, které vyjadřuje skutečný zájem o technologii, jelikož uživatel musel obětovat svůj čas, aby přispěl k vývoji daného projektu a tím k jeho vylepšení.

I když se 9% zastoupení nezdá jako příliš velké, není tato hodnota tak špatná, s přihlédnutím k faktu, že jsou decentralizované technologie, které jsou poměrně nové a většinou stále ve fázi vývoje, porovnávány s nejpoužívanějšími a léty ověřenými centralizovanými technologiemi. Přesto může toto malé množství ovlivňovat u některých decentralizovaných technologií jejich dostupnost.

Technologie	Architektura	Hvězdičky	Kopie repositáře	Přispěvatelé
Angular	centralizovaná	59 200	28 100	1 703
Asp.net	centralizovaná	31 500	9 000	1143
MongoDB	centralizovaná	23 600	5 500	658
MySQL Server	centralizovaná	9 000	3 400	97
Node.js	centralizovaná	94 800	25 700	3 220
React	centralizovaná	206 000	42 900	1 615
Spring	centralizovaná	51 500	36 000	725
Symphony	centralizovaná	28 200	9 100	2760
Vue.js	centralizovaná	37 000	6 800	378
BigChainDB	decentralizovaná	4 000	796	72
GunJS	decentralizovaná	17 100	1 200	140
IPFS	decentralizovaná	22 000	1 500	78
Lotus (FileCoin)	decentralizovaná	2 600	1 200	197
OrbitDB	decentralizovaná	7 600	552	65
Siad (Sia)	decentralizovaná	105	22	13
Solidity (Ethereum)	decentralizovaná	20 000	690	568
Storj	decentralizovaná	2 700	386	95
ThreadDB	decentralizovaná	419	63	17

Tabulka 1 Data pro zjištění oblíbenosti technologií (k datu 15.2.2023)
Zdroj: autor, github.com



Graf 1 Podíl centralizovaných a decentralizovaných technologií (v %)
Zdroj: autor

9 Shrnutí výsledků

V rámci praktické části bylo ukázáno, že GUN JS nabízí dostatek funkcí pro správu dat a jejich zabezpečení. Zároveň běžný uživatel nemusí, až na doporučenou dobrovolnou zálohu klíčů, dělat žádné kroky navíc v porovnání s centralizovanými aplikacemi. GUN JS je též velmi rychlý, což je způsobeno přístupem k datům skrze nejbližší uzly a faktem, že tyto data jsou uložena i na samotném uzlu. Tento protokol je však stále ve fázi vývoje a některé funkcionality lze stále vylepšit. Zatím je rozhodně velkým nedostatkem potřeba superuzlů a relay uzlů pro propojení klientů, kteří spolu nedokáží napřímo komunikovat.

Co se týče decentralizovaných aplikací obecně, problémem je obnova účtů při zapomenutí hesla či krádeži účtu, jelikož tvůrci aplikace nemají přímý přístup k datům a vše je pouze v rukou uživatelů. Navíc jsou uživatelé často neopatrní a volí slabá nebo stejná hesla k většině aplikací, což v případě centralizovaných systémů není tak velkou překážkou, hlavně díky přítomnosti podpory a možnosti účet obnovit.

Technologií existujících pro vývoj decentralizovaných aplikací a distribuci dat mezi uzly existuje mnoho a neustále přibývají. Většina z nich je založená na blockchainu, který je vyhlášen svou bezpečností, co se integrity dat týče a schopností řešit nedůvěru mezi uzly. Tyto technologie jsou však spíše využívány pro tvorbu decentralizovaných finančních aplikací (DEFI). Pro decentralizovanou webovou aplikaci nejsou příliš vhodné, především kvůli pomalému zpracování požadavků, které je nevyhnutelné vzhledem k nutnosti hlasovat o platnosti požadavku mezi uzly sítě. Nicméně je možné je využít alespoň pro kontrolu integrity dat.

Decentralizovaných technologií, které naopak nejsou založeny na blockchainu, není tolik a komunita používající tyto technologie není příliš velká, i přestože jsou oproti blockchainu mnohem rychlejší a jsou vhodné pro vývoj jiných než jen DEFI aplikací.

10 Závěry a doporučení

V teoretické části byla přiblížena problematika blockchainu, rozebrány požadavky na decentralizované aplikace a byly zde popsány konkrétní technologie pro decentralizované ukládání dat a souborů a jejich výhody a nevýhody.

Součástí praktické práce bylo zvolení vhodné kombinace technologií, s jejichž pomocí lze vyvíjet webové aplikace. Konkrétně se jednalo o Gun JS pro distribuované ukládání a dat a IPFS pro ukládání souborů, respektive statické stránky.

Přes fakt, že vyzkoušené technologie fungují velmi dobře, co se rychlosti týče, mohou nastat problémy v případě dostupnosti. GUN JS pro korektní fungovaní vyžaduje superuzly, kterých by mělo být více, aby se konkrétní aplikace dala považovat vůbec za decentralizovanou s distribuovanými daty. Nicméně není nutné, aby tyto servery byly tak výkonné jako v případě centralizovaných systémů.

Rychlost IPFS je opět velmi pěkná, avšak vždy je lepší provozovat vlastní IPFS uzel, jakožto formu zálohy, nebo si nějaký zakoupit formou služby. Navíc za předpokladu, že se nepodaří vylepšit rychlost IPNS, bude správa webové aplikace na IPFS s její rostoucí velikostí stále těžší a krkolomnější z důvodu udržování aktuálního odkazu na CID.

I přesto, že existuje poměrně mnoho technologií, většinou vychází z nějaké již existující. Několik zde zmíněných například rozšiřuje IPFS a některé využívají blockchain. Základy mají tedy velmi podobné. Největší nevýhodou všech těchto technologií však je, že mají poměrně malou komunitu a nejsou příliš známé. I přesto se jejich vývojáři a komunita vývoji neustále věnují a tyto technologie se zlepšují a posouvají dopředu.

Lze tedy konstatovat, že nedostatečné povědomí o těchto technologiích zapříčiňuje jejich horší dostupnost, a proto je vhodné je zatím používat pouze pro vývoj malých webových aplikací. Ve stavu, v jakém se momentálně nacházejí, je nepředstavitelné vyvíjet aplikace složitějšího charakteru. Je tedy možné tyto technologie použít pro vývoj decentralizovaných webových aplikací, ale je potřeba počítat s tím, že jsou stále ve fázi vývoje a mohou být nestabilní. Avšak i v této fázi nabízí velmi dobré zabezpečení dat a lze s nimi, až na pár detailů, pracovat tak,

aby běžný uživatel nepoznal rozdíl, a tudíž pro něj ovládání systému mohlo zůstat uživatelsky přívětivé.

Pokud je ale nutné se na daný systém opravdu spolehnout, je stále lepší využít centralizovanou variantu, která nabízí mnohonásobně větší množství ověřených technologií.

11 Zdroje

[1] TERRA, John. What is Web 1.0, Web 2.0, and Web 3.0? Definitions, Differences & Similarities. *Simplilearn.com* [online]. 25. duben 2022 [vid. 2023-02-18]. Dostupné z: https://www.simplilearn.com/what-is-web-1-0-web-2-0-and-web-3-0-with-their-difference-article

[2] HIREMATH, B K a Anand Y KENCHAKKANAVAR. An Alteration of the Web 1.0, Web 2.0 and Web 3.0: A Comparative Study. *A Comparative Study. imperial journal of interdisciplinary research. 2. 705-710.* 2016, **2**(4).

[3] RUDMAN, Riaan a Rikus BRUWER. Defining Web 3.0: opportunities and challenges. *The Electronic Library* [online]. 2016, **34**(1), 132–154. ISSN 0264-0473. Dostupné z: doi:10.1108/EL-08-2014-0140

[4] NATH, Keshab, Sourish DHAR a Subhash BASISHTHA. Web 1.0 to Web 3.0 - Evolution of the Web and its various challenges. In: *2014 International Conference on Optimization, Reliabilty, and Information Technology (ICROIT)*: *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)* [online]. Faridabad, Haryana, India: IEEE, 2014, s. 86–89 [vid. 2023-02-19]. ISBN 978-1-4799-2995-5. Dostupné z: doi:10.1109/ICROIT.2014.6798297

[5] NATH, Keshab a Raja ISWARY. What Comes after Web 3.0? Web 4.0 and the Future [online]. nedatováno, **2015**. Dostupné z: https://www.researchgate.net/publication/281455061_What_Comes_after_Web_30_Web_40_and_the_Future

[6] AGHAEI, Sareh. Evolution of the World Wide Web : From Web 1.0 to Web 4.0. *International journal of Web & Semantic Technology* [online]. 2012, **3**(1), 1–10. ISSN 09762280. Dostupné z: doi:10.5121/ijwest.2012.3101

[7] CHOUDHURY, Nupur. World Wide Web and Its Journey from Web 1.0 to Web 4.0. *International Journal of Computer Science and Information Technologies, Vol. 5 (6) , 2014, 8096-8100 World Wide Web and Its* [online]. 2014, **5**. Dostupné z: https://docslib.org/doc/2952074/world-wide-web-and-its-journey-from-web-1-0-to-web-4-0

[8] WHAT IS WEB 4.0? *Criticalcase* [online]. 26. červenec 2018 [vid. 2023-02-21]. Dostupné z: https://www.criticalcase.com/blog/what-is-web-4-0.html

[9] PODISHETTI, Akash. *What is Web 5.0?* [online]. 4. červenec 2022 [vid. 2023-02-21]. Dostupné z: https://www.business-standard.com/podcast/technology/what-is-web-5-0-122070400030_1.html

[10] VENKATRAMAN, Karthik. Web 5.0 the future? *Geek Culture* [online]. 3. červenec 2022 [vid. 2023-02-21]. Dostupné z: https://medium.com/geekculture/what-is-web-5-0-f67cb71d07b6

[11] WESTON, Georgia. What is Web 5.0 - Explained. *101 Blockchains* [online]. 8. červenec 2022 [vid. 2023-02-21]. Dostupné z: https://101blockchains.com/web-5-0/

[12] RAVAL, Siraj. *Decentralized applications: harnessing Bitcoin's blockchain technology*. Beijing ; Boston: O'Reilly, 2016. ISBN 978-1-4919-2454-9.

[13] KASTHALA, Venkat. *Blockchain hash* [online]. Picture of blockchain hash. [vid. 2023-02-16]. Dostupné z: https://miro.medium.com/v2/resize:fit:4800/format:webp/1*mNdCyhj2WRSzmgTOVztaUg.png

[14] ALABDULWAHHAB, Faten Adel. Web 3.0: The Decentralized Web Blockchain networks and Protocol Innovation. In: *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*: *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)* [online]. Riyadh: IEEE, 2018, s. 1–4 [vid. 2023-02-19]. ISBN 978-1-5386-4427-0. Dostupné z: doi:10.1109/CAIS.2018.8441990

[15] *Blockchain Explained: How does a transaction get into the blockchain? | Euromoney Learning* [online]. [vid. 2023-02-24]. Dostupné z: //www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain

[16] SRIMAN, B, S. Ganesh KUMAR a P. SHAMILI. Blockchain Technology: Consensus Protocol Proof of Work and Proof of Stake. In: Subhransu Sekhar DASH, Swagatam DAS a Bijaya Ketan PANIGRAHI, ed. *ICICA*: *Intelligent computing and applications: proceedings of ICICA 2019*. Singapore: Springer, 2021, Advances in intelligent systems and computing, 1172. ISBN 9789811555664.

[17] HUONG NGUYEN MAI. Blockchain technology and auditing - A Survey on Awareness and Understanding of Auditors in Finland about Blockchain and Auditing [online]. 2021. Dostupné z: https://urn.fi/URN:NBN:fi:amk-2021112521663

[18] ANTONOPOULOS, Andreas M. a Gavin WOOD. *Mastering Ethereum: building smart contracts and DApps*. First edition. Sebastopol, CA: O'Reilly, 2019. ISBN 978-1-4919-7194-9.

[19] JENSEN, Johannes Rude, Victor VON WACHTER a Omri ROSS. An Introduction to Decentralized Finance (DeFi). *Complex Systems Informatics and Modeling Quarterly* [online]. 2021, (26) [vid. 2023-03-08]. ISSN 22559922. Dostupné z: doi:10.7250/csimq.2021-26.03

[20] WANG, Qin, Rujia LI, Qi WANG a Shiping CHEN. Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges [online]. 2021 [vid. 2023-03-08]. Dostupné z: doi:10.48550/ARXIV.2105.07447

[21] DANIEL, Erik a Florian TSCHORSCH. IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks [online]. 2021 [vid. 2023-03-08]. Dostupné z: doi:10.48550/ARXIV.2102.12737

[22] BENET, Juan. IPFS - Content Addressed, Versioned, P2P File System [online]. 2014 [vid. 2023-02-19]. Dostupné z: doi:10.48550/ARXIV.1407.3561

[23] KUMAR, Randhir a Rakesh TRIPATHI. Implementation of Distributed File Storage and Access Framework using IPFS and Blockchain. In: *2019 Fifth International Conference on Image Information Processing (ICIIP)*: *2019 Fifth International Conference on Image Information Processing (ICIIP)* [online]. Shimla, India: IEEE, 2019, s. 246–251 [vid. 2023-02-19]. ISBN 978-1-72810-899-5. Dostupné z: doi:10.1109/ICIIP47207.2019.8985677

[24] CHEN, Yongle, Hui LI, Kejiao LI a Jiyang ZHANG. An improved P2P file system scheme based on IPFS and Blockchain. In: *2017 IEEE International Conference on Big Data (Big Data)*: *2017 IEEE International Conference on Big Data (Big Data)* [online]. Boston, MA: IEEE, 2017, s. 2652–2657 [vid. 2023-02-19]. ISBN 978-1-5386-2715-0. Dostupné z: doi:10.1109/BigData.2017.8258226

[25] DE FIGUEIREDO, Sammy, Akash MADHUSUDAN, Vincent RENIERS, Svetla NIKOVA a Bart PRENEEL. Exploring the storj network: a security analysis. In: *SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing*: *Proceedings of the 36th Annual ACM Symposium on Applied Computing* [online]. Virtual Event Republic of Korea: ACM, 2021, s. 257–264 [vid. 2023-02-19]. ISBN 978-1-4503-8104-8. Dostupné z: doi:10.1145/3412841.3441908

[26] MCCONAGHY, Trent. The DCS Triangle. *Medium* [online]. 2. září 2020 [vid. 2023-03-09]. Dostupné z: https://blog.bigchaindb.com/the-dcs-triangle-5ce0e9e0f1dc

[27] PREGUIÇA, Nuno, Carlos BAQUERO a Marc SHAPIRO. Conflict-Free Replicated Data Types CRDTs. In: Sherif SAKR a Albert ZOMAYA, ed. *Encyclopedia of Big Data Technologies* [online]. Cham: Springer International Publishing, 2018 [vid. 2023-03-09], s. 1–10. ISBN 978-3-319-63962-8. Dostupné z: doi:10.1007/978-3-319-63962-8_185-1

[28] ZHENG, Gavin. *Ethereum smart contract development in solidity*. Singapore: Springer, 2021. ISBN 9789811562181.

[29] NADAL, Mark. Gun JS dokumentace. *Gun Docs* [online]. 2023. Dostupné z: https://gun.eco/docs/API

[30] CLAUDIO ROCCHINI. *Radix tree* [online]. 17. květen 2007 [vid. 2023-02-28]. Dostupné z: https://en.wikipedia.org/wiki/File:Patricia_trie.svg

[31] IRVINE D. a PÖYHTÄRI S. OrbitDB. *Orbit DB Field Manual* [online]. 8. březen 2023 [vid. 2023-03-09]. Dostupné z: https://github.com/orbitdb/field-manual

[32] ThreadDB dokumentace. *Docs Textile* [online]. 2023. Dostupné z: https://docs.textile.io/threads/

[33] MCCONAGHY, Trent. BigchainDB: A Scalable Blockchain Database [online]. 2016. Dostupné z: https://gamma.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor: **Bc. Daniel Schmid**

Studium: I2100078

Studijní program: N1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Název diplomové práce: **Použitelnost decentralizovaných technologií pro vývoj webových aplikací (Web 3.0)**

Název diplomové práce AJ: Applicability of decentralized technologies for web application development (Web 3.0)

Cíl, metody, literatura, předpoklady:

Cíl práce:

Cílem diplomové práce je zjistit použitelnost decentralizovaného řešení pro vývoj webových aplikací, pro které je typická centralizovaná architektura. Hlavními kritérii je jednoduchost ovládání pro běžného uživatele, zabezpečení uživatelských dat a množství výběru technologií u decentralizovaného řešení.

Osnova práce:

Web 3.0 a předchozí verze webu

Decentralizované aplikace

Blockchain

Popis decentralizovaných technologií pro ukládání dat a souborů, jejich výhody a nevýhody

Zabezpečení decentralizovaných technologií

Praktická část s ukázkami vybraných technologií

Závěr a shrnutí výsledků

YANO, M., DAI, CH., MASUDA, K., KISHIMOTO, Y. Blockchain and Crypt Currency. (2020). ISBN 9789811533754.

ANTONOPOULOS, Andreas M. Mastering Ethereum. (2018). ISBN 9781491971949.

ZHENG, G., GAO L., HUANG, L., GUAN J. Ethereum Smart Contract Development in Solidity. (2020). ISBN 9789811562174.

RAVAL, Siraj. Decentralized applications: harnessing Bitcoin's blockchain technology. Beijing ; Boston: O'Reilly, 2016. ISBN 978-1-4919-2454-9.

NADAL, Mark. Gun JS dokumentace. Gun Docs [online]. 2023. Dostupné z: https://gun.eco/docs/API

Zadávající pracoviště: Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu

Vedoucí práce: prof. RNDr. PhDr. Antonín Slabý, CSc.

Datum zadání závěrečné práce: 11.1.2023