Decentralizovaná aplikace pro privátní sdílení dokumentů s ověřenou integritou

Bc. Jan Bureš

Diplomová práce

2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Jan Bureš**
Osobní číslo:	**A21823**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Decentralizovaná aplikace pro privátní sdílení dokumentů s ověřenou integritou**
Téma práce anglicky:	**Decentralized Application for Private Sharing of Documents with Verified Integrity**

Zásady pro vypracování

1. Nastudujte a popište problematiku vývoje aplikací pro Web3.0.
2. Popište technologii Ethereum blockchain a její využití v oblasti ověřování dokumentů.
3. Zvolte a popište vhodné technologie pro vývoj webové aplikace v kontextu tématu práce.
4. Navrhněte vlastní decentralizovanou webovou aplikaci pro sdílení a ověřování dokumentů v privátním (firemním) prostředí.
5. Proveďte vlastní implementaci aplikace.
6. Výsledné řešení vhodně otestujte a porovnejte se stávajícími řešeními.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. TAPSCOTT, Don a Alex TAPSCOTT. Blockchain Revolution: How the technology behind bitcoin and other cryptocurrencies is changing the world. Penguin Books, 2018. ISBN 9781788473040.
2. ANTONOPOULOS, Andreas M. a Gavin WOOD. Mastering Ethereum: Building Smart Contracts and DApps. 2018. ISBN 9781491971949.
3. SOLORIO, Kevin, Randall KANNA a David H. HOOVER. Hands-On Smart Contract Development with Solidity and Ethereum. O'Reilly Media, 2019, 270 s. ISBN 978-1-492-04526-7.
4. RIPPON, Carl. Learn React with TypeScript 3. Packt Publishing Limited, 2018. ISBN 1789610257.

Vedoucí diplomové práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **2. prosince 2022**
Termín odevzdání diplomové práce: **26. května 2023**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 25. 5. 2023 Bc. Jan Bureš

podpis studenta

ABSTRAKT

V rychle se měnícím digitálním prostředí je adaptabilita klíčovým faktorem úspěchu. Tato diplomová práce zkoumá výzvy a příležitosti současného digitálního vývoje, zejména v kontextu evoluce webu od Web1.0 k Web3.0. S důrazem na rostoucí potřebu efektivního zabezpečení informačních systémů, práce představuje nový přístup, který kombinuje osvědčené techniky Web2.0 s inovacemi Web3.0. Cílem je poskytnout pevný základ pro budoucí vývoj, který spojuje solidní zabezpečení s inovativním využitím dostupných technologií.

Klíčová slova: Web2.0, Web3.0, Ethereum, MongoDB, vývoj, aplikace

ABSTRACT

In the rapidly changing digital environment, adaptability is a key factor for success. This thesis investigates the challenges and opportunities of current digital development, particularly in the context of the web's evolution from Web1.0 to Web3.0. With a focus on the growing need for efficient information system security, this work introduces a new approach that combines proven Web2.0 techniques with Web3.0 innovations. The goal is to provide a solid foundation for future development that combines robust security with innovative use of available technologies.

Keywords: Web2.0, Web3.0, Ethereum, MongoDB, development, application

Při této příležitosti, bych chtěl poděkovat vedoucímu své diplomové práce Ing. Petrovi Žáčkovi, Ph. D. za odborné rady a nasměrování v klíčových otázkách vývoje aplikace.

Dále bych chtěl poděkovat Bc. Davidovi Rábelovi za odbornou konzultaci v oblasti škálovaní backendové části aplikace a Bc. Michalovi Bernátkovi a Bc. Davidovi Sedlářovi za cenné konzultace v oblasti webového hostingu.

V neposlední řadě děkuji své rodině a přátelům za morální podporu po dobu celého studia.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

ÚVOD **11**
I TEORETICKÁ ČÁST **13**
1 WEB 2.0 ..... **14**
1.1 FRONTEND (FE) 14
1.1.1 HyperText Markup Language (HTML) 14
1.1.2 Cascading Style Sheets (CSS) 15
1.1.3 Javascript (JS) 15
1.1.4 Typescript (TS) 17
1.1.5 Architektonické vzory 17
1.1.7 React 21
1.1.8 Styled-components 22
1.2 BACKEND (BE) 23
1.3 DATABÁZE 23
1.3.1 SQL databáze 23
2 WEB 3.0 ..... **27**
2.1 BLOCKCHAIN 27
2.2 VÝHODY A NEVÝHODY WEB3.0 VŮČI WEB2.0 28
2.3 VYUŽITÍ BLOCKCHAINOVÝCH TECHNOLOGIÍ 30
2.3.1 Decentralizované aplikace (Dapps) 30
2.3.2 Non-Fungible Token (NFT) 31
2.3.3 DeFi 32
2.3.4 Decentralizovaná Autonomní Organizace (DAO) 33
2.3.5 Decentralizované sdílení záznamů 33
2.3.6 Internet of Things (IoT) 34
2.4 VALIDAČNÍ MECHANISMY 34
2.4.1 Proof of Work (PoW) 34
2.4.2 Proof of Stake (PoS) 35
2.5 BITCOIN 37
2.6 ETHEREUM 38
2.7 BLOCKCHAINOVÉ SÍTĚ 38
2.7.1 Hlavní síť (Mainnet) 38
2.7.2 Řešení 2. vrstvy (Layer-2 solutions) 39
2.7.3 Testovací sítě (testnet) 40
2.8 POPULÁRNÍ WEB 3.0 NÁSTROJE A SLUŽBY 41
2.8.1 InterPlanetary File System (IPFS) 41
2.8.2 Remix IDE 42
2.8.3 Truffle Suite 42
2.8.4 Metamask 42
2.8.5 Polygonscan 43
3 VERZOVÁNÍ PROJEKTU **44**
3.1 GITHUB 44

3.2 GITLAB 44
4 OVĚŘOVÁNÍ DOKUMENTŮ 46
4.1 OVĚŘOVÁNÍ DOKUMENTŮ VE WEB2.0 46
4.1.1 Digitální podpis 46
4.2 OVĚŘOVÁNÍ DOKUMENTŮ V (ETHEREUM) BLOCKCHAINU 48
5.1 DOMÉNA 49
5.2 AMAZON WEB SERVICES (AWS) 49
5.2.1 Amazon Simple Storage Service (S3) 50
5.2.2 Amazon Route 53 50
5.2.3 Amazon Cloudfront 52
5.2.4 Amazon Elastic Cloud Computing (EC2) 53
II PRAKTICKÁ ČÁST 55
6 VÝBĚR TECHNOLOGIÍ PRO VÝVOJ APLIKACE 56
6.1 VÝBĚR FRONTENDOVÝCH TECHNOLOGIÍ 56
6.3 VÝBĚR BACKENDOVÝCH TECHNOLOGIÍ 58
6.4 VÝBĚR VERZOVACÍ TECHNOLOGIE 58
6.5 VÝBĚR HOSTINGOVÝCH TECHNOLOGIÍ 58
7 PŘÍPRAVA TECHNOLOGIÍ 60
7.1 VYUŽITÉ NPM BALÍČKY 60
7.2 PŘÍPRAVA BLOCKCHAINOVÝCH TECHNOLOGIÍ 61
7.2.1 Založení Metamask peněženky 62
7.2.2 Přidání Mumbai sítě do peněženky 62
7.2.3 Získání testovacích tokenů 63
8 KOMUNIKACE APLIKACE 64
8.1 KOMUNIKACE APLIKACE S BLOCKCHAINEM 64
8.1.1 Smart contract pro aplikaci 65
8.1.2 Ukládání dokumentu do blockchainu 66
8.1.3 Přečtení dat z blockchainu 68
8.2 KOMUNIKACE FE S BE 69
8.3 STRUKTURA BACKENDU 70
8.3.1 Endpointy aplikace 73
9 POPIS A STRUKTURA APLIKACE 74
9.1 HLAVNÍ STRÁNKA APLIKACE 74
9.2 STRÁNKA DETAIL APLIKACE 74
9.3 STRÁNKA PRO ULOŽENÍ DOKUMENTU 75
9.4 STRÁNKA OVĚŘENÍ 76
9.5 PROCES VERIFIKACE INTEGRITY 76

9.6 Stránka nastavení 78
10 HOSTOVÁNÍ APLIKACE 79
10.1 Registrace domény pomocí Amazon Route 53 79
10.2 Konfigurace Amazon S3 bucket 80
10.3 Konfigurace Amazon Route 53 80
10.4 Zisk SSL certifikátu 81
10.5 Konfigurace Amazon Cloudfront 82
10.6 Konfigurace Amazon EC2 84
11 POROVNÁNÍ APLIKACE S EXISTUJÍCÍMI ŘEŠENÍMI 86
11.1 DocuSign 86
11.2 Adobe Document Cloud 87
11.3 Diplomachain 87
11.4 IBM Blockchain 87
12 SUMARIZACE APLIKACE 89
ZÁVĚR 90
SEZNAM POUŽITÉ LITERATURY .....ERROR! BOOKMARK NOT DEFINED.
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 96
SEZNAM OBRÁZKŮ 97
SEZNAM TABULEK 99
SEZNAM PŘÍLOH 100

ÚVOD

Žijeme ve skvělé době. Pokrokové, moderní, dynamické. Mění se prostředí, ve kterém žijeme, styl myšlení, lidé i technologie.

Dříve byl nejvyšší ukazatel profesní hodnoty inteligenční kvocient (IQ), poté kvocient emoční (EQ). Dnes však stále více výzkumů říká, že klíčovým faktorem úspěchu v našem rychle se měnícím světě, je kvocient adaptability (AQ).

Tuhle myšlenku potvrzuje i známé motivační rčení: „Improvise. Adapt. Overcome“. Já osobně s těmito myšlenkami naprosto souhlasím. V moderní době, nesmírně dynamické a rychlé době, se kladou stále větší nároky na vše. Na rychlejší auta, chutnější jídla v restauracích, pevnější, ale lehčí a pohodlnější obuv, inovativní řešení.

Doba elektrifikace, doba prvních počítačů, počátky internetu s následným vznikem protokolu WWW a prvních webových stránek (Web1.0), to vše už je dlouhé roky pryč. Hlavní iniciativu od přelomu tisíciletí určuje éra Web2.0. Od roku 2008 se na svět ale derou digitální měny a nyní už deset let decentralizované aplikace včele s technologiemi Ethereum, které radikálním způsobem mění dosavadní vnímání webového vývoje a vzbuzují otázku další inovace.

S těmito pokroky však přichází i nové výzvy. Moderní technologie, orientace ve světě informačních technoligií a schopnost s nimi pracovat, lákají však nejen nadšence po pokroku, ale také ty, kteří jsou motivováni nekalými úmysly. Tudíž nejen naše schopnost informační systémy zabezpečit, ale také um je prolamovat a zneužít, stále roste.

Své osvědčené místo na slunci mají známé metody zabezpečení, v čele s šifrováním a digitálním podpisem. Je však dle mého jen otázkou času, kdy i tyto metody nebudou samostatně dostatečné či méně vhodné. Je tedy nutné i v této oblasti neusnout na vavřínech a hledat nová, inovativní řešení zabezpečení našich dat. Vidinu naplnění této myšlenky slibují decentralizované technologie, možnost na nich programovat a svá data ukládat.

Ve své práci se proto nesnažím o inovaci tohoto století… teplá voda již byla vynalezena... Snažím se však vnuknout myšlenku nového přístupu: neomezovat svou volbu mezi výběrem jen osvědčených technik světa Web2.0 nebo naopak pouze přístup technologií světa Web3.0, jak tomu činí spousta moderních přístupů, ale mým zaměrem je nastínit možnou spolupráci obou těchto protipólů.

Hlavní motivací nebylo vytvořit dokonalou aplikaci jako takovou, ale implementovat pevný základ prostředí v čele se solidním zabezpečením aplikace a zajímavým mixem využitých technologií.

I. TEORETICKÁ ČÁST

1 WEB 2.0

První éra internetu, označována také jako web 1.0, může být charakteristicky shrnuta jako generace statických webových stránek [1].

Současná éra druhé generace internetu, tedy **web 2.0**, zastřešuje aktuální podobu internetu, tak, jak jej zná široká veřejnost, a která přinesla s sebou řadu přelomových vlastností. Umožňuje uživatelům generovat obsah na základě jejich interakce, nové způsoby sdílení dat (mimo jiné například cloudové uložiště [1]) a personalizaci obsahu – tedy generování obsahu každému uživateli na míru, dle relevantnosti na základě dat sesbíraných z doposud prohlíženého obsahu [1].

Velkého rozmachu se zde dočkaly sociální sítě, kde lidé hledají nejen zábavu, ale také vzdělání a aktuální informace o dění ze světa. Web 2.0 také přinesl nové možnosti pro vzdělávání a výuku, jako jsou virtuální třídy a distanční vzdělávání, a nové obchodní modely, jako jsou online reklamy a SaaS aplikace [2]. Web 2.0 lze rozlišit základním rozděleném na Frontendovou a Backendovou část [1].

1.1 Frontend (FE)

Za Frontend se označuje část webové prostředí, která má na starost zobrazení a zpracování uživatelské rozhraní (UI) a prvky se kterými uživatel může interagovat, jako jsou tlačítka, vstupy, či navigační menu. Frontend se také stará o zpracování dynamického obsahu, tedy ku příkladu překreslení UI bez nutnosti znovunačtení stránky, routing, validaci vstupních dat atd. Základními jazyky pro tvorbu frontendového obsahu jsou HTML, CSS a Javascript.

1.1.1 HyperText Markup Language (HTML)

HTML je základním značkovacím jazykem pro tvorbu struktury webových stránek. Syntaxe využívá značky – „tagy“, které mohou být párové i nepárové. Jsou význačné tím, že jsou píšou do hranatých závorek [2].

[1] **Cloudové uložiště** = webová služba, která umožňuje uživatelům pracovat se soubory na vzdálených serverech – „cloudech“
[2] **SaaS (Software as a Service)** aplikace = umožňují uživateli využívat cloudové aplikace přes internet (email, kalendářové a kancelářské aplikace atd.) [55]

1.1.2 Cascading Style Sheets (CSS)

Kaskádové styly jsou jazykem využívaný pro vizuální prezentaci webových stránek, například tvar, barvu, zarovnání, pozadí či animace jednotlivých prvků. Doplňuje tedy HTML grafickými změnami a spolu tvoří základní strukturu těchto stránek.

CSS pomáhají udržet jednotný vzhled a šetří mnoho času při vývoji, neboť změna ve stylu na jednom místě, se projeví ve všech prvcích, kde je tento styl použit.

1.1.3 Javascript (JS)

JavaScript je skriptovací jazyk 3 pro tvorbu dynamických a interaktivních webových stránek. Umožňuje **manipulovat s prvky v reálném čase**, například reagovat na události uživatele, jako je kliknutí na tlačítko, změna hodnoty ve formuláři a atd.

Může také vytvářet, měnit a manipulovat s HTML a CSS na webové stránce, což umožňuje vytvářet komplexní uživatelské rozhraní a zlepšovat uživatelskou zkušenost. Bez použití JS by většina stránek byla statického rázu, kdy jedním z mála dynamických prvků by byly CSS animace.

Dnes je Javascript jedním z nejpoužívanějších jazyků na světě a je základním stavebním kamenem pro webový vývoj – je součástí více než 98 % (údaj k březnu 2023) všech webových stránek [3]; [4].

Navíc, je také základem jak backendových technologií, například Node.js, tak nejpoužívanějších frontend frameworků, jako je React či Vue.js, což ještě přidává na jeho významu v tomto odvětví.

1.1.3.1 Node.js

Node.js je JavaScritové open-source multiplatformní prostředí, které umožňuje psát serverové aplikace. Díky němu lze psát JS kód, který běží přímo na počítači, který by jinak byl spouštěný pouze ve webových prohlížečích. Podporuje balíčkový systém npm (Node

[3] **Skriptovací jazyk** = interpretován přímo ve webovém prohlížeči a nevyžaduje tedy předchozí kompilaci kódu

Package Manager), který dále ulehčuje práci s JS knihovnami. NPM je de-facto standardem a obsahuje tisíce různých balíčků, které lze v JavaScriptových aplikacích využít.

Yarn je alternativní správce balíčků pro Node.js, který byl vyvinut týmy, jako jsou Facebook a Google. Cílem Yarnu bylo řešit řadu problémů, které vývojáři zažívali při používání npm, jako jsou problémy s rychlostí, bezpečností a spolehlivostí. Yarn nabízí několik výhod oproti npm. Je obecně rychlejší, protože využívá paralelní stahování balíčků, a má robustnější závislostní management. Kromě toho, Yarn používá yarn.lock soubor, který zajišťuje, že instalace balíčků jsou konzistentní mezi různými prostředími. Yarn je také kompatibilní s npm balíčkovým registrem, takže můžete používat jakékoli balíčky, které jsou dostupné na npm.

Express.js

Express.js je jednoduchý a flexibilní **framework pro Node.js**, který umožňuje rychlou a snadnou tvorbu webových aplikací a API. Je známý svou minimalistickou architekturou a robustním sadou funkcí pro vývoj webových a mobilních aplikací [5].

S pomocí Express.js lze vytvářet sofistikované webové aplikace s dodržením standardů **HTTP** a podporou různých **middlewares** pro snadnější manipulaci s požadavky a odpověďmi v HTTP cyklu. Express.js také umožňuje vývojářům definovat směrování URL, což je klíčové pro tvorbu webových stránek a API endpointů [5].

Express.js je **součástí „MEAN“** (MongoDB, Express.js, Angular, Node.js) a „MERN“ (MongoDB, Express.js, React, Node.js), které jsou oblíbenými technologiemi pro full-stack vývoj webových aplikací. Rozšířenost Express.js a jeho integrace s dalšími populárními nástroji a knihovnami z něj činí jeden z nejpoužívanějších a nejdůležitějších nástrojů v ekosystému Node.js [6].

1.1.4 Typescript (TS)

Typescript je **open-source** vyvinutá Microsoftem, a je **typovací nadstavbou nad Javascriptem**. Jeho hlavní funkcí je tedy přidání syntaxe, statických typů 4, tříd či rozhraní do Javascriptu, což dovoluje vývojářům vytvářet **přehlednější, robustnější, a hlavně bezpečnější kód** [7].

Bezpečnější proto, že minimalizuje chyby v kódu tím, že vyžaduje, aby proměnné a funkce byly používány v souladu s jejich definovanými typy. Předchází tak chybám statických typů a null hodnot 5, které mohou vést k neočekávanému chování programu.

Typescript je zároveň kompatibilní s jakýmkoliv funkčním JS kódem, což znamená i zpětnou kompatibilitu bez přepisování a možnost kombinovat oba zmíněné jazyky ve stejném projektu.

V posledních letech se Typescript stal stále populárnějším v oblasti webového vývoje, zejména v kombinaci s moderními frontendovými frameworky jako jsou React a Angular, které ho často doporučují pro své typové bezpečnostní funkce a lepší nástroje pro vývoj kódu.

1.1.5 Architektonické vzory

Architektonické vzory jsou obecně použitelné řešení pro běžné problémy v architektuře softwaru. Jsou to, zjednodušeně řečeno, šablony, které lze aplikovat při návrhu struktury systému pro dosáhnutí zejména modularity či škálovatelnosti SW.

Tyto vzory jsou nezávislé na konkrétních programovacích jazycích a jsou použitelné napříč jak různými typy aplikací (webové, mobilní, desktopové), tak platformami (mobil, desktop, tablet).

Existuje mnoho architektonických vzorů. Všechny mají jedno společné, a to cíl strukturovat kód do co nejčitelnější podoby a zároveň optimální pro testování a udržitelnost. Většina z nich má společné dvě hlavní logické části, Model a View, které bývají doplněny třetí

[4] **Statické typování** = datové typy proměnných a funkcí jsou známy již při kompilaci (ne až při běhu programu, jak je tomu u JS)
[5] **Null** = speciální hodnota, která představuje absenci či neplatnou hodnotu; lze ji také označit jako „žádná hodnota“, „žádný objekt“

logickou částí. Nejpoužívanější vzory jsou MVC (Model-View-Controller), MVVM (Model-View-ViewModel) a MVP (Model-View-Presenter).

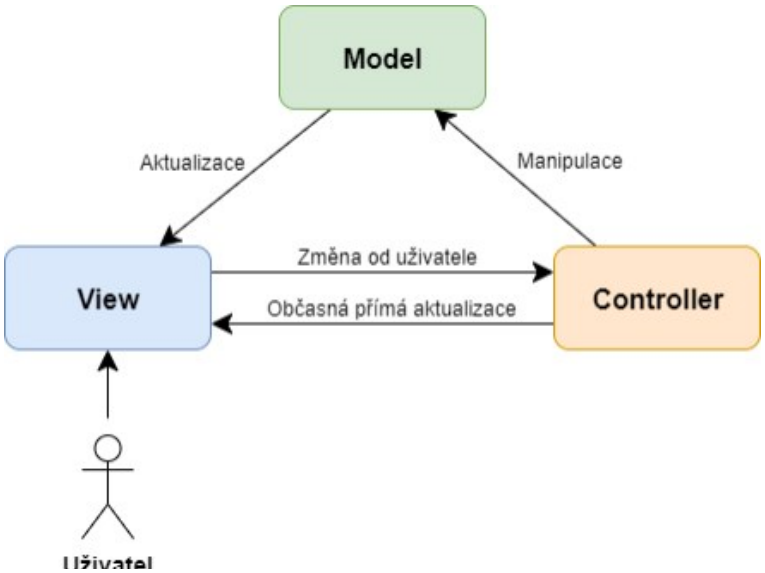
1.1.5.1 MVC architektura (Model-View-Controller)

MVC je velmi populární architektonický vzor, který rozděluje aplikace na tři hlavní logické části – Model, View a Controller.

Model je zodpovědný za definici množiny tříd (objektů), které popisují hlavní logiku dat a práci s nimi. Model může také obsahovat ViewModel, což je speciální typ Modelu, který sdružuje data do jedné třídy tak, aby poskytl soubor dat přizpůsobený potřebám svého View [8].

View na druhou stranu zahrnuje UI, tedy veškeré grafické komponenty, které jsou zobrazeny uživateli a se kterými interaguje.

Část **Controlleru** funguje jako spojovací článek mezi Modelem a View. Je zodpovědný za příjem vstupních požadavků od uživatele, manipulaci s daty v Modelu, ukládání změn do databáze, pokud je k aplikaci připojena a rozhodování o tom, které View se má uživateli zobrazit. V rámci MVC aplikace může být pro jediné View přidruženo klidně více Controllerů.



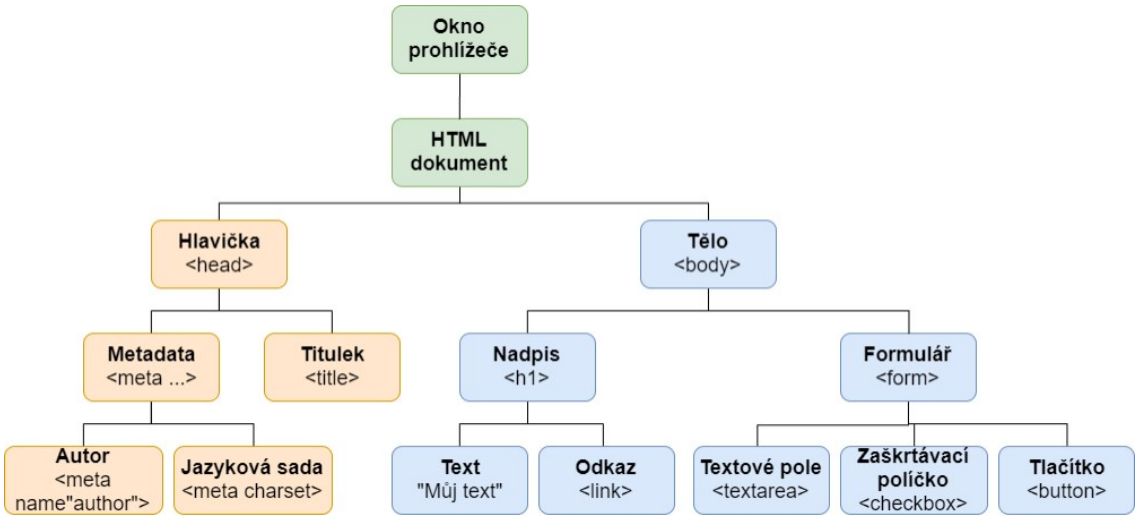
Obrázek 1: Architektonický vzor MVC

(zdroj: vlastní zpracování dle [61])

1.1.6 Document Object Model (DOM)

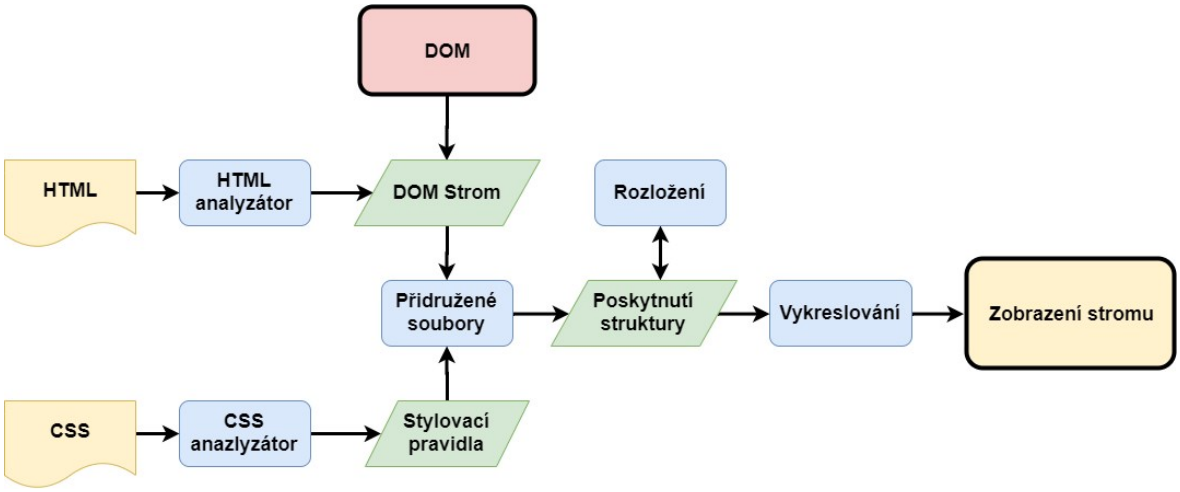
DOM je hiearchická **reprezentace dokumentu** (v případě webových stránek HTML prvků) **se stromovou strukturou**, kde každý prvek je reprezentován ve formě objektu. Umožňuje programům dynamicky přistupovat k jeho obsahu, struktuře a tím jej efektivně upravovat.

Po této změně nastává krizová část, kdy práci s touto změnou je velkou podstatou a rozdílem moderních technologiemi, které zpracovávají Frontendový obsah.



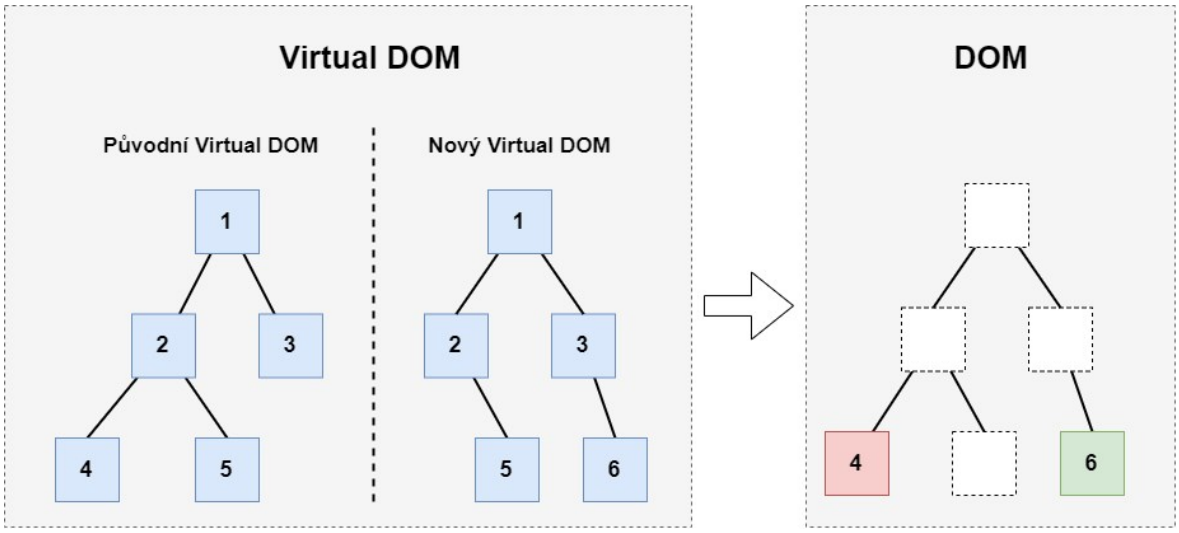
Obrázek 2: Grafické znázornění příkladného DOM
(zdroj: vlastní zpracování dle [2])

U základního (klasického) DOM při každé změně totiž nastává nový cyklus, kdy se nejprve analyzuje aktuální stav DOM, tedy HTML dokument, ze kterého se dále vytvoří aktualizované rozložení a obsah jednotlivých prvků, a nakonec se celá stromová struktura celá znovu vykresluje. Tento proces dosti časově náročný, tedy při dnešních vysokých uživatelských nárocích, nežádoucí.



Obrázek 3: DOM cyklus
(zdroj: vlastní zpracování dle [56])

Proto moderní Frontendové frameworky, jako jsou React, Angular či Vue využívají jiného, rychlejšího způsobu zpracování DOM. Asi nejznámějším a nejpoužívanějším typem DOM je „**Virtual DOM**", jež byl zpopularizován především Reactem, ale již řadu let je využíván například Angular 2 a Vue.js. Jak napovídá sám název, jedná se o virtuální reprezentaci DOM, který existuje v paměti, je **kopií aktuálního klasického DOM a umožňuje rychlejší aktualizaci obsahu**, bez nutnosti načítat opětovně celou stránku. Při změně obsahu se abstraktně vytvoří nový, tedy druhý DOM. Tyto dva DOMy (původní a aktuální) se porovnají, vyznačí se změny. Pro aktualizaci zobrazovaného DOM se následně vyrenderují pouze tyto změny. Tento proces se používá často pro jednostránkové webové aplikace - „**SPA**", kdy je nejvíce efektivní (proto je velká část React aplikací právě SPA) [9].



Obrázek 4: Virtuual DOM

(zdroj: vlastní zpracování dle [9])

Inkrementální DOM je speciálním typem DOM, který je využíván například v renderovacím jádru Angularu Ivy. Tento typ DOM umožňuje co nejlepší výkon aplikací na mobilních zařízeních. Komponenta zde interpretována přímo frameworkem, ale odkazuje na instrukce, což umožňuje při kompilaci odstranit nepoužité instrukce a tím snížit celkovou velikost aplikace pomocí "**tree-shaking**". Dále se paměť alokuje pouze pro změny v DOM, což ušetří paměť a zlepší celkový výkon aplikace [9].

1.1.7 React

React je Javascriptová knihovna (často nazývaný Javascript frameworkem), o jejíž **vývoj** se od jeho vzniku v roce 2013 stará **Facebook** (dnes už **Meta**). Je založen na unikátním architektonickém vzoru „flux“, který byl vyvinut společností Facebook a stará o správu stavu aplikace. Mezi nejznámější knihovny, které se starají o globální stav aplikace patří například Redux či Zustand. V případě Zustand je komunikace komunikace velmi obdobná komunikaci s lokálními stavy aplikace – „useState“ [10].

Oproti největším konkurentům jako jsou Angular a Vue.js, je React označován jako „**lightweight**“, protože se zaměřuje hlavně na – obstarává vrstvu View v architektuře MVC. Díky tomu je React modulární, a především velmi datově malý na rozdíl od konkurenčních frontend frameworků. Pracuje se speciálním typem JS (kombinace JS a HTML) – „JSX“.

Díky tomu, že používá renderování svých komponent **Vitrual DOM**, je velmi vhodný pro vývoj aplikací typu **SPA** [11].

React je „**component-based**“, který razí heslo (konkrétně React Native): „learn once, use everywhere“, tedy metodu znovupoužitelných malých komponent a multiplatformního přístupu. Tuto filozofii ještě dříve podporovalo rozdělení komponent dle jejich velikosti – „Atomic design“, tj. nejmenší atomy – například tlačítko, nadpis; větší molekuly (skládající se z atomů) – například menu položek v hlavičce; ještě větší organismy (skládající se z atomů a molekul) – například celá hlavička; až po největší stránky. Tento způsob rozdělení komponent zajišťuje velkou přehlednost, snadnou a efektivní práci. Od tohoto stylu se ale v poslední době i mezi React vývojáři odchyluje, protože je velmi striktní a v některých případech náročný na implementaci [12].

Obrázek 5: Atomic design
(zdroj: [57])

1.1.8 Styled-components

Styled-components je **CSS knihovna**, která je speciálně **navržena pro React**. Díky této knihovně mohou vývojáři vytvářet styly pro své komponenty pomocí tzv. "tagged template literals", což umožňuje využívat Javascriptové proměnné přímo v CSS. To umožňuje jednoduché a dynamické měnění stylů v závislosti na stavu aplikace. Navíc se tím zjednodušuje tvorba znovupoužitelných komponent, protože lze snadno předávat proměnné jako vstupy pro styly. Tento přístup umožňuje vývojářům v Reactu velmi efektivně tvořit a spravovat komponenty se specifickými styly.

1.2 Backend (BE)

Backendová, neboli serverová část aplikace, je oblast zaměřená na správu, ukládání a přenos dat mezi uložištěm (s po většinou databází) a práci a API [6], které umožňuje komunikaci mezi jednotlivými částmi systému.

Backend také zpracovává požadavky pro frontendovou část aplikací. Tyto požadavky mohou zahrnovat získávání, aktualizaci, vytváření či mazání dat. Další klíčovou rolí backendu je zabezpečení systémů, neboť na rozdíl od frontendové části probíhá na serveru, tj. mimo nenásilný dosah uživatele, kde je možné implementovat silné bezpečnostní mechanismy, aby byla data chráněna.

1.3 Databáze

Nedílnou součástí backendu jsou databáze, tedy strukturované soubory nebo systémy, jež jsou navrhnuty k ukládání a organizaci a vyhledávání různorodých dat. Jsou základním stavebním prvkem všech moderních informačních systémů. Umožňují rychlý a efektivní přístup k uloženým datům, podporují jejich sdílení mezi uživateli či systémy a starají se o jejich zabezpečení.

Existuje mnoho typů databází, včetně relačních, hierarchických a objektově orientovaných. Relační databáze, které používají tabulky a vztahy mezi nimi k organizaci dat, jsou nejběžnějším typem.

1.3.1 SQL databáze

SQL (Structured Query Language) databáze jsou nejčastějším typem relačních databází. Používají jazyk SQL (Structured Query Language), tedy strukturovaný dotazovací jazyk – SQL, které má předdefinované schéma. Data ukládá do zpravidla do tabulek.

SQL databáze jsou navíc vertikálně škálovatelné, což znamená, že mohou přidávat více výkonu tím, že přidávají více zdrojů, jako je CPU, paměť nebo uložiště, k jedné instanci databázového serveru [13].

[6] **API (Aplication Programming Interface)** = sada protokolů a nástrojů pro aplikace a jejich komunikaci

Mezi výhody největší SQL databází patří přehlednost dat, protože jsou zažitě zobrazování v n-ticích, schémata jsou normalizovaná do tabulek a dají se efektivně slučovat SQL dotazy. Takovým příkazem SQL dotazu může být například:

```
SELECT
    Kniha.Název,
    Kniha.Rok_Vydání,
    COUNT(Kniha.Verze) AS Počet_Verzí,
    Autor.Jméno AS Auro
FROM
    Kniha
INNER JOIN
    Autor ON Kniha.Autor_ID = Autor.ID
WHERE
    Kniha.Rok_Vydání > 2000
GROUP BY
    Autor.Jméno,
    Kniha.Název,
    Kniha.Rok_Vydání
```

1.3.1.1 NoSQL databáze

NoSQL (**Not only SQL**) databáze jsou databáze, které nepoužívají relační model, ani SQL jazyk a mají **dynamické schéma s nestrukturovanými daty** – lze přidávat různé druhy záznamů do stejného pole (kolekce).

Byly navrženy pro provoz s velkými **horizontálně škálovatelnými clustery**, proces také nazývaný „sharding“, což je koncept, kterým se vyznačuje systém, jež dokáže zvládnout větší zátěž rozdělením data a požadavků na více serverů. Tento přístup velmi přínosný pro práci s velkými datovými soubory a zpracování vysokého množství dotazů [13].

Existuje několik typů NoSQL databází. Typ klíč-hodnota a Dokumentové databáze jsou těmi nejpoužívanějšími. Dále jsou to Grafové databáze a Sloupcové databáze.

Databáze typu klíč-hodnota je nejjednodušší a velmi flexibilní typ databáze. Data jsou ukládána jako pár klíče a hodnot, a lze si je představit jako klasickou hash tabulku. Klíč vystupuje jako jedinečný identifikátor, který slouží k vyhledávání nebo adresování hodnot

v těchto databázích. Tento typ je velmi rychlý a snadno škálovatelný, a tak se hodí především pro databáze s velkým množstvím zápisů dat o menších datových velikostech. Představiteli těchto databází jsou například Oracle NoSQL databáze, Redis, či DynamoDB od Amazonu [14]; [15].

V **dokumentových databázích** jsou data uložena jako dokumenty, zpravidla v samo popisujících, stromových strukturách (vnořené asociativní matice, apod.), obvykle ve formátu **XML** nebo **JSON**. Tento typ databází je vhodný pro práci s komplexními daty o velké datové velikosti či s přímou vazbu na reálné dokumenty. Příkladem je **MongoDB** [15].

Grafové databáze jsou navrženy tak, aby manipulovali s daty, která jsou přirozeně uspořádány jako grafy. Zaměřují se na ukládání entit a vztahů mezi nimi. Jsou vhodné především pro aplikace vyžadující práci s komplexními a hlubokými vztahy, jako nalezneme u doporučovacích systémů nebo sociálních sítí. Konkrétním příkladem je databáze Neo4j [16]; [15].

V **sloupcových databázích** jsou data organizována, atypicky vůči zažitým metodám, do sloupců místo řádků. Díky tomu poskytují efektivnější práci s daty na úrovni sloupce, což je vhodné pro analytické úlohy. Příkladem je Apache Cassandra [15].

Existují také NoSQL nadstavby relačních databází, které kombinují prvky klasických relačních databází s NoSQL funkcemi.

Tabulka 1: Porovnání SQL a NOSQL

(zdroj: vlastní zpracování dle [13]; [14]; [15]; [16])

Oblast (kritérium)	SQL databáze	NOSQL databáze
Model dat	Relační, strukturovaný	Ne-relační, nestrukturovaný
Jazyk	SQL	Různé (dle typu NOSQL a konkrétní db)
Schéma	Statické	Dynamické
Typy	Tabulkové (Relační)	Klíč-hodnota, dokumentové, grafové, sloupcové
Škálovatelnost	Vertikální	Horizontální (sharding)
Vhodné pro	Transakční zpracování, komplexní dotazy	Velké datové soubory, obrovské množství dotazů, flexibilní struktury
Příklady databází	MySQL, MS SQL Server, Oracle, PostgreSQL	MongoDB, DynamoDB, Neo4j, Oracle NoSQL, Apache Cassandra

1.3.1.2 MongoDB

MongoDB je NoSQL databázový systém, navržený pro **ukládání nestrukturovaných dat**. Poskytuje vysokou flexibilitu při změně datových modelů, kdy není za potřebí žádná migrace těchto modelů. Dokáže ukládat data v **BSON** (binrání reprezentace JSON) a jeho výhodou oproti klasickému JSON je, že dokáže zpracovat více datových typů. Vyznačuje se rychlou odezvou a dobrou škálovatelností, kdy umožňuje snadné přidání nových serverů do databázových clusterů pro schopnost zpracování zvýšeného objemu dat a provozu. Ukládá data jako záznamy do „kolekcí“ (obdoba SQL tabulek, ale umožňuje ukládání dat s různými strukturami) a „dokumentů“ (obdoba řádků v tabulkách). Díky těmto vlastnostem se MongoDB často používá při práci s moderními webovými aplikacemi, protože poskytuje efektivní a rychlé zpracování dat [17].

2 WEB 3.0

Nastupující generace internetu, označována jako web 3.0, představuje novou éru online prostředí. Je přirovnávaná k „sémantickému webu“ či „inteligentními webu“, který klade důraz na strojové zpracování informací, poskytování sofistikovanějších a personalizovanějších online zážitku a přesun kontroly dat na stranu uživatele.

Klíčovou roli tohoto přechodu ztvárňují **blockchainové technologie**, které poskytují infrastrukturu pro **decentralizované** a bezpečné interakce v digitálním prostoru.

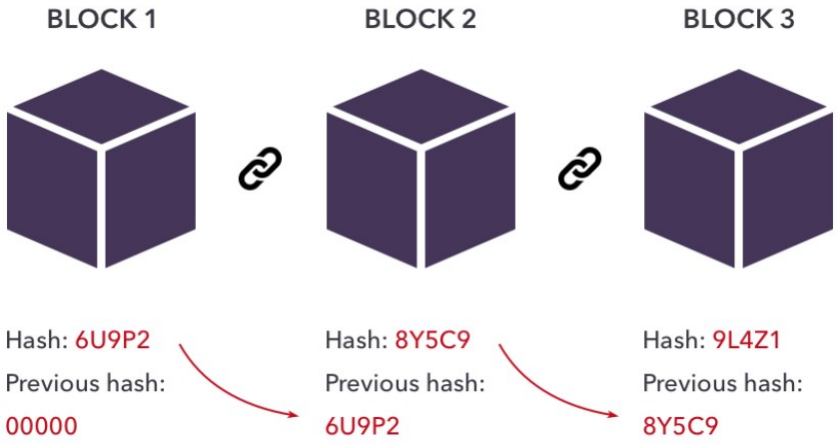
Tato decentralizovanost je zároveň základním stavebním kamene decentralizovaných aplikací – „dApps“, které hrají prim ekosystému web 3.0. Posilují **transparentnost** a zvyšování důvěry uživatelů a budí pocit bezpečnějšího webového prostředí.

2.1 Blockchain

Blockchain, je **decentralizovaná databáze**, a jak již samotný název napovídá skládá se z řetězce bloků. Velikost bloku, tedy množství dat, které lze do daného bloku zapsat se liší v závislosti na použité síti. V současné době existují stovky různých blockchainových sítí. Mezi ty nejznámější (nejpoužívanější patří Bitcoin, Ethereum, Solana, Cardano či DogeCoin).

Každý blok je po naplnění své kapacity připojen za blok předešlý, pomocí odkazu na jeho hash [7] odkazem na jeho **hash**, a umístěn do své sítě, kde spočine již natrvalo. Jakýkoliv zápis nelze nijak smazat či upravit, jedná se tedy o systém zápisu typu „**append-only**“, takže v principu se jedná o bezpečnější práci s daty, než je tomu u centralizovaných řešení [18].

[7] **Hash** = jednosměrná (zpětně nelze dekódovat) kryptografická funkce pro transformaci vstupu na pevně dlouhý výstup



Obrázek 6: Řetězení bloků dat v blockchainu

(zdroj: [58])

Pro zjednodušení je možné si představit účetní knihu, která nese transakční informace.

Tato účetní kniha je veřejná (pokud se však nejedná o soukromou blockchainovou síť, určenou obvykle pro korporátní organizaci), a tak jsou i data v ní uložena stále k dipozici všem k nahlédnutí i zápisu. Entity zapisující do těchto sítí jsou ale zpravidla známy „pouze“ adresou blockchainové peněženky (je zapsána v transakci), ke které byly přihlášeni, kterou podepsali danou transakci dat a zaplatili příslušný poplatek.

2.2 Výhody a nevýhody Web3.0 vůči Web2.0

Oproti světě Web2.0 se ve světě web3.0 využívá například decentralizovaného uložiště. Hlavní výhodou Web3.0, a speciálně blockchain technologie, je její "**append-only**" charakter ukládání dat. Jakmile jsou data zapsána do blockchainu, nelze je změnit nebo smazat, ale lze k nim pouze přidávat nové informace. Tento aspekt přináší výraznou transparentnost a sledovatelnost změn, což je obzvláště užitečné v situacích vyžadujících důkaz o autenticitě a integrity dat.

Uložení dat ve světě Web2.0 je většinou odkázáno na klasickou databázi, jejíž data jsou z principu upravitelná a geografické uložení serveru je soustředěno do často jednoho místa. I

v případě využití několika záloh nepřesáhneme spolehlivé redistrubuce[8] nebo redistribuované hostování webové stránky, které využívá mimo jiné i Cloudfront od AWS, jsou to však **centrální autority**, které poskytují nadstandartní služby.

Web3.0 řeší tuto problematiku prostřednictvím použití blockchainu. Ten ukládá data na **stovky uzlů** - "nodes", čímž poskytuje vyšší úroveň redundance a odolnosti vůči výpadkům. Tento decentralizovaný přístup zvyšuje bezpečnost a dostupnost dat, a zároveň redukuje riziko, že by data byla narušena nebo ztracena.

Přestože Web3.0 nabízí mnoho výhod, existují také některé nevýhody. Například může být náročné udržovat a spravovat tak rozsáhlé a distribuované systémy. Navíc, přestože blockchain zvyšuje bezpečnost, může také zkomplikovat procesy údržby dat a může vyvolat otázky týkající se soukromí a regulací. [19]; [20].

Tabulka 2: Porovnání výhod Web2.0 a Web3.0
(zdroj: vlastní zpracování dle [19]; [20])

Oblast (kritérium)	**Web2.0**	**Web3.0**
Data	Po uložení upravitelná (-) riziko fraud změny (+) možnost chtěné změny	Po uložení NEupravitelná (+) menší riziko napadení dat
Centralizace	Centrální autorita: (+) plná moc nad daty (odborná správa) (-) plná moc nad daty (riziko zneužití)	Decentralizace (+) vyšší odolnost vůči útokům (-) není zodpovědná entita v případě napadení systému
Bezpečnost	Závislost na centrální autoritě, možná úprava uložených dat, nutná robustní opatření • Věrohodná autorita: (+) • Nevěrohodná autorita: (-)	(+) zvýšená bezpečnost dat díky decentralizaci (+) složitější napadnutí dat
Správa a údržba	(+) méně náročná, díky centralizaci	(-) náročná správa dat a distribuovaných systémů
Redundance	Nutnost extra služby - globální redistribuce serverů	Automatická redistribuce dat do uzlů (+) odolnost vůči výpadkům

[8] **Interplanetary File System (IPFS)** = protokol pro ukládání a sdílení dat v distribuovaném souborovém systému

	• Bez distribuce: riziko výpadků (-) • S distribucí: (+)	
Soukromí a regulace	Závislost na centrální autoritě, uchování citlivých dat • Věrohodná autorita: (+) • Nevěrohodná autorita: (-)	Bez centrální autority, uživatel známý pouze adresou peněženky • Anonymita: ochrana osobních údajů (+) • Anonymita: riziko zneužití (-)
Efektivita	(+) zpravidla rychlé vyřízení požadavku	(-) rychlost vyřízení požadavku závislé na rychlosti zápisu bloků dat do dané blockchainové sítě

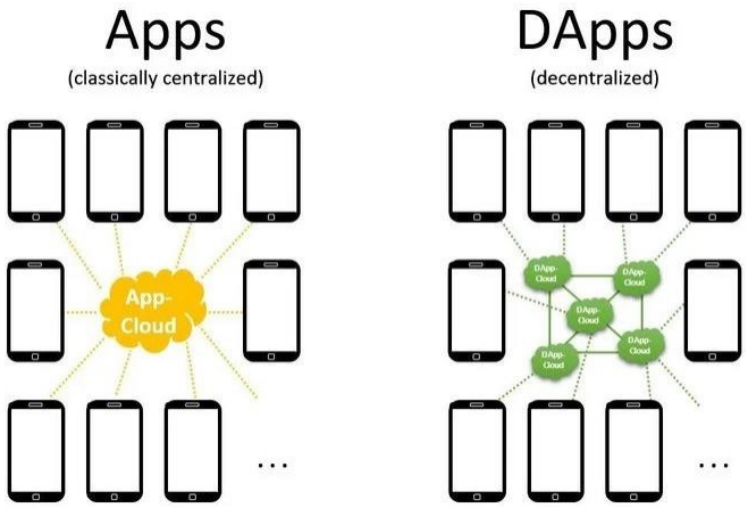
2.3 Využití blockchainových technologií

Prostředí web2.0 doplněné o blockchainové technologie se říká web3.0. Tyto technologie nacházejí stále širší využití v různých oblastech a spektrem aplikací.

2.3.1 Decentralizované aplikace (Dapps)

Decentralizované aplikace, známé jako Dapps, jsou jedním z významných využití blockchainu. Dapps běží na decentralizované síti, což znamená, že jejich backend není umístěn na centralizovaném serveru, ale je distribuován mezi **peer-to-peer** [9] počítači (**nody**). Tím se odlišují od klasických aplikací webu2.0. Klientská část Dapps může být napsána ve stejných jazycích a frameworkách jako běžné aplikace. Nejznámější blockchainovou platformou pro vývoj Dapps je Ethereum, které drží prvenství v počtu těchto aplikací. Nicméně existují i další platformy, jako Solana, Polkadot, EOS a další, které nabízejí možnosti vývoje a provozování Dapps.

[9] **Peer-to-peer (P2P) =** decentralizovaná síť bez centrální autority; účastnící komunikují mezi sebou napřímo (bez prostředníka)



Obrázek 7: Apps vs Dapps

(zdroj: [59])

2.3.2 Non-Fungible Token (NFT)

Nezaměnitelný token lze chápat jako unikátní ID ve formátu uint256, ve standardu ERC-721, či aktualizované verzi od vývojářů Azuki, **ERC-721A**, jehož hlavním přínosem je umožnění ceny „mintování“ 10 několika tokenů najednou, za obdobnou cenu jako mintování tokenu jediného. NFT je vedle kryptoměn v nynějším světě hlavním důvodem povědomí širší veřejnosti o blockchainových technologiích.

Většině lidí je známo využití NFT pro data ve formátu obrázků (.jpg). Tato technologie ale nabízí mnohem širší využití – dá se použít prakticky všude, kde je potřeba jednoznačný identifikátor. V současné době již existují projekty, kdy jsou NFT aplikované na textové dokumenty, zvukové záznamy – i web3.0 obdoby Spotify. Dále zmíněné předměty v kolekcích, rozsáhlé uplatnění nachází také v různorodých formách v herním odvětví, dále jako přístupové klíče, číslovaná sedadla do divadla/sportovní utkání, záznamy o pacientovi v nemocnici, vlastnictví pozemku a další.

[10] **Mint** = proces vytváření nového tokenu

Obrázek 8: NFT z kolekce Bored Ape Yacht Club

(zdroj: [60])

Protipólem NFT je Multi-Token Standard, který umožňuje vyrobit více kusů (kopií) daného tokenu, používá se ve standardu s označením ERC-1155.

2.3.3 DeFi

V tradičním finančním systému, jak ho známe, je naše ekonomika řízena řadou centralizovaných institucí. Vedle jistých výhod má tento systém pochopitelně i několik nevýhod. Bankéři, právníci a úředníci mají vedle zákonů významný vliv na konečná rozhodnutí, a tedy i nezanedbatelnou kontrolu nad našimi financemi.

Služby poskytované centralizovanými finančními institucemi mohou být cenzurované, vyžadují komplikovaná povolení, jejich podmínky často nejasně definované a jejich poskytnutí je často finančně náročné, zčásti kvůli nákladům na lidskou práci.

Na druhé straně máme **decentralizované finance**, známé jako DeFi. Tento systém je založen na třech základních pilířích: kryptoměně, blockchainu, který slouží jako databáze (případně v kombinaci s tradičními databázemi připojenými na blockchain - tzv. "Oracles"), a Smart Contractech, ve kterých často vystupují DeFi protokoly jako Compound nebo AAVE.

DeFi pokrývá pět hlavních oblastí: stabilní kryptoměny (stablecoins), okamžité půjčky (flash loans), měnové směny, pojištění a obchodování s pákou. Všechny tyto oblasti představují nový přístup k finančním službám, který je ve srovnání s tradičním systémem otevřenější a přístupnější.

2.3.4 Decentralizovaná Autonomní Organizace (DAO)

Decentralizované autonomní organizace jsou obchodní struktury, kde je **rozhodovací pravomoc rozložena mezi všechny její členy**, namísto toho, aby byla soustředěna v rukou jen několika vybraných jedinců či dokonce pouze jediné entity.

V DAO hraje klíčovou Smart Contract, který musí zajišťovat v zásadě autonomní chod organizace. Tímto způsobem je eliminována obava z nedůvěryhodnosti správců organizace nebo možnosti, že by její činnost byla zastavena vyšší státní správou, jak je to možné v tradičním světě.

Dalším významným přínosem je fakt, že DAO je obvykle open-source 11. To znamená, že kdokoli může nahlédnout do kódu a navrhnout opravy existujících nedostatků. Ani její vývoj tak není závislý na určité skupině vývojářů, ale klidně celé komunitě.

DAO se postupem času rozvíjí a expanduje, přičemž jeho obsah se zvětšuje. Pravidla DAO jsou předem pevně dána a členové organizace mohou ovlivnit její chod pouze navrhováním a hlasováním o jednotlivých změnách. Například, v DAO se může nacházet několik členů, z nichž každý vlastní určitý počet hlasovacích tokenů – čím více tokenů, tím více hlasů. Může existovat třeba seznam položek, řekněme nejlepších restaurací v různých krajích. Každý člen DAO může navrhnout novou restauraci ke vstupu do tohoto seznamu a má právo hlasovat o jejím přijetí. Perioda pro přidání další položky nebo nových členů do seznamu jsou také pevně stanovena.

Tyto organizace představuje vidinu slibné inovace. DAO ale také však **nezajišťuje rovnoprávnost všech členů** – členové s velkým počtem DAO tokenů totiž často díky nim mají větší pravomoc, díky více rozhodovacím hlasům. Stále se ovšem jedná o poměrně nový, nezažitý koncept a existují k němu různé právní a regulační nejasnosti.

2.3.5 Decentralizované sdílení záznamů

V současné době jsou různé záznamy v tradičním systému, například zdravotní informace, rozptýleny mezi mnoho institucemi. V příkladu našich zdravotních záznamů, jsou zpravidla rozděleny mezi obvodního lékaře, zubaře a další zdravotnické specialisty. Toto roztřídění

[11] **Open-source =** SW, jehož zdrojový kód je volně přístupný a umožňuje svobodnou modifikaci a redistribuci

může v několika případech komplikovat proces získávání kompletního obrazu o zdraví pacienta.

Zde blokchainové technologie přináší jistou alternativu, návrh řešení. Při jeho využití mohou být totiž jeho záznamy zároveň centralizované a bezpečně uloženy na jednom místě, a to při zachování integrity záznamů. V případě, kdy lékař potřebuje získat kompletní zdravotní stav a historii pacienta (řekněme i například v případě, že pacient není schopen kvůli zdravotním omezením obstarat výpis záznamů od jednotlivých institucí), může přistoupit k danému blockchainu s pomocí unikátního ID či blockchainové adresy pacienta.

2.3.6 Internet of Things (IoT)

Fúze dvou progresivních technologií – IoT a blockchainu, má rozhodně potenciál širokého využití. Blockchain může přidat potřebnou vrstvu bezpečnosti do IoT sítí a snížit tak jisté obavy z potenciálních bezpečnostních útoků na data či síť jako takovou. Navíc, díky přirozené decentralizaci a široké globální síti blokchainových nodů, může dojít ke zrychlení procesů komunikace mezi prvky IoT, jelikož data nemusí být přenášena ze vzdálených serverů.

2.4 Validační mechanismy

Různé blockchainové sítě používají různých validačních mechanismů, též označovaných jako konsenzuální mechanismus, pro validaci příchozích transakcí, jejich přidání do sítě a vytvoření nových tokenů. Mezi nejznámější patří **Proof of Work, Proof of Stake, Proof of Concept a Proof of History**.

2.4.1 Proof of Work (PoW)

Proof of Work je tradiční, nejstarším krypto konsenzem, prvně, a dosud používaný pilotní blockchainovou sítí – **Bitcoinem**. Původně také Ethereem (**Ethereum 1.0**) a dále Dogecoinem či Litecoinem. Tento typ validace je úzce spojen s těžením – „**mining**“. Název je skutečně výstižný, protože požaduje **obrovské množství výpočetního výkonu** [21].

Blockchainu, využívající tento typ validace, spoléhají na validátory, kteří soutěží o to, kdo první vyřeší verifikační, velmi složitý matematický problém. Vítěz poté aktualizuje síť

s touto poslední verifikovanou transakcí a je odměněn ve formě předdefinované částky dané kryptoměny.

PoW má tu výhodu, že je léty prověřeným a robustním typem validace. Rozrůstající se sítě lákají více validátorů – těžařů, což zvyšuje sílu a bezpečnost. Běžným zvykem je, že těžaři spojují své (výpočetní) síly, čímž zvyšují svou celkovou výpočetní sílu, a tedy i šanci na jejich selekci pro validaci dané transakce. Výsledný zisk si poté rozdělují.

Pro tento fakt je pro jednotlivce či skupiny, disponující slabým výpočetním výkonem, takřka zbytečné ucházet se o pozici vybraného validátora.

Obrovskou nevýhodou, a častým argumentem nepříznivců Proof of Work, je obrovská energetická náročnost, která se váže na spotřebovanou výpočetní. Podle odhadů je spotřeba elektřiny potřebná pro těžbu Bictoinu srovnatelná s celkovou spotřebou některých států [22]!

Teoretickým rizikem zde je „problém 51 %. Ten nastává, když útočník či skupina útočníků kontroluje většinu, tedy více než 50 % celkového výpočetního výkonu sítě. Útočník by v takovém případě mohl provést “double spending”, což znamená, že by mohl utratit svou kryptoměnu vícekrát a zneužít tak svého nadměrného vlivu v síti [22].

2.4.2 Proof of Stake (PoS)

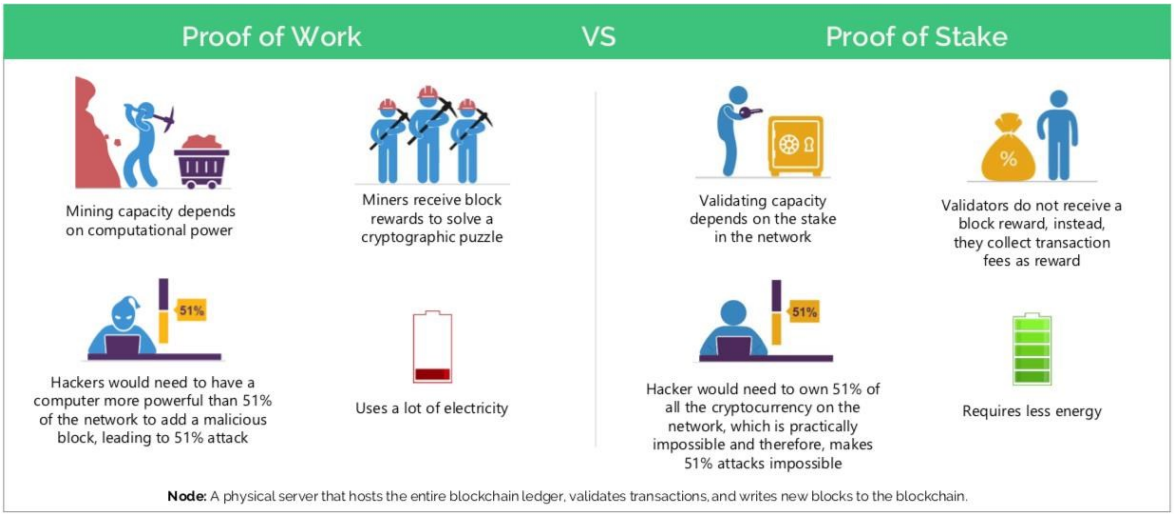
Proof of Stake je inovativní konsenzový algoritmus, vznikl jako alternativa PoW a prvně byl implementován v kryptoměně Peercoin. Postupně se dostal do širšího povědomí a s rostoucím zájmem o udržitelné kryptoměnové platformy.

Na rozdíl od Proof of Work, při validaci se neřeší složité matematické problémy, takže se ušetří spousta elektrické energie. Jmenuje se validátor na základě množství zastavených tokenů – proto „**stake“ a náhodného procesu výběru**. Validátor je navíc motivován ztrátou svých zastavených tokenů v případě své špatně provedené validace. Pokud však validuje transakci správně, dostane se mu odměny ve formě tokenů (tentokrát ne přímo od adresy sítě, jako je tomu u PoW), za transakční poplatky „gas fees“, které platí uživatel při podepisování transakce.

Myšlenkou k zamyšlení se zde nabízí centralizace sítě, v případě koncentrace zastavených tokenů u malého počtu validátorů, což jde protipólem základní filozofii blockchainových technologií – decentralizovanost. Rizikem Proof of Stake je také takzvaný "nothing at stake" problém. Ten spočívá v tom, že validátoři mohou teoreticky validovat více verzí

blockchainu, neboť za to nejsou penalizováni, což by mohlo vést k chaosu a nejistotě v síti. Avšak, moderní implementace PoS, jako je Ethereum 2.0 či Solana, představují různé mechanismy k řešení tohoto problému [22].

Problém 51 % zde nehrozí (tedy opět ano, ale ještě ve více teoretické rovině), protože pro provedení takového útoku by musel validátor vlastnit nadpoloviční většinu všech tokenů také sítě. PoS přístup se nachází mimo **Ethereum 2.0** (ETH) a **Solany** (SOL) také u **Cosmos** (ATOM), **Polkadot** (DOT), **Avalanche** (AVAX) či **Binance Smart Chain** (BNB) [23].



Obrázek 9: PoW vs PoS grafické znázornění

(zdroj: [22])

Tabulka 3: Porovnání PoW a PoS
(zdroj: vlastní zpracování dle [21]; [22]; [23])

Oblast (kritérium)	Proof of Work (PoW)	Proof of Stake (PoS)
Výběr těžaře/validátora dle	Výpočetního výkonu (složitý matematický problém)	Množství zastavených tokenů a náhodného výběru
Odměna (tokeny)	Předdefinovaný počet přímo ze sítě	Transakční poplatky uživatelů
Energetická náročnost	Velmi vysoká	Nízká (u Etherea 2.0 cca 0.01 % z Ethereum 1.0)
Bezpečnost	Robustní, vyžaduje >50% výpočetní výkon pro útok	Robustní, vyžaduje >50% všech tokenů pro útok
Příklad sítě	Bitcoin, Ethereum (do verze 1.0), Litecoin, Dogecoin	Ethereum (od verze 2.0), Solana, Cosmos, Polkadot

2.5 Bitcoin

Vznik blockchainu i pilotní a zároveň pro širokou veřejnost nejznámější blockchainové sítě, Bitcoinu, se datuje do roku **2008**. Autorství této kryptoměny se připisují osobě nebo skupině vystupující pod pseudonymem **Satoshi Nakamoto** (Satoshi je zároveň jednotka Bitcoinu 12).

Bitcoin se stal odrazovým můstkem pro digitální, a především blockchainový a kryptoměnový svět. Jeho decentralizovaný charakter a odolnost vůči cenzuře přinesly nové možnosti finančních transakcí a způsobu uchování hodnoty.

Bitcoinová síť využívá konsenzu **Proof of Work**, byla původně navržena a až donedávna sloužila pouze jako kryptoměna (BTC), tedy jako investiční aktivum. Na přelomu roku 2022 a 2023 se ale začali využívat incripce 13 k vytvoření „Ordinals“ – obdoba NFTs 14) [24]; [25].

[12] **Satoshi** = nejmenší jednotka Bitcoinu; 1 Satoshi = 0,00000001 BTC
[13] **Inscripce** = transakce a jejich poplatky síti Bitcoinu
[14] **NFT** (Non Fungible Token) = jedinečný digitální token, který reprezentuje garantuje unikátnost daného záznamu

2.6 Ethereum

Ethereum je další velmi významnou blockchainovou platformou a zároveň také kryptoměnou, nazývanou Ether (ETH). Bylo představeno v roce 2013 díky tvůrci **Vitaliku Butterinu**. Ethereum bylo oproti Bitcoinu vytvořeno jako plně programovatelná platforma, jež umožňuje vývoj „**smart contractů**“, zjednodušeně řečeno programů a tím pádem i decentralizovaných aplikací – „**Dapps**“. Ethereum, tak jako i ostatní programovatelné platformy, využívají tokeny své kryptoměny (v tomto případě ETH) na zaplacení poplatků za jednotlivé transakce, operace v síti [26].

Od svého vzniku až po září loňského roku (přesněji 15. září 2022) bylo Ethereum postaveno na konsenzu Proof of Work. Přechodem na konsenzus **Proof of Stake nyní šetří zhruba 99.95 % své původní spotřeby energie**. Tato změna přinesla řadu výhod, jako je vyšší škálovatelnost, rychlejší zápis transakcí a nižší provozní náklad, včetně poplatků, a i díky tomu, patří Ethereum stále za jednu z nejpevněji postavených blockchainových platforem na trhu [26]; [27]; [28] .

2.7 Blockchainové sítě

Existuje již řada typů blockchainových sítí, společné mají řadu vlastností, a zároveň se ve spoustě z nich navzájem odlišují. Každá je vhodná pro jiný případ užití.

2.7.1 Hlavní síť (Mainnet)

Na hlavních blockchainových sítích se nacházejí produkční verze smart contractů. Tato síť, která je plně vyvinuta, plně funkční, poskytuje kompletní funkcionality daného blockchainu. Umožňuje provádění skutečných transakcí v reálném, ostrém provozu s přenosem ETH [29]; [30].

Při ukládání transakcí hrají hlavní roli těžaři (v případě využití konsenzu Proof of Work) a validátoři (v případě využití Proof of Stake – od září 2022 i případ Ethereum blockchainu).

2.7.2 Řešení 2. vrstvy (Layer-2 solutions)

Tato řešení jsou technologie, které se vyvíjí, abstraktně řečeno, nad hlavní vrstvě. Je zapotřebí, protože každá síť má omezenou kapacitu a ani hlavní síť Etherea není výjimkou. Proto byly vyvinuty další **vrstvy nad tou hlavní**, které jim **odlehčují práci a zrychlují celkový rychlost zapisování transakcí**. Řešení vyšších vrstev tedy přispívá ku zlepšení škálovatelnosti a snížení nákladů za transakce. Průměrný poplatek za transakci bývá zhruba 10x nižší ve vrstvě 2 (zhruba $0.104) oproti průměrnému poplatku hlavní vrstvy (dnešním dnem, 11. 5. 2023 dokonce 20x nižší oproti hlavní vrstvě, $2.769) [31]; [32].

Tyto vrstvy vytvářejí podsítě, které mohou zpracovávat větší množství transakcí, vytvářejí však pouze souhrny transakcí, které následně zapisují zpětně do hlavního řetězce. Mezi nejznámější druhé vrstvy v případě Etherea jednoznačně Polygon (MATIC), další známe sítě jsou Mantle, Optimism, Arbitrum a Gnosis. Pro porovnání, tržní kapitalizace u Polygonu je takřka 7,9 miliard dolarů, zatímco u druhého nejvyššího, Mantle, je to pouze 672 milionů dolarů (k 13. únoru 2023) [33]; [34].

2.7.2.1 Polygon

Polygon (používá token s názvem **MATIC**) je nejpoužívanější řešení druhé vrstvy nad Ethereem. Jeho markantní výhodou je, že je **plně kompatibilní s hlavní Ethereum sítí**, což znamená, že vývojáři mohou použít stejné nástroje a infrastrukturu. To usnadňuje přechod a integraci mezi oběma platformami. Polygon také podporuje více různých typů blockchainů, což ještě přidává flexibilitě a různorodosti aplikací [35].

Na druhou stranu, jako všechna technologická řešen, má Polygon i své nevýhody. První z nich se týká zabezpečení. Ačkoli používá vlastním systém zabezpečení, není tak silný jako Ethereum. Příčinou je, že ne všechny transakce na Polygonu jsou zabezpečeny sítí Ethereum, což představuje určité riziko v případě útoku. Další nevýhodou je, že i když razantně snižuje transakční poplatky, je stále závislý na poplatky na Ethereu, jinak řečeno, zvýší-li se poplatky na Ethereu, zvýší se také na Polygonu. Zároveň je jakožto druhá vrstva považován za méně decentralizované řešení [33]; [35].

Tabulka 4: Porovnání ETH mainnet a Polygonu
(zdroj: vlastní zpracování dle [33]; [34]; [35]; [36])

Oblast (kritérium)	Ethereum 2 mainnet (ETH)	Polygon (MATIC)
Průměrná cena transakčního poplatku	$1.68	$0.0026
Konsenzuální mechanismus	Proof of Stake	Proof of Stake
Počet transakcí / s	11,2	34,9
Kompatabilita	Ethereum samotné	Plně kompatabilní s Ethereum
Bezpečnost	Vysoká – všechny tranaskce jsou zabezpečeny	Nižší – ne všechny transakce jsou zabezpečeny Ethereum
Tržní kapitalizace	Velmi vysoká: 216 miliard dolarů (USD)	Nízká v porovnání s Ethereum: 7,9 miliard dolarů (USD)
Decentralizace	Plně decentralizovaná (500 000+ validátorů)	Méně decetralizovaný (100 validátorů)
Škálovatelnost	Omezená, ovlivněna kapacitou sítě	Vyšší, díky odlehčení mainnetu

2.7.3 Testovací sítě (testnet)

Testovací sítě umožňují vývojářům a uživatelům testovat aplikace a další funkcionality bez rizika finanční újmy. Používá se totiž testovací měny, která nemá žádnou reálnou hodnotu a lze ji získat z testovacích „**faucet**“ (vodovodů). Na testovací síti je tak možno poměrně věrně simulovat chování a interakci před nasazením aplikace a provozem v hlavní síti.

Pro Ethereum blockchain existuje mnoho testovacích sítí. Mezi ty nejznámější pro hlavní vrstvy patří Goerli, Sepolia a Rinkeby. Každá ze sítí má své vlastní charakteristiky a mechanismy pro ověřování transakcí [37].

2.7.3.1 Mumbai polygon testnet

Jedná se o **testovací síť druhé vrstvy – Polygonu**, která je s hlavní sítí plně kompatibilní a umožňuje tak vývojářům simulovat reálné podmínky v bezpečném testovacím prostředí. Síť je kompatibilní také s EVM, což usnadňuje přenos aplikací mezi sítěmi Ethereum a Polygon. Dává možnost získat zdarma testovací MATIC tokeny z „**faucetu**“ [15] pro testování transakcí a aplikací. Tyto tokeny lze bezplatně získat, ale mají i reálnou hodnotu (aktuálně $0.009, údaj k 20. květnu 2023), a tak je čerpání tokenů z faucetu na danou peněženku omezené. Vysoká rychlost bloků a kapacita transakce umožňuje rychlý chod aplikací [38].

2.8 Populární Web 3.0 nástroje a služby

K realizaci decentralizované vize přispívají různé nástroje a služby, které umožňují práci a s decentralizovanými aplikacemi a jejich provoz. Mezi nejvíce používané patří InterPlanetary File Systém, Remix IDE, Metamask a Truffle Suite. Každý z těchto nástrojů přináší unikátní funkcionality a vlastnosti, které otevírají nové možnosti pro obě strany – vývojáře i uživatele.

2.8.1 InterPlanetary File System (IPFS)

IPFS je inovativní systém pro **ukládání a sdílení dat v decentralizované síti**. Jde o globální systém typu **peer-to-peer**. Data v něm jsou identifikovány pomocí jedinečných hashů. Díky tomu jsou data dostupná kdekoli a kdykoli.

Tento systém přináší hned několik klíčových výhody. Zlepšuje rychlost a efektivitu přenosu dat tím, že umožňuje získávání dat z nejbližších uzlů místo stahování ze vzdálených serverů. Z principu blockchainu také kopíruje data na do dalších uzlů, a ty tak zůstávají dostupná i při výpadku některých uzlů. A také IPFS posiluje soukromí a bezpečnosti uživatelů tím, že umožňuje anonymní sdílení dat a zabraňuje cenzuře.

V kontextu Web 3.0, slouží IPFS jako nástroj pro vytváření decentralizovaných aplikací a platforem, které jsou zároveň odolné vůči cenzuře a chrání soukromí uživatelů. Stejně jako řešení 2. vrstvy v blockchainových sítích, IPFS odlehčuje hlavní síti tím, že distribuuje data na různé uzly, což zvyšuje celkovou škálovatelnost a efektivitu sítě.

[15] **Faucet** – umožňuje uživatelům si zažádat o zaslání malého množství MATIC testovacích tokenů na konkrétní adresu peněženky

2.8.2 Remix IDE

Remix je open-source nástroj pro **vývoj a testování smart contractů pro Ethereum blockchain**. Má jak webovou, tak desktopovou verzi. Obě umožňují vývojářům psát, testovat, nasazovat a ladit smart contracty napsané v jazyce Solidity (nativní jazyk pro Ethereum).

Obsahuje také například sadu nástrojů pro statickou analýzu kódu. Vývojáři mohou také nasadit smart contracty na vývojovou síť Ganache nebo integrovat řadu nástrojů pro vývoj blokchainu, jako je třeba Metamask.

2.8.3 Truffle Suite

Jedná se o sadu vývojových nástrojů pro Ethereum, která zahrnuje nástroje Truffle, Ganache a Drizlle. Truffle slouží pro vývoj, testování a nasazování („deploy“) smart contractů. Ganache je osobním blockchainem pro Ethereum, který umožňuje vývojářům vytvářet a deployovat smart contracty, provádět transakce a kontrolovat stav blockchainu bez nutnosti jeho nasazení do veřejného blokchainu. Drizzle doplňuje sadu, jako knihovna pro vytváření UI pro decentralizované aplikace [39].

2.8.4 Metamask

Metamask **je nejpoužívanější kryptoměnová peněženka pro EVM** [16] blockchainy. Instaluje se jako rozšíření webového prohlížeče, což umožňuje uživatelům snadný přístup k decentralizovaným aplikacím na Ethereu blockchainu přímo z prohlížeče [40].

Funguje jako most mezi tradičními mezi Ethereum blockchainem a prohlížečem a uživatelé tedy mohou jednoduše spravovat svou identitu, ukládat a spravovat své etherové a ERC-20 tokeny a interagovat s kontrakty v decentralizovaných aplikacích [40].

V Metamask peněžence jsou privátní klíče uživatelů uloženy bezpečně na jejich zařízeních a nikoli na vzdálených serverech, jako v případě většiny web2.0 přístupů. Dále dovoluje nastavení transakční poplatky (v daných hranicích) a vypisuje podrobné informace o každé transakci ještě před jejím odesláním [40].

[16] **Ethereum Virtual Machine (EVM) =** prostředí pro Ethereum smart contracty

2.8.5 Polygonscan

Polygonscan je nástroj (webová stránka), který slouží jako **průzkumník transakcí a smart contractů v síti Polygon.** Poskytuje tedy detailní informace o veškeré aktivitě na této síti. Nabízí několik funkcí, mezi které patří sledování transakcí v reálném čase, prohlížení podrobností o konkrétních bloků, zobrazení informací o smart contractech, včetně jejich kódu a transakčních historií. Umožňuje také sledování pohybu ERC-20 a ERC-721 tokenů v rámci sítě. V neposlední řadě, uživatelé mohou využít Polygonscan k ověření smart contractů, což dopomáhá transparentnosti a důvěry v blockchainové aplikace. Jeho obdobou pro Ethereum síť je etherscan a *mumbai.polygonscan.com* pro Mumbai testovací síť.

3 VERZOVÁNÍ PROJEKTU

Klíčovou součástí vývoje software je jeho verzování. Umožňuje sledovat změny v kódu, spolupracovat efektivněji ve vývojovém týmu a vrátit se k předchozím verzím kódu, pokud je to potřeba. Dva nejpoužívanější verzovací nástroje jsou GitHub a GitLab. Obě platformy poskytují takřka stejný seznam funkcí pro správu verzí – viz **Tabulka 5**, takže popis k jednotlivým platformám níže, nespecifikuje všechny funkcionality každé z nich.

3.1 Github

Github poskytuje rozhraní pro git, což je nejpoužívanější systém pro správu verzí. Github je oblíbený pro své uživatelsky přívětivé rozhraní, obrovskou komunitu a rozsáhlé integrace s řadou externích nástrojů [41].

Umožňuje vývojářům hostovat a sdílet vyvíjený kód, včetně jeho historie. Dále podporuje pull requesty, což je metoda, při které se zkoumají provedené změny dané verze před jejím začleněním do hlavní větve kódu. Mimo to nabízí Github další funkce jako je správa úkolů či automatizaci prostřednictvím Github Actions [41].

3.2 Gitlab

GitLab je kompletní DevOps [17] platforma, která je dodávána jako jedna celistvá aplikace. Zahrnuje řadu nástrojů a funkcí pro správu verzí, CI/CD [18], monitorování a další aspekty vývoje SW [42].

Podporuje jak veřejné, tak soukromé repozitáře, umožňuje týmovou spolupráci a obsahuje nástroje pro plánování, sledování a řízení projektů. Gitlab také nabízí možnosti automatizace pro vývoj a testování [42].

[17] **Development (IT) Operations (DevOps) =** přístup k vývoji SW, který zdůrazňuje komunikaci a spolupráci mezi vývojářem a odborníky na IT z provozu
[18] **Coninuous Integration / Continuous Deployment (CI/CD) =** proces průběžné integrace a dodání/deploy kódu pro zvýšení efektivity vývoje

Tabulka 5: Porovnání funkcí verzovacích platforem Github a Gitlab

(zdroj: vlastní zpracování dle [41]; [42])

Vlastnost	Github	Gitlab
Správa verzí	Ano	Ano
Soukromé repozitáře	Ano	Ano
Veřejné repozitáře	Ano	Ano
Spolupráce v týmu	Ano	Ano
Sledování „issues“	Ano	Ano
CI/CD integrované nástroje	Ano (pomocí Github Actions)	Ano
Vlastní CI/CD runners	Ne	Ano
Hostování zdarma	Ano	Ano
Placené plány	Ano	Ano

4 OVĚŘOVÁNÍ DOKUMENTŮ

Ověřování dokumentů je zásadní proces, kdy se zajišťuje, že dokumenty, které jsou sdíleny, zachovávají svou autenticitu a nedošlo u nich k poškození nebo manipulaci. Tento proces často zahrnuje jak potvrzení identity odesílatel, tak ověření integrity dat.

4.1 Ověřování dokumentů ve Web2.0

Ve světě web 2.0 ověřování dokumentů obvykle závisí na centralizovaných serverech nebo databázích. Tato centralizovaná entita uchovává záznamy a je zodpovědná za jejich bezpečné uložení a sdílení. Zajištění autenticity a integrity lze zde dosáhnout mnoha způsoby. Častým způsobem je šifrování dat k zajištění bezpečnosti dokumentů. **Šifrování** zabezpečuje obsah dokumentu tak, že ho transformuje do nečitelné podoby, kterou lze obnovit pouze s použitím správného dešifrovacího klíče. To zajišťuje, že data v dokumentu zůstanou utajena a chráněna před neoprávněným přístupem.

Moderním přístupem je třeba využití **časových razítek –** „timestamps“, které se zaměřuje na sledování vytvoření dokumentu a přesný čas jeho jednotlivých změn. Tím pravděpodobně nejvíce spolehlivým způsobem je ale **digitální podpis**.

Přesto, že centralizované ověřování dokumentů může být účinné, existují některé nevýhody tohoto přístupu. Jednou z hlavních výtek je závislost na centralizovaném subjektu, který má kontrolu nad všemi záznamy. To znamená, že pokud by takový subjekt selhal nebo byl napaden, mohlo by to vést ke ztrátě dat nebo ohrožení bezpečnosti.

Dalším problémem je nedostatek transparentnosti a důvěryhodnosti. Uživatelé musí spoléhat na centralizovanou entitu, že správně uchová jejich data. Tím vzniká možnost manipulace s daty a nedůvěryhodných praktik.

4.1.1 Digitální podpis

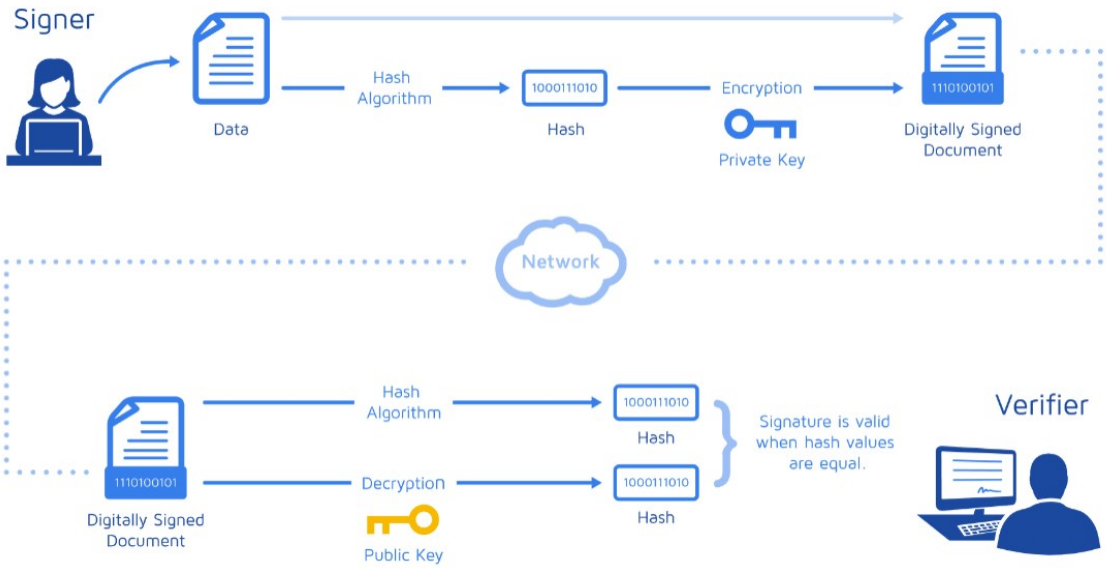
Digitální podpis, též **DSA** (Digital Signature Algorithm) je matematický algoritmus, který umožňuje vytvoření unikátního a nezaměnitelného záznamu, který je připojený k určitému dokumentu. Tento podpis slouží k ověření identity odesílatele a zaručuje, že dokument pochází od oprávněného zdroje. Při ověřování digitálního podpisu je využíván **asymetrický kryptografický systém**, který se skládá ze dvou klíčů, **veřejného a soukromého**.

Soukromý klíč slouží k podpisu dokumentu, zatímco veřejný klíč je používán k ověření platnosti šifrovaného podpisu.

Princip digitálního podpisu je tedy následující:

1. odesílatel spočítá otisk zprávy;
2. tento otisk zašifruje svým soukromým klíčem a spolu s vlastní zprávou jej pošle adresátovi;
3. příjemce vypočítá veřejným klíčem odesílatele také otisk přijaté zprávy a tento výsledek; porovná s otiskem, který také obdržel od odesílatele;
4. pokud jsou oba otisky totožné, je zachována integrita dokumentu.

Digitální podpis má **široké využití** v různých oblastech, jako je elektronický podpis dokumentů, zabezpečení elektronických transakcí, ochrana před paděláním a další. Poskytuje spolehlivý a **efektivní způsob ověřování** a zajišťuje důvěryhodnost digitálních dokumentů.



Obrázek 10: Digitální podpis

(zdroj: [62])

4.2 Ověřování dokumentů v (Ethereum) blockchainu

Při ověřování dokumentů, čili jejich pravosti, s využitím blockchainu**, se spoléhá zpravidla na jeho základní princip, kterým je neschopnost změnit data po jejím zapsaní** (případná úprava se provádí zapsáním změněné informace v dalším bloku). Proces obvykle zahrnuje vytvoření digitálního otisku – „hash“, protože je datově úsporný (větší datová velikost znamená zpravidla i vyšší cenu za uložení této hodnoty na blockchain) a následné zapsání této hodnoty do blockchain sítě. Obvyklý proces se tedy často skládá z těchto kroků:

1. **Vytvoření hash dokumentu**
2. **Zápis hashe do blockchain sítě**
3. **Ověření dokumentu** – vytvoření hashe z ověřovaného dokumentu a porovnání obou hashů

Běžnými metodami ověření dokumentů s využitím blockchainu jsou například **digitální podpisy**, ověření **časových razítek** nebo ověření vlastnictví pomocí digitálních certifikátů nebo **NFT**.

Ve stručnosti lze tedy říci, že blockchain využívá ověřených metod ze světa Web2.0 a dává mi věrohodnou bezpečnostní vrstvu navíc, v podobě uložení nezměnitelného záznamu ve své technologii.

5 HOSTING

Hosting poskytuje „hostování“ webové stránky na serveru. Tento server by měl být dobře zabezpečen vůči různému druhu útoků jako jsou například DDoS útoky [19]. Zabezpečení by mělo zahrnovat fyzickou ochranu serverů, dále přístupové kontroly, monitorování útoků a samozřejmě šifrování dat pro případ prolomení.

Moderní hostingový poskytovatelé disponují například velkým množstvím serverů rozdistribuovanými geograficky po celém světě, mimo jiné pro zachování co nejlepší latence bez ohledu na lokalitu uživatele.

5.1 Doména

Jedná se o jedinečný název, pomocí něhož se identifikuje webová stránka. Je součástí systému doménových jmen, který usnadňuje navigaci v oblasti internetu. Každá doména musí být registrována u autorizovaného poskytovatele domén. Jsou úzce spojeny s hostingovými službami pro správu webového obsahu.

5.2 Amazon web services (AWS)

AWS je největší dodavatel cloudový služeb na světě, v posledním čtvrtletí roku 2022 to bylo **32 % trhu**, 2. byl Microsoft Azure s 23 % a poté Google Cloud s 10 % pokrytím zákaznického trhu. AWS disponuje **širokou nabídkou pro ukládání, správu a zpracování dat v cloudu**. Vyniká především vysokou **škálovatelností** a **bezpečností** služeb. Nabízí využití hned několika operačních systémů a open-source řešení [43]; [44].

Oproti konkurenci vyniká dostupnými službami na vysoké úrovni a individuálního nastavení, kdy uživatel platí jen za reálné využití výpočetních kapacit či skutečný provoz na své hostované webové stránce – „on-demand“ [43].

Mezi nejznámější produkty patří AWS Amplify pro správu postupu při vývoji, distribuci a hostování SPA, Simple Storage Service (Amazon S3) pro uživatelsky přívětivé uložiště produkční verze aplikace o libovolné velikosti nebo třeba Elastic Cloud Computing (Amazon EC2), která poskytuje škálovatelné výpočetní cloudové zdroje [43].

[19] DDoS (Distributed Denial of Service) = útok s cílem přehlcení kapacit serveru obrovským množstvím požadavků

5.2.1 Amazon Simple Storage Service (S3)

S3 je služba pro **ukládání souborů**, která nabízí škálovatelnost, vysokou dostupnost a výkon srovnatelný s nejlepšími službami tohoto typu. Díky tomu může klient s jakýmikoliv požadavky a nároky ukládat a zabezpečit libovolné množství dat pro prakticky jakýkoliv účel, ku příkladu pro ukládání obrovských datových souborů, cloudových či mobilních aplikací. Služba je koncipovaná s možností na míru zákazníkovi, a ten tak platí je za své individuální požadavky a skutečné využití služby [45].

Sám Amazon zjednodušeně popisuje S3 jako objektové uložiště pro přístup k libovolnám datům odkudkoliv.

Obrázek 11: Amazon S3 – grafické znázornění funkcionality

(zdroj: [45])

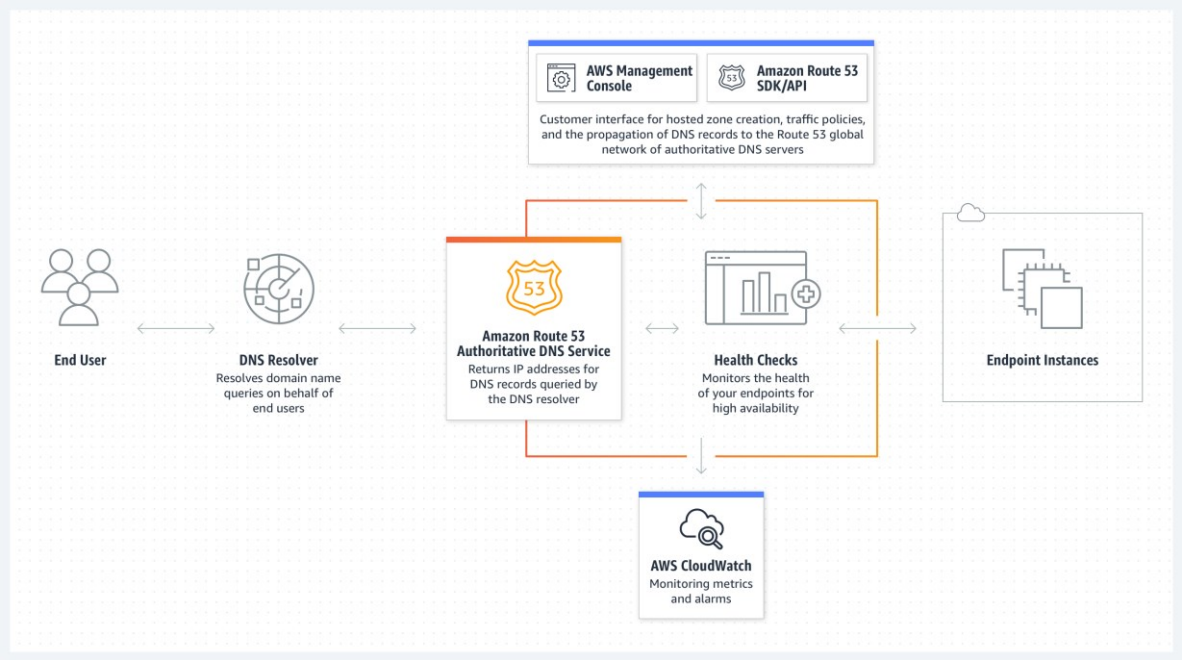
5.2.2 Amazon Route 53

Route 53 je dle slov Amazonu spolehlivý, cenově efektivní způsob routování k cílovým uživatelům internetových aplikací. Poskytuje globálně škálovatelnou a spolehlivou službu pro **správu domén a směrování na internetu**. Služba je pojmenována po protokolu DNS, jež je zodpovědný za překlad jmen jednotlivých domén na IP adresy [46].

S pomocí AWS Route 53 mohou uživatelé registrovat nové domény, přenést stávající od jiných poskytovatelů a spravovat jednotlivé DNS záznamy. Mezi nejpoužívanější typy záznamů patří NS (Name Server) pro jména serverů, A záznamy pro mapování jmen na IP adresu (především pro směrování na konkrétní webovou stránku), dále CNAME (Canonical Name) záznamy k vytvoření aliasu pro existující doménové jméno (přesměrování doménového jména na jiné, například diplomka.cz na www.diplomka.cz),

MX (Mail Exchanger) záznamy pro doručování e-mailových zpráv a TXT (Text) záznamy, přidání textových dat do DNS, které slouží především pro ověřování domény pro e-mailové služby [46].

Díky globální distribuci serverů AWS navíc umožňuje spolehlivou dostupnost s nízkou latencí pro uživatele kdekoliv na světě. Služba je také snadno spolupracuje s dalšími službami AWS, což nabízí možnost opravdu komplexní konfigurace [46].



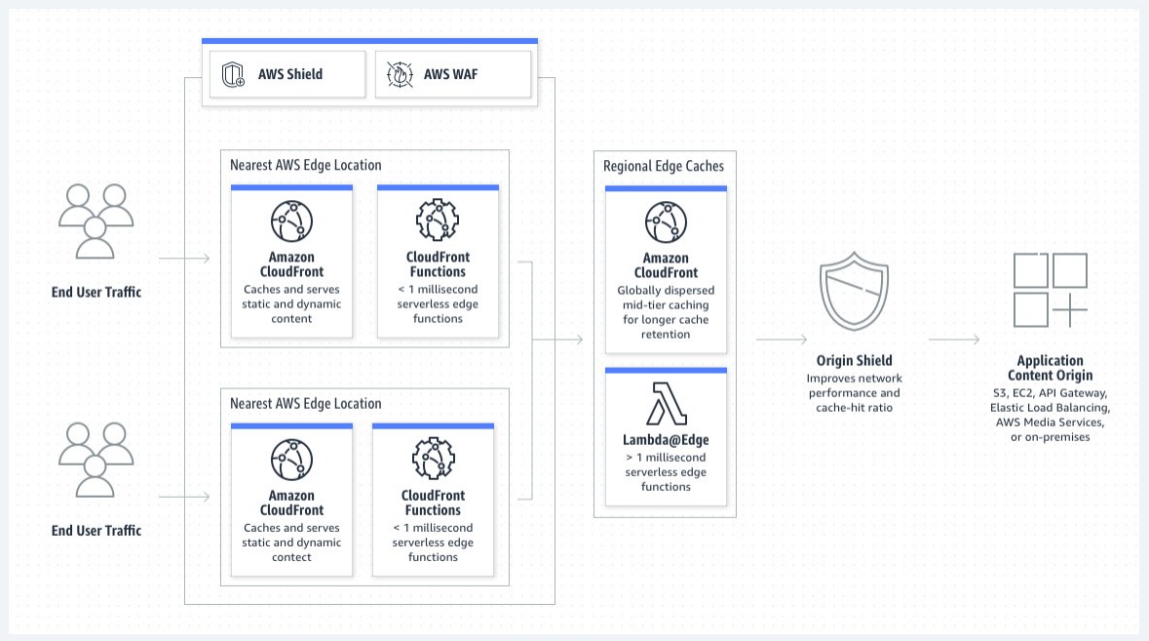
Obrázek 12: Amazon Route 53 - grafické znázornění funkcionality

(zdroj: [46])

5.2.3 Amazon Cloudfront

Amazon Cloudfront je vysoce výkonná služba, která umožňuje rychlé a bezpečné doručení různorodého obsahu na celém světě. Funguje jako **Content Delivery Network** (CDN), což znamená, že umožňuje efektivní distribuci obsahu z webových aplikací, streamovaného videa a dalších souborů tohoto typu. Využívá svou síť globálních serverů – „edge locations" četné geografické rozdistribuování pro zajištění nízké latence, a tedy rychlého načítání obsahu [44].

Nabízí celou řadu funkcí, včetně cachování obsahu, komprese **SSL/TLS šifrování** [20], dynamického obsahu atd. Stejně jako zbytek služeb z AWS rodiny, je i Amazon Cloudfront plně kompatabilní s ostatními službami tohoto dodavatele [44].



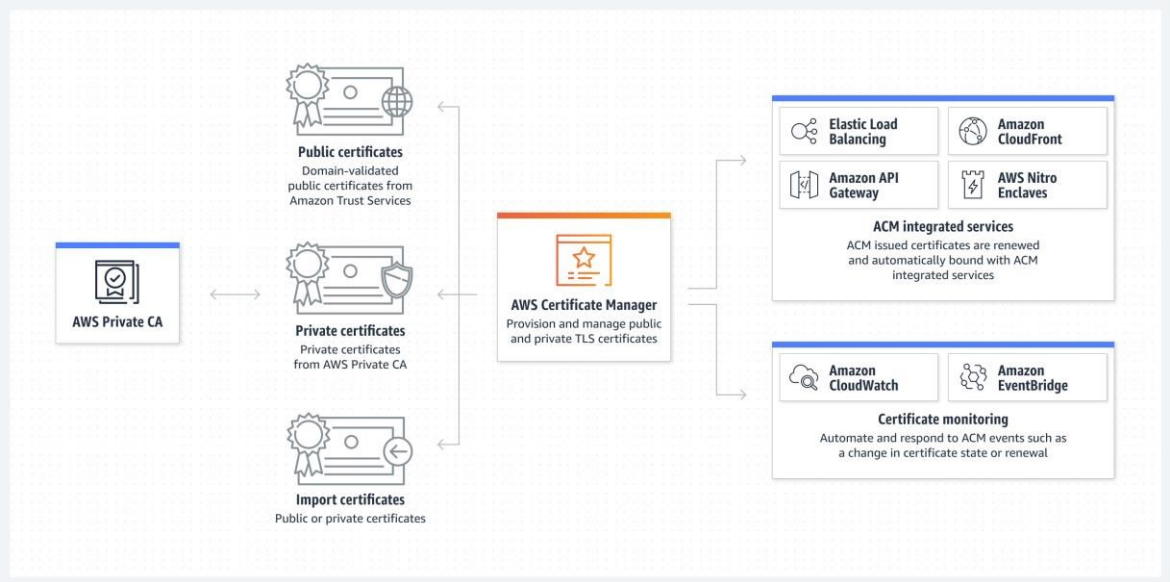
Obrázek 13: Amazon Cloudront – grafické znázornění funkcionality

(zdroj: [44])

[20] **SSL/TLS (Secure Socket Layer/Transport Layer Security) šifrování = p**rotokol pro zabezpečený přenos dat mezi webovým serverem a prohlížečem

5.2.3.1 Amazon Certificate Manager (ACM)

Certifikační manažer od Amazonu umožňuje, jak napovídá samotný název, jednoduché získání, **správu a nasazení SSL/TLS certifikátů**, tedy certifikátů jejichž hlavní funkce je zajištění šifrovaného, jinými slovy poměrně bezpečného přenosu dat mezi webovou aplikací a serverem, na němž je její obsah uložen, a to protokolu HTTPS (protokol HTTP doplněný SSL certifikátem). Tyto certifikáty nabízí Amazon zcela zdarma. Dále se využívají například v Amazon Cloudfront, kdy ověření vlastníka probíhá pomocí DNS záznamů [47].



Obrázek 14: Amazon Certificate Manager – grafické znázornění funkcionality (zdroj: [47])

5.2.4 Amazon Elastic Cloud Computing (EC2)

AWS EC2 je výkonná cloudová služba, která umožňuje uživatelům si pronajímat **virtuální servery v cloudu**. Zprostředkovává tvorbu a správu virtuálních serverů, pomocí "instancí" přímo v cloudové službě. Tyto instance nabízejí mnohostrannou schopnost škálovat podle aktuálních potřeb, což přináší velkou flexibilitu a spolehlivost.

V rámci EC2 je k dispozici široký výběr instancí, s různou kombinací vlastností, jako je výkon procesoru, typ operačního systému, velikost paměti, velikost uložiště a síťová

nastavení. Tato široká paleta umožňuje přizpůsobit konfiguraci doslova na míru potřebám hostované aplikace. Zdarma, v rámci **bezplatného programu** lze například získat až:

- **30 GB EBS uložiště** (typ General Purpose nebo Magnetic);
- **750 hodin** měsíčně instancí typu **t2.micro nebo t3.micro** (instance obvykle obsahuje 1 vCPU a 1 GB RAM);
- **2 milony I/O operací** (v rámci komunikace s EBS uložištěm);
- **1 GB přenesených dat/měsíc** (přenos dat z EC2 instancí do internetu);
- **750 hodin/měsíc Elastic Load Balancer** (automaticky distribuuje příchozí provoz napříč více EC2 instancemi) [48].

SLužba je navíc dostupná po celém světě dle regionů, přičemž každý region má několik dostupných zón. Toto rozložení zajišťuje minimalizace latence a robustnost vůči výpadkům. Pokud jde o síťové možnosti, EC2 umožňuje vytvářet privátní virtuální sítě pomocí služby Virtual Private Cloud (**VPC**). Pro účely ukládání dat nabízí EC2 různé varianty, včetně perzistentního blokového úložiště prostřednictvím Amazon Elastic Block Store (EBS) a instancí se zabudovaným úložištěm.

Co se týče škálování, EC2 nabízí jak **horizontální**, tak **vertikální** možnosti **škálování**. Horizontální škálování znamená přidávání nebo odebírání instancí podle aktuální zátěže, zatímco vertikální škálování znamená úpravu kapacity existujících instancí [49].

II. PRAKTICKÁ ČÁST

6 VÝBĚR TECHNOLOGIÍ PRO VÝVOJ APLIKACE

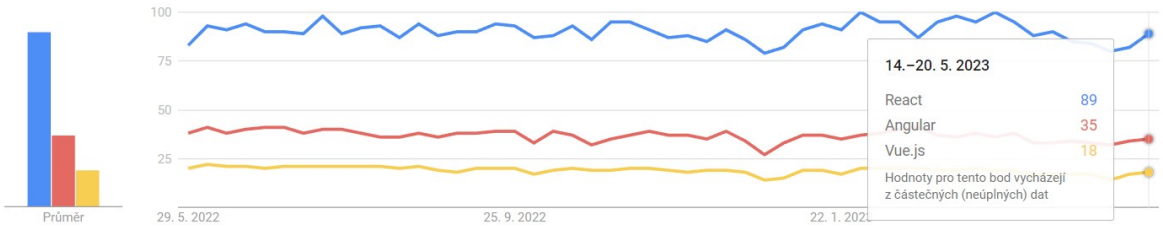
Při výběru technologií pro vývoj této aplikace, bylo přihlíženo k vícero faktorům, včetně těch hlavních, v nichž bylo přihlíženo k **moderním trendům, oblíbenosti** mezi vývojáři a **zabezpečení**.

Hlavní priorita byla **stabilita** a **dlouhodobá udržitelnost technologií**. Výběr tedy závisel na osvědčených s dlouho historií a poměrně jistou budoucností. Hlavní bezpečnostní složku zde plní backendová a blockchainová část, na které byl brán ještě zvýšený důraz.

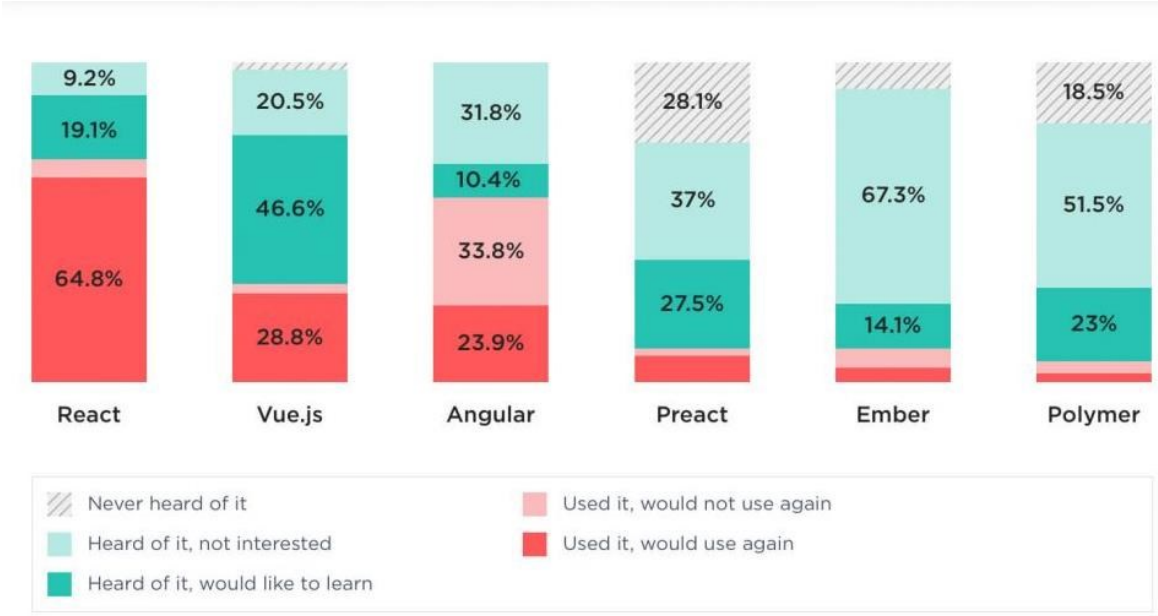
6.1 Výběr frontendových technologií

Pro práci s uživatelské rozhraním byla vybrána JS knihovna React, doplněná o typování v podobě Typescriptové nadstavby. Jako správce nodejs balíčků byl vybrán **yarn**, jelikož představuje jakousi vylepšenou verzi npm.

Výběr Reactu je založen také na jeho obrovské podpoře komunity, a především vývojářů (Meta), snadné škálovatelnosti a oblíbenosti, a to hlavně z pohledu vývojářů. Aktuální data z Google trends ukazují **průměrné množství výsledků o Reactu 2,5x vyšší než u přímé konkurence v podobě Angularu a 5x vyšší než u Vue**.



Obrázek 15: Porovnání dat vyhledávání Reactu, Angularu a Vue dle Google trends
(zdroj: vlastní zpracování dle [63])



Obrázek 16: Grafické porovnání frontendových frameworků dle oblíbenosti
(zdroj: [64])

6.2 Výběr blockchainových technologií

Z pohledu koncového uživatele je myslet hned na několik faktorů. Ty hlavní u webové stránky, a týká se to i práce s blockchainovými technologiemi (dApps), jsou bezesporu vysoké nároky na bezpečnost, rychlost, a co nejnižší náklady.

Co se týče bezpečnosti a celkové spolehlivosti, jsou blockchainové programy, u Ethereum decentralizované aplikace nazývané smart contracty, do většinové míry přímo závislé na kvalitě zabezpečení konkrétní smart contractové implementace. Tedy zejména sadě ošetření, množství rizikových funkcí a verzi jazyka, ve kterém je napsán (zde **Solidity 0.8.18**). Rizikové funkce zde nefigurují vůbec, alespoň ne na první pohled jasné, neboť jen jediná funkce upravuje již uloženou strukturu dokumentu a to „pouhým“ přidání řetězce do pole, nikoli například úpravou konkrétní již existující položky tohoto pole.

Rychlost je do značné míry závislé na aktuálním provozu sítě, jinými slovy, kolik transakcí aktuálně daná blockchainová síť zpracovává. Nicméně Ehtereum obecně se po přechodu na validační konsenzus Proof of Stake pyšní solidní rychlostí zpracováním transakcí i na své hlavní síti – „mainnetu“. Při užití sítě druhé vrstvy je tato rychlost transakčního zpracování ještě značně rychlejší, při nejnáročnějších transakcí, uložení nové struktury či přidání

zmíněné položky do pole existující struktury, trvá zpracování od podepsání transakce přibližně 5 vteřin.

Náklady na práci s blockchainovou technologií jsou v tomto případě skutečně nulové, protože je využito **Mumbai Polygon testnet**, což je testovací síť druhé Ethereum vrstvy, Polygonu.

6.3 Výběr backendových technologií

Při volbě backendových technologií bylo zohledněno několik faktorů, které jsou kritické pro efektivní fungování této webové aplikace. Mezi hlavní aspekty patřily nároky na flexibilitu, rychlost a škálovatelnost. V tomto kontextu se **MongoDB** jeví jako ideální řešení.

MongoDB je schopná ukládat data v BSON (binární reprezentaci JSON), což představuje výhodu oproti klasickému JSON, neboť umožňuje zpracování více datových typů. Rychlost a škálovatelnost jsou dalšími důležitými faktory. MongoDB se vyznačuje rychlou odezvou a dobrým škálováním, které umožňuje snadné přidání nových serverů do databázových clusterů pro zvládání zvýšeného objemu dat a provozu.

6.4 Výběr verzovací technologie

Verzovací služba Github byla využita pro verzování vývoje aplikace. Tento výběr byl motivován především předešlými zkušenostmi z touto platformou. Jedná se zároveň o nejpopulárnější platformu git repozitářů s obrovskou komunitou a podporou

6.5 Výběr hostingových technologií

Pro hostování obou čisté aplikace – frontendové i serverové, byla vybrána kombinace služeb od AWS**: S3 bucket, Route 53, Cloudfront s využitím Certificate Manager a Amazon Elastic Compute Cloud**. Tato kombinace poskytuje robustní řešení, které nad míru splňuje potřeby obou částí aplikace.

Prvním klíčovým faktorem výběru tohoto řešení je široká nabídka služeb a vysoká škálovatelnost, kterou AWS nabízí. AWS poskytuje široký výběr služeb pro správu a zpracování dat v cloudu. To zahrnuje vše od výpočetních a úložných služeb až po databázové

služby. Tato široká nabídka umožňuje přizpůsobit konfiguraci podle konkrétních potřeb aplikace, zatímco vysoká škálovatelnost těchto služeb zajišťuje, že systém může rychle reagovat na změny v požadavcích na zatížení aplikace.

Dalším důležitým faktorem je vysoká dostupnost a výkon, které tyto služby poskytují. S použitím služeb AWS S3 pro úložiště, Route 53 pro správu DNS, Cloudfront jako CDN a EC2 pro hostování serverové části na vypůjčeném cloudovém serveru, je aplikace schopna poskytnout vysokou dostupnost a výkon pro své uživatele po celém světě. Tyto služby také přispívají k vysokému výkonu aplikace, což vede ke zvýšené uživatelské spokojenosti.

Bezpečnost je další klíčový faktor. AWS je známý svými vysokými standardy v oblasti bezpečnosti, včetně šifrování dat v transitu a v klidu, pokročilých možností řízení přístupu a dalších funkcí. Tento závazek k bezpečnosti je evidentní v Certificate Manager, který umožňuje snadné získání, správu a nasazení SSL/TLS certifikátů pro zabezpečení komunikace mezi aplikací a serverem.

AWS také nabízí snadnou integraci mezi svými službami, což usnadňuje správu a konfiguraci celého systému. Například, S3 může být použit pro ukládání obsahu, zatímco Cloudfront je použit pro jeho distribuci, Route 53 pro správu DNS, ACM pro poskytování SSL/TLS certifikátů a EC2 pro hostování serverové části. Tato integrace umožňuje jednoduchou správu celého životního cyklu aplikace.

Poslední, ale taktéž důležitým faktorem je platba jen za skutečné využití služeb. AWS totiž umožňuje uživatelům platit pouze za skutečné využití služeb, což pomáhá udržet náklady na minimum.

7 PŘÍPRAVA TECHNOLOGIÍ

Inicializace aplikace je velmi jednoduchá. Stačí mít naintalovaný npm, který je instalován zároveň s Node.js. Samotná inicializace se provede v terminálu příkazem *npx create-react-app <názevAplikace> --template typescrip*t. Poté stačí otevřit kořenový adresář aplikace ve vhodném vývojovém prostředí, Visual Studio Code je rozhodně správnou volbou a začít kódovat. Pro efektivnější a spolehlivější přes npm získat správce balíčků yarn. V tomto bodě je již aplikace připravena pro vývoj a testování na lokálním serveru.

7.1 Využité npm balíčky

NPM disponuje obrovským množstvím knihoven (v dubnu roku 2020 tomu bylo přes **1,3 milionu balíčků**). Obecným pravidlem je využít co nejefektivněji instalované knihovny. Tedy instalovat jich, pokud možno co nejméně a pokud možno se vyhnout instalaci celé obrovské knihovny, kvůli jediné implementaci jejího malého prvku [50].

Tabulka 6: Instalované NPM balíčky a jejich využití

(zdroj: vlastní zpracování)

NPM knihovna	FE / server / blockchain	Funkce
react-router	FE	Směrování (routing)
react-router-dom	FE	Směrování (routing)
styled-components	FE	CSS stylování
zustand	FE	Globální stavy aplikace
crypto-js	FE	Šifrování dat
react-select	FE	Knihovna pro výběr (včetně využitého multiselect)
sweetalert2	FE	Modal knihovna (vhodná pro zobrazení stavu aplikace; úspěšnost operací, varování atd.)
ethers	Blockchain	Komunikace s blockchainem

express	Server	Webový FW pro Node.js; umožňuje snazší vytváření serverů
mongodb	Server	Nativní ovladač Node.js pro MongoDB databázi
mongoose	Server	Snazší manipulace s MongoDB skrze modely a schémata
multer	Server	Middleware pro Express.js pro manipulaci s multipart/form-data; vhodné pro nahrávání formátovaných souborů
bcrypt	Server	Zabezpečení hesel skrze hash funkce a "salt"
cors	Server	Nastavení CORS policies (Cross-Origin Resource Sharing) na serveru
jsonwebtoken	Server	Vytváří a ověřuje JSON webové tokeny pro autentizaci přístupu
forever	Server	Spouští server jako službu (ten pak běží napořád; v případě chyby/spadnutí jej automaticky „nahodí" zpět)
nginx	Server	Slouží jako reverzní proxy server

7.2 Příprava blockchainových technologií

Jak bylo již zmíněno, ke komunikaci s blockchainovými smart contracty jsou potřeba tokeny na podepisování transakcí na komunikační peněžence dané sítě, kde je contract nasazen.

7.2.1 Založení Metamask peněženky

Pro založení Metamask peněženky je třeba nainstalovat **rozšíření** Metamask do kompatibilního **webového prohlížeče**, jako je například Google Chrome nebo Mozilla Firefox. Po instalaci rozšíření je třeba provést počáteční konfiguraci peněženky. To zahrnuje vytvoření nové peněženky nebo obnovení existující peněženky pomocí soukromého klíče nebo seed fráze. Tento Seed je důležitý, protože slouží k obnovení peněženky v případě ztráty přístupu.

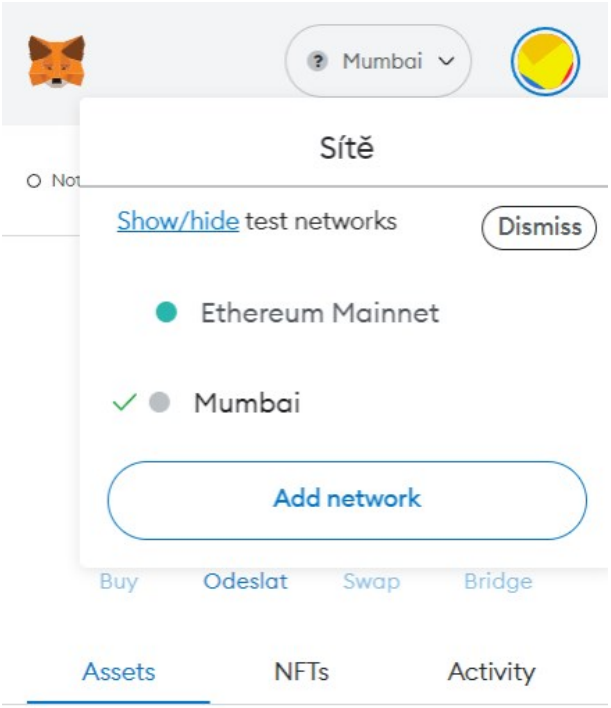
7.2.2 Přidání Mumbai sítě do pěněženky

Přidání Mumbai sítě do Metamask peněženky je velmi jednoduché, a dá se provést dvěma základními způsoby. První, složitější z nich je manuálně, zahrnuje několik dílčích kroků, jako je povolení zobrazení testovacích sítí a následné přidání parametrů:

- Network name: Mumbai Testnet;
- New RPC URL[21]: *https://rpc-mumbai.maticvigil.com*;
- Chain ID: 80001;
- Currency symbol: MATIC;
- Block Explorer URL: *https://mumbai.polygonscan.com*.

Druhou, jednodušší variantou pro přidání Mumbai sítě, je jít na url https://mumbai.polygonscan.com/, kde v patičce stačí kliknout na tlačítko pro přidání sítě do Metamask peněženky, a následně potvrdit požadované kroky.

[21] **Uniform Resource Locator (URL) =** identifikuje unikátní adresu pro zdroj nebo službu na internetu



Obrázek 17: Metamask – přidaná síť Mumbai

(zdroj: vlastní zpracování)

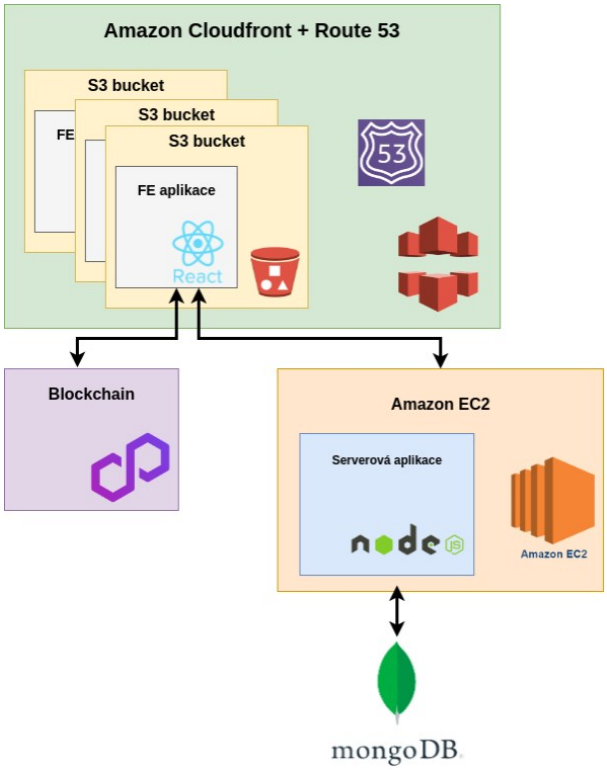
7.2.3 Získání testovacích tokenů

Zisk testovacích tokenů pro pokrytí nákladů na transakční poplatky je v případě Mumbai sítě opravdu snadný. Stačí mít zprovozněnou blockchainovou pěněženku, v našem případě Metamask. Pak stačí zkopírovat adresu peněženky, jít na adresu „**kohoutku**“: *https://faucet.polygon.technology/*, vybrat správný token a síť a vložit adresu své zkopírované peněženky. Pak už stačí jen vyčkat pár okamžiků a tokeny by měly dorazit. Po tomto kroku je Metamask připravena pro komunikaci se smart contractem.

Je vhodné poznamenat, že faucet ověřuje aktuální stav konta tokenů této peněženky, aby předešel zneužití tohoto bezplatného čerpání – testovací token má reálnou hodnotu, sice velmi malou, ale má. V případě, že máte více než dostatečné množství tokenů na velké množství transakcí, upozorní Vás na tuto zkušenost a tokeny Vám na tuto adresu peněženky nepošle.

8 KOMUNIKACE APLIKACE

Komunikaci naší aplikace lze rozdělit do dvou základních táborů: s backendem a s blockchainem. Mumbai Polygon Testnet (blockchain) zde slouží jako decentralizovaná verifikační autorita. Ukládá totiž data malé datové velikosti. Backendová část, uložená na cloudovém serveru EC2 komunikuje s databází MongoDB. Obě části komunikují vzájemně s FE React aplikací, nikdy však mezi sebou navzájem.



Obrázek 18: Přehled služeb aplikace a jejich spolupráce

(zdroj: vlastní zpracování)

8.1 Komunikace aplikace s blockchainem

Komunikace je realizována za pomoci npm knihovny **ethers**. Definici pro komunikaci mezi frontendovou částí a smart contractem lze realizovat různými způsoby. Jedním z nich a pravděpodobně tím nejvíce efektivním je definice celého ABI [22]. Toho lze dosáhnout specifické deklarace, která zahrnuje celé **ABI**, poskytovatele, se kterým se budou transakce

[22] **ABI (Application Binary Interface)** = soubor definic funkcí a událostí, které smart contract obsahuje

komunikovat a adresu, na které je kontrakt nasazen. Výsledná podoba s využitím npm knihovny *ethers* poté vypadá obdobně:

```
const contractDocuments = new Contract(
    '0xb2F4886eCC469c29808b0094aD42ff504E3F1730',
    [
        'function storeDocument(string memory title, string memory metadataHash, string memory branchID, string memory contentHash) public',
        'function addMetadata(string memory contentHash, string memory metadataHash, string memory branchID, string memory title) public',
        'function getDocument(string memory contentHash) view returns (address author, tuple(string metadataHash, string branchID, string title)[] memory metadata)',
        'event DocumentStored(string indexed contentHash, string title, address author, tuple(string metadataHash, string branchID, string title)[] metadata)',
        'event MetadataAdded(string indexed contentHash, string metadataHash, string branchID)',
    ],
    provider
);
```

Pro úplnost je potřeba získat poskytovatele a předat jej kontstruktoru kontraktové instance.

```
web3 > TS contract.tsx > ...
import { Contract } from 'ethers';
import { Web3Provider } from '@ethersproject/providers';
import { JsonRpcProvider } from '@ethersproject/providers';

const useDocumentContract = () => {
  let provider: any;
  if ((window as any).ethereum !== undefined) {
    provider = new Web3Provider((window as any).ethereum);
  } else {
    provider = new JsonRpcProvider(process.env.REACT_APP_RPC_URL);
  }

  const contractDocuments = new Contract(
    '0x22FD38b2DEdCBD75A101CB4DF9c1A2faa2A020BC',
    [
      'function storeDocument(string memory title, string memory metadataHash, string memory branchID, string memory contentHash) public',
      'function addMetadata(string memory contentHash, string memory metadataHash, string memory branchID, string memory title) public',
      'function getDocument(string memory contentHash) view returns (address author, tuple(string metadataHash, string branchID, string title)[] memory metadata)',
      'event DocumentStored(string indexed contentHash, string title, address author, tuple(string metadataHash, string branchID, string title)[] metadata)',
      'event MetadataAdded(string indexed contentHash, string metadataHash, string branchID)',
    ],
    provider
  );

  return contractDocuments;
};

export default useDocumentContract;
```

Obrázek 19: Inicializace smart contractu pro FE

(zdroj: vlastní zpracování)

8.1.1 Smart contract pro aplikaci

Pro psaní a nasazení bylo využito webového prostředí nástroje **Remix IDE** s Metamask peněženkou jako zprostředkovatele nasazení. Samotný smart contract je poměrně jednoduchý, protože jeho úkolem je věrohodně ukládat data o dokumentu do struktury. Z

tohoto důvodu byly vytvořeny dvě struktury, jedna hlavní pro dokument a druhá vnořená pro pole metadat těchto dokumentů.

```
struct Metadata {
        string metadataHash;
        string branchID;
        string title;
}
struct Document {
        address author;
        Metadata[] metadata;
        mapping(string => bool) metadataExists;
        string contentHash;
}
```

Dále byly vytvořeny tři funkce – viz 8.1, které slouží k vytváření nové struktury dokumentu, přidávání položky Metadat do existujícího dokumentu a získání dat o celé struktuře dokumentu.

K těmto funkcím byly závěrem přidány události - "**Eventy**" – viz 8.1, které slouží pro zaznamenávání a monitorování definovaných aktivit či akcí daného smart contractu. V konkrétním případu užití aplikace se čeká na tuto událost při ukládání nového dokumentu do blockchainové sítě, a až teprve poté se dokumentová data ukládají do databáze.

8.1.2 Ukládání dokumentu do blockchainu

Ukládání dokumentu do blockchainu je zápis do sítě, a proto stojí poplatek. Pro podepsání těchto transakcí tedy potřebujeme podepisovatele s dostatečným množstvím tokenů na Metamask peněžence.

Samotné uložení probíhá dvěma způsoby, protože dokumenty jsou mapovány dle *contentHash*. **Prvním je případ**, kdy dokument je svým obsahem – ***contentHash* unikátní**

a jeho položka tedy zatím neexistuje, vytvoří se tedy nová struktura dokumentu s jednou položkou metadat.

```
try {
  const response = await contractWithSigner.storeDocument(
    bcData.metadata[0].title,
    bcData.metadata[0].metadataHash,
    `${bcData.metadata[0].branchID}`,
    bcData.contentHash
  );
  setStoringDoc(true);

  const result = await new Promise<boolean>((resolve) => {
    contractWithSigner.once('DocumentStored', () => {
      setStoringDoc(false);
      showDocStoredSuccesfully();
      resolve(true);
    });
  });
  return result;
} catch (error: any) {
  if (error.code === 4001) {
    console.log('ACTION_REJECTED');
    showUserRejectedTx();
  }
  console.log(error);
  setStoringDoc(false);
  return false;
}
}
```

Obrázek 20: FE kód pro uložení nové struktury dokumentu do blockchainu

(zdroj: vlastní zpracování)

Druhým případem je, když **struktura dokumentu již existuje, ale jeho *metadataHash* v této struktuře ještě nikoli**. Pokud však existuje již i položka se stejnou hodnotou *metadataHash,* dostaneme od contractu revert, který zvrátí operaci uložení.

```
const docExistsBC = await verifyDocExistsBC(bcData.contentHash);

const signer = (documentContract.provider as JsonRpcProvider).getSigner();
const contractWithSigner = documentContract.connect(signer);

if (docExistsBC) {
  try {
    await contractWithSigner.addMetadata(
      bcData.contentHash,
      bcData.metadata[0].metadataHash,
      `${bcData.metadata[0].branchID}`,
      bcData.metadata[0].title
    );
    setStoringDoc(true);
    const result = await new Promise<boolean>((resolve) => {
      contractWithSigner.once('MetadataAdded', () => {
        setStoringDoc(false);
        showDocStoredUpdateBcStruct();
        resolve(true);
      });
    });
    return result;
  } catch (error: any) {
    if (error.code === 'ACTION_REJECTED') {
      showUserRejectedTx();
    } else {
      showCustomInfoMessage();
    }

    setStoringDoc(false);
    return false;
  }
```

Obrázek 21: FE kód pro uložení metadat do stávající struktury dokumentu do blockchainu

(zdroj: vlastní zpracování)

8.1.3 Přečtení dat z blockchainu

Přečtení dat z blockchainu vyžaduje volání funkce *getDocument,* která je typu ***view***, a tudíž bezplatná, a proto zde není potřeba podepisovatele. Tato funkce zpravidla trvá i méně času než zápis. Na ukázce získáme data do konstanty *response*, se kterými dále pracujeme dle potřeby.

```
const getBcDocData = async () => {
  try {
    const response = await documentContract.getDocument(docContentHash);
    setBcDocData({
      contentHash: docContentHash,
      metadata: response.metadata.map((metadataItem: any[]) => {
        return {
          metadataHash: metadataItem[0],
          branchID: metadataItem[1],
          title: metadataItem[2],
        };
      }),
    });
  } catch (error) {
    console.error('Error fetching document from contract:', error);
  }
};
```

Obrázek 22: FE kód pro přečtení záznamu z blockchainu

(zdroj: vlastní zpracování)

8.2 Komunikace FE s BE

Serverová aplikace je napsána v jazyku **Node.js**. Komunikace FE s BE probíhá na serverové straně přes knihovnu ***express.js***. Je také přidáno další ošetření aplikace ve formě **CORS** [23], které zajišťuje komunikaci pouze s definovanou URL adresou (v našem případě *https://www.docverifier.link/*).

```
require('dotenv').config();
const express = require('express');
const cors = require('cors');
const connectDB = require('./config/dbConfig');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const PORT = process.env.PORT || 5000;
const app = express();
connectDB();

app.use(
  cors({
    origin: 'https://www.docverifier.link',
    credentials: true,
  })
);
app.use(express.json());
app.use('/api/users', require('./routes/user'));
app.use('/api/documents', require('./routes/document'));
app.use(bodyParser.json());

// Use cors middleware to enable CORS
mongoose.connection.once('open', () => {
  console.log('Connected to MongoDB');
  app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
});
```

Obrázek 23: FE kód pro server spojení
(zdroj: vlastní zpracování)

Instance připojení do MongoDB databáze pak probíhá ve funkci *connectDB*, ve které figuruje knihovna ***mongoose*** pro vytvoření spojení pomocí ***connectionString.***

```
1   require('dotenv').config();
2   const mongoose = require('mongoose');
3
4   const connectDB = async () => {
5     try {
6       await mongoose.connect(process.env.MONGODB_URI, {
7         useUnifiedTopology: true,
8         useNewUrlParser: true,
9       });
10    } catch (err) {
11      console.error(err);
12    }
13  };
14
15  module.exports = connectDB;
16
```

Obrázek 24: FE kód pro připojení k MongoDB

[23] **Cross-Origin Resource Sharing (CORS)** = mechanismus, umožňující webovým stránkám komunikovat s jinými doménami a přistupovat k jejich zdrojům v bezpečném a kontrolovaném způsobu

Následné vytvoření api instance na straně klienta pro komunikaci s jednotlivými serverovými endpointy se zde provádí využitím knihovny ***axios*** (v našem případě na URL adresu *https://api.docverifier.link*).

Obrázek 25: FE kód pro inicializaci komunikace s endpointy backendu

(zdroj: vlastní zpracování)

8.3 Struktura backendu

Na backendové části jsou uložena citlivá data, která je potřeba řádně ošetřit tak, aby i když by nebyl přístup k jednotlivým metodám na frontendové části ošetřen, či byla poslána serveru nekorektní data, tak se k jejich uložišti dostal jen autorizovaný uživatel a server si i s tímto požadavkem dokázal poradit.

Není pro to radno mít plochou strukturu databáze. V této aplikaci je tedy backendová část hierarchicky uspořádána tak, aby se co nejefektivněji zaručila bezpečnost a stabilita aplikace. Struktura se skládá z několika klíčových vrstev: rout, modelů, middleware a kontrolerů. Každá z těchto vrstev hraje důležitou roli při zpracování uživatelských požadavků a komunikaci s databází.

Routy jsou první vrstvou, řekněme první zastávkou uživatelského požadavku. Deklarují směrování na middleware, a dále na kontroléry dle URL požadavku uživatele. Představují jakousi mapu a určují, jaká funkce (middleware nebo kontroleru) se má vykonat pro obsloužení tohoto požadavku.

Modely, podobně jako například v architektuře MVC, definují strukturu dat, která je ukládána do databáze. Každý model v MongoDB reprezentuje dokument v kolekci a určuje

jeho podobu a obsah dat, které mohou být do tohoto dokumentu uloženy. Modely jsou klíčové pro zachování konzistence dat a integrity dat. Toto je důležité zejména u NOSQL databází, jakou MongoDB je – jinak by bylo možné uložit data různých struktur, což v případě této aplikace není žádoucí.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const userSchema = new Schema(
  {
    userName: {
      type: String,
      required: true,
    },
    password: {
      type: String,
      required: true,
    },
    role: {
      type: Number,
      required: true,
    },
    docBranches: {
      type: [String],
      required: false,
    },
  },
  {
    // indexed on init (userName)
    autoIndex: true,
  }
);
module.exports = mongoose.model('Users', userSchema);
```

Obrázek 26: Schéma modelu uživatele

(zdroj: vlastní zpracování)

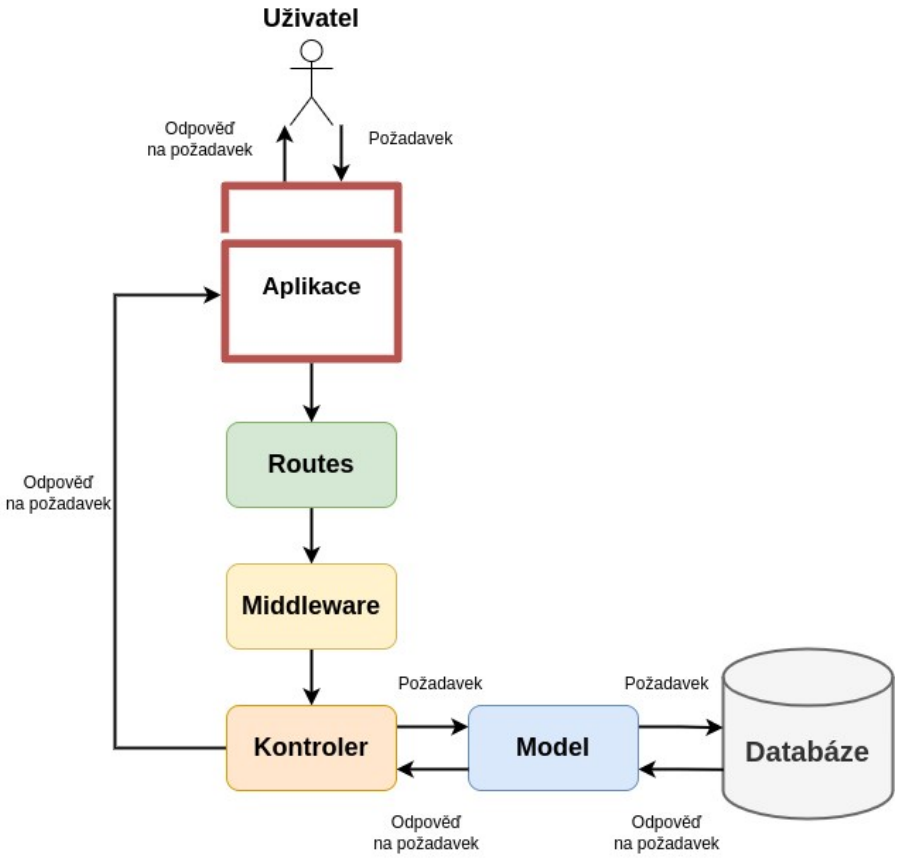
```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const documentScheme = new Schema(
  {
    branchID: {
      type: String,
      required: true,
    },
    files: {
      type: [
        {
          _id: {
            type: Schema.Types.ObjectId,
            required: true,
          },
          title: String,
          authorName: String,
          authorAddress: String,
          group: String,
          dateCreated: String,
          metadataHash: String,
          contentHash: String,
          content: Buffer,
        },
      ],
      required: false,
    },
  },
  {
    // indexed on init (branchID)
    autoIndex: true,
  }
```

Obrázek 27: Schéma modelu dokumentu

(zdroj: vlastní zpracování)

Middleware je vrstva mezi routami a kontroléry, která slouží jako prostředník. Provádí různé úkony na požadavcích předtím, než je předávají kontrolérům. Těchto middleware funkcí může být za sebou v rámci zpracování požadavku libovolné množství. Může zde probíhat například ověření uživatele, filtrování dat, případně zpracování chyb. V naší aplikaci jsou častými middleware funkce na ověření, zda je uživatel přihlášen a zda se jedná o administrátoru či prostého uživatele.

Kontroléry jsou poslední vrstvou před databází. Obsahují funkce pro přístup k datům a vrací uživateli odpovědi na jeho požadavky. Jsou tedy zodpovědné za komunikaci s databází, včetně vkládání, aktualizace, načítání a mazání dat.



Obrázek 28: Ukázka zpracování požadavku uživatele

(zdroj: vlastní zpracování)

8.3.1 Endpointy aplikace

Endpointy aplikace představují různé URL adresy a funkce, které umožňují interakci s aplikací. Tyto endpointy jsou k dispozici prostřednictvím HTTP metod, jako je GET, POST, PUT a DELETE, a slouží k práci s dokumenty a uživateli. Díky nim lze získávat informace o dokumentech, ukládat nové dokumenty, upravovat uživatelská práva nebo provádět autentizaci uživatelů. Endpointy aplikace tedy poskytují rozhraní pro efektivní správu dokumentů a uživatelů v rámci aplikace.

Tabulka 7: Seznam všech endpointů aplikace

(zdroj: vlastní zpracování)

HTTP metoda	URL	Funkce
GET	documents/	Vrací všechny dokumenty
POST	documents/	Ukládá dokument do dané větve dokumentů, včetně formátovaného obsahu
GET	documents/:documentId	Vrací dokument dle daného id
GET	documents/branches/:userName	Vrací větve dokumentů, na něž má daný uživatel právo
GET	documents/branch/:branchId	Vrací všechny dokumenty dané větve
GET	users/	Vrací všechny uživatele
POST	users/	Vytváří nového uživatele
PUT	users/	Upravuje (přidává/maže) uživateli větve dokumentů, na které má právo
GET	users/user/:userName	Vrací uživatele dle uživatelského jména
DELETE	users/:userID	Smazání uživatele dle userID
PUT	users/password	Aktualizuje heslo uživatele (parametry se posílají v body požadavku)
POST	users/login	Přihlásí uživatele a vrátí autorizační JWT token, přes který se přihlášený uživatel dále autorizuje
GET	users/admin	Vrací boolean hodnotu, zda je je přihlášený uživatel administrátorem

9 POPIS A STRUKTURA APLIKACE

Jelikož se jedná o webovou aplikaci kancelářského typu a nikoli o aplikaci, kde by byly velmi žádaná a nutné vysoké nároky na grafickou úpravu, a aplikace zároveň slouží především jako **příkladná forma možného hybridního souznění centralizovaného uložiště a decentralizované verifikační autority,** namísto použití jen blockchainových technologií či pouze centralizovaného uložiště (klasické databáze), je design pojat poměrně stroze, ostatně jak je ve většině kancelářských aplikací zvykem.

Frontendová část je rozdělena na pět rout, kdy každá slouží pro zobrazení jiného obsahu:

- **Hlavní stránka** (“/”): zobrazení větví dokumentů, ke kterým má uživatel přístup
- **Detail stránka** (“/documents/:id”): zobrazení všech dokumentů
- **Ukládací stránka** (“/store”): ukládání nových dokumentů do daných větví
- **Verifikační stránka** (“/verify”): ověření integrity dokumentů dle contentHash
- **Nastavení** (“settings”): slouží pro změnu hesla uživatele a správu uživatelů v případě přihlášení admina

9.1 Hlavní stránka aplikace

Hlavní stránka zobrazuje tabulku větví dokumentů, ke kterým má přihlášený uživatel právo. Administrátor má pochopitelně právo ke všem větvím.

9.2 Stránka detail aplikace

Detail stránka zobrazuje podrobnosti o dokumentech uložených v dané větvi. Nabízí také možnost stáhnutí obsahu těchto dokumentů (*pozn.: cloud server **EC2 v kombinaci s některými webovými prohlížeči bohužel blokuje stažení** těchto dokumentů).

9.3 Stránka pro uložení dokumentu

Přes **stránka pro ukládání** nových dokumentů se dá po přihlášení jak do aplikace, tak do **Metamask peněženky, uložit do blockchainu a databáze nahrány dokument**. Admin má právo uložit do libovolné větve, uživatel pouze do svých větví, a navíc má právo vytvořit novou větev s novým dokumentem. Uživatel tedy musí vyplnit všechny tyto údaje:

- **větev** dokumentu – na výběr má pouze z těch, ke kterým má právo;
- **název** dokumentu – název se může shodovat s již uloženým(i) dokument(y), ke konfliktu a upozornění uživatele dojde jen v případě, že je již uložen dokument se stejnými oběmi hashi
- **jméno** autora – je vyplněno automaticky dle přihlášeného uživatele
- **adresa** autora – uživatel se musí přihlásit do Metamask peněženky (adresa je tedy adresa této autorem vybrané peněženky)
- **soubor** dokumentu pro uložení – obsah souboru je do databáze uložen v binární podobě a utf-8 kódování

Po vyplnění všech vstupních polí proces následující:

1. **Výpočet metadataHash a contentHash**
    - metadatahash = title+authorName+authorAddress+dateCreated
        - dataCreated obsahuje aktuální datum a čas, **včetně milisekund (pojištění zaručení unikátní metadataHash)**;
    - contentHash = obsah souboru v binární podobě a utf-8 kódování
2. **Inicializace blockchainové transakce** – otevře se Metamask peněženka požadující podpis
3. **Podepíšeme transakci v Metamask** aplikaci – v případě zamítnutí se transakce i zbytek ukládacího procesu zruší
4. **Probíhá zápis do blockchainu, kontrakt čeká na zapsání dokumentu** – sleduje událost
5. Po **úspěšném zapsání na blockchain se spustí ukládací proces do databáze**

Obrázek 29: Formulář aplikace pro uložení dokumentu

(zdroj: vlastní zpracování)

9.4 Stránka ověření

Stránka *verify* slouží pro ověření integrity dokumentů. Je proto pravděpodobně nejdůležitější stránkou naší aplikace. Dokumenty se ověřují pomocí vyhledání se zadáním ***contentHash***, který lze **zkopírovat ze *stránky detail***, ve které je dokument uložen. Výsledek nám ukazuje, zda je integrita dokumentu stále zachována, či nikoli.

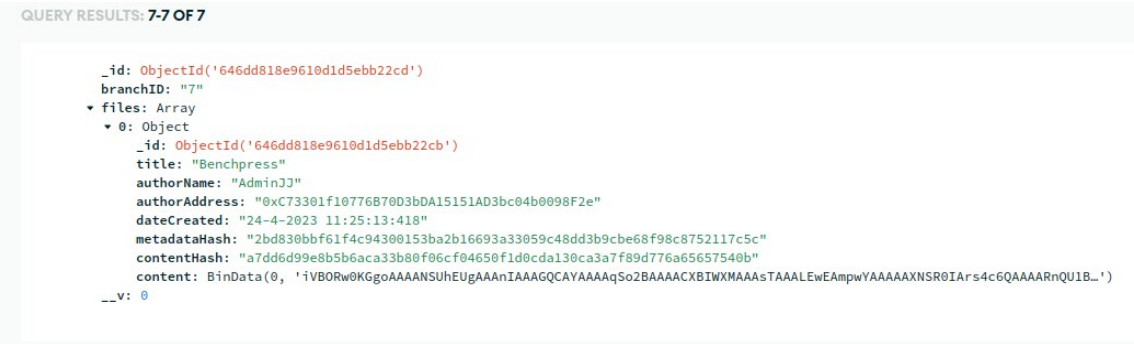
9.5 Proces verifikace integrity

Samotné ověření integrity se ověřuje hned dvěma způsoby:

- Ověření uložené metadataHash z databáze vůči hash vypočítané ze získaných atributů metadat z databáze (jméno autora, adresa peněženky autora, název dokumentu a čas jeho uložení)
- Ověření vypočítané metadataHash z atributů z databáze vůči uloženému názvu dokumentu ve vnořené struktuře Metadata na blockchainu

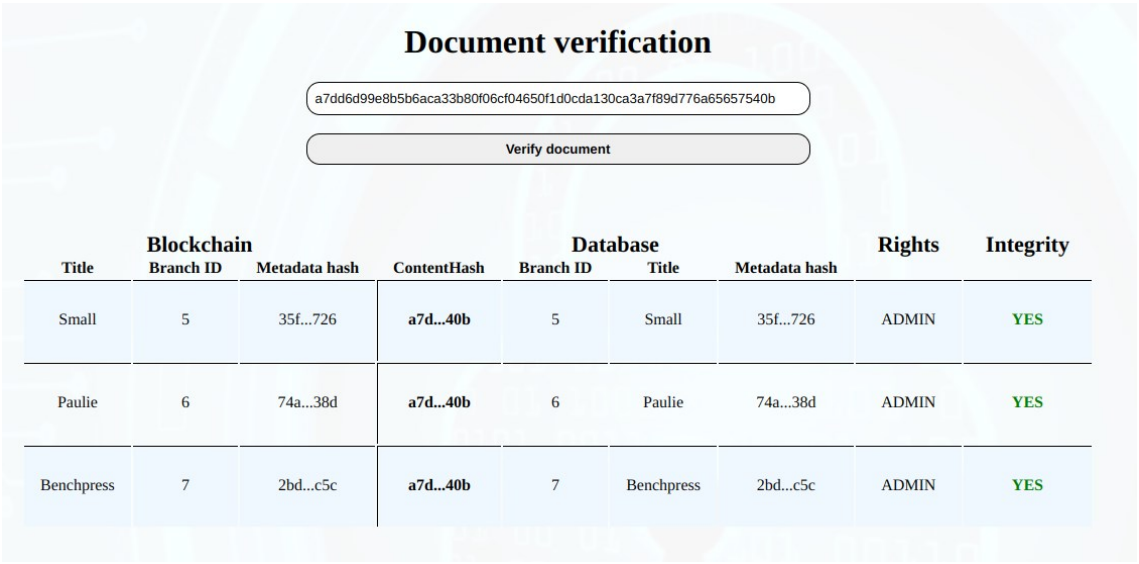
Všechny tři hodnoty o metadataHash se musí shodovat.

Po prvotním uložení dat je vidět, že je zachovaná integrita u dokumentu s názvem Benchpress a (obrázkem s Paulie Garandem a dalšími atributy) – viz uložení **Obrázek 29**.



Obrázek 30: Ukázka uloženého dokumentu v MongoDB

(zdroj: vlastní zpracování)



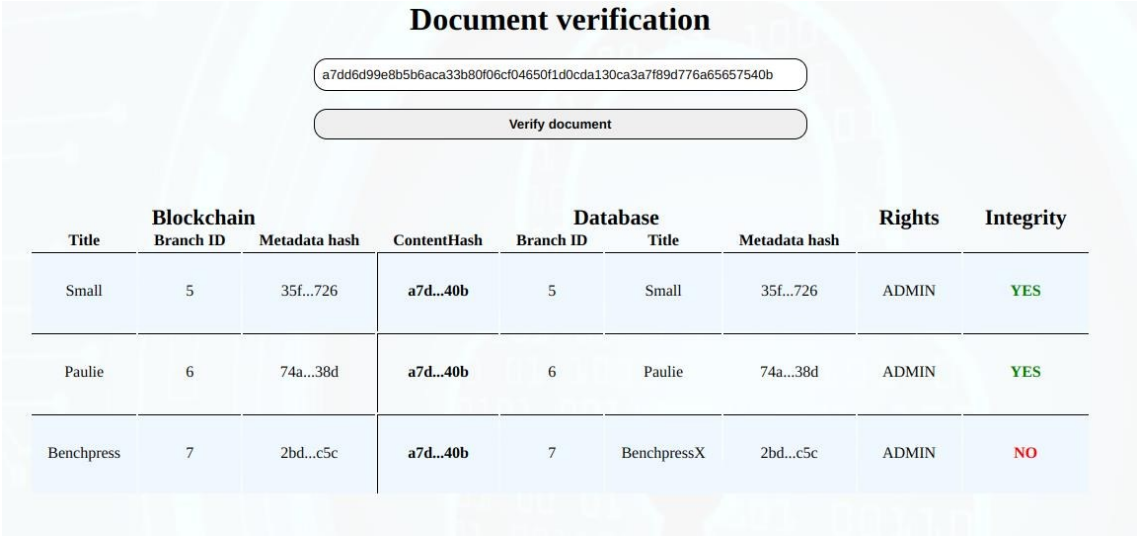
Obrázek 31: Ukázka ověření integrity v aplikaci

(zdroj: vlastní zpracování)

Po změně dokumentu přímo v databázi (zde přidáním "**X**" do názvu dokumentu), nás verifikační stránka upozorní, že data již nejsou originální, což se patřičně zobrazí na výsledku ověření - viz **Obrázek 32** a **Obrázek 33**.

Obrázek 32: Ukázka změny názvu dokumentu v MongoDB

(zdroj: vlastní zpracování)



Obrázek 33: Ukázka ověření integrity v aplikaci po změně názvu souboru

(zdroj: vlastní zpracování)

9.6 Stránka nastavení

Tato stránka slouží jako **správce uživatelů** pro přihlášeného **administrátora**. Má zde možnost přidat nového uživatele a přidat či odebrat větve dokumentů, na něž mají tito uživatelé právo.

Prostý **uživatel** má pak možnost si **změnit své vlastní heslo**, při vložení správného aktuálního a opětovném zvolení hesla nového

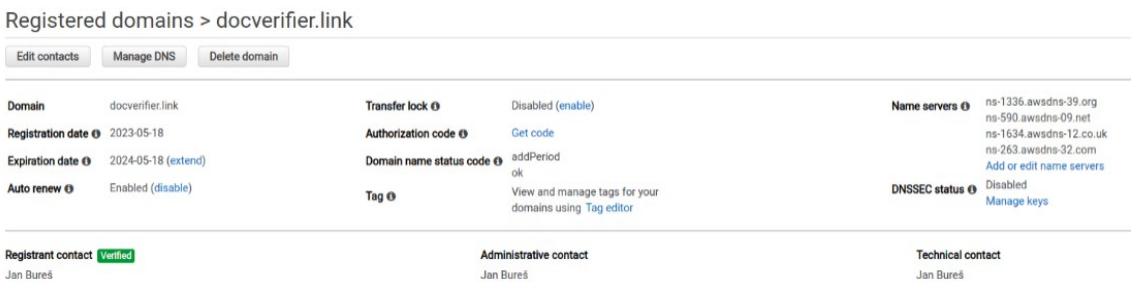
10 HOSTOVÁNÍ APLIKACE

Výběr hostingových služeb je nesmírně podstatný. Často určuje úspěch či neúspěch v podobě zavděčení se uživateli z pohledu jeho nároků a celkového uživatelského zážitku. Jsou to totiž právě tyto služby, které přenáší obsah aplikaci do uživatelských zařízení. Je proto opravdu důležité dát výběru a hostingové konfiguraci patřičnou pozornost. Pro tuto příležitost byla vybrána AWS konfigurace, která obsahuje:

- **1x Amazon Doména;**
- **2x Amazon S3 bucket;**
- **Amazon Route 53;**
- **AWS Certificate manager;**
- **Amazon Cloudfront;**
- **Amazon EC2.**

10.1 Registrace domény pomocí Amazon Route 53

Amazon figuruje mimo jiné jako správce a registrátor doménových jmen. Registrovat doménu zde lze přes službu Route 53, která nabízí přes 70 různých top-level domén [24] (z různou cenou). Stačí tedy zvolit z nabídky doménové jméno, zvolit top-level doménu a zkontrolovat její dostupnost – zda ještě není nikým registrovaný. Posledním krokem zbývá vyplnit kontaktní údaje na osobu registrující osobu a vyčkat na ověření (otázka několika minut). V tomto bodě jsme právoplatnými vlastníky domény a máme tak kontrolu na správu jejich DNS záznamů atd.



Obrázek 34: Registrovaná doména pomocí Amazon Route 53

[24] **Top-level doména** = označení nejvyšší úrovně hierarchie doménového jmenného systému (DNS); jsou umístěny za poslední tečkou v internetové adresy

10.2 Konfigurace Amazon S3 bucket

Dále je za potřebí vytvořit a nastavit uložiště, kam se bude nahrávat obsah aplikace, tedy její zdrojový kód. V případě frontendové části aplikace, kterou chceme hostovat na právě koupené doméně, se jedná o produkční verzi aplikace. Jako toto uložiště (balíček) v naší konfiguraci slouží Amazon S3 bucket.

V tomto případě byla vybrána konfigurace pomocí **dvou Amazon S3 bucket**. První, se jménem včetně "***www"*** prefixu - "*www.docverifier.link",* bude sloužit jako **autoritativní** lodivod, do kterého se nahraje zmiňovaná produkční verze aplikace a povolí se veřejný přístup. **Druhý bucket,** bez prefixu - *"docverifier.link",* **bude sloužit k přesměrování**. Neponese tedy žádný obsah, ale bude sloužit se směrování uživatelů na doménu s prefixem (v případě, že zadají do adresního řádku doménovou adresu bez tohoto prefixu). Je důležité zmínit, že **názvy bucketů musí nést stejné jméno jako doménové jméno**.

10.3 Konfigurace Amazon Route 53

Hostovaná zóna u Route 53 musí opět **nést jméno doménového jména** - "*docverifier.link*". Po inicializaci jsou zóně přiřazeny v základu dva DNS záznamy: **NS a SOA**. NS (Name Server), je identifikátor serverů, které jsou zodpovědné za poskytování DNS informací a správu přiřazené domény (AWS přiřazuje v základu čtyři různé servery) a SOA (Start Of Authority), což je identifikátor autoritativního DNS serveru, který má na starost skupinu domén (v tomto případě Amazon DNS server).

V tomto bodě je vhodné přejít na konfiguraci ostatních služeb a na konec nastavit zbývající záznamy. Pro účel aplikace přibude ještě 5 dalších DNS záznamů. Dva záznamy typu **CNAME** (Canonical Name), tedy alias pro přesměrování na jinou doménu, se vloží na žádost v AWS certifikačním správci automaticky do DNS zóny. Zde tyhle záznamy slouží jako ověření skutečného vlastníka pro SLL certifikáty.

Další dva záznamy typu **A** (Address), které jsou slouží pro mapování/směrování doménového jména na odpovídající IPv4 adresu. Jeden pro doménu s a druhý pro doménu bez *"www"* prefixu a obě nesoucí jméno příslušné Cloudfront distribuce.

Poslední záznam je typu taktéž typu A, ale přesměrovává se na **subdoménu - adresu EC2 server**, na kterém běží serverová části aplikace.

Tabulka 8: Seznam DNS záznamů pro hostování ukázkové aplikace

(zdroj: vlastní zpracování)

Jméno záznamu (pseudo)	Typ záznamu	Hodnota/směrování na (pseudo)
docverifier.link	NS	ns-1336.awsdns-39.org. ns-590.awsdns-09.net. ns-1634.awsdns-12.co.uk. ns-263.awsdns-32.com.
docverifier.link	SOA	ns-1336.awsdns-39.org. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
docverifier.link	A	d2vb5ghat4jnva.cloudfront.net.
www.docverifier.link	A	d2l69zkdrag911.cloudfront.net.
api.docverifier.link	A	35.175.112.251
_6713e73eba20ee761d7abb0fd6dc4ddb.docverifier.link	CNAME	_e0dae1821b32efc66c31ce380316aa82.sxgfwvgjsz.acm-validations.aws.
_c4283eea56194b00f88d55f7ccc5658a.www.docverifier.link	CNAME	_0397d342473196a1a39942e4e1223c23.sxgfwvgjsz.acm-validations.aws.

10.4 Zisk SSL certifikátu

Pro šifrovaný přenos dat přes protokol https, který je v dnešní době opravdu nutností je nutné mít autoritativně podepsaný SSL certifikát. Tyto certifikáty nabízí služba AWS Certificate Manager.

Pro jeho zisk je třeba být majitelem domény, pro kterou tento certifikát nebo alespoň mít právo úpravy jejich DNS záznamů. Poté stačí přes zmiňovaného správce zažádat o certifikát, vyplnit doménové jméno, v našem případě dvě, vybrat ověření přes DNS a typ šifrovacího algoritmu.

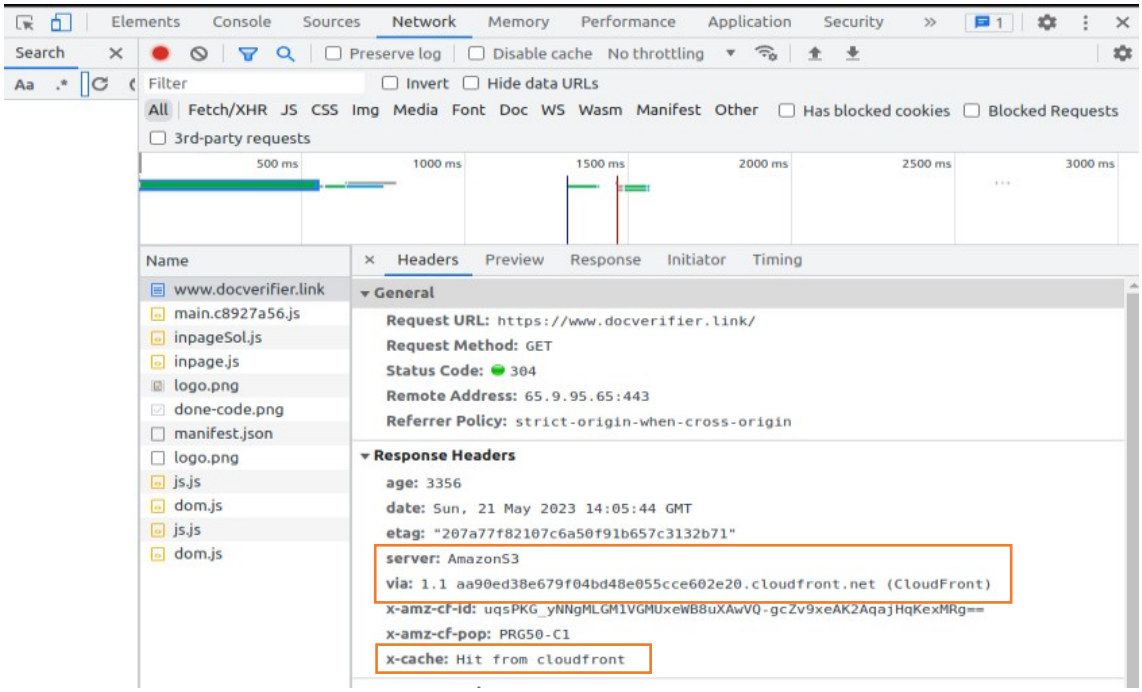
Pro vytvoření ověřovacích DNS záznamů poté stačí zažádat v jeho instanci o přidání CNAME záznamů do Route 53. Toto ověření zabere většinou pouze pár okamžiků.

10.5 Konfigurace Amazon Cloudfront

Po získání SSL certifikátu je na čase vytvořit vhodnou Cloudfront distribuci. Tato distribuce nabízí opět širokou škálu nastavení, obdobně jako všechny služby AWS. Zde je vhodné zdůraznit, že jako jméno domény („Origin domain") není vhodné zvolit přímo nabízené možnosti z aktuálních názvů S3 bucketů, nýbrž zkopírovat vložit **URL adresu statického webového hostingu**. Tuto adresu lze nalézt v záložce vlastností – „Properties" daného S3 bucketu. Pro náš případ se dvěma S3 buckety, je nutné vytvořit dvě Cloudfront distribuce. Poté pro osvědčenou konfiguraci se základním nastavením parametrů pro naši aplikaci vypadá následovně (distribuce pro bucket s prefixem *www* je obdobná):

- Origin Domain Name:
    - *http://docverifier.link.s3-website.eu-central-1.amazonaws.com;*
- Enable Origin Shield – přidaná cache vrstva, která pomáhá redukovat tlak na origin a pomáhá udržet přístupnost díky nižší latenci;
- Viewed Protocol Policy: HTTP to HTTPS;
- Allowed HTTP methods: GET, HEAD, OPTIONS;
- Alternate Domain Names: *docverifier.link;*
- Custom SSL Certificate: vytvořený pro stejnou doménu.

Ověření, že naše aplikace skutečně využívá Cloudfront distribuci, lze provést například ve vývojářských nástrojích webového prohlížeče v záložce "*Network*".



Obrázek 35: Potvrzení užití Cloudfront v DevTools

(zdroj: vlastní zpracování)

Je taktéž na místě poznamenat, že Cloudfront distribuce ukládá obsah S3 bucketu do cache paměti[25], tudíž se změna obsahu neprojeví na hostovaných webových stránkách ihned, může to trvat až několik hodin (interval aktualizace lze změnit v nastavení Cloudfront distribuce). Pokyn k aktualizaci lze ale dát manuálně pomocí invalidace (záložka "Invalidations"). Základní příkaz "/*" vymaže během pár okamžiků obsah celé složky (pokud nám stačí aktualizace pouze části zdrojového kódu, zadáme pouze tu; tj. čím větší datový obsah invalidace aktualizuje, tím vyšší je i cena za ni... u menších aplikací řádově pár centů USD) z mezipaměti Cloudfront a ten si tedy nově nahraje aktualizovaný obsah.

[25] **Cache paměť** = mezipaměť pro ukládání dočasných dat pro rychlejší opakovaný přístup k nim

10.6 Konfigurace Amazon EC2

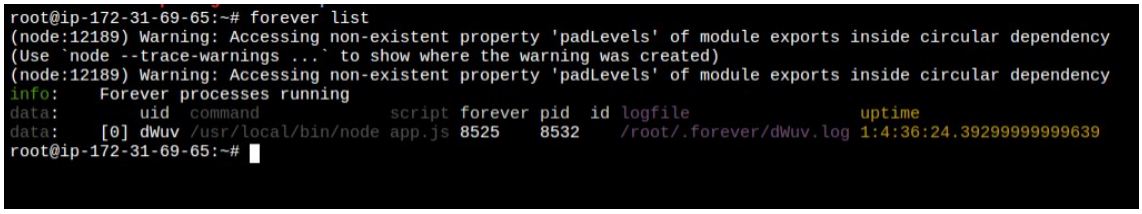
Pro pronájem cloudového serveru stačí vytvořit EC2 instanci a vyplnit několik parametrů. Kromě výchozího nastavení je vhodné upravit tyto parametry:

- Name: <názevInstance>;
- OS: zde bylo zvoleno **Ubuntu** z důvodu dalšího nastavení knihoven závislých na tomto OS;
- Allow HTTPS traffic from the internet;
- Allow HTTP traffic from the internet;
- Advanced details -> IAM instance profile: IAM role s právy na *"**AmazonSSMManagedInstanceCore**".*

Po vytvoření instance je nutné se připojit do terminálu serveru, tam nahrát zdrojový kód serverové části naší aplikace. Pro získání SSL certifikátu byly využity knihovny **nginx,** sloužící jako reverse proxy, za kterým běží naše serverové Node.js API**.** Nginx je zodpovědný za ukončení TLS, které je generováno pomocí **Certbota** a **Let's Ecrypt.** Certbot je klientská aplikace, která umožňuje generování certifikátů z Let's Encrypt a automatizuje jejich správu.

V tomto bodě by měl být nastaven šifrovaný přenos přes protokol HTTPS. Nicméně spuštění naší serverové aplikace pomocí příkazu **"n*ode app.js*"**, protože takto se aplikace spustí jako **ssh relace**, pod kterou běží naše serverová aplikace, což je výchozí nastavení serveru pro opatření vůči přetěžování nevyužívaného cloud výpočetního výkonu. To způsobí **pouze dočasné spuštění**, které po určité době neaktivity vyústí v **přerušení spojení serveru z portu**, kde běží hostingová služba (naše aplikace).

Řešením je spuštění jako samostatný proces, který není podprocesem ssh relace. To je možné dosáhnout například s pomocí **npm balíčku "*forever*",** který zajišťuje, že daný skript se spouští kontinuálně, takže jej sleduje a v případě jeho výpadku, jej opět spustí. Službu takto spustíme příkazem **"*forever start app.js*".**



Obrázek 36: Spuštění serverové aplikace jako samostatný proces

(zdroj: vlastní zpracování)

Pro zprovoznění, tedy nasměrování tohoto cloudového serveru s běžící hostovanou serverovou službou je potřeba přidat pouze jediný DNS záznam ve službě Route 53 hostované domény frontendové části. Konkrétně záznam typu A, který směruje provoz na veřejnou adresu, v tomto případě veřejnou IPv4 adresu pronajatého cloudového serveru).

11 POROVNÁNÍ APLIKACE S EXISTUJÍCÍMI ŘEŠENÍMI

V aktuální době nabízí řada poskytovatelů opravdu propracované a hlavně robustní řešení ověření integrity dokumentů. Metoda digitálního podpisu jednoznačně dominuje při slovech vyřčených směrem k tomuto požadavku. Do hry se dostávají také doplňující služby jako je verzování dokumentů nebo velmi pokročilé funckcionality jako je například ověřování dokumentů pomocí biometrie.

Někteří poskytovatelé mají dokonce velkou právní důvěru, protože splňují mezinárodní standardy a činí je tak právně závazné a globálně uznávané, jiní zase spoléhají na rozmach nejmodernějších technologií, jako je blockchain, který díky svému základnímu „append-only“ principu, plní roli spolehlivé decentralizované autority.

Naše aplikace sice nevyužívá digitálního podpisu, ani ověření pomocí technik biometrie, či podobným, stejně pokročilým. Spojuje ale základní metody ukládání informací o dokumentu pomocí databází a přidává několik prvků navíc, ne zas tolik obvyklých, **jako je verzování dokumentu, sdílení formátovaného obsahu dokumentů a především přední techniky oblasti blockchainu – podobně jako konkurenční řešení**.

11.1 DocuSign

DocuSign je velmi rozšířeným poskytovatelem řešení pro digitální transakce, což zahrnuje zejména elektronické podpisy. Tato platforma nabízí uživatelům snadný a bezpečný způsob **elektronického podpisu** a správy dokumentů a smluv.

Autentizace uživatele probíhá několika způsoby, včetně ověřování e-mailem, SMS, telefonními hovory a osobními otázkami. DocuSign podporuje také pokročilejší metody, jako je **biometrie** a ověřovací kódy.

Integrita dokumentu je zajištěna šifrováním dokumentu. Jakékoli změny v dokumentu po podepsání jsou tedy snadno detekovatelné.

DocuSign také splňuje různé mezinárodní bezpečnostní a právní standardy, jako je **eIDAS** v EU a **ESIGN** v USA, což zajišťuje, že elektronické podpisy jsou právně závazné a globálně uznávané [51].

11.2 Adobe Document Cloud

Adobe Document Cloud je platforma pro digitální správu dokumentů, která zahrnuje solidní množství nástrojů pro manipulaci, sdílení a ověřování dokumentů. Jednou z klíčových složek Adobe Document Cloud je Adobe Acrobat DC, který umožňuje vytváření, úpravy a konverzi dokumentů ve formátu PDF. Zahrnuje také funkcionalitu pro **elektronický podpis** dokumentů, což je základní princip ověření integrity sdílených dokumentů. Tento elektronický podpis je **právně závazný** a snadno sledovatelný, což zaručuje autenticitu a neporušenost dokumentu.

Adobe Document Cloud také poskytuje funkcionalitu pro sledování změn v dokumentech, která umožňuje detekci jakýchkoli modifikací po vytvoření nebo podepsání dokumentu. To zvyšuje důvěru v integritu a autenticitu dokumentů sdílených prostřednictvím platformy [52].

11.3 Diplomachain

Diplomachain, na jejímž vznikem stojí hned několik institucí, mezi něž patří i zlínský PT LAB, ELA Blockchain Services či Gordic, je decentralizované řešení ověření pravosti dokumentu, původně **vzniklé za účelem ověření pravosti především certifikátů, digitálních osvědčení a diplomů**. Systém ověření spočívá v uložení informací o dokumentu v pomocí hashů. Nelze tak nijak získat zpětnou rekonstrukci původních dat, a tak je zajištěna současně i ochrana těchto možných citlivých dat. Díky blockchainu je obtížné data padělat nebo změnit, což posiluje důvěru v dokumenty [53].

11.4 IBM Blockchain

Jedná se o robustní blockchainovou platformu, navrženou pro zvýšené množství důvěry a tranparentnosti digitálních transakcí, včetně manipulaci s dokumenty.

Princip ověření integrity dokumentů v IBM Blockchain je založen na základní vlastnsti vlastnostech blockchainové technologie. Každá tr**ansakce nebo úprava dokumentu je totiž zaznamenávána jako blok na blockchainu**, který je následně spojen s předchozími bloky za využití kryptografických funkcí. Tento proces zaručuje neporušitelnost a autenticitu

dokumentu, jelikož jakékoli následné změny by vyžadovaly modifikaci všech následujících bloků v řetězci, což je v praxi téměř nemožné.

IBM Blockchain také umožňuje decentralizovanou kontrolu a správu dokumentů. To znamená, že žádná jediná strana nemůže manipulovat s dokumentem bez vědomí ostatních účastníků v síti. Tato vlastnost zvyšuje důvěru a transparentnost při manipulaci s dokumenty [54].

12 SUMARIZACE APLIKACE

Implementovaná aplikace je **unikátní** ve svém řešení z pohledu spojení více přístupů do jediného. Přístup s privátním sdílením **v kombinaci s decentralizovanou autoritou (blockchainem) pro ověření integrity dokumentů, dále sdílení formátovaných dokumentů a verzování**, by rozhodně mohl vzbudit patřičný zájem.

V moderní době, kdy má uživatel často vysoké nároky na rychlost zpracování svých dat, by mohlo být zpracování **blockchainovou** částí, kde konečné zapsání transakce do bloku je dosti náchylné na aktuální vytížení sítě, poněkud **zdlouhavé**, i přestože je využito dosti svižného řešení z této oblasti, což lze zajisté brát jako nevýhodu této implementace.

Server, který aktuálně zpracovává požadavky aplikace je jediným serverem. To čas od času způsobí, že uživatelský zážitek není 100%, například z důvodu přetížení tohoto serveru, což má za následek velmi pomalé, ba dokonce občas žádné odpovědi na požadavek uživatele směrem k databázi (vypršením stanovené maximální doby čekání na odpověď). **Přínosným vylepšením,** a zároveň "best-practice" [26], by bylo rozhodně tento systém škálovat. Jinak řečeno, **distribuovat** na více serverů, kdy v případě neschopnosti hlavního, prvotně pověřeným zpracováním požadavku, delegovat toto zpracování na některý dostupný distribuovaný server, popřípadě využít techniky paralelního zpracování dat několika servery současně - "**sharding**". Dalším krokem k vylepšení by rozhodně bylo přidání mnohokrát osvědčených metod moderního zabezpečení, tedy **aplikovat asymetrické šifrování**, například ve formě digitálního podpisu.

[26] **Best-practice** = označení zpracování řešení vhodným/moderním způsobem pro danou problematiku

ZÁVĚR

Cílem této práce bylo nastudování technologií Web2.0 a Web3.0, s následným cílem vyběru těch nejvíce vhodných k tvorbě decentralizované aplikace pro privátní sdílení dokumentů.

V teoretické části byly podrobně analyzovány technologie Web2.0 a Web3.0. Současné trendy, zažité techniky a postupy byly zkoumány, s cílem identifikovat jejcih kladné i negativní stránky, stejně tak jako možnosti aktuálního a budoucího využití.

Důležitou část tvořila analýza specifických mechanismů blockchainu Ethereum, jeho druhů sítí a z nich plynoucí výhody, nevýhody a specifika využití.

Praktická část práce se zaměřuje na zdůvodnění výběru technologií pro vývoj decentralizované aplikace pro privátní sdílení dokumentů, a to na základě poznatků z teoretické části.

Pro frontendovou část aplikace byl zvolen **React** v kombinaci **Typescriptem**. Serverová část je psaná v jazyce **Node.js.** Byly zvoleny a aplikovány vhodné vrstvy pro lepší škálovatelnost backendové části s cílem zvýšení robustnosti aplikace. Konkrétně byl použit framework **Express.js**, který slouží jako střední vrstva mezi serverem a databází **MongoDB**.

Co se týče hostování aplikace, byla zvolena komplexní sestava služeb rodiny **Amazon Web Services.** Jmenovitě byly zvoleny služby: **S3 bucket** pro ukládání dat**, Route 53** pro správu doménovách jmen**, Cloudfront** jako distributor obsahu S3 bucketu**, Certificate Manager** správcem SSL/TLS certifikátů a **EC2** pro virtuální servery. Kombinace těchto služeb poskytuje opravdu silné a škálovatelné řešení pro hostování webových aplikací

Dále byly detailně popsány postupy při práci, specifika a nastavení jednotlivých technologií, zdůvodnění konkrétních implementací, důležité kroky při vývoji. Nezbytnou součástí bylo také popsání způsobu vzájemné komunikace jednotlivých částí aplikace.

K závěru bylo **porovnáno řešení aplikace s konkurenčními řešeními** z oblasti Web2.0 i Web3.0. Z této částí vyplynulo, že aplikaci chybí techniky šifrování nebo biometrického ověřování. Také nesplňuje parametry pro oficiální verifikační autority, jako je tomu u některých konkurenčních služeb. **Ukazuje však spojení technik Web2.0** (**verzování** a **ukládání komplexních dokumentů** o větších datových velikostech do **databází**) **a Web3.0** (**decentralizované ověření uložených hash otisků**, obdobným způsobem jako konkurenční služby z této sféry).

Práce na této diplomové práci poskytla hlubší vhled zejména do oblastí backendového vývoje a hostingu. Byly nabity vědomosti, které budou cenným přírůstkem pro další profesní vývoj.

SEZNAM POUŽITÉ LITERATURY

[1] KENTON, Will. What Is Web 2.0? Definition, Impact, and Examples. In: *What Is Web 2.0* [online]. .: Investopedia, 2022 [cit. 2023-04-09]. Dostupné z: https://www.investopedia.com/terms/w/web-20.asp

[2] W3SCHOOLS. *HTML Introduction* [online]. In: . 2021. Dostupné také z: https://www.w3schools.com/html/html_intro.asp

[3] JavaScript Statistics 2023: Key Insights And Trends: JavaScript: Statistics Overview. In: *Gitnux* [online]. Dostupné také z: https://blog.gitnux.com/javascript-statistics/

[4] MEGIDA, Dillion. *What is JavaScript? A Definition of the JS Programming Language* [online]. In: . 2021. Dostupné také z: https://www.freecodecamp.org/news/what-is-javascript-definition-of-js/

[5] *Express.js* [online]. In: . GeeksForGeeks. Dostupné také z: https://www.geeksforgeeks.org/express-js/

[6] DEVATHON. *MEAN vs MERN Stack Development* [online]. In: . Medium. Dostupné také z: https://medium.com/@devathon_/mean-vs-mern-stack-development-5ba3e517bc68

[7] *TypeScript Introduction* [online]. In: . W3Schools [cit. 2023-05-16]. Dostupné z: https://www.w3schools.com/typescript/typescript_intro.php

[8] AGRAWAL, Vardhan. *Design Patterns for Cocoa: MVC and MVVM: MVC (Model-View-Controller)* [online]. In: . Envato plus [cit. 2023-05-17]. Dostupné z: https://code.tutsplus.com/articles/cocoa-architectural-patterns-mvc-and-mvvm--cms-30265

[9] NRWL. *Understanding Angular Ivy: Incremental DOM and Virtual DOM* [online]. In: . Dostupné také z: https://blog.nrwl.io/understanding-angular-ivy-incremental-dom-and-virtual-dom-243be844bf36

[10] CHAUDHARY, Ray. *Its Zustand Vs Redux!* [online]. In: . [cit. 2023-05-10]. Dostupné z: https://medium.com/readytowork-org/its-zustand-vs-redux-8e24424df713

[11] *Introducing JSX: Why JSX?* [online]. In: . React [cit. 2023-05-10]. Dostupné z: https://legacy.reactjs.org/docs/introducing-jsx.html

[12] React Native: Learn once, write anywhere. In: *React Native* [online]. [cit. 2023-05-10]. Dostupné z: https://reactnative.dev/

[13] SHIFF, Laura a Walker ROWE. *SQL vs NoSQL Databases: What's The Difference?* [online]. In: . bmc. Dostupné také z: https://www.bmc.com/blogs/sql-vs-nosql/

[14] *What are NoSQL databases?: Discover NoSQL, a type of database design that offers more flexibility than a traditional database* [online]. In: . IBM [cit. 2023-05-12]. Dostupné z: https://www.ibm.com/topics/nosql-databases

[15] DOC. ING. ZDENKA PROKOPOVÁ, CSC. *Úvod do „pokročilých" databází: Od SQL k NoSQL*. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky. Prezentace.

[16] HEJAZI, Yasmine. *Introduction to NoSQL Graph Databases: An overview of graph database types, structures, and properties* [online]. In: . TowardsDataScience. Dostupné také z: https://towardsdatascience.com/introduction-to-nosql-graph-databases-fb2feac7a36

[17] GILLIS, Alexander a Bridget BOTELHO. *MongoDB: How does MongoDB work?* [online]. In: . [cit. 2023-05-10]. Dostupné z: https://www.techtarget.com/searchdatamanagement/definition/MongoDB

[18] HAYES, Adam. *Blockchain Facts: What is it, How it works, and How it can be used: What Is a Blockchain?* [online]. Tel Aviv, Israel: Investopedia, 2022 [cit. 2022-11-19]. Dostupné z: https://www.investopedia.com/terms/b/blockchain.asp

[19] Welcome to the IPFS docs. In: *IPFS docs* [online]. .: IPFS, . [cit. 2023-04-04]. Dostupné z: https://docs.ipfs.tech/

[20] BECHER, Brooke. What Are Blockchain Nodes and How Do They Work?. In: *BuiltIn* [online]. .: BuiltIn, 2022 [cit. 2023-04-04]. Dostupné z: https://builtin.com/blockchain/blockchain-node

[21] BECHER, Brooke a Brennan WHITFIELD. *What Is a Consensus Mechanism?: Consensus mechanisms keep nodes in a distributed ledger on the same page.* [online]. In: . BuiltIn. Dostupné také z: https://builtin.com/blockchain/consensus-mechanism

[22] *Exploring Proof-of-Work's Electricity Consumption* [online]. In: . Moralis [cit. 2023-05-13]. Dostupné z: https://academy.moralis.io/blog/exploring-proof-of-works-electricity-consumption

[23] OVADIA, Guy. *One Coin to Rule Them All: The Best Proof of Stake Coins To Watch in 2023* [online]. In: . MoneyMade. Dostupné také z: https://moneymade.io/learn/article/best-proof-of-stake-crypto-coins

[24] CRYPTOSLAV, Ivan. What Are Bitcoin Ordinals? The Ultimate Guide To Bitcoin NFTs. In: *Coinmarketcap* [online]. .: coinmarketcap, 2023 [cit. 2023-04-04]. Dostupné z: https://coinmarketcap.com/alexandria/article/what-are-bitcoin-ordinals-the-ultimate-guide-to-bitcoin-nfts

[25] NELSON, Jason. *What Are Ordinals? A Beginner's Guide to Bitcoin NFTs: The Bitcoin community has gone wild over Ordinal Inscriptions, but what is the new thing taking over Crypto Twitter?* [online]. In: . [cit. 2023-05-11]. Dostupné z: https://decrypt.co/resources/what-are-ordinals-a-beginners-guide-to-bitcoin-nfts

[26] *The Most Popular Blockchain Networks* [online]. In: . Kriptomat [cit. 2023-05-11]. Dostupné z: https://kriptomat.io/blockchain/most-popular-blockchain-networks/

[27] *ETHEREUM ROADMAP: The Merge* [online]. In: . Ethereum, 2022 [cit. 2023-05-11]. Dostupné z: https://ethereum.org/en/roadmap/merge/

[28] *Top 9 blockchain platforms to consider in 2023* [online]. In: . Tech target [cit. 2023-05-11]. Dostupné z: https://www.techtarget.com/searchcio/feature/Top-9-blockchain-platforms-to-consider

[29] *NETWORKS* [online]. In: . Ethereum [cit. 2023-05-11]. Dostupné z: https://ethereum.org/en/developers/docs/networks/

[30] NNABUNEYI JR., Chuks. *Mainnet and Testnet: what they are and their differences* [online]. In: . Crypto TV Plus [cit. 2023-05-11]. Dostupné z: https://cryptotvplus.com/2023/04/mainnet-and-testnet-what-they-are-and-their-differences/

[31] *Ethereum for everyone: Scaling Ethereum without compromising on security or decentralization.* [online]. In: . [cit. 2023-05-11]. Dostupné z: https://ethereum.org/en/layer-2/

[32] *Ethereum Average Transaction Fee (I:EATFND)* [online]. In: . YCharts. Dostupné také z: https://ycharts.com/indicators/ethereum_average_transaction_fee

[33] *11 Best Layer 2 Crypto Projects To Check Out Post-Merge (2023)* [online]. In: . Bybit Learn. Dostupné také z: https://learn.bybit.com/investing/best-ethereum-layer-2-crypto-projects/

[34] *Polygon: Market Cap* [online]. In: . CoinMarketCap [cit. 2023-05-17]. Dostupné z: https://coinmarketcap.com/currencies/polygon/

[35] *Polygon: Blockchains for mass adoption* [online]. In: . Polygon [cit. 2023-05-17]. Dostupné z: https://polygon.technology/

[36] MAKORI, Josiah. *Polygon vs. Ethereum: DeFi, NFTs, Gas Fees, and More: Polygon vs. Ethereum* [online]. In: . CoinGecko. Dostupné také z: https://www.coingecko.com/learn/polygon-vs-ethereum

[37] *Ethereum Testnets: List of 3 Testnets on Ethereum* [online]. In: . Alchemy [cit. 2023-05-11]. Dostupné z: https://www.alchemy.com/list-of/testnets-on-ethereum

[38] *MUMBAI/WMATIC: Real-time On-chain DEX data* [online]. In: . CoinMarketCap [cit. 2023-05-20]. Dostupné z: https://coinmarketcap.com/dexscan/polygon/0xe0dbb35750747b488f1ae81fa93c991ad82d582b/

[39] *Truffle: The most comprehensive suite of tools for smart contract development* [online]. In: . Truffle Suite [cit. 2023-05-17]. Dostupné z: https://trufflesuite.com/

[40] *Metamask - A crypto wallet & gateway to blockchain apps: Start exploring blockchain applications in seconds. Trusted by over 30 million users worldwide.* [online]. In: . Metamask [cit. 2023-05-13]. Dostupné z: https://metamask.io/

[41] *Let's build from here: Introducing GitHub Copilot X* [online]. In: . Github [cit. 2023-05-17]. Dostupné z: https://github.com/

[42] *Software. Faster.: GitLab is the most comprehensive AI-powered DevSecOps Platform.* [online]. In: . GitLab [cit. 2023-05-17]. Dostupné z: https://about.gitlab.com/

[43] AMAZON. *Amazon Web Services: Web Hosting* [online]. In: . [cit. 2023-05-10]. Dostupné z: https://aws.amazon.com/websites/

[44] STATISTA. *Cloud infrastructure services vendor market share worldwide from 4th quarter 2017 to 4th quarter 2022* [online]. In: . [cit. 2023-05-10]. Dostupné z: https://www.statista.com/statistics/967365/worldwide-cloud-infrastructure-services-market-share-vendor/

[45] *Amazon S3: Object storage built to retrieve any amount of data from anywhere* [online]. In: . [cit. 2023-05-10]. Dostupné z: https://aws.amazon.com/s3/

[46] *Amazon Route 53: A reliable and cost-effective way to route end users to Internet applications* [online]. In: . AWS [cit. 2023-05-11]. Dostupné z: https://aws.amazon.com/route53/

[47] *AWS Certificate Manager: Provision and manage SSL/TLS certificates with AWS services and connected resources* [online]. In: . AWS [cit. 2023-05-11]. Dostupné z: https://aws.amazon.com/certificate-manager/

[48] *Dashboard | Amazon EC2* [online]. In: . AWS [cit. 2023-05-23]. Dostupné z: https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Home:

[49] *What is Amazon EC2?: Secure and resizable compute capacity for virtually any workload* [online]. In: . AWS [cit. 2023-05-23]. Dostupné z: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

[50] NASSRI, Ahmad. *Npm Blog (Archive)* [online]. In: . NPM. Dostupné také z: https://blog.npmjs.org/post/615388323067854848/so-long-and-thanks-for-all-the-packages.html

[51] *DocuSign Standards-Based Signatures: Easy-to-use digital signatures* [online]. In: . Docu Sign [cit. 2023-05-24]. Dostupné z: https://www.docusign.com/products/digital-signature

[52] *Adobe Document Cloud* [online]. In: . Adobe [cit. 2023-05-24]. Dostupné z: https://www.adobe.com/documentcloud.html

[53] *Diplomachain: Důvěryhodný digitální otisk dokladu dosažené kvalifikace.* [online]. In: . PT LAB, ELA Blockchain Services, Gordic, NEXPRO, STYRAX, Vysoká škola logistiky [cit. 2023-05-24]. Dostupné z: https://www.diplomachain.cz/

[54] *IBM Blockchain: Achieve operational agility with trust* [online]. In: . IBM [cit. 2023-05-24]. Dostupné z: https://www.ibm.com/blockchain

[55] CHAI, Wesley a Kathleen CASEY. Software as a Service (SaaS). In: *Software as a Service (SaaS)* [online]. TechTarget [cit. 2023-04-09]. Dostupné z: https://www.techtarget.com/searchcloudcomputing/definition/Software-as-a-Service

[56] TON, Kimberly. *Tech Talk: What is the Virtual DOM?* [online]. In: . Fullstack Academy, 2017 [cit. 2023-05-10]. Dostupné z: https://www.youtube.com/watch?v=d7pyEDqBDeE

[57] WONG, Janelle. *Https://medium.com/@janelle.wg/atomic-design-pattern-how-to-structure-your-react-application-2bb4d9ca5f97: Design Patterns* [online]. In: . Dostupné také z: https://medium.com/@janelle.wg/atomic-design-pattern-how-to-structure-your-react-application-2bb4d9ca5f97

[58] HAPSE, Shubham. *Blockchain 101: The Simplest Guide You Will Ever Read* [online]. In: . Velotio [cit. 2023-05-12]. Dostupné z: https://www.velotio.com/engineering-blog/introduction-to-blockchain-and-how-bitcoin-works

[59] GOLDMANN, Martin. *App Vs. dApp* [online]. In: . Cryptopolitan. Dostupné také z: https://www.cryptopolitan.com/app-vs-dapp/

[60] Top 5 Most Expensive Bored Ape Yacht Club NFTs. In: *Top 5 Most Expensive Bored Ape Yacht Club NFTs* [online]. .: Bored Ape Yacht Club, 2022 [cit. 2022-11-28]. Dostupné z: https://blockgeeks.com/5-most-expensive-bored-ape-yacht-club-nfts/

[61] *MVC* [online]. In: . MDN web docs. Dostupné také z: https://developer.mozilla.org/en-US/docs/Glossary/MVC

[62] *Understanding digital signatures: What is a digital signature, and how can you create one?* [online]. In: . Docu Sign [cit. 2023-05-21]. Dostupné z: https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq

[63] *Google trends: Porovnání Reactu, Angularu a Vue.js* [online]. In: . Google [cit. 2023-05-23]. Dostupné z: https://trends.google.com/trends/explore?q=%2Fm%2F012l1vxv,%2Fg%2F11c6w0ddw9,%2Fg%2F11c0vmgx5d&hl=cs

[64] ELLIOT, Eric. *Top JavaScript Frameworks and Technology 2023: Front End Frameworks* [online]. In: . Javascript Scene [cit. 2023-05-23]. Dostupné z: https://jelvix.com/blog/best-javascript-frameworks

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ABI	Application Binary Interface
BSON	Binary JavaScript Object Notation
BTC	Bitcoin
CDN	Content Delivery Network
CI/CD	Continuous Integration/Continuous Deployment
CSS	Cascading Style Sheets
DAPP	Decentralized Application
DevOps	Development and (IT) Operations
DNS	Domain Name Systém
DOM	Document Object Model
ETH	Ether
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protokol
HTTPS	HyperText Transfer Protokol Secure
JSON	JavaScript Object Notation
NPM	Node Package Manager
NS	Name Server
PoS	Proof of Stake
PoW	Proof of Work
SOA	Start Of Authority
SPA	Single Page Aplikace
SSL/TLS	Secure Socket Layer/Transport Layer Security
URL	Uniform Resource Locator
WWW	World Wide Web

SEZNAM OBRÁZKŮ

Obrázek 1: Architektonický vzor MVC 18
Obrázek 2: Grafické znázornění příkladného DOM 19
Obrázek 3: DOM cyklus 20
Obrázek 4: Virtuual DOM 21
Obrázek 5: Atomic design 22
Obrázek 6: Řetězení bloků dat v blockchainu 28
Obrázek 7: Apps vs Dapps 31
Obrázek 8: NFT z kolekce Bored Ape Yacht Club 32
Obrázek 9: PoW vs PoS grafické znázornění 36
Obrázek 10: Digitální podpis 47
Obrázek 11: Amazon S3 – grafické znázornění funkcionality 50
Obrázek 12: Amazon Route 53 - grafické znázornění funkcionality 51
Obrázek 13: Amazon Cloudront – grafické znázornění funkcionality 52
Obrázek 14: Amazon Certificate Manager – grafické znázornění funkcionality 53
Obrázek 15: Porovnání dat vyhledávání Reactu, Angularu a Vue dle Google trends 56
Obrázek 16: Grafické porovnání frontendových frameworků dle oblíbenosti 57
Obrázek 17: Metamask – přidaná síť Mumbai 63
Obrázek 18: Přehled služeb aplikace a jejich spolupráce 64
Obrázek 19: Inicializace smart contractu pro FE 65
Obrázek 20: FE kód pro uložení nové struktury dokumentu do blockchainu 67
Obrázek 21: FE kód pro uložení metadat do stávající struktury dokumentu do blockchainu 68
Obrázek 22: FE kód pro přečtení záznamu z blockchainu 68
Obrázek 23: FE kód pro server spojení 69
Obrázek 24: FE kód pro připojení k MongoDB 69
Obrázek 25: FE kód pro inicializaci komunikace s endpointy backendu 70
Obrázek 26: Schéma modelu uživatele 71
Obrázek 27: Schéma modelu dokumentu 71
Obrázek 28: Ukázka zpracování požadavku uživatele 72
Obrázek 29: Formulář aplikace pro uložení dokumentu 76
Obrázek 30: Ukázka uloženého dokumentu v MongoDB 77
Obrázek 31: Ukázka ověření integrity v aplikaci 77
Obrázek 32: Ukázka změny názvu dokumentu v MongoDB 78
Obrázek 33: Ukázka ověření integrity v aplikaci po změně názvu souboru 78

Obrázek 34: Registrovaná doména pomocí Amazon Route 53 79
Obrázek 35: Potvrzení užití Cloudfront v DevTools 83
Obrázek 36: Spuštění serverové aplikace jako samostatný proces 85

SEZNAM TABULEK

Tabulka 1: Porovnání SQL a NOSQL 26
Tabulka 2: Porovnání výhod Web2.0 a Web3.0 29
Tabulka 3: Porovnání PoW a PoS 37
Tabulka 4: Porovnání ETH mainnet a Polygonu 40
Tabulka 5: Porovnání funkcí verzovacích platforem Github a Gitlab 45
Tabulka 6: Instalované NPM balíčky a jejich využití 60
Tabulka 7: Seznam všech endpointů aplikace 73
Tabulka 8: Seznam DNS záznamů pro hostování ukázkové aplikace 81

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH CD

PŘÍLOHA P I: OBSAH CD

Struktura obsahu přiloženého CD:

- **fulltext.pdf** - obsahuje text diplomové práce ve formátu PDF
- Komprimovaný soubor **prilohy.zip –** obsahuje všechny přílohy
    - Komprimovaný soubor **JanBures_app.zip –** obsahuje zdrojové kódy aplikace
    - Soubor **application_info.txt –** obsahuje data pro práci s aplikací