# ZADÁNÍ DIPLOMOVÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Wingender** Jméno: **Antonín** Osobní číslo: **466328**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Softwarové inženýrství**

## II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

**Webová Aplikace pro demonstraci volebního systému založeném na blockchainu**

Název diplomové práce anglicky:

**Web application to demonstrate a blockchain-based voting system**

Pokyny pro vypracování:

Diplomové práce bude mít za cíl prozkoumat koncept volebního systému založeném na technologii blockchain, specificky za použití Ethereum Smart Contracts. Diplomová práce by měla odpovědat na tyto otázky:
- Je volební systém založený na blockchainu bezpečný? Pokud ano, je bezpečnější než tradiční, centralizované volební systémy? Proč?
- Je takovýto volební systém využitelný z logistického hlediska? Je možné aby byl dostatečně jednoduchý pro průměrného voliče?
- Převažují výhody tohoto systému nad případnou cenou jeho implementace na celostátní úrovni?

Následně bude implementována jednoduchá verze tohoto webového systému v podobě webové aplikace. Tato webová aplikace bude zprostředkovávat možnost volby pro fiktivního politického kandidáta ve fiktivních politických volbách.
Výstup diplomové práce tedy bude obnášet:
- Text, který podrobně prodiskutuje témata, kterými se zabývají výše položené otázky a pokusí se na ně odpovědět.
- Rešerše dostupných technologií které souvisí s Ethereum Smart Contracts a porovnání mezi nimi
- Webové demo aplikace, která splňuje výše uvedené požadavky a adekvátně demonstruje závěry práce.

Seznam doporučené literatury:


- A systematic review of blockchain: https://jfin-swufe.springeropen.com/track/pdf/10.1186/s40854-019-0147-z.pdf
- Ethereum: A secure decentralised generalised transaction ledger: https://files.gitter.im/ethereum/yellowpaper/VIyt/Paper.pdf
- Towards secure e-voting using ethereum blockchain: https://ieeexplore.ieee.org/document/8355340


Jméno a pracoviště vedoucí(ho) diplomové práce:

**Ing. Jiří Šebek, kabinet výuky informatiky FEL**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **12.08.2021** Termín odevzdání diplomové práce: **04.01.2022**

Platnost zadání diplomové práce: **19.02.2023**

| Ing. Jiří Šebek<br>podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | prof. Mgr. Petr Páta, Ph.D.<br>podpis děkana(ky) |
|---|---|---|

## III. PŘEVZETÍ ZADÁNÍ

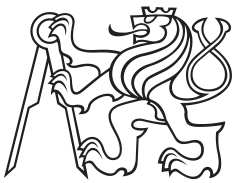
Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ _____

Datum převzetí zadání Podpis studenta

Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

# Webová Aplikace pro demonstraci volebního systému založeném na blockchainu

Antonín Wingender

Vedoucí: Ing. Jiří Šebek
Studijní program: Otevřená informatika
Srpen 2021

# Poděkování

Děkuji panu inženýru Jiřímu Šebkovi za vedení a odezvu k této diplomové práci.


# Prohlášení

Tuto práci jsem vytvořil sám, pouze za pomoci zdrojů uvedených v bibliografii, internetových zdrojů a znalostí získaných během svého studia.

V Praze, dne 4. ledna 2022

# Abstrakt

Online hlasování je trendem, který v moderní společnosti nabírá na síle. Má velký potenciál snížit organizační náklady a zvýšit volební účast. Odpadá nutnost tisknout hlasovací lístky nebo otevírat volební místnosti – voliči mohou hlasovat odkudkoli, kde je připojení k internetu. I přes tyto výhody, nevýhody online volebních systémů spočívají v bezpečnosti - jediný útočník s přístupem do databáze může změnit celé volby. Blockchainové technologie se díky své decentralizované povaze jeví jako potenciální řešení tohoto problému.

Cílem této diplomové je popsat stav moderních blockchainových technologií a současný stav voleb, ať už papírových či elektronických. Následně se tato práce zabývá návrhem a vytvořením volební aplikace za použití moderních blockchainových technologií. Vytvořená aplikace umožní uživateli vytvářet volby, přidat do voleb účastníky a těmto účastníkům umožní přihlásit se do uživatelského rozhraní a odvolit.

Tato práce také posoudí blockchainové technologie a výslednou aplikaci z pohledu bezpečnosti, přepoužitelnosti a popíše další důležité faktory, jako například škálovatelnost či zaručení anonymity voliče.



**Klíčová slova:** blockchain, volby, web aplikace

**Vedoucí:** Ing. Jiří Šebek
Technická 2, Fakulta elektrotechnická, Praha 6



# Abstract

Online voting is a growing trend in modern society. It has great potential to reduce organizational costs and increase turnout. There is no need to print ballots or open polling stations - voters can vote from anywhere with an internet connection. Compared to the advantages, disadvantages of online voting systems lie in security - a single attacker with access to the database can change entire choices. Due to their decentralized nature, blockchain technologies appear to be a potential solution to this problem.

The aim of this diploma thesis is to describe the state of modern blockchain technologies and the current state of elections, whether paper or electronic. Subsequently, this work deals with the design and creation of an election application using modern blockchain technologies. The created application will allow the user to create elections, add participants to the elections, and allow those participants to log in to the user interface and vote.

This work will also assess blockchain technologies and the resulting application in terms of security, usability and describe other important factors, such as scalability or guaranteeing the anonymity of the voter.



**Keywords:** blockchain, voting, web application

**Title translation:** Web application to demonstrate a blockchain-based voting system

# Obsah

# Obrázky

# Kapitola 1

# Úvod

Cílem této diplomové práce je vytvořit webovou aplikaci, která bude demonstrovat využitelný volební systém založený na blockchainové technologii. Součástí cíle bude provést podrobnou rešerši a analýzu takového systému z několika úhlů pohledu.

**Diplomová práce má následující strukturu:**

První kapitola je tento úvod.

V druhé kapitole se práce věnuje technologii blockchain. Tato kapitola shrne, co blockchain je, na jakém principu blockchain funguje a proč se dá využít pro volební systém. Bude zde uvedeno několik příkladů, k čemu se blockchain využívá dnes a jaké jsou obecné trendy ve vývoji této oblasti do budoucna.

V třetí kapitola bude popsáno, jaké požadavky by obecně měly volby splňovat. Následně popíšu současné způsoby voleb, jejich výhody a nevýhody, a posoudím je z hlediska bezpečnosti.

Čtvrtá kapitola se bude hlouběji zabývat využitím technologie blockchain pro volební systém. Popíše výhody a nevýhody této technologie a výzvy, které tento přístup skýtá. Podívá se na různé způsoby, kterými se řeší požadavky na kvalitu voleb, které byly popsány ve třetí kapitole.

Pátá kapitola se bude zabývat současnými návrhy a řešení volebních systémů, porovnám jejich výhody, nevýhody a společné vlastnosti. Budou popsány funkční a nefunkční požadavky aplikace.

V šesté kapitole bude definován návrh aplikace, podoba jednotlivých komponent a entit, které v aplikaci budou figurovat. Součástí této kapitoly bude architektura aplikace.

V sedmé kapitole budou popsány použité technologie a nástroje, o kterých bylo

rozhodnuto porovnáním mezi možnými alternativami a vysvětleno, přesně podle jakých kritérií byly zvoleny.

V osmé kapitole bude popsán průběh implementace aplikace, adresářová strukturu aplikace, způsob jakým byla vyvinuta a popsány, jaké funkce nabízí.

V deváté kapitole bude zdůvodnění, proč je aplikace dle mého názoru dobře využitelná jako přepoužitelná šablona pro vývoj dalších aplikací.

Desátá kapitola zanalyzuje aplikaci podle požadavků na volby z kapitoly 3. Popíše, které z nich aplikace splňuje a které jí chybí.

Jedenáctá kapitola popíše, jakým směrem by se mohl vývoj na aplikaci ubírat do budoucna.

Nakonec, dvanáctá kapitola shrne výstup diplomové práce.

# Kapitola 2

# Blockchain

## 2.1 O blockchainu

V obvyklém případě jsou databáze centralizované. Často bývají nasazené na výkonných serverech s výkonem stovek až tisíců normálních počítačů. Jsou optimalizované pro to, aby informace na nich byly získány a zapisovány velice rychle - v řádech milisekund. Provozovatelé výkonných databází pak potřebují dostatečný výkon na to, aby velká množství uživatelů mohly používat databázi najednou a dostatečně rychle.

Blockchain je možné popsat jako speciální druh databáze. Ve světě blockchainu je tato databáze nazývaná "ledger", tedy seznam. Od normální databáze se liší způsobem, jakým zapisuje data - zatímco tradiční databáze obvykle zapisují informace do oddělených tabulek s řádky a sloupci, blockchain je databáze složená z bloků, které jsou navzájem pevně propojené hashovacími algoritmy. [1]

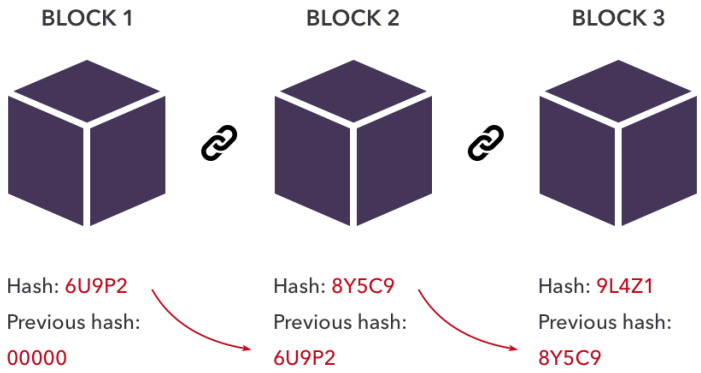
V následujících kapitolách budu používat pojmy **blockchainová databáze**, **seznam**, a **databáze** zaměnitelným způsobem. Všechny tyto pojmy znamenají řetězec bloků, na kterém jsou uložena nějaká data.

### 2.1.1 Datová struktura blockchainu

Blockchain ukládá informaci do bloků. Jeden blok se skládá ze samotných dat (ta mohou vypadat téměř jakkoli, ale obvykle mají tato data předem určenou strukturu), hashe který identifikuje blok, a hashe který identifikuje předchozí blok. Toto uspořádání vytváří řetězovou datovou strukturu (proto také "blockchain"- z angličtiny řetěz bloků), kde bloky jsou spolu pevně propojeny a je tak velice obtížné změnit stav takového řetězce. Tyto bloky

mohou uložit pouze omezené množství dat - pro víc dat se musí vytvořit víc bloků, které je uloží.

Jedna z nejdůležitějších vlastností blockchainu je, že je velice obtížné změnit jiný blok, než ten úplně poslední. Pokud změníme blok, tak jeho hash se změní taky. To znamená, že následující blok bude mít jiný, chybný hash, a celý řetěz se tímto přeruší. Pro změnu jednoho článku řetězu by se musel vypočítat nový hash pro každý následující článek řetězu. Toto bývá obvykle velice obtížně proveditelné v závislosti na způsobu, jakým konkrétní implementace blockchainu zpracovává nové bloky. Lépe vidíme tento koncept na obrázku 2.1.




**Obrázek 2.1:** Ilustační obrázek principu blockchainu.

## 2.2 Vlastnosti blockchainu

### 2.2.1 Decentralizace

Blockchain samotný nemusí být decentralizovaný, ale populární implementace blockchainu (např. Bitcoin, viz kapitola **??**) ho implementují decentralizovaným způsobem. Decentralizace se obvykle považuje za základní vlastnost blockchainových databází.

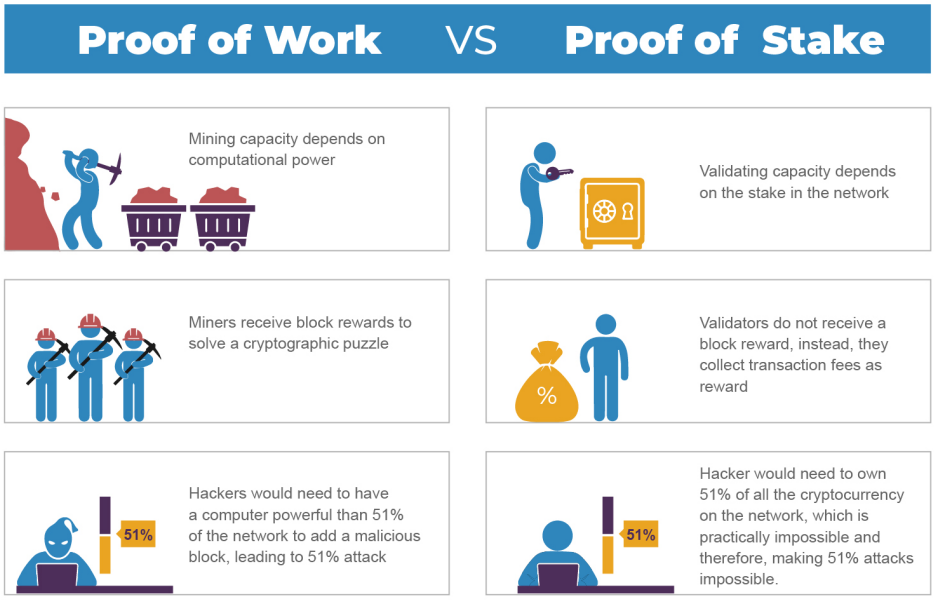
V praxi to probíhá tak, že místo centralizovaného místa, kam ukládat blockchainovou databázi, každý počítač, tedy "uzel", který databázi používá, má uloženou kopii celé databáze (např. v případě Bitcoinu všech transakcí, které v jeho síti kdy byly vykonány). Toto má výhodu v tom, že neexistuje jediný zranitelný bod, který by potenciální útočník mohl zneužít. Aby útočník úspěšně změnil blockchainovou databázi, musel by úspěšně změnit více jak 50% všech kopií databáze. Při desítkách tisíc, až milionech kopií, je tento typ útoku extrémně obtížný, ne-li nemožný. Tomuto typu útoku se říká "Útok 50%". [2]

### 2.2.2 Transparentnost

Pokud je blockchainová databáze decentralizovaná, je zároveň také transparentní - jakýkoli uživatel může kdykoli vidět všechna data, která byla v databázi zapsána. Tato data jsou obvykle anonymizovaná - identita každého účastníka je pouze identifikátor a tato identita je zabezpečena takzvaným SSH klíčem - párem veřejného a soukromého klíče, bez něhož identita uživatele nemůže být potvrzena. Soukromý klíč vlastní pouze koncový uživatel.

## 2.3 Algoritmus konsenzu

Algoritmus konsenzu (v angličtině Consensus Algorithm) je iniciován ve chvíli, kdy chceme vložit blok transakcí do blockchainové databáze. Tento algoritmus zprostředkovává souhlas jednotlivých uzlů v blockchainové síti, že tuto transakci je možné uložit na konec řetězce bloků. Nejznámější algoritmy konsenzu jsou tzv. Proof-of-Work, který se používá například u kryptoměny Bitcoin, a Proof-of-Stake, který se používá u kryptoměn, jako je například Ethereum, nebo u sítí které nemají přímou souvislost s umožňováním finančních transakcí, jako je například Hyperledger Fabric. Na obrázku 2.2 můžeme vidět porovnání mezi dvěma nejpopulárnějšími algoritmy konsenzu.




**Obrázek 2.2:** Ilustační obrázek porovnávající Proof-of-Stake a Proof-of-Work.

### 2.3.1 Proof-of-Work

V Proof-of-Work algoritmu konsenzu musí jedna strana prokázat ostatním stranám, že vynaložila dostatečné množství výpočetní práce. Toto množství musí být dostatečně velké a obtížné na provedení, ale zároveň zvládnutelné. Základním předpokladem je, že vynaložené množství výpočetní práce je takové, že by odradilo kohokoli od snahy obejít pravidla systému. Útočníka tento algoritmus může odradit tak, že by na změnu blockchainu potřeboval vynaložit více jak 50% energie která byla vynaložena na jeho vytvoření. Toto obvykle bývá extrémně obtížné, až nereálné.

Bitcoin nejvíce popularizoval tento algoritmus konsenzu. Spočívá v řešení obtížných kryptografických algoritmů, kde počítače musí hledat výsledky hashe určitých čísel. Jelikož je pro nalezení hashe potřeba zkoušet různé hashe jeden po druhém, získat úspěšný výsledek je výpočetně extrémně náročné.

### 2.3.2 Proof-of-Stake

Proof-of-Stake algoritmus je založen na tom, že bloky jsou validovány pomocí sítě předem určených validátorů. Validátoři se určují tak, že se zavážou určitou cennou hodnotou (například podílem v síti či finančním obnosem). O tuto cennou hodnotu může validátor kdykoli přijít pokud se prokáže, že jednal proti zájmům blockchainové sítě. Toto je motivace pro validátora, aby validoval transakce dle zájmu sítě. Útok na takovou síť by vyžadoval přesvědčení více jak 50% validátorů, aby jednali proti zájmu sítě a riskovali tak ztrátu cenné hodnoty, kterou se zavázali.

Tento algoritmus se obvykle používá ve firemních blockchainových sítích, jako například Hyperledger Fabric. Je také používán známou kryptoměnou Cardano[3] a Ethereum, jedna z nejpopulárnějších blockchainových platforem na světě, plánuje přejít na Proof-of-Stake v roce 2022. [4]

Oproti Proof-of-Work spotřebuje Proof-of-Stake mnohem méně energie (až 1% toho, co spotřebuje Proof-of-Work), protože u něj není potřeba řešit náročné výpočetní algoritmy.

## 2.4 Hashování

Hashování je algoritmus, kdy pro vstup variabilní délky získáme výstup fixní délky, zvaný hash. Důležitou vlastností takového výstupu je to, že každý vstup produkuje úplně jiný výstup, a že je velice obtížné získat z tohoto výstupu původní vstup. U moderních hashovacích funkcí je možné získat původní

hodnotu z hashe pouze náhodným zkoušením různých vstupů a porovnávání výstupů. Toto je příliš výpočetně náročné pro netriviálně velkou množinu možných vstupních hodnot. [5]

Hashování je pro blockchain základním pilířem celé technologie. Každý blok v blockchainu obsahuje hash svého předchůdce (tím se tvoří onen řetězec bloků). Hash bloku je spočítán z hashe předchozího bloku a serializovaného obsahu současného bloku. Vzhledem k vlastnosti hashování, že i pro triviálně změněný vstup vyprodukuje úplně jiný výstup, jakákoli změna v bloku ovlivní všechny bloky po něm.

### 2.4.1 Populární hashovací algoritmy

#### SHA-256

Hashovací algoritmus SHA-256, neboli také Secure Hashing Algorithm 256, je populární algoritmus používaný například u kryptoměny Bitcoin a mnohých odnoží této měny. Jak už jeho název napovídá, vytvoří 256-bitový řetězec pro jakýkoliv vstup.

Funkce z třídy hashovacích algoritmů SHA-2, do které SHA-256 patří, se používají u mnoha celosvětově používaných protokolů, jako například protokoly TLS, SSL, SSH, IPsec a další.

#### SHA-3

Algoritmus SHA-3, zvaný také ETHASH, je hashovací algoritmus vytvořený pro použití v síti Ethereum. Je postavený na rodině kryptografických algoritmů zvaných Keccak. Je neoficiálním následovníkem rodiny SHA-2.

Algoritmy z rodiny SHA-3 demonstrovaly obecně lepší odolnost proti útokům oproti algoritmům z rodiny SHA-2. [6]

#### scrypt

scrypt byl publikován v roce 2016 a byl vytvořen jako odpověď pro vysokoškálové útoky proti šifrám, protože útok hrubou silou (tzv. "brute force attack") vyžaduje pro tento algoritmus velké množství paměti, tudíž je nerealistické že by ho útočník mohl prolomit. Používá se například v kryptoměnách **Litecoin** a **Dogecoin**.

## 2.5 Smart contracts

**„Smart contract“** je termín používaný k popisu počítačového kódu, který automaticky provádí dohodu mezi dvěma stranami a je uložen na platformě založené na blockchainu. [7] Kód může být buď jedinou dohodou mezi dvěma stranami, nebo může doplňovat tradiční papírovou smlouvu a vytvářet určitá ustanovení, jako je například převod finančních prostředků ze strany A na stranu B. Samotný kód je replikován napříč několika (obvykle všemi) uzly blockchainu, a proto, stejně jako standardní data uložené v blockchainové databázi, je bezpečný, stálý a neměnný. Tato replikace také znamená, že je smart contract vždy vyhodnocen, pokud je nový blok přidán do blockchainové databáze. Strany mohou uvést potřebné parametry pro spuštění smart contractu, a kód pak s těmito parametry bude pracovat jako např. standardní funkce v programovacích jazycích. Pokud žádná taková transakce nebyla zahájena, kód se nespustí. Většina smart contractů je napsána v jednom z programovacích jazyků přímo vytvořených pro psaní smart contractů, jako je například Solidity. Obrázek 2.3 ilustruje podobu kódu Solidity.

```
contract token {
    mapping (address => uint) public coinBalanceOf;
    event CoinTransfer(address sender, address receiver, uint amount);

  /* Initializes contract with initial supply tokens to the creator of the contract */
  function token(uint supply) {
       if (supply == 0) supply = 10000;
       coinBalanceOf[msg.sender] = supply;
    }

  /* Very simple trade function */
    function sendCoin(address receiver, uint amount) returns(bool sufficient) {
        if (coinBalanceOf[msg.sender] < amount) return false;
        coinBalanceOf[msg.sender] -= amount;
        coinBalanceOf[receiver] += amount;
        CoinTransfer(msg.sender, receiver, amount);
        return true;
    }
}
```

**Obrázek 2.3:** Ilustrační jednoduchý Smart Contract napsaný v jazyce Solidity.

V následujících bodech popíšu několik příkladů, jak jsou smart contracts reprezentovány u nejpopulárnějších platforem:

- **Bitcoin:** Hlavní případ použití Bitcoinu není vytváření smart contracts, ale jako médium výměny (tedy jako měna). Je ale stále možné na Bitcoinové síti smart contracty vytvářet pomocí jazyku Script. Jedním z příkladů je například FastKitten, který usnadňuje vytváření smart contractů na Bitcoinové síti. [8]
- **Ethereum:** Pro Ethereum jsou smart contracty hlavní případ užití této blockchainové sítě. Ethereum má vlastní programovací jazyk pro své smart contracty, zvaný Solidity. U Etherea je pro spouštění smart contractů na veřejné síti potřeba platit kryptoměnou zvanou Ether.

- **Hyperledger Fabric:** Na platformě Hyperledger Fabric se používají smart contracty v kompilovaném formátu zvaném ChainCode. Pro samotný vývoj smart contractů je možné používat známe programovací jazyky, jako Node.js, Java nebo Go. [9]

## 2.6 Známé kryptoměnové platformy

Kryptoměnové platformy jsou v současnosti nejpopulárnější způsob využití blockchainových databází na světě. Tyto platformy se používají pro zprostředkovávání důvěrných transakcí mezi uživateli. Např. v případě Bitcoinu jako zprostředkovávání finančních transakcí mezi lidmi. V případě Etherea to mohou být jakékoli důvěrné transakce, které je možné popsat v programovacím jazyce, jako například smlouvy.

Kryptoměnové platformy jsou již z jejich záměru veřejně přístupné. To znamená, že kdokoli v nich může vyhledat jakoukoli transakci a předložit novou transakci, kterou musí následně celá síť schválit. Jelikož jsou veřejně přístupné, musí používat algoritmy konsenzu založené na pravděpodobnosti, jako např. Proof-of-Work (viz kap. 2.3.1).

### 2.6.1 Bitcoin

Bitcoin je nejpopulárnější implementace blockchainové technologie na světě. Koncept Bitcoinu byl v roce 2009 předložen neznámým člověkem používající pseudonym Satoshi Nakamoto v jeho práci *Bitcoin: A Peer-to-Peer Electronic Cash System* [1]. Představil ho jako "čistě peer-to-peer systém elektronických peněz který umožňuje posílat peníze od jednoho člověka k druhému bez nutnosti důvěryhodné třetí strany". Zároveň v této práci prezentuje Proof-of-Work algoritmus konsenzu (viz kap. 2.3.1). Praví, že díky nutnosti poskytnout co největší množství energie je možné předpokládat, že řetězec, do kterého bylo vloženo největší množství výpočetního výkonu, je ten správný a že potencionální útočník na systém by byl tímto požadavkem demotivován.

Bitcoin je v současnosti největší kryptoměnová platforma na světě. V době psaní tohoto textu je valuace všech bitcoinů na světě kolem 960 miliard dolarů, víc než dvojnásobek Etherea, které představuje druhou největší kryptoměnu na světě. [10] Vzhledem k jeho rychle rostoucí ceně se z něj stala populární investiční komodita. Stinnou stránkou této rychle rostoucí popularity je mimo jiné energetická spotřeba Bitcoinu - díky tomu, že používá Proof-of-Work (kap. 2.3.1) jako svůj algoritmus konsenzu, vytváří mnoho lidí velké "těžební"operace, které neustále hledají nové hashe, kterými by mohli vydělat

[1] `https://bitcoin.org/bitcoin.pdf`

na odměně za vytvoření nového bloku. Tyto těžební operace spotřebovávají značné množství energie. V únoru 2021 již Bitcoin využíval pro svou těžbu "víc energie než celá země Argentina"[11].

### 2.6.2 Ethereum

Ethereum je druhá největší blockchainová platforma na světě. Její součástí je i její vlastní kryptoměna Ether, a vlastní programovací jazyk pro smart contracty, Solidity.

Největší výhodou Etherea je schopnost publikovat vlastní kód na blockchainové síti. Tento kód je ověřen všemi vlastníky sítě, kteří poskytnou svou výpočetní sílu výměnou za částku v kryptoměně Ether. Následně, vždy když někdo chce spustit tento kód, jeho výsledek je opět ověřen všemi uzly v síti. Toto zaručuje, že výsledek je plně transparentní, ale také to znamená horší škálovatelnost a zvyšující se cenu za každé spuštění, která fluktuuje podle fluktuace ceny Etheru.

Algoritmus konsenzu platformy Ethereum je v době psaní této práce Proof-of-Work, který využívá hashovací algoritmus zvaný ıscrypt (kap. 2.4.1). Dlouhodobým plánem je však přejít na tzv. Proof-of-Stake algoritmus (kap. 2.3.2). To dle tvůrců platformy zabezpečí lepší škálovatelnost systému do budoucna, a značně sníží spotřebu energie celého systému.

## 2.7 Známé business blockchainové platformy

Business blockchainové platformy se od kryptoměnových platforem odlišují především svým primárním využitím. Zatímco kryptoměnové platformy se používají především pro zprostředkovávání transakcí mezi jakýmikoli uživateli, business blockchainové platformy se zabývají zprostředkováváním vysledovatelných transakcí v rámci organizací. Často se využívají pro nastavení pravidel jednoznačné komunikace mezi odlišnými organizacemi, ale mohou být používány i mezi organizacemi ve vnitřní struktuře jedné firmy.

Jejich specializace je soukromí. Tyto platformy obvykle mají způsoby, jak zabezpečit přístup k blockchainové databázi (narozdíl od kryptoměn, kde kdokoli má přístup k celé databázi). To také znamená, že nemusí používat algoritmy konsenzu založené na pravděpodobnosti, jako třeba Bitcoin, a nejsou díky tomu náchylné k útoku 50%. Také to znamená, že obvyklá rychlost zpracování transakcí na této síti je obvykle mnohem vyšší než u veřejných kryptoměnových sítí, především díky obvykle nižšímu počtu uzlů, jejichž souhlas je nutný pro vykonání transakce.

### 2.7.1 Hyperledger Fabric

Hyperledger je open-source modulární blockchainová platforma vytvořena společností IBM a mnoha jinými přispěvateli. [12] Hyperledger je zamýšlen jako základ, nad kterým se dají vystavět blockchainové produkty. Skládá se z různých "modulů", které jsou každý vytvořen za jiným účelem použití ve firemním prostředí. Jedním z těchto modulů je právě Hyperledger Fabric, který je zamýšlen jako "framework poskytující základ pro vytváření aplikací s modulární architekturou".

Hlavní výhoda pro firemní prostředí je, že Hyperledger Fabric není jedna globální blockchainová síť, a tedy je možné vytvořit soukromé sítě s vlastním systémem ověřování uživatelů. Má také vlastní systém tzv. Smart Contracts (podobně jako Ethereum), které běží v programovacím jazyku zvaném Chain code.

V roce 2020 vyšla verze 2.0, která má za úkol mimo jiné poskytnout lepší škálovatelnost a snížit výpočetní cenu transakcí.

### 2.7.2 Corda

Corda je open-source blockchainová platforma, která zjednodušuje správu právních smluv a dalších sdílených informací mezi vzájemně si důvěřujícími stranami. Corda byla původně navržena pro aplikace ve finančních institucích. Existuje i její komerční edice – Corda Enterprise, která je naopak používaná pro správu podniků. Corda Enterprise rozšiřuje edici Corda o další funkce, včetně podpory komerčních databází (Oracle, MSSQL), HSM, vylepšuje výkon (podporuje paralelizaci), umožňuje nastavení uzlů s vysokou dostupností a vylepšuje nástroje pro nasazení sítě. [13]

# Kapitola 3

# Způsoby voleb současnosti

Tato kapitola popíše základní požadavky na kvalitní volební systém. Těmito požadavky se budu řídit při posuzování současných volebních systémů i demonstrační aplikace, která je tématem této diplomové práce. Následně popíše současné volební metody, od papírových voleb po elektronické volby.

## 3.1 Základní požadavky pro dobrý volební systém

Dle mého názoru by volební systém měl volební systém splňovat tato kritéria:

- **Soukromí** - Každý hlas by měl být anonymní a nevysledovatelný.
- **Průkaznost** - Každý hlas může pocházet pouze od právoplatných voličů. Nikdo jiný by neměl být schopný volit.
- **Nepřepoužitelnost** - Každý volič by měl moci hlasovat pouze jednou, a duplicitní hlasy by se neměly počítat.
- **Férovost** - Nikdo by neměl mít moc získat výsledky kde nebyly započítány všechny hlasy a touto znalostí volby ovlivnit
- **Celkovost** - Pro právoplatný výsledek, všechny hlasy by měly být započítány.
- **Škálovatelnost** - Volební systém by mělo být možné použít i u voleb čítající miliony voličů.

## 3.2 Papírové volby

Volební systémy, převážně ty používané v politické sféře, musí být důvěryhodné často pro miliony voličů. Současný způsob volení, používané u vysoko profilových voleb jako například volby hlavy státu či volby do poslanecké sněmovny, je prováděn skrze papírové hlasy. Obvykle takové volby probíhají tak, že se člověk přihlásí u volební stanice, což zabezpečí, že je právoplatným voličem (tedy vlastnost **Průkaznost**). Následně vhodí svůj anonymní hlas, do volební schránky. Tento hlas poté již nelze spojit s voličem, který ho vhodil, protože je identický jako ostatní hlasy pro stejného kandidáta. Toto zabezpečí anonymizaci hlasu (tedy vlastnost **Soukromí**).

### 3.2.1 Výhody papírových voleb

Papírové volby mají oproti elektronickým systémům značnou výhodu v jejich robustnosti a odolnosti vůči potencionálním útočníkům. Jsou decentralizované, což v praxi vypadá tak, že každá volební oblast počítá své vlastní hlasy, které se nakonec na konci sečtou.

Díky jejich fyzické podstatě je prakticky nemožné změnit na dostatečně veliké škále výsledky voleb. Pokud by útočník chtěl změnit počty hlasů, musel by se fyzicky dostat do volebních místností a vyměnit stejný počet hlasů za hlasy pro jiného kandidáta. Provést takovýto útok by vyžadovalo koordinovanou akci pro stovky, až tisíce volebních okrsků v rámci několika dnů, kdy je obvykle možné volit. Úspěšný útok by také nesměl být vysledovatelný, jinak se volby mohou opakovat. Jak je vidět, narušit tímto způsobem integritu voleb je extrémně obtížné, a kromě velice těsných volebních výsledků je mizivá šance, že by takovýto útok mohl opravdu ovlivnit výsledek voleb.

Další výhodou je také "zavedenost"tohoto volebního systému. Lidé obvykle tento způsob volby znají a věří mu, a neexistuje tedy nedůvěra kvůli novotě systému nebo jeho "neodzkoušenosti".

### 3.2.2 Nevýhody papírových voleb

Tradiční papírové metody ale mají problémy, díky kterým lidé hledají efektivnější řešení. Tyto problémy jsou:

- Zorganizovat hlasování papírovými hlasy je náročné. Při větším množství voličů se musí v dojezdové vzdálenosti každého voliče postavit místo, kde může svůj hlas vhodit a kde může být zkontrolována jeho identita. Stavba a provoz těchto míst vyžaduje nemalé lidské i finanční náklady.

- Hlasy se musí fyzicky spočítat jeden po druhém, což může v některých případech trvat i dny.
- Účast ve volbách vyžaduje fyzický transport na místo voleb, což může některé lidi odradit od hlasování, což snižuje důvěru ve správnost výsledku voleb.
- Volič nemá způsob, jak zjistit, zda se jeho hlas opravdu spočítal.

## 3.3 Elektronické volby bez použití blockchainové technologie

Elektronické volby jsou volby skrze elektronické zařízení, obvykle počítač nebo mobilní zařízení. Volič se v nich obvykle identifikuje centralizovaným identifikačním systémem, jako například občanský průkaz.

V dnešní době je na centralizované elektronické volby pohlíženo s nedůvěrou, a jsou používany převážně pro méně důležitá rozhodnutí či při relativně malém počtu účastníků. Existují ovšem výjimky. Ve velkoformátových volbách, tedy například národních nebo krajských volbách jsou online volby nyní součástí voleb například v **Estonsku** (viz kapitola 5.1) a **Švýcarsku**.

Systémy elektronických voleb prováděné přes centrálního poskytovatele stojí a padají na onom poskytovateli. Audit poskytovatele těchto služeb je obvykle vyžadován od státních institucí. Různí poskytovatelé volebních systémů se snaží minimalizovat všechny bezpečnostní problémy které elektronické systémy mohou mít, jako:

- Systém s centralizovanou databází je vystaven elektronickým či dokonce fyzickým útokům.
- Autorita která poskytuje centralizovaný systém hlasování má plnou kontrolu nad jeho databází, a tedy je možná manipulace s hlasy.

### 3.3.1 Výhody elektronických voleb

Elektronické volby mají výhody oproti papírovým volbám. Tyto výhody vyplývají z povahy tohoto způsobů voleb. Zejména:

- Výrazně menší náklady na organizaci voleb. Není nutné vytvářet fyzická místa, kam musí voliči chodit, a není nutné hledat lidi, co budou tato místa provozovat.

- Počítání hlasů je instantní. Výsledek může být oznámen do několika minut po skončení hlasování.
- Snadnost voleb by teoreticky mohla znamenat, že voliči budou volit ve větším množství, což zvýší věrohodnost výsledků voleb. Existují však studie, které tuto teorii zpochybňují.

### 3.3.2 Nevýhody elektronických voleb

V roce 2000 byl povolán bezpečnostní audit na internetové volby v Arizoně. Výsledkem bylo identifikováno několik bezpečnostních hrozeb pro volby, jako denial-of-service útoky, krádež uživatelských dat a napadání zranitelných volebních počítačů.

Hlavní nevýhody elektronických voleb spočívají především v obavách o jejich zabezpečení. V bodech:

- Potencionální úspěšný útočník může jednoduše ovlivnit výsledek voleb změnou např. čísel v databázi. Tato změna by pravděpodobně byla nevystopovatelná, vzhledem k anonymní povaze hlasů.
- Případné chyby v systému způsobené ve vývoji, tzv. "bugs", mohou mít dalekosáhlý dopad na výsledek voleb.
- Někteří lidé nechtějí nebo nemohou volit elektronicky. Mohou to být například starší lidé kteří nevlastní nebo ani nechtějí vlastnit elektronická zařízení. Toto znamená, že v dohledné době by se tyto volby musely provádět jako doplněk stávajících způsobů. Tak to například probíhá dnes v Estonsku.

# Kapitola 4

# Využití technologie blockchain k vytvoření volebního systému

V této kapitole uvedu, jaké specifické vlastnosti by blockchainový volební systém měl splňovat, a jaké jsou jeho největší výhody v kontextu voleb. Následně vezmu některé požadavky z kapitoly 3 a u každého popíšu, jakým způsobem by je blockchainový volební systém mohl zaručit.

## 4.1 Důvod k využití blockchainu pro volební systém

Rád bych nejdříve vyjmenoval nejdůležitější prvky blockchainové architektury, které nám mohou pomoci při vytváření volebního systému založeném na této technologii.

- **Šifrování** - Transakce na blockchainové architektuře je bezpečně validována kryptografickými důkazy o identitě mezi zúčastněnými stranami.
- **Nezměnitelnost** - Transakce v blockchainové databázi nemohou být, kromě extrémních situací, mazány ani upravovány.
- **Ověřitelnost** - Jelikož mají všichni účastníci systému přístup do seznamu transakcí, můžeme vysledovat transakce, které s uživatelem souvisí.
- **Decentralizace** - Kopie databáze je uložena na všech zúčastněných uzlech. Způsob, jak se uzly shodnou na nových transakcích, se nazývá algoritmus konsenzu (viz 2.3).
- **Anonymita** - Jelikož transakce jsou prováděny identifikačním číslem peněženky (tzv. Wallet), identita uživatele zůstává skrytá ostatním.

- **Odolnost vůči útoku** - Útok na blockchainovou databázi je z důvodu její decentralizované povahy velice obtížný a od útočníka vyžaduje velké zdroje, a ve většině případů je neproveditelný.

U volebních systémů je bezpečnost jedna z nejdůležitějších vlastností systému. Dle mého názoru je právě obava z bezpečnosti systému hlavním důvodem, proč elektronické volby ještě nejsou skutečností ve většině zemí na světě, včetně zemí s bohatými zkušenostmi s vývojem softwaru.

## 4.2 Požadavky na volby z pohledu blockchainových technologií

V následujících podkapitolách pojednám o bodech z kap. 3.1. U každé této vlastnosti vyzdvihnu, jak by bylo možné jí řešit z pohledu blockchainového volebního systému.

### 4.2.1 Soukromí

Pro volební systém je důležité, aby hlas člověka byl striktně anonymní. Nikdo kromě samotného voliče nesmí vědět, jak člověk volil. Soukromí je dle mého názoru nejobtížnější dosáhnout v rámci elektronické volební aplikace. Je obtížné spojit soukromí a nepřepoužitelnost - pokud víme jaké identifikační číslo volilo, nemusí být hlas úplně anonymní. Z mé rešerše jsem v literatuře našel tři zajímavé způsoby, jak tento problém řešit - tzv. slepé podpisy, homomorfické šifrování a tzv. mix-networks.

#### Slepé podpisy

Slepý podpis (v angličtině Blind signature) je podepisování zprávy, kde zpráva je nejprve zašifrována předtím, než je podepsána. Díky tomu se ten, kdo zprávu podepisuje, nemůže dozvědět obsah zprávy. Pro ověření identity zprávy se používá tzv. Zero-knowledge proof, což je způsob, jak ověřit průkaznost uživatele aniž by byla odhalena jeho identita. [14]

V roce 1992 byl v práci od Fujioka[15] navržen následující systém: Právoplatný volič pošle jeho hlas validátorovi. Validátor ověří, že volič je právoplatný, podepíše již zašifrovaný hlas, a dá ho voliči zpět. Volič následně získá podpis pro nezašifrovaný hlas a pošle ho k zapsání, kde se validuje podpis validátora a přijme hlas.

V pozdějších letech byly navrženy vylepšené modely které zlepšily odolnost vůči vynucení voliče třetí stranou (tzv. coercion), nebo také například přidaly možnost změnit svůj hlas. Problémem tohoto způsobu je vliv toho, kdo podepisuje - musí existovat seznam vytvořených podpisů, který musí logicky korespondovat se seznamem právoplatných voličů. Pro anonymizaci by se měly hlasy i podpisy posílat přes zabezpečený kanál.

#### Homomorfické šifrování

Homomorfické šifrování je způsob šifrování, který dovolí vykonávat matematické operace nad zašifrovanými daty bez toho, aniž bychom je museli dešifrovat. Například, pokud máme funkci $c()$, která nám zašifruje zprávu, pak $c(a) + c(b) = c(a + b)$. [16]

V kontextu volebního systému nám homomorfické šifrování může zabezpečit, abychom nemuseli dešifrovat jednotlivé hlasy po konci volby - můžeme jen sečíst hlasy a následný výsledek dešifrovat. Dále, pro zabezpečení toho aby žádná jedna strana nemohla mít dopředu přístup k výsledkům voleb, můžeme dát část dešifrovacího klíče dostatečnému počtu stran (ale není nutné ho rozdělit rovnoměrně mezi všechny strany, jinak by jeden uzel odmítl kooperovat a nemohli bychom získat výsledek voleb).

#### Mix-networks

Mix-networks byly vynalezeny Davidem Chumem v roce 1980. Idea je taková, že informace prochází několika servery, kde se zamíchá a zašifruje znovu. Pokud alespoň jeden mix-server skutečně zamíchá informacemi, není možné vystopovat odkud pocházejí. [17]

Mix-networks bohužel také vytvářejí problém se škálovatelností - hlas musí projít více servery, a to by u blockchainových aplikací mohlo prezentovat problémy při vysokém počtu uzlů.

### 4.2.2 Průkaznost

Průkaznost systému je pro každou volbu odlišná, a tudíž jí nepovažuji za něco, co bych musel řešit v rámci této diplomové práce. Například ve volbách do poslanecké sněmovny v České republice by musel systém zabezpečit, že vytvoří pouze ty uživatele, kteří jsou legitimními voliči. Toto je možné zajistit například přes unikátní číslo občanského průkazu, ale ne všichni, co vlastní občanský průkaz, jsou právoplatnými voliči.

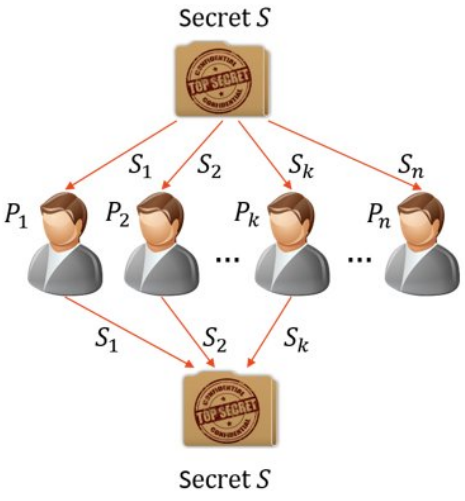
### 4.2.3 Férovost

Férovost je možné zabezpečit tak, že všechny hlasy jsou zašifrované, a rozšifrovány pouze na konci voleb. Důležité je, aby neměla jen jedna strana dešifrovací klíč, protože pak je možné aby tato strana, pokud má přístup k mezivýsledkům, rozšifrovala mezivýsledky a ovlivnila tak průběh voleb.

Rozdělit takto dešifrovací klíče jde pomocí tzv. Secret Sharing Scheme. [18]

#### Secret Sharing Scheme

Secret sharing scheme je metoda, která bezpečně rozdá části tajemství mezi skupinu účastníků. Volební systémy mohou použít tzv. Shamir Secret Sharing Scheme (SSSS), která rozdá tajemství pomocí polynomiální interpolace. U tohoto způsobu stačí, aby se pouze určitá skupina účastníků účastnila dešifrování. Bohužel, SSSS stále spoléhá na nezúčastněnou třetí stranu která rozdá části klíče mezi účastníky. Také nefunguje, pokud je ve skupině účastníků dostatečně velká skupina, která se odmítne účastnit dešifrování. [19]




**Obrázek 4.1:** Shamir Secret Sharing Scheme

### 4.2.4 Škálovatelnost

Současně existující volební systémy založené na blockchainových technologiích se nepoužívají ve volbách s velkým počtech voličů. Toto plyne z určitých problémů se škálovatelností, které v současných blockchainových platformách existují. Vzhledem k tomu, že jedním z principem blockchainu je nutnost

toho, aby se všechny uzly shodly na všech transakcích, se zvyšujícím se počtem uzlům se zhoršuje škálovatelnost systému. Popsáno společností Deloitte: *"systémy založené na blockchainu jsou pomalé v porovnání s jinými řešeními. Pomalá rychlost transakcí na blockchainu je proč se velké společnosti obávají migrovat na ně své stávající systémy."* [20]

V následující tabulce bych rád porovnal škálovatelnost známých blockchainových platforem.

| Porovnání škálovatelnosti blockchainových platforem | | | | |
|---|---|---|---|---|
| Název | Hash rate | Transakce/sek | Kryptografický algoritmus | Škálovatelnost |
| Hyperledger Fabric | - | 3500 | ECC | Dobrá |
| Ethereum | 168.59 Th/s | 15 | ECDSA | Špatná |
| Bitcoin | 900 Th/s | 7 | ECDSA | Velmi špatná |
| Ripple | - | 1500 | RPCA | Dobrá |
| Dogecoin | 1.4 Th/s | 33 | scrypt | Špatná |

**Tabulka 4.1:** Porovnání škálovatelnosti známých blockchainových platforem.

Dle porovnání transakcí za sekundu je vidět, že Hyperledger Fabric je v současnosti nejrychlejší blockchainová platforma. Výhodou Hyperledger Fabric je, že má omezení přístupu, tedy není možné aby se kdokoli připojil k jeho síti. Bohužel však ani Hyperledger Fabric nestačí dosáhnout počtu transakcí za sekundu jako má například společnost Visa, jejíž oficiální čísla se pohybují kolem 24 000 transakcí za sekundu.

# Kapitola 5

# Analýza současných řešení a požadavků

Tato kapitola analyzuje stávající řešení online hlasování, ať už jsou založená na blockchainových technologiích nebo ne. Následující podkapitoly pojednávají o jednotlivých firmách či útvarech, které elektronické volby zprostředkovávají.

## 5.1 Online volby v Estonsku

Estonsko začalo zvažovat elektronické hlasování přibližně v roce 2001, kdy progresivní koalice stavící se pozitivně k novým technologiím sestavila v Estonsku vládu. V roce 2007 byly online volby v Estonsku poprvé převedeny do reality. Přibližně 44% estonské populace volilo přes internet v roce 2019. [21]

Celý systém spoléhá na estonský občanský průkaz, který zároveň slouží jako čip pro bezpečnou autentifikaci u estonských úřadů. Hlasování po internetu je dovoleno před volbami, obvykle šestý až čtvrtý den před začátkem voleb. Volby jsou prováděny skrze webové rozhraní. Voličův hlas může být změněn neomezeně krát v tomto období, a voličova volba která je platná koncem tohoto období, se počítá.

### 5.1.1 Bezpečnost online voleb v Estonsku

Bezpečnost estonských voleb byla vystavena mnohé kritice od doby jejich zavedení. Serverový kód volebního systému byl v roce 2013 zpřístupněn jako open-source po tlaku široké veřejnosti.

V roce 2014, tým nadnárodních bezpečnostních systémů vydal vědeckou práci která systém odsoudila, a tvrdila že útočník může proniknout do systému,

měnit hlasy a počty hlasů, a zahladit po sobě jakékoliv stopy. Tým doporučil Estonsku aby online hlasování zadrželi. Estonská národní volební rada se proti tvrzení ohradila, s tvrzením že tyto útoky nejsou možné v reálném prostředí a nebo že je s nimi počítáno v samotném návrhu systému. [22]

## 5.2 Polyas

Polyas je společnost založena ve Finsku v roce 1996. Společnost využívá blockchainové technologie, aby poskytla veřejnému a soukromému sektoru elektronický hlasovací systém. Společnost Polyas byla v roce 2016 akreditována německým Spolkovým úřadem pro informační bezpečnost jako dostatečně bezpečná pro aplikace elektronického hlasování. Mnoho významných společností v celém Německu používá Polyas k provádění systémů elektronického hlasování. Polyas má nyní zákazníky po celých Spojených státech a Evropě.

## 5.3 Helios

Helios je online volební systém, který slibuje plně ověřitelný volební proces. Volební proces v systému Helios probíhá tak, že tvůrce voleb zadá seznam emailů a čas, kdy volby začnou. Volič použije svoje údaje získané v emailu, aby volil přes webové rozhraní. Toto je možné ověřit po konci voleb pomocí svého hesla.

### 5.3.1 Bezpečnost systému Helios

Administrátor voleb nemá žádný přístup k heslům a nemá žádný způsob, jak volby zkazit, jelikož každý hlas by měl být plně ověřitelný. Ale, jak už je případem u centralizovaných způsobů online voleb, útok na samotný server Helios je v tomto případě možný. Helios samotný je tedy jediný, kdo může zabezpečit volby proti hackingu. [23]

Protože se autentizujeme pouze emailem, je možné aby někdo např. vyhrožoval uživateli a donutil ho volit pro někoho, pro koho on sám nechce volit.

## 5.4 Voatz

Voatz je mobilní aplikace, která využívá blockchain pro svoji databázovou technologii. Autentifikace je prováděna přes moderní autentifikační metody

na mobilních zařízení, jako je například skener otisku prstů nebo sken obličeje. Voatz dle jejich vlastního tvrzení zaručuje, že tyto autentifikační metody jsou striktně anonymizované.

Volby probíhají tak, že Voatz dostane od organizátora voleb databázi všech uživatelů, kteří se budou účastnit voleb. Před tím, než uživatel odvolí, musí se autentifikovat naskenováním občanského průkazu a otisku prstů. Aplikace potom zvaliduje průkaz voliče [24]. Následně může volič odvolit.

### 5.4.1 Bezpečnost systému Voatz

Voatz tiskne papírové hlasy, ale tyto hlasy jsou potom ověřeny přes jejich server, aby mohl každý volič volit jen jednou. To v teorii zabezpečuje systém proti duplicitnímu hlasování. Samozřejmě, Voatz pouze tvrdí, že nemá přístup k hlasovacím datům. Jelikož Voatz získá od poskytovatele voleb databázi všech hlasujících, může si s ní dělat co chce. Bohužel se spoléháme tedy opět na jednu stranu, která může a nemusí být důvěryhodná.

## 5.5 Analýza požadavků

V následující kapitole budou shrnuty požadavky pro aplikaci, která je tématem této práce. Nejprve budou popsány funkční a následně nefunkční požadavky. Tyto požadavky vycházejí z textu v předchozích kapitolách.

### 5.5.1 Funkční požadavky

Jako hlavní funkční požadavky pro demo aplikaci bych identifikoval tyto:

1. **Možnost vytvořit volby:** Požaduji, abych jako administrátor mohl vytvořit volby pro určité časové období a s určitým počtem subjektů.
2. **Přihlášení se do systému:** Požaduji, abych se jako volič mohl úspěšně přihlásit do aplikace pomocí identifikačního kódu, který mi bude zaslán.
3. **Udělení hlasu:** Požaduji, abych jako volič mohl udělit jeden hlas jednomu kandidátovi v rámci jedněch voleb.
4. **Umožnění kontroly hlasu:** Požaduji, abych jako volič mohl zkontrolovat svůj udělený volební lístek a získat o něm informace.
5. **Možnost vidět všechny hlasy:** Požaduji, abych jako uživatel, administrátor nebo subjekt mohl vidět všechny transakce v databází, které se týkají dokončené volby, které jsem součástí.

6. **Možnost přidat voliče do volby:** Požaduji, abych jako administrátor přidat uživatele do mnou vytvořené volby. Tomuto uživateli bude přidělen kód unikátní pro danou volbu.

7. **Spočítání hlasů:** Požaduji, aby systém automaticky po skončení volebního období spočítal hlasy všech subjektů a umožnil tento výsledek vidět všem zúčastněným stranám.

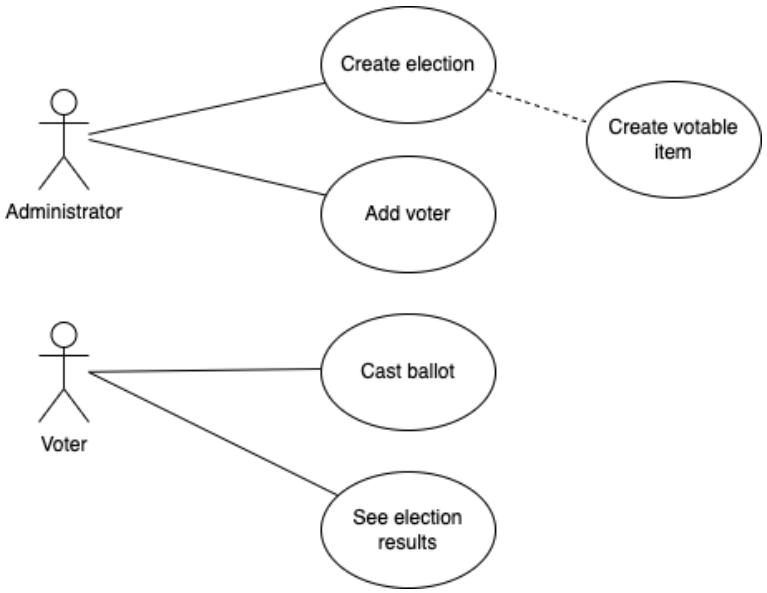
### 5.5.2 Nefunkční požadavky

V kapitole 3.1 jsem již popsal požadavky pro kvalitní volební systém. V kapitole 4 jsem následně popsal, jak každý z těchto požadavků zaručit v kontextu volebního systému založeného na blockchainu. V následujících bodech shrnu ty nefunkční požadavky, které se vztahují přímo na aplikaci:

1. **Důvěryhodnost systému:** Požaduji, aby aplikace poskytovala uživateli pocit důvěry v zabezpečenost voleb.

2. **Škálovatelnost systému:** Požaduji, aby byla aplikace dobře škálovatelná pro menší až střední množství uživatelů. Více o škálovatelnosti v kapitole 4.2.4.

## 5.6 Use case diagram/Diagram případů užití




**Obrázek 5.1:** Diagram případů užití

Pro diagram případů užití jsem zvolil tři aktéry - administrátora voleb (zde zvaného **Administrator**), voliče (zde zvaného **Voter**), a subjekt pro který se volí (zde nazvaný **Votee**).

Administrátor může vytvořit volby a podívat se na výsledky voleb. Volič může odvolit a podívat se na výsledek voleb poté, co volby skončí. Subjekt, pro který se volí, se může podívat na výsledek voleb a může získávat hlasy od voličů.

# Kapitola 6

# Návrh řešení

V této kapitole popíšu návrh aplikace, která bude řešit problém diskutovaný v předchozí kapitole. Popíšu základní myšlenku za fungováním mého řešení, následně zobrazím řešení pomocí různých diagramů (class diagram, sekvenční diagram).

## 6.1 Popis řešení

Záměrem projektu je demonstrovat vlastnosti volební aplikace založené na blockchainové technologii a analyzovat jí ze stránky bezpečnosti, jednoduchosti provedení, a splnění předpokladů pro volební systém.

Volební aplikace bude navenek obvyklá webová aplikace, do které se uživatel přihlásí pomocí anonymizovaného identifikačního čísla, které mu bude přiděleno. Toto číslo bude unikátní pro každého voliče v jedné volbě (to znamená, že když se jeden volič bude účastnit více voleb, bude mu přiděleno jiné identifikační číslo). Do aplikace se bude možné přihlásit pouze v časovém období, které bude definováno administrátorem příslušných voleb. Administrátor bude moct vytvořit volby pomocí formuláře, kde přidá časové období kdy se volba odehrává, a přidá jednotlivé voliče skrz uživatelské rozhraní po vytvoření volby.

Poté, co se uživatel přihlásí do aplikace v příslušném časovém období, bude mu nabídnuta možnost dát jeden hlas jednomu kandidátovi, kterému chce uživatel dát hlas. U každého kandidáta bude existovat popis s informacemi o daném kandidátovi.

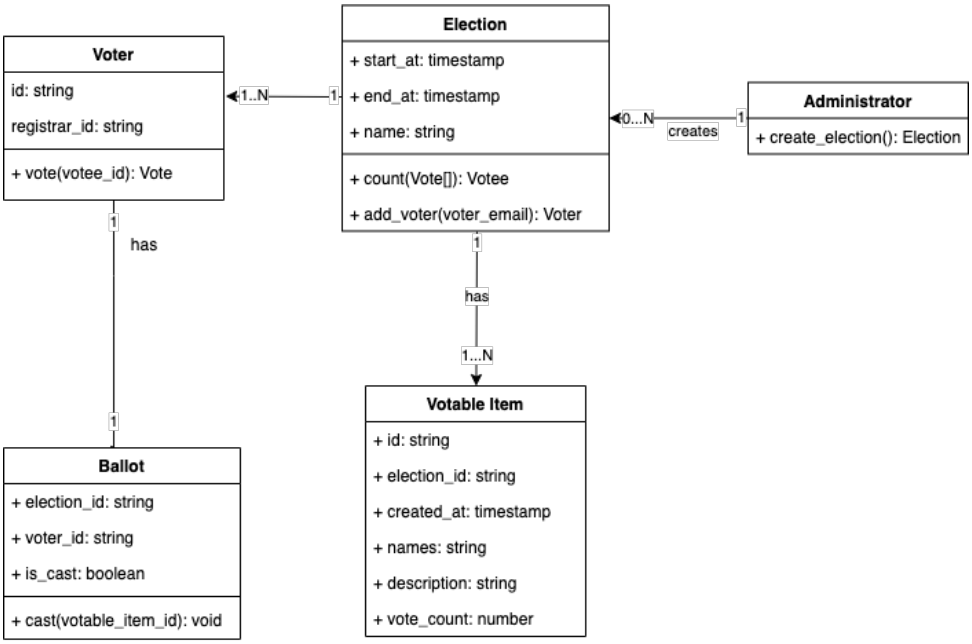
Po udělení hlasu (odvolení volebního lístku) se hlas zapíše do blockchainové databáze a volič už ho nemůže změnit. Udělení hlasu je striktně anonymní, v databázi neexistuje souvislost mezi volebním lístkem a počtem hlasů, který

kandidát má.

Po skončení volebního období se aplikace ujistí, že součet počtu hlasů jednotlivých kandidátů se rovná počtu odvolených volebních lístků v databázi.

Po ukončení volebního období se bude uživatel moct podívat na svůj volební lístek tak, jak je zapsán v databázi. Seznam všech volebních lístků a volebních subjektů v rámci voleb, kterých se uživatel účastní, bude uživateli prezentován po skončení volby. Toto bude reprezentováno možností získat informace z veřejného API, které bude propojené s blockchainovou databází.

## 6.2 Class/UML diagram




**Obrázek 6.1:** Class diagram (diagram tříd)

Pro Class diagram, neboli diagram tříd, jsem zvolil pět hlavních tříd, které reprezentují důležité entity v mojí aplikaci.

Třída **Voter** reprezentuje uživatele, který volí. Uživatel má své identifikační číslo a má schopnost vytvořit hlas. Ke každému voliči může být přiřazen buď jeden, nebo žádný hlas.

Třída **Ballot** reprezentuje volební lístek. Každému voliči je přiřazen jeden volební lístek. Volební lístek má v sobě identifikaci volby a voliče. Volební lístek může být odvolen, což přičte hlas k počtu hlasů jednotlivého volebního

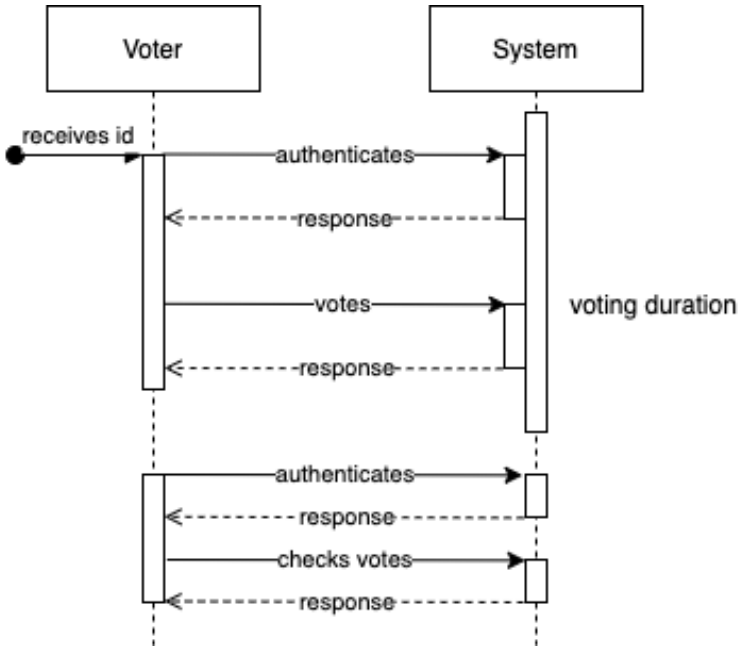
subjektu.

Třída **Votable Item** reprezentuje subjekt, pro který voliči volí. Každý subjekt má své identifikační číslo, jméno a popis, který usnadňuje rozhodování voličů. Každý subjekt může získat až N hlasů, kde maximální N je počet zaregistrovaných voličů.

Třída **Election** reprezentuje volby, ve kterých voliči volí pro subjekty. Každá volba má datum kdy začíná a datum kdy končí. Každé volby mají také unikátní jméno. Do volby můžeme přidat voliče. Po spočítání hlasů již nemůžeme přidávat voliče.

Třída **Administrator** reprezentuje stranu, která volby vytváří. Tato strana vytvoří volby s přednastaveným počtem subjektů (Votable Item) a zvolí základní parametry, jako kdy začíná a kdy končí. Tato strana také může přidávat voliče předtím, než začne volební období.

## 6.3 Sekvenční diagram




**Obrázek 6.2:** Sekvenční diagram

Pro sekvenční diagram jsem zvolil zobrazení dvou hlavních způsobů, jak bude uživatel se systémem interagovat. První způsob je odvolení, kde pokud je uživatel přihlášen do systému ve volebním období určeném administrátorem voleb, tak může udělit svůj jeden hlas. Pokud se přihlásí do systému po konci

volebního období, může už jen vidět odvolené hlasy. Před začátkem volebního období se volič do systému nemůže přihlásit.

## 6.4 Shrnutí kapitoly

V této kapitole jsem shrnul, jak bude vypadat aplikace, kterou pro účely této diplomové práce vytvořím. Tato aplikace se bude sestávat ze dvou částí, administrátorské a uživatelské.

# Kapitola 7

# Analýza technologií

V této kapitole bych rád definoval technologie a nástroje, které jsem zvolil pro vytvoření demonstrační aplikace. Nakonec zvolené technologie shrnu v diagramu nasazení.

Aplikace se bude sestávat ze tří částí:

- **Frontendová aplikace**, kterou používá koncový uživatel.
- **Backendová aplikace**, která vytvoří API, s kterým komunikujeme s databází
- **Blockchainová databáze**, která nám bude sloužit pro ukládání hlasů a volebních subjektů.

Rád bych nejdříve začal diskuzí o blockchainové databázi kterou jsem použil, protože blockchainová technologie je stěžejním tématem této práce.

## 7.1 Blockchainová databáze

V této podkapitole bude nejprve řešeno, proč nebyla pro řešení zvolena technologie Ethereum Smart Contracts, která je zmíněna v oficiálním tématu této práce. Následně budou porovnána dostupná řešení blockchainových databází a bude zvoleno řešení, které se použije ve vývoji aplikace.

### 7.1.1 Ethereum Smart Contracts

V oficiálním tématu této diplomové práce je zmíněno, že při vytváření aplikace použiji speciálně Ethereum Smart Contracts. Od tohoto požadavku jsem nakonec ustoupil po shodě s vedoucím práce.

Bylo od něj odstoupeno z toho důvodu, že jsme považovali za lepší udělat podrobnou rešerši všech dostupných technologií na trhu a z nich zvolit tu nejlepší pro tento případ použití. Jak je vidět níže v porovnání, síť Ethereum nedostatečně podporuje soukromé blockchainové sítě, které jsou nutnou vlastností pro volební aplikaci. Také, jak bylo již nastíněno v kapitole 4.2.4, škálovatelnost sítě Ethereum je malá v porovnání s jinými dostupnými řešeními.

### 7.1.2 Výběr blockchainové databáze

Blockchainová databáze je nejdůležitějším článkem této demonstrační aplikace, a tématem celé této diplomové práce. Poskytovat nám bude ukládání prvků v blockchainovém seznamu, správu identity a přístupu, a anonymizaci jednotlivých transakcí. Samotnou logiku, kterou aplikace vyžaduje, budu implementovat pomocí tzv. **smart contract** (viz kapitola 2.5), což je kus kódu který je transparentní a všechny uzly jej musí vykonat, aby byl jeho výsledek uznán a přidán do blockchainového seznamu.

Pro blockchainovou platformu, na kterou postavím zadanou aplikaci, jsem si zvolil několik důležitých kritérií. Tato kritéria byla:

- **Anonymita:** Blockchainová databáze by měla mít funkce, které dokážou zabezpečit anonymitu transakcí.
- **Soukromost:** Měla by mít správu privilegií, protože nechceme umožnit komukoli přistupovat k naší databázi, pouze právoplatným voličům.
- **Rychlost:** Měla by být dostatečně rychlá, alespoň v porovnání se současnými řešeními založenými na blockchainu.
- **Škálovatelnost:** Blockchainová databáze by měla být škálovatelná pro volby co mají potencionálně miliony uživatelů.

Dle mé rešerše jsem zvolil několik kandidátů, které jsem na základě těchto kritérií porovnal. Tito kandidáti jsou **Hyperledger Fabric**, **Ethereum**, **Bitcoin**, **Corda**, **Ripple** a další. Tyto jsou nejvíce používané sítě v současných řešeních založených na blockchainových technologiích.

Informace o škálovatelnosti jednotlivých blockchainových platforem jsou také popsány v kapitole 4.2.4.

Zde je tabulka, která porovnává kandidáty dle výše vypsaných kritérií:

| Porovnání Blockchain platforem | | | | |
|---|---|---|---|---|
| Název | Anonymita | Soukromost | Rychlost | Škálovatelnost |
| Hyperledger Fabric | Ano, PDC (private data channels) | Ano | 3500 op/s | Dobrá |
| Ethereum | Ano, dle implementace | Ne nativně | 15 op/s | Špatná |
| Bitcoin | Ne | Ne | 4-6 op/s | Velmi špatná |
| Corda | Ano | Ano | 700 op/s | Dobrá |
| Ripple | Ne | Ano | 1500 op/s | Dobrá |

**Tabulka 7.1:** Výběr blockchainové databáze.

Po porovnání následujících kritérií jsem zvolil **Hyperledger Fabric**. Nabízí jak možnost vytvořit soukromou síť, tak dobrou škálovatelnost, rychlost a dokumentaci. Je možné jej nasadit v rámci stávájících komerčních řešení (jako např. IBM Blockchain Platform) nebo v rámci vlastního řešení. Pro demonstraci aplikace použiji IBM Blockchain Platform k jeho nasazení, hlavně z důvodu jednoduchosti.

## 7.2 Výběr jazyka pro API backend

Backendové API je potřeba vystavit z toho důvodu, že musíme s frameworkem Hyperledger Fabric komunikovat. Toto je nejsnadnější pomocí již vytvořených SDKs (Software Development Kits). Je mnoho způsobů, jak je možné vytvořit backendové API. Existují například řešení jako **Node.js**, **Go**, **Rust**, a mnohé další. Při výběru backendového API jsem vybíral pomocí několika kritérií:

- **Familiárnost:** Toto je hlavní kritérium, pomocí kterého si volím backendové API. Vzhledem k tomu, že pracuji jako vývojář, radši využiji jazyk a framework se kterým mám zkušenosti, ať mě učení se nového jazyka zbytečně nezdržuje od implementace.
- **Přepoužitelnost:** Pokud to jde, rád bych použil stejný jazyk pro psaní smart contractu a tohoto API.
- **Dokumentace:** Jazyk nebo framework by měl být kvalitně zdokumentovaný. Pokud ho neznám, měl bych se v něm být schopný rychle zorientovat.

Dle mé rešerše jsem zvolil tři kandidáty, které jsem porovnal podle výše zmíněných kritérií. Tito kandidáti jsou **Node.js**, **Go** a **Java**. Toto jsou také

tři jazyky které je možné použít pro psaní smart contractů ve výše zvoleném blockchainovém frameworku Hyperledger Fabric.

V následující tabulce jsem tyto tři kandidáty porovnal:

| Porovnání jazyků pro API backend | | | |
|---|---|---|---|
| Název | Familiárnost | Přepoužitelnost | Dokumentace |
| Node.js | Ano, pracuji v JavaScriptu | Ano, dokonce i s frontendem (typy apod.) | Výborná |
| Go | Ne, velmi malá | Ano, pouze s blockchanovou vrstvou | Výborná |
| Java | Průměrná | Ano, pouze s blockchainovou vrstvou | Důkladná, ale nepřehledná |

**Tabulka 7.2:** Výběr jazyku pro API backend.

Dle porovnání pomocí těchto kritérií jsem vybral **Node.js**. Nejenže jsem v něm již pracoval a mám s ním dlouhodobé zkušenosti, ale také mohu přepoužívat typy z TypeScriptu a různé společné funkce. Go je jazyk, který už mě dlouhodobě zajímá, ale bohužel jsem se s ním ještě příliš neseznámil.

## 7.3 Výběr frontendového frameworku

Frontend aplikace je prezentační vrstva aplikace, tedy ta vrstva, se kterou bude interagovat koncový uživatel. Proto je potřeba, aby frontend byl kvalitně vystaven, jak po designové stránce tak po stránce uživatelského zážitku (UX).

Jazyk, ve kterým bude frontend napsán, je JavaScript. V dnešním světě prakticky není jiné možnosti jak postavit interaktivní webové aplikace, vzhledem k tomu, že internetové prohlížeče ani žádný jazyk neinterpretují (Je také možnost použít C++ a kompilovat ho do WebAssembly, ale to je způsob který není široce adoptovaný a používá se pro aplikace kde výkon je kritická priorita).

Proto, když vybíráme, jak budeme vytvářet frontend webových stránek, obvykle volíme mezi frameworky JavaScriptu. Těch je mnoho, ale mezi nejpopulárnější patří **React**, **Vue** a **Angular**. Jako nadstavbu nad těmito základními frameworky existují také tzv. meta-frameworky, které poskytují ucelený způsob, jak vytvářet webové stránky, kde standardizované postupy jsou již vyřešeny za developera (jako například routing či server-rendering). Mezi nejpopulárnější meta-frameworky patří např. Next.js (nadstavba nad Reactem) nebo Nuxt.js (nadstavba nad Vue). Angular je díky jeho filozofii

meta-framework sám o sobě.

Kritéria, podle kterých vybírám frontendový framework, jsou:

- **Familiárnost:** Stejně jako při výběru backendu, i u frontendu je u mě nejdůležitější familiárnost. Jsem v práci primárně frontend vývojář, proto bych při vývoji rád zvolil nástroje, se kterými jsem seznámen a ve kterých umím pracovat.

- **Kvalitní dokumentace:** Dobrá dokumentace je základem každého kvalitního vývojového nástroje a vždy se snažím zjistit, jak je dokumentace provedená a jak snadno se v ní vyhledávají věci, které pomocí frameworku chceme učinit.

- **Snadnost použití:** U používání frameworku je pro mě důležitá nízká komplikovanost, co se syntaxe a používání základních vlastností frameworků týče (např. podmíněné zobrazování prvků).

Jako kandidáty jsem vybral výše zmíněné meta-frameworky, tedy **Next.js**, **Nuxt.js** a **Angular**. Všechny tyto nástroje jsou populárními volbami pro vývojáře a se všemi se dá vyvinout kvalitní webová aplikace.

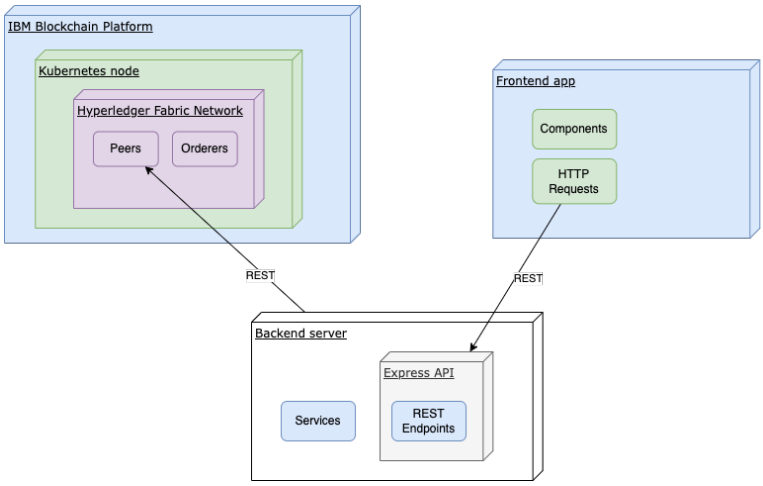
Následující tabulka shrnuje mé subjektivní posouzení těchto kandidátů dle výše vypsaných kritérií:

| Porovnání JavaScript frameworků | | | |
|---|---|---|---|
| Název | Familiárnost | Kvalitní dokumentace | Snadnost použití |
| Next.js | Ano, skvělá | Ano | Snadno se používá |
| Nuxt.js | Ano, dobrá | Ano | Snadno se používá |
| Angular | Ne | Ano | Průměrně obtížně se používá |

**Tabulka 7.3:** Výběr JavaScript frameworku.

Dle tabulky je patrně poznat, že jsem pro vývoj frontendu aplikace zvolil framework **Next.js**. S Next.js jsem dobře seznámený, už dlouho s ním pracuji a poskytuje mnoho funkcí, které mi usnadní vývoj, jako například vestavěná podpora pro renderování na serveru.

## 7.4 Diagram komponent




**Obrázek 7.1:** Component diagram

Diagram komponent znázorňuje a shrnuje, jak je volební aplikace strukturována. Jako databázovou vrstvu máme ledger poskytovaný frameworkem Hyperledger Fabric. Tento ledger běží na IBM Blockchain Platform, která obsahuje Kubernetes Cluster. Kubernetes je služba od společnosti Google, která spravuje kontejnerizované aplikace.

Pro API vrstvu jsem zvolil backendovou aplikaci založenou na Node.js, a API postavené na frameworku Express. Toto API slouží jako mezistupeň mezi frontendovou aplikací, kterou používá uživatel, a blockchainovou databází popsanou výše. Backend komunikuje s výše zmíněným Hyperledger Fabric pomocí SDK [1].

Pro frontendovou aplikaci jsem zvolil Next.js. Next.js komunikuje s API backendem pomocí protokolu REST [2].

## 7.5 Shrnutí kapitoly

Každá část aplikace byla vybrána pomocí několika kritérií. Pro frontendovou a backendovou část jsem si vybíral hlavně podle mé vlastní familiárnosti s daným jazykem/frameworkem, protože každá z prezentovaných možností je dostatečně kvalitní pro splnění požadavků na kvalitu a použitelnost, pokud se s ní dobře zachází. Výběr blockchainového frameworku mi trval nejdéle,

[1] `https://www.npmjs.com/package/fabric-network`
[2] `https://restfulapi.net/`

protože jsem s blockchainem v pracovním ani osobním životě nepracoval a nemám s ním tedy předchozí zkušenosti. Věřím ale, že Hyperledger Fabric je nejlepší kandidát pro případ použití, který se touto aplikací snažím vyřešit.

# Kapitola 8

# Implementace aplikace

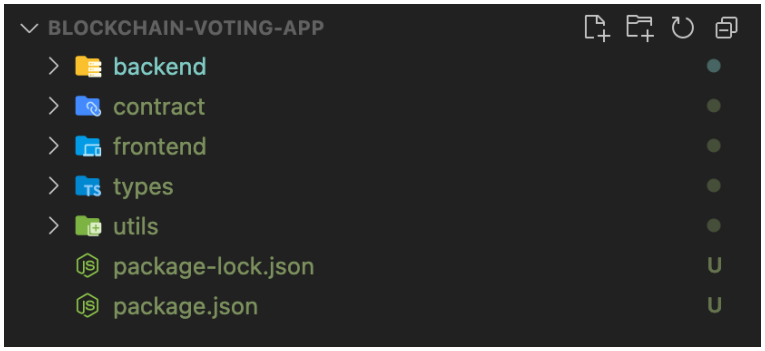
V této kapitole se budu věnovat implementaci samotné aplikace. Nejprve pohovořím o výzvách, se kterými jsem se setkal při vytváření smart contractu, který zaštiťuje hlavní logiku celého programu, a jak jsem je nakonec vyřešil. Potom promluvím o vytváření backendového API, v některých ohledech nejjednodušší části aplikace, která ale sama o sobě skýtala některé náročnější výzvy. Nakonec se rozepíšu o frontendové části, jak je postavená a jakým způsobem jsem ji navrhl.

## 8.1 Adresářová struktura

Jelikož má aplikace tři části, které sdílejí stejný programovací jazyk (JavaScript a jeho nadstavbu, TypeScript), rozhodl jsem se jako strukturu aplikace použít tzv. monorepo.

Monorepo je adresářová struktura, kde se do jednoho "repa"(tedy repozitáře v rámci verzovacího systému Git), dají různé projekty, které mezi sebou nemají závislosti.

Adresářová struktura mnou vytvořeného monorepa vypadá takto:




**Obrázek 8.1:** Adresářová struktura

Složka **contract** obsahuje smart contract. Jeho vytváření popisuji níže v kapitole 8.2.

Složka **backend** obsahuje API backend vytvořený s pomocí frameworku Express. Jeho vytváření popisuji níže v kapitole 8.3.

Složka **frontend** obsahuje frontendovou webovou aplikaci postavenou na frameworku Next.js. Jeho implementaci popisuji níže v kapitole 8.4.

Složka **types** obsahuje typy, které popisují struktury jako například **Voter**, **Election** a jiné. Tyto se používají ve všech třech částech této aplikace.

Složka **utils** obsahuje funkce, jejichž použití není specifické pro jednu aplikaci. Například dešifrování struktury **Buffer** na text je jedna z těchto funkcí.

## 8.2 Vytváření smart contractu

Vytváření smart contractu bylo nejnáročnější částí vytváření samotné aplikace. Toto bylo převážně z důvodu, že jsem předtím neměl žádné zkušenosti s vytvářením smart contractů, nebo s programováním pro blockchain. Naštěstí jsem se díky existujícím YouTube tutoriálům a předpřipraveným šablonám pro vývoj kontraktů zorientoval a podařilo se mi úspěšně vytvořit kontrakt.

Hyperledger Fabric kontrakty jsou soubory, které jsou kompilované do tzv. ChainCode. Hyperledger Fabric poskytuje řešení pro kompilaci ze tří jazyků, JavaScript (a jeho nadstavbu TypeScript), Go, a Java. Pro vývoj svého smart contractu jsem si zvolil jazyk **TypeScript**, vzhledem k tomu, že je přepoužitelný ve všech třech aspektech (frontend, backend i contract) této aplikace a sám s ním mám zkušenosti z pracovního prostředí.

IBM poskytuje pro vývoj smart contractů vlastní rozšíření do známého editoru VSCode (jednoho z nejpoužívanějších editorů kódu na světě, hlavně

pro JavaScript). Toto rozšíření se jednoduše jmenuje IBM Blockchain Platform Extension. [25] Bez tohoto rozšíření by dle mého názoru byl vývoj pro Hyperledger Fabric mnohem obtížnější.

V souboru **contract/src/index.ts** reprezentujeme smart contract třídou VoterContract. VoterContract obsahuje všechny transakce, které v tomto kontraktu mohou proběhnout. Tyto transakce bych rád rozdělil na dvě části - ty, které mění stav blockchainové databáze (tzv. mutations) a ty, které se na něj pouze ptají (tzv. queries).

### 8.2.1 Queries

V následujících odstavcích budu používat **id** jako reprezentaci unikátního identifikátoru pro určitou entitu v databázi.

Queries jsou transakce, které pouze vrátí stav seznamu, obvykle filtrovaný nějakým kritériem, jako je například id předmětu, který chceme získat. Obvykle je používáme pro ověření stavu, jako například stavu voleb, nebo získání stavu voličova volebnímu lístku.

Některé transakce jsou již poskytnuty s šablonou, kterou poskytuje IBM pro vývoj kontaktů na platformě HyperLedger Fabric. Tyto jsou:

- **readMyAsset**: Generická funkce pro přečtení informace ze seznamu dle id.
- **myAssetExists**: Generická funkce pro zjištění, zda informace s určitým id existuje, nebo neexistuje.
- **queryAll**: Generická funkce pro získání všech informací ze seznamu.
- **queryWithQueryString**: Generická funkce, která přijímá tzv. query, která může obsahovat parametry pro filtrování specifického požadavku.

Tyto funkce slouží k základní interakci s blockchainovým seznamem a jsou používány v téměř všech mnou vytvořených transakcích.

Následující queries jsem přidal já:

- **getElectionWorldState**: Transakce, která vrátí stav seznamu pro danou volbu, aby volič mohl mít možnost podívat se na všechny transakce. Vrátí volby, kandidáty a hlasovací lístky.
- **getVotablesByElectionId**: Transakce, která vrátí kandidáty, kteří jsou zapsaní v jednotlivé volbě.

- **getElectionByVoterId**: Transakce, která vrátí informace pro volby, do kterých je volič pod daným id zapsán.

### 8.2.2 Mutations

Mutations jsou transakce, který mění stav blockchainové databáze. Zatímco u queries je potřeba účast jen jednoho uzlu v blockchainové síti, u mutations je potřeba, aby se všechny uzly v síti shodly na provedení dané transakce. Jednotlivé uzly (Peers) podepisují vlastním podpisem svůj souhlas, a následně se transakce seřadí do bloků a provede na všech uzlech najednou.

To, že všechny uzly musí s danou transakcí souhlasit, způsobuje problém škálovatelnosti (rychlosti jednotlivých transakcí při větším počtu uživatelů) u blockchainových sítí. Ale vzhledem k tomu, že na případ použití této aplikace by bylo potřeba souhlasu maximálně několika tisíc bloků (narozdíl od několika milionů jako například u sítě Ethereum), škálovatelnost je mnohem lepší než u jiných blockchainových sítí.

Tyto entity přijímají argumenty jako pole stringů (textových řetězců), takže pro posílání složitějších informací jsem využil formát JSON (JavaScript Object Notation), který jsem před kompilací kontroloval pomocí TypeScript typů. Obrázek níže je příklad mutace pro vytváření voleb:

```
async createElection(ctx: Context, args: string) {
    const { votableItems, name, startDate, endDate }: CreateElectionArgs =
        JSON.parse(args);

    // Validate the request

    if ( isDateBeforeToday(startDate) || !isDateBeforeOtherDate(startDate, endDate) )
        return { error: "An election cannot start in the past and start date must be before end date.", };

    for (const votableItem of votableItems) {
        if ( !votableItem?.description?.length || !votableItem?.name?.length )
            return { error: "There is an error with the format of one of the votable items.", };
    }

    // Generate the votable item ids
    const mappedVotableItems = map(votableItems, votableItem => ({
        ...votableItem,
        type: "votableItem",
        voteCount: 0,
        id: generateId(),
    }))

    // Create the election
    const election: Election = {
        id: generateId(),
        votableItemIds: map(mappedVotableItems, "id"),
        name,
        startDate,
        endDate,
        type: "election",
    };

    // Add the necessary election and votable items into state

    await addToState(ctx, election.id, election);

    for (const votableItem of mappedVotableItems) {
        await addToState(ctx, votableItem.id, votableItem);
    }

    // Return the election that we created.

    return { success: 'The election has been added' }
}
```

**Obrázek 8.2:** Ilustrační příklad kódu mutace.

V kontraktu VoterContract existují tyto mutations [1]:

- **createElection**: Vytvoří volbu, pokud žádná podobná volba ještě neexistuje. Přijímá jako parametry jméno volby, datum začátku a konce volby, a entity, které budeme volit.
- **createVoter**: Vytvoří voliče. Přijímá parametry registrační číslo (například číslo občanky) a jméno voliče.
- **deleteVoter**: Smaže voliče. *Tato funkce je vytvořená pro testovací účely.*

[1] Celý kód kontraktu je možné vidět zde: `https://github.com/Rednegniw/blockchain-voting-app/blob/main/contract/src/index.ts`

- **deleteMyAsset**: Smaže jakýkoli záznam v seznamu pomocí id.
- **castVote**: Přidá hlas kandidátovi a označí volební lístek jako použitý. Přijímá identifikační číslo kandidáta a identifikační číslo voliče.
- **restartWorldState**: Restartuje stav seznamu. *Používá se pro testovací účely.*

### 8.2.3 Testování

Smart contract je testovaný pomocí automatických TypeScript testů, které využívají testovací framework Jest [2]. Tyto testy jsou ve formátu tzv. unit-testů. Každý test testuje jednu transakci probíhající ve smart contractu.

IBM Blockchain Platform VSCode Extension (viz kapitola 8.2) má funkci pro vytvoření šablony pro psaní automatických testů. Tyto funkce se vyskytují v souboru *ts-smart-contract-util.ts*. Poskytují funkce pro připojení se ke konfigurované blockchainové databázi a následné spuštění transakce.

Testy jsou spuštěné oproti skutečné běžící instanci Hyperledger Fabric (obvykle spouštěná lokálně). Toto zaručuje, že testy skutečně kontrolují použití v reálném prostředí.

## 8.3 Vytváření API backendu

Backendové API slouží jako most mezi frontendovou aplikací a smart contractem. Pro komunikací se smart contractem využívá tzv. *fabric-network* SDK.[3]

Pro frontedovou aplikaci vystavuje vlastní REST API založené na známem frameworku pro Node.js, Express [4]. Express poskytuje jednoduché rozhraní pro vytvoření REST API endpointů. Tyto jsou vytvořeny podle specifických potřeb aplikace.

Pro všechny endpointy využívám abstrahovanou funkci **contractRequest**. Tato funkce se připojí k běžící instanci Hyperledger Fabric, a vyvolá v ní transakci s vloženými argumenty. Takto daná funkce vypadá:

[2] `https://jestjs.io/`
[3] `https://www.npmjs.com/package/fabric-network`
[4] `https://expressjs.com/`

```
*/
export const contractRequest = async ({
  res,
  isQuery,
  method,
  args = [],
  userName = appAdmin,
}: ContractRequestProps) => {
  const networkObj = await network.connectToNetwork(userName || appAdmin);

  if (isNetworkObj(networkObj)) {
    try {
      const response = await network.invoke(networkObj, isQuery, method, args);
      const parsedResponse: any[] = await decodeAndParse(response as Buffer);

      res.send(parsedResponse);
    } catch (err) {
      res.status(500).send({ error: err });
    }
  } else {
    res.status(500).send(networkObj);
  }
};
```

**Obrázek 8.3:** Funkce **contractRequest**

Níže vyjmenuji jednotlivé endpointy které toto API vystavuje:

- **queryAll**: Stejné jako transakce **queryAll**. Získá veškeré informace z databáze.
- **getElectionWorldState**: Stejné jako transakce **getElectionWorldState**. Vrátí celkový stav probíhající volby. Tento API endpoint vrací výsledky až poté, co volba skončí.
- **getElection**: Tento API endpoint používá transakci **getElectionByVoterId**. Přijímá jako argument identifikaci voliče a vrátí zpět voliče, pokud existuje. Jinak vrátí chybovou hlášku.
- **castBallot**: Tento API endpoint používá transakci **castVote**. Má stejné argumenty jako tato transakce a vrací stejný výsledek.
- **createElection**: Tento endpoint používá transakci **createElection**. Má stejné argumenty jako ve výše popsané transakci.
- **registerVoter**: Tento API endpoint vytvoří voliče. Pokud volič ještě neexistuje, vytvoří pro něj tzv. Wallet (tedy adresář s jeho soukromým klíčem), kterým se jako tento volič přihlašuje. Více o této funkci v kapitole níže.
- **validateVoter**: Vrátí, zda volič existuje. Používá se pro kontrolu, že neexistuje volič se stejným registračním číslem (jako například číslo občanského průkazu).
- **reinitWorldState**: Používá transakci **restartWorldState**. *Používá se pouze pro testovací účely.*

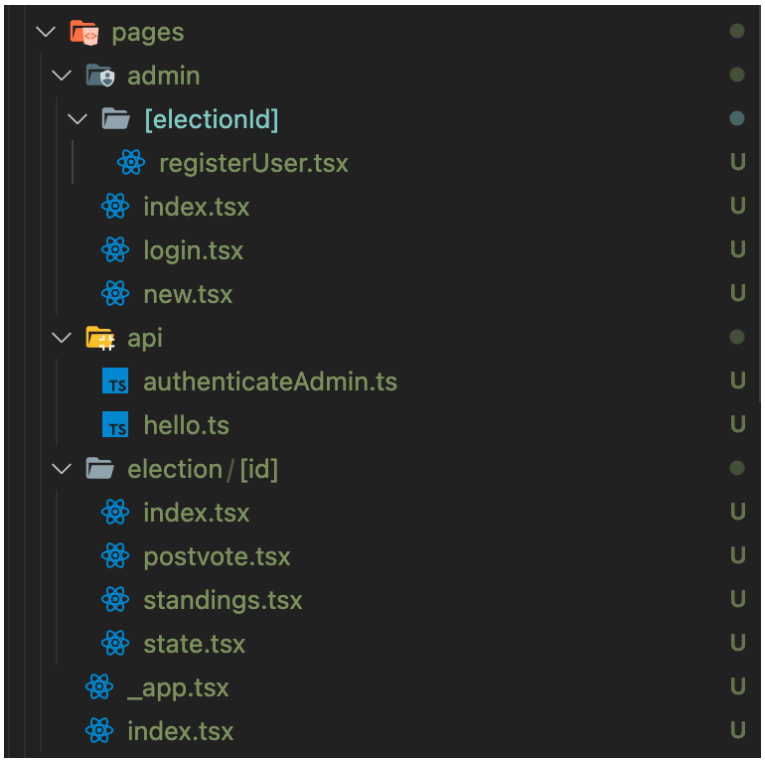
## 8.4 Vytváření frontendové aplikace

Frontendová aplikace zaštiťuje uživatelské rozhraní aplikace. Uživatel ji používá, aby mohl interagovat se systémem a odvolit.

Jak jsem již uvedl v kapitole 7.3, rozhodl jsem se pro vytvoření frontendu ve frameworku Next.js [5].

### 8.4.1 Struktura podstránek

V Next.js je aplikace vytvořená z takzvaných **pages**, což jsou komponenty, který každý reprezentuje jednu stránku aplikace. Například, pokud máme stránku **post.ts**, stránka bude dostupná pod url **/post**. V rámci volební aplikace jsem se rozhodl vytvořit podstránky takto:




**Obrázek 8.4:** Adresářová struktura složky **pages**

Adresářová struktura je rozdělená na dvě části - uživatelskou a administrátorskou. Zatímco do uživatelské části se uživatel přihlašuje pomocí jemu přiřazeného id, do administrátorské části se administrátor přihlašuje pomocí

[5] `https://nextjs.org/`

přiřazeného uživatelského jména a hesla. V současnosti je administrátorské heslo pouze "natvrdo"zakódované. Vytváření administrátorských účtů není v záběru této diplomové práce.

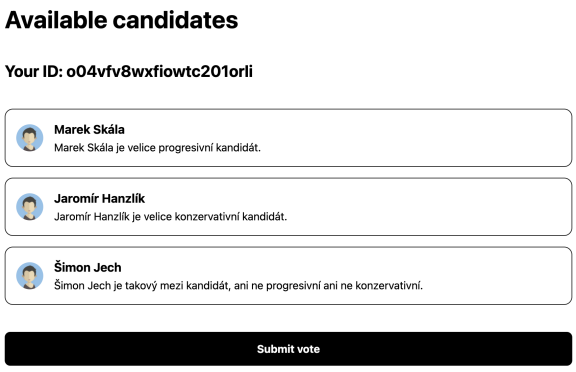
### 8.4.2 Získávání informací před vykreslením stránky

V rámci této aplikace často využívám schopnost frameworku Next.js spustit na serveru kód před načtením stránky. Pro tuto funkci mohu využít funkci **getServerSideProps** [6]. Tato funkce mi umožňuje např. zkontrolovat, zda je uživatel skutečně přiřazen do některé z probíhajících voleb, či zda tato volba stále běží.

### 8.4.3 Uživatelský průchod

Uživatel nejprve začne na stránce `"/"`, kde mu bude prezentována možnost zadat svoje id, které mu bylo zasláno administrátorem voleb (například v dopisu).

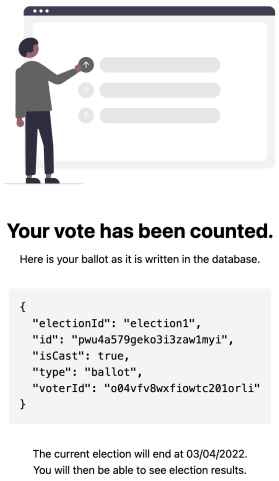
Po stisknutí tlačítka ho aplikace přesune do stránky **"/election/[voterId]"** (kde voterId je identifikační číslo uživatele které zadal na předchozí stránce). Tato stránka může vypadat například takto:




**Obrázek 8.5:** Stránka kde se uživatel odvolí.

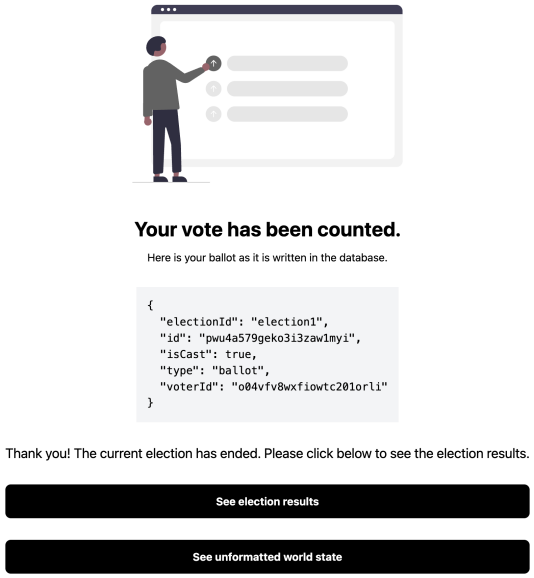
[6] `https://nextjs.org/docs/basic-features/data-fetching#getserversideprops-server-side-rendering`

Na této stránce uživatel zvolí jednoho kandidáta a po stisknutí tlačítka uživatel odvolí. Následně uživatele, pokud byla volba úspěšná, přesuneme na podstránku **"/election/postvote"**, kde uživatel uvidí následující:




**Obrázek 8.6:** Stránka potom, co uživatel odvolí.

Dokud volba neskončí, tak uživatel uvidí tuto obrazovku. Pokud se do aplikace přihlásí poté, co volba skončí, objeví se dvě tlačítka. Mohou být viděna v následujícím obrázku:




**Obrázek 8.7:** Stránka potom, co uživatel odvolí a volby skončí.

Po kliknutí na tlačítko **"See election results"** uživatele vezmeme na stránku, kde sloupcovým grafem vyobrazujeme stav voleb, tedy počet hlasů, který jednotliví kandidáti získali.

Po kliknutí na tlačítko **"See unformatted world state"** uživatele vezmeme na stránku, kde je ve formátu JSON vypsán stav voleb - tedy volby, kandidáti, a hlasovací lístky.

### 8.4.4 Testování frontendové aplikace

Pro testování aplikace jsem vytvořil end-to-end testy s pomocí nástroje Cypress [7]. Cypress je framework pro automatizované testování webových stránek pomocí automaticky běžícího webového prohlížeče, který se snaží chovat jako běžný uživatel.

V rámci testování jsem otestovat uživatelský průchod, tedy registraci uživatele, přihlášení uživatele, volbu uživatele, a následující stránky po skončení voleb.

[7] `https://www.cypress.io/`

## 8.5 Nasazení blockchainové databáze

Pro nasazení blockchainové databáze jsem se rozhodl použít IBM Blockchain Platform na IBM Cloud [8]. IBM Blockchain Platform je jediný poskytovatel automatického způsobu nasazení pro aplikace vyvíjený pomocí Hyperledger Fabric. V mé rešerši se mi nepodařilo najít jiný předpřipravený způsob, jak aplikaci nasadit.

IBM Cloud propojí instanci služby IBM Blockchain Platform s instancí služby Kubernetes, taktéž běžící na IBM Cloud. Bohužel, pokud chceme tuto Kubernetes instanci vytvořit zadarmo, bude smazána po 30 dnech. Pro nasazení aplikace musím tedy periodicky vytvářet nové instance služby Kubernetes.

[8] `https://www.ibm.com/cz-en/blockchain/platform`

# Kapitola 9

# Volební aplikace jako přepoužitelný framework

Hlavní výhodou této aplikace, dle mého názoru, je, že je možné jí pře použít jako základ pro vlastní volební aplikaci či dokonce ji vylepšit pro vlastní účely. Smart contract, který jsem vytvořil (viz kapitola 8.2), je použitelný na jakýkoli typ voleb, a poskytuje základovou strukturu pro vytváření volební aplikace na platformě Hyperledger Fabric.

## 9.1 Proč by měly být volební aplikace vytvářeny na Hyperledger Fabric?

Hyperledger Fabric je neustále zlepšující se open-source technologie, která se snaží být modulárním rozhraním pro vytváření blockchainových aplikací, převážně v rámci businessu. Dle mé rešerše pevně věřím, že Hyperledger Fabric je z pohledu škálovatelnosti, bezpečnosti a rychlosti nejlepší platforma pro vytváření aplikací na blockchainu.

## 9.2 Co moje aplikace nabízí?

Projekt je plně open-source, což znamená že kdokoli může do zdrojového kódu nahlédnout. Můžeme se na něj podívat na této adrese: `https://github.com/Rednegniw/blockchain-voting-app`

Dle mé rešerše neexistuje žádná aplikace, která by nabízela to samé, co moje aplikace nabízí. Nejblíže můžeme najít aplikaci IBM Evote [1], jejíž strukturou

[1] `https://github.com/IBM/evote`

jsem se velice inspiroval. Můj projekt můžeme určitým způsobem vnímat také jako převedení této aplikace do současné doby. Oproti této aplikace nabízí moje aplikace vytváření voleb, získání informací v závislosti na jednotlivé volbě, je plně napsaná v TypeScriptu, je plně otestovaná, a je založená na nejnovější verzi Hyperledger Fabric. Aplikaci můžeme vzít jako základ při vytváření skutečné volební aplikace založené na blockchainu - a plně doufám, že v ní někdo najde užitek.

Nejlépe přepoužitelná část aplikace je dle mého názoru smart contract, který jsem vytvořil. Poskytuje základ, který potřebujeme pro postavení volební aplikace - vytváření voleb, vytváření voličů, počítání hlasů, a další.

## 9.3 Jak můžeme aplikaci použít?

Pro zprovoznění smart contractu ho bude uživatel buď muset nasadit na platformě IBM Blockchain Platform, nebo vytvořit vlastní instanci Hyperledger Fabric. Tento proces je komplikovaný a nemám s ním zkušenosti, proto se v této kapitole zaměřím na počáteční spuštění aplikace v lokálním prostředí.

### 9.3.1 Spuštění aplikace lokálně

Pro spuštění aplikace lokálně je potřeba provést následující kroky:

1. Nainstalovat editor kódu VSCode. Tento editor je volně stažitelný. [2]
2. Pokud již nemáte nainstalovaný na svém zařízení, nainstalujte Docker. [3]
3. Stáhněte repozitář [4] do svého zařízení a otevřete ho v editoru VSCode.
4. Po instalaci nainstalujeme rozšíření IBM Blockchain Platform Extension. Můžeme jej nainstalovat skrze podmenu "Extensions"nebo skrze webovou stránku. [5]
5. Na levé straně VSCode (navigační menu) by se měla objevit ikonka čtverce, která vede do menu "IBM Blockchain Platform".
6. Mělo by se objevit levé menu se čtyřmi částmi - **Smart Contracts**, **Fabric Environments**, **Fabric Gateways** a **Fabric Wallets**.

[2] `https://code.visualstudio.com/`
[3] `https://www.docker.com/products/docker-desktop`
[4] `https://github.com/Rednegniw/blockchain-voting-app`
[5] `https://marketplace.visualstudio.com/items?itemName=IBMBlockchain.ibm-blockchain-platform`

7. Pod částí **Fabric Environments** spusťte výchozí prostředí - Mělo by se jmenovat **1 Org Local Fabric**. Nyní se inicializuje prostředí.

8. Klikněte na kanál **mychannel**, který se objeví. Následně klikněte na **"+ Deploy smart contract"**.

9. Měla by se objevit obrazovka s nadpisem **"Deploy smart contract"**. Na této stránce, z menu "Select smart contract", vybereme smart contract který je v kořenovém adresáři tohoto projektu. Měl by se jmenovat **blockchain-voting-app-contract@1.0.0.cds**. Klikněte dvakrát na tlačítko **"Next"** a následně na tlačítko **"Deploy"**.

10. Smart contract by měl být úspěšně nasazen na naši lokální instanci IBM Blockchain Platform. Nyní můžeme posílat transakce skrze přiložené backendové API.

## 9.4 Pro koho je tato aplikace určena?

Rád bych v této diplomové práci dokázal, že pro většinu případů použití je začínat s mojí volební aplikací výhodnější než začínat tzv. "od nuly". Vše samozřejmě závisí na uživatelově případu použití.

Uživatel, pro kterého je přepoužití této aplikace výhodné, je takový uživatel, který je přesvědčen o výhodách použití blockchainu pro účely volební aplikace ale neví jak začít. Moje volební aplikace mu poskytne výchozí platformu, na které může začít a nezačínat tak od nuly. Potencionálně si tak může ušetřit mnoho desítek, až stovek hodin práce.

# Kapitola 10

# Testování vlastností volební aplikace

## 10.1 Bezpečnostní analýza aplikace

V této podkapitole bych se rád podíval na aplikaci z bezpečnostního hlediska. Jako kritéria, proti kterým aplikaci posuzuji, jsem zvolil 10 kritérií OWASP[1]. Následuje výčet jednotlivých kritérií, jejich popis a jak je možné podle nich zhodnotit bezpečnost blockchainové volební aplikace:

### 10.1.1 Broken Access Control

Access Control, tedy řízení přístupu, vynucuje v systému pravidla, která zaručují, že uživatelé nemohou jednat mimo zamýšlená oprávnění. Selhání obvykle vedou k neoprávněnému zveřejnění informací, úpravě nebo dokonce zničení všech dat. Špatné řízení přístupu bylo označeno jako nejčastější bezpečnostní chybou v aplikacích v roce 2021.

Hyperledger Fabric používá X.509 certifikáty, které automaticky vydává speciální část systému - tzv. Certificate Authority. Každý uživatel se musí přihlásit svým soukromým klíčem, který je následně validován veřejným klíčem, uloženým v aplikaci. V Hyperledger Fabric také existují tzv. ACL, neboli Access Control Lists, které umožňují řešit přístup na úrovni jednotlivců nebo skupin jednotlivců.

Bezpečnost systému tedy z pohledu řízení přístupu závisí na bezpečnosti vlastnění soukromého klíče. Pokud je tento klíč odcizen, tak je aplikace stejně zranitelná jako každá jiná aplikace. Blockchainová povaha aplikací tedy dle mého názoru nepřináší žádné bezpečnostní výhody, co se tohoto kritéria týče.

[1] `https://owasp.org/Top10/`

### 10.1.2 Cryptographic Failures

Cryptographic Failures, tedy chyby v šifrování, je kritérium zameřené na zaručení bezpečnosti citlivých dat. Posuzuje, zda neexistují zranitelné toky v systému, které by útočník mohl zachytit a získat tak citlivé informace.

Pro účastníky systému je datová úroveň v mojí aplikaci transparentní, tedy citlivé informace neexistují, pokud už člověk má přístup do systému. Hyperledger Fabric používá pro posílání dat standardní webový protokol HTTPS, který bezpečně zašifruje příchozí a odchozí komunikaci.

### 10.1.3 Injection

Injection, jejíž součástí je také známý útok SQL Injection, je způsob útoku kdy vložíme nebezpečný kus kódu do části aplikace, kde může uživatel vkládat vlastní text. Pokud tomuto není dostatečně bráněno, útočník může donutit systém aby vykonával úkony, které nebyly součástí původního systému, jako například odesílání citlivých dat jiných uživatelů útočníkovi.

Jediný možný vstup pro uživatele, který je následně zobrazen jiným uživatelům, je vytváření voleb administrátorem. Toto je zabezpečeno tím, že framework React má v sobě vestavěné "očišťování"řetězců v aplikaci[2]. I kdyby byla skrze Injection předložena změna do blockchainové, jelikož je aplikace založená na blockchainu, případná změna dat neovlivní ostatní uzly systému.

### 10.1.4 Insecure Design

Insecure Design je pro OWASP nová kategorie, představena v roce 2021[26]. Zaměřuje se na chyby v architektuře systému, jako například citlivé informace zahrnuté v chybových hláškách, špatné ukládání citlivých dat a špatné chránění citlivých dat. Nezabezpečená architektura z definice nemůže být opravena bezpečnou implementací.

Vzhledem k tomu, že toto kritérium je poměrně široké ve svém záběru, je obtížné ho posoudit z pohledu volební aplikace. Aplikace používá standardní bezpečnostní postupy jako validaci hodnot na serveru, standardní chybové hlášky a jiné další obvyklé postupy. Blockchainová povaha aplikace zabezpečuje další úroveň bezpečnosti, například odolnost vůči změně dat nebo chybným transakcím.

[2] `https://reactjs.org/docs/dom-elements.html#dangerouslysetinnerhtml`

### 10.1.5 Security misconfiguration

Security misconfiguration je pravidlo, které se zaměřuje na chybné konfigurace v aplikaci. Chybná konfigurace může být například konfigurace zanechané v systému z vývoje, jako například posílání tzv. "stack trace"v produkční aplikaci nebo umožněný Cross Origin Resource Sharing (CORS).

Tento úhel pohledu nesouvisí z blockchainovými technologiemi, tudíž je aplikace stejně zranitelná vůči těmto chybám, jako kdyby na blockchainu nebyla.

### 10.1.6 Vulnerable and outdated components

Kritérium Vulnerable and outdated components pohlíží na "aktualizovanost"jednotlivých komponent třetích stran v aplikaci, jako například externích knihoven.

Toto kritérium aplikace splňuje. Všechny knihovny třetích stran jsou v aplikaci na nejnovější verzi v době, kdy píšu tuto diplomovou práci.

### 10.1.7 Identification and Authentication Failures

Toto kritérium pohlíží na chybějící nebo nedostatečně implementované bezpečnostní prvky pro přihlašování uživatelů. Jako příklad můžeme uvést chybějící "two-factor authentication", tedy přihlašování ve dvou krocích, nebo přihlašovací formulář nezabezpečený vůči tzv. brute-force útokům.

Aplikace toto kritérium nesplňuje, protože tyto prvky nebyly zaměřením této diplomové práce a vyžadovaly by si mnoho vývojového úsilí a investice, které by ale nepřinesly nic navíc pro tuto diplomovou práci.

Tyto nedostatky nesouvisí s blockchainovou povahou aplikace a blockchainové aplikace jsou vůči těmto nedostatkům stejně zranitelné jako jiné aplikace.

### 10.1.8 Software and Data Integrity Failures

Toto kritérium je nové kritérium pro rok 2021[27]. Zabývá se nedostatečným zabezpečením tzv. "pipelines", což jsou automatizované způsoby jak nepřetržitě vydávat aktualizace k softwarovým aplikacím. Pokud tyto "pipelines"nejsou dostatečně důvěryhodné, útočník by přes ně mohl automaticky aktualizovat aplikace svým škodlivým kódem.

Vzhledem k tomu, že moje aplikace nepoužívá žádný takový způsob aktualizace softwaru, není zranitelná vůči útokům tohoto typu. Blockchainové aplikace obecně jsou stejně zranitelné jako jiné aplikace, které tyto způsoby používají.

### 10.1.9 Security Logging and Monitoring Failures

Kritérium Security Logging and Monitoring Failures se zabývá nedostatečným nebo nebezpečně implementovaným "logováním", tedy zapisováním důležitých událostí v běhu software. Porušení tohoto kritéria je například nedostatečné logování důležitých událostí v aplikaci (např. přihlášení uživatele), nebo nedostatečné informace v průběhu logování.

Volební aplikace toto kritérium nesplňuje. Nebylo v záběru aplikace vytvořit kvalitní zaznamenávání událostí vytvořených uživatelem. Blockchainové aplikace obecně jsou vůči tomuto kritériu stejně zranitelné jako jiné aplikace.

### 10.1.10 Server Side Request Forgery (SSRF)

Server Side Request Forgery je způsob útoku, kde útočník může poskytnout aplikaci vadné URL, které poté aplikace z vlastního prostředí sama zavolá, často bez obvyklých bezpečnostních procedur, které by aplikace spustila pro externího uživatele.

Volební aplikace se tomuto kritériu vyhýbá, protože neposílá URL v žádném parametru požadavků, které míří na backendový server. Pro blockchainové aplikace je toto kritérium stejně platné jako pro jiné aplikace.

### 10.1.11 Shrnutí podkapitoly

Pro většinu z 10 bodů OWASP, blockchainové aplikace jsou stejně zranitelné jako jiné aplikace, protože tyto typy útoků obvykle nesouvisejí s blockchainovou povahou aplikace. Blockchainová povaha aplikace pomáhá u pravidla **Injection**, kde útok na databázi obvykle nezpůsobí dalekosáhlé poškození vzhledem k decentralizované povaze blockchainové databáze.

## 10.2 Analýza z pohledu požadavků

V kapitole 3.1 jsem popsal důležité vlastnosti, které by volební aplikace založená na blockchainu měla mít. Znovu je tu vyjmenuji:

- **Soukromí** - Každý hlas by měl být anonymní a nevysledovatelný
- **Průkaznost** - Každý hlas může pocházet pouze od právoplatných voličů. Nikdo jiný by neměl být schopný volit.
- **Nepřepoužitelnost** - Každý volič by měl moci hlasovat pouze jednou, a duplicitní hlasy by se neměly počítat.
- **Férovost** - Nikdo by neměl mít moc získat výsledky kde nebyly započítány všechny hlasy a touto znalostí volby ovlivnit
- **Celkovost** - Pro právoplatný výsledek, všechny hlasy by měly být započítány.
- **Škálovatelnost** - Volební systém by mělo být možné použít i u voleb čítající miliony voličů.

Rád bych demonstrační volební aplikaci vytvářenou v této diplomové práci porovnal s těmito vlastnostmi a zjistil, jaké z nich aplikace plní a na jakých by se ještě dalo pracovat.

### 10.2.1 Soukromí

Tuto vlastnost dle mého názoru aplikace splňuje. Vzhledem k tomu, že v databázovém modelu aplikace není volební lístek nijak propojený s počtem hlasů u daného kandidáta (musí pouze sedět počet odhlasovaných volebních lístků k součtu hlasů všech kandidátů), není možné spojit hlasovací lístek s jakýmkoliv jednotlivým hlasem. Toto bohužel také znamená, že není možné pro uživatele spojit svůj hlas s kandidátem, kterého volil. Ví jistě pouze, že odhlasoval a který volební lístek je jeho. Toto je ovšem stejná nevýhoda, s jakou se v současnosti potýkají papírové volby - ani u nich není volič schopný po odvolení identifikovat svůj volební lístek.

### 10.2.2 Průkaznost

Tato vlastnost je dle mého názoru mimo záběr volební aplikace, kterou jsem vytvořil. Každý volič v mé aplikaci musí být přidaný s unikátním registračním číslem. Toto číslo reprezentuje údaj, pomocí kterého je možné voliče jednoznačně identifikovat, jako například občanský průkaz. Způsob, jakým způsobem bychom před přidáním voliče ověřili že je legitimní, je již na samotném administrátorovi voleb. V případě státních voleb by to mohlo být například API které zkontroluje, zda zadané číslo občanského průkazu existuje v databázi. Tento způsob by ideálně měl být transparentní a nespoléhat pouze na důvěryhodnou třetí stranu

### 10.2.3 Nepřepoužitelnost

Nepřepoužitelnost je v mé aplikaci prakticky zaručena na úrovni kontraktu. Pokud již je voličův hlasovací lístek označen jako odhlasovaný, kontrakt požadavek odmítne a hlas se nepočítá. Na aplikační úrovni není uživateli dovoleno dostat se na hlasovací obrazovku a je okamžitě přesunut na obrazovku, ze které nemůže hlasovat.

### 10.2.4 Férovost

Tuto vlastnost bohužel v současnosti aplikace nesplňuje. Kdokoli s administrátorským přístupem může získat okamžitý stav blockchainové databáze a podívat se ne průběžné výsledky voleb.

Toto by mohlo eventuálně být vyřešeno změnou architektury aplikace, například přidáním zašifrované entity která reprezentuje voličův hlas. Následně by se využilo zašifrování identifikačního údaje kandidáta (tedy případného parametru **votableitemid**) a hlasy by se počítaly až po skončení volby, místo průběžně. Současně je na aplikační úrovni bráněno přístupu uživatele na obrazovku, která zobrazuje výsledky voleb (viz kapitola 8.4).

### 10.2.5 Celkovost

Součet hlasů jednotlivých kandidátů v aplikaci při skončení volby se musí rovnat počtu odhlasovaných volebních lístků v dané volbě. Toto zaručuje celkovost výsledku voleb. Každý volič může vidět, že je jeho unikátní volební lístek započítán do volby. Stejně tak může po konci voleb vidět průběh voleb, takže je možné si jednoduše spočítat celkovost voleb.

### 10.2.6 Škálovatelnost

Ačkoli je mnou vytvořený kontrakt lépe škálovatelný, než kdyby byl postavený na ostatních blockchainových platformách (viz porovnání rychlostí v kapitole 7.1), dle mého názoru není aplikace škálovatelná na miliony voličů. Jak jsem již psal v kapitole 4.2.4, toto se jeví jako limitace současné blockchainové technologie.

Existuje ale možnost nálož na volební systém zmírnit umožněním déle trvajících voleb. V případě státních voleb by při použití internetového volebního systému odpadla nutnost provozovat velkou logistickou operaci. Toto by případně mohlo umožnit "rozprostřít"nálož na systém přes delší dobu.

## 10.3 Shrnutí kapitoly

V této kapitole jsem analyzoval vytvořenou volební aplikaci tak, že jsem jí nejprve posoudil v rámci deseti kriterií OWASP. Následně jsem aplikaci posoudil šesti kritérii, které by měla dle mého názoru volební aplikace založená na blockchainu splňovat. Z těchto vlastností jsem čtyři splnil, jednu nesplnil, a jednu částečně nesplnil. Férovost aplikace může být zajištěna při jiné architektonické koncepci aplikace. Škálovatelnost může být dle mého názoru zaručena zlepšující se technologií blockchainu, která jde v současnosti velice rychle kupředu.

# Kapitola 11

# Další možný vývoj aplikace

V této kapitole budou popsány různé směry, kterými by se aplikace mohla dál rozšiřovat či vyvíjet.

## 11.1 Férovost

Z vlastností, které jsem z pohledu volební aplikace analyzoval v kapitole 10, byla nejvíc nedostatečně splněná vlastnost průkaznost. Aby byla aplikace skutečně využitelná pro náročné případy použití, musí zabezpečit aby nikdo nemohl znát průběžné výsledky voleb, dokud volba neskončí.

Můj návrh pro implementaci této vlastnosti vypadá takto: Pro každého voliče vytvoříme v databázi entitu, která reprezentuje hlas samotný. Tato entita by neměla být propojitelná s původním voličem, ale měla by obsahovat informaci o kandidátovi, pro nějž by se mělo hlasovat. Tato informace by mohla být zašifrovaná. Po skončení voleb bychom mohli rozšifrovat tuto informaci pomocí částečných soukromých klíčů a homomorfického šifrování (viz kapitola 4.2.1).

## 11.2 Umožnění integrace s validátory identity

Pro umožnění přidání voliče do systému a zabezpečení vlastnosti, že každý volič musí být právoplatným voličem, bychom mohli nabídnout integraci programu s vlastním API, které by pro nás kontrolovalo voličovu průkaznost, když jej přidáváme do systému.

Ještě pokročilejším způsobem by bylo umožnění hromadného přidávání voliče podle již existující databáze např. občanských průkazů. Toto by mohlo značně

zrychlit proces přidávání a je pravděpodobně nutné pro použití ve volbách čítající více jak nízký počet voličů.

## 11.3 Přihlášení uživatele vlastním certifikátem

V současné chvíli se certifikát voliče ukládá v lokální paměti na aplikačním serveru. Toto není ideální, protože soukromý klíč by měl být pouze v rukou voliče a měl by ho nahrát ve chvíli, kdy se do aplikace přihlašuje. Funkce by byla podobná jako např. různé bankovní aplikace. Bohužel se mi nepodařilo stihnout tuto funkci do aplikace implementovat do data odevzdání této diplomové práce.

# Kapitola 12

# Závěr

V této diplomové práci bylo zadáním analyzovat současný stav elektronických voleb, prodiskutovat současné použití blockchainových technologiích nejen v rámci volebních aplikací, a následně vytvořit demonstrační volební aplikaci založenou na blockchainových technologiích.

Práce splnila zadání. Byla vytvořena funkční volební aplikace založená na platformě Hyperledger Fabric, kterou je případně možné použít jako základ pro vylepšené či dokonce komerční využití tohoto konceptu.

V této diplomové práci bylo nejdříve popsáno, co je blockchain, jak funguje a nejznámější platformy, které jsou na blockchainových technologiích založeny. Následně jsem popsal stav voleb v současnosti, nejprve "klasických"papírových voleb, pak elektronických voleb bez použití blockchainu, a nakonec současný stav volebních aplikací založených na blockchainu.

Poté jsem popsal analýzu požadavků na aplikaci kterou chci vytvořit, formálně jsem aplikaci naplánoval pomocí diagramů a následně jsem popsal implementaci tří částí aplikace - smart contractu, backendového API a frontendové aplikace.

Po implementaci jsem analyzoval aplikaci z pohledu přepoužitelnosti a následně jsem posoudil, zda splňuje body, které jsem si pro aplikaci vytyčil v kapitole 3.1.

Výstup této diplomové práce může dle mého názoru sloužit jako dobrý základ pro skutečně využitelnou aplikaci, která díky blockchainovým technologiím poskytuje lepší decentralizovanost a transparentnost než tradiční volební aplikace.

# Literatura


[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *www.bitcoin.org*, p. 9, October 2008.

[2] S. Sayeed and H. Marco-Gisbert, "Assessing blockchain consensus and security mechanisms against the 511788, April 2019.

[3] Cardano.org. [Online]. Available: https://cardano.org/ouroboros/

[4] "Proof-of-stake (pos)," December 2021. [Online]. Available: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/

[5] (2021, April) Hashing. [Online]. Available: https://www.techopedia.com/definition/14316/hashing-cybersecurity

[6] D. Gligoroski, "Length extension attack on narrow-pipe sha-3 candidates," pp. 5–10, January 2011.

[7] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 2901–2925, 2021. [Online]. Available: https://doi.org/10.1007/s12083-021-01127-0

[8] *FastKitten: Practical Smart Contracts on Bitcoin*, 2019.

[9] Smart contracts and chaincode. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/smartcontract/smartcontract.html

[10] [Online]. Available: https://coinmarketcap.com/currencies/bitcoin/

[11] C. Criddle. (2021, February) Bitcoin consumes 'more electricity than argentina'. [Online]. Available: https://www.bbc.com/news/technology-56012952

[12] E. Androulaki, A. Barger, V. Bortnikov, and Cachin, "Hyperledger fabric: A distributed operating system for permissioned blockchains," 01 2018.

[13] R. Brown, J. Carlyle, I. Grigg, and M. Hearn, "Corda: An introduction," 09 2016.

[14] *Advances in Cryptology: Proceedings of CRYPTO '83*. New York: Plenum, 1983. [Online]. Available: http://link.springer.com/book/10.1007/978-1-4684-4730-9

[15] J. Seberry and Y. Zheng, Eds., *A practical secret voting scheme for large scale elections*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993.

[16] *E-voting System Using Homomorphic Encryption and Blockchain Technology to Encrypt Voter Data*, 11 2021.

[17] (2010) Mix network. [Online]. Available: https://cryptography.fandom.com/wiki/Mix_network

[18] J. Seberry and Y. Zheng, Eds., *A practical secret voting scheme for large scale elections*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993.

[19] Y. M. Chee, Z. Guo, S. Ling, F. Shao, Y. Tang, H. Wang, and C. Xing, Eds., *Secret-Sharing Schemes: A Survey*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[20] D. Schatsky. (2018, September) Blockchain and the five vectors of progress. [Online]. Available: https://www2.deloitte.com/us/en/insights/focus/signals-for-strategists/value-of-blockchain-applications-interoperability.html

[21] (2019) Voting results in detail, estonian elections. [Online]. Available: https://rk2019.valimised.ee/en/voting-result/voting-result-main.html

[22] W. Drechsler and Ü. Madise, *Electronic Voting in Estonia*. 21st ACM Conference on Computer and Communications Security, November 2004, pp. 97–108.

[23] L. Alonso, M. Gascó, D. Marcos del Blanco, j.-a. Hermida, J. Barrat, and H. Alaiz Moreton, "E-voting system evaluation based on the council of europe recommendations: Helios voting," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, pp. 1–1, 11 2018.

[24] J. David and D. Buell, "What we don't know about the voatz blockchain internet voting system," pp. 1–10, May 2019.

[25] IBM, "Ibm blockchain platform extension." [Online]. Available: https://marketplace.visualstudio.com/items?itemName=IBMBlockchain.ibm-blockchain-platform

[26] "A04:2021 – insecure design." [Online]. Available: https://owasp.org/Top10/A04_2021-Insecure_Design/

[27] "A08:2021 – software and data integrity failures." [Online]. Available: https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/

# Příloha A

## Slovník pojmů

**Databáze** - struktura, která nám umožňuje efektivně ukládat data a číst uložená data.

**Blockchain** - struktura databáze, kde jsou data ukládaná do rozdělených "bloků". Každý blok je spojený s předchozím blokem skrze jeho hash.

**Hashing, hashování, hash** - viz kapitola 2.4.

**Smart contract** - viz kapitola 2.5.

**Případ použití** - Okolnosti, které nám umožňují něco použít efektivním způsobem.

**Blockchain wallet** - struktura, která v rámci blockchainové platformy drží citlivé informace o uživateli, jako například soukromý klíč nebo peníze. Používá se jako identifikátor uživatele.

**Algoritmus konsenzu** - viz kapitola 2.3.

**Peer-to-Peer** - způsob komunikace na internetu, kde jednotlivé uzly komunikují přímo mezi sebou, bez nutnosti centralizovaného serveru.

**Framework** - Soubor funkcí či vzorců, které poskytují základ při vytváření softwaru.

**Open-source** - software, jehož zdrojový kód je veřejně přístupný na internetu.

**Frontend** - Prezentační část aplikace, rozhraní, se kterým uživatel obvykle interaguje na webu.

**Backend** - Veškerá logika potřebná pro chod aplikací, ke které uživatel nemá přístup. Zašťiťuje například komunikaci s databází nebo náročné výpočty.

**TypeScript** - nadstavba nad jazykem JavaScript, která ho doplňuje o typování proměnných a další funkce. `https://www.typescriptlang.org/`

**WebAssembly** - binární formát instrukcí, který může interpretovat webový prohlížeč. `https://webassembly.org/`

**Verzovací systém** - způsob, kterým sledujeme změny v kódu projektu. Nejpopulárnější verzovací systém na světě je Git.

**Monorepo** - způsob organizace projektů ve verzovacích systémech, kde jeden repozitář obsahuje více oddělených projektů.