VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DECENTRALIZOVANÁ APLIKACE PRO QUESTION/ANSWER SEKCI VYUŽÍVAJÍCÍ BLOCKCHAIN

DECENTRALIZED APPLICATION FOR QUESTION/ANSWER SESSION BASED ON BLOCKCHAIN

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAKUB ČUHANIČ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. IVAN HOMOLIAK, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

148144

Ústav:	Ústav inteligentních systémů (UITS)
Student:	**Čuhanič Jakub**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Decentralizovaná aplikace pro Question/Answer sekci využívající blockchain**
Kategorie:	Informační systémy
Akademický rok:	2022/23

Zadání:

1. Seznamte se s principy blockchainu, smart kontraktů a decentralizovaných aplikací (DAPPs).
2. Nastudujte si programovací prostředí pro vývoj mobilních aplikací.
3. Navrhněte systém a protokol vhodný pro decentralizované řízení Q/A sekcí; zaměřte se na motivační model, více relací, správu identit.
4. Vytvořte DAPP pro smartphone s využitím navrhovaného protokolu.
5. Vyhodnoťte aplikaci a diskutujte o jejích výhodách a nevýhodách, stejně jako o možných rozšířeních.

Literatura:

- Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." *Ethereum project yellow paper* 151.2014 (2014): 1-32.

Při obhajobě semestrální části projektu je požadováno:
Items 1 to 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Homoliak Ivan, Ing., Ph.D.**
Vedoucí ústavu:	Hanáček Petr, doc. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	3.1.2023

Abstrakt

Tato práce se zaměřuje na vytvoření decentralizované mobilní aplikace, která nabízí Q&A sekci. Problém transparentnosti procesu dotazování v Q&A sekcích, cenzury otázek a spamu je vyřešen pomocí technologie blockchainu a jeho napojení na systém pro správu identit. Výsledné decentralizované aplikace jsou řešením na všechny tyto problémy. Jejich využití může být kdekoliv, kde je potřeba Q&A prostor, především ve veřejnoprávních médiích nebo na internetu. Díky všem vlastnostem aplikací vyplývajících z blockchainu by se mělo dosáhnout stavu, ve kterém nebudou moci moderátoři relací manipulovat ani cenzurovat uživatele a jejich otázky.

Abstract

This work focuses on creating a decentralized mobile application that offers a Q&A section. The problem of transparency process of the Q&A section, question censorship and spamming is solved by using blockchain technology and linking it to an identity management system. The resulting decentralized applications are the solution to all these problems. Their use can be anywhere where Q&A space is needed, especially in public media or on the Internet. Thanks to all the features of the applications resulting from the blockchain, a state should be achieved in which session moderators will not be able to manipulate or censor users and their questions.

Klíčová slova

decentralizovaná aplikace, blockchain, smart kontrakt, Solidity, React Native, Q&A, cenzura

Keywords

decentralized application, blockchain, smart contract, Solidity, React Native, Q&A, censorship

Citace

ČUHANIČ, Jakub. *Decentralizovaná aplikace pro Question/Answer sekci využívající blockchain.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ivan Homoliak, Ph.D.

Decentralizovaná aplikace pro Question/Answer sekci využívající blockchain

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Ivana Homoliaka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jakub Čuhanič
8. května 2023

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Ivanu Homoliakovi, Ph.D. za odborné vedení práce, pravidelné konzultace, cenné rady, trpělivost a uchotu.

Obsah

Kapitola 1

Úvod

Dnes, v 21. století, máme na výběr tolik informací, jako jsme ještě nikdy neměli. Poprvé jsme nuceni vybírat si informace a kontrolovat, zda jsou pravdivé a objektivní nebo naopak lživé a zavádějící. Jak víme, že to, co nám média prezentují, je objektivní? Není možné, že nám ukazují jen část pravdy, tu část, kterou chtějí?

Především v televizních a rozhlasových médiích jsou populární relace (v zahraničí přezdívané Q&A), ve kterých se diváci dotazují moderátora, známých osobností či odborníků na různé otázky, které chtějí zodpovědět. Jsme schopni zjistit, zda otázky, které jsou položeny skrze různé aplikace, nebyly nějak zmanipulovány nebo cenzurovány? Jednotlivé dotazy tak mohly být neoprávněně pozměněny, skryty či odstraněny. Spolu s tím je možné, že všechny otázky byly položeny pouze produkčním týmem nebo jednotlivci, kteří se pod rouškou internetu vydávají za „běžné“ publikum.

Smyslem této práce je vytvořit decentralizovanou mobilní aplikaci, ve které bude uživatel moci pokládat jednotlivé otázky. Přínos aplikace bude v tom, že díky použití technologie zvané blockchain nedovolí žádné straně cenzurovat otázky ani s nimi manipulovat. Druhou, neméně cennou předností bude, že uživatelé, kteří se budou chtít zúčastnit relace, si musí ověřit svoji identitu. To zamezí umělému navyšování počtu otázek a zajistí tak přirozený průběh relace. Využití aplikace by mohlo nastat především ve veřejnoprávních médiích, která by měla být objektivní a transparentní, ale i kdekoliv na internetu.

1.1 Struktura práce

V kapitole 2 jsou vysvětleny potřebné pojmy spojené s blockchainem, které jsou nutné k pochopení problematiky. Kapitola 3 stručně popisuje programovací prostředí pro vývoj mobilních aplikací. Dále si ve 4. kapitole představíme použité technologie potřebné k vytvoření práce. 5. kapitola vysvětluje přínos práce a porovnává již existující aplikace. V kapitole 6 si popíšeme použité smart kontrakty a aplikace. Nakonec v 7. kapitole dojde na popis testování aplikací, odchýlení se od zadání práce či analýze nevhodného chování aktérů aplikací.

Kapitola 2

Potřebné znalosti k pochopení decentralizovaných aplikací

Přestože je tu blockchain již déle než 10 let, nacházejí se mezi informatiky tací, kteří o něm moc nevědí. Pro vytvoření decentralizované aplikace je nutné si vysvětlit základní pojmy, které nám pomohou chápat problematiku blockchainu a decentralizovaných aplikací. V případě, že se kdekoliv v této práci budeme bavit o adrese nebo transakci, budou vždy tyto pojmy znamenat blockchainovou adresu a blockchainovou transakci.

Na následujících stránkách si proto vysvětlíme pojmy jako blockchain, konsenzus, smart kontrakt nebo decentralizovaná aplikace a pojmy s nimi spojené.

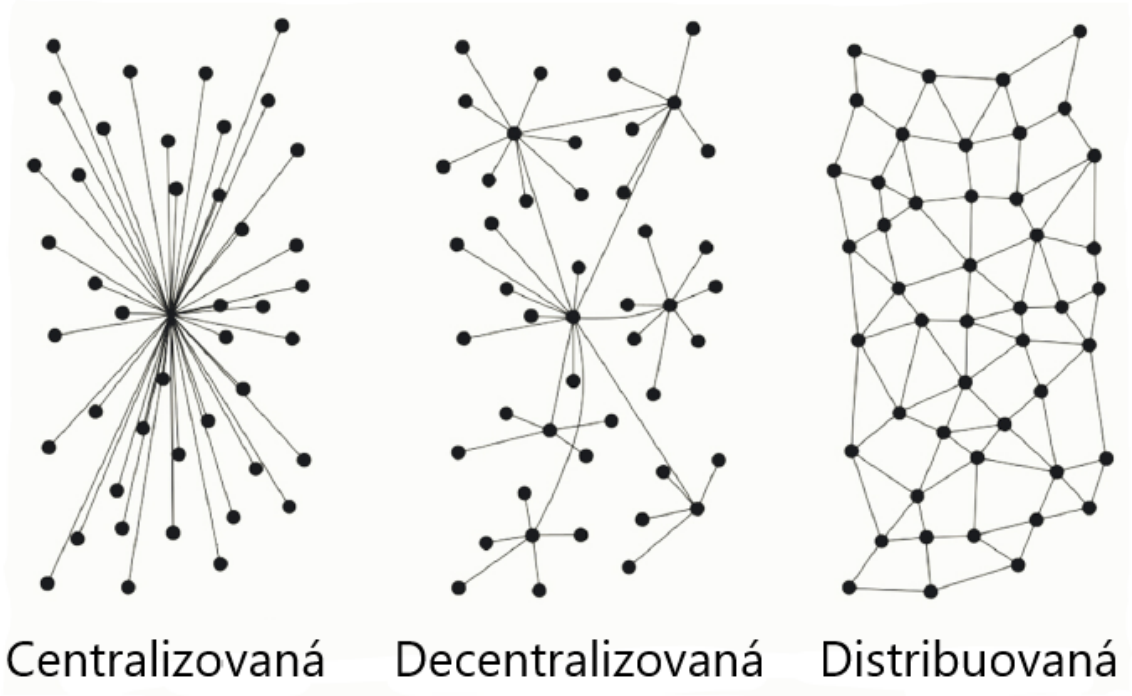
2.1 Blockchain a jeho části

Historie blockchainu sahá až do roku 2008, kdy neznámý autor (případně skupina autorů) pod pseudonymem Satoshi Nakamoto sepsal první návrh fungování blockchainu, ze kterého vyšla první kryptoměna zvaná Bitcoin [47]. Koncept blockchainu a Bitcoinu byl reakcí na nedostatky tradičního bankovního systému. Pro potřeby této práce si definujeme blockchain jako distribuovanou decentralizovanou databázi, která ukládá veškeré informace o transakcích, které proběhly mezi uživateli systému [8]. Na obrázku 2.1 jsou znázorněny rozdílné síťové topologie, ze kterých blockchain vychází. Mezi nesporné výhody decentralizovaného blockchainu patří neschopnost data smazat nebo je jakkoli upravit. Díky tomu se objem dat postupem času může pouze navyšovat.

2.1.1 Základní stavební kámen blockchainu: Blok a jeho vlastnosti

V terminologii blockchainu si pod pojmem blok představme datovou strukturu. Dělí se na 2 hlavní části, hlavičku a tělo [38]. Mezi součásti hlavičky patří hash hlavičky předchozího bloku, Merkle root (hash všech transakcí zahrnutých v bloku, česky kořenový uzel), časové razítko a hodnota nonce. V těle se nachází seznam transakcí včetně jejich dat. Části bloku jsou znázorněny na obrázku 2.2. Hash hlavičky nám udává jednoznačný identifikátor bloku, který se uloží do následujícího bloku. Tímto způsobem se bloky utvářejí v jeden dlouhý řetězec (blockchain). Časové razítko symbolizuje čas zařazení bloku do řetězce. Nonce je číslo, které se přidává k datům v bloku a slouží k tomu, aby se vypočítal hash bloku [48].

V případě, že by v bloku proběhla i byť nepatrná změna, narušila by celý řetězec, protože by došlo k přepočítání hashe hlavičky. Tento mechanismus zajišťuje, že bloky v blockchainu nelze jednoduše modifikovat bez povšimnutí, což zvyšuje důvěru a bezpečnost této tech-



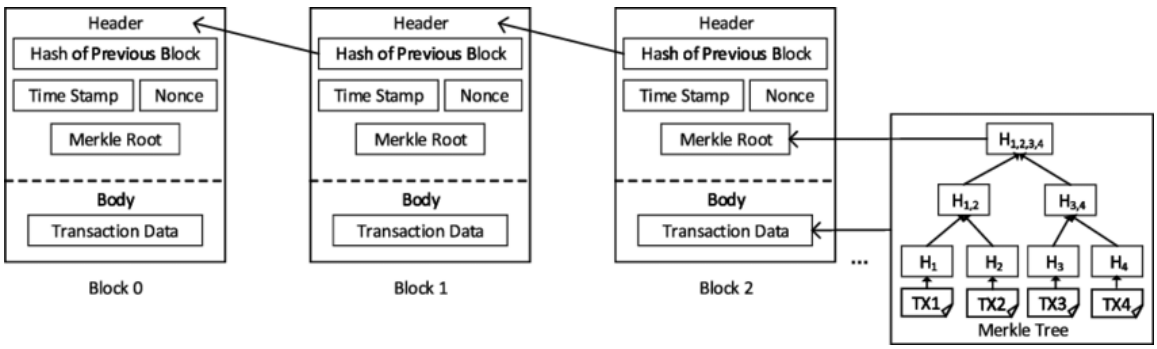
Obrázek 2.1: Typy síťových topologií [52].

nologie. Nutno ještě podotknout, že v odlišných implementacích blockchainu se struktura bloku může lišit.

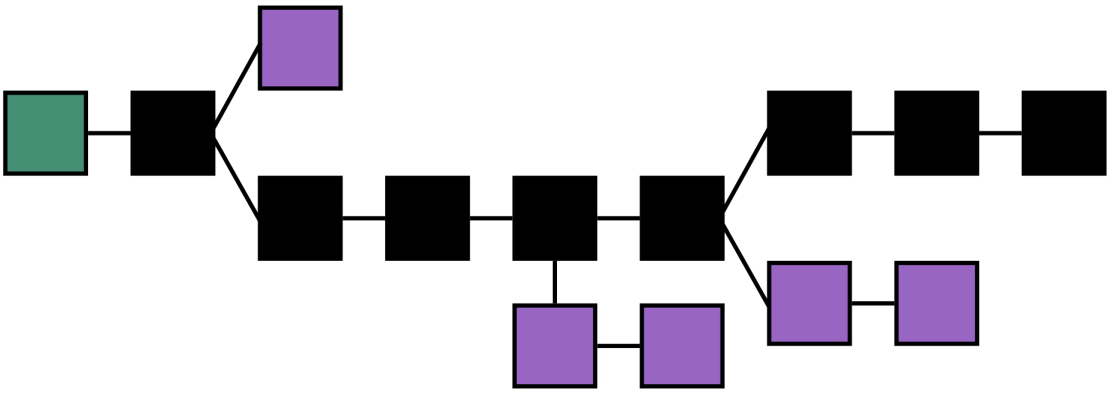
2.1.2 Transakce v blockchainu

Významem blockchainových transakcí je převod prostředků mezi adresami v síti [8]. Nicméně to není jejich jedinou úlohou. V závislosti na implementaci blockchainu mohou zajišťovat i uložení trvalých informací do blockchainu, protože odesílatel je schopen do nich vložit i určitý objem metadat. V tomto ohledu se podobají tedy databázovým transakcím, které mění uložená data databáze. Zde se bavíme o případě, kde databází je blockchain.

Samotné transakce však nejsou uskutečněny okamžitě, nýbrž se nejprve slučují do tzv. mempoolů. Každý těžař (pojem těžař bude vysvětlen dále) má vlastní mempool, ze kte-



Obrázek 2.2: Schéma blockchainu [37].



Obrázek 2.3: Řetězec schválených bloků [60].

rého vybírá transakce do bloku a kde uspořádává transakce podle výše poplatku, který je v konečném pojetí jeho odměnou. V aktivních špičkách sítě je nutné zaplatit buď vyšší poplatek nebo počkat, až těžař vybere transakci a zařadí ji do bloku. Vše je samožřejmě podmíněno škálovatelností sítě a dalšími parametry, které ovlivňují množství a rychlost zpracovaných transakcí. Ve chvíli, kdy je transakce zapsána do bloku a tudíž se provedla, je nutné počkat na určitý počet potvrzení, tzn. kolik bloků bylo po zveřejnění transakce zapsáno do blockchainu (doporučený počet potvrzení bloku v Ethereu je 12, což trvá zhruba 3 minuty). Tato vlastnost se nazývá finalita bloku. Dlouhý čas do finality bloku může přinést náhodné forky, které vytvářejí paralelní a nekonzistentní zobrazení blockchainu [31].

V případě, že uživatel nepočká dostatečně dlouho, může se jeho transakce prohlásit za neplatnou. Na obrázku 2.3 vidíme černě nejdelší řetězec bloků a modře dočasné paralelní větvě, které se mohly jevit v danou chvíli jako nejdelší. Mohlo by se totiž stát, že se najde jiný a delší řetězec bloků, který uznají uzly v síti jako hlavní, v němž by se daná transakce nemusela nacházet a tudíž by byla neplatná [25]. Existují ale blockchainy, jejichž finalita bloků je velice rychlá a zmíněné problémy zde nenastávají.

2.1.3 Uzel v blockchainové síti

Uzel (angl. node) se primárně podílí na konsenzu blockchainu (viz sekce 2.2) a sekundárně slouží jako uložiště informací blockchainu. V uzlech jsou uloženy částečné nebo celé řetězce bloků (kopie blockchainu). Mohli bychom je přirovnat k serverům, které taktéž ukládají potřebné informace pro uživatele. Mezi jejich další činnost lze zařadit legitimaci bloků s transakcemi či zasílání nových dat pomocí broadcastu ostatním uzlům [1].

Není ale uzel jako uzel a proto je dobré poznamenat, v čem se odlišují. Opět jsou rozdílné podle implementace blockchainu a dělí se na 2 základní typy - full node a light node. Full node je uzel, který v sobě uchovává celou kopii blockchainu a je zpravidla odpovědný za ověřování transakcí a bloků v blockchainové síti. Jeho podtypem je konsenzuální uzel (angl. konsenzus node), který odpovídá za těžbu nových bloků a jejich přidávání do blockchainu. Light node v sobě uchovává pouze část blockchainu a zpravidla se jedná o hlavičky bloků [53].

2.2 Konsenzus v blockchainu: Zajištění konzistence a integrity sítě

K čemu je dobré se v blockchainu shodnout? Abychom zajistili jediný zdroj pravdy v celé blockchainové síti a aby všechny uzly měly dostupné stejné a pravdivé informace. Výstupem této shody je blok, který bude zařazen do blockchainu. Tento proces se nazývá konsenzus. Díky němu se uživatelé sítě blockchainu dohodnou na tom, jaký blok vybrat i přes to, že si vzájemně nedůvěřují [36]. Za tímto procesem stojí konsenzuální uzly, které za vidinou odměny v podobě nativní kryptoměny a poplatků přidávají do dané sítě bloky. Aby vůbec uzly mohly bloky přiřadit do blockchainu, musí vložit určitou zálohu, nejčastěji se jedná o výpočetní sílu nebo nativní měnu sítě. Tato záloha by měla odradit uzly od nečestného chování, jelikož by jim při jejich prohřešku nebyla vrácena. Konsenzuální uzly a způsoby, jak konsenzu dosáhnout, se napříč blockchainovými sítěmi liší. Následující podkapitoly obsahují nejznámější typy konsenzuálních protokolů dohody [10].

2.2.1 Proof of work

Anglicky proof of work (zkr. PoW), česky důkaz práce, je první a nejstarší z možností, jak dojít ke konsenzu v blockchainové síti. Způsobem, kterým docílit shody mezi všemi uzly v síti je obětování značného množství výpočetní síly k vyřešení „kryptografického puzzle“ [22]. Těžař (angl. miner), jakožto osoba (případně skupina osob), která se snaží toto puzzle řešit, musí najít na základě konkrétních vstupů předem definovaný hash při minimální odchylce. Svoji odpověď následně odešle do sítě, kde poté ostatní uživatelé snadno ověří řešení a pokud je opravdu správné, nálezce získá odměnu. Složitost této činnosti závisí na celkové výpočetní síle všech těžařů, tzv. hashrate [55]. Při použití PoW je vysoká spotřeba elektrické energie, za což je tento algoritmus mnohdy kritizován. Jeho hlavním konkurentem je Proof of Stake (zkr. PoS), který byl z důvodu negativní spotřeby energie PoW algoritmu vytvořen.

2.2.2 Proof of stake

Principem proof of stake (česky důkaz podílu) je ověřování blokových transakcí na základě počtu „investovaných“ mincí [23]. Tento mechanismus motivuje uživatele k držení a použití mincí jako zástavy (tzv. stake) za vidinou zisku transakčních poplatků. Jednotliví validátoři (uživatelé, kteří mají zastavené mince) jsou vybíráni náhodně, kde pravděpodobnost jejich zvolení je přímo úměrná počtu vsazených mincí. Bloky jsou validovány více validátory a po ověření správnosti bloku je samotný blok finalizován a uzavřen [55].

Jednotlivé blockchainové sítě mohou mít různě implementovaný PoS, který se pro běžné uživatele liší primárně v minimálním možném počtu mincí, které musí být zastaveny, aby mohli obdržet odměnu za validaci transakcí. Např. u sítě Ethereum je minimální vsazené množství 32 ETH [50], což aktuálně (ke dni 21. 3. 2023) činí zhruba 1,25 milionu Kč [14]. Samotný PoS by měl snížit oproti PoW riziko centralizace a nabízet validátorům lepší vstupní podmínky, nicméně nedosažitelné vstupy k decentralizaci nepřispívají. Pro mnohé uživatele není možné investovat takový obnos peněz a proto byly vytvořeny tzv. staking pooly, do kterých uživatelé delegují jen určitou část potřebnou pro stake [19]. Odměna je následně počítána podle množství investovaných prostředků. V případě PoW je také typické se sdružovat, a to do tzv. mining poolů.

2.3 Definice a vlastnosti smart kontraktů v blockchainu

Prvotní myšlenka implementace smart kontraktů (angl. smart contract) vznikla v roce 1994 ústy Nicka Szaba. Tou bylo provádět automatizované procesy bez nutnosti důvěry ve třetí stranu [65]. „Je zaručeno, že smart kontrakty budou probíhat předem definovaným a deterministickým způsobem, bez zásahů žádné konkrétní třetí strany“ [57]. Samotnou definici můžeme tedy popsat následovně: Smart kontrakt je kód nasazený v prostředí blockchainu, který je vyvolán blockchainovou transakcí při splnění určitých podmínek a způsobuje změnu stavu blockchainu nebo čte jeho hodnoty [44]. Při vytváření smart kontraktu bychom měli dbát na jeho maximální optimalizaci, protože každá instrukce stojí určitý poplatek v síti, tzv. gas. Uveďme si několik výhod, které smart kontrakty skýtají:

- Vyšší efektivitu kontraktů, a to pomocí kontroly kontraktu uživateli, který je kdykoliv dostupný a zároveň nemožnosti zanést do nich chyby prostřednictvím jejich ručního přepisování.
- Odstranění nutné důvěry mezi smluvními stranami, jelikož je kontrakt proveden na základě předem stanovených podmínek bez potřeby prostředníka, s čímž se pojí i nižší náklady. I přesto je někdy nutné pomocí centralizovaného prostředníka dodat data do blockchainu.
- Nezávislost na centrální autoritě.

Bohužel lze najít i několik nevýhod, se kterými musíme počítat:

- Nutné zabezpečení kódu a ověření jeho správnosti.
- Nevratnost kontraktů. V případě, že nastane v kódu chyba nebo jiný omyl, nelze se vrátit do stavu před jeho vykonáním.
- Oproti běžným smlouvám nelze smart kontrakty soudně vymáhat.
- Nemožnost úpravy tradičního smart kontraktu po jeho nasazení. To však neplatí v rámci „upgradeable contracts“, které lze upravit i beze změny jejich adresy [45]. Je proto nutné, aby se změnou souhlasily pomocí hlasování všechny zúčastněné strany a nedošlo tak k nečestnému jednání.

Někomu nevýhody mohou připadat jako zanedbatelné, avšak opak je pravdou. V případě širší adopce je nezbytné všechny nedokonalosti smart kontraktů důkladně zvážit a být velice obezřetný, protože by mohly nadělat více trápení než užitku. Většina implementací smart kontraktů v právní oblasti stále ještě není ani zdaleka přijata nebo dokonce není nijak užitečná. Jedním z důvodů, kromě toho, že je to poměrně pořád nové téma, je způsob přenosu citlivých a relevantních informací do smart kontraktů. Proto se občas musíme spolehnout na takzvané „oracles“ – služby důvěryhodných třetích stran, jejichž prostřednictvím jsou příslušné vstupy do kontraktů poskytovány. Blockchain sám o sobě neumí používat jiná data, než jaká jsou v něm uložená [7].

Ethereum Virtual Machine: principy a architektura

Jedná se o nejznámější a nejrozšířenější platformu pro nasazení smart kontraktů. Ethereum Virtual Machine (zkr. EVM) je stavový stroj, který „zpracovává výpočty a stavy kontraktů

DATA	OPCODE	GAS	Description
0x50	POP	2	Remove item from stack.
0x51	MLOAD	3	Load word from memory.
0x52	MSTORE	3	Save word to memory.
0x53	MSTORE8	3	Save byte to memory.
0x54	SLOAD	800	Load word from storage.
0x55	SSTORE	20000	Save word to storage.
0x56	JUMP	8	Alter the program counter.
0x57	JUMP1	10	Conditionally alter the program counter.

Obrázek 2.4: Cena jednotlivých instrukcí [62].

a je postaven na zásobníkovém jazyce s předdefinovanou sadou instrukcí a odpovídajících argumentů“ [61]. Dílčí smart kontrakty se však nepodobají strukturou nízkoúrovňovým programovacím jazykům, které používají jednoduché sady instrukcí, ale naopak vysokoúrovňovým jazykům používající vlastnosti jako polymorfismus nebo dědičnost. Jejich kód se až při překladu právě pomocí EVM překládá na jednoduché instrukce. Namísto specifických programovacích jazyků by bylo možné psát kód v instrukcích, avšak to by bylo zbytečně složité.

Jak je popsáno v předchozí kapitole, jednotlivé instrukce mají svou cenu a stojí určitý poplatek. Cena některých instrukcí je znázorněna na obrázku 2.4. Účelem těchto poplatků bylo, aby autoři byli nuceni psát optimální kódy a zamezilo se tak zbytečně dlouhým kódům, díky kterým by se mohl virtuální stroj přetížit [61]. Můžeme říci, že EVM je prostředím, ve kterém existují všechny účty a chytré smlouvy Etherea [35]. Tento virtuální stroj rozhodně není jediným napříč blockchainy. Mnoho dalších sítí má své stroje, jako jsou například NEO, Cardano, EOS atd. Ovšem nezáleží jen na výběru platformy, ale i na jazyce smart kontraktů, který ovlivní samotný vývoj decentralizované aplikace. Podívejme se tedy, s čím lze EVM využít.

2.3.1 Solidity

Programovací jazyk Solidity slouží pro tvorbu smart kontraktů na platformě Ethereum [54]. V blockchainové komunitě (a hlavně té okolo Etherea) patří mezi uživatelsky nejoblíbenější a nejrychleji rostoucí jazyky pro vytváření smart kontraktů. Jeho vývoj závisí na komunitě, která ho posouvá stále kupředu. Solidity je vysokoúrovňový, staticky typovaný jazyk a jeho syntaxe se podobá stálicím jako C++, Python nebo JavaScript. Podobně i Solidity nabízí vlastnosti jako polymorfismus, dědičnost či použití externích knihoven.

Dílčí smart kontrakty se podobají třídám u objektově orientovaných jazyků a vyžadují přítomnost konstruktoru, který je vykonán při nasazení smlouvy do blockchainu. Jednotlivé části kódu je nutné zapisovat do funkcí, které jsou vyvolány uživateli sítě [61]. Solidity nabízí použití speciálních proměnných v globálním prostředí, které slouží k poskytnutí informací o blockchainu nebo samotných transakcích. Týká se to proměnných `block`, `msg` nebo `tx` [12]. `Block` zahrnuje informace o bloku, kupříkladu číslo aktuálního bloku či jeho časové razítko, `msg` poskytuje informace například o odesílateli transakce či hodnotě wei, která je zaslána transakcí a `tx` zase třeba výši poplatku za transakci (tzv. gas). Další speciální vlastností Solidity je nativní funkce `require`, která slouží k ověření podmínek ve funkcích.

```
function addUsersToRelation(address[] memory usersList, uint relationID) public {
    require(msg.sender == getRelation(relationID).author, "You must be creator of relation to add users");
    require(false == getRelation(relationID).isPublic, "You can add users only to private relation");
    require(getRelation(relationID).votePhase == false, "You can not add users in vote phase");
    for(uint i = 0; i < usersList.length; i++) {
        relations[relationID].privateList.push(usersList[i]);
    }
}
```

Obrázek 2.5: Funkce `addUsersToRelation` z vytvořeného smart kontraktu `Queans` sloužící k přídaní uživatelů do soukromé relace napsaná v jazyce Solidity.

```
@public
def addUsersToRelation(usersList: address[100], relationID: uint256):
    assert msg.sender == getRelation(relationID).author, "You must be creator of relation to add users"
    assert not getRelation(relationID).isPublic, "You can add users only to private relation"
    assert not getRelation(relationID).votePhase, "You can not add users in vote phase"
    for i in range(len(usersList)):
        relations[relationID].privateList.push(usersList[i])
```

Obrázek 2.6: Funkce z obrázku 2.5 napsaná v jazyce Vyper.

Pokud není podmínka splněna, funkce se ukončí s chybovou hláškou nadefinovanou v těle `require`. Na obrázku 2.5 je znázorněno použití speciální proměnné `msg` i funkce `require`. Solidity dále nabízí i datovou strukturu `mapping`, která se velice podobá asociativnímu poli nebo slovníku u jiných jazyků.

2.3.2 Vyper

Obdobně jako Solidity je Vyper jazyk orientovaný na vytváření smart kontraktů, jejichž nasazení probíhá na EVM. Samotný Vyper je silně typovaný programovací jazyk, mezi jehož hlavní cíle se řadí následující vlastnosti [58]:

- Bezpečnost – Vyper by měl být schopen přirozeně produkovat bezpečné smart kontrakty.
- Jednoduchost – Implementace jazyka a kompilátoru by měla být jednoduchá.
- Auditovatelnost – Kód psaný ve Vyperu má být pro člověka co nejčitelnější a zároveň musí být složité napsat zavádějící kód.

Vyper je natolik jednoduchým jazykem, že zde nenajdeme dědičnost, rekurzivní volání ani přetěžování operátorů či funkcí. Počet funkcí je záměrně omezen na co nejmenší počet, díky čemuž jsou kontrakty bezpečnější a snáze kontrolovatelné [59]. Samotný jazyk nevznikl jako konkurent Solidity, ale má ho doplňovat svou zvýšenou bezpečností [3].

2.3.3 Scilla

Scilla je staticky typovaný jazyk pro tvorbu smart kontraktů. Obdobně jako Vyper se zaměřuje na jejich bezpečnost, zejména v oblasti DeFi (decentralizovaných financí) [64]. Tento jazyk je vědecky ověřován a zvýšení bezpečnosti by měl zajistit skenr, který pomocí formální verifikace a statické analýzy dokáže hledat potenciální chyby, bezpečnostní problémy a další nedostatky v kódu. Scilla byl navržen tak, aby byl jednoduchý na použití, ale zároveň dostatečně silný na to, aby umožnil programátorům psát komplexní smart kontrakty [63].

```
transition addUsersToRelation(usersList : List (ByStr20), relationID : Uint256) {
    assert(_sender() == getRelation(relationID).author, "You must be creator of relation to add users");
    assert(not getRelation(relationID).isPublic, "You can add users only to private relation");
    assert(not getRelation(relationID).votePhase, "You can not add users in vote phase");
    let i: Uint256 = Uint256 0;
    let len: Uint256 = List.length(usersList);
    for (; i < len; i++) {
        let user: ByStr20 = List.get(usersList, i);
        Relations[relationID].privateList.push_back(user);
    }
}
```

Obrázek 2.7: Funkce z obrázku 2.5 napsaná v jazyce Scilla.

2.4 Decentralizované aplikace a jejich přínosy

Decentralizované aplikace (zkr. dAPPs) jsou aplikacemi běžícími na blockchainu. Oproti centralizovaným aplikacím běží v uživatelském prohlížeči a nejsou tudíž závislé na centralizovaném serveru, ze kterého fungují centralizované aplikace. Druhým rozdílem je, že využívají blockchain jako databázi, respektive stavy jednotlivých kontraktů a jejich proměnných. Fungují jako rozhraní, skrze které uživatel dokáže měnit stav sítě pomocí vyvolaných transakcí ze smart kontraktů, které má blockchain v sobě uložen. Ideální blockchainová aplikace by měla být provozuschopná bez jakéhokoli lidského zásahu [11]. Zde uvedu 4 body, které by měly ziskové dAPPs splňovat [51]:

- Open source zdrojový kód – Každá decentralizovaná aplikace by měla mít volně dostupný kód. Uživatelé, kteří chtějí využít jejích služeb by měli mít možnost si zkontrolovat zdrojový kód a sami vyhodnotit, zda nemá aplikace chyby nebo pochybná místa v kódu. Open source kód může být ale dvousečnou zbraní a v konkurenčním světě poskytne konkurenci velkolepý užitek.

- Interní měna – Jakým způsobem si vývojář dAPP vydělá peníze za provoz svojí aplikace? Jestliže jeho aplikace má open source kód a její provoz je zpoplatněn, může ji kdokoliv odcizit, mírně poupravit a nabídnout zdarma. Proto se vývojáři uchylují k vytvoření interního tokenu (měny), jehož cena roste v závislosti na použití aplikace. Následně pak svůj podíl tokenů mohou prodat a těšit se ze zisku.

- Decentralizovaný koncenzus – Je nespornou výhodou, kterou blockchain nabízí. Pro všechny aplikace, které vyžadují shodu všech uživatelů na určitých problémech, je blockchain správnou cestou. Nejedná se o to, kdo kupříkladu finalizoval poslední blok v síti, ale o to, že se všichni shodnou na tom, že se to konkrétní osobě povedlo.

- Žádný centrální bod selhání – Blockchainová síť obsahuje nespočet nezávislých uzlů, které udržují síť při životě. V případě, že některý z nich selže, ostatní uzly to nijak neohrozí a jsou dále provozuschopné. Decentralizované aplikace tedy nelze jednoduše vypnout, protože jejich existence závisí na již zmíněných uzlech v síti a ne na konkrétním centrálním serveru.

V čem se decentralizovaná aplikace liší od té běžné, centralizované? Proč by alespoň v nějakých oblastech měly dAPPs převládat? Pokusím se nastínit konkrétní výhody a nevýhody, které mohou určit, zda by vývojář měl vůbec nad těmito možnostmi uvažovat

a jestli mu mohou faktické rozdíly usnadnit nebo naopak zkomplikovat jeho práci [29].

Výhody dAPPs:

- Smart kontrakty by měly nabídnout ochranu soukromí uživatelů. Kontrakty spolu mohou komunikovat bez nutnosti napojení na centralizovanou stranu, která by byla schopna zneužít citlivá data uživatelů.
- Síť není závislá na centrálním serveru a tudíž stačí jediný dostupný uzel, aby nedošlo k výpadku aplikace, přestože její výkon může být v danou chvíli snížen.
- Nepožadují vysoké náklady na instalaci serveru, včetně údržby a jeho správy a v konečném důsledku jsou náklady na provoz minimální.
- Neexistuje zde žádný centrální bod, který by mohl ohrozit bezpečnost aplikace.
- Decentralizované aplikace jsou odolné vůči cenzuře. Tato výhoda bude pro naši aplikaci zásadní.
- Poskytují podnikatelskou příležitost pro nové uživatele. Neexistuje zde žádný subjekt, který by blokoval vstup na trh či jinak kontroloval uživatele.

Nevýhody dAPPs:

- Nedostatečně prozkoumaná technologie, vývoj je v počáteční fázi a netuší se, jak by aplikace fungovaly při jejich vysokém zatížení či zatížení sítě.
- Náročnost provádění úprav zdrojového kódu za účelem vylepšení nebo opravy chyb. Samotné aplikace se musí po úpravě znovu nasadit na blockchain. Výjimku tvoří dAPPs s využitím „upgradeable contracts“. Tento bod souvisí s nevýhodami smart kontraktů ze sekce 2.3.
- Při špatně zvolené síti může nastat v dAPP problém s dlouhou dobou potvrzení transakce, tedy čekáním uživatelů na vykonání určité změny v aplikaci.
- Přestože se snažíme vyhnout se opakovanému zanesení chyb do smart kontraktů, jsme pouze lidé, a je téměř jisté, že v kódu budou chyby obsaženy.
- Dříve či později se zveřejní aplikace, které nám svoji podstatou mohou škodit. Díky jejich decentralizaci je nebudeme schopni dobře omezit.

Jak jsme si zmínili, dAPPs mohou mít své plusy i mínusy. Jejich osvojení širokou veřejností však ještě chvíli potrvá. Nejdříve bude zapotřebí usnadnit vývojářům vývoj aplikací, ale i přiblížení nových možností běžným uživatelům. Někteří tvrdí, že tyto aplikace mohou být převratným bodem v mnoha odvětvích. Ve finančním sektoru se jedná o zlevnění transakčních poplatků, které aktuálně inkasují banky a velké finanční korporace, dále budou schopny zastoupit roli autoritativních společností, ukládat a dále používat bezpečně naše citlivá data či nahradit sociální média a umožnit všem vyjadřovat svobodně své názory [24]. Jednoduše řečeno, možností je nespočet. Pouze a jen čas nám ukáže, jakým směrem se budou decentralizované aplikace ubírat.

Kapitola 3

Programovací prostředí pro vývoj mobilních aplikací

V této kapitole projdeme možnosti vývoje mobilních aplikací a jejich programovací prostředí. Nabízí se nám 3 základní varianty vývoje, jimiž jsou nativní, webové nebo multiplatformní [28]. Rozhodující faktor pro výběr, jakým způsobem aplikaci tvořit, není většinou jeden, ale je jich celá řada. Jedná se například o celkové náklady na vývoj, kvalitu vzhledu aplikace, platformu pro cílové uživatelé nebo rychlost aplikace [49]. Dále je potřeba vzít na vědomí, zda máme i schopné vývojáře, kteří umí používat konkrétní programovací jazyk nebo framework a vyznají se v problematice mobilních aplikací. Přeci jenom je to trochu něco jiného, než tvořit aplikaci pro počítač.

3.1 Nativní aplikace

V první řadě se od webových a multiplatformních aplikací liší tím, že cílí pouze na jednu konkrétní platformu. Mezi nejznámější platformy patří Android a iOS, případně sem lze zařadit i Windows či Blackberry. Rozhraní nativních aplikaci jsou vždy spojena s mobilním operačním systémem a tyto aplikace tak mohou přistupovat ke všem nativním API, jako jsou Bluetooth, geolokace, mikrofon nebo kamera [28]. Výhodou nativního přístupu je velice přívětivé uživatelské rozhraní, dobrá výkonnost aplikací a poměrně příznivá škálovatelnost samotné aplikace, nicméně vyvíjení nativních aplikací se pojí s vyšší časovou, ale i finanční náročností. Pro vývoj se používá jazyk konkrétní platformy, v případě Androidu se jedná o Javu nebo Kotlin a v případě iOS o jazyk Swift nebo Objective-C [39]. Dnešním problémem je taktéž nedostatek zkušených programátorů nativních aplikací, kteří by měli s jejich vývojem vysokou zkušenost [6]. Pro přesun aplikací na jinou platformu je nutné přepsat kód, jelikož nelze jednoduše pozměnit. To stejné platí pro změnu frontendu aplikace [32].

3.2 Multiplatformní aplikace

Multiplatformní (hybridní) aplikace jsou kombinací nativních a webových aplikací. Jejich jedinečná vlastnost je, že jsou dostupné napříč všemi platformami s použitím jediného kódu [28]. Občas se tento přístup nazývá „write-once-run-anywhere“ [32]. Výkonnost je oproti nativním aplikacím běžně nižší a to stejné platí i pro flexibilitu. Je mnohdy obtížné použít nativní funkce jako fotoaparát, mikrofon nebo geolokaci. Zároveň se s tím ale pojí menší časové a finanční náklady, které jsou vykoupeny kvalitou výsledných aplikací. Multiplat-

formní aplikace se tvoří ve frameworcích jako React Native, Flutter či Xamarin [39]. Zatímco React Native poskytuje nativní komponenty pro práci s multiplatformním kódem, Flutter a Xamarin kompilují tento kód do nativního kódu pro lepší výkon [6].

3.2.1 Flutter

Jedná se o open-source framework pro tvorbu mobilních aplikací pro různé platformy vyvinutý společností Google v roce 2017. Jeho silné stránky jsou vykreslovací engine, testovací a integrační rozhraní API a widgety. Vykreslovací engine zodpovídá za rasterizaci scén při změně obrazu [26]. Widgety slouží ve Flutteru jako „stavební bloky“ a každý objekt je widget. Tento framework nejenže nabízí širokou škálu předem vytvořených widgetů, ale také umožňuje uživatelům je tvořit a upravovat [30]. Flutter používá programovací jazyk Dart, který představuje klientsky optimalizovaný a staticky typovaný, objektově orientovaný programovací jazyk. Dart je schopen kompilace do nativního kódu pro konkrétní mobilní zařízení. Samotný Flutter dokonce umožňuje rozšířit stávající mobilní projekt na webovou a počítačovou aplikaci [5]. Kladem frameworku je i fakt, že nabízí funkci okamžitého načtení aplikace po její změně, kterou při vývoji ocení každý vývojář.

3.2.2 React Native

Jde opět o open-source framework pro vývoj hybridních mobilních aplikací vytvořený organizací Meta (dříve Facebook) začátkem roku 2015 používající oblíbený jazyk JavaScript [41]. React Native je znám především svoji rozsáhlou komunitou, která se aktivně stará o chod všech knihoven. Ty jsou nedílnou součástí frameworku React Native, protože nabízejí rychlý vývoj aplikací pro Android i iOS. Na základě toho můžeme říci, že je framework velmi kompatibilní s pluginy třetích stran [30]. React Native vychází z frameworku React používaného pro vývoj webových aplikací, ale umožňuje používat nativní prvky uživatelského rozhraní v mobilních aplikacích. To je možné díky abstrakční vrstvě známé jako „bridge“, která umožňuje vyvolat vykreslovací API v Javě pro Android nebo v Objective-C pro iOS. Specifikem React Native je JSX (JavaScript XML) umožňující reprezentovat objekty na bázi XML sloužící k popisu uživatelského rozhraní. Stylování prvků ve frameworku je velice podobné CSS. Jednotlivé objekty lze oddělit do samostatných souborů a použít je opakovaně v různých projektech [18]. React Native umožňuje načítání aplikací ihned po jejich změně a nabízí taktéž asynchronní provádění operací. Společně s Flutterem se jedná o dva nejoblíbenější frameworky pro tvorbu mobilních aplikací.

3.2.3 Xamarin

Představuje platformu pro vývoj multiplatformních mobilních aplikací koupenou firmou Microsoft v roce 2016, která je aktuálně součástí platformy .NET [43]. Xamarin je napsán v jazyce C# a je integrován do vývojového prostředí VisualStudio. C# nabízí oproti jazykům Java nebo Objective-C například dynamické funkce jazyka, paralelní programování, funkcionální konstrukce, jako jsou lambdy, a další. Výhodou jazyka je i dynamická alokace paměti a garbage collector. Xamarin poskytuje robusní typovou kontrolu v době kompilace i během vývoje. Celkově to vede ke snížení chyb a vyšší kvalitě aplikací. Dále umožňuje přístup ke všem funkcím nativního SDK a samotný vzhled aplikací je zcela nativní. Platforma poskytuje možnosti přímého volání knihoven jazyků Objective-C, Java, C a C++ a nabízí tak použití kódu, který byl dříve vytvořen. Xamarin nabízí i obchod s komponentami, jelikož věří ve spolupráci programátorů. Vývojář si tak může vybrat a použít

Platforma	Xamarin.Forms	React Native	Flutter	Android	iOS
Doba spuštění	Pomalá	Střední	Střední	Rychlá	Rychlá
Velikost aplikace	Velká	Střední	Střední	Malá	Malá
Využití paměti	Střední	Střední	Vysoké	Malé	Malé
Využití CPU	Střední až vysoké	Střední až vysoké	Střední	Střední	Střední
Zkušenosti s vývojem	Střední	Střední	Velmi dobré	Dobré	Dobré

Obrázek 3.1: Porovnání vlastností mobilních aplikací vytvořených frameworky a nativními způsoby [49].

různé cizí komponenty a nemusí pracně produkovat své vlastní [34]. Součástí platformy je i open-source framework uživatelského rozhraní zvaný Xamarin.Forms.

3.3 Webové mobilní aplikace

Posledním možným vývojem mobilních aplikací jsou právě ty webové. Jsou napsány v jazycích HTML, aktuálně ve verzi HTML5, CSS nebo JavaScript. Běží ve webovém prohlížeči stejně jako webová stránka, ale je zde jeden rozdíl, který rozlišuje webovou stránku a webovou aplikaci. Jedná se o to, že webová aplikace je určena k interakci mezi uživatelem a aplikací, přičemž webová stránka slouží pouze ke čtení. Od předchozích dvou přístupů se odlišuje tím, že ji lze spustit na počítači i telefonu, a dále že běží ve webovém prohlížeči a nelze ji tedy stáhnout. K tomu se pojí i její závislost na internetu, bez kterého není schopna provozu. Za výhodu by se dalo považovat, že šetří paměť v telefonu [56]. S multiplatformními aplikacemi sdílí plus v použití různých platforem a tedy jednotného zdrojového kódu [17].

Kapitola 4

Použité technologie při tvorbě decentralizovaných aplikací

V následujících sekcích si probereme platformy a frameworky, které velkou mírou přispěly k vytvoření výsledných aplikací, které jsou dále popsány v kapitole 6.

4.1 Truffle

Nástroj Truffle je jedním z poskytovaných nástrojů v komplexní sadě Truffle Suite. Uživatelům nabízí vývojové prostředí k tvorbě smart kontraktů, testovací framework využívající EVM pro jejich následné testování a nakonec příležitost k nasazení smart kontraktů na sítě Etherea. Ve své práci jsem použil Truffle primárně k nasazení smart kontraktů na testovací síť Etherea a k testování smart kontraktů (viz kapitola 7).

V první řadě je po stažení nástroje nutné si vytvořit projekt příkazem `truffle init`, který v aktuální složce utvoří následující strukturu:

```
contracts/
migrations/
test/
truffle-config.js
```

Do složky `contracts` je potřeba umístit všechny smart kontrakty, které chceme používat. To však neplatí pro importy, které byly staženy pomocí správce balíčků `npm` nebo `yarn` do složky s externími moduly `node_modules`. Ve složce `migrations` mají být uloženy soubory pro migraci. V ní se mají nacházet javascriptové soubory, podle kterých se provede nasazení smart kontraktů na síť. Jeden soubor zpravidla slouží pro migraci jednoho smart kontraktu a v případě rozsáhlejšího projektu je nezbytné vytvořit souborů více. Takto vypadá migrační soubor pro hlavní smart kontrakt `Queans`:

```
const Queans = artifacts.require("Queans");

module.exports = function(deployer) {
  deployer.deploy(Queans);
};
```

Následující složka `test` může zůstat prázdná. Pokud ale chceme otestovat smart kontrakty, tak toto je správné místo pro uložení testů, které jsou napsány opět v JavaScriptu.

Nakonec se dostáváme ke konfiguračnímu souboru `truffle-config`. Zde si můžeme specifikovat sítě, na které bude náš smart kontrakt nasazen. V základním nastavení je dostupná síť `development`, která běží lokálně. Při práci jsem použil testovací síť Etherea Sepolia (viz sekce 4.4), které dodávám informace o poskytovateli a síti. Poskytovatel je zde tvořen pomocí knihovny `@truffle/hdwallet-provider`, kde je zapotřebí vyplnit klíčovou frázi peněženky, privátní klíč hlavního účtu v peněžence a URL koncového zařízení, v našem případě se jedná o službu Infura (viz sekce 4.3). Adresa prvního účtu v peněžence je implicitně brána jako tvůrce smart kontraktu. Při správném nastavení se v této práci smart kontrakt nasadí na síť příkazem `truffle migrate --network sepolia`. Migrace je provedena na základě všech souborů uložených ve složce `migrations` [16].

4.2 Ganache

Ganache je dalším z nástrojů dostupných v sadě Truffle Suite. Jde o nástroj pro vytvoření osobního blockchainu určeného k rychlému vývoji aplikací běžících na síti Ethereum. Ganache lze použít dvěma způsoby. Zaprvé jako desktopovou aplikaci, která obsahuje nástroje určené ke sledování stavu sítě, transakcí, bloků a poskytuje i další možnosti pro řízení sítě. Zadruhé ji je možno použít skrze příkazový řádek, který nabízí taktéž plno funkcí, ale již bez pěkného uživatelského rozhraní.

Při vývoji aplikace jsem používal pouze formu s příkazovou řádkou, jelikož pro mé základní potřeby byla plně dostačující. Nástroj mi byl nápomocen zejména po úpravě funkcí smart kontraktu k jejich rychlému otestování pomocí příkazu `truffle test` (viz kapitola 7). V raných fázích vývoje před připojením na testovací síť Etherea jsem používal používal osobní blockchain taktéž jako testovací síť.

Po stažení nástroje probíhá spuštění osobního blockchainu v konzoli pomocí příkazu `ganache`. Ten vyvolá vytvoření soukromého blockchainu a nabídne uživateli 10 různých účtů včetně testovacích prostředků a privátních klíčů uživatelů. Nasazení smart kontraktu lze docílit příkazem `truffle migrate --network development` nebo zkráceně `truffle migrate`, jelikož síť development je brána implicitně. Použití nástroje Truffle je při testování na blockchainu pocházejícího z Ganache závislé. Pro příkaz `truffle test` a `truffle migrate` (platí pouze pro síť development) je nutné mít zapnutý osobní blockchain. Výhodou testovacího prostředí je prakticky neomezený počet prostředků a nepotřebnost konfigurace ostatních souborů [15].

4.3 Infura

Jedná se o webovou službu, která v podstatě poskytuje vzdálený uzel Etherea, ke kterému lze přistupovat prostřednictvím rozhraní API. Vývojářům tak umožňuje vytvářet a nasazovat aplikace na hlavní nebo testovací sítě Etherea, aniž by museli provozovat vlastní uzel blockchainu. Při vytvoření účtu si uživatel vytvoří projekt, ke kterému obdrží přidružené Project ID (API klíč).

Tento klíč používám při zadávání údajů o poskytovateli v souboru `truffle-config`. Po tomto propojení již mohu spustit příkaz `truffle migrate --network sepolia`, který úspěšně nasadí smart kontrakt na testovací síť Sepolia [33].

4.4 Sepolia

Sepolia je jedna ze dvou aktuálně funkčních testovacích sítí Etherea s cílem napodobovat hlavní síť. Byla navržena tak, aby simulovala náročné podmínky sítě s rozdílem kratší doby tvorby bloku, což vývojářům umožňuje rychlejší potvrzení transakcí a zpětnou vazbu. Sepolia oproti druhé testovací síti Goerli nabízí dostatek tokenů pro testování a nenastává tak problém s jejich nedostatkem. Testovací měnu lze získat pomocí tzv. faucetů, kde po zadání adresy účtu obdrží uživatel testovací prostředky. Omezené zdroje získané z faucetů tak zamezují přetížení sítě [4].

4.5 Expo

Platforma a open-source framework Expo slouží pro vývoj webových aplikací i mobilních aplikací pro platformy Android a iOS, které běží nativně v daných systémech. Expo nabízí také mobilní aplikaci Expo Go, která slouží pro testování aplikací React Native, aniž by vývojář musel cokoliv lokálně vytvářet. Tato aplikace je rovněž doporučena na oficiálních stránkách React Native pro nové vývojáře mobilních aplikací, díky čemuž jsem ji použil v průběhu tvorby aplikací. Pomocí Expa lze testovat aplikace buď na fyzickém telefonu nebo s použitím emulátoru. Aplikaci lze jednoduše spustit na počítači příkazem `npx expo start` a v případě fyzického telefonu si naskenovat QR kód, který v Expo Go otevře konkrétní aplikaci. Expo dále nabízí platformu Snack, která dynamicky sestaví a kompiluje kód a funguje jako testovací prostředí, které umožňuje bezprostředně spouštět React Native aplikace v prohlížeči a poskytuje zobrazení jak na zařízení iOS a Android, ale i pro web. Expo SDK zároveň poskytuje přístup k mnoha funkcím zařízení a systému jako fotoaparát, kalendář, kontakty a další. S pomocí funkce EAS Build, další vychytávkou platformy, je uživatelům umožněno vytvořit produkční sestavu aplikace, která je připravena k odeslání do obchodů s mobilními aplikacemi iOS i Android [21].

4.6 Emulátor v Android studiu

Android Studio je oficiální integrované vývojové prostředí (IDE) pro vývoj aplikací pro Android. V našem projektu nás bude zajímat pouze jedna část vývojového prostředí, a to rychlý a funkčně bohatý emulátor, který využijeme namísto fyzického mobilního telefonu se systémem Android. Vývojové prostředí nabízí celou řadu emulátorů s různými rozlišeními a verzemi operačního systému, díky nimž lze dobře otestovat výslednou aplikaci [27].

4.7 MetaMask

Další potřebnou součástí projektu je kryptoměnová peněženka. Pro své řešení jsem si vybral MetaMask, což je softwarová kryptoměnová peněženka, kterou lze nalézt jako webové rozšíření nebo mobilní aplikaci a je stavěna přímo k interakci se sítěmi Etherea. Kromě toho, že nabízí základní funkce jako jiné peněženky, tak také umožňuje připojení k decentralizovaným aplikacím a uživatelům tedy umožňuje komunikaci s blockchainem. Mimo peněženku nabízí Metamask i svůj SDK (Software Development Kit), který nabízí vývojářům jednoduchý způsob propojení aplikací s peněženkou MetaMask [42].

4.8 Etherscan

Etherscan je platforma pro Ethereum, která slouží jako prohlížeč blockchainu a zajišťuje spravedlivý přístup k datům. Díky tomu, že všechny interakce na Ethereu jsou veřejné, prohlížeč nám jednoduše umožňuje je sledovat. Nabízí se nám sledování transakcí, bloků, adres, smart kontraktů a dalších různých dat v blockchainu [9]. Jeho důležitou rolí je i ověření smart kontraktů, které může jejich autor pomocí zdrojového kódu ověřit. V ověřených smart kontraktech můžeme vidět jejich zdrojový kód a sledovat, zda se chovají tak, jak mají. Ověřené smart kontrakty nabízí i jednoduché rozhraní, skrze které lze otestovat jejich chování bez použití decentralizovaných aplikací [20].

Kapitola 5

Současný stav Q&A řešení

Tato kapitola má za cíl krátce objasnit pojem Q&A sekce společně s možnostmi výběru otázek v médiích a v další řadě přiblížit aplikace nabízející Q&A sekci. Na konci kapitoly provedu kritické zhodnocení zmíněných aplikací a návrh, jak se zmíněným nevýhodám existujících řešení vyvaruji v samotné práci.

5.1 Q&A sekce

Sekcí otázka/odpověď rozumíme prostor, kde lidé mohou klást otázky, na které chtějí získat odpovědi. Obecně se může jednat o interaktivní dialog mezi hostem, případně moderátorem, a publikem [46]. Během relace mohou diváci nebo posluchači pokládat své otázky, které jsou následně zodpovězeny. Cílem těchto relací je umožnit divákovi nebo posluchači možnost aktivně se zapojit do diskuze a získat dodatečné informace o dané problematice. Nejčastěji jde o otázky relevantní pro dané programy a často se týkají aktuálních událostí, ať již politických, kulturních nebo sportovních. Tento formát se mnohdy využívá v talk show, rozhovorech, panelových diskuzích nebo při přednáškách. V této práci bych se chtěl zaměřit na televizní a rozhlasové relace, kde vidím největší přínos aplikace.

Výběr otázek v médiích

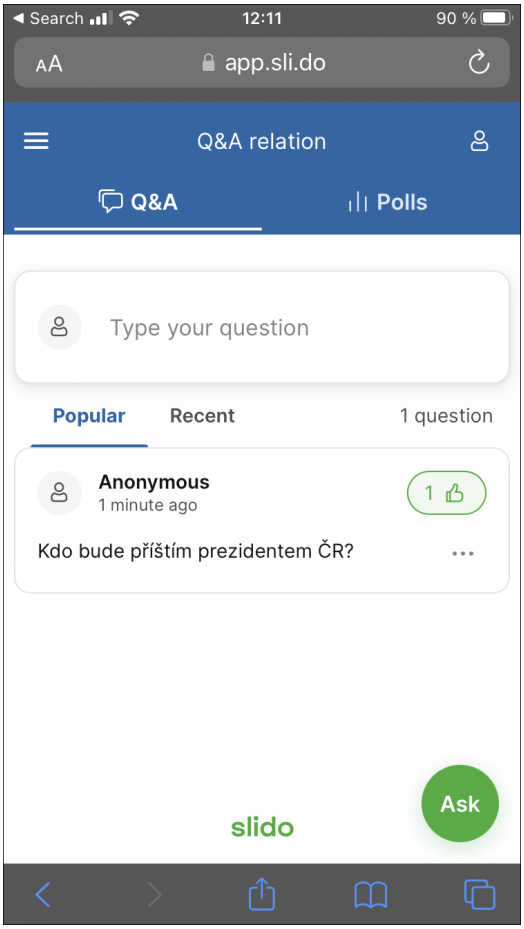
Způsob výběru otázek se samozřejmě může lišit a záleží jen na konkrétní relaci a pořadateli, jak k výběru přistoupí. V každém případě by měla být zajištěna transparentní a férová selekce otázek. Díky ní publikum obdrží odpověď na to, co si žádá. Bohužel nelze zaručit, že dotazy nebudou žádným způsobem cenzurovány ani zmanipulovány. Níže jsou popsány 3 způsoby, jak média přistupují k výběru otázek.

1. Předem zvolené otázky – Pořadatelé si před začátkem relace sestaví seznam otázek, na které bude host odpovídat. Výběr konkrétních otázek se děje na základě předem definovaných kritérií, jimiž jsou aktuálnost tématu, relevance pro publikum či nápaditost.

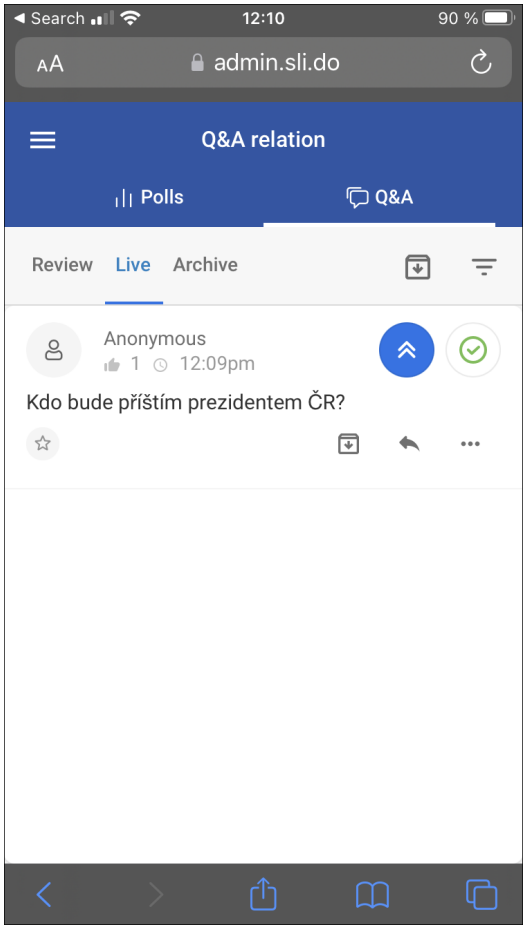
2. Předem získané otázky publika – Posluchači a diváci jsou před začátkem relace osloveni, aby zaslali své otázky skrze sociální média nebo e-mail. Pořadatelé následně vybírají ty, které subjektivně vyhodnotí jako nejlepší či nejvhodnější.

3. Živé otázky – Během probíhající relace je umožněno jejímu publiku pokládat otázky živě nebo pomocí mobilního telefonu (ať již telefonátem nebo zprávou), chatu či jiné platformy. Moderátor následně vybírá otázky, které budou zodpovězeny.

5.2 Analýza současných Q&A aplikací



Obrázek 5.1: Slido - pohled uživatele.

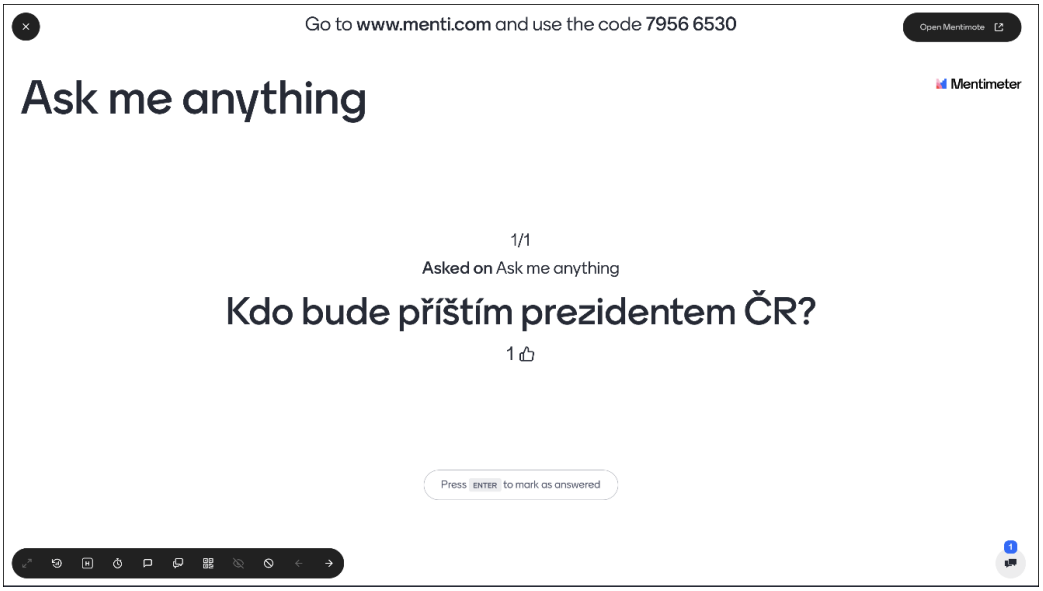


Obrázek 5.2: Slido - pohled přednášejícího.

5.2.1 Slido

Jedná se o online aplikaci pro interakci mezi účastníky a přednášejícími, která se používá zejména při přednáškách, prezentacích, workshopech, konferencích a dalších podobných akcích. Aplikace nabízí různou paletu nástrojů, avšak my se zaměříme pouze na jednu její část – sekci otázek a odpovědí. Slido umožňuje jednoduše zadávat otázky po naskenování QR kódu nebo po zadání hesla pro konkrétní relaci [13]. Na obrázcích 5.1 a 5.2 jsou zobrazeny náhledy webové aplikace na mobilním telefonu z pohledu uživatele i přednášejícího.

Diváci nemají žádné omezení na počet otázek, který mohou položit a jsou schopni jakoukoliv z vlastních otázek během 5 minut od jejího položení editovat. Vzájemně mají prostor k hlasování pro otázky druhých. Tímto způsobem je zajištěna dynamika pořadí nejoblíbenějších otázek. Zobrazení dotazů je možné dvěma způsoby, a to podle popularity otázek



Obrázek 5.3: Mentimeter - pohled prezentujícího, širokoúhlé zobrazení.

nebo podle času, kdy byly zadány. Publikum si dále může vybrat, zda chce položit dotazy anonymně nebo pod svým jménem. Ze strany účastníka jsme tímto zmínili všechny jeho možnosti, které jsou pro Q&A sekci nabídnuty.

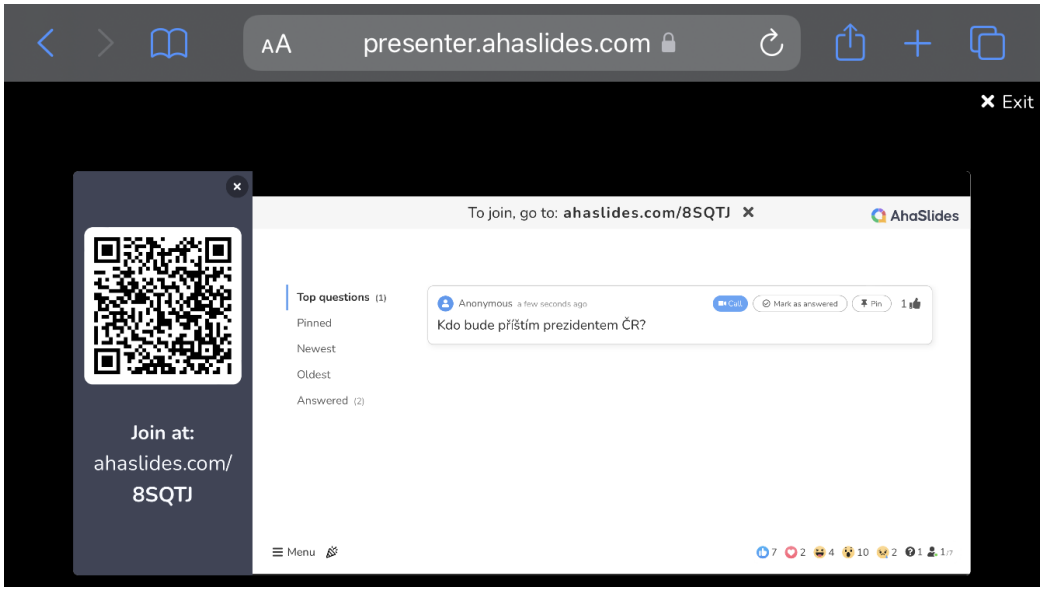
Nyní se přesuneme na stranu organizátora Q&A sekce, která je už poněkud obsáhlejší. Ten má možnost otázky označit jako oblíbené, zodpovězené (otázka se následně archivuje a publiku již není dostupná) nebo zvýraznit konkrétní otázku ve chvíli, kdy na ni odpovídá. Dále má příležitost jakoukoliv otázku upravit (po editaci se zobrazí text oznamující úpravu otázky, kde ale není zřejmé, jestli proběhla uživatelem nebo organizátorem), textově na ni odpovědět, archivovat nebo smazat. Slido nabízí další dodatečné funkce, jako jsou exporty otázek, různé souhrny nebo i variantu, ve které se nejprve dotaz odešle ke schválení organizátorovi, který podle jeho obsahu uzná za vhodné, zda dotaz zveřejnit. Část z vyjmenovaných funkcí je ale podmíněna placeným měsíčním členstvím.

5.2.2 Mentimeter

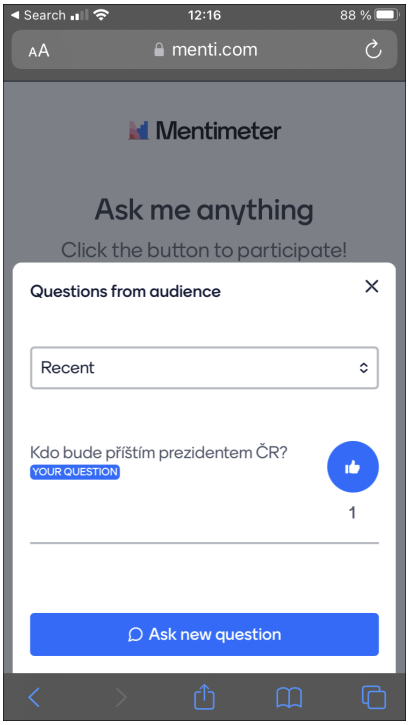
Mentimeter je jedna z řady webových aplikací, která umožňuje založit Q&A sekci při prezentaci. Využívá se především v průběhu nebo na konci přednášek s prezentací, kde se na snímku zobrazí výčet dotazů, na které chce publikum znát odpověď. Mentimeter nabízí několik různých druhů snímků pro sestavení prezentace. Do sekce se účastník dostane pomocí naskenování QR kódu nebo zadáním 8 místného hesla [40]. Na obrázcích 5.3 a 5.5 je znázorněno zobrazení aplikace na mobilním zařízení.

Uživatelské rozhraní je minimalistické a nabízí jen několik málo možností, jak s otázkami nakládat. Dotazy lze položit pouze anonymně, nikoliv pod konkrétním jménem. Diváci mají příležitost hlasovat pro otázky, které se jim líbí. Nakonec si mohou urovnat pořadí otázek podle počtu hlasů nebo podle času, kdy byly vytvořeny.

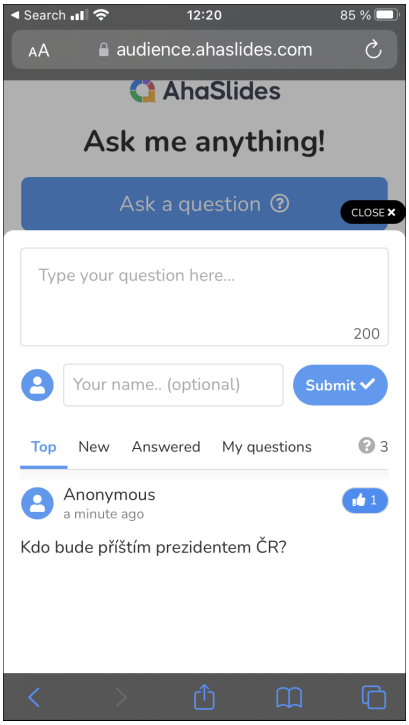
Samotný moderátor před začátkem prezentace nastavuje, zda publikum může pokládat otázky na všech druzích snímků nebo pouze na snímcích označených jako Q&A. Dále je schopen nastavit, jestli ostatní účastníci vidí dotazy ostatních či ne. V případě, že nevidí, nelze pro žádné hlasovat. Při prezentaci je prezentujícímu umožněno označit otázky za zod-



Obrázek 5.4: AhaSlides - pohled prezentujícího, širokoúhlé zobrazení.



Obrázek 5.5: Mentimeter - pohled uživatele.



Obrázek 5.6: AhaSlides - pohled uživatele.

povězené nebo je odstranit. Placený plán nabízí volbu, při které lze dotazy schvalovat ještě před tím, než jsou zobrazeny ostatním.

5.2.3 AhaSlides

Poslední aplikací, kterou si popíšeme, je AhaSlides. Organizátorům poskytuje možnost vytvoření prezentace, ve které lze použít Q&A snímky [2]. Jedná se o velice podobnou alternativu k aplikaci Mentimeter. Po naskenování QR kódu nebo použitím speciálního odkazu se uživateli zobrazí pohled z obrázku 5.6.

Rozhraní dovoluje uživateli pokládat otázky pod zvoleným jménem, které se zadává pro každou otázku zvlášť, nebo anonymně. Publikum má příležitost hlasovat pro otázky ostatních, aby se dostaly na lepší pozici a zvýšila se šance na jejich zodpovězení. Poslední možností je opět srovnání dotazů podle jejich hodnocení nebo času, kdy byly položeny.

Prezentující má možnost otevřít další okno pro moderátora, který může přicházející dotazy před zveřejnením schválit nebo zamítnout. Dále může otázky třídit do následujících kategorií: nejlepší otázky, označené, nejnovější, nejstarší a zodpovězené. AhaSlides také nabízí filtrování explicitních výrazů a možnost zavolat autorovi konkrétní otázky. Jeho pohled je znázorněn na obrázku 5.4.

5.3 Porovnání existujících aplikací s cíli stanovenými pro tuto práci

Jak je uvedeno výše, existuje několik aplikací pro vytvoření Q&A sekce. Další z těch, které tu nebyly zobrazeny, fungují na velice podobném principu jako Mentimeter či AhaSlides a jejich použití je optimalizováno na prezentace ve školách nebo při přednáškách. Všechny tři aplikace jsou vytvořeny jako webové, přičemž Mentimeter a AhaSlides jsou vytvořeny z pohledu prezentujícího výhradně pro počítač a v mobilním zobrazení nejsou responzivní. Jejich ovládání je na telefonu poněkud složité z důvodu malých tlačítek, a to i při širokoúhlém zobrazení. Po vyzkoušení aplikací a zamyšlení se, proč neexistují v podobě mobilních aplikací, jsem došel k závěru, že není optimální aplikaci stahovat, zejména pokud je nedostatek času. Zkrátka jednodušším řešením je načíst webovou aplikaci, připojit se a rovnou se zapojit do komunikace. Výjimku zde tvoří Slido, které nabízí mobilní aplikaci, ale pouze pro publikum, ve které však bohužel nevidím oproti webové variantě žádný přínos a označil bych ji tak za téměř zbytečnou. Slido oproti dvěma zbylým variantám nabízí jednoduché a přehledné rozhraní pro přednášející i na samotném telefonu. Z pohledu účastníka jsou všechny tři varianty velice podobné a uživatelsky přívětivé.

Postoupíme-li o krok dále a zaměříme-li se na decentralizované aplikace, musím konstatovat, že jsem nebyl schopen žádnou decentralizovanou aplikaci, která by nabízela Q&A sekci, vyhledat.

Nyní si rozebereme jednotlivé nevýhody, které existujicí řešení obsahují. První z nich je samotná cenzura. Všechna tři řešení nabízí omezování dotazů několika různými způsoby. Jedním z nich je příležitost kontrolovat příchozí otázky a na základě jejich obsahu je schválit nebo odstranit, případně s nimi nic nedělat. V případě Slida se jedná ještě o další možnost, kterou je editace zpráv. Po změně zprávy se označí jako „edited“ a žádný z uživatelů nemá šanci zjistit, zda ji změnil sám autor nebo organizátor. Druhou nevýhodou označím příležitost ke spamu. Pokud pomineme možnosti, že lze omezit celé publikum, aby nemohlo pokládat otázky, a kontrolu příchozích otázek, nastává riziko, že se do relace do-

stane nechtěný účastník, který ji bude zahlcovat velkým množstvím dotazů. I přes kontrolu příchozích otázek může toto chování organizátorům velice znepříjemnit průběh relace.

Jak je zmíněno již v kapitole 1, aplikace by mohla mít své využití ve veřejnoprávních médiích, ale i všude tam, kde chceme zaručit transparentnost samotného procesu dotazování se skrze online prostor. Dvě nevýhody, které jsem zmínil v předešlém odstavci, se snaží má práce odstranit. Neschopnost cenzury otázek a ověření nemožné cenzury bude umožněno díky vlastnostem blockchainu, kódu a transparentnosti smart kontraktu a aplikaci sloužící jako rozhraní, skrze které nebude nikdo schopen jakkoli omezovat publikum relace. Druhou, neméně důležitou problematiku spamu znemožní maximální možný počet otázek, kterým může uživatel přispět do relace. Aktuálně je tento počet nastaven na hodnotu 1. Dalším pomocníkem je smart kontrakt určený ke správě identit, který dovolí se do relace dostat pouze pod podmínkou, že uživatel prokáže správci identit svoji identitu. V aplikaci je tento způsob pouze simulací reality, nicméně při reálném provozu by se předpokládalo například předložení bankovní identity nebo občanského průkazu k ověření identity uživatele.

Kapitola 6

Návrh a implementace

Na následujících stránkách bude nejprve představen návrh aplikace a následně uveden protokol pro decentralizované řízení Q&A sekcí. Dále bude popsáno, jaké hlavní funkce poskytují smart kontrakty `identityRegistry` a `Queans`. Nejdříve se pojďme podívat, jak by mohla výsledná aplikace vypadat.

6.1 Návrh aplikace

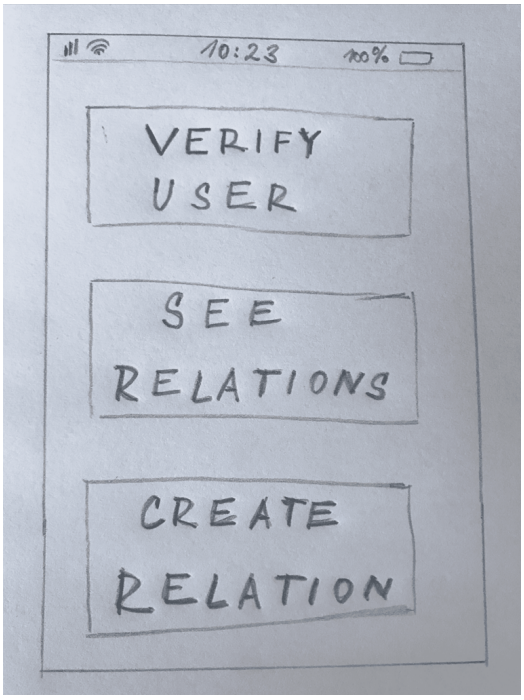
Při návrhu je nejdůležitější si rozmyslet, co a kde bude umístěno a jak by se co mělo chovat. Zde se mi nabízely 2 způsoby řešení. Jedním z nich bylo vytvořit jednu aplikaci, ve které si lze vybrat, zda chce být uživatel moderátorem nebo běžným účastníkem relace. Druhý způsob nabízel tvorbu dvou aplikací, kde první z nich by byla pro publikum relací a druhá čistě pro moderátory. Na obrázcích 6.1 a 6.2 je prvotní návrh samostatné aplikace. Nakonec jsem se však vydal opačnou cestou, přestože dlouhou dobu to vypadalo, že bude vyhovující pouze jedna aplikace. Postupem času se ale smart kontrakt `Queans` měnil a s ním i podoba aplikace.

Dále jsem vytvořil třetí aplikaci nad rámec práce, která představuje proces ověřování u poskytovatele identit. V původním návrhu měly být identity ověřeny pouze skrze tlačítko `verify user` podle obrázku 6.1, kde by se ověřil aktuálně přihlášený uživatel, který by následně mohl využívat aplikaci. Aplikace využívá smart kontrakt `identityRegistry` popsaný v sekci 6.3.

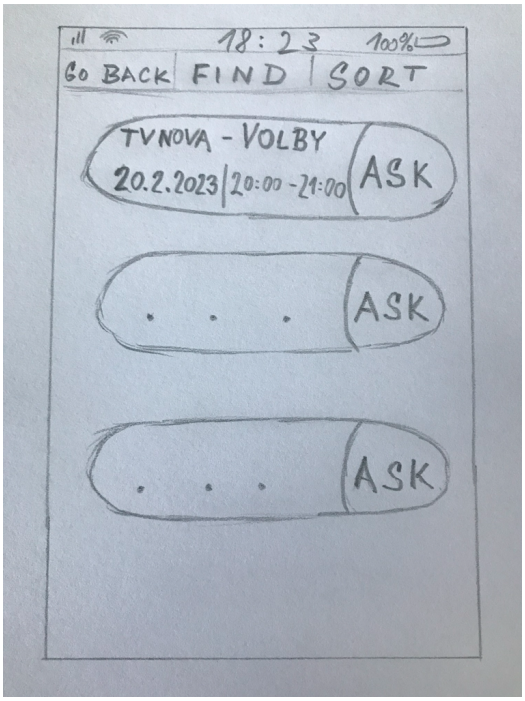
Mým cílem bylo vytvořit všechny aplikace nabízející rozhraní, které bude mít základní funkcionalitu, ale bude plně funkční. První aplikace s pracovním názvem Voter slouží pro běžné uživatele. Druhá má název Moderator a je využita výhradně moderátory relací. Poslední z nich se nazývá Management a slouží pro správu identit uživatelů. K interakci s blockchainem slouží 2 hlavní smart kontrakty. Prvním je již zmiňovaný `identityRegistry`, který rozšiřuje rozhraní `IIdentityRegistry` a používá smart kontrakty `MultisigActionMembers` a `MultisigMembers`. Druhým je smart kontrakt `Queans` popisovaný v sekci 6.4, který jsem vytvořil jako součást práce.

6.2 Návrh protokolu vhodného pro decentralizované řízení Q&A sekcí

Před samotnou implementací smart kontraktu zajišťujícího decentralizované řízení Q&A sekcí bylo třeba navrhnout protokol, podle kterého se bude implementace řídit. Návrh



Obrázek 6.1: Návrh úvodní obrazovky.



Obrázek 6.2: Návrh přehledu relací.

protokolu je popsán na obrázku 6.3. Jak si můžete všimnout, je tvořen 5 základními entitami a 4 fázemi, kde entita blockchain zahrnuje v protokolu všechny smart kontrakty, které aplikace používá.

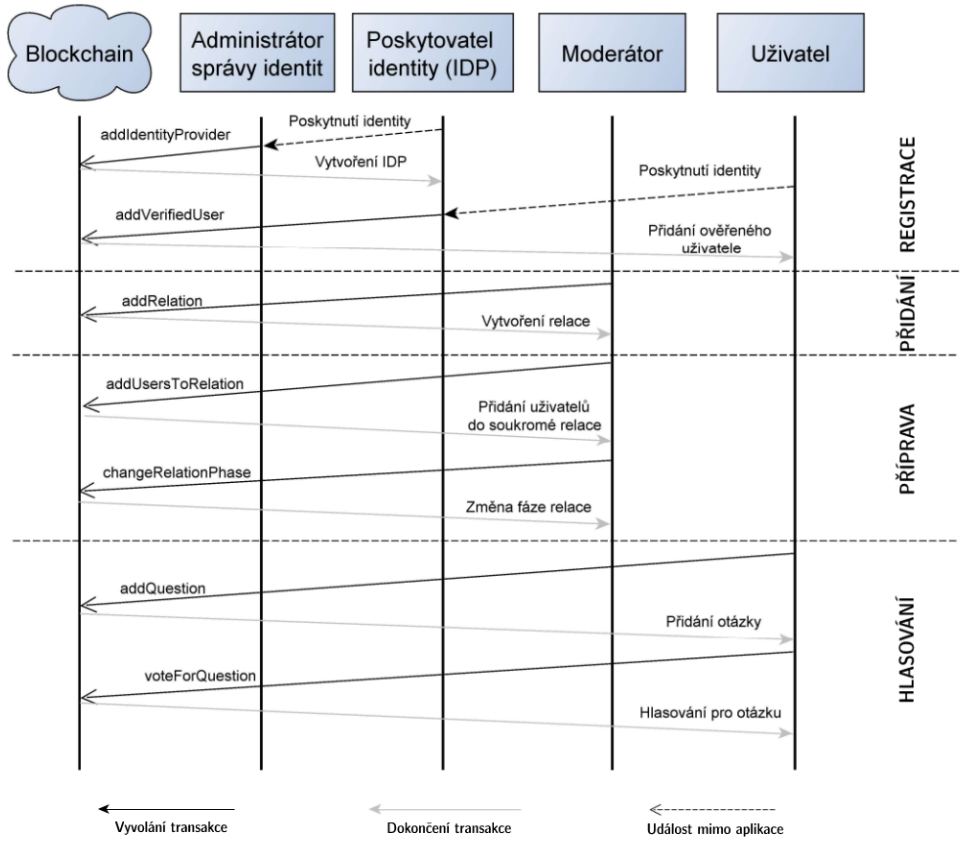
V první fázi se vyskytují administrátor správy identit, poskytovatel identit a uživatel. Administrátor je osoba, případně skupina osob, která je zodpovědná za přidávání nových poskytovatelů identit (angl. identity providers, zkr. IDPs) na základě jejich digitální identity. Poskytovatelé identit mají za úkol ověřovat identitu poskytnutou od uživatelů (v případě této práce jde o blockchainovou adresu uživatele) a následně přidat ověřené uživatele do smart kontraktu. Všechny funkce první fáze jsou implementovány v aplikaci s názvem Management.

Druhá a třetí fáze je implementovaná v aplikaci pro moderátory. Druhá fáze zobrazuje proces tvorby relace a změnu uživatele na moderátora. Ve třetí fázi dochází k přípravě relace na hlasování. Zde může moderátor zvát do soukromé relace účastníky a následně změnit stav relace, aby mohla začít poslední fáze.

Ve čtvrté fázi již figuruje pouze uživatel, který může pokládat otázky do různých relací a hlasovat pro otázky ostatních uživatelů. Tato fáze je implementovaná v aplikaci Voter.

Motivace uživatelů

Čím jsou uživatelé, kteří se podílejí svoji účastí v relaci, motivováni? Pokud se už někdo do relace připojí, nejspíše tak jedná za účelem se pobavit nebo se dozvědět nové informace. Většina lidí nepředpokládá, že by za svoji účast mohla být odměněna ještě jiným způsobem, přestože organizátoři některých akcí oceňují aktivní publikum dárky. V našem případě by těmito „dárky“ mohla být finanční odměna v podobě mincí nebo tokenů sítě, na které by byl smart kontrakt spuštěn. Této implementaci se však má práce nevěnuje. Pokusím se tedy nastínit nápady, které by mohly být v rozšíření aplikace uskutečněny. Po ukončení relace je



Obrázek 6.3: Protokol komunikace pro řízení Q&A sekcí.

možné odměnit otázku s nejvíce hlasy určitým poměrem získaných prostředků stanoveným podle počtu celkových otázek dané relace. Stejným způsobem by se dalo ocenit i nejlepších „X“ otázek, jen by se upravil poměr odměn za jednotlivá místa. Další možností je ohodnotit i tvůrce samotné relace, a to za aktivitu uživatelů, kteří platí poplatky za pokládání dotazů a hlasování. Další možností je i označení aktivních účastníků relací v samotné aplikaci, aby dali ostatním najevo, že jsou po dlouhou dobu nedílnou součástí relací.

6.3 identityRegistry

Tato sekce se zabývá smart kontraktem `identityRegistry`, který mi byl poskytnut k použití vedoucím práce doktorem Homoliakem, stejně jako smart kontrakty `MultisigActionMembers` a `MultisigMembers` a rozhraní `IIdentityRegistry`. Jedná se o smart kontrakt, ve kterém se vyskytují 3 typy entit a slouží zejména k ověření identit uživatelů. Entitami jsou administrátoři správy identit (zkr. IDM admin), poskytovatelé identit a uživatelé, kteří chtějí ověřit svoji identitu. Níže si popíšeme tři nejdůležitější funkce smart kontraktu, které jsou využity ve výsledných aplikacích. Po nasazení smart kontraktu máme pouze administrátora, kterým se stane osoba, která ho nasadila na síť.

6.3.1 Funkce addIdentityProvider

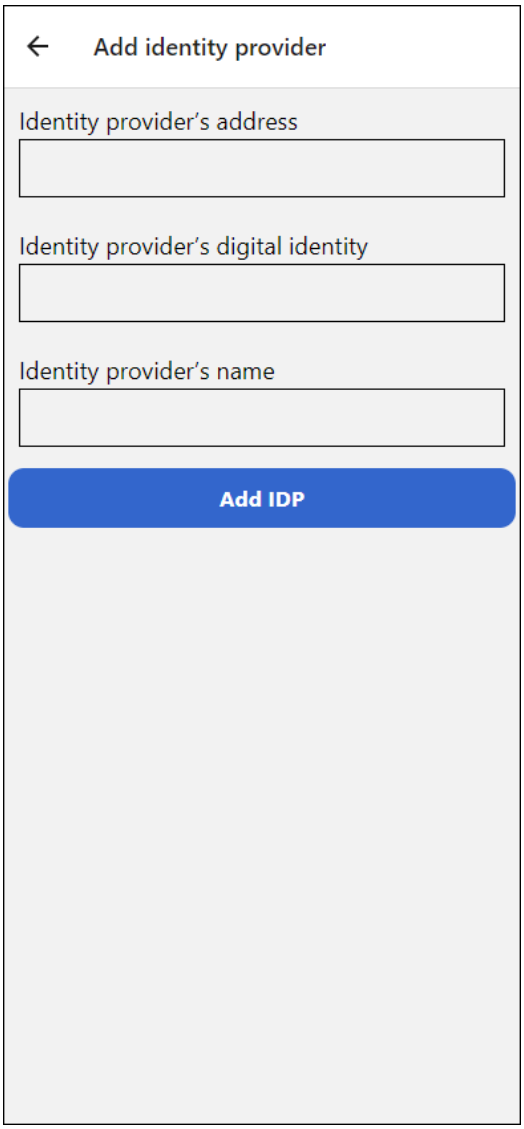
První funkcí, kterou si zmíníme, je funkce sloužící k přidání poskytovatele identity. K vytvoření nového poskytovatele identity je potřebné zadat jeho adresu, digitální identitu a jméno, jak je zobrazeno na obrázku 6.4. Funkci může provádět výhradně administrátor správy identit. V případě, že by administrátorů bylo více, musí nadpoloviční většina schválit přidání konkrétního poskytovatele identity. Pro zjednodušení funkčnosti aplikace však nelze v rozhraní výsledné aplikace další administrátory přidávat.

6.3.2 Funkce addVerifiedUser

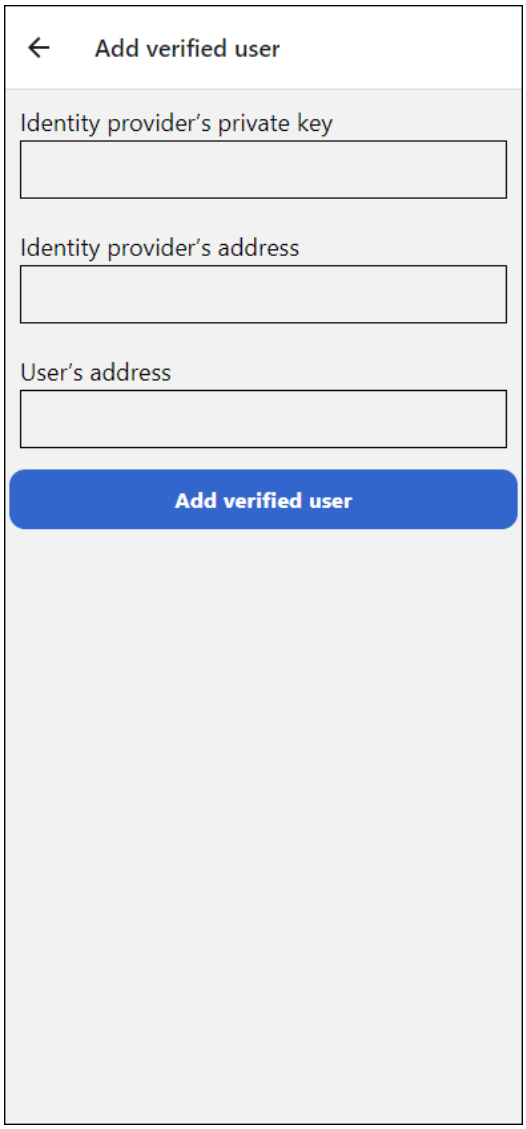
Nyní přejdeme k funkci, která se stará o přidávání ověřených uživatelů do smart kontraktu. Za tuto činnost jsou zodpovědni poskytovatelé identity, jejichž úkolem je ověřovat uživatele na základě jimi poskytnuté identity a následně je buď zařadit nebo nezařadit mezi ověřené uživatele ve smart kontraktu. Pro provedení operace využívají rozhraní zobrazené na obrázku 6.5, kde poskytovatel identity vyplňuje svůj privátní klíč, svoji veřejnou adresu a adresu uživatele. Pomocí privátního klíče se podepisuje adresa uživatele s textovým řetězcem a adresou smart kontraktu. Ve funkci `addVerifiedUser` se následně porovná pomocí funkce `ecrecover` veřejný klíč s podpisovými daty, který slouží ke kontrole, že operace byla provedena autorizovanou osobou.

6.3.3 Funkce verifyIdentity

Nakonec se dostáváme k externí funkci, jejíž úkolem je potvrdit, že identita uživatele (v našem případě adresa), již byla ověřena poskytovatelem identity. Funkce se volá přes rozhraní `IIdentityRegistry`, jehož funkce je specifikována ve smart kontraktu `identityRegistry`. V konstruktoru smart kontraktu `Queans` je napevno nastavena adresa smart kontraktu `identityRegistry` a přiřazena do inicializovaného rozhraní. `VerifyIdentity` ověřuje adresu odesílatele blockchainové transakce a používá se ve všech funkcích smart kontraktu `Queans`, které mění stav blockchainu (smart kontraktu). Tímto způsobem je demonstrováno ověření identit třetí stranou (jiným smart kontraktem).



Obrázek 6.4: Rozhraní aplikace znázorňující přidání nového poskytovatele identity.



Obrázek 6.5: Rozhraní aplikace znázorňující přidání ověřeného uživatele.

6.4 Queans

V této sekci se podíváme na smart kontrakt `Queans`, který byl jednou ze dvou hlavních částí práce. `Queans` je naprogramován podle protokolu v sekci 6.2 a tvoří jádro samotné aplikace. Níže si podrobně povíme o uživatelích, relacích a otázkách.

6.4.1 Uživatelé

Uživatelé jsou nedílnou součástí většiny aplikací. Ani v našem případě tomu není jinak. Mezi uživatele se z protokolu na obrázku 6.3 řadí kromě jich samotných i moderátor, který je speciálním typem uživatele. Uživatel se moderátorem stane tak, že vytvoří relaci, ve které získá roli moderátora, nicméně ve všech ostatních relacích je pouhým uživatelem. Abychom mohli pracovat s jejich adresami, bylo nutné si vytvořit datovou strukturu, kde budou adresy uloženy. Tou je množina adres přejatá z knihovny `@openzeppelin`.

EnumerableSet.AddressSet users

Oproti klasickému poli se jedná o optimalizovanou strukturu pro ukládání jednoho typu hodnot, v našem případě adres, která ještě navíc nabízí funkci `contains()` pro ověření, zda se konkrétní prvek nachází v dané množině. Do množiny se uživatel dostane automaticky po použití jedné z funkcí měnících stav blockchainu (smart kontraktu). Podmínkou však je, že byl již dříve ověřen poskytovatelem identity.

6.4.2 Relace

Po uživatelích jsou relace druhou nejdůležitější částí kontraktu. Nejprve si zobrazíme strukturu relace a povíme si o jejích nejdůležitějších částech.

```
struct Relation {
    address author;
    uint creationTime;
    bool isPublic;
    bool votePhase;
    address[] privateList;
    uint ID;
    uint startOfRelation;
    uint questionCloseTime;
    string name;
    uint[] questionsKeys;
}
```

Většina částí tvořících relaci je nezajímavých a od založení relace se nemění. Nicméně bych rád vyzdvihl čtyři z nich, které značně ovlivňují fungování celé aplikace. První je proměnná `isPublic`. Ta udává, zda je relace otevřená všem uživatelům nebo je omezená pouze pro pozvané účastníky. S tím se pojí i pole `privateList`, které v sobě uchovává adresy všech pozvaných uživatelů do relace. V případě, že je relace veřejná, pole `privateList` zůstane prázné a nepracuje se s ním. Jestliže ale moderátor nastaví relaci jako soukromou, musí také přidat účastníky za pomoci funkce `addUsersToRelation`, aby se někdo mohl relace zúčastnit. Další důležitou proměnnou je `votePhase`, která nám udává fázi konkrétní relace. `VotePhase` při hodnotě `false` znamená, že hlasovací fáze relace ještě nezačala a tento

stav nazývám v aplikaci jako „preparation“. Změna je pouze jednorázová, tudíž si musí moderátor rozmyslet, v jaký okamžik relaci přepne. V přípravné fázi lze pouze zvát uživatele do relace a případně si zobrazit, koho už autor pozval. V hlasovací fázi je již umožněno pozvaným účastníkům pokládat otázky a hlasovat pro otázky druhých. Poslední nezbytnou částí relace je pole čísel `questionKeys`, které uchovává ID jednotlivých otázek, čímž získáme přehled o všech otázkách relace, kterými uživatelé přispějí.

Mapování relací

K uložení všech relací byla použita datová struktura `mapping`, která funguje jako asociativní pole nebo slovník v jiných programovacích jazycích. Byla vybrána z důvodu, protože poskytuje efektivní přístup k datům při jejich velkém objemu pomocí mapování ID relace na strukturu relace. K mappingu byla využita proměnná `numRelations`, která slouží jako čítač relací a uchovává jejich celkový počet.

```
mapping(uint=>Relation) relations;
```

Funkce addRelation

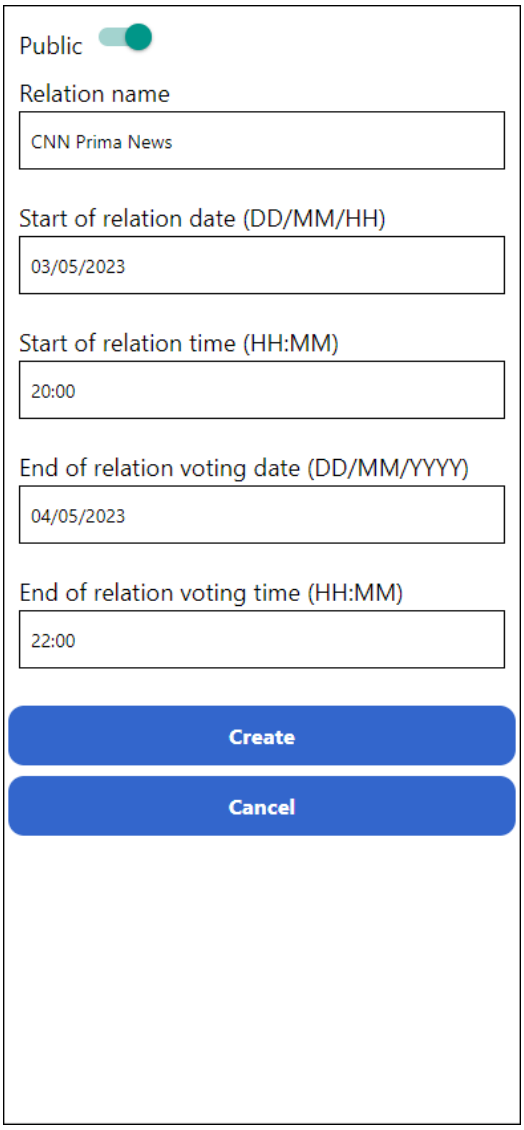
Začněme tedy první funkcí, která samotnou relaci vytvoří. Moderátor při tvorbě relace zadává, zda je relace veřejná či soukromá, dále její název, její začátek a čas, do kdy lze zadávat v relaci otázky. Čas vytvoření relace je stanoven podle hodnoty `block.timestamp`, ID je podle proměnné `numRelations`. Počáteční fází je vždy ta přípravná. Moderátor vytváří relace pomocí rozhraní aplikace znázorněného na obrázku 6.6. Funkce má speciální typ `payable`, který označuje, že se jedná o transakci, která umožňuje poslat ether do smart kontraktu. Spuštění funkce je podmíněno moderátorovou ověřenou identitou a zaplacením požadované částky za vytvoření relace. Výše požadované částky je pevně nastavena v aplikaci.

Funkce addUsersToRelation

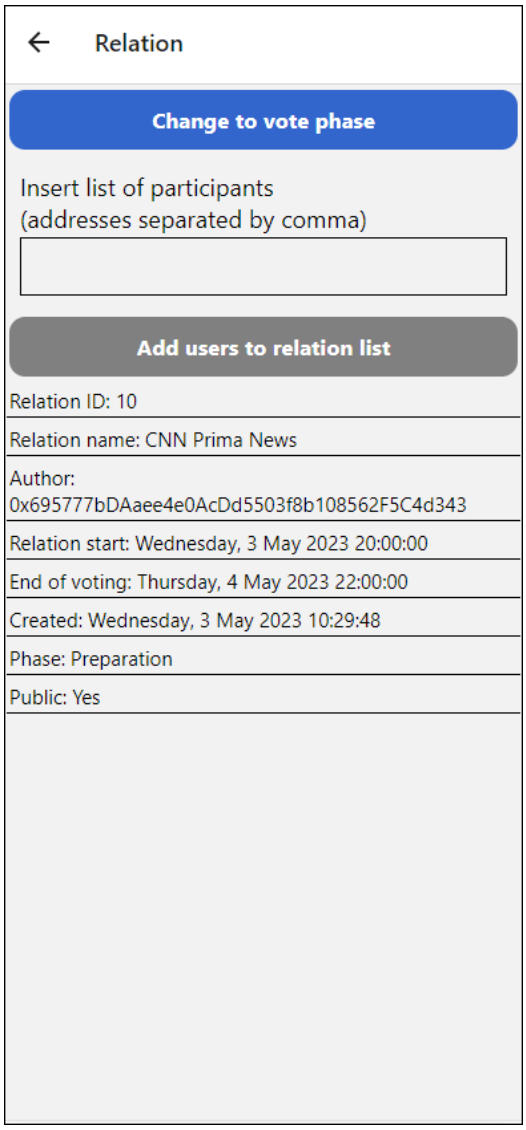
Nyní se přesuneme k funkci přidávající uživatele do soukromé relace. Pro úspěšný průběh funkce musí být odesílatel transakce autorem relace a dále se musí relace nacházet v přípravné fázi a být veřejná. Při splnění všech podmínek lze následně přidat uživatele jednotlivě nebo jako pole adres. Pole vytváří sama aplikace z uživatelského vstupu, ve kterém jsou jednotlivé adresy oddělené čárkou. Obrázek 6.7 znázorňuje finální aplikaci, kde je zobrazen rámeček na vložení adres. Pod ním je tlačítko na odeslání transakce. Šedá barva symbolizuje jeho nefunkčnost a to z důvodu, že se jedná o veřejnou relaci, ve které nelze uživatele přidávat. Zároveň může mít šedou barvu i z důvodu, že se relace nachází v hlasovací fázi. V opačném případě by bylo tlačítko zabarveno modře. Vložené adresy se zde mohou opakovat z toho důvodu, že by mohla být transakce při kontrole stejných adres neúspěšná a bylo by zbytečně komplikované volat transakci znovu. Jelikož všechny ostatní funkce zmíněné v této sekci mají kontrolu ověření adres, tak zde není potřeba stejné adresy kontrolovat.

Funkce changeRelationPhase

Zde se dostáváme k poslední funkci přímo ovlivňující relace, a to změnu jejich fází. Jak již bylo zmíněno na začátku této sekce, jde pouze o jednorázovou záležitost. V případě, že by se umožnilo moderátorovi měnit fáze opakovaně, mohl by nastat problém s tím,



Obrázek 6.6: Rozhraní aplikace pro vytvoření nové relace.



Obrázek 6.7: Zobrazení konkrétní relace z pohledu moderátora.

že tímto způsobem bude moci omezovat účastníky, aby přidávali nové otázky a hlasovali, jelikož v přípravné fázi lze pouze zvát účastníky. Jak je vidět na obrázku 6.7, tlačítko je zabarveno modře, což znamená, že ho lze stisknout. Poté, co se fáze změní, tlačítko zešedne a moderátor již bude vědět, že změna fáze proběhla.

6.4.3 Otázky

Nakonec se dostáváme k samotným otázkám v relacích a k hlasování s nimi spojeném. Datová struktura otázek je sama o sobě poměrně strohá. Jak je vidět na úryvku kódu níže, tvoří ji pouze text, pole adres uchovávající hlasy dané otázky a autor otázky.

```
struct Question {
    string text;
    address[] votes;
    address author;
}
```

Mapování otázek

Jednotlivé dotazy jsou uloženy do mappingu `questions`, stejně jako tomu je u relací. ID dotazů, které jsou uloženy v relacích v poli `questionKeys`, po vložení do mappingu `questions` vrátí konkrétní otázky, které lze posléze uživateli zobrazit. Není proto potřeba ani duplicitně ukládat jejich samotné ID.

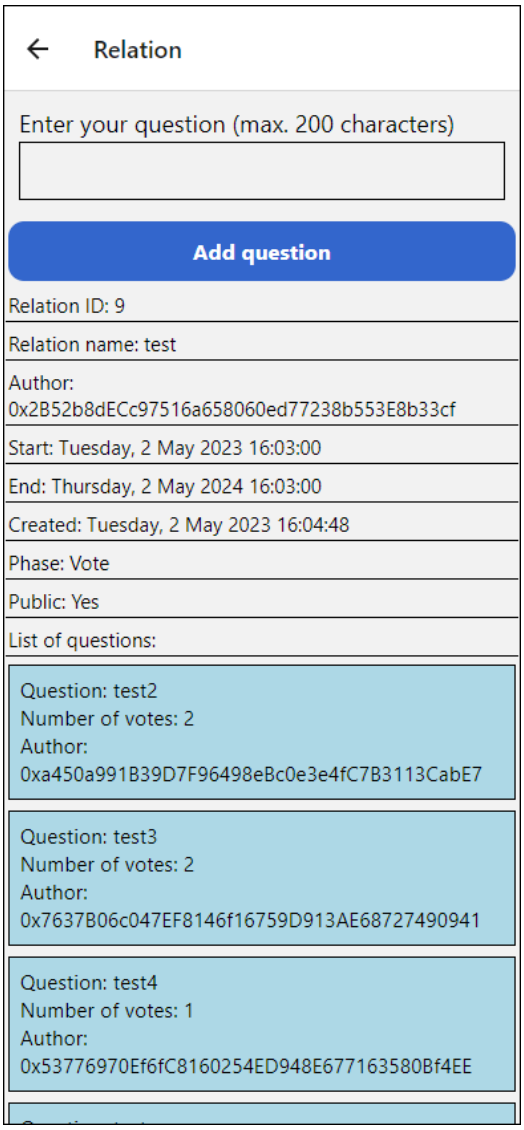
```
mapping(uint=>Question) questions;
```

Funkce addQuestion

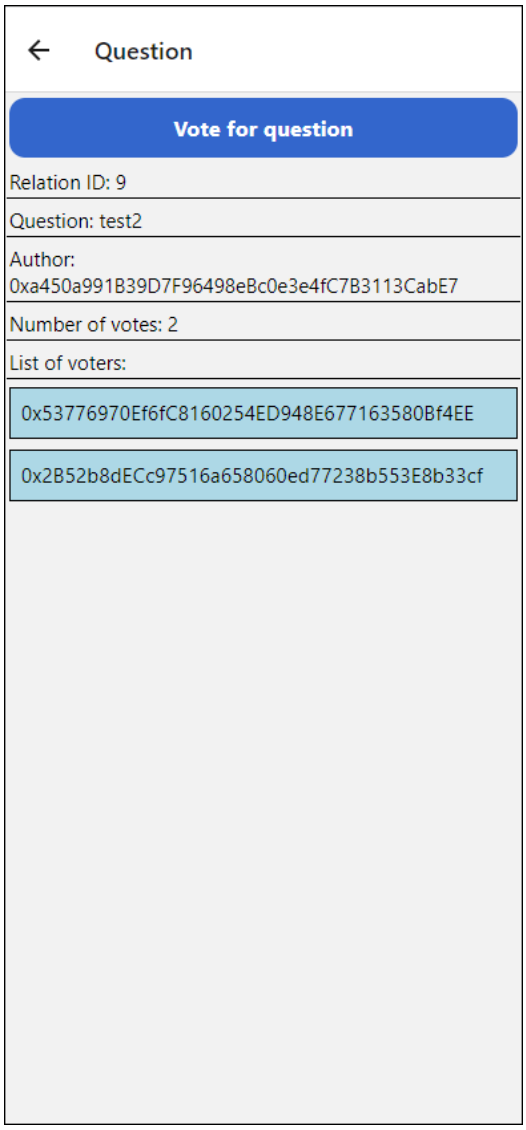
`AddQuestion` je funkcí zodpovědnou za vytvoření otázek a jejich přidávání do konkrétních relací. Její důležitou součástí je kontrola, zda uživatel již zadal otázku do relace. Ta má mít společně s ověřením uživatele za cíl zamezení spamu otázek v relacích. Uživatel se tak musí rozhodnout, jakou otázku chce položit a předem si ji dobře promyslet. V případě soukromé relace probíhá kontrola odesílatele, jestli se nachází mezi pozvanými účastníky. Aplikace Voter zde ještě kontroluje délku zadané otázky, která může mít nanejvýš 200 znaků. Na obrázku 6.8 je zobrazena konkrétní relace s možností přidat otázku. Níže je vidět seznam otázek, který je seřazen podle počtu hlasů.

Funkce voteForQuestion

Druhou funkcí operující s otázkami je `voteForQuestion`. Jejím úkolem je zařídit přidání hlasu (adresy odesílatele) ke konkrétní otázce. Hlasy zajišťují oblíbenost otázky, na jejichž základě si může moderátor vybrat nejoblíbenější otázky z publika. V případě soukromé relace zde probíhá stejná kontrola jako při vytváření otázek. Navíc se zde kontroluje, zda již uživatel hlasoval pro danou otázku a zamezilo se tak duplicitnímu hlasování. Zobrazení hlasů a možnost hlasování lze vidět na obrázku 6.9.



Obrázek 6.8: Zobrazení konkrétní relace s otázkami z pohledu uživatele.



Obrázek 6.9: Zobrazení konkrétní otázky se seznamem hlasujících z pohledu uživatele.

Kapitola 7

Evaluace a zhodnocení

Procesem testování by měl správně procházet vývoj všech aplikací. Proto zde představím postupy, které byly uplatněny při testování výsledných aplikací a smart kontraktů. Testy probíhaly již od prvního návrhu smart kontraktu a byly postupně upravovány podle potřeby. V druhé části kapitoly si probereme odchylku od zadání a nedostatky a možné budoucí rozšíření aplikace a nakonec zhodnotíme výsledek práce.

7.1 Testování

Unit testy

První způsob testování se zaměřoval na funkčnost smart kontraktů `Queans` a `identityRegistry`. Testy probíhaly na základě změn smart kontraktu `Queans`, které byly potřeba provést po jeho úpravě, a měly za úkol kontrolovat, že všechny funkce se chovají správným způsobem. Jednalo se hlavně o kontrolu stavů, ve kterých by funkce neměla projít, jelikož nesplnila některou z podmínek pro její korektní průchod. K tomu byly využity unit testy s použitím platformy Truffle Suite. Unit testy jsou rozděleny do 2 částí, první se zaměřuje na smart kontrakt `identityRegistry`, jehož správné chování je nezbytné pro fungování smart kontraktu `Queans`, který se nachází v druhé části testů. Testují se všechny funkce, které mohou měnit stav blockchainu. Nejčastěji odhalené chyby byly zpravidla špatně zvolené relační operátory nebo chybná práce s datovými strukturami.

Konzole a pomocné nástroje

Druhým způsobem testování, které bylo použito během tvorby aplikací, je zpětná vazba zejména z chybových výpisů v konzoli, statusu transakcí u nástroje Etherscan či Truffle nebo peněženky MetaMask. Konzolové výpisy společně s nástrojem Etherscan dokázaly odhalit, v jaké části funkce nastala chyba, která byla omylem zanesena do aplikace. Peněženka MetaMask se prakticky vždy, kdy má dojít k neúspěšné blockchainové transakci, dotazuje, zda má být konkrétní transakce provedena. Tudíž ještě před utracením prostředků dochází k varování uživatele. I přes úspěšné unit testy docházelo v aplikaci k chybnému zapsání kódu pro volání funkcí smart kontraktu, kde nebyly zpravidla rozlišeny metody `call` od metody `send` či byly zcela opomenuty. Chybové konzolové výpisy odhalovaly hlavně špatné použití komponent frameworku React Native.

7.2 Diskuze

Nejprve bych zmínil, že během vývoje došlo k odchylce od zadání práce, kde namísto mobilní aplikace byly implementovány aplikace webové, k čemuž jsem se uchýlil ze dvou důvodů.

Prvním důvodem bylo uvědomění si, že proces pokládání otázek a hlasování by měl být co možná nejsnažší, aby neodradil uživatele. K tomu mi přispěla analýza současných aplikací ze sekce 5.2, kde jsem zjistil, že všechny existující aplikace jsou webové, díky čemuž si uživatel nemusí stahovat žádnou aplikaci a rychleji se může do relace zapojit.

Druhým důvodem, který vedl k tomuto rozhodnutí, se stal fakt, že peněženka MetaMask vykazovala nekonzistentní chování při připojování se k aplikaci, což by vedlo k špatné uživatelské zkušenosti. Na obrázku 7.1 je vyobrazen screenshot konzole a emulátoru mobilního telefonu, kde se aplikace připojuje k peněžence, ale nedojde k samotnému připojení. Při pokusu o opětovné připojení se vyskytují chybová hlášení `Keys are not exchanged` a následně `Wrong id`. Podle vlastního testování se toto chování ještě zhoršilo po importu knihovny `web3`. Přestože jsem se několikrát po delší čekací době úspěšně připojil do peněženky, spojení s peněženkou bylo záhy přerušeno.

Mimo MetaMask SDK jsem vyzkoušel i službu WalletConnect, která bohužel ve verzi 1.x.x pro React Native nefungovala a již není aktualizovaná a verze 2.x.x pro React Native ještě není dokončena, kde termín dokončení je stanoven na konec června letošního roku 2023. V případě, že bude jedna z těchto možností funkční, není problém dokončit práci podle stanoveného zadání. Úprava by spočívala pouze v úpravě funkce `load` ve zdrojovém kódu decentralizovaných aplikací, kde by se podle podmínky níže byla aplikace schopná napojit přímo na aplikaci mobilní peněženky MetaMask. Všechny ostatní funkce nepotřebují žádnou dodatečnou změnu a měly by být plně funkční.

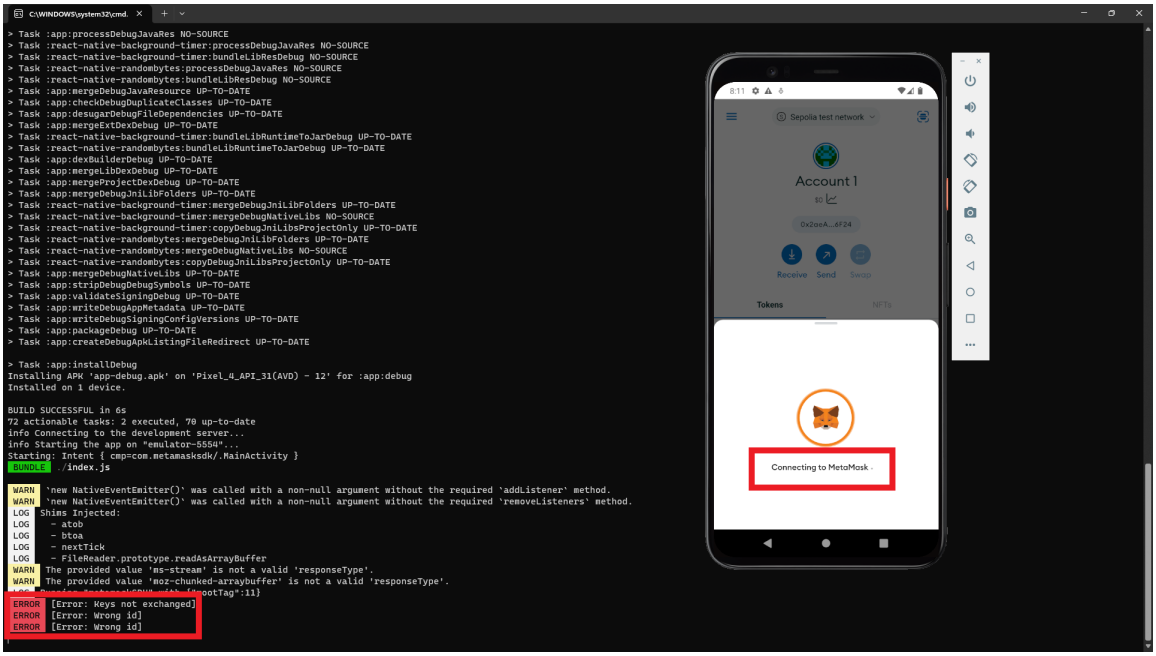
```
if (Platform.OS === 'web') {
    //aktuální způsob připojení
} else if (Platform.OS === 'ios' || Platform.OS === 'android') {
    //nové připojení na mobilní aplikaci peněženky MetaMask
}
```

7.2.1 Nedostatky současných aplikací

Současný stav aplikací provází několik nedostatků, které by měly být opraveny. První nedostatek spočívá v samotném designu aplikací. Uživatelské prostředí nevypadá dvakrát přívětivě a rozhodně by stálo za to ho upravit tak, aby bylo pro uživatele vlídnější. K dalšímu nedostatku bych zařadil způsob zadávání dat a času při tvorbě relace. Uživatel se může snadno přepsat a udělat tak chybu, která již nepůjde opravit. Nicméně se mi nepodařilo nalézt vhodnou knihovnu, která by byla kompatibilní zároveň pro mobilní i webové aplikace a nabízela jednoduchou volbu data a času pomocí jednotného vstupu. Jako poslední nedostatek je nutnost manuálního přepnutí fáze relace, aby bylo uživatelům umožněno pokládání otázek a hlasování. Moderátorovi se může snadno stát, že zapomene fázi změnit a do relace se následně nedostanou žádné otázky.

7.2.2 Budoucí rozšíření aplikací

V případě, že by aplikace měly dojít do stavu, kdy by mohly být použity širší veřejností, je třeba je ještě rozšířit a opravit nedostatky zmíněné v předchozí sekci 7.2.1. Přestože aplikace fungují, jak mají, pro širší adopci je zejména důležitý jejich vzhled, který by bylo



Obrázek 7.1: Problémy při připojení k mobilní peněžence MetaMask.

nutné přepracovat. Dále by bylo dobré přidat filtrování relací podle toho, zda se jedná o nadcházející, probíhající či ukončené relace. Pro zvýšení atraktivnosti je dalším krokem zavedení motivačního modelu, který je popsán v sekci 6.2. Ten by motivoval uživatele k větší interakci s vidinou finanční nebo jiné odměny, která by byla důležitá zejména pro rozšíření aplikací mezi více uživatelů a tím by mohly získat větší popularitu. Další možností rozvoje je okamžitá aktualizace dat po provedení úspěšné blockchainové transakce, která funguje aktuálně při přidání nové relace, nicméně např. při přidání hlasů se musí uživatel vrátit na přehled relací a zpět na konkrétní otázku, aby viděl nově přidaný hlas. Posledním návrhem je zrychlení načítání relací a otázek v aplikaci, jelikož při jejich vysokém počtu by aktuálně mohlo načítání trvat delší dobu, protože aplikace načítá všechny relace i otázky relací současně.

7.3 Bezpečnostní analýza

V této sekci se podíváme na situace, které by mohly nastat v případě zneužití práv, jež mají jednotliví aktéři aplikací.

7.3.1 Nevhodné chování poskytovatele identit

Nejprve se podíváme na možné problémy u poskytovatele identity. Co by se stalo, kdyby se choval tak, že odmítne ověřit uživatele po předložení identity? Uživatel bude mít možnost si vybrat jakéhokoliv poskytovatele identity. V případě, že by někdo nechtěl uživatele ověřit, není problém vyhledat jiného poskytovatele a ověřit se u něho.

Další problém by mohl nastat ve chvíli, kdy poskytovatel identity bude přidávat ověřené uživatele s úmyslem poškodit celou aplikaci nebo relaci. Mohl by tak vytvořit nespočet uživatelů, kteří by mohli zahlcovat relace nevhodnými dotazy či neustále vytvářet nové relace, které by znepřehledňovaly aplikace. Na tyto problémy navážeme v sekci 7.3.2.

Je přidání centralizovaného poskytovatele identity snížením decentralizace všech aplikací? Dalo by se říci, že ano, ale jen trochu. Aplikace pro správu identit je jen jednou z částí Q&A relací. Centralizace nebude tak velká, protože poskytovatelů identity bude celá řada. Mimo to známe jejich identitu a v případě jejich nekorektního chování by bylo možné s nimi zúčtovat.

7.3.2 Nevhodné chování uživatelů

Nyní se přesuneme k uživatelům a jejich tvorbě nevhodných otázek, což by mohlo zčásti souviset i se spamem. Prvním prvkem, který částečně limituje množství těchto zpráv, je maximální počet otázek, který mohou uživatelé položit, s aktuální hodnotou 1. Díky tomu, že aplikace nikomu neumožňuje cenzurovat žádné otázky, se může stát, že se v relacích nevhodné dotazy objeví. V případě, že by nastal problém s poskytovatelem identit popisovaný v předchozí sekci 7.3.1, který by ověřil účty bez poskytnutí identity, se zde nachází další prvek zamezující nevhodnému chování v podobě ceny za položení otázky (aktuálně je cena na nízké úrovni, aby se dalo aplikaci vyzkoušet, ale v případě nasazení aplikace by se zavedla cena, která by nebyla tak nízká). Uživatel by následně musel vynaložit určité množství prostředků, aby vytvořil více otázek. To však nemění nic na tom, že pokud pro tyto otázky nebudou ostatní uživatelé hlasovat, tak skončí na nižších pozicích a moderátor se o ně nebude zajímat.

Dalším typem nevhodného chování by mohlo být nadměrné vytváření relací. Proti tomuto problému by měla zasáhnout opět cena za vytvoření relace. Aby neměl uživatel problém s vyhledáním správné relace, měl by mu pomoci moderátor nebo organizátor relace, který uživateli poskytne ID relace, podle kterého ji následně jednoduše vyhledá. Podle adresy autora by si měl uživatel taktéž zkontrolovat, že se jedná o tu relaci, za kterou se vydává.

7.4 Zhodnocení aplikací

Mezi výhody aplikací bych zařadil jejich jednoduché rozhraní, které není složité na pochopení. Dále, že se jedná o funkční projekt, který úspěšně demonstruje, jak by aplikace mohla vypadat a být použita. Třetí výhodou je fakt, že aplikace byly nakonec vytvořeny jako webové a nabízí tak snažší použití pro uživatele. Navíc jsou dostupné i na počítači, což není rozhodně na škodu.

Jako nevýhody lze zařadit již zmíněné nedostatky, které se objevily, zejména nutnost manuálního přepnutí fáze relace, na kterou si musí aktuálně moderátor dávat pozor. S tím je následně spojena i nemožnost pokládat otázky do relace a hlasovat. Nevýhodou také může být nepřehlednost zobrazení relací při jejich vyšším počtu, kde ale funguje možnost filtrování podle ID relace, které by měl uživatel obdržet od moderátora.

Kapitola 8

Závěr

Cílem práce bylo vytvořit decentralizovanou mobilní aplikaci poskytující Q&A sekci s využitím blockchainu. Mezi hlavní přednosti aplikací se řadí odolnost vůči cenzuře, kterou nabízí propojení s blockchainem a zamezení spamu, čehož bylo docíleno pomocí propojení se systémem pro správu identit poskytnutým od doktora Homoliaka.

Před začátkem práce bylo zapotřebí prostudovat problematiku blockchainu, smart kontraktů a decentralizovaných aplikací, která mi pomohla objasnit, jak jsou na sobě zmíněné pojmy závislé. Následně jsem studoval programovací prostředí vhodné pro mobilní aplikace, které mi pomohlo k rozhodnutí, jak aplikace tvořit. Na konci teoretické části jsou popsány použité technologie, které byly nezbytnou součástí při tvorbě výsledných aplikací.

Na začátku praktické části popisuji již existující centralizované aplikace a zhodnocuji výhody a nevýhody jednotlivých řešení. Poté se již přesouvám k návrhu aplikace a protokolu, podle kterého následně proběhla implementace aplikací, a samotnému popisu funkcí smart kontraktů. V úplném závěru práce je popsán způsob testování společně s odchylkou práce včetně jejích nedostatků a možnosti k rozšíření výsledných aplikací.

Výsledek práce byl splněn s rozdílem, že místo mobilní aplikace byly vytvořeny aplikace webové. K tomu došlo ze dvou důvodů. Prvním je, aby se uživatelé mohli rychleji a snáze zapojit do relace. Druhý důvod je aktuální nekonzistentní chování kryptoměnové peněženky MetaMask. Nakonec bych změnu označil za pozitivní a výsledné aplikace díky ní nabízejí lepší vlastnosti.

Do budoucna bych rád otestoval aplikace v reálných podmínkách a zkusil je nabídnout organizátorům přednášek, od kterých bych následně dostal cennou zpětnou vazbu. Jako rozšíření by bylo do aplikace vhodné implementovat filtrování relací podle různých klíčů, aby bylo uživatelům usnadněno vyhledávání relací nejenom podle jejich ID a zobrazení relací by se následně stalo přehlednějším.

Bakalářská práce mě naučila větší trpělivosti a samostatnosti při řešení různých problémů. Také jsem si osahal postupy tvorby mobilních aplikací, ve kterých bych v budoucnu rád pokračoval. Největším přínosem pro mne však bylo hlubší pochopení problematiky blockchainu, protože v něm vidím veliký potenciál a osobně mě tato problematika zajímá.

Literatura

[1] Abrol, A. *What Are Blockchain Nodes? Detailed Guide* [online]. 2022 [cit. 2023-03-16]. Dostupné z: https://www.blockchain-council.org/blockchain/blockchain-nodes/.

[2] AhaSlides. *The Best Free Presentation Software Online | AhaSlides* [online]. [cit. 2023-04-12]. Dostupné z: https://ahaslides.com/.

[3] Alchemy Insights. *Vyper - Web3 Languages - Alchemy* [online]. [cit. 2023-04-4]. Dostupné z: https://www.alchemy.com/dapps/vyper.

[4] Alchemy Insights. *What is the Sepolia testnet?* [online]. [cit. 2023-04-19]. Dostupné z: https://www.alchemy.com/overviews/sepolia-testnet.

[5] AltexSoft. *Pros and Cons of Flutter App Development | AltexSoft* [online]. 2018 [cit. 2023-04-6]. Dostupné z: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-flutter-app-development/.

[6] Amazon Web Services. *Mobile Application Development* [online]. [cit. 2023-04-6]. Dostupné z: https://aws.amazon.com/mobile/mobile-application-development/.

[7] Asociace pro komunikační nástroje a internet věcí. *FinTech-Roadmap* [online]. [cit. 2023-03-25]. Dostupné z: https://www.ctit.cz/wp-content/uploads/2022/12/FinTech-Roadmap-2.0.pdf.

[8] Bashir, I. *Mastering Blockchain* [online]. 1. vyd. Birmingham: Packt Publishing Ltd, 2017 [cit. 2023-02-25]. ISBN 9781839211379. Dostupné z: https://books.google.cz/books?hl=cs&lr=&id=urkrDwAAQBAJ&oi=fnd&pg=PP1&dq=BASHIR,+Imran.+Mastering+Blockchain&ots=Iwfo7g8yYJ&sig=HnIwXoDFe5iFQQCmmjSbjY8DCKg&redir_esc=y#v=onepage&q=BASHIR%2C%20Imran.%20Mastering%20Blockchain&f=false.

[9] Binance Academy. *Co je Etherscan a jak ho používat? | Binance Academy* [online]. [cit. 2023-04-20]. Dostupné z: https://academy.binance.com/cs/articles/what-is-etherscan-and-how-to-use-it.

[10] Binance Academy. *What Is a Blockchain Consensus Algorithm?* [online]. 2018 [cit. 2023-03-20]. Dostupné z: https://academy.binance.com/en/articles/what-is-a-blockchain-consensus-algorithm.

[11] Cai, W., Wang, Z., Ernst, J. B., Hong, Z., Feng, C. et al. Decentralized Applications: The Blockchain-Empowered Software System. *IEEE Access* [online]. 2018, sv. 6, s. 53019–53033, [cit. 2023-03-25]. DOI: 10.1109/ACCESS.2018.2870644. Dostupné z: https://ieeexplore.ieee.org/document/8466786.

[12] CIBEREXPLOSION. *Solidity - Special Variables - GeeksforGeeks* [online]. [cit. 2023-04-1]. Dostupné z: https://www.geeksforgeeks.org/solidity-special-variables/.

[13] CISCO SYSTEMS. *Slido - Audience Interaction Made Easy* [online]. [cit. 2023-04-12]. Dostupné z: https://www.slido.com/.

[14] COINMARKETCAP. *Ethereum price today, ETH to USD live, marketcap and chart | CoinMarketCap* [online]. [cit. 2023-03-21]. Dostupné z: https://coinmarketcap.com/currencies/ethereum/.

[15] CONSENSYS SOFTWARE. *Ganache | Overview - Truffle Suite* [online]. [cit. 2023-04-19]. Dostupné z: https://trufflesuite.com/docs/ganache/.

[16] CONSENSYS SOFTWARE. *Truffle | Overview - Truffle Suite* [online]. [cit. 2023-04-19]. Dostupné z: https://trufflesuite.com/docs/truffle/.

[17] CRAIG, W. *Native App vs. Mobile Web App: A Quick Comparison - WebFX* [online]. [cit. 2023-04-9]. Dostupné z: https://www.webfx.com/blog/web-design/native-app-vs-mobile-web-app-comparison/.

[18] DANIELSSON, W. *React Native application development* [online]. 2016 [cit. 2023-04-8]. Dostupné z: https://www.diva-portal.org/smash/get/diva2:998793/FULLTEXT02.pdf.

[19] ETHEREUM. *Pooled staking | ethereum.org* [online]. [cit. 2023-03-21]. Dostupné z: https://ethereum.org/en/staking/pools/.

[20] ETHERSCAN. *Ethereum (ETH) Blockchain Explorer* [online]. [cit. 2023-04-20]. Dostupné z: https://etherscan.io/.

[21] EXPO. *Expo Documentation* [online]. [cit. 2023-04-20]. Dostupné z: https://docs.expo.dev/.

[22] FRANKENFIELD, J. *What Is Proof of Work (PoW) in Blockchain?* [online]. 2016 [cit. 2023-03-21]. Dostupné z: https://www.investopedia.com/terms/p/proof-work.asp.

[23] FRANKENFIELD, J. *What Does Proof-of-Stake (PoS) Mean in Crypto?* [online]. 2017 [cit. 2023-03-21]. Dostupné z: https://www.investopedia.com/terms/p/proof-stake-pos.asp.

[24] FRANKENFIELD, J. *Decentralized Applications (dApps): Definition, Uses, Pros and Cons* [online]. 2018 [cit. 2023-03-28]. Dostupné z: https://www.investopedia.com/terms/d/decentralized-applications-dapps.asp.

[25] GOCARDLESS. *How to validate Bitcoin transactions* [online]. 2021 [cit. 2023-03-4]. Dostupné z: https://gocardless.com/en-us/guides/posts/bitcoin-transaction-verification/#what-does-bitcoin-transaction-confirmation-do.

[26] GOOGLE. *Flutter - Build apps for any screen* [online]. [cit. 2023-04-6]. Dostupné z: https://flutter.dev/.

[27] GOOGLE DEVELOPERS. *Meet Android Studio | Android Developers* [online]. [cit. 2023-05-1]. Dostupné z: https://developer.android.com/studio/intro.

[28] Gunawardhana, L. K. P. D. Native or Web or Hybridwhich is better for Mobile Application. *Turkish Journal of Computer and Mathematics Education* [online]. 2021, sv. 12, č. 6, s. 4643–4649, [cit. 2023-04-5]. Dostupné z: https://www.turcomat.org/index.php/turkbilmat/article/view/8450/6630.

[29] guptavivek0503. *Pros, Cons, and Examples of dApp* [online]. 2022 [cit. 2023-03-26]. Dostupné z: https://www.geeksforgeeks.org/pros-cons-and-examples-of-dapp/.

[30] Herembourg, K. *8 Most Popular Mobile App Development Frameworks in 2023* [online]. 2022 [cit. 2023-04-5]. Dostupné z: https://www.purchasely.com/blog/mobile-app-frameworks.

[31] Homoliak, I., Breitenbacher, D., Hujnak, O., Hartel, P., Binder, A. et al. SmartOTPs: An Air-Gapped 2-Factor Authentication for Smart-Contract Wallets. In: *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. New York, NY, USA: Association for Computing Machinery, 2020, s. 145–162 [cit. 2023-04-15]. AFT '20. DOI: 10.1145/3419614.3423257. ISBN 9781450381390. Dostupné z: https://doi.org/10.1145/3419614.3423257.

[32] IBM. *Introduction to Mobile Application Development | IBM* [online]. [cit. 2023-04-3]. Dostupné z: https://www.ibm.com/topics/mobile-application-development.

[33] Infura. *Frequently Asked Questions* [online]. [cit. 2023-04-19]. Dostupné z: https://www.infura.io/faq/general.

[34] JavaTpoint. *What is Xamarin - Javatpoint* [online]. [cit. 2023-04-11]. Dostupné z: https://www.javatpoint.com/what-is-xamarin.

[35] Joshua, carrabre, d1onys1us, Wackerow, P., vanhooser north et al. *Ethereum Virtual Machine (EVM) | ethereum.org* [online]. [cit. 2023-03-31]. Dostupné z: https://ethereum.org/en/developers/docs/evm/.

[36] Lashkar, B. a Musilek, P. A Comprehensive Review of Blockchain Consensus Mechanisms. *IEEE Access* [online]. 2021, sv. 9, s. 43620 – 43652, [cit. 2023-03-20]. DOI: 10.1109/ACCESS.2021.3065880. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9376868.

[37] Liang, Y.-C. *The structure of a Blockchain* [online]. [cit. 2023-03-24]. Dostupné z: https://www.researchgate.net/figure/The-structure-of-a-Blockchain-A-block-is-composed-of-a-header-and-a-body-where-a-header_fig1_337306138/.

[38] Malik, A., Gautam, S., Abidin, S. a Bhushan, B. Blockchain Technology-Future Of IoT: Including Structure, Limitations And Various Possible Attacks. 2019, sv. 1, s. 1100–1104, [cit. 2023-04-16]. DOI: 10.1109/ICICICT46008.2019.8993144. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8993144.

[39] Marchuk, A. *Native vs Cross-Platform Development: How to Choose* [online]. [cit. 2023-04-2]. Dostupné z: https://www.uptech.team/blog/native-vs-cross-platform-app-development.

[40] Mentimeter AB. *Interactive presentation software - Mentimeter* [online]. [cit. 2023-04-12]. Dostupné z: https://www.mentimeter.com/.

[41] Meta Platforms. *React Native · Learn once, write anywhere* [online]. [cit. 2023-04-8]. Dostupné z: https://reactnative.dev/.

[42] MetaMask. *The crypto wallet for Defi, Web3 Dapps and NFTs | MetaMask* [online]. [cit. 2023-04-20]. Dostupné z: https://metamask.io/.

[43] Microsoft. *Xamarin | Open-source mobile app platform for .NET* [online]. [cit. 2023-04-5]. Dostupné z: https://dotnet.microsoft.com/en-us/apps/xamarin.

[44] Mohanta, B. K., Panda, S. S. a Jena, D. An Overview of Smart Contract and Use Cases in Blockchain Technology. [online]. 2018, s. 1–4, [cit. 2023-03-23]. DOI: 10.1109/ICCCNT.2018.8494045. Dostupné z: https://ieeexplore.ieee.org/document/8494045.

[45] Moralis Blog. *What are Upgradable Smart Contracts? Full Guide - Moralis Web3 | Enterprise-Grade Web3 APIs* [online]. [cit. 2023-03-30]. Dostupné z: https://moralis.io/what-are-upgradable-smart-contracts-full-guide/#what-are-upgradable-smart-contracts.

[46] Mrvova, K. *How to Facilitate a Successful Q&A Session - Slido Blog* [online]. 2021 [cit. 2023-04-12]. Dostupné z: https://blog.slido.com/fix-your-qa-session/.

[47] Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System* [online]. 2008 [cit. 2023-02-17]. Dostupné z: https://bitcoin.org/bitcoin.pdf.

[48] Narayanan, A., Bonneau, J., Felten, E., Miller, A. a Goldfeder, S. *Bitcoin and Cryptocurrency Technologies* [online]. 2016 [cit. 2023-03-2]. Dostupné z: https://www.lopp.net/pdf/princeton_bitcoin_book.pdf.

[49] Nawrocki, P., Wrona, K., Marczak, M. a Sniezynski, B. A Comparison of Native and Cross-Platform Frameworks for Mobile Applications. *Computer* [online]. 2021, sv. 54, č. 3, s. 18–27, [cit. 2023-04-2]. DOI: 10.1109/MC.2020.2983893. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9378923.

[50] Pennella, L., Smith, C., Dan, HaoTian, Awosika, E. et al. *PROOF-OF-STAKE (POS)* [online]. [cit. 2023-03-21]. Dostupné z: https://ethereum.org/cs/developers/docs/consensus-mechanisms/pos/.

[51] Raval, S. *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology* [online]. 1st. O'Reilly Media, Inc., 2016 [cit. 2023-03-28]. ISBN 1491924543. Dostupné z: https://books.google.cz/books?id=fvywDAAAQBAJ&printsec=frontcover&redir_esc=y#v=onepage&q&f=false.

[52] Rehman, J. *Centralized-vs-decentralized-vs-distributed-processing* [online]. [cit. 2023-03-22]. Dostupné z: https://www.itrelease.com/2017/11/difference-centralized-decentralized-distributed-processing/.

[53] S., J. *Blockchain: What are nodes and masternodes?* [online]. 2018 [cit. 2023-03-15]. Dostupné z: https://medium.com/coinmonks/blockchain-what-is-a-node-or-masternode-and-what-does-it-do-4d9a4200938f.

[54] Solidity Team. *Solidity Programming Language | The Solidity language portal is a comprehensive information page for the Solidity programming language. It features documentation, binaries, blog, resources & more* [online]. [cit. 2023-04-5]. Dostupné z: https://soliditylang.org/.

[55] Sriman, B., Ganesh Kumar, S. a Shamili, P. *Blockchain Technology: Consensus Protocol Proof of Work and Proof of Stake* [online]. Singapore: Springer Singapore, 2021 [cit. 2023-03-20]. 395 - 406 s. ISBN 978-981-15-5566-4. Dostupné z: https://link.springer.com/chapter/10.1007/978-981-15-5566-4_34#citeas.

[56] Tuama, D. O. *What is the Difference Between Web App & Mobile App? - Code Institute Global* [online]. 2022 [cit. 2023-04-9]. Dostupné z: https://codeinstitute.net/global/blog/web-app-vs-mobile-app/.

[57] Voshmgir, S. Disrupting governance with blockchains and smart contracts. *Strategic Change* [online]. Září 2017, sv. 26, s. 499–509, [cit. 2023-03-22]. DOI: 10.1002/jsc.2150. Dostupné z: https://onlinelibrary.wiley.com/doi/10.1002/jsc.2150.

[58] Vyper Team. *Vyper — Vyper documentation* [online]. [cit. 2023-04-4]. Dostupné z: https://docs.vyperlang.org/en/stable/.

[59] Wackerow, P., Thevenet, M.-A., Ohtamaa, M., Joshua, Richards, S. et al. *Smart contract languages | ethereum.org* [online]. [cit. 2023-04-4]. Dostupné z: https://ethereum.org/en/developers/docs/smart-contracts/languages/.

[60] Wander, M. *Blockchain landscape - Blockchain – Wikipedie* [online]. [cit. 2023-03-29]. Dostupné z: https://cs.wikipedia.org/wiki/Blockchain.

[61] Wohrer, M. a Zdun, U. Smart contracts: security patterns in the ethereum ecosystem and solidity. [online]. 2018, s. 2–8, [cit. 2023-03-25]. DOI: 10.1109/IWBOSE.2018.8327565. Dostupné z: https://ieeexplore.ieee.org/document/8327565.

[62] Wood, G. *Ethereum: A secure decentralised generalised transaction ledger* [online]. [cit. 2023-03-31]. Dostupné z: http://gavwood.com/paper.pdf.

[63] Zilliqa Research. *Scilla - Home* [online]. [cit. 2023-04-4]. Dostupné z: https://scilla-lang.org/.

[64] Zilliqa Research. *Zilliqa | Scilla is made for DeFi* [online]. [cit. 2023-04-4]. Dostupné z: https://www.zilliqa.com/language.

[65] Zoltu, M., Wackerow, P., Joshua, Jadhav, P., Kumar, N. et al. *Introduction to smart contracts | ethereum.org* [online]. [cit. 2023-04-1]. Dostupné z: https://ethereum.org/en/developers/docs/smart-contracts/.

Příloha A

Obsah přiloženého média

Přiložená SD karta obsahuje následující položky:

- latex/ – zdrojové soubory textu bakalářské práce
- Management/ – aplikace Management
  - assets/ – ikony služby Expo
  - src/ – zdrojové kódy
    - build/ – zkompilované smart kontrakty
    - contracts/ – zdrojové kódy smart kontraktů
    - migrations/ – soubor pro nasazení smart kontraktu
    - identityRegistry.js – komponenta kontraktu identityRegistry
    - package-lock.json – definice závislostí smart kontraktu
    - package.json – manifest závislostí smart kontraktu
    - truffle-config.js – konfigurační soubor frameworku Truffle
  - App.js – zdrojový kód decentralizované aplikace
  - app.json – konfigurace mobilní aplikace
  - babel.config.js – konfigurační soubor pro Babel
  - package-lock.json – definice závislostí projektu
  - package.json – manifest závislostí projektu
- Moderator/ – aplikace Moderator
  - assets/ – ikony služby Expo
  - src/ – zdrojové kódy
    - build/ – zkompilované smart kontrakty
    - contracts/ – zdrojové kódy smart kontraktů
    - migrations/ – soubor pro nasazení smart kontraktu
    - test/ – soubor s unit testy
    - package-lock.json – definice závislostí smart kontraktu
    - package.json – manifest závislostí smart kontraktu
    - queans.js – komponenta Queans kontraktu

        * truffle-config.js – konfigurační soubor frameworku Truffle
    - App.js – zdrojový kód decentralizované aplikace
    - app.json – konfigurace mobilní aplikace
    - babel.config.js – konfigurační soubor pro Babel
    - package-lock.json – definice závislostí projektu
    - package.json – manifest závislostí projektu
- Voter/ – aplikace Voter
    - assets/ – ikony služby Expo
    - src/ – zdrojové kódy
        * build/ – zkompilované smart kontrakty
        * contracts/ – zdrojové kódy smart kontraktů
        * migrations/ – soubor pro nasazení smart kontraktu
        * test/ – soubor s unit testy
        * package-lock.json – definice závislostí smart kontraktu
        * package.json – manifest závislostí smart kontraktu
        * queans.js – komponenta Queans kontraktu
        * truffle-config.js – konfigurační soubor frameworku Truffle
    - App.js – zdrojový kód decentralizované aplikace
    - app.json – konfigurace mobilní aplikace
    - babel.config.js – konfigurační soubor pro Babel
    - package-lock.json – definice závislostí projektu
    - package.json – manifest závislostí projektu
- README.md – návod ke spuštění aplikace a nasazení a otestování smart kontraktů
- video.mp4 – video demonstrující práci s aplikacemi
- xcuhan00.pdf – text bakalářské práce