



Marcin Gromadzki

Implementacja blockchain dla sieci
przedsiębiorstw

Blockchain implementation for the network
of enterprises

Praca licencjacka

Promotor: dr Agata Jolanta Filipowska

Pracę przyjęto dnia:

podpis Promotora

Kierunek: Informatyka i ekonometria

Specjalność: Informatyka w gospodarce i administracji

Poznań 2018

Spis treści

1 Wstęp	1
1.1 Motywacja	1
1.2 Cele pracy	2
1.3 Metodyka badawcza	3
1.4 Struktura pracy	3
2 Technologia Blockchain	5
2.1 Definicja blockchain	6
2.2 Istota funkcjonowania	6
2.2.1 Rekord transakcji jako nośnik informacji	8
2.2.2 Weryfikacja tożsamości oraz ochrona danych	9
2.2.3 Przetwarzanie oraz dowód pracy	10
2.3 Technologia dla blockchain	11
2.3.1 Łańcuch bloków, bloki oraz rekordy	11
2.3.2 Techniczna implementacja blockchain	13
2.3.3 Rola protokołów Peer-to-Peer	14
2.3.4 Otwartość pod względem technicznym	15
2.4 Bezpieczeństwo zastosowania blockchain	16
2.4.1 Anonimowość użytkowników i danych	16
2.4.2 Kryptografia gwarantem własności	16
2.4.3 Ustanawianie brakujących regulacji prawnych	17
2.4.4 Proces teoretyczny falsyfikacji bloków	17
2.4.5 Punkty krytyczne wraz z analizą opinii eksperckiej	18
2.5 Mankamenty realizacji rozwiązania	20

2.5.1	Publiczna dostępność zasobów	20
2.5.2	Ograniczenia skalowalności blockchain	20
2.5.3	Powolność oraz wadliwość algorytmów	21
2.5.4	Specyficzność zastosowań	22
2.5.5	Niestabilność ekosystemu narzędzi deweloperskich	22
2.6	Dostępność technologii dla zastosowań adaptujących	23
2.6.1	Globalna platforma (model Paas)	23
2.6.2	Udostępnione oprogramowanie (model SaaS)	23
2.7	Podsumowanie	24
3	Architektura oprogramowania w implementacji blockchain	26
3.1	Motywacja doboru technologii	27
3.1.1	Aspekty natury biznesowej	27
3.1.2	Aspekty dotyczące aplikacji	29
3.1.3	Koncepcja architektury	30
3.2	Full-stack JavaScript (technologia wiodąca aplikacji)	31
3.2.1	Analiza wartościowych rozwiązań alternatywnych	32
3.2.2	Natywny JavaScript i serwerowy NodeJS	33
3.3	Technologie aplikacji klienckiej	36
3.3.1	React – Single Page Application	36
3.3.2	NextJS – Server Side Rendering	38
3.3.3	Progressive Web App	39
3.3.4	React Redux/Saga – zarządzanie stanem	40
3.3.5	ImmutableJS, Reselect, Code Splitting – wydajność	41
3.3.6	Styled Components – stylowanie w kodzie JS	43
3.3.7	Flow – statyczna analiza typowania	43
3.3.8	GraphQL Client	44
3.3.9	SocketIO Client	45
3.4	Technologie aplikacji serwerowej	46
3.4.1	NodeJS Express	46

3.4.2	Mongoose ODM	46
3.4.3	Flow	47
3.4.4	GraphQL Server	47
3.4.5	SocketIO Server	48
3.5	Wspomagające narzędzia deweloperskie	48
3.5.1	Repozytoria Git, Code Review oraz Continuous Integration . . .	48
3.5.2	Testy jednostkowe	49
3.5.3	Docker – wirtualizacja środowisk deweloperskich	50
3.6	Technologie bazodanowe	51
3.6.1	MongoDB	51
3.6.2	BigchainDB	52
3.7	Kompletna architektura	53
3.8	Podsumowanie	53
4	BigchainDB – skalowalna baza danych dla blockchain	55
4.1	Motywacja oraz koncepcja rozwiązania	55
4.2	Technologiczna istota funkcjonowania	56
4.2.1	Integracja w sieci oraz model danych	56
4.2.2	Zasób jako podstawowa jednostka informacyjna transakcji . . .	57
4.2.3	Typy transakcji łańcucha bloków	59
4.2.4	Dowód funkcjonowania	60
4.3	Przykładowe przypadki użycia	61
4.4	Podsumowanie	62
5	Implementacja sieci przedsiębiorstw funkcjonujących na blockchainie	63
5.1	Inspiracja wdrażania blockchain w biznesie	64
5.2	Metodyka przeprowadzenia wdrożenia	64
5.3	Analiza problemu biznesowego	66
5.4	Architektura blockchain we wdrożeniu	71
5.5	Proces biznesowy	73
5.6	Implementacja procesu biznesowego	75
5.7	Ewaluacja procesu na przykładzie przypadku użycia	82

5.8	Empiryczna próba falsyfikacji danych sieci	86
5.9	Efekt końcowy oraz konkluzja wdrożenia	87
5.10	Podsumowanie	88
6	Podsumowanie	89

Rozdział 1

Wstęp

1.1 Motywacja

Odczuwając potrzebę podążania za zmianą, moim obowiązkiem jako informatyka oraz badacza jest poszukiwanie technologii, które mają potencjał wpływać na przyszłość społeczeństwa, zmienić naszą rzeczywistość i sposób postrzegania. Analiza wielu współczesnych trendów rozwoju IT wskazuje na niektóre istotne obszary, m.in. Internet Rzeczy (ang. IoT) lub też uczenie maszynowe (ang. machine learning), które w najbliższym czasie mogą zrewolucjonizować naszą cywilizację. W gronie tych technologii znajduje się również **blockchain**, który wybrano na temat pracy badawczej.

Technologia zdecentralizowanego zaufania oferuje świat pozbawiony pośredników, zmniejszając liczbę instytucji będących uczestnikami procesów biznesowych, usprawniając funkcjonowanie procesów, katalogując zasoby oraz rozliczając transakcje, ograniczając koszty, co przekłada się na zwiększenie zysków jednostek gospodarczych. Ponadto, stanowi ona zdecentralizowane, niezależne rozwiązanie ustalania zaufania pomiędzy podmiotami Sieci, gwarantując transparentne reguły współdziałania stron analogicznie do idei demokracji, lecz na skalę danych, bez jednostki zarządczej, będącej punktem ryzyka, korupcji oraz zepsucia w istniejącym porządku rzeczy.

Blockchain może okazać się ogromną szansą dla istniejących biznesów, działających na zasadzie rozbudowanych sieci kontrahentów komercyjnych, które operują na zasadach scentralizowanych lub nie mają zinformatyzowanych powiązań

pomiędzy sobą, pomimo współdzielenia procesu biznesowego. Technologia zdecentralizowanego zaufania ma potencjał wiele zmienić, jeśli tylko potwierdzą się jej rozpatrywane atuty, a nurtujące problemy zostaną rozwiązane.

Możliwość pracy nad blockchainem przyczyni się również do zwiększenia moich umiejętności zawodowych oraz poszerzy horyzont znajomości technologii, pozwalaając nabyć cenne doświadczenie, które z pewnością okaże się przydatne w przyszłości. Dzielenie się wiedzą sprawia mi satysfakcję, cieszę się więc, że tym sposobem mogę wykazać swój wkład w postęp nowoczesnej technologii.

1.2 Cele pracy

Obrany cel główny pracy stanowi:

Implementacja technologii blockchain w aplikacji dowodu poprawności (ang. proof of concept) dla wdrożenia w biznesowej sieci kontrahentów celem wypracowania nowej wartości dodanej.

Z którego spełnieniem wiążą się następujące cele szczegółowe:

1. Przedstawienie oraz omówienie idei technologii blockchain, jej zasad funkcjonowania, wraz z oferowaną wartością dodaną dla przedsiębiorstw.
2. Omówienie architektury oprogramowania z propozycją technologii wartościowych w zastosowaniu implementacji blockchain.
3. Analiza wybranego rozwiązania blockchain, dostępnego do wykorzystania w procesie wdrażania dla podmiotów gospodarczych.
4. Wyekstrahowanie procesu biznesowego licytacji obrazu z komercyjnego rynku sztuki oraz opracowanie aplikacji blockchain ewaluującej proces z nową wartością dodaną.
5. Walidacja funkcjonowania zdecentralizowanej sieci na podstawie próby ataku.

1.3 Metodyka badawcza

Spełnienie poszczególnych celów pracy wiązało się z wykorzystaniem szeregu metod badawczych, m.in. w rozdziałach poświęconych omawianiu idei blockchainu, architektury oprogramowania czy narzędzia blockchain posłużono się metodą **analizy literatury**. Do wyekstrahowania procesu biznesowego oraz implementacji wykorzystano metodę **badania empirycznego i analizy przypadku**, podczas gdy próba ataku na sieć blockchain ma charakter **eksperymentu**.

1.4 Struktura pracy

Praca składa się z 6 rozdziałów, w tym wstępu i zakończenia.

Pierwszy rozdział przedstawia motywację do podjęcia badań, cele, które postanowiono zrealizować, metodykę prowadzenia prac oraz strukturę poszczególnych rozdziałów.

Rozdział drugi wprowadza w koncepcję blockchain, omawia jaka jest jej idea, czym się cechuje, jakie problemy potencjalnie rozwiązuje, a także porusza kluczowe kwestie bezpieczeństwa, wady podczas próby implementacji rozwiązania dla jednostek adaptujących oraz dostępne narzędzia technologii zdecentralizowanego zaufania dla procesu wdrażania.

W rozdziale trzecim zawarto opis architektury rozwiązania wraz z umotywowaniem. Rozdział skupia się na szczegółowości architektury, od doboru technologii wiodącej, po analizę dostępnych narzędzi strony klienta oraz jemu odpowiednich zależności, a także analizę strony serwera oraz jej poszczególnych segmentów, składając się na kompletną architekturę technologiczną, która z powodzeniem może zostać zastosowana w nowatorskim systemie blockchain.

Czwarty rozdział zadeykowano wybranemu rozwiązaniu blockchain, które zostanie wykorzystane do implementacji technologii zdecentralizowanego zaufania w rozdziale następnym. Rozdział poświęcono motywacji oraz krytyce rozwiązania, omówieniu jego technologicznych aspektów funkcjonowania, a także wprowadzeniu podstawowych atrybutów, którymi cechuje się wybrane narzędzie.

Rozdział piąty poświęcono syntezie technologii zdecentralizowanego zaufania i biznesu, w postaci procesu biznesowego, którego implementacja oraz wdrożenie zostanie udowodnione w formie praktycznej - aplikacji dowodu poprawności (ang. Proof of Concept) wraz z ewaluacją oraz próbą falsyfikacji danych sieci.

Ostatni rozdział zawiera podsumowanie całokształtu pracy oraz efekty realizacji postawionych celów badawczych, z uwzględnieniem wyników wraz z komentarzem.

Rozdział 2

Technologia Blockchain

W rozdziale przedstawiono wprowadzenie do technologii blockchain, zdefiniowano jaka jest jej idea, czym się ona cechuje, jakie przyjmuje założenia i które problemy potencjalnie rozwiązuje. W wielu wypadkach wprowadzane zagadnienia bazują na mechanizmach pochodzących z kryptowalut (głównie Bitcoin, ale również Lisk, Ethereum itp.), od których technologia ta wywodzi się i potwierdzone jest jej produkcyjne funkcjonowanie na przestrzeni lat. Przedstawione przykłady dążą do uwolnienia warstwy blockchain z kryptowalut, obrazując ją jako niezależną jednostkę obracającą dowolną informacją, cenną z punktu widzenia podmiotu np. biznesu. Rozdział przeprowadza przez funkcjonowanie blockchain z minimalną informacją strony technicznej, mając na celu klarowne zrozumienie zasady działania, a następnie - w miarę czytania - rozwija (często złożone) kwestie technologiczne, istotne dla funkcjonowania rozwiązania.

W drugiej części rozdziału poruszczone kwestie wykorzystywania technologii blockchain, kluczowe elementy bezpieczeństwa, które mogą stanowić zagrożenie dla działania przedsiębiorstwa, wady z którymi deweloperzy oprogramowania będą musieli się mierzyć, aby zrealizować wymagania logiki biznesowej oraz dostępne narzędzia, na których można w praktyce implementować technologię blockchain.

2.1 Definicja blockchain

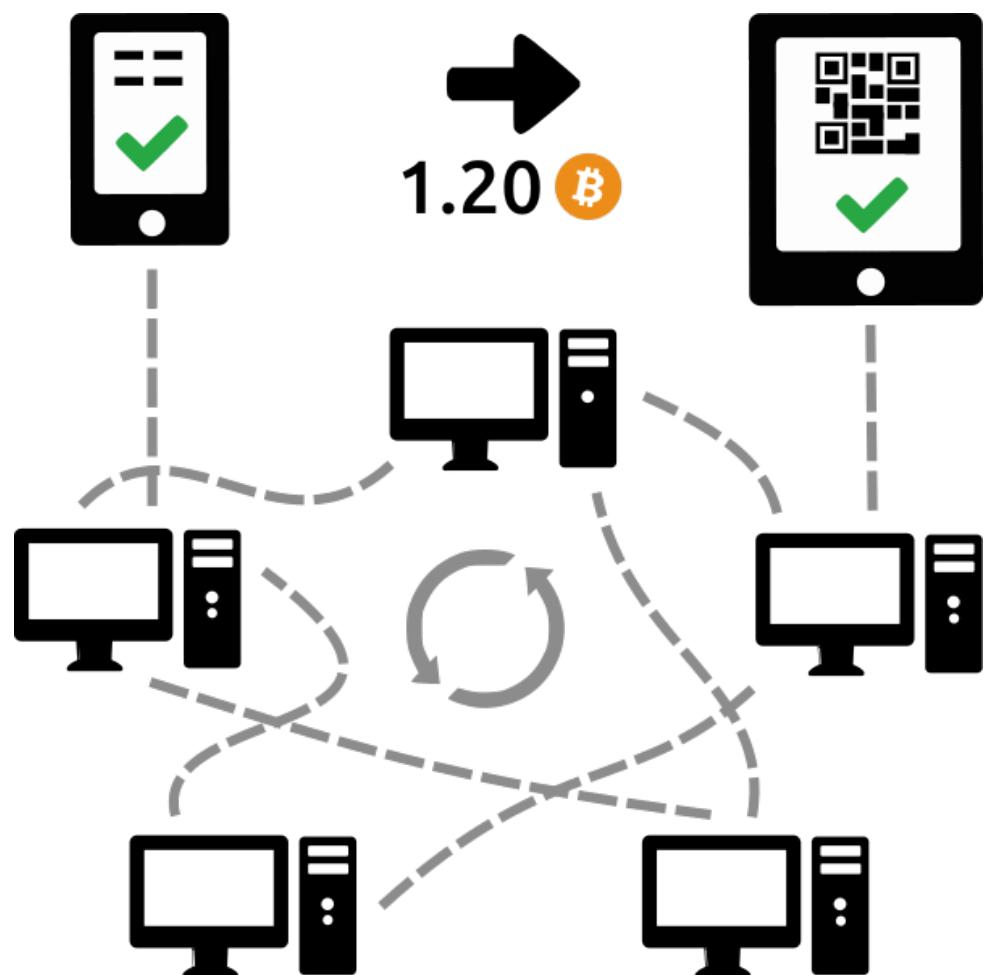
Abstrahując od kryptowaluty Bitcoin, która stanowi prototyp tej gałęzi technologii, blockchain (pol. łańcuch bloków) to rozproszony rejestr transakcji zawieranych pomiędzy stronami w ramach sieci, który jest współdzielony między wszystkich użytkowników z niego korzystających (Mearian, 2018). Blockchain umożliwia wymianę cyfrowych zasobów (np. informacji o ilości posiadanej kryptowaluty) pośród potencjalnie niezaufanymi stronami ustalając reguły, których muszą one przestrzegać w celu korzystania z platformy. Eliminuje on w ramach sieci potrzebę istnienia centralnego autorytetu, odpowiadającego za walidowanie podejmowanych akcji przez użytkownika (CBINSIGHTS, 2017).

Od strony technicznej blockchain składa się z łańcucha bloków, które zawierają dany zasób informacji (Narayanan, Bonneau, Felten, Miller i Goldfeder, 2016), przypominając standardową strukturę tabeli w relacyjnych bazach danych, która przyjmuje coraz to nowsze wiersze (w rzeczywistości jednak jest to znaczne uproszczenie). Każdy kolejny blok wskazuje swojego poprzednika oraz pozwala (po części) zapewnić o poprawności kilkunastu bloków wstecz dając tym samym gwarancję spójności zasobów danych i niepodatności na próby modyfikacji (Narayanan i in., 2016).

Tak stworzona platforma przyjmuje nowe informacje z sieci i jeśli zostają one uznane za prawidłowe (w ramach transparentnych reguł), do blockchain dodawany jest blok je zawierający, ten natomiast zostaje dystrybuowany poprzez sieć do wszystkich użytkowników przetwarzających blockchain (za pomocą technologii P2P) (Iansiti i Lakhani, 2017). W praktyce możliwości blockchainu pomimo powiązania z systemami transakcyjnymi, nie są ograniczone wyłącznie do takiego przypadku wykorzystania (The Economist, 2015).

2.2 Istota funkcjonowania

Technologia blockchain, rozwijającą się na bazie Bitcoin (rysunek 2.1), wyekstrahowała elementy niezbędne do funkcjonowania takowej platformy zdecentralizowanego zaufania użytkowników.



Rysunek 2.1. Wizualizacja działania sieci transakcyjnej Bitcoin.

Źródło: (Bitcoin.org, 2018)

Poprzez ostatnie lata doświadczeń produkcyjnych, niezależne przedsięwzięcia wypróbowały różnorodne podejścia do rozwiązyania problemu centralnego autorytetu, przy czym wiele z nich odniosło sukces w skali globalnej, wskazując dalszą drogę rozwoju blockchainu. Niemal każda z implementacji zawiera (orzaz poszerza) pewne typowe elementy jego architektury.

2.2.1 Rekord transakcji jako nośnik informacji

Podstawową istotą blockchain jest przechowywanie informacji w ustrukturyzowany sposób, podobnie jak w standardowych bazach danych (Iansiti i Lakhani, 2017), z tą różnicą, że jest ona rozproszona poprzez użytkowników z jednakowymi uprawnieniami dostępu i/lub dodania nowych zasobów. W przypadku Bitcoin wartością dodaną każdego bloku stanowią transakcje w ww. walucie (rysunek 2.1), dzięki którym odtwarzana jest historia i saldo końcowe konkretnego użytkownika tego rozwiązania (Bitcoin.org, 2018), będąc tym sposobem jedynym źródłem prawdy o istnieniu cyfrowego zasobu (posiadanej waluty) i możliwości jego przekazania (jako środka płatniczego) dla strony trzeciej.

Każdy z użytkowników operujących posiada więc własną kopię blockchain, którą na bieżąco aktualizuje z innymi użytkownikami, będącymi w identycznym położeniu. Dołączenie nowego bloku (niosącego informację) jest przekazywane do pozostałych użytkowników, aby stan blockchainu pozostawał zsynchronizowany i jak najbardziej aktualny (rysunek 2.1). Przy czym, każdorazowe dodanie informacji (w postaci bloku) wiąże się ze sprawdzeniem przy każdym, pojedynczym użytkowniku sieci blockchain, czy aby dodana informacja jest prawidłowa, możliwa i bezpieczna, zapobiegając próbom oszustwa (Mearian, 2018). Początkowo ustalone reguły w rozwiązyaniu blockchain są permanentne dla funkcjonowania sieci i jej wszystkich użytkowników, oraz nie istnieje możliwość (prócz błędu programisty), aby uchylić się od ich przestrzeżenia.

Rozwijającą się koncepcją technologii blockchain jest wykorzystanie zasobów jego użytkowników jako platformy pod budowę zdecentralizowanych aplikacji, tak jak ma to miejsce np. w kryptowalucie Ethereum. W ramach zasobu informacyjnego

przekazywanego w łańcuchu bloków możliwe jest dodanie wcześniej przygotowanego kontraktu (ang. smart contracts), którego wykonanie gwarantuje każdy pojedynczy użytkownik blockchain (Ethereum Foundation, 2018). W takiej postaci Ethereum istnieje jako niezależny pośrednik w zawartej umowie pomiędzy dwiema stronami, dając niemal pełną pewność, że gdy określony warunek zostanie spełniony, to kontrakt będzie wykonany w ramach sieci, a jedna ze stron zyska przedmiot kontraktu. Idea platformy aplikacji jest również realizowana w kryptowalucie Lisk, która prócz egzekwowania logiki biznesowej, umożliwia osadzenie całego kodu programowego w ramach sieci, napisanego w technologii JavaScript. Aplikacja wdrożona w Lisk zostaje uruchomiona w ramach użytkowników blockchainu, którzy gwarantują jej nieprzerwane działanie (Lisk Foundation, 2018).

2.2.2 Weryfikacja tożsamości oraz ochrona danych

Blockchain z założenia udostępnia wszelkie dane swoim użytkownikom, brak centralnego autorytetu uniemożliwia jednak przeprowadzenie autoryzacji w ramach sieci zgodnie ze standardowymi rozwiązaniami. Dlatego dla weryfikacji tożsamości wykorzystywana jest kriptografia asymetryczna (Narayanan i in., 2016). Zastosowane rozwiązanie pozwala na zdecentralizowanie zaufania do prawidłowego rozpoznawania użytkownika i zezwolenie na dokonywanie określonych akcji, takich jak przeprowadzanie transakcji kryptowalutowych, w przykładzie Bitcoin.

Po stronie indywidualnego użytkownika w ramach portfela waluty istnieje klucz prywatny, który służy do cyfrowego podpisania transakcji, stanowiąc kryptograficzny dowód, że operacja transakcyjna pochodzi od prawowitego właściciela portfela (kryptowaluty). Zastosowany podpis cyfrowy wyklucza również zmianę raz zaszyfrowanej transakcji przez strony trzecie (Bitcoin.org, 2018). Dodawana (przez użytkownika) informacja do bloku jest więc szyfrowana oraz podpisywana za pomocą pary kluczy prywatny/publiczny, które znajdują się na jego/jej urządzeniu dostępowym, w ramach aplikacji przygotowującej operacje (np. portfel). Klucz prywatny jest jedynym gwarantem dostępu właściciela do operowania zapisem informacji (np. kryptowaluty) i stanowi przedmiot najściślejszej ochrony. Natomiast klucz publiczny, który jest

powiązany z kluczem prywatnym jest dystrybuowany i wykorzystywany w ramach blockchain do identyfikowania danej transakcji z konkretnym użytkownikiem (w praktyce, jako zapis hash) oraz jego pochodna stanowi identyfikacyjny adres użytkownika (portfela) (D'Aliessi, 2016).

Pomimo, że informacje transakcyjne w kryptowalucie Bitcoin są jawnie dostępne dla wszystkich posiadaczy blockchainu, w przypadku ogólnym możliwym jest wykorzystanie szyfrowania dla ukrycia informacji wrażliwych przed osobami trzecimi oraz ich deszyfrowanie dla osoby docelowej. Zabezpieczenie kryptograficzne jest uważane za dostatecznie bezpieczne, dopóki rozwój systemów informatycznych nie doprowadzi do momentu, w którym wykorzystanie mocy obliczeniowej pozwoli na jego złamanie, w ww. wypadku odtworzenie klucza prywatnego na bazie publicznego (Stallings, 1999). W długim okresie wszelkie informacje zaszyfrowane w ramach blockchain mogą stać się jawne dla osób postronnych, gdy zastosowana kryptografia okaże się zbyt słaba w porównaniu do zwiększającej się mocy komputerów, a autoryzacja użytkowników przestanie być relevantna.

2.2.3 Przetwarzanie oraz dowód pracy

Funkcjonowanie sieci blockchain wymaga osobnych węzłów przechowujących własne kopie rozproszonego rejestru, a także przetwarzających informacje nadsyłane od użytkowników w sposób zdecentralizowany. Dla przypadków blockchainów zamkniętych stosunkowo łatwo jest ustanowić węzły w ramach ograniczonej grupy docelowych odbiorców, natomiast w otwartych zastosowaniach (np. kryptowalut) każdy może stanowić rolę węzła, aby zapewnić rozwój takiej sieci, za wykonywaną pracę przetwarzania otrzymuje on wynagrodzenie (w postaci waluty blockchainowej).

Bazując na przykładzie Bitcoin, wykonane przez użytkowników transakcje są przekazywane do wszystkich węzłów w ramach sieci (wykorzystując protokoły Peer-to-Peer), jednak zanim zostaną zatwierdzone wymagają przetworzenia. Węzeł (w Bitcoin tzw. kopacz (ang. miner)) przechowuje niepotwierdzone transakcje (sprawdzając uprzednio, czy w jego rozproszonym rejestrze przeprowadzone operacje są dopuszczalne) i upakowuje je w bloki, które umieszcza w blockchainie. Aby nowy blok

mógł zostać dodany potrzebna jest moc obliczeniowa na rozwiązywanie zadania kryptograficznego, wymuszając konkurencję pomiędzy węzłami, ograniczając możliwość zdominowania decyzyjności sieci od jednego z nich. Ten z węzłów, który jako pierwszy doda swój blok do blockchainu rozpropagowuje swoje rozwiązanie pomiędzy pozostałymi, a gdy te zaakceptują (i przyjmą) je jako prawidłowe, to transakcja w Bitcoin zostaje zatwierdzona. Tak opracowany mechanizm konsensusu jest tzw. dowodem pracy blockchain, teoretycznie udowodnionym modelem pozwalającym na zdecentralizowanie informacji (Bitcoin.org, 2018; CBINSIGHTS, 2017).

Pochodne rozwiązania blockchainowe opracowywały własne dowody pracy, stosując inne podejścia do samokontroli sieci. Wychodząc naprzeciw zwiększającym się problemom z wymaganiami utrzymania węzłów bazujących na mocy obliczeniowej (znaczne zużycie energii elektrycznej) podejmowane są próby nad zrezygnowaniem z tego procesu na rzecz tzw. dowodu stawki (ang. Proof-of-Stake), który zamiast wymagać łamania zadania kryptograficznego, przeniosiłby odpowiedzialność zarządzania blockchainem na najbardziej rozbudowane węzły (posiadające największą ilość waluty) nie generując dodatkowych kosztów czy szkody dla podmiotów trzecich (Ishizaki, 2018).

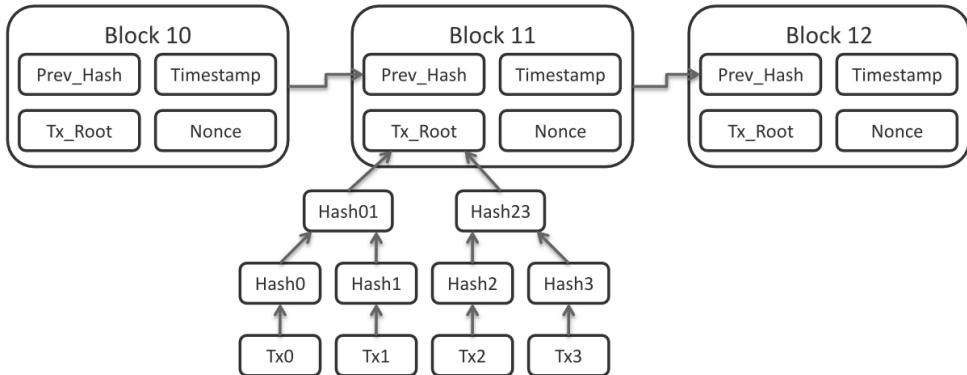
2.3 Technologia dla blockchain

Blockchain abstrahuje od technologii bazodanowej przyjmując strukturę rekordowego przechowywania informacji, wzmacniając ją poprzez powiązania pomiędzy rekordami oraz zabezpieczenie w postaci dowodu pracy. Pomimo wielu różnic pomiędzy implementacjami, pewne elementy pozostają niezmiennie wprowadzane, jako ustandaryzowane i udowodnione w praktyce. Dynamiczny rozwój technologii wciąż redefiniuje jej ostateczną postać, usprawniając funkcjonowanie blockchain.

2.3.1 Łańcuch bloków, bloki oraz rekordy

Łańcuch bloków w blockchain rozpoczyna się od bloku zerowego, tzw. Genesis Block (Sherry, 2018). Twórcy rozwiązania dodają go jako pierwszy do łańcucha ze

względzie na brak bloku poprzedzającego, który zostałby wskazywany w jego strukturze danych (jako referencja zapewniająca spójność). Tak spreparowany blockchain może zostać wykorzystywany (przez węzły i użytkowników) zgodnie z ustalonym dowodem pracy.



Rysunek 2.2. Struktura łańcucha bloków (na podst. Bitcoin).
 Źródło: (Murch, Satoshi i Community ☺, 2013)

Struktura łańcucha, prócz informacji biznesowej, zawiera pewne podstawowe elementy (rysunek 2.2), które gwarantują zachowanie spójności blockchain:

Timestamp (czas utworzenia) – moment powstania bloku, czyli czas w formie Uniksowej (POSIX), w którym blok został dodany do blockchain.

Prev_Hash (referencja do poprzedniego bloku) – hash bloku, na którego dodany został bieżący, w łańcuchu każdy blok jest połączony tym hashem z każdym poprzednim, tworząc w ten sposób ciąg powiązanych elementów zawierających informację.

Tx_Root (tzw. merkle root) – w Bitcoin, zredukowana reprezentacja transakcji załatwionych w ramach bloku, stanowi hash elementów informacji zawartych w bloku.

Target – trudność utworzenia nowego bloku, wyrażająca bieżącą wartość złożoności zadania dodania go do blockchain (a więc przekładająca się na zużycie zasobów mocy obliczeniowej).

Nonce – wybrana arbitralnie liczba, aby zwiększyć entropię dodawanego bloku, bez przebudowy merkle root (dodająca losowość powstającemu blokowi).

Hash – hash własny, identyfikacja utworzonego bloku,

Łańcuch rośnie w miarę wykorzystywania go wewnętrz sieci, łącząc ze sobą rekordy informacji zawarte w blokach. Struktura opisana na podstawie rysunku 2.2 jest **nagłówkiem**, który umożliwia bezpieczne przechowywanie danych wewnętrz bloku.

2.3.2 Techniczna implementacja blockchain

Oryginalnie (na podstawie Bitcoin) implementacje technologii blockchain powstały w językach C, C++ (listing 2.1), prawdopodobnie ze względu na niskopoziomową naturę rozwiązania, dla której język ten posiada zalety, m.in. złożoną kontrolę typowania zmiennych (przykłady: int32_t, arith_uint256, int64_t, etc.) ułatwiające dokładne zaprogramowanie algorytmu zgodnie z oczekiwaniami (istotne w wypadku kryptografii) (Bitcoin.org GitHub i GitHub Community, 2018). Dodatkowo, implementacja w C++ zyskuje na wydajności, kosztem wsparcia wieloplatformowości i wygody rozwoju.

Chociaż język C++ jest nadal powszechny w opracowywaniu blockchainów, nie ma przeciwwskazań, które eliminowałyby inne języki programowania w próbie implementacji algorytmów tej technologii. Istnieją zewnętrzne inicjatywy, mające na celu przenoszenie oryginalnych implementacji blockchain na inne języki programowania, np. Python (Wats0ns, cdecker i Community ☺, 2015), ułatwiając zrozumienie składni i wykorzystanych w algorytmach rozwiązań. Nie są one jednak ściśle synchronizowane z oryginalnym projektem, toteż zdarzają się w nich braki implementacyjne, które z reguły uniemożliwiają ich powszechnie wykorzystanie.

```
class CBlockFileInfo
{
public:
    unsigned int nBlocks;           //!< number of blocks stored in file
    unsigned int nSize;             //!< number of used bytes of block file
    unsigned int nUndoSize;         //!< number of used bytes in the undo file
    unsigned int nHeightFirst;      //!< lowest height of block in file
    unsigned int nHeightLast;       //!< highest height of block in file
    uint64_t nTimeFirst;            //!< earliest time of block in file
    uint64_t nTimeLast;             //!< latest time of block in file

    ADD_SERIALIZE_METHODS;

    template <typename Stream, typename Operation>
    inline void SerializationOp(Stream& s, Operation ser_action) {
        READWRITE(VARINT(nBlocks));
        READWRITE(VARINT(nSize));
        READWRITE(VARINT(nUndoSize));
    }
};
```

```

READWRITE(VARINT(nHeightFirst));
READWRITE(VARINT(nHeightLast));
READWRITE(VARINT(nTimeFirst));
READWRITE(VARINT(nTimeLast));
}

void SetNull() {
    nBlocks = 0;
    nSize = 0;
    nUndoSize = 0;
    nHeightFirst = 0;
    nHeightLast = 0;
    nTimeFirst = 0;
    nTimeLast = 0;
}
CBlockFileInfo() {
    SetNull();
}
std::string ToString() const;
// ...
}

```

Listing 2.1. Fragment implementacji *chain.h*. Źródło: (Bitcoin.org GitHub i GitHub Community, 2018)

Znaczne zainteresowanie zagadnieniami kryptowalut skupiło wokół siebie wielu programistów Open Source, którzy znacznie przyczynili się do rozwoju implementacji blockchain. Oryginalne repozytorium bitcoin¹ powiększyło się o ponad 16 0000 addy-cji (commitów) do kodu źródłowego wykonanych na przestrzeni kilku lat, decydując o dalszym rozwoju pionierskiej kryptowaluty² (Bitcoin.org GitHub i GitHub Community, 2018; trottier i GitHub Community, 2018).

2.3.3 Rola protokołów Peer-to-Peer

Nieodzownym elementem sieci rozproszonej typu peer-to-peer jest wzajemne odnajdywanie się węzłów oraz użytkowników, w celu efektywnego funkcjonowania. W odróżnieniu od architektury klient-serwer, w przypadku decentralizacji informacji unika się tworzenia centralnych punktów odpowiedzialnych, dlatego wykorzystano odpowiedzialność strony dostarczającej oprogramowanie. Pierwsze uruchomienie węzła wiąże się z pobraniem listy znanych węzłów sąsiednich, np. poprzez wykonanie zapytania DNS na jeden z zapisanych w kodzie adresów (niezależnych i wielu różnych) (Schwartz, ODell i Community ☺, 2013). Pierwsze węzły w sieci wystarczą,

¹<https://github.com/trottier/original-bitcoin>

²<https://github.com/bitcoin/bitcoin>

aby stopniowo pobierać kolejne adresy węzłów w ramach komunikacji z nimi, podobnie jak i te, które łączą się do naszego punktu, rozpraszają informację o nowym węźle po całej sieci.

Oryginalnie wykorzystywane były protokoły IRC, z których jednak zrezygnowano na rzecz metod DNS, ze względu na lepszą skalowalność (Schwartz i in., 2013). Ostatecznie ze wzrostem czasu spędzonego online w sieci, nasz adres (z założenia zmieniany rzadko lub definitywnie statyczny) jest znany powszechnie dla wszystkich węzłów, tak samo jak i my posiadamy informacje o ich adresach, bez centralnego punktu krytycznego.

2.3.4 Otwartość pod względem technicznym

Ze względu na udostępniony publicznie kod źródłowy, Bitcoin doczekał się wielu następców waluty, m.in. LiteCoin, DodgeCoin itp., wszystkich bazujących na podstawowym blockchainie od pionierskiej kryptowaluty (tzw. fork). Wskazuje to na łatwość adaptacji raz utworzonego rozwiązania i możliwość przejęcia go niemal w każdym momencie, tworząc nową kryptowalutę pod własnym szyldem. Jednakże to sieć decyduje o wartości waluty, także trudność rozpropagowania swojego rozwiązania stanowi ostateczny wskaźnik sukcesu kryptowaluty.

Z biznesowego punktu widzenia każdy może przejąć rozsystrzybuowane rozwiązanie blockchain stanowiąc ryzyko inwestycyjne w przedsięwzięcie. Dla programistów takiej firmy natomiast może to oznaczać uzyskanie łatwego wsparcia ze strony Open Source (poprzez portale typu GitHub). Ekosystem kryptowalutowy jest bardzo dynamiczny pod względem zmian rynkowych, które mogą być druzgocące dla funkcjonowania firmy.

Bezpieczniejszym rozwiązaniem pod względem dystrybucji może się okazać zamknięcie na określoną grupę docelową użytkowników/instytucji, których zaufanie wynika z czynników spoza sieci. Natomiast nawet tak ograniczona grupa odbiorców może przekazać kody źródłowe z zamkniętej grupy do stron trzecich, poza nią.

Rozwój blockchainu publicznego sprzyja adaptowaniu najnowszych, innowacyjnych rozwiązań i może stanowić solidny fundament dla długiego zaadaptowania

rozwiązań.

2.4 Bezpieczeństwo zastosowania blockchain

2.4.1 Anonimowość użytkowników i danych

W blockchainach kryptowalutowych informacją identyfikującą jest adres portfela, np. dla Bitcoin *3FkenCiXpSLqD8L79intRNXUgjRoH9sjXa* (adres portfela społeczności Bitcoin.org), który będąc hashem klucza publicznego sparowanego z konkretnym kluczem prywatnym, nie stanowi informacji o osobie fizycznej będącej posiadaczem adresu. Niemniej jednak jest to jedyna przeszkoda dzieląca anonimowość od posiadania informacji o wszystkich wykonanych transakcjach w walucie przez konkretną osobę.

Jeżeli adres reprezentujący danego użytkownika zostanie powiązany z konkretną jednostką, np. Bitcoin.org, informacja ta umożliwia śledzenie jej powszechnie dostępnych danych z publicznego blockchain, stanowiąc zagrożenie dla instytucji. W przypadku Bitcoin ujawniane są jedynie przeprowadzone transakcje powiązane z tym adresem, jednak w kontekście szerszego wykorzystania blockchain ujawnienie danych wrażliwych jest ryzykiem funkcjonowania zarówno implementacji, jak i biznesu na nim funkcjonującego.

2.4.2 Kryptografia gwarantem własności

Wzrost mocy obliczeniowej z czasem powoduje łatwe łamanie zabezpieczeń kryptograficznych, wystawiając dane na ekspozycję dla strony trzeciej i/lub paraliżując działanie rozwiązania. Jeżeli blockchain nie będzie w stanie nadążyć rozwojem algorytmu szyfrującego (co często nie jest łatwe do zaadaptowania), to ostatecznie użytkownicy będą mogli forsować zmiany poza regułami bezpieczeństwa systemu.

Coraz częściej spotykanym zagrożeniem jest również to ze strony *komputerów kwantowych*, według Emerging Technology (2017): „*Ogromna moc obliczeniowa komputerów kwantowych będzie w stanie złamać zabezpieczenia Bitcoin w ciągu 10 lat (...)*”. Choć

termin wydaje się być fantastyczną naukową, coraz częstsze jego wspominanie w kontekście kryptowalut wydaje się urealniać płynące zagrożenie z rozwoju technologii.

2.4.3 Ustanawianie brakujących regulacji prawnych

Technologia zdecentralizowanego zaufania powstała niezależnie od instytucji państwowych czy korporacji, nie została więc jeszcze wprowadzona legislacja takiej formy wymiany informacji, co jest szczególnie istotne w przypadku obrotów środkami płatniczymi, czyli właśnie kryptowalutami. Pomimo natury niezależności, podejmowane są próby regulacji rynku, egzekwowania podatków czy też wręcz prawne zakazanie obracaniem walutami typu Bitcoin (Norry, 2017).

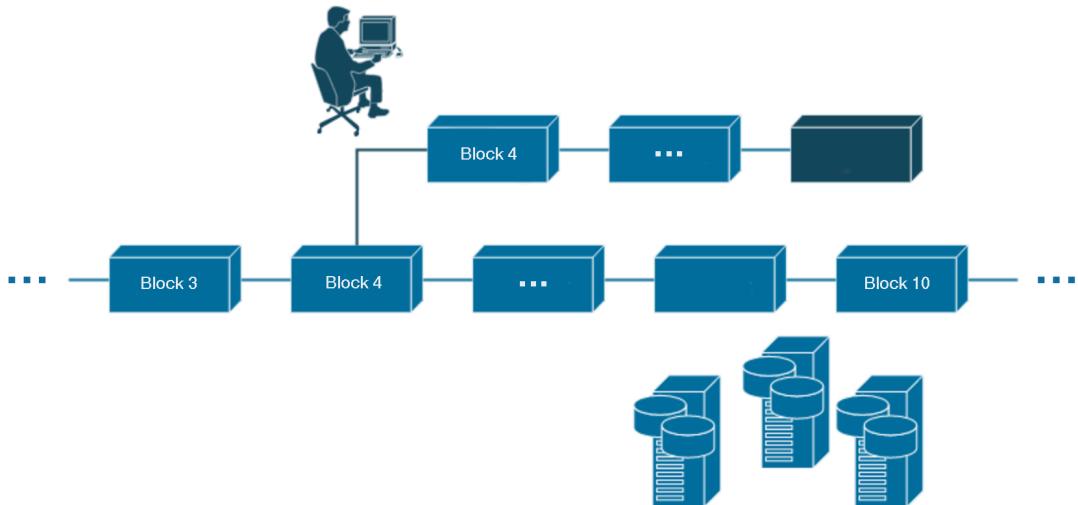
Poza przypadkiem środków pieniężnych, informacja jest również chroniona, tak jak np. dane osobowe, prawo do ingerencji we wprowadzone zasoby (włącznie z możliwością usuwania ich), czy współdzielenie cennych danych biznesowych będących przedmiotem ochrony. Wiele typowych problemów rynku obrotu środkami pieniężnymi może być odniesione do kryptowalut, dlatego zastosowanie rozwiązania wymaga solidnego podłoża prawnego, aby działalność biznesowa mogła być prowadzona w sposób prawowy.

2.4.4 Proces teoretyczny falsyfikacji bloków

Najbardziej oczywistym przypadkiem fałszerstwa jest próba modyfikacji raz dodanego bloku (co jest sprzeczne z założeniem braku zmian wstecz). Blockchain chroni się przed tym mechanizmami dowodu pracy oraz spójności bloków w łańcuchu.

Zakładając, że użytkownik zamierza zmodyfikować **blok nr 4** (rysunek 2.3), na podstawie (Murch, Patoshi i Community ©, 2013):

1. Sieć pracuje na odległym bloku, np. **numer 10**.
2. Użytkownik przyjmuje blok **nr 4** za bieżący i dodaje sfałszowane informacje, aby umieścić go w blockchainie wykorzystując moc obliczeniową na łamanie zagadki kryptograficznej.
3. Aby zmiany użytkownika zostały przyjęte musi on dogonić sieć **od bloku numer 4 do 10** włącznie, zużywając każdorazowo czas i moc obliczeniową.



Rysunek 2.3. Próba falsyfikacji bloku oraz jej następstwa.

Źródło: opracowanie własne na podstawie (Zeev, 2015)

4. Jeżeli przez czas łamania **7 bloków** nikt w sieci nie odnajdzie bloku numer **11**, to sieć przyjmuje sfalsyfikowany łańcuch użytkownika.

Nakład wymagany na sfałszowanie bloku oraz ryzyko podjęcia się takiego działania jest bardzo duże i nieopłacalne w kontekście wydobywania kryptowaluty. W praktyce, aby wyprzedzić tempo odkrywania bloku przez sieć, należałoby wielokrotnie przewyższyć moc obliczeniową wszystkich jej węzłów, co z założenia jest niemożliwe (decentralizacja, brak punktu skupiającego) oraz bardzo kosztowne.

2.4.5 Punkty krytyczne wraz z analizą opinii eksperckiej

Pomimo nadmiarowości danych, rozproszonych po węzłach w sieci, zaprojektowane rozwiązanie decentralizacji posiada punkty krytyczne ze strony użytkownika. Proces identyfikacji w sieci odbywający się na zasadzie par kluczy prywatny/publiczny jest formą naiwnego rozróżniania użytkownika, z pominięciem wielokrotnego zduplikowania portfeli będących w posiadaniu jednej osoby. Nie jest więc możliwe stwierdzenie, czy pomiędzy danymi adresami (kluczami publicznymi) faktycznie istnieje zależność do jednego, konkretnego podmiotu. Z drugiej strony, dywersyfikacja portfeli ma służyć zwiększeniu poziomu anonimowości użytkownika, tak aby ujawnienie się (publiczne powiązanie klucza i jego właściciela) nie było aż tak dotkliwe (ponieważ transakcje rozproszone są pomiędzy wieloma portfelami). Natomiast

miast model pojedynczego portfela skutkuje najbardziej dotkliwym punktem krytycznym, utrata klucza prywatnego permanentnie zdyskwalifikuje prawowitego właściciela informacji do zarządzania nią, odbijając się również na całej sieci, gdyż tak zagubiony rekord informacji pozostanie niewykorzystany (wg założeń) do końca istnienia rozwiązania blockchain.

Niepokojące okazują się opinie eksperckie sektora finansowego, bankowego oraz rządowego. Wielokrotnie podejmowane są zagadnienia dynamicznej natury kryptowaluty, przyrównując Bitcoin do jednej z największych historycznie baniek spekulacyjnych (*tulipomania, wiek 17-ty, Holandia*) (Lam, 2017). Otwarte wygłaszczenie takich opinii przez instytucje darzone renomą skłaniają opinię publiczną do obawiania się rozwiązań typu blockchain, oddalając jej akceptację w szerokim rozumieniu, stanowiąc zagrożenie odrzucenia modelu biznesowego przez konsumenta docelowego. Według władz chińskich to bank centralny powinien sprawować autorytet nad kryptowalutą, usiłując całkowicie zdominować rozwiązanie z założenia pozbawione kontroli. Rosyjski bank centralny przestrzega przed charakterem piramidy finansowej zbudowanej na rynku kryptowaluty, idąc w kierunku ścisłej kontroli przepływu funduszy i eliminowania ich z funkcjonowania. Bank francuski ostrzega przed koncepcją „prywatnej waluty”, za którą brak oficjalnego autorytetu będącego gwarantem bezpieczeństwa, insygnując szybki i gwałtowny jej upadek, za bardzo prawdopodobny. W Indiach ze względu na prowadzoną politykę przeciwko praniu brudnych pieniędzy i finansowaniu działalności terrorystycznej, kryptowaluty zostały zakazane prawnie. (Lam, 2017).

Prócz zdecydowanie negatywnego opiniowania tematyki blockchain, pojawiają się również pozytywne wypowiedzi o jego rewolucyjnych rozwiązaniach: bank centralny Wielkiej Brytanii podkreśla innowacyjny charakter decentralizowania procesu księgowania transakcji, który mógłby wspomagać instytucje rządowe w wykrywaniu cyberprzestępstw, natomiast bank Brazylii nie dostrzega zagrożenia dla państwowego systemu gospodarczego i zachęca wręcz do inicjatywy oddolnej mogącej „*wspomóc innowacje, (...) czyniąc system finansowy bezpieczniejszym oraz bardziej efektywnym*” w skali całego kraju. Holandia oraz część krajów skandynawskich doszukują się zalet kryptowaluty dla funkcjonowania państwa, włączając opcję opracowania jej własnej wer-

sji dla celów wewnętrznych. Bank Kanady wskazuje na istotny element technologii blockchain, czyli śledzenie zasobów, mogących rozwinąć obecne formy kontroli nad fizycznymi rezerwami (Lam, 2017).

Jeżeli technologia blockchain okaże się społecznie nieakceptowalna, zostanie odrzucona przez opinię publiczną, zanim jeszcze osiągnie pełen możliwy potencjał. Problem ten natury socjologicznej może okazać się o wiele bardziej krytyczny dla funkcjonowania biznesu, niż obszerne wyzwania techniczne adaptacji rozwiązania.

2.5 Mankamenty realizacji rozwiązania

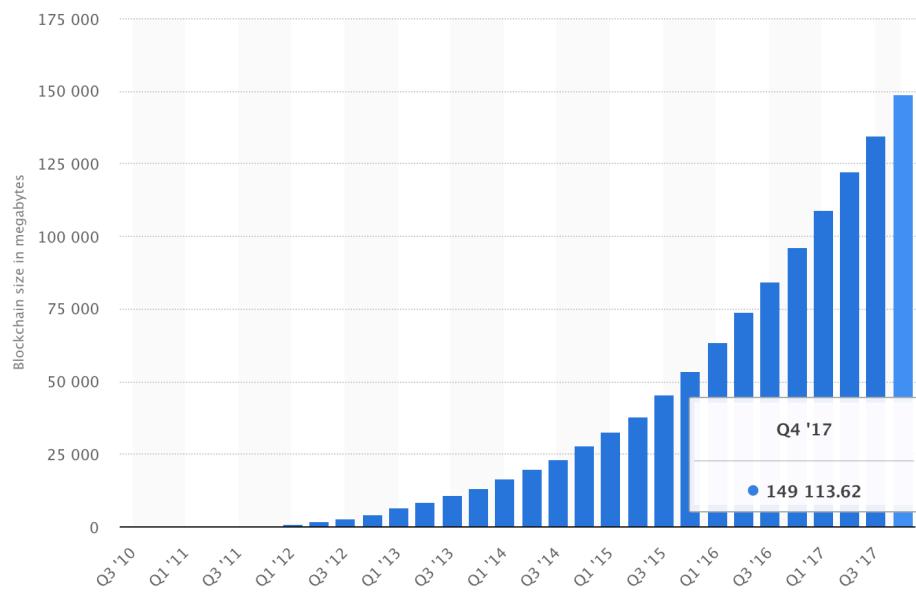
2.5.1 Publiczna dostępność zasobów

Decentralizacja zasobów blockchain wymusza ujawnienie informacji cennych biznesowo, wszystkim podmiotom wykorzystującym dane rozwiązanie. Dla przypadku kryptowalut, które przechowują zapisy transakcji, jest to ograniczone ryzyko ujawnienia bilansu konkretnej osoby prywatnej (chroni ją dodatkowo anonimowość danych transakcyjnych), jednakże w wielu przypadkach zastosowań komercyjnych nie jest możliwe ujawnienie wszelkich informacji bez szkody dla firmy, wynikającej z punktu widzenia biznesowego, prawnego lub etycznego. Dodatkową ochroną może okazać się szyfrowanie zawartości, aczkolwiek może to spowodować utrudnienie w działaniu logiki biznesowej oraz nie jest to zabezpieczenie permanentne.

2.5.2 Ograniczenia skalowalności blockchain

Łańcuch bloków zezwala na operacje dodawania informacji do jego struktury, oznacza to więc, w długim okresie, rozrost do niebotycznych rozmiarów, utrudniając decentralizację oraz funkcjonowanie blockchainu.

Według Statista (2018) rozmiar blockchain dla Bitcoin pod koniec roku 2017 wynosił już niemal **150 GB** (rysunek 2.4). Prawdopodobnie nieograniczony rozrost ostatecznie zmusi użytkowników blockchain do wypracowania wspólnego rozwiązania, ze szkodą dla spójności danych.



Rysunek 2.4. Rozmiar fizyczny blockchain Bitcoin (w megabajtach).
Źródło: ([Statista, 2018](#))

2.5.3 Powolność oraz wadliwość algorytmów

Opracowanie rozwiązania typu blockchain narzuca ścisłe reguły operowania na nim, dla wszystkich, włączając użytkowników, węzły oraz samych twórców. Raz udostępnione i zaadaptowane traci możliwość samoaktualizowania się, gdyż to doprowadzałoby do rozbieżności w wersjach pomiędzy użytkownikami. Twórcy w odróżnieniu od standardowego podejścia, nie mają bezpośredniej kontroli nad wprowadzaniem zmian w kodzie źródłowym, a ostatecznym decydującym czynnikiem jest rozdystybuowanie wersji w sieci. Jeżeli węzły w znacznej większości, nie przyjmą nowszej wersji blockchain, twórcy nie są w stanie wymusić na użytkownikach swojego rozwiązania.

Z braku możliwości prostej i szybkiej aktualizacji wynika wadliwość założonych rozwiązań algorytmicznych. Przykładowo, w Bitcoin algorytm zatwierdzania transakcji przewiduje *10 minut oczekiwania* ([Bitcoin.org, 2018](#)). W rzeczywistości jednak przez rozrost sieci ten okres wynosi od *30 minut* do ponad *16 godzin* w znacznym stopniu ograniczając płynność waluty ([Buchko, 2017](#)). Jest to przypadek problemu post factum, który bardzo trudno jest przewidzieć na etapie tworzenia oprogramowania, a dodatkowo jego naprawa jest utrudniona. Jednym z proponowanych rozwiązań było

utworzenie z kryptowaluty Bitcoin nowej waluty na bazie tego samego blockchain, lecz ze zaktualizowanymi algorytmami – tak powstał tzw. *BitcoinCash* (Garner, 2017).

2.5.4 Specyficzność zastosowań

Technologia zdecentralizowanego zaufania użytkowników okazuje się rozwiązywać wiele problemów opartych na jednostce, lecz sama obostrzona jest wieloma ograniczeniami, które eliminują ją z wielu specyficznych rozwiązań. Cytując CBINSIGHTS (2017): „*Blockchains are really good at a couple of things and absolutely awful at others.*” (ang. Blockchain jest naprawdę dobry w kilku rzeczach i absolutnie okropny w innych.) wskazuje na problem zadecydowania, czy rozwiązanie typu blockchain istotnie jest pożądane (i możliwe) dla danego przypadku, pomijając fakt jego uzasadnienia dla biznesu.

2.5.5 Niestabilność ekosystemu narzędzi deweloperskich

Proces tworzenia oprogramowania wymaga sprawdzonych, przetestowanych i gotowych do zastosowań produkcyjnych rozwiązań. Tematyka blockchain jest stosunkowo nowatorska i niestandardyzowana, tymczasem produkty informatyczne potrzebują czasu na rozwój i eliminowanie błędów oraz poszerzanie zakresu funkcjonowania.

Praca z oprogramowaniem będącym w fazach niestabilnych (np. *alfa, beta*) świadczy o niegotowości rozwiązania do zastosowania w biznesie, ze względu na zagrożenia, które mogą wystąpić w trakcie jego funkcjonowania. Trudna natura aktualizowania blockchain dodatkowo komplikuje naprawę błędów, a te nieprzewidziane mogą okazać się drastyczne w skutkach.

Niezależne instytucje i stojący za nimi innowatorzy usiłują rozwinąć istniejące oprogramowanie do poziomu pozwalającego na jego bezpieczne wykorzystanie. W zastosowaniu praktycznym są implementacje typu dowodu poprawności (ang. Proof-of-Concept), które stanowią obiecujący efekt pracy, mogący w niedalekiej przyszłości urzeczywistnić najbardziej niespodziewane zastosowania blockchain.

Przeświadczenie o zagrożeniu wynikającym z technologii stanowi znaczny czynnik ryzyka inwestycji biznesu w blockchain, jako priorytet musi zostać opracowane oprogramowanie stabilne, które można zastosować w scenariuszach finansowych, bez obaw o straty pieniężne.

2.6 Dostępność technologii dla zastosowań adaptujących

Ewolucja technologii blockchain doprowadziła do powstania wielu różnorodnych rozwiązań, które można sztucznie wydzielić według kryteriów otwartości i dostępności. Każda z nich reprezentuje odrębne podejście do problemu zarządzania zdecentralizowaną informacją i w różnym stopniu nadaje się do zaadaptowania w rzeczywistości biznesowej.

2.6.1 Globalna platforma (model Paas)

Globalna platforma jako usługa (PaaS) umożliwia wykorzystywanie blockchain bez wymogu własnoręcznego ustawiania go na serwerach, które są w naszym posiadaniu. Podmioty, które widzą potencjał w zastosowaniu tej technologii w swoim biznesie mogą z niej skorzystać na zasadzie uzyskanych tokenów ([Ethereum](#)) lub umieszczenia kodu źródłowego ([Lisk](#)) w ramach udostępnionego publicznego blockchainu (Zimnoch, 2017).

Prekursorem blockchain (np. w przypadku Bitcoin) jest otwarte rozwiązanie platformy do wymiany informacji o transakcjach kryptowaluty. W praktyce platforma może być zawężona tylko do docelowych użytkowników, jednak nadal pozostaje tworem globalnym, który przetwarza dane w sposób rozproszony (możliwie również poza wybraną grupą), tak jak np. [Ripple](#) (Zimnoch, 2017).

2.6.2 Udostępnione oprogramowanie (model SaaS)

Udostępnienie oprogramowania jako usługę (SaaS) umożliwia deweloperom przejęcie danego rozwiązania i jego adaptację do potrzeb przedsiębiorstwa. Wiąże się to

z wymogiem posiadania zasobów sprzętowych do wykonywania blockchainu, jednak gwarantuje, że kontrola nad zastosowanym rozwiązaniem jest po stronie instytucji, zamiast strony trzeciej (**Hyperledger Project, BigchainDB**). Tak opracowany blockchain może funkcjonować w zamkniętej (wybranej przez instytucję) grupie docelowych użytkowników lub być udostępniony publicznie, jako własna niezależna platforma (Zimnoch, 2017).

Dostępność kodu źródłowego i kontrola nad oprogramowaniem stanowi podwaliny biznesowego wykorzystania blockchain w wyspecjalizowanych przypadkach (np. wewnętrznego obrotu rozliczeniami), gwarantując bezpieczeństwo potrzebne dla danej firmy. Jednocześnie jednak ogranicza to potencjał rozwiązania, negując globalnych charakter technologii zdecentralizowanego zaufania.

2.7 Podsumowanie

W rozdziale zaprezentowano charakterystykę technologii blockchain, wychodząc od jej funkcjonowania, poprzez szczegóły techniczne, po wyzwania stawiane przed ją adaptującymi. Poruszono kwestie bezpieczeństwa, wyjaśniając m.in. zasadę ochrony danych przed falsyfikacją, krytykę publiczną. Omówiono problemy, które wymagają rozwiązań w rzeczywistości biznesowej, takie jak powszechna dostępność danych, skalowalność i wady algorytmów, a także dokonano przekroju kierunków rozwoju technologii od globalnych do zamkniętych rozwiązań.

Powszechnie z technologią blockchain wiązane są duże nadzieje na rewolucyjną (oraz innowacyjną) zmianę podejścia do obdarzania zaufaniem instytucji, ponieważ blockchain decentralizuje informację eliminując potrzebę istnienia pośrednika, który musi nią zarządzać. Może więc w pełni wykluczać obecnie powszechnie instytucje: pośredników kart płatniczych, sektor bankowy jako kontrolujący przepływ pieniędzy, organy wymiany zasobów pomiędzy stronami na światowych giełdach, przechowywanie rekordów informacji o zasobach oraz tożsamości, potrzebę powoływania instytucji do nadzorowania procesów globalnych, np. państwowego głosowania oraz wiele innych (CBINSIGHTS, 2018). Zastosowanie blockchain teoretycznie umożliwia znaczną redukcję kosztów, które stanowiły wynagrodzenie insty-

tucji pośredniczącej, reprezentując duży potencjał zysku dla przedsiębiorstw. Jednocześnie, zagrożenia wynikające z technologii dla sektora finansowego, paradoksalnie przyspieszają jej rozwój.

Wstęp teoretyczny do tematyki funkcjonowania blockchain wprowadza stopniowo w możliwości potencjalnego wykorzystania tego rozwiązania w innych aspektach niż kryptowaluty. Stanowi tym samym podwaliny dla dalszej części pracy, mianowicie badania możliwości praktycznej implementacji technologii zdecentralizowanego zaufania w rzeczywistości biznesowej.

Rozdział 3

Architektura oprogramowania w implementacji blockchain

Fundamentem rozwoju oprogramowania jest jego architektura, która odpowiednio przemyślana na początku pozwala na wygodne i efektywne opracowywanie implementacji rozwiązania IT bez przeszkód w docieraniu do finalnego efektu. W rozdziale skupiono się na odpowiednim umotywowaniu elementów jej schematu, wyprowadzając pierwotną koncepcję i stopniowo uzupełniając o szczegółowe elementy technologiczne z dbałością o harmonijność powiązanych narzędzi.

Strona podmiotu gospodarczego wymaga odpowiedniego oprogramowania, aby realizować bez przeszkód własne procesy i cele biznesowe, a analiza dostępnych rozwiązań pozwala wykluczyć te, które będą powodowały potencjalne problemy oraz wyróżnić inne, które wykażą bezkonkurencyjność adaptującego blockchain biznesu na rynku.

Rozdział skupia się na zaprezentowaniu architektury rozwiązania, od doboru technologii wiodącej, po analizę dostępnych narzędzi strony klienta oraz odpowiednich zależności. Przeprowadzono także analizę strony serwerowej oraz jej poszczególnych segmentów, składającą się w całokształt na kompletną architekturę technologiczną, która z powodzeniem może zostać zastosowana w nowatorskim systemie blockchain. W rozdziale poruszono również zakres narzędzi deweloperskich, które pomimo mniejszego wpływu na biznes, mają istotny wpływ dla rozwoju oprogramowania od strony opracowującego go zespołu.

3.1 Motywacja doboru technologii

Biznes, aby osiągnąć sukces dzięki blockchain potrzebuje ponadprzeciętnej technologii, gdyż samo zastosowanie tego rozwiązania sugeruje innowacyjność, będąc wciąż w fazie dynamicznego rozwoju. Aby nie popełnić błędu, istotne jest zrozumienie kompletnej istoty rozwiązania, od strony biznesowej, jak i zarówno technologicznej.

3.1.1 Aspekty natury biznesowej

Dowolny podmiot gospodarczy, czy to firma, fundacja lub jednostka rządowa, w odpowiednim stopniu musi respektować zasady panujące na rynku i odpowiadać za wszelkie uwarunkowania natury biznesowej, m. in. na podstawie (nadzw. dr hab. Sylwia Sysko-Romańczuk, Roszkowska i Niedźwiecka, 2012):

- świadczenie usług (lub dostawy towarów) wartościowych dla społeczeństwa,
- spełnianie warunku dobrej jakości oraz należytej ceny,
- tworzenie przyrostu wartości dodanej,
- dystrybuowanie dobrobytu pośród pracowników oraz społeczności.

Według nadzw. dr hab. Sylwia Sysko-Romańczuk i in. (2012): „*Firma jest główną instytucją redystrybucji bogactwa ekonomicznego i dzieli je między swoich interesariuszy (...) równocześnie firma musi zapewnić sobie i swoim właścicielom przyszłość, inwestując w nowe technologie i aktywa*”. Wnioskiem jest więc daleko idąca alokacja funduszy w innowację dla wspólnej korzyści i rozwoju, która (w długim terminie) zapewni podmiotowi gospodarczemu stabilny wzrost, a społeczności przyszłość.

Wielu niezależnych ekspertów upatruje w technologii blockchain nadzieję na zmianę porządku rzeczy, oferując naszej cywilizacji rewolucję w podejściu do zarządzania informacją. Takie podejście do tematyki technologii zdecentralizowanego zaufania jest korzystne od strony biznesu adaptującego blockchain i może stanowić solidne fundamenty budowania zysku z usług na nim opartych. Wiele dotychczas sprawdzonych inicjatyw na rynku kryptowalut (waluty takie jak Ethereum, Tether, Neo, Vechain, Cardano, Stellar oraz inne według CoinMarketCap (2018)) spotkało się z optymistycznym przyjęciem i inwestycjami na kwotę bilionów dolarów (CoinMarketCap, 2018).

Wybranie technologii *blockchain* bazując na istniejącym trendzie, może wiązać się z osiągnięciem ponadprzeciętnych zysków w przyszłości.

Aby możliwa była bezpośrednia adaptacja rozwiązania w biznesie, potrzebne są narzędzia sprawdzone, z wieloletnim doświadczeniem produkcyjnym, oraz szeroka gama pracowników lub współpracowników, potrafiących w dobrym stopniu opracować docelowe oprogramowanie: bezpieczne, pozbawione błędów, realizujące scenariusze biznesowe oraz łatwo docierające do grupy docelowych użytkowników (klientów). Blockchain może być postrzegany jako innowacja w fazie przedprodukcyjnej, dobrą narzędzią natomiast powinien zostać podyktowany stabilnymi rozwiązaniami.

Usługa typu blockchain jest przykładem przedsiębiorstwa z dziedziny IT, które zarządza jednym głównym produktem (ang. one product). Posiadając zespoły programistów, które zamiast ograniczać się do realizacji zadań zgodnie z terminami, mogą skupić się na ulepszaniu już istniejącego oprogramowania, pozwalając ludziom na kreatywność w podejściu do problemu oraz możliwość autorskiego wkładu i własnej inwencji. Nowoczesne inicjatywy IT wiążą się z nowatorskim podejściem, z technologiami pozwalającymi wytworzyć zupełnie nową wartość dodaną w porównaniu do przeciętnego poziomu rynku lub oferty konkurencji. Dlatego tak istotnym jest, aby start-up z blockchainem wybijał się na tle innych zaawansowaną technologią na miarę roku bieżącego, z perspektywami na przyszłość.

Nowoczesna technologia oznacza łatwiejsze pozyskanie inwestora, większą atrakcyjność inicjatywy, prostsze zaangażowanie użytkownika oraz wyższą konwersję. Poza wieloma zaletami finansowymi, przekłada się również na wizerunek biznesu jako firmy innowacyjnej oraz wizjonerskiej, a także na poziom zadowolenia pracowników (m.in. programistów), głównie dla młodych oraz ambitnych osób, które kreatywnie doszukują się niebanalnych rozwiązań problemów poza utartymi schematami (które powstrzymują postęp innowacji). Cechy te mogą okazać się nieocenione w rozwoju wciąż nieustandaryzowanej technologii blockchain.

3.1.2 Aspekty dotyczące aplikacji

Oprogramowanie dla biznesu przede wszystkim musi spełniać kryterium bezpieczeństwa. Sytuacje ujawnienia danych poufnych lub też niespójności prowadzą do błędów, a te do nieufności wobec systemu i firmy. Wybrana technologia wiodąca powinna więc być na tyle sprawdzona produkcyjnie, aby ograniczyć zagrożenie płynące z sytuacji losowych do minimum. Zawiera się to w długoletnim procesie rozwoju języka oraz wysoce rozbudowanej społeczności programistów, tworzących do niego narzędzia. Te natomiast - wynikające z szerokiego wykorzystania - przekładają się na dużą ilość dobrze opracowanych dokumentacji, poradników przypadków użycia, wyznaczników poprawności i dbałości o najlepsze praktyki w kodzie.

Innowacja wymaga kreatywności, wybrana technologia nie powinna więc (pomimo akceptowalnego poziomu spójności i bezpieczeństwa) ograniczać twórcy w procesie rozwiązania problemu. Istnieją języki oprogramowania, które zwiększą bezpieczeństwo i stabilność nakładem większej ilości pracy poświęconej na ich wykonanie, to się natomiast przekłada na ilość czasu poświęconą na proces niekreatywny, odtwarzalny, nadmiarowe powiększanie zespołu oraz rozpraszanie od celu głównego, jakim jest opracowanie unikatowej aplikacji, z bezkonkurencyjną wartością dodaną.

Dobór nowatorskiej oraz dynamicznie ewoluującej technologii umożliwia osiągnięcie rezultatu niemal niemożliwego w jakiekolwiek innej ich konfiguracji, dodatkowo wytyczając szlak rozwoju. Dotychczasowe aplikacje jako próbę zaadaptowania nowoczesnych rozwiązań mogą posiłkować się interoperacyjnymi metodami wymiany informacji w architekturze technologicznej, a także wykorzystywać architekturę mikroserwisową, które za pomocą wymiany danych (np. REST API) mogą wspólnie działać, pozwalając na wypracowanie ponadprzeciętnego efektu. Jednakże wzrasta przez to złożoność aplikacji, trudność w zarządzaniu zdyweryfikowaną bazą kodu oraz zwiększym zapotrzebowaniem na specjalistów. Istotnym jest, aby zdecydować się na technologię, która rokuje na powodzenie jej rozwoju w przyszłości, tak aby unikać sytuacji zacofania i kosztownego przerabiania projektu.

Start-upowy charakter blockchainu stawia również wymogi związane z jej rozwojem. Aplikacja musi zostać opracowana szybko i utrzymywać wystarczający poziom jakości, a koncepcja ta tzw. *rapid development*, skupia się na takim doborze języków

programowania, aby ekosystem narzędzi deweloperskich pozwalał na możliwie jak najszybsze opracowanie minimalnego funkcjonującego produktu (ang. MVP), zanim rozwiązania konkurencyjne zdominują rynek pozostawiając podmiot stratnym.

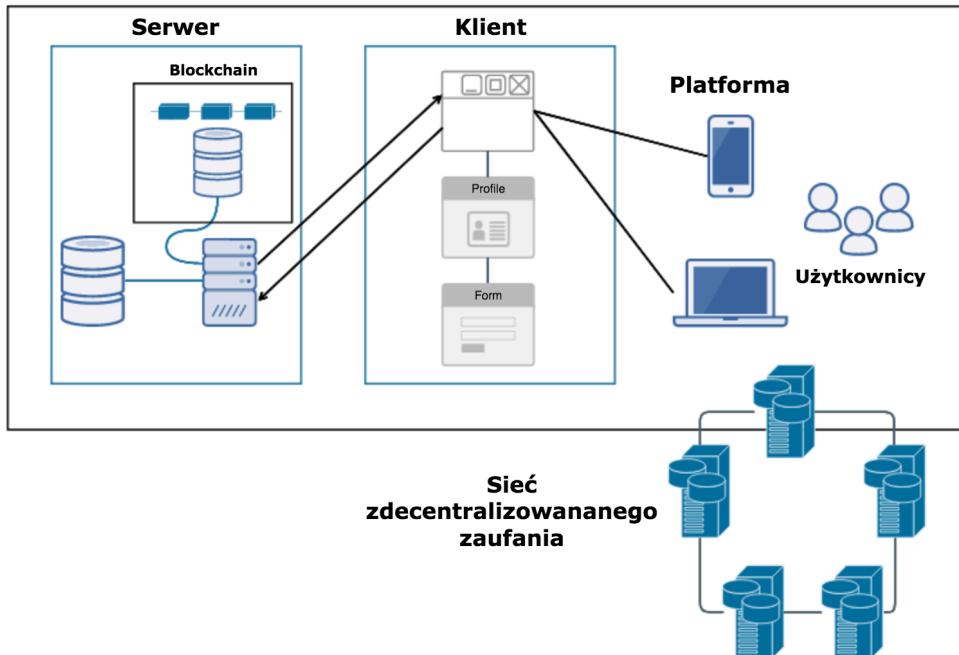
Przedsiębiorstwo podejmując starania o zwiększenie konwersji usługi poszerza dostęp klienta do oprogramowania, w kontekście aplikacyjnym przekłada się to na zagadnienie wieloplatformowości, m.in. wciąż popularnych desktopów, aczkolwiek również Internetu oraz urządzeń mobilnych, a także aplikacji TV oraz konsol do gier (i wielu pozostałych). Aplikacja perspektywiczna powinna pozwalać na jedną bazę kodu na wszystkie główne platformy, z ich minimalnym dopasowaniem w procesie przenoszenia na inną, np. z smartfona Android na iOS.

IT jest dziedziną o dużym dynamizmie, powiązany z ciągłym procesem uczenia. W efekcie w przeciągu kilku lat jedne obiecujące rozwiązania okazują się przestarzałe i wymagają doprowadzenia do aktualnego poziomu rozwoju technologii. Proces udoskonalania jest ciągły i dotyczy projektu w całości, wartościowym jest rozpoczęć od wysokiego pułapu technologicznego, aby przyszłość nie wprowadziła przedsiębiorstwa w koszty przepisywania projektu. Konkurencja wymusi postęp każdej inicjatywy IT ze względu na szeroki dostęp do technologii dzięki Open Source, w innym wypadku zostanie ona porzucona, a klienci wybiorą ofertę innego usługodawcy.

3.1.3 Koncepcja architektury

Rozwiązanie blockchain od strony architektonicznej poszerza model aplikacyjny *klient - serwer* i ujmuje w nim strukturę globalnej, rozproszonej sieci. Biznesowo od aplikacji oczekujemy, że umożliwi klientowi interakcję, prowadząc do osiągnięcia wysokiej konwersji, dlatego instancja blockchain będzie zarządzana przez typową infrastrukturę aplikacyjną, z opracowanym serwerem, odpowiadającym za przekazywanie informacji oraz klientem odbierającym, prezentującym i przekazującym informację powtórnie do serwera (rysunek 3.1).

Częścią łączącą struktury jest serwer, który zapewnia wymianę informacji pomiędzy aplikacją a blockchainem. Obie części są od siebie niezależne, posiadają odrębne bazy danych przechowujące informację odpowiednio: aplikacji oraz łańcucha



Rysunek 3.1. Draft architektury implementacji.
Źródło: opracowanie własne

bloków. Aplikacja integruje blockchain nie wpływając na jego implementację (za pomocą komunikacji REST). Całokształt jest rozproszony w postaci niezależnych węzłów, z odpowiednimi dostosowaniami lokalnymi dla podmiotu, pozostawiając jednak blockchain nienaruszony i transparentny w ramach sieci.

Istotnym jest uzyskanie łatwego sposobu rozprowadzania oprogramowania dla stron zainteresowanych, tak, aby proces był prosty i intuicyjny dla deweloperów. Kluczowe jest również odpowiednie środowisko kolaboracji i wirtualizacji do celów deweloperskich aplikacji. Podejście do realizacji oprogramowania jako usługi wymaga dodatkowych nakładów na dbałość o wysoką jakość rozwiązania.

3.2 Full-stack JavaScript (technologia wiodąca aplikacji)

Wychodząc naprzeciw oczekiwaniom, z szerokiego grona technologii wybrano **JavaScript**, który realizuje wymogi natury biznesu i aplikacji oraz wnosi szereg zalet do oprogramowania zarówno dla części serwerowej, jak i klienckiej (lecz również niektóre wady).

3.2.1 Analiza wartościowych rozwiązań alternatywnych

Z wielu powszechnych technologii można rozważyć m.in. rodzinę języków C: **C++** charakteryzujący się niższym poziomem składni jest cenionym wyborem w kontekście optymalizacji i kontroli danych na poziomie ich struktur, nie jest to natomiast język wygodny do tworzenia *złożonych aplikacji użytkowych*, wymaga większego nakładu pracy na stworzenie oprogramowania, *więcej poświęconego czasu*, a także nie spełnia *wymogu wieloplatformowości*. **C#** jest z powodzeniem wykorzystywany w aplikacjach korporacyjnych, utrzymywany i rozwijany przez Microsoft, jest ustandaryzowanym wyborem dla dużych biznesów. Blockchain jako innowacja *potrzebuje nowych funkcjonalności ponad standaryzacją*, wybór tego języka ponadto *wiąże oprogramowanie z rozwiązaniami wyłącznymi* Microsoftu, takimi jak *Windows*, a aplikacja nowoczesna powinna być niezależna od platformy (czy też zewnętrznej instytucji).

Cennym z punktu widzenia innowacji jest natomiast **Python**, i mógłby z powodzeniem konkurować z wyborem JavaScript. Jedynym niepodważalnym zarzutem jest *brak możliwości wykorzystania tej technologii po stronie klienckiej*. Powstają rozwiązania desktopowe, nie istnieje jednak koncepcja Pythona dla przeglądarek internetowych czy aplikacji mobilnych. Ponadto pozostają dyskusyjne kwestie optymalności rozwiązania i jej wydajności po stronie serwerowej w porównaniu do NodeJS (JavaScript).

Inną technologią popularną w wykorzystaniu korporacyjnym jest **Java**, która spełnia kwestię wieloplatformowości, natomiast zarzucane są jej duże braki w wydajności (szczególnie na platformach Windows). Wiążą się ona też z dużymi kosztami zespołów programistycznych, które wykonują nadmiarową pracę w porównaniu do języków takich jak Python oraz JS. Według Kadivar (2018) implementacja NodeJS w porównaniu do dotychczasowych rozwiązań Java *zredukowała średni czas odpowiedzi o 35%*, pozwoliła na zmniejszenie zapotrzebowania na liczbę programistów o połowę, umożliwiła wykorzystanie jednego języka w całej aplikacji oraz wpłynęła na *płynniejsze osiągnięcie liniowej skalowalność usługi*. Cytując efekt empirycznego testu implementacji: „Surprisingly, they [PayPal] got unbelievable result team of 2 [NodeJS] developers caught the team of 5 java developers in prototype development (...)” (pol. „Zaskakując [PayPal] uzyskał niewiarygodny wynik, zespół 2 programistów [NodeJS] prześcignął zespół 5 programistów java w opracowywaniu prototypu”). Język programowania nie powinien

ograniczać twórcy w procesie tworzenia dzieła.

Coraz większym zainteresowaniem cieszy się język **Go**, który może okazać się wartościowym wyborem dla aplikacji serwerowej, jest on jednak młodszym rozwiązaniem i mogą istnieć braki w istniejących pakietach bądź dokumentacjach.

Duże nadzieje powiązane są z językiem **Swift**, znanym z rozwiązań Apple, od paru lat wprowadzanym jako rozwiązanie Open Source. Analizy wydajnościowe podają, że może okazać się bardziej optymalny niż NodeJS oraz pozwalać na jedną bazę kodu za pomocą transpilacji kodu Swift do JavaScript ze względu na wiele podobieństw i dróg rozwoju obu języków. Nie jest on natomiast tak szeroko wykorzystywany jak JS i wybór tej technologii wiąże się z większym ryzykiem braku łatwo wykorzystywalnych istniejących implementacji, dokumentacji oraz specjalistów.

3.2.2 Natywny JavaScript i serwerowy NodeJS

Rozwiązanie typu **full-stack** oznacza wykorzystanie jednego języka oprogramowania po obu stronach architektury klient - serwer. Dla biznesu wiąże się to z ograniczonymi kosztami dywersyfikacji specjalistów ze względu na technologię, a także wydajniejszą pracą, gdyż skrajne stanowiska mogą siebie łatwiej zrozumieć mając wspólną bazę technologiczną. Dla programistów jest to natomiast brak ograniczeń w rozwoju oprogramowania, możliwość wymienności w różnych strukturach wcześniej rozdzielonych przez technologię, łatwiejsze zrozumienie koncepcji oraz biegłej komunikacji.

Aplikacja z technologią full-stack rozwiązuje problemy wcześniej niemożliwe do pogodzenia, empiryczny przypadek narzędzia analizującego wykresy, który w standardowej aplikacji funkcjonuje w przeglądarce, może łatwo przygotować wykres dla użytkownika po stronie klienckiej. W żadnej innej technologii nie pozwoli jednak na proste opracowanie wykresu po stronie serwera. W technologii NodeJS, po przekazaniu danych uzyskanie obrazu wykresu było *identyczną implementacją niczym po stronie klienta*, rozwiązanie w aplikacji klienckiej automatycznie spełniło wymaganie po stronie serwera, zajmując programiście minuty czasu pracy na skopiowanie rozwiązania z repozytorium klienta na serwerowe. W dotychczasowych przypadkach wymóg wykonania aplikacji klienckiej na serwerze jest niemożliwy i wymusiłby na zespole

wiele godzin pracy nad obejściem problemu. Technologia full-stack to także łatwiejsza integracja, mniejsza duplikacja kodu, wspólne dobre praktyki oraz kompletna umiejętność w rękach programistów i biznesów.

JavaScript jest jedną z nielicznych technologii, które umożliwiają opracowanie wspólnej aplikacji klienckiej niemal na każdą platformę. Prócz natywnego środowiska przeglądarki, w JS za pomocą **Ionic** można opracowywać hybrydowe aplikacje mobilne, które wykorzystując wbudowane przeglądarki systemowe smartfonów, pozwalają uruchomić aplikację na wielu systemach: Android, iOS, Windows Phone w ramach jednej bazy kodu. W innym wypadku każda platforma wymaga osobnej aplikacji natywnej, które wykorzystują technologie kolejno: Java, Swift, C#, co oznaczałoby 3 odrębne zespoły programistyczne o zupełnie różnych umiejętnościach oraz niebotycznych kosztach osiągnięcia wieloplatformowości. Rozwiążanie hybrydowe jest obarczone ograniczoną wydajnością, dlatego dla implementacji wymagających szybkości istnieją rozwiązania takie jak **React Native**, które stosują półhybrydowe podejście z transpilacją kodu JS do kodów natywnych. Inne narzędzia, jak m.in. **Electron** służą opracowywaniu desktopowych aplikacji, funkcjonujących na Windowsie, macOS oraz dystrybucjach Linux. Wykorzystanie JavaScriptu oznacza więc szeroki zakres pokrycia rynku urządzeń, czyli łatwiejszy (oraz wydajny kosztowo) dostęp do klienta.

Rozwój języka JavaScript był bardzo długo ograniczony przez adaptację w przeglądarkach, których powstało wiele o różnym stopniu rynkowym wykorzystania. Każda z nich posiadała własne interpretacje języka, wydłużając proces adaptacji propozycji ECMAScript, które standaryzują i wytaczają szlak rozwoju języka JS.

```
var DecorateListItem = require('package');
var withFancyStyle = require('../components/styled').withFancyStyle;
// ...
var List = {
  defaultListItems: [
    "one", "two", "three"
  ],
  constructor: function () {
    ListComponent.constructor.call(this);
    this.state = { selectedItem: 0 };
  },
  ...
}
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

```

currentlySelectedItem: function (list) {
  var selectedItem = this.state.selectedItem
    return list
      ? list[selectedItem]
      : List.defaultListItems[selectedItem]
}
// ...
};

module.exports = DecorateListItem(withFancyStyle)(List);

```

Listing 3.1. Standard ES5 (ECMAScript 5) od roku 2009 do czasów obecnych. Źródło: opracowanie własne

Propozycje ES5 roku 2009 (listing 3.1) są globalnie wspierane przez topowe przeglądarki na satysfakcyjnym poziomie dopiero od kilku lat, jednak na rok 2018 są one przestarzałe i nieodpowiednie do wydajnego tworzenia oprogramowania, tym bardziej wprowadzania innowacji. Język JavaScript uwolnił się od opóźnienia integracji dzięki wykorzystaniu komplikacji ze źródła do źródła (ang. source to source), czyli tzw. transpilerów, które implementują syntaktykę najnowszych propozycji ECMAScript już od draftów, czyli niemal w przeciągu kilku dni od wysunięcia propozycji (ich adaptacja pozostaje w gestii dewelopera, dobrą praktyką jest integrowanie wyłącznie potwierdzonych propozycji z fazy 3-ciej i 4-tej). Najpopularniejszym z nich jest **Babel**, w zastosowaniu przetwarza kod z wyższych standardów (np. ES8, listing 3.2) do kodu standardu ES5, działającym bez przeszkód na współczesnych przeglądarkach.

```

import DecorateListItem from 'package'
import { withFancyStyle } from 'components/styled'

// ...

@DecorateListItem(withFancyStyle)
export default class List extends ListComponent {
  static defaultListItems = [
    "one", "two", "three"
  ]
  state = { selectedItem: 0 }

  currentlySelectedItem = ({ list }) => {
    const { selectedItem } = this.state

    return list
      ? list[selectedItem]
      : List.defaultListItems[selectedItem]
  }
// ...

```

Listing 3.2. Najnowszy ES2017 (ECMAScript 8). Źródło: opracowanie własne

Pozwala to na zastosowanie najnowocześniejszych rozwiązań już obecnie, zanim będą szeroko wspierane przez przeglądarki, prawdopodobnie po roku 2020. JavaScript cechuje dynamiczny, nieograniczony rozwój, z perspektywą na niemal natychmiastową adaptację innowacji.

Poza cechami unikatowymi, język JS osiąga ponadprzeciętne wyniki w testach wydajnościowych, ze względu na wieloletni przymus adaptacji w przeglądarkach, które dysponowały ograniczonymi zasobami. Szeroko wykorzystywana jest w nim **asynchroniczność** pozwalająca na proste konkurencyjne wykonywanie składni, niczym bardziej utrudnione standardowe wątki. Cechuje go elastyczny, skryptowy charakter, gdyż od zawsze służył do prostego osiągnięcia docelowego efektu na stronach internetowych. Pozwala na łatwe implementacje technologii webowych, takich jak obsługa HTTP, AJAX, WebSocketów, które w innych językach pozostają obca koncepcją. NodeJS prócz pełnej asynchroniczności bazuje na architekturze **middleware**, która umożliwia nieprzerwany przepływ danych i wygodne ich modyfikowanie w trakcie procesu.

Od początku istnienia stack technologiczny dedykowany jest do zastosowań czasu rzeczywistego, takich jak chat, gry multiplayer, współdzielenie dokumentów, itp. Wykorzystanie JavaScript (NodeJS) powinno zapewnić zapas wydajnościowy, kluczowy w dynamicznym wykonywaniu transakcji na blockchainie oraz zawsze pożądany w efekcie finalnym.

3.3 Technologie aplikacji klienckiej

3.3.1 React – Single Page Application

Biblioteka React umożliwia deklaratywne budowanie interfejsów użytkownika za pomocą zastosowania kompozycji oraz komponentów. Komponent jest podstawową jednostką interfejsu, która zawiera informacje o swoim wyglądzie (stylach), a także implementuje istotną dla siebie porcję funkcjonalności, która składa się na aplikację. Kompo-

nenty są wielokrotnie wykorzystywane i w architekturze React zagnieżdża się jedne w drugich, uzyskując strukturę od bardziej złożonej (ogólnej) do prostszej (szczegółowej) (listing 3.3) (Facebook Open Source, 2018d).

```
import React from 'react'  
// ...  
  
import { Col, Row, FormControl } from 'react-bootstrap'  
  
import RedButton from 'components/Buttons/RedButton'  
import Form from 'components/Form'  
  
// ...  
  
@connect(mapStateToProps, mapDispatchToProps)  
export default class Form extends React.Component {  
    static displayName = 'Form'  
  
    state = {  
        form: {  
            country: '',  
        },  
    }  
  
    updateFormInput = (event) => { /* ... */ }  
    sendMessage = () => { /* ... */ }  
  
    render() {  
        const { form } = this.state  
        const { countries } = this.props  
  
        return (  
            <Form>  
                <Row>  
                    <Col md={3} sm={6} xs={12}>  
                        <FormGroup>  
                            <ControlLabel>Select country:</ControlLabel>  
                            <FormControl  
                                componentClass="select"  
  
                                name="country"  
                                onChange={this.updateFormInput}  
                                value={form.country}  
                            >  
                            {  
                                countries.map((country) => (  
                                    <option  
                                        key={country.get('id')}  
                                        value={country.get('name')}  
                                    >{ country.get('name') }</option>  
                                ))  
                            }  
                        </FormControl>  
                    </FormGroup>  
                </Col>  
            </Row>  
        )  
    }  
}
```

Listing 3.3. Przykładowy komponent formularza w React. Źródło: opracowanie własne

React służy do tworzenia aplikacji SPA (ang. Single Page Application), które w porównaniu do przeciętnej strony internetowej, działają bezpośrednio w przeglądarce użytkownika i operują na minimalnych istotnych danych (dzięki REST), zamiast przetwarzać oraz zwracać za każdym razem nowy kod strony. Strona SPA ładuje się oraz funkcjonuje wydajniej, jest bogatsza w interakcje oraz ułatwia budowanie złożonych narzędzi ze względu na zarządzanie własnym stanem w porównaniu do prezentowania danych co odświeżenie (Facebook Open Source, 2018d).

Struktura aplikacji React, dobór narzędzi czy koncepcja architektoniczna jest zupełnie dowolna i leży w gestii dewelopera. Zalecaną praktyką jest korzystanie z gotowych preprojektów, które zawierają w sobie najbardziej powszechnie wykorzystywane wzorce, a ich działanie jest sprawdzone produkcyjne, m.in. popularny **React-Boilerplate**. Cały ekosystem składa się na **framework**, który pozwala na kompletne wykonanie aplikacji klienckiej.

Alternatywnym rozwiązaniem jest znany **Angular 2**, który w podobnym stopniu wykorzystuje kompozycję w celu budowania aplikacji SPA, nie jest natomiast na *elastycznym i wygodnym rozwiązaniem* w porównaniu do React, co podobno nieoficjalnie przyznali jego twórcy. Oparty jest na **TypeScript**, wersji JS od Microsoftu poszerzonej o obiektowe koncepcje znane z C# czy Javy, odseparowane od standaryzacji ECMAScript i przedkładające stabilność kodu ponad nowoczesne rozwiązania.

Często rozpoznawalną technologią jest również **Vue.js**, która stara się łączyć koncepcje React i Angular (wersji 1 i 2) w ramach jednego framework'u. Pomimo wzrastającego znaczenia jest to mniej ustandaryzowane narzędzie o mniejszym stopniu rozpropagowania i mniejszej społeczności programistów.

3.3.2 NextJS – Server Side Rendering

Pomimo ogromnej zalety zastosowania SPA posiada ona kluczową wadę, strona renderowana jest po stronie klienta w przeglądarce, oznacza to, że pierwsze załadowanie oraz zbudowanie strony przeniesione jest na stronę kliencką architektury. Znaczącym

problemem jest również **SEO** (ang. Search Engine Optimization), gdyż wiele indekserów nie jest w stanie odczytać aplikacji SPA i zobaczy jedynie pusty dokument HTML z odnośnikami do kodu JS.

Dla biznesu ruch organiczny jest ważny niemalże zawsze, dlatego warto posiłkować się rozwiązaniem **SSR** (ang. Server Side Rendering) w postaci np. **NextJS**. Wykorzystuje ono możliwości **full-stack JavaScript** i podczas pierwszego załadowania strony SPA, poprzez NodeJS wykonuje pierwszy render aplikacji SPA i zwraca ją jako kod HTML do przeglądarki włącznie ze stanem, która to następnie przejmuje działanie w przeglądarce i funkcjonuje dalej jako standardowe SPA bez renderowania po stronie serwera (Zeit, 2018). Tak opracowana strona w kodzie będzie posiadała treść poszukiwaną przez indeksery.

Oprócz zalety dla biznesu wiąże się to również zwykle (ale nie zawsze) z szybszym czasem ładowania aplikacji, zwiększeniem wydajności oraz wykorzystaniem zasobów serwera podobnie jak w standardowym podejściu, pod warunkiem, że są one wolne do rozdysponowania (nie opóźnimy przez to funkcjonowania wymiany danych).

3.3.3 Progressive Web App

React spełnia koncepcję tzw. **PWA**, inicjatywy budowania internetu aplikacji zbliżonych do mobilnych i odejścia od modelu strony w przeglądarce. Aplikacja PWA wykorzystuje szereg mechanizmów, takich jak Service Worker odpowiadający za pamięć podręczną oraz wsparcie funkcji natywnych, oraz *manifest.json*, który gwarantuje rozpoznawalność PWA w nowoczesnych przeglądarkach (Nath, 2017).

Dzięki Progressive Web App użytkownik może odwiedzić stronę internetową aplikacji, która funkcjonuje identycznie jak natywna aplikacja mobilna i przypiąć ją do głównego panelu smartfona, dzięki czemu nie można jej odróżnić od aplikacji z oficjalnego sklepu Android/iOS (Nath, 2017). Inicjatywa oznacza łatwiejsze zaangażowanie użytkownika, bez potrzeby dystrybucji aplikacji w sklepach, bez oczekiwania na zainstalowanie.

3.3.4 React Redux/Saga – zarządzanie stanem

Asynchroniczność aplikacji znakomicie zwiększa wydajność, natomiast utrudnia jej opracowywanie, ze względu na odejście od prostej idei synchronicznej „receptury” wykonywania kodu krop po kroku. W przypadku asynchronicznym zadania, takie jak pozyskiwanie danych z serwera wykonywane są konkurencyjnie, a narzędzia m.in. **Redux Saga** umożliwiają dogodniejszą pracę.

Rozwój złożoności aplikacji SPA w dużym stopniu uniemożliwił przewidywanie jej stanu, a aby funkcjonowanie aplikacji w postaci globalnej mogło przebiegać bez zarzutu zaproponowano koncepcję **Flow** – jej najpopularniejszą implementacją jest **Redux** (Redux, 2018).

Według idei Redux powstał podział akcji, zdarzeń oraz danych dostępnych w aplikacji na (na podstawie (Redux, 2018)):

Actions – fragment zawierający informację o zdarzeniu, który jest przekazywany do *Store*.

Reducers – nasłuchujące na dane zdarzenia (akcje) i odpowiednio przetwarzające dane do *Store*.

Store – globalny zbiór stanu aplikacji, dostępny z każdego jej miejsca niezależnie.

Dodatkowym elementem jest **Redux Saga**:

Sagas – Redux Saga (2018): „(...) is like a separate thread in your application that's solely responsible for side effects.” (pol. „(...) jest niczym oddzielny wątek w twojej aplikacji, który jest wyłącznie odpowiedzialny za efekty postronne”), umożliwiając pozyskiwanie danych w formie asynchronicznej za pomocą wykorzystania **generatorów**, tak aby można było na nich pracować niczym w kodzie **synchronicznym**.

```
// actions
export const loadProfile      = ({ userId }) => ({ type: LOAD_PROFILE, userId })1
export const loadProfileSuccess = ({ profile }) => ({ type: LOAD_PROFILE_SUCCESS, profile })2
export const loadProfileError   = (errors) => ({ type: LOAD_PROFILE_ERROR, errors })3
// sagas
export function* loadProfileRequest({ userId }) {4
  try {5
    const response = yield call(api.getProfile, userId);6
    yield put(loadProfileSuccess(response));7
  } catch (error) {8
    yield put(loadProfileError(error.message));9
  }
}
```

```

try {
  const resp = yield call(request, `${API}/user/${userId}/profile`, { method: 'GET' })
  8
  9
  10
  11
  12
  13
  14
  15
  16
  17
  18
  19
  20
  21
  22
  23
  24
  25
  26
  27
  28
  29
  30
  31
  32
  33
  34
  35
  36
  37
  38
  39
  40
  41
  1
  2
  3
  4
  5
  6
  7
}

export function* loadSelfProfileLifecycle() {
  yield takeLatest(LOAD_PROFILE, loadProfileRequest)
}

// reducer
export default function usersReducer(state = initialState, action) {
  switch (action.type) {
    case LOAD_PROFILE:
      return state
        .set('isLoading', true)

    case LOAD_PROFILE_SUCCESS:
      return state
        .set('profile', action.profile)
        .set('isLoading', false)

    case LOAD_PROFILE_ERROR:
      return state
        .set('errors', action.errors)
        .set('isLoading', false)

    default:
      return state
  }
}

```

Listing 3.4. Redux + Redux Saga. Źródło: opracowanie własne

Jest to obecnie najbardziej rozpowszechniona koncepcja operowania na stanie i komunikacji z serwerem w React (listing 3.4). Poza Redux warto wymienić **MobX**, który zamiast operować na asynchronicznych **ES6 Promise** funkcjonuje na bazie bardziej złożonych **Observable**, które zawierają o wiele więcej stanów (wywodzące się z Angular2). MobX pozwala za pomocą dekoratorów reagować na zmiany stanu aplikacji bezpośrednio, zamiast oczekiwania na efekt postronny operacji asynchronicznej i doszukiwania odpowiedniego wzorca.

3.3.5 ImmutableJS, Reselect, Code Splitting – wydajność

Zaletą aplikacji SPA jest dynamika, dlatego istnieje wiele rozwiązań skupiających się na poprawie optymalności. **ImmutableJS** stanowi otoczkę (ang. wrapper) obiektów natywnych JavaScript tak, aby każdorazowa próba mutacji obiektu skutkowała powstaniem nowego obiektu (listing 3.5). Ważne jest to w kontekście React, który

wykonuje render za każdym razem, gdy stan komponentu wyższego zmienia się. Gdyby sprawdzenie zmiany stanu miało funkcjonować na obiektach natywnych, każde pole należałyby sprawdzić z osobna iterując, co znacznie pogorszyłoby wydajność. Dzięki użyciu **ImmutableJS** każda zmiana prowadzi do powstania nowego obiektu, czyli jego referencja będzie się różniła, także React automatycznie wykryje całkowitą zmianę (lub jej brak) i wyrenderuje komponenty bez głębokiego sprawdzania pól.

Reselect odpowiada za zapamiętywanie ponownie wykorzystywanych stanów, które często są pozyskiwane w aplikacji w wielu różnych komponentach. Pole formularza pozwalające wybrać kraj z listy wykorzystane w kilku formularzach będzie potrzebowało każdorazowo jednej, niezmiennej listy krajów, toteż dane mogą zostać pobrane za pomocą **Reselect**, który zapamięta ostatni wynik i zwróci go bezpośrednio bez konieczności kosztownego przetwarzania (listing 3.5).

```
// Immutable
const countries = {
  USA: {
    name: 'United States of America',
  },
  PL: {
    name: 'Polska',
  },
};

const countriesMap = Immutable.Map(Immutable.fromJS(countries));
const poland = countriesMap.getIn(['PL', 'name']);

// Reselect
const getStatics = (state) => state.get('statics')

const getCountries = createSelector(
  getStatics,
  (staticsState) => staticsState.get('countries', new Immutable.Map())
)
```

Listing 3.5. ImmutableJS oraz Reselect. Źródło: opracowanie własne

Poza wymienionymi wzorcami stosuje się również tzw. **code splitting**, czyli wydzielenie komponentów z aplikacji do asynchronicznego załadowania ad hoc, zmniejszając rozmiar i dołączając jedynie wymagane jej części, inkrementalnie integrując aplikację krok po kroku. Są to wyłącznie wybrane elementy optymalizacji aplikacji JavaScript, tematyka jest bardzo szeroka i bogata w różnorodność rozwiązań.

3.3.6 Styled Components – stylowanie w kodzie JS

Komponentowa architektura budowania aplikacji SPA skupiła cechy oraz funkcjonalność w ramach możliwych do ponownego wykorzystania struktur, jednakże stylowanie nadal musiało być wykonywane w osobnych plikach *.css. Aby ujednolicić oraz ułatwić pracę ze stylami opracowano rozwiązanie **CSS w JavaScript**, które umożliwia na stylowanie komponentów bezpośrednio w ich kodzie JS (Styled Components, 2018) (listing 3.6).

```
const FancyButton = styled.button'          1
  background: transparent;                  2
  border: 2px solid antiquewhite;           3
  color: antiquewhite;                     4
  font-size: 20px;                         5
  margin: 0 1px;                          6
  padding: 5px 10px;                      7
';
' ;                                         8
```

Listing 3.6. Styled component, CSS w JS. Źródło: opracowanie własne na podstawie (Styled Components, 2018)

3.3.7 Flow – statyczna analiza typowania

Język JavaScript dla elastyczności rozwiązania został opracowany jako *luźno typowany*, jest to w wielu wypadkach zaleta, ale wiąże się również z wieloma implikacjami negatywnymi, mianowicie ryzykiem nieprzewidzianego stanu aplikacji (w efekcie błędu logiki) oraz pogorszoną czytelnością składni, często utrudniającą pracę nad kodem w zespole.

Aby rozwiązać problemy i zwiększyć produktywność programistów dołączających do projektu oraz jednocześnie zachować elastyczność natywnego JS opracowano **statyczny kontroler typowania** (ang. static type checker) (Facebook Open Source, 2018c), którego zadaniem jest analiza typowania zmiennych i reagowanie w czasie rzeczywistym na wszelkie nieprawidłowości podczas rozwoju oprogramowania (listing 3.7).

```
// Basic Flow
function square(n: number): number {
  return n * n;
}

square("2"); // Error!                                1
                                                        2
                                                        3
                                                        4
                                                        5
// React with Flow
type Props = {                                     6
  foo: number,                                      7
}                                                 8
                                                 9
                                                 10
```

```

    bar?: string,
    };
11
12
13
14
15
16
17
18
19
20

class MyComponent extends React.Component<Props> {
  render() {
    this.props.doesNotExist; // Error! You did not define a 'doesNotExist' prop.
    return <div>{this.props.bar}</div>;
  }
}

```

Listing 3.7. Flow, analiza typowania składni. Źródło: opracowanie własne na podstawie (Facebook Open Source, 2018c)

Typowanie zmiennych pozwala na bezpieczniejsze projektowanie oprogramowania, unikniecie kosztownych błędów oraz wpływa na jakość pracy programistów, standaryzując proces operowania na blockchainie w ramach aplikacji.

3.3.8 GraphQL Client

Proces wymiany informacji poprzez REST wymaga każdorazowego tworzenia nowego adresu URL, który będzie zwracał dane według określonego kontraktu. Utrudnia to rozwój złożonych aplikacji i wymusza obustronne wprowadzanie stron, zarówno na serwerze jak i kliencie.

Aby ułatwić oraz zunifikować proces wymiany danych klient-serwer opracowano rozwiązanie **GraphQL**, który wystawia jeden jedyny adres, pod którym dane mogą być dowolnie ekstrahowane ze zdefiniowanych na serwerze schematów (**GraphQL2018**) (listing 3.8).

```

const ExchangeRateQuery = gql`
query rates($currency: String!) {
  rates(currency: $currency) {
    currency
    rate
  }
}

export default graphql(ExchangeRateQuery, {
  props: ({ data }) => {
    if (data.loading) {
      return { loading: data.loading };
    }
    if (data.error) {
      return { error: data.error };
    }
    return {
      loading: false,
      rates: data.rates.rates
    };
  }
}

```

```

})(ExchangeRateList);                                23
// Response                                         24
/*
{
  "data": {
    "rates": [
      {
        "currency": "AED",
        "rate": "3.67",
        "__typename": "ExchangeRate"
      },
      {
        ...
      }
    ]
  }
}
*/                                              37

```

Listing 3.8. Wymiana danych za pomocą GraphQL. Źródło: opracowanie własne na podstawie ([Facebook Open Source, 2018b](#))

3.3.9 SocketIO Client

Oprogramowanie dla sprawnej komunikacji w czasie rzeczywistym wykorzystuje *Sockety*, które są stworzone do nieblokującej, obustronnej wymiany informacji między klientami a serwerem. Ze względu na fakt, że praca z WebSocketami w natywnym JavaScript jest utrudniona i nieefektywna, dlatego powstały gotowe rozwiązania, m. in. **Socket.IO** ([Socket.IO, 2018](#)) (listing 3.9).

```

socket.on('user left', function (data) {
  log(data.username + ' left');
  addParticipantsMessage(data);
  removeChatTyping(data);
});                                              1
2
3
4
5
6

```

Listing 3.9. Reagowanie na zdarzenie WebSocket z Socket.IO. Źródło: opracowanie własne na podstawie ([Socket.IO, 2018](#))

Dzięki wykorzystaniu WebSockets informacje będące w posiadaniu blockchainu będą mogły być przekazywane w czasie rzeczywistym, zwiększając wiarygodność aplikacji i jej wartość biznesową.

3.4 Technologie aplikacji serwerowej

3.4.1 NodeJS Express

Podstawą strony serwera NodeJS jest opracowanie REST API, dzięki któremu możliwy jest swobodny przepływ informacji między klientami a serwerem. Dla ułatwienia procesu stosuje się szkielety (ang. framework), m.in. **Express.js** (Node.js Foundation, 2018), które ułatwiają sprawne i proste opracowywanie API (listing 3.10).

Technologia NodeJS cechuje się wydajnością, jest architekturą w pełni asynchroniczną, dla ułatwienia więc stosuje się typowe dla niej operatory, umożliwiające pracę na kodzie asynchronicznym, **async await** (listing 3.10), w składni zbliżonej do kodu synchronicznego.

```
router.get('/user/:id', async (req, res, next) => {
  try {
    const user = await getUser({ id: req.params.id })
    res.json(user);
  } catch (e) {
    // will eventually be handled by your error handling middleware
    next(e)
  }
})
```

Listing 3.10. NodeJS Express z Async Await. Źródło: opracowanie własne na podstawie (Node.js Foundation, 2018)

Od wersji NodeJS *v8.4* dostępna jest również wersja protokołu **HTTP/2** i pomimo, że jej produkcyjne wykorzystanie wciąż wymaga dopracowania, to rozwiązanie posiada znaczny potencjał na przyspieszenie komunikacji między klientami, a serwerem, dzięki zastosowaniu binarnej transmisji danych, nieblokującego protokołu, obustronnej komunikacji i wielu innych usprawnień.

3.4.2 Mongoose ODM

Praca z technologią bazy danych wymaga odpowiedniego zaprojektowania tabel/kolekcji (w zależności od typu bazy) oraz ich efektywnego wykorzystania w kodzie. W podejściu z ang. *code first*, modelowanie bazy danych przebiega bezpośrednio podczas tworzenia prototypu aplikacji i na podstawie opracowanych modeli odtwarzana jest struktura bazy.

Dla bazy MongoDB stosowane jest popularne rozwiązanie **Mongoose**, które pozwala na zastosowania podejścia code first oraz szybkie opracowanie wymaganych modeli bez bezpośredniego ingerowania w bazę danych (LearnBoost, 2018) (listing 3.11).

```
var blogSchema = new Schema({
  title: String,
  author: String,
  body: String,
  comments: [{ body: String, date: Date }],
  date: { type: Date, default: Date.now },
  meta: {
    votes: Number,
    favs: Number
  }
});
```

Listing 3.11. Schemat Moongose kolekcji MongoDB. Źródło: opracowanie własne na podstawie (LearnBoost, 2018)

3.4.3 Flow

Podobnie jak w aplikacji klienckiej, strona serwera również może korzystać z rozwiązania **Flow** (Facebook Open Source, 2018c) (listing 3.12), zwiększając tym samym stabilność oraz wygodę pracy z oprogramowaniem w NodeJS.

```
/** 
 * Return all items in the inventory
 */
getAll(req: $Request, res: $Response): void {
  res.status(200).json(inventory);
}
```

Listing 3.12. Flow w wykorzystaniu NodeJS. Źródło: opracowanie własne na podstawie (Facebook Open Source, 2018c).

3.4.4 GraphQL Server

W celu wykorzystania rozwiązania klienckiego **GraphQL**, strona serwera musi integrować odpowiednie schematy nałożone na dane, którymi operuje w bazie (Facebook Open Source, 2018b), oraz udostępnić link (listing 3.13), pod którym zapytania GraphQL będą przyjmowane, przetwarzane i zwrocona zostanie odpowiedź z danymi.

```
var schema = buildSchema(`
type Query {
  hello: String
}
```

```
');

var root = { hello: () => 'Hello world!' };

app.use('/graphql', graphqlHTTP({
  schema: schema,
  rootValue: root,
  graphiql: true,
}));
```

Listing 3.13. Prosty pogladowy schemat oraz URL dla GraphQL. Źródło: opracowanie własne na podstawie ([Facebook Open Source, 2018b](#))

3.4.5 SocketIO Server

Integracja rozwiązania **SocketIO** jest zbliżona do części klienckiej, z tą różnicą, że odpowiada za punkt łączący wielu klientów i jednego serwera, opiera się więc na odbieraniu informacji i przekazywaniu jej odpowiednim klientom za pomocą *broadcast.emit()* ([Socket.IO, 2018](#)) (listing 3.14).

```
io.on('connection', function (socket) {
  socket.on('new message', function (data) {
    socket.broadcast.emit('new message', {
      username: socket.username,
      message: data
    });
  });
  // ...
});
```

Listing 3.14. Propagacja nowej wiadomości od nadawcy do obiorców w SocketIO. Źródło: opracowanie własne na podstawie ([Socket.IO, 2018](#))

3.5 Wspomagające narzędzia deweloperskie

3.5.1 Repozytoria Git, Code Review oraz Continuous Integration

Profesjonalne projekty wymagają odpowiedniego środowiska programistycznego, które pozwoli na wydajną pracę wielu osób. Stosowane są **repozytoria GIT**, których zadaniem jest podział prac na członków zespołu, a następnie proste i bezkonfliktowe ich połoczenie w całokształt aplikacji z niezależnych gałęzi rozwoju. Narzędzia te zarządzają uprawnieniami dostępu, wspierają dyskusję nad wprowadzanymi implementacjami (tzw. Code Review) oraz pozwalają na ostateczną decyzję do przyjęcia

bądź odrzucenia implementacji. Zwiększa to wydajność pracy oraz pozwala na łatwe dołączenie nowych programistów w procesie rozwoju aplikacji.

Code review jest ciągłym procesem recenzowania kodu, wpływającym na ujednolicenie praktyk programistycznych w zespole, oraz wskazywanie złych nawyków, nawet u najbardziej doświadczonych programistów.

Koncepcja **Continous Integration** poszerza ideę repozytoriów, dodając do nich możliwości automatycznego wdrożenia kodu na serwery testowe i/lub produkcyjne, bez poświęcania czasu programisty. Służy do zaoszczędzenia nakładu pracy, zwiększenia wygody oraz skupieniu się na procesie rozwoju oprogramowania zamiast rozwiązywania problemów meta-aplikacyjnych (np. **CircleCI**, **TravisCI**).

3.5.2 Testy jednostkowe

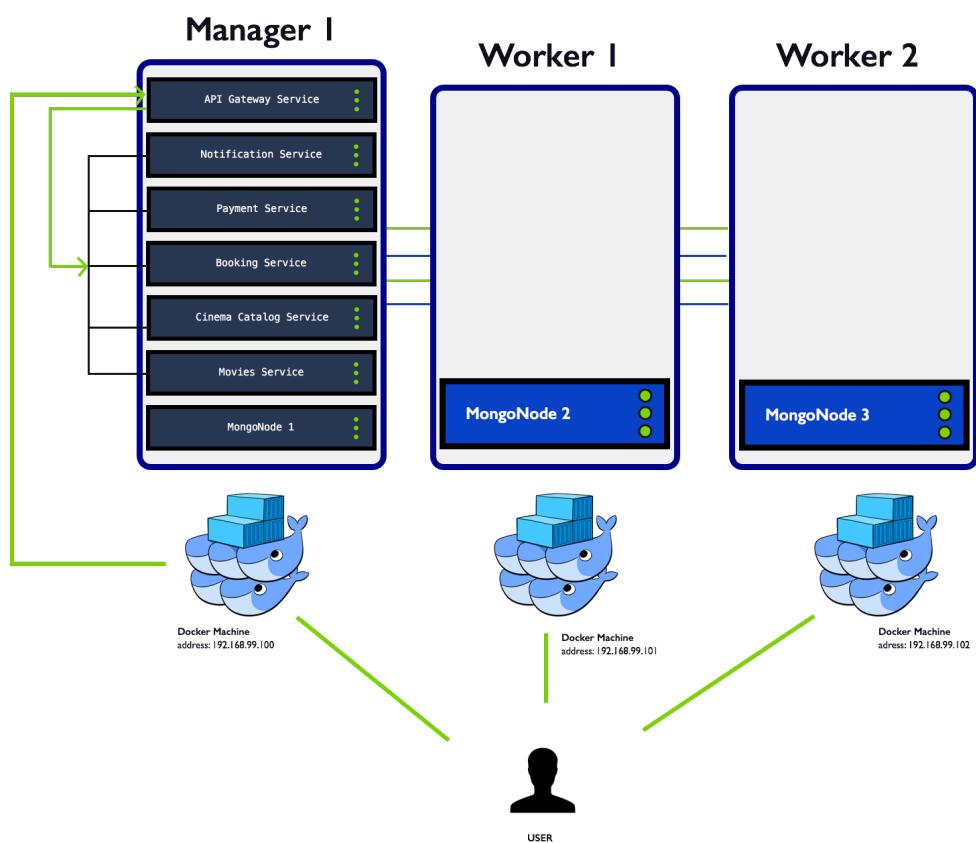
Proces współpracy z wieloma programistami oraz rozwoju złożonego oprogramowania często obarczony jest błędami pochodnymi w wyniku zmiany, która w efekcie powoduje konflikt w innej części oprogramowania (opracowanej nawet wiele tygodni przed wprowadzaną implementacją).

Aby uniknąć sytuacji niestabilności aplikacji w wyniku jej dynamicznego rozwoju, stosuje się **testy jednostkowe**, w czasie wdrażania nowej wersji do testów manualnych, są one uruchamiane automatycznie (dzięki *continuous integration*) i zwracają informację, czy partie aplikacji, dla których testy zostały napisane, rzeczywiście wykonują się zgodnie z ich założeniami. Dla zastosowania JavaScript, po obu stronach serwera i klienta powszechny jest **Jest**, narzędzie testów jednostkowych bez wymogu konfiguracji środowiskowej (Facebook Open Source, 2018a).

Testy jednostkowe są dodatkowym wyznacznikiem jakości oprogramowania, dzięki narzędziom analizującym pokrycie kodu testami (ang. coverage). Im wyższy jest ten współczynnik tym bardziej świadczy to o przykładaniu się dewelopera do stworzenia oprogramowania, które funkcjonuje zgodnie z założeniami.

3.5.3 Docker – wirtualizacja środowisk deweloperskich

Blockchain jako technologia zdecentralizowanego zaufania w procesie rozwoju wymusza opracowywanie rozwiązania w ramach przynajmniej szcątkowej sieci. Deweloperzy z reguły pracują na pojedynczej maszynie, a próba uruchamiania wielu instancji identycznej aplikacji wiąże się z niedogodnością i dużym nakładem pracy na usuwanie konfliktów. Kluczowym jest więc zastosowanie środowiska wirtualizacyjnego, takiego jak **Docker** (Docker Inc., 2018), które pozwala bez bezpośredniej konfiguracji, zasymulować wiele instancji (węzłów) sieci w ramach pojedynczego komputera (lub serwera) (rysunek 3.2).



Rysunek 3.2. Przykład wykorzystania Docker dla opracowywania aplikacji.
Źródło: (Ramirez, 2017)

W porównaniu do standardowych wdrożeń na wielu serwerach znacznie ułatwia pracę programisty w rozwoju aplikacji oraz zmniejsza nakład na posiadanie wielu maszyn i ich odrębne testowanie. Dodatkowo, serwisy hostingowe takie jak **Amazon AWS** od podstaw działają na kontenerach dockerowych, co ułatwia wykonywanie wdrożeń oraz sprawne zarządzanie obciążeniem (ang. load balancing) przez

instancjonowanie np. kolejnej bazy danych według architektury z obrazu, zapewniając tym samym skalowalność usługi.

3.6 Technologie bazodanowe

3.6.1 MongoDB

Z wielu rozwiązań bazodanowych, obiecujących w kontekście rozwoju, są **bazy nierelacyjne** cechujące się liniową skalowalnością, większą prędkością odczytu i/lub zapisu oraz konkurencyjnością w tematyce tworzenia nowoczesnych aplikacji. Pozwalają one na przechowywanie większego wolumenu danych, łączenia wielu baz w klastry oraz tworzenie kopii zapasowych danych w ramach sieci.

MongoDB to nierelacyjna, dokumentowa baza danych, przechowująca informacje w formacie JSON w ramach kolekcji (listing 3.15) (MongoDB, Inc., 2018a). Prócz zalet bazy nierelacyjnej, jest to rozwiązanie przeznaczone do tworzenia aplikacji, stawiające na łatwość wykorzystania, wygodę języka zapytań i szybkie prototypowanie. MongoDB jest również natywnie zbliżona do JavaScript, ze względu na identyczny format danych JSON (ang. JavaScript Object Notation), oraz popularnie wykorzystywana w zbliżonych architekturach technologicznych, dzięki czemu jest odpowiednio udokumentowana i bogata w gotowe implementacje do wykorzystania.

```
{  
    "_id": ObjectId("549fffb7e0658f67708b457f"),  
    "type": "person",  
    "email": "example@example.com",  
    "username": "John",  
    "password": "3bmvgv7vVgg8FBmDToPJvXNzUDZQVlsnmr4NUaPStd5d9h+S+0  
        QiPAcwgLUL6c9Ny4guVVERPate2cA==",  
    "roles": [  
        "ROLE_USER"  
    ],  
    "userMetadata": {  
        "lastActivity": new Date("2016-10-29T07:16:32+0200")  
    },  
    "profile": {  
        "$ref": "people",  
        "$id": ObjectId("549fffb7e0658f67708b4581"),  
        "$db": "database_name",  
        "_doctrine_class_name": "Domain\\Person\\Model\\Person"  
    }  
}
```

Listing 3.15. Przykładowy obiekt kolekcji *users* w MongoDB. Źródło: opracowanie własne

Alternatywami rozwiązań nierelacyjnego jest **baza relacyjna**, która stawia na bezpieczeństwo danych, lecz ogranicza skalowalność i prędkość odczytu/zapisu. Utrudnia również pracę programistom przez wymuszenie zastosowania relacji i ścisłe trzymanie się schematu, więc każdorazowa próba przemodelowania schematu wiąże się z potrzebą jego zmiany lub ponownego instancjonowania bazy (m. in **MySQL**, **PostgreSQL**, **TSQL/ OracleDB**). Z kategorii baz nierelacyjnych możliwe jest wykorzystanie bardziej nietypowych rozwiązań **kolumnowych** lub **grafowych**, są to natomiast przypadki specyficzne i zależne od rozwiązywanego problemu biznesowego, który jest poza przypadkiem użycia aplikacji z blockchainem.

3.6.2 BigchainDB

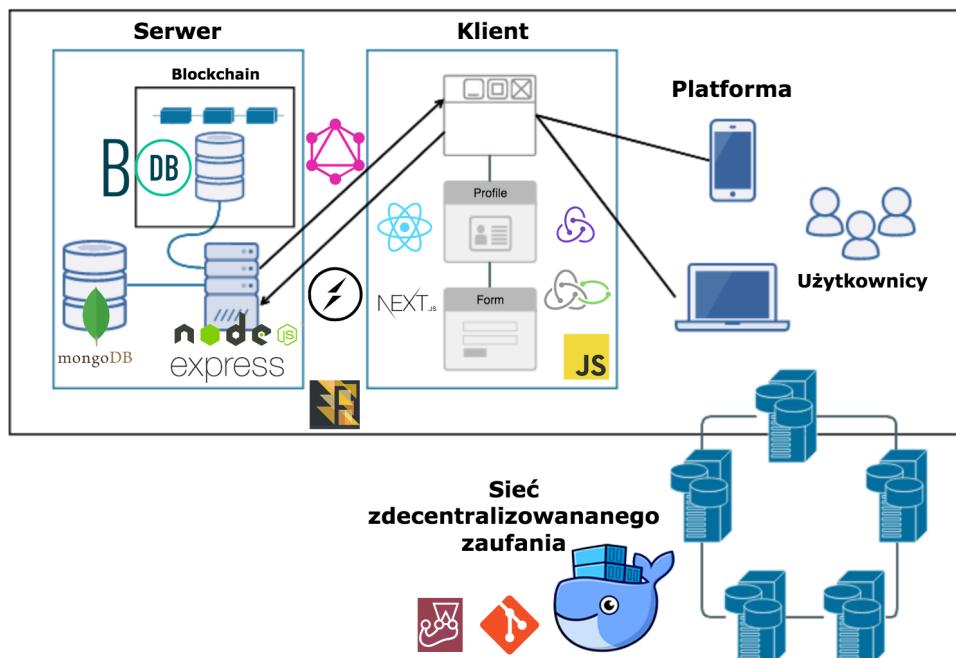
Blockchain klasyfikowany jako technologia bazodanowa (rozproszona baza danych) jest reprezentowany przez szereg narzędzi deweloperskich o różnym stopniu stabilności, udostępnionych na licencji Open Source do użytku implementujących go deweloperów (czyli adaptujących biznesów). Między innymi jest to **BigchainDB**, skalowalna blockchainowa baza danych (BigchainDB GmbH, 2018c), który jest gotowym rozaniem blockchain z opracowanym API, które pozwala interoperacyjnie z poziomu aplikacji operować na łańcuchu bloków i wdrażać rozwiązanie blockchain.

W procesie integrowania blockchain w aplikacji możliwe są do wykorzystania otwarte, blockchainowe platformy aplikacyjne takie jak **Ethereum** czy **Lisk**. Dla przypadku opracowywania rozwiązania zamkniętego, wewnątrz grupy firm spełniających wspólny proces biznesowy dostępny jest również m.in. prowadzony przez IBM **Project Hyperledger**, który posiada wsparcie globalnych organizacji finansowych i przeznaczony jest dla rozwiązań korporacyjnych fintech.

Mechanizmy istoty funkcjonowania, motywacja oraz krytyka rozwiązania są częścią tematyki pracy i zostały szczegółowo poruszone w następnym rozdziale, dedykowanym narzędziu **BigchainDB**.

3.7 Kompletna architektura

Analiza narzędzi umożliwiła zaproponowanie najlepszych możliwych technologii w tworzonej aplikacji blockchain, prowadząc do uzyskania nowatorskiej architektury według rysunku 3.3.



Rysunek 3.3. Kompletna architektura po procesie doboru technologii.
 Źródło: opracowanie własne

Każde z zastosowanych rozwiązań stanowi wartość dodaną dla projektu oraz dla biznesu, gwarantując prawidłowe rozwiązanie problemu biznesowego zgodnie ze standardami i dbałością o najwyższą jakość i odpowiedni poziom bezpieczeństwa.

3.8 Podsumowanie

W rozdziale omówiono aspekty natury technologicznej i biznesowej motywacji doboru architektury aplikacji blockchain, przeprowadzono analizę dostępnych rozwiązań spełniających założenia pierwotnej architektury, porównując je z narzędziami konkurencyjnymi oraz wskazując na ich wady oraz zalety wykorzystania, każdorazowo omawiając ich zasadę funkcjonowania z możliwie przejrzystym przykładem. Stopniowo wprowadzono stronę aplikacji klienckiej, serwerowej oraz części

bazodanowej, prezentując rozwiązania od elementu najbardziej ogólnego po dobró narzędzi zależnych. Omówiono także wybór rozwiązań wspierających rozwój oprogramowania blockchain.

Motywem przewodnim doboru komponentów architektury były czynniki innowacyjności oraz stabilności, kosztem ustandaryzowania, wskazując jednocześnie na ulotną naturę technologii blockchain. Technologie zaproponowane w aplikacji blockchain są elastyczne na tyle, że pozwolą jej na nieograniczony rozwój na przestrzeni lat oraz będą razem z nią ewoluowały w miarę dynamicznego rozwoju języka jakim jest JavaScript.

Uzyskana kompletna architektura jest sprawdzona produkcyjnie na przestrzeni lat, wielokrotnie testowana w rzeczywistości biznesowej, a tak opracowany stos technologiczny może zostać z powodzeniem wykorzystany w projektach o dowolnej skali, również globalnej, stanowiąc perfekcyjną „ramę” rozwiązania blockchain dla adaptującego biznesu. Stanowi to punkt odniesienia do właściwego elementu implementacji technologii zdecentralizowanego zaufania w aplikacji, czyli w istocie blockchainu, którego osobnej analizie poświęcony został następujący rozdział.

Rozdział 4

BigchainDB – skalowalna baza danych dla blockchain

Najistotniejszym elementem aplikacji będącej tematem pracy jest odpowiedni dobór technologii blockchain, która spełni założenia techniczne oraz biznesowe pracy. Rozdział poświęcono motywacji oraz krytyce rozwiązania, omówieniu jego technologicznych aspektów funkcjonowania, a także prezentacji atrybutów, którymi cechuje się wybrane narzędzie.

Ważnym jest, aby na dostatecznie wysokim poziomie zrozumieć zasadę funkcjonowania blockchainu, a następnie odpowiednio odnieść ją do części implementacyjnej tak, aby proces biznesowy został z pełnym sukcesem zaadaptowany w myśl idei technologii zdecentralizowanego zaufania, prowadząc do wypracowania nowej wartości dodanej przedsiębiorstwa.

4.1 Motywacja oraz koncepcja rozwiązań

Według autorów narzędzia BigchainDB GmbH (2018c): „*BigchainDB [has] high throughput, sub-second latency and powerful functionality to automate business processes, BigchainDB looks, acts and feels like a database with added blockchain characteristics.*” (pol. „BigchainDB [posiada] wysoką przepustowość, mniej niż sekundowe opóźnienie i potężną funkcjonalność do automatyzacji procesów biznesowych, BigchainDB działa i sprawia wrażenie bazy danych z dodanymi cechami blockchainu.”). Charakterystyka rozwiązania

nia umożliwia pozbycie się części problemów cechujących blockchain, m.in. problemów ze skalowalnością oraz skupia się na ułatwieniu programistycznego oprogramowaniem w osiągnięciu celu biznesowego.

BigchainDB został opracowany w języku **Python** z wykorzystaniem aktualnych standardów i umożliwieniem łatwego rozwijania oprogramowania dla potrzeb społeczności. Wybór tego języka stanowi **zaletę w kontekście czytelności i prostoty**, może jednak okazać się *wadą wydajnościową* (w porównaniu do języków rodziny C). Dla celów implementacji rozwiązań zamkniętych w obrębie wybranych podmiotów gospodarczych, to ograniczenie nie powinno stanowić problemu. Podejście BigchainDB do decentralizacji informacji opiera się na koncepcji **zasobu** (ang. **asset**) (BigchainDB GmbH, 2018b). Zasób może zostać utworzony w ramach transakcji *CREATE* lub przekazany pomiędzy stronami dzięki transakcji *TRANSFER*, te natomiast zgodnie z ideą zostają dołączone do powiązanych bloków.

W tworzeniu rozwiązania BigchainDB autorzy sugerują, aby **zamiast skupiać się na procesie biznesowym, zwrócić uwagę na zasób**, który jest jego przedmiotem, takim jak np. informacja o rozliczeniu pozycji faktury, ale również przedmiotem fizycznym takim, jak samochód czy rezerwa złota (BigchainDB GmbH, 2018b). W istocie więc zasób reprezentuje informację powiązaną z jej posiadaczem oraz dołączone do niej metadane.

Blockchain jest współdzielony przez wybrane strony i funkcjonuje spójnie w ramach wspólnej sieci. Strony uczestniczą w sieci na równych zasadach zgodnie z mechanizmem consensusu, którym w BigchainDB jest wariant dowodu stawki (ang. Proof of Stake), oparty na zasadzie głosowań na transakcje między węzłami.

4.2 Technologiczna istota funkcjonowania

4.2.1 Integracja w sieci oraz model danych

Autorzy narzędzia wyróżniają następującą terminologię sieci blockchain (na podstawie (BigchainDB GmbH, 2018d)):

Węzeł (BigchainDB Node) – instancja serwera BigchainDB i powiązanego z nią opro-

gramowania kontrolowana przez jedną, niezależną jednostkę.

Klaster (BigchainDB Cluster) – sieć węzłów, łączących niezależne jednostki w ramach wspólnego rozwiązania blockchain.

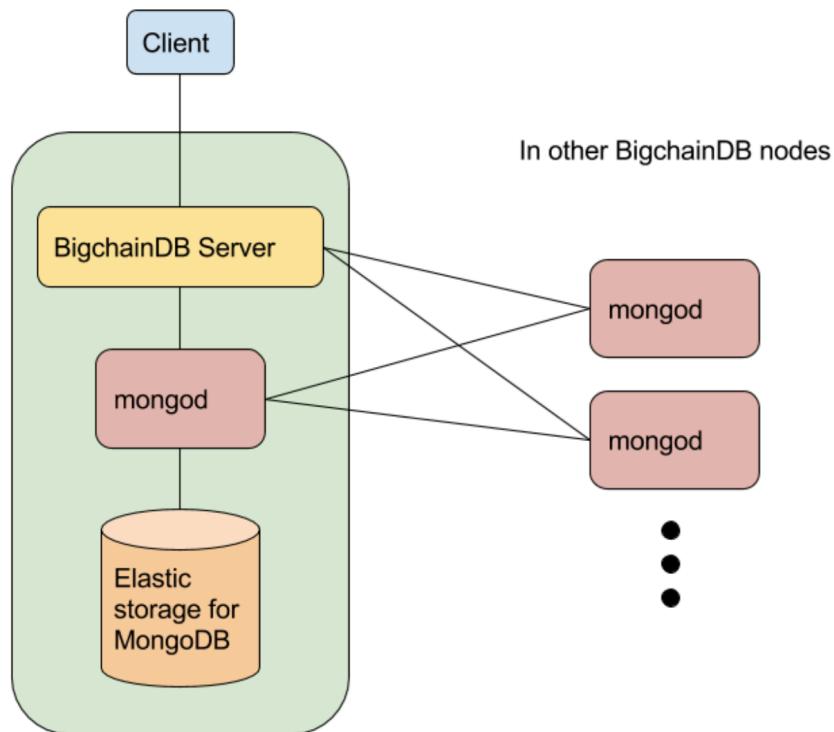
Konsorcjum (BigchainDB Consortium) – ogół jednostek, organizacji itp., które składają się na klaster oraz odpowiadają wspólnie za podejmowanie decyzji.

Konsorcjum zrzesza podmioty niezależnie operujące węzłami i jest jednostką o charakterze biznesowym (organizacyjnym), klaster natomiast reprezentuje sieć węzłów, dzięki którym sieć blockchainowa może funkcjonować. W praktyce od strony technologicznej BigchainDB dla utworzenia sieci wykorzystuje istniejący w MongoDB mechanizm **replica set**, który służy do budowania wieloinstancyjnych architektur bazodanowych (BigchainDB GmbH, 2018a). Według dokumentacji MongoDB, Inc. (2018b) mechanizm ten to: „*A replica set in MongoDB is a group of mongod processes that maintain the same data set*” (pol. „replica set w MongoDB to grupa procesów mongod, które utrzymują ten sam zestaw danych”). Koncepcja idealnie uzupełnia ideę rozproszonej sieci oraz stanowi wieloletni, sprawdzony produkcyjnie standard dla decentralizacji, możliwy do zaadaptowania w rozwiązaniu blockchain. Klaster jest więc powiązany za pomocą *replica set* pomiędzy niezależnymiinstancjami MongoDB oraz serwerów BigchainDB (rysunek 4.1), umożliwiając funkcjonowanie sieci blockchain.

4.2.2 Zasób jako podstawowa jednostka informacyjna transakcji

Łańcuch składając się z bloków i zawartych w nim transakcji posiada wewnątrz **zasoby**, które reprezentują informację istotną dla biznesu. Według BigchainDB GmbH (2018b) jako *zasób* traktować można m.in. prawo własności czy token – przykłady zawarte w tabeli 4.1.

Dowolność postaci zasobu w BigchainDB pozwala na elastyczne wykorzystanie narzędzia w implementacji procesu biznesowego, umożliwiając opracowanie niemal dowolnej koncepcji wykorzystania blockchain dla potrzeb przedsiębiorstwa.



Rysunek 4.1. Model sieci blockchain oraz węzły w BigchainDB.
 Źródło: (BigchainDB GmbH, 2018b)

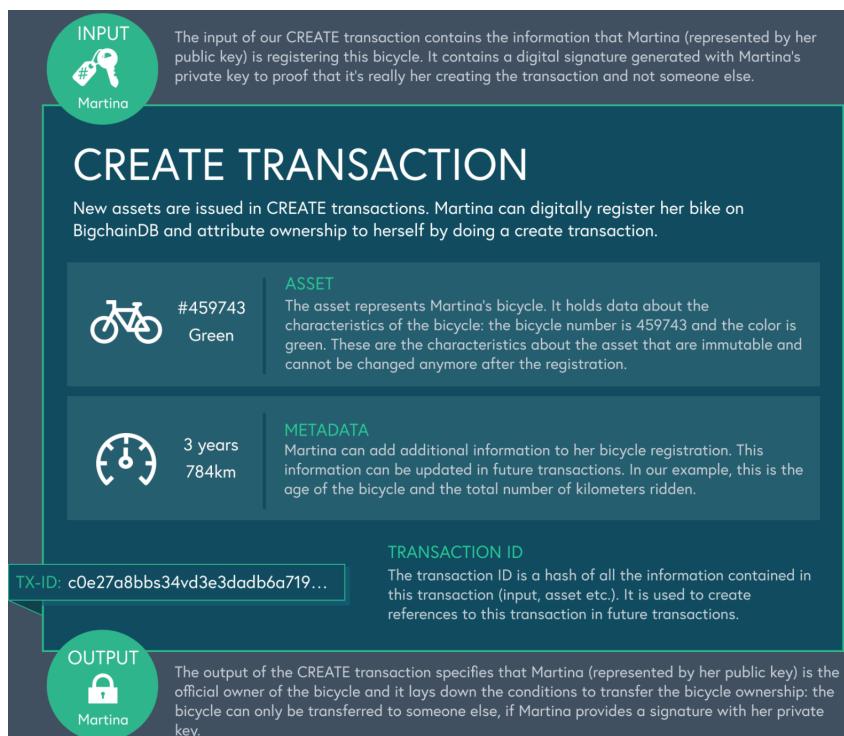
Tabela 4.1. Przykładowe zasoby w przypadkach użycia blockchain dla biznesu.

Zasób	Scenariusz	Przykłady usług/produktów
Prawo własności	Śledzenie oraz przekazywanie posiadania	np. nieruchomości, komputer
Token	Platformy aplikacyjne bazujące na tokenach	koncepcje Ethereum, smart contracts
Dokument wersjonowany	Transparentne edycje	systemy globalnej współpracy
Seria czasowa danych	Obrazowanie zmian w czasie	np. IoT dla trans. danych sensorycznych
Przechowywanie stanów	Bezpieczne, transakcyjne zmiany stanów	ścisłe współpracowanie z postacią danych
Zezwolenie (permisje)	Nadawanie uprawnień	hierarchie aplikacyjne i rzeczywiste

Źródło: opracowanie własne

4.2.3 Typy transakcji łańcucha bloków

Dodanie **zasobu** do blockchainu jest początkiem istnienia informacji istotnej biznesowo w łańcuchu bloków. Aby możliwe było śledzenie bądź przekazanie posiadania do drugiej osoby, należy utworzyć jej ślad w blockchainie, co odbywa się za pomocą transakcji typu *CREATE* (rysunek 4.2).



Rysunek 4.2. Opis transakcji *CREATE*.

Źródło: (BigchainDB GmbH, 2018b)

Istniejący zasób w blockchain po transakcji jest powiązany z konkretnym użytkownikiem za pomocą kryptografii, par kluczy prywatny/publiczny składających się na cyfrowy podpis zasobu. W przypadku BigchainDB wykorzystywany jest standard kryptograficzny **ED25519-SHA-256**, a klucze przechowywane są w kodowaniu **Base58**. Następnie zasób może zostać **przekazany** do innego użytkownika sieci dzięki wykorzystaniu transakcji *TRANSFER* (rysunek 4.3).

Po zweryfikowaniu zgodności cyfrowego podpisu, użytkownik przekazuje posiadany przez siebie zasób do innego użytkownika, stary podpis zostaje zastąpiony nowym, zawierającym sygnaturę kolejnego właściciela, wskazując na jego niezależną własność. Transakcja natomiast na zasadzie mechanizmu consensusu zostaje zatwierdzona.



Rysunek 4.3. Opis transakcji TRANSFER.

Źródło: (BigchainDB GmbH, 2018b)

zona i rozpropagowana w ramach sieci, świadcząc o **prawie do własności** w świecie rzeczywistym.

4.2.4 Dowód funkcjonowania

Kluczowy proces, jakim jest zatwierdzanie nowych bloków w sieci blockchain, opiera się na zasadzie głosowania (ang. voting), zgodnie z którą każdy z węzłów posiada prawo głosu, a gdy jeden z nich utworzy nowy blok i będzie usiłował dodać go do łańcucha w sieci, węzły wspólnie zagłosują, czy jest to blok prawidłowy, analizując zawarte transakcje z zasobami. Proces ideowo zbliżony jest do demokracji, gdzie każdy z węzłów niezależnie dba o spójność swojej kopii rozproszonego rejestru.

Osobna kolekcja bazodanowa przechowuje w ramach blockchain wspólną strukturę głosu (listing 4.1), wskazując na decyzję danego węzła w kontekście do dodawanego bloku.

```
{
  "node_pubkey": "<Klucz publiczny głosującego węzła>",
  "vote": {
```

1

2

3

```

    "voting_for_block": "<ID bloku, na który węzeł głosuje>",
    "previous_block": "<ID bloku poprzedzającego, do tego, na który głosuje węzeł>",
    "is_block_valid": "<true LUB false>",
    "invalid_reason": null,
    "timestamp": "<Uniksowy czas stworzenia głosu>"
},
"signature": "<Sygnatura wewnętrzna obiektu głosu>"  

}

```

Listing 4.1. Poglądowa struktura głosu w blockchain. Źródło: (BigchainDB GmbH, 2018a)

System głosowań jest transparentną reprezentacją decentralizacji zaufania poprzez każdąinstancję blockchainu, gdzie niezależnie každa ze stron na własną rękę waliduje bloki, eliminując wszelkie próby zagrożenia konsorcjum.

4.3 Przykładowe przypadki użycia

Autorzy oprogramowania BigchainDB przygotowali szereg dostępnych poradników obrazujących przypadki wykorzystania ich technologii do adaptacji blockchain¹. Główne grupy dzielą branże na wykorzystanie w obszarach: własności intelektualnej, identyfikowania tożsamości, łańcucha dostaw, czy zastosowań administracyjnych.

Zastosowanie znalazły scenariusze śledzenia zasobów, w postaci dzieł sztuki, które są unikatowe w skali globalnej i mogą z powodzeniem zostać sklasyfikowane z charakterystycznymi dla nich informacjami w blockchainie. Inny przypadek, utworzenia cyfrowego odwzorowania pojazdu IoT pokazuje jak można pogodzić dane sensoryczne z przechowywaniem w łańcuchu bloków, pozwalając na wypracowanie nowego rodzaju usług. Poradnik RBAC (ang. role-based access control) obrazuje globalny rejestr przywilejów powiązanych z tożsamością oraz możliwości nadawania uprawnień. Natomiast przykład tokenów ukazuje możliwość współdzielenia jednego zasobu przez wielu użytkowników dla potrzeb specyficznych zastosowania blockchain.

Opracowania są uzupełnione w motywację biznesową oraz gotowe listingi kodu, które umożliwiają deweloperom proste zrozumienie podstawowych koncepcji blockchainu BigchainDB. Stanowią też profesjonalny materiał biznesowy dla firm rozważających adaptację rozwiązania.

¹<https://www.bigchaindb.com/usecases/>

Ukazane przykłady uświadamiają zainteresowanych, że osiągnięcie rozmaitych realizacji procesów biznesowych z blockchain jest możliwe dzięki ich oprogramowaniu.

4.4 Podsumowanie

W rozdziale zawarto analizę rozwiązania blockchain BigchainDB, od jej istoty funkcjonowania po technologiczne aspekty działania, możliwe opcje działania na łańcuchu bloków oraz zasadę decentralizowania z wykorzystaniem instancji procesów *mongod*. Zwrócono uwagę na podejście do zasobu zamiast procesu biznesowego oraz podano przykładowe scenariusze wykorzystania rozwiązania w praktyce.

Dysponując kompletną architekturą oprogramowania oraz klarownym zrozumieniem istoty funkcjonowania rozwiązania blockchain w postaci **BigchainDB**, następnym krokiem w pracy jest praktyczna implementacja procesu biznesowego, która opracowana została w kolejnym rozdziale.

Rozdział 5

Implementacja sieci przedsiębiorstw funkcjonujących na blockchainie

Rozdział dotyczy syntezy technologii zdecentralizowanego zaufania i biznesu, w postaci procesu biznesowego, którego implementacja oraz wdrożenie zostanie zrealizowane w formie praktycznej - aplikacji dowodu poprawności (ang. Proof of Concept) dla rozwiązania postawionego problemu.

Sekcje poświęcone zagadnieniu biznesowemu stanowią kontekst oraz wprowadzenie do problemu, który potencjalnie rozwiązuje blockchain. Omówiono w nich inspiracje idei technologii zdecentralizowanego zaufania w biznesie, dokonano analizy sektora zastosowania oraz opracowano proces biznesowy, będący przedmiotem badania.

Techniczne sekcje rozdziału omawiają architekturę aplikacji blockchain, przystosowaną pod kątem wdrożenia w instytucjach, implementację praktyczną procesu biznesowego w postaci listingów kluczowych elementów aplikacji oraz ich krytyczną analizę. Ostateczna forma weryfikacji badanej koncepcji zawiera się w sekcjach ewaluacji oraz próby falsyfikacji, których wynik rzutuje na osiągnięcie celu pracy badawczej.

Szczegóły czynności dokonanych w procesie badawczym zawarto w opisie metodyki realizacji eksperymentu, natomiast wnioski wynikające z efektów pracy przedstawiono w ostatniej sekcji rozdziału.

5.1 Inspiracja wdrażania blockchain w biznesie

Blockchain jako zdecentralizowany rejestr informacji umożliwia **redukcję kosztów podmiotu gospodarczego** poprzez eliminację **pośredników**, którzy dotychczas musieli tę informacją zarządzać. W istocie stanowi to szansę na prostszą i efektywniejszą *wymianę informacji* pomiędzy powiązanymi stronami biznesowymi, przekazywanie wiedzy *w czasie rzeczywistym* oraz pełną *transparentność* podejmowanych czynności, dzięki historii łańcucha bloków.

Wiele przedsięwzięć o charakterze globalnym jest trudno adaptowalnych lub wręcz niemożliwych do wdrożenia w zastosowaniu biznesu małego i średniego szczebla, toteż celem jest **opracowanie implementacji blockchain, która z powodzeniem może zostać wykorzystana w istniejącej sieci kontrahentów** i wzbogacić tej sieci o **nową wartość dodaną**.

Opracowany w pracy scenariusz będzie możliwy do powielenia w wielu branżach przy różnych procesach biznesowych, wychodząc z założenia, że biznes to przede wszystkim **sieć powiązań**, którą blockchain jest w stanie *z powodzeniem uporządkować dla osiągnięcia korzyści i wspólnego dobra* podmiotów gospodarczych.

W ramach wzrostu skali wdrożeń, niezależne sieci blockchainowe będą mogły współpracować i wymieniać się informacjami, dążąc do automatyzacji procesów biznesowych (takich jak łańcuch dostaw, księgowość, zasoby) na skalę globalną biznesu z postępem dla rozwoju społeczeństwa.

5.2 Metodyka przeprowadzenia wdrożenia

Realizacja celu pracy stanowi syntezę podejścia ekonomicznego w informatyce, integrując konkretne rozwiązania technologiczne na potrzeby istniejącego problemu biznesowego. Łącząc obie dziedziny w zakresie informatyki ekonomicznej, metodyka pracy polega na zidentyfikowaniu i opisaniu procesu biznesowego oraz przeplata się z inżynierskim podejściem do projektowania oprogramowania. Realizacja badania przebiegała będzie w krokach opisanych poniżej.

1. Analiza problemu biznesowego:

- a) identyfikacja stron procesu,
 - b) dyskusja potencjalnych rozwiązań,
 - c) wypracowanie wartości dodanej dla przedsiębiorstwa w proponowanym rozwiązaniu.
- 2. Wyekstrahowanie procesu biznesowego (zasobu) do adaptacji na blockchainie:**
- a) opis szczegółowy wybranego rozwiązania,
 - b) wstępna koncepcja implementacji.
- 3. Opracowanie zarysu architektury technologicznej wdrażanego rozwiązania:**
- a) strona aplikacji klienckiej,
 - b) strona aplikacji serwerowej,
 - c) współlegzystowanie elementów.
- 4. Przystosowanie architektury i technologii do wdrożenia:**
- a) implementacja strony klienta, serwera oraz środowiska aplikacyjnego,
 - b) adaptacja rozwiązań deweloperskich, szablonów (ang. *boilerplates*) oraz wstępnej struktury docelowej aplikacji.
- 5. Implementacja procesu biznesowego w technologii zdecentralizowanego zaufania:**
- a) opis procesu implementacji,
 - b) analiza elementów kluczowych w gotowym rozwiązaniu,
 - b) całokształt implementacji w myśl idei blockchain.
- 6. Ewaluacja procesu biznesowego na przykładowym scenariuszu użycia:**
- a) sprawdzenie, czy osiągnięto nową wartość dodaną,
 - b) monitorowanie procesu w ramach całej sieci.
- 7. Obserwacja oraz analiza rezultatu między wieloma węzłami:**
- a) efekt przeprowadzenia procesu,
 - b) przewaga wartości dodanej w porównaniu do standardowego podejścia.
- 8. Podjęcie próby falsyfikacji bloku w ramach sieci blockchain:**
- a) zdefiniowanie procesu ataku,

b) ewaluacja oraz monitorowanie szkody w ramach całej sieci.

9. Analiza efektu ataku na sieć:

a) ocena skali ataku oraz ryzyka niesionego dla podmiotu.

10. Wnioski oraz komentarz do uzyskanych wyników.

W efekcie kompletny proces wdrażania blockchainu zazębia się z wymogami biznesowymi, a także aplikacyjnymi, umożliwiając prawidłowe rozwiązywanie problemu biznesowego oraz wypracowanie nowej wartości dodanej dla podmiotu gospodarczego.

5.3 Analiza problemu biznesowego

Sztuka będąca wyrazem twórczości artystycznej jest manifestem dobra kulturowego społeczeństwa. Niejednokrotnie niedoceniana niemal na równi z przecenianiem jej realnej wartości, prowadząc do dysproporcji, między nieznaczącym dziełem przeciętnego, a uważanego za wybitnego - artysty.

Egzystencja rynku sztuki opiera się na indywidualach tworzących dzieła oraz jednostkach stanowiących o ich wartości: **muzeach** przechowujących i konserwujących dzieła o walorach kulturowych, **instytucjach** katalogujących i archiwizujących informacje o transferach dzieł, wystawach, twórcach oraz właścicielach, a także docelowo **domach aukcyjnych**, nadających oraz spienieżających dzieła i przekazujących realną wartość między konkretnymi interesariuszami.

Komercyjny rynek sztuki w rozumieniu Bartosiewicz (2016): „(...) w naszym kraju [Polska] w dalszym ciągu pozostaje elitarnym hobby”, a jego głównym przedmiotem jest **malarstwo** (ponad 60,4% wolumenu). Miejsce dominujące na rynku sztuki zajmują cztery największe domy aukcyjne: Desa Unicum, Agra-Art, Rempex i Polswiss Art, osiągając wspólnie obroty na kwotę ponad 115 mln PLN w roku 2015 (Bartosiewicz, 2016).

Prócz elementu sztuki, np. XIX-wiecznego obrazu, o jego wartości stanowią wszelkie informacje dotyczące egzemplarza, ze względu na *unikatowość dzieła stworzonego jednokrotnie przez niepowtarzalnego artystę*. Dane takie jak np.: informacje o autorze, okolicznościach powstania, materiałach wykonania, pochodzeniu, certyfikacji,

transferach pomiędzy wystawami oraz właścicielami prywatnymi określa się jako **proweniencję** obrazu, kompletną historię elementu sztuki, od jego stworzenia poprzez wszelakie ingerowanie w jego aktualny stan. Wspólnym elementem łączącym kontrahentów rynku sztuki stanowi więc **obraz oraz jego prowienicja** (rysunek 5.1), będąc głównym zasobem procesu biznesowego handlu dziełami sztuki na rynku (rysunek 5.2).



Tadeusz Brzozowski

(1918 Lwów - 1987 Rzym)

"Fraucymer", 1959 r.

olej/płótno, 165 x 129 cm

opisany na bieżtramie: "165 x 129" oraz na odwrociu papierowe nalepki wystawowe z opisem pracy: z Museum of Modern Art w Nowym Jorku, ze Seattle World's Fair, Gres Gallery w Waszyngtonie oraz Minneapolis Institute of Arts

Cena wylicytowana: 800 000 zł ~

Estymacja: 500 000 - 800 000 zł ~

Pochodzenie

- Gres Gallery, Waszyngton, Stany Zjednoczone
- kolekcja prywatna, Stany Zjednoczone

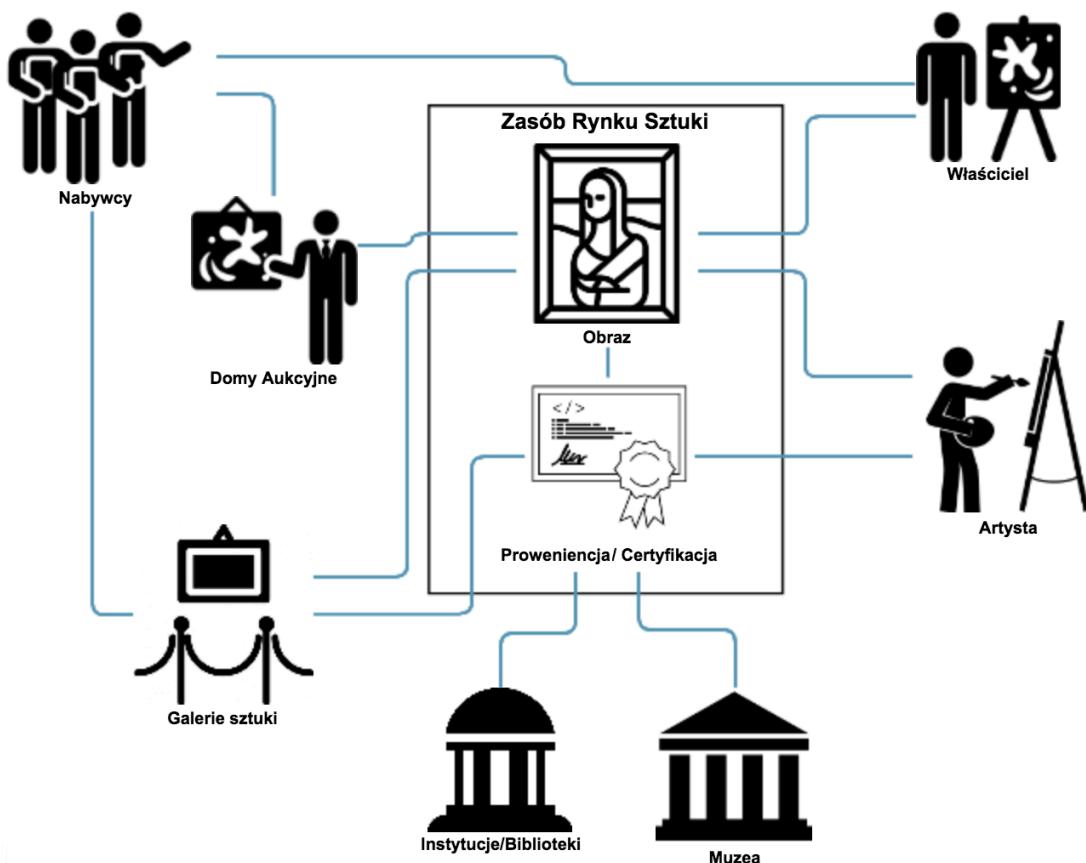
Wystawiany

- "Art since 1950 - American and International Seattle World's Fair", Rose Art Museum, Seattle, 21.04-21.10.1962
- "15 Polish Painters", Museum of Modern Art, Nowy Jork, 1961 (wystawa objazdowa: Carnegie Institute, Pittsburgh; Minneapolis Institute of Art, Minneapolis; Washington University, St. Louis; William Proctor Institute, Utica Munson, 1962; Museum of Fine Arts, Montreal, 1962; The National Gallery of Canada, Ottawa, 1962)
- "Contemporary Polish Painting and Sculpture", Gres Gallery, Waszyngton, 1961
- "Brzozowski", Gres Gallery, Chicago, 1961
- "Polsk maleri", Frederiksberg Raadhus, Kopenhaga, 14.11.1959-3.12.1959
- "Poolse schilderkunst van nu", Stedelijk Museum, Amsterdam, 2.10.1959-2.11.1959
- "15 Polish painters", Museum of Modern Art, Nowy Jork, 1961 (później Pittsburg, Carnegie Institute, 1961; Minneapolis, Minneapolis Institute of Arts, 1961; St.Louis, Washington University, 1961; Utica Munson, William Proctor Institute, 1962; Montreal, Museum of Fine Arts, 1962; Ottawa, The National Gallery of Canada, 1962)
- "Polsk maleri", Frederiksberg Raadhus, Kopenhaga, 14 listopada - 3 grudnia 1961
- "Poolse schilderkunst, van nu", Stedelijk Museum, Amsterdam, 2 października - 2 listopada 1961

Rysunek 5.1. „*Fraucymer*” Tadeusza Brzozowskiego wraz z prowienicją, zlicytowany za ponad 800 tys. PLN w domu aukcyjnym DESA Unicum.

Źródło: (DESA Unicum, 2015)

Sieć powiązań współczesnego rynku sztuki obraca jedynym, unikatowym zasobem obrazu, pomijając jednak kluczową informację pomiędzy ściśle związanymi jednostkami biznesowymi. Obraz po opuszczeniu artysty, przykładowo: przekazany do domu aukcyjnego, następnie do jego nabywcy, tam kolejno wystawiony w galerii sztuki lub przekazany do dalszej odsprzedaży, całą tę drogę powinien mieć **ujętą w prowienicji**. Niestety ta droga zamiast trafić do odpowiednich **instytucji**, zostaje zagubiona na trasie wymiany pomiędzy tymczasowo zaangażowanymi jednostkami. Pomimo, że rynek sztuki wypracował własne metody weryfikowania prowienicji,



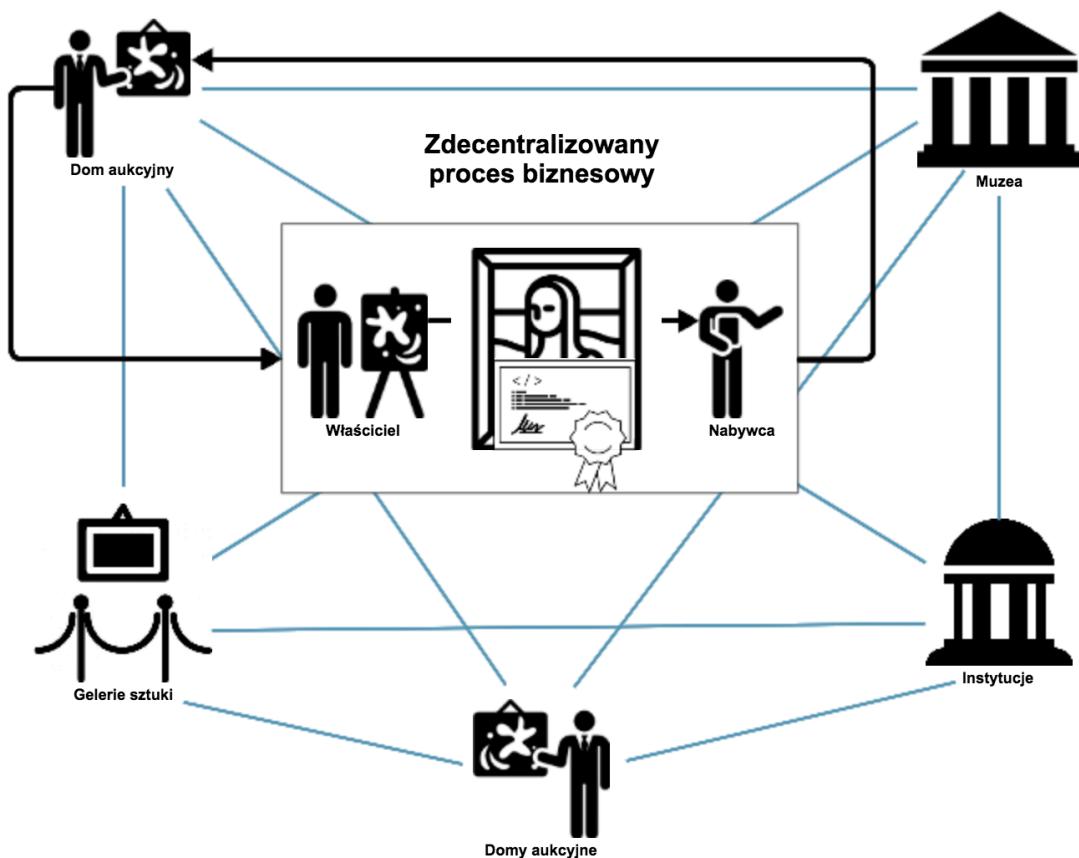
Rysunek 5.2. Zasób biznesowy w sieci powiązań komercyjnego rynku sztuki.
Źródło: opracowanie własne na podstawie (Bailey, 2018; Bartosiewicz, 2016)

według Bailey (2018): „(...) the real problem is that the provenance research and documentation is often disconnected from the art object itself. In that way, future buyers and sellers are often starting from scratch in provenance research rather than building on previous research and documentation” (pol. „(...) prawdziwym problemem jest to, że badania nad proweniencją i dokumentacja są często odłączone od samego przedmiotu sztuki. W ten sposób przyszli kupujący i sprzedający często rozpoczynają od zera badania dotyczące proweniencji, zamiast opierać się na wcześniejszych badaniach i dokumentacji”). Wspomniane, kluczowe biznesowo informacje dzięki rozwiązaniu blockchain mogłyby zostać transparentnie powiązane z unikatowym dziełem, bez możliwości ich niepożądanej zmiany.

Dzięki zastosowaniu blockchainu pomiędzy kontrahentami rynku sztuki: **muzeami, galeriami, instytucjami** oraz przede wszystkim **domami aukcyjnymi**, firmy wystawiające obrazy na aukcje uzyskają możliwość **prostego i bezkosztowego** uzyskania proweniencji, **poprawiając jakość wyceny** dzieł sztuki wystawionych na sprzedaż oraz **ograniczając koszty „ręcznego” badania proweniencji** (poprzez np. czasochłonne wystosowywanie pism do muzeów, bibliotek, fizycznego przeszukiwania archiwów, poszukiwania poprzednich właścicieli itp.), jednocześnie mając możliwość osiągania **wyższych zysków**, wprowadzając **nową wartość dodaną** dla przedsiębiorstw sieci rynku sztuki (rysunek 5.3).

Ponadto, posiadanie wspólnej sieci informacyjnej dot. obrazów ułatwia **ustanawianie ich prawowitego właściciela**, stanowiąc niezbity, niepodrabialny dowód własności zasobu w zdecentralizowanym blockchainie. To znacznie **upłynni obieg dokumentów** związanych z transferami obrazu oraz **przyspieszy przebieg procesów formalnych**. Dodatkowo, każdorazowe uzupełnienie proweniencji obrazu będzie pełniło formę **wersjonowania historii zasobu w czasie**, umożliwiając w niezmiennym blockchainie pełny wgląd w jej zmiany, dokonane przy każdorazowym transferze, przez każdego właściciela, ułatwiając wynajdywanie informacji spreparowanych.

Oprócz zysków instytucji, również artyści odnajdą zalety w zastosowaniu nowatorskiego rozwiązania, mianowicie będąc zapisanymi na samym początku jako jedyni i niepodważalni autorzy dzieła, **wyeliminując problem kradzieży praw autorskich** oraz podszywania się pod czyjasz twórczość.



Rysunek 5.3. Współdzielony proces biznesowy w zdecentralizowanej sieci kontrachentów operujących zasobem kolektywnym.

Źródło: opracowanie własne

W odróżnieniu od standardowego podejścia, zamiast korzystania z usług pośrednika, który agregowałby dane obrazów dla instytucji niezależnie, rozwiązanie oparte na blockchain **decentralizuje informację poprzez strony rynku sztuki** eliminując potrzebę pośrednika, dając **właściwe narzędzie w ręce środowiska artystycznego** działającego dla wspólnego dobrobytu, zamiast centralizując usługę w rękach przedsiębiorstwa, działającego wyłącznie dla zysku.

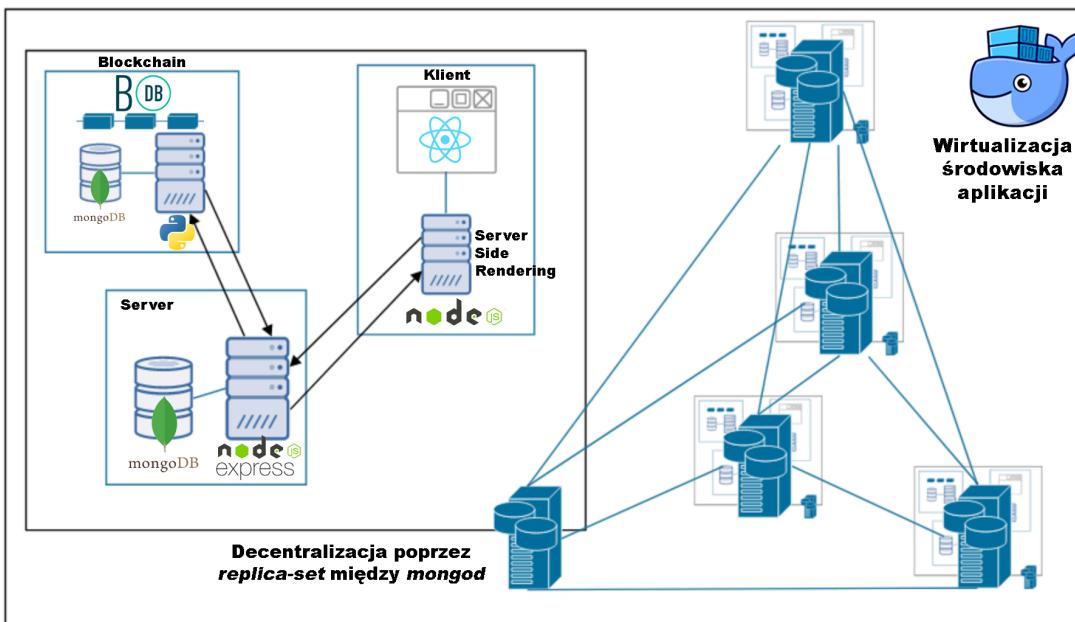
5.4 Architektura blockchain we wdrożeniu

Opracowanie aplikacji dowodu poprawności (ang. Proof of Concept) umożliwia zweryfikowanie teoretycznych założeń odnośnie funkcjonowania i zalet blockchainu w przyjętym scenariuszu zdecentralizowanego rynku sztuki, stanowiąc potwierdzenie jego potencjału biznesowego dla zainteresowanych przedsiębiorstw.

Jednocześnie aplikacja typu PoC jest okrojoną wersją potencjalnej wersji produkcyjnej, opracowana w szybkim tempie z możliwie najniższymi kosztami, wyłącznie dla celów zobrazowania i udowodnienia możliwości realizacji procesu biznesowego. Przedstawiona aplikacja blockchain udowadnia, że możliwym jest zastosowanie takiego rozwiązania w sieci podmiotów gospodarczych oraz ma na celu wykazanie istnienia wartości dodanej dla potencjalnych inwestorów.

Przełożenie biznesowej koncepcji zdecentralizowanego rynku sztuki (rysunek 5.3) na technologiczną architekturę aplikacji (rysunek 5.4) ma w możliwie jak najbliższym stopniu symulować faktyczne istnienie sieci kontrahentów: muzeów, instytucji, galerii i domów aukcyjnych, oraz umożliwić przeprowadzenie wspólnego zarządzania zasobami, np. obrazów. Każdy z węzłów reprezentuje organizację konsorcjum dzieł sztuki (w myśl terminologii BigchainDB), która może niezależnie przeprowadzać proces biznesowy np. wystawienia i sprzedania obrazu na aukcji, a informację o wspólnym zasobie przetwarzać globalnie dzięki blockchain.

Ze względu na ograniczenia zasobów i minimalność aplikacji PoC, w testowanej sieci zdecentralizowanego zaufania zdefiniowano 5 instytucji (np. domów aukcyjnych, galerii, muzeów itd.), jednak nic nie stoi jednak na przeszkodzie, aby sieć składała się z 10, 15, czy 50 jednostek, kosztem zasobów fizycznych oraz (prawdopodobnie)



Rysunek 5.4. Schemat rozproszenia węzłów w sieci z ujęciem architektury aplikacji oraz głównymi technologiami.

Źródło: opracowanie własne

niezależnych serwerów w posiadaniu powiązanych podmiotów. Przyjęta architektura przekłada się w ramach jednego węzła na: 2x instancje bazy danych MongoDB (aplikacja oraz blockchain), 3x instancje aplikacji serwerowej (BigchainDB, aplikacja serwerowa, serwer dla SSR aplikacji klienckiej) w technologiach 1x Python, 2x NodeJS oraz aplikacji klienckiej zbudowanej w technologii React. Stosując tak przyjętą strukturę na potrzeby Proof of Concept, ewaluacja kompletnego rozwiązania wiążałaby się z pięciokrotnością architektury jednego węzła, czyli ponad 10instancjami bazodanowymi i 15 serwerowymi + 5 klienckimi, toteż aby nie nadawać aplikacji poglądowej zbędnego narzutu obliczeniowego, symulację zdecentralizowanego rynku sztuki ograniczono do jednej aplikacji operującej i 5-ciu instancji bazodanowych, powiązanych wspólnie w MongoDB *replica-set* prawidłowo symulujących warunki sieci, z możliwością przełączania między podmiotami w ramach jednej aplikacji operującej. Całość przystosowano do zastosowania w środowisku wirtualizacyjnym Docker, upraszczając symulację sieci w 5 instancjach węzłów wirtualnych na jednej maszynie fizycznej, np. pojedynczym serwerze albo nawet najprostszym, aczkolwiek wydajnym laptopie.

Architektura według rysunku 5.4 umożliwia każdej z 5-ciu instytucji, wdrożenie

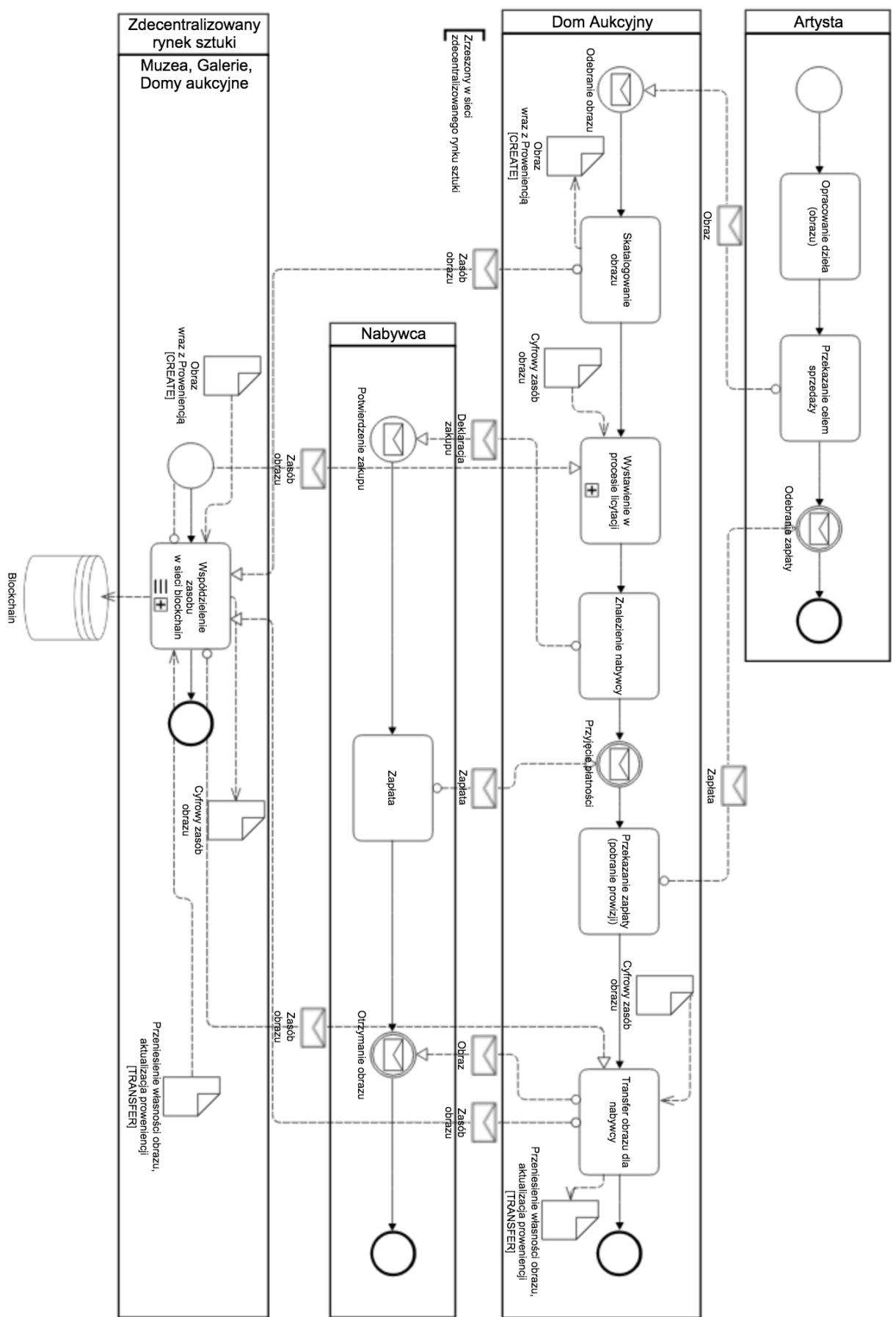
własnej wersji aplikacji blockchain, która będzie pomimo niezależności, zarządzać wspólnie siecią łańcucha bloków zgodnie z regułami, umożliwiając jednocześnie dostosowanie pod własne potrzeby tylko tej części aplikacji, która jest ich wyłączną domeną (pozostawiając blockchain autonomiczną częścią wspólną sieci), mogąc dzięki temu w pełni wykorzystać szanse jakie oferuje zdecentralizowany rynek sztuki dla biznesu społeczności środowiska artystycznego.

5.5 Proces biznesowy

Podstawowym procesem biznesowym domu aukcyjnego jest obrót dziełami sztuki, co upraszczając sprowadza się do przekazania dzieła od obecnego właściciela do nabywcy za wylicytowaną kwotę, która pomniejszona jest o prowizję stanowiącą zysk domu aukcyjnego będącego organizatorem transakcji. Blockchain poszerza istotę wspomnianego procesu sprzedaży o współdzielenie informacji dotyczącej transakcji oraz jej zasobu (rysunek 5.5).

W procesie wyodrębnić można proces utworzenia zasobu oraz jego transferu do nowego właściciela, a także każdorazowo towarzyszące im uzupełnienia metadanych cyfrowego zasobu obrazu o dane proweniencji (np. dane nabywcy, metoda wykonania dzieła, cenę, pośredniczący dom aukcyjny itp.). Informacje cyfrowego zasobu są następnie współdzielone pomiędzy instytucjami zdecentralizowanego rynku sztuki, nadając zalety biznesowe blockchainu dla istniejącego procesu biznesowego, niemalże wcale w niego nie ingerując. Poszerzenie dotychczasowego procesu licytacji obrazu wymaga zmian jedynie w punktach katalogowania oraz przekazania dzieła, stanowiąc minimalny próg wejścia zdecentralizowanego rejestru oraz minimalne koszty wdrożenia i przeszkolenia personelu w kwestii obsługi.

Przedstawiony proces biznesowy jest podstawą koncepcji przechowywania cyfrowej referencji unikatowego dzieła fizycznego oraz jego współdzielenia, a także aktualizacji o najnowsze dane zgodnie z regułami blockchainu. Proces ten zostanie odwzorowany transakcjami typu CREATE oraz TRANSFER, a samo potwierdzenie funkcjonowania, jego ewaluacją w empirycznym procesie działania aplikacji.



Rysunek 5.5. Diagram procesu obrotu dziełem sztuki w BPMN.

Źródło: opracowanie własne

5.6 Implementacja procesu biznesowego

Efektem opracowania aplikacji sieci blockchain jest obszerna baza kodu, której proste ujęcie w formie pisemnej załączonych listingów jest utrudnione szerokim spektrum jej funkcjonowania. Niemniej jednak z rozwiązania wybrano kilka elementów świadczących o kluczowej funkcjonalności, udowadnianej w zaprezentowanym procesie biznesowym. W celu dokładnego zbadania oraz empirycznego sprawdzenia funkcjonowania aplikacji, jej kod źródłowy dostępny jest publicznie na portalu **GitHub**. Symulacja sieci blockchain opiera się na wykorzystaniu narzędzia wirtualizacyjnego **Docker**, w postaci zdefiniowania kontraktu elementów architektury stanowiących integralną całość (listing 5.1).

```
version: '3'

# App
# ...

services:
  mongodb_blockchain:
    image: mongo:3.6
    volumes:
      - dbdata_mongodb_blockchain:/data/db
    ports:
      - 27019:27017
    command: mongod --replSet bigchain-rs --smallfiles --oplogSize 128

  bigchaindb:
    image: bigchaindb/bigchaindb:latest
    ports:
      - 59984:9984
    volumes:
      - data_bigchaindb:/data
    environment:
      - BIGCHAINDB_DATABASE_HOST=mongodb_blockchain
    depends_on:
      - mongodb_blockchain
    stdin_open: true
    tty: true

  # ...

# Network
  mongodb_node_1:
    image: mongo:3.6
    ports:
      - 30001:27017
    command: mongod --replSet bigchain-rs --smallfiles --oplogSize 128

  mongodb_node_2:
    image: mongo:3.6
    ports:
      - 30002:27017
    command: mongod --replSet bigchain-rs --smallfiles --oplogSize 128

  # ...
```

Listing 5.1. Fragmenty *docker-compose.yml* dla aplikacji oraz sieci blockchain. Źródło: opracowanie własne

W praktyce są to dwa odrębne pliki *docker-compose*, które niezależnie uruchamiają: architekturę aplikacyjną oraz sieć węzłów blockchain w MongoDB *replica-set*, za pomocą zdefiniowanej komendy w *Makefile*, ***make start/make start-network***. Utworzone instancje są symulowane wirtualnie na jednej maszynie fizycznej w ramach własnej, niezależnej wewnętrznej sieci, do której dostęp jest możliwy z maszyny-hosta poprzez wystawione porty (np. 27019, 59984, 3001 itd.).

Istotą współdziałania sieci blockchain jest wymiana informacji pomiędzy poszczególnymi węzłami, walidującymi wprowadzane dane do łańcucha bloków. Aby ograniczyć koszty działania aplikacji dowodu poprawności (ang. Proof of Concept) zamiast symulowania środowiska 5-ciu węzłów, węzeł jest tylko jeden, natomiast jest on rekonfigurowany na postać innego węzła za pomocą komendy ***make switch-node-initial/make switch-node-1*** (listing 5.2):

```
# Makefile
switch-node-initial:
    make configure-initial
    make restart-bigchaindb
switch-node-1:
    make configure-node-1
    make restart-bigchaindb
# ...
configure-initial:
    docker cp .bigchaindb blockchain-api_bigchaindb_1:/data/.bigchaindb
# Node-specific unique settings for each BigchainDB standalone instance
# "database": {
#     "host": "mongodb_node_1",
#     "port": 27017,
#     "name": "bigchain",
#     "replicaset": "bigchain-rs",
# },
# "keypair": {
#     "public": "2GNbgmnnCDAExEVsE6NmFU56Uqm5J61h3tCdqw4FjxRe",
#     "private": "9YDizFjdCH4KGiiUSGRAYP9ViQ4kZdM313jFCFki9Sff"
# },
# "keyring": [
#     "BL85xU5ihwkndJ7ZJVrbj4kecfiKqTri2P8vGTTv56HY",
#     "9gb5dSqgy1as4bFwim1YjGu2XbwHPYwVQJUtJnQBkHc",
#     "EYaLLEMl8yTk2ygCWtZYE6gCrDXxpz4WooQYADtQzyU3d",
#     "3BYsicEzcjLJKtLMKTsZFXA94B1e1o1CBnoxhEQJEZnM"
# ],
```

Listing 5.2. Rekonfiguracja węzła BigchainDB z jednej instancji w drugą, wraz z nadpisywanyymi parametrami unikatowymi. Źródło: opracowanie własne

W efekcie każdorazowe przełączenie pomiędzy węzłami powoduje podmianę: instancji MongoDB będącej w kontroli węzła, pary kluczy prywatny/publiczny reprezentujących daną instancję BigchainDB oraz aktualizację tzw. **keyring**, który zawiera klucze publiczne węzłów w sieci z wykluczeniem własnego.

Z praktycznego punktu widzenia stworzenie transakcji w jednym węźle będzie skutkowało dodaniem go do sieci blockchainu *replica-set*, jednak bez natychmiastowej walidacji jego egzystencji w łańcuchu. Wraz z momentem przełączenia się na inny węzeł, pobierze on informację o istniejących blokach i wykryje, że nie wprowadził swojego głosu dla nowopowstałego bloku (można to potraktować jako symulację przestoju sieci, np. timeout z powodu braku połączenia). Węzeł następnie podejmie walidację i odda swój głos tak, jakby to miało miejsce w pełnej, wieloinstancyjnej sieci węzłów BigchainDB.

Aplikacja kliencka obsługuje wykonywanie podstawowych operacji na blockchainie oraz stanowi warstwę reprezentacyjną dla użytkownika odpowiadającą za UX (ang. user experience) całokształtu rozwiązania. Zawiera również podstawowe informacje o projekcie, wytyczne do jego obsługi oraz postanowienia licencyjne. Z poziomu klienta dostępne są strony (listing 5.3).

```
import React from 'react'
import { Switch, Route } from 'react-router-dom'

// ...

const App = () => (
  <div>
    { /* ... */ }

    <Header />

    <main>
      <Switch>
        <Route path="/dashboard" component={DashboardPage} />
        <Route path="/create-transaction" component={CreateTransactionPage} />
        <Route path="/transfer-transaction" component={TransferTransactionPage} />
        <Route path="/asset/:id" component={AssetPage} />

        <Route path="/about" component={AboutPage} />
        <Route path="/terms-of-service" component={TermsPage} />

        <Route exact path="/" component={HomePage} />
        <Route path="*" component={NotFoundPage} />
      </Switch>
    </main>
  </div>
)
```

```

        </main>
        <Footer />
        { /* ... */
    </div>
)
export default App

```

Listing 5.3. Opracowane strony aplikacji klienckiej z poziomu routingu w głównym komponencie *App*. Źródło: opracowanie własne

Część z nich zawiera jedynie informacje (m. in. */about*, */terms-of-service*), podczas gdy inne zawierają implementacje funkcjonalności tworzenia/transferowania zasobu (*/create-transaction*, */transfer-transaction*). Podstawowy podgląd zasobów dostępnych w blockchainie jest możliwy z poziomu */dashboard* z wyszczególnieniem konkretnego zasobu pod */asset/:id*. Kluczowym elementem strony klienta są strony odpowiadające za operowanie zasobem, m.in. formularz tworzenia zasobu dostępny pod linkiem */create-transaction* (listing 5.4).

```

// ...
import { Field, reduxForm } from 'redux-form/immutable'
@reduxForm({
    form: 'createAssetTransaction',
})
export default class CreateTransactionForm extends React.PureComponent<Props> {
    render(): React.Node {
        const { handleSubmit } = this.props
        return (
            <form onSubmit={handleSubmit}>
                <div className="form-group">
                    <label htmlFor="name">Asset Name</label>
                    <Field type="text" className="form-control" name="name" component="input" />
                    <label htmlFor="author">Author</label>
                    <Field type="text" className="form-control" name="author" component="input" />
                </div>
                <div className="form-group">
                    <label htmlFor="type">Asset Category</label>
                    <Field className="form-control" name="type" component="select">
                        <option value="Painting">Painting</option>
                        <option value="Drawing">Drawing</option>
                        <option value="Sculpture">Sculpture</option>
                    </Field>
                    <label htmlFor="technique">Technique</label>
                    <Field type="text" className="form-control" name="technique" component="input" />
                </div>
                <div className="form-group">
                    <label htmlFor="price">Price estimation</label>

```

```

<Field type="text" className="form-control" name="price" component="input" />

36
<label htmlFor="auctionHouse">Auction House</label>
37
<Field type="text" className="form-control" name="auctionHouse" component="input" />
38
</div>
39
<div className="form-group clearfix">
40
  <label htmlFor="preview">
41    <span className="btn btn-primary mb-2 float-left">Upload preview</span>
42  </label>
43
  <Field id="preview" className="form-control" name="preview" component={FileField} />
44
45
  <button type="submit" className="btn btn-primary mb-2 float-right">Submit</button>
46
</div>
47
</form>
48
)
49
}
50
}
51

```

Listing 5.4. Formularz tworzenia zasobu dla sieci blockchain w aplikacji klienckiej.
Źródło: opracowanie własne

Komponent umożliwia wprowadzanie danych, walidację oraz bezpośredni podgląd zasobu, który po wysłaniu zostanie umieszczony w sieci blockchain. Jest on wizualizacją informacji istotnych biznesowo, które będą współdzielone pomiędzy instytucjami wspólnego rynku sztuki, m.in.: tytuł dzieła, autor, zdjęcie, cena rynkowa, metoda wykonania oraz inne, dowolne cechy proweniencji wraz z adnotacjami. Dane po przekazaniu do aplikacji serwerowej są przetwarzane w kontrolerze, np. dla transakcji tworzenia zasobu w *postCreateTransaction* (listing 5.5).

```

// ...

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

/** 
 * Create new asset in blockchain
 *
 * @param req
 * @param res
 * @param next
 */
exports.postCreateTransaction = async (req: express$Request, res: express$Response,
  next: express$NextFunction): mixed => {
  try {
    const { asset, metadata }: CreateTransactionForm = req.body

    // access currently authorized user public/private keys
    const userKeys = await new User().userKeys(USER_1)

    const txCreate = Transaction.makeCreateTransaction(
      asset,
      metadata,
      [
        Transaction.makeOutput( // eslint-disable-line function-paren-newline
          Transaction.makeEd25519Condition(userKeys.publicKey))
      ],
      userKeys.publicKey,
    )
  }
}


```

```

)
const txSigned = Transaction.signTransaction(txCreate, userKeys.privateKey)
await bigchaindb.postTransaction(txSigned)

res.status(204).send() // successfully proceeded
} catch (e) {
next(e)
}
}

```

Listing 5.5. Transakcja tworzenia zasobu umieszczonego w sieci blockchain po stronie aplikacji serwerowej. Źródło: opracowanie własne

Metoda *Transaction.makeCreateTransaction* przyjmuje informacje nadane w formularzu i tworzy z nich odpowiednią transakcję typu CREATE, która jest podpisywana kluczem prywatnym właściciela fizycznego zasobu i rozpropagowywana po sieci blockchain wraz z kluczem publicznym oraz sygnaturą podpisu do procesu walidacji.

Osobną kwestią pozostają klucze wykorzystywane w transakcji, które stanowią metodę weryfikacji tożsamości danej osoby fizycznej posiadającej zasób. Wyłącznie na potrzeby aplikacji dowodu poprawności są one przechowywane w formie stałych, wystarczających do udowodnienia funkcjonalności. Natomiast w świecie rzeczywistym prawdopodobnie zostałyby powiązane z istniejącymi kontami aplikacji poza blockchainem i zabezpieczone odpowiednim hasłem (ang. passphrase) klucza prywatnego, znany jedynie użytkownikowi albo byłby przechowywane poza aplikacją w dodatkowo opracowanej aplikacji klienckiej zawierającej klucze typu „portfel”, tak jak ma to miejsce w przypadku kryptowalut. Zakres wykorzystania i zabezpieczenia kluczy asymetrycznych stanowi zagadnienie bezpieczeństwa, a decyzję powinien podjąć doświadczony zespół ekspertów.

Analogicznie tworzona jest transakcja TRANSFER, z tą różnicą, że dotyczy ona istniejącego zasobu, rozpoznawanego za pomocą identyfikatora istniejącej transakcji w sieci, a w jej przebiegu wskazywany jest klucz publiczny nowego właściciela. Po podpisaniu kluczem prywatnym poprzednika, traci on do niego prawo, a nowy posiadacz ma od tej pory pełną kontrolę nad cyfrowym zasobem (listing 5.6).

```

/**
 * Transfer the asset in blockchain from current owner to another
 *
 * @param req
 * @param res
 * @param next
 */
exports.postTransferTransaction = async (req: express$Request, res: express$Response

```

```

    , next: express$NextFunction): mixed => {
try {
  const { txCreatedId, metadata }: TransferTransactionForm = req.body
  // access currently authorized users public/private keys
  const currentOwnerKeys = await new User().userKeys(USER_1)
  const newOwnerKeys = await new User().userKeys(USER_2)

  const txCreated = await bigchaindb.getTransaction(txCreatedId)

  const createTransfer = Transaction.makeTransferTransaction(
    [{ txCreated, output_index: 0 }],
    [
      Transaction.makeOutput( // eslint-disable-line function-paren-newline
        Transaction.makeEd25519Condition(currentOwnerKeys.publicKey))
    ],
    currentOwnerKeys.publicKey,
    metadata,
  )

  // Sign with the key of the new owner of the painting
  const signedTransfer = Transaction
    .signTransaction(createTransfer, newOwnerKeys.privateKey)
  await bigchaindb.postTransaction(signedTransfer)

  res.status(204).send() // successfully proceeded
} catch (e) {
  next(e)
}
}

```

Listing 5.6. Fragmenty transakcji transferu zasobu od pierwotnego właściciela do wtórnego w aplikacji serwerowej. Źródło: opracowanie własne

Efektem wykonania procesu biznesowego jest powstanie prawidłowego bloku transakcji wraz z zasobami i metadanymi (listing 5.7).

```

{
  "id": "3848a91f24e174b774ecea61bb05846e89da967d15fe51e7704ab5f677f120ed",
  "block": {
    "timestamp": "1526385602",
    "transactions": [
      {
        "inputs": [
          { ... }
        ],
        "outputs": [
          {
            "public_keys": [
              "GfePXmrRWCeYfghDa34vtpjKwTHsQSFka6Sn9nKiEc7b"
            ],
            "condition": { ... },
          }
        ],
        "operation": "CREATE",
        "version": "1.0",
        "id": "12a80b779d7601dee583c42bb343e49052618eafbaa100538fc0d08132b5adf5"
      }
    ],
    "node_pubkey": "BL85xU5ihwkndJ7ZJVrbj4kecfiKqTri2P8vGTTv56HY",
    "voters": [
      "EYaleML8yTk2ygCWtZYE6gCrDXXpz4WooQYADtQzyU3d",
      "9gb5dSQgty1as4bFwim1YjGu2XbwHPYwVQJUtJnQBkHc",
      "2GNbgmnnCDAExEVsE6NmFU56Uqm5J61h3tCdqw4FjxRe",
    ]
  }
}

```

”3BYsicEzcjLJKtLMKTsZFXA94B1e1o1CBnoxhEQJEZnM”,	28
”BL85xU5ihwkndJ7ZJVrbj4kecfiKqTri2P8vGTTv56HY”	29
]	30
},	31
”signature”: ”4yv7RvoyiKtkKd7HJrPmwRZXgbhRSwGepqU7g7MgXZaRGQ1GzAPxiuJHfM(…)”	32
}	33

Listing 5.7. Fragmenty bloku transakcji umieszczonego w sieci blockchain. Źródło: opracowanie własne

Blok ten, umieszczony i zweryfikowany w sieci blockchain, jest kolekcją transakcji reprezentujących cyfrowe zasoby biznesowe, współzielone i dostępne w ramach zdecentralizowanego rynku sztuki.

5.7 Ewaluacja procesu na przykładzie przypadku użycia

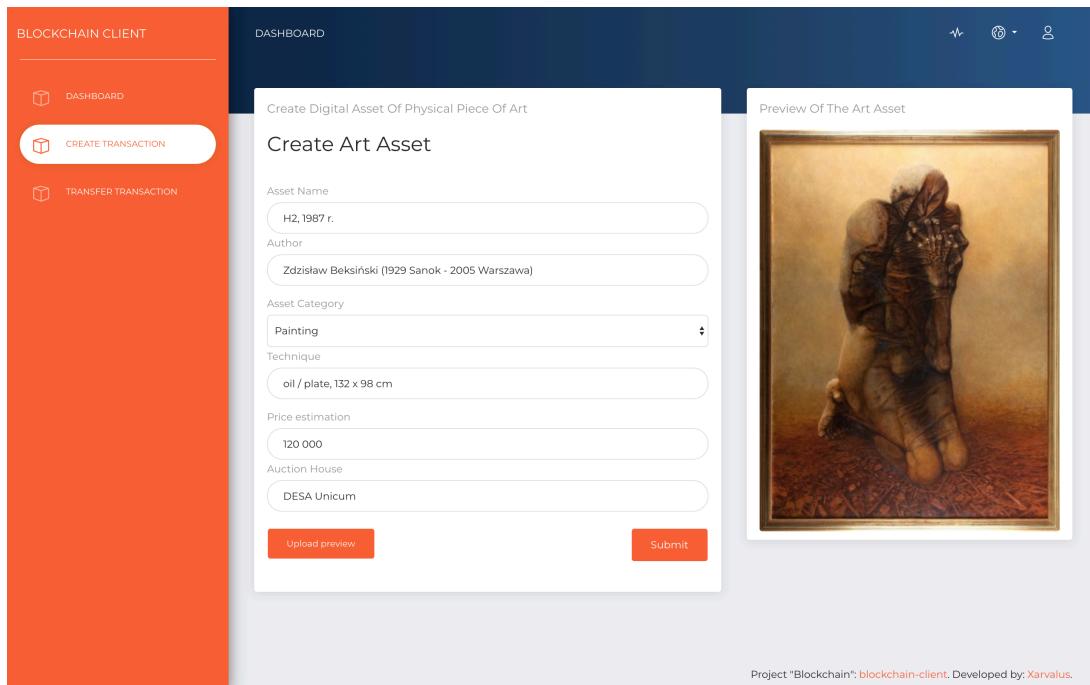
Zgodnie z przedstawionym procesem biznesowym (oraz jego implementacją), przyjętym przypadkiem użycia jest scenariusz skatalogowania i przekazania obrazu, opisany następującymi krokami:

- stworzenie nowego rekordu obrazu (cyfrowego zasobu),
- zweryfikowanie dostępu poprzez odrębne węzły (instytucje),
- transfer zasobu od właściciela pierwotnego do wtórnego,
- uzyskanie wyniku pozytywnej walidacji bloku (głosowania).

Powodzenie realizacji procesu w opracowanym przez autora narzędziu, udowadnia możliwość wdrożenia oraz funkcjonowania blockchainu dla rynku sztuki wraz z proponowaną wartością dodaną istniejącą przy opracowanym rozwiążaniu.

Po uzyskaniu obrazu dla celu licytacji dom aukcyjny przeprowadza skatalogowanie dzieła. Odpowiednio przygotowane informacje np. autor, estymacje cenowe oraz proweniencja zostają wpisane w **formularz tworzenia zasobu** (rysunek 5.6), który zostaje wysłany do aplikacji serwerowej walidującej poprawność danych i warunków kryptograficznych, a w efekcie podpisującej transakcję kluczem prywatnym właściciela.

Sukces zakończonej operacji powoduje utworzenie cyfrowego zasobu obrazu, powiązanego z transakcją CREATE w nowo dodanym bloku sieci blockchain (rysunek 5.7). Cyfrowy zasób obrazu jest dostępny z dowolnej instytucji zdecentralizowanego rynku

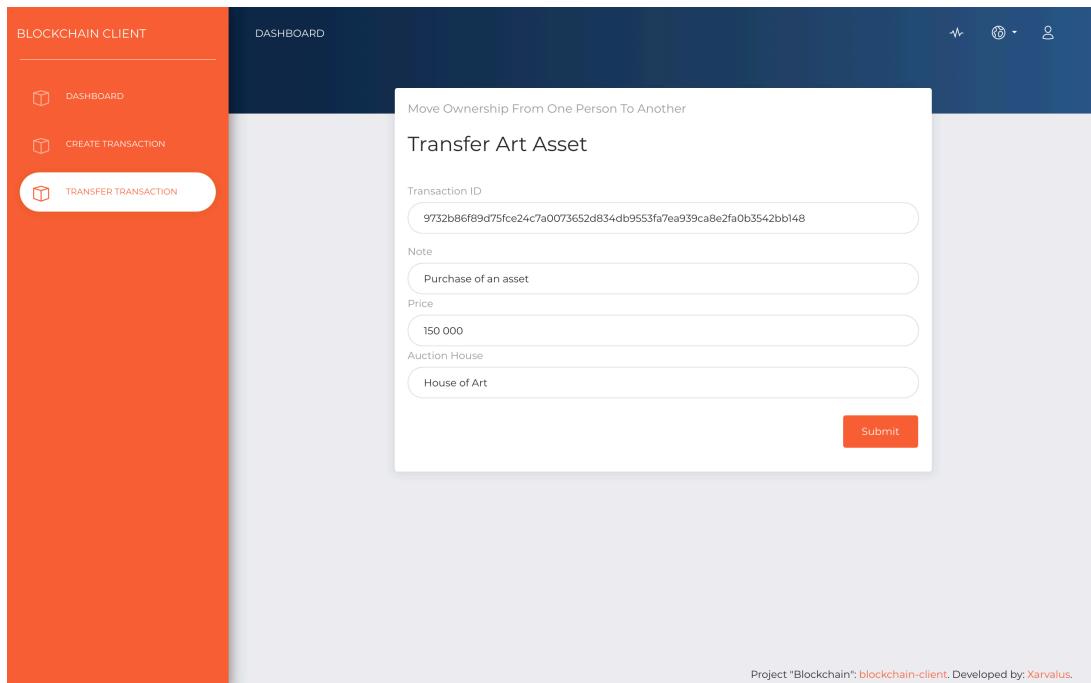


Rysunek 5.6. Formularz katalogowania obrazu wraz z danymi zasobu.
 Źródło: zrzut ekranu, dane obrazu: (DESA Unicum, 2018)

Rysunek 5.7. Stworzony blok łańcucha w efekcie procesu tworzenia zasobu obrazu.
 Źródło: zrzut ekranu

sztuki, wraz ze wszelkimi danymi szczegółowymi.

Po zakończeniu procesu sprzedaży, istniejący zasób sztuki jest przekazywany od istniejącego właściciela do nowego nabywcy. Dom aukcyjny w tym celu wykorzystuje **formularz transferu zasobu** (rysunek 5.8) aktualizując dane skatalogowanego obrazu.



Rysunek 5.8. Przeniesienie własności obrazu oraz aktualizacja danych w formularzu.

Źródło: zrzut ekranu

Od momentu stworzenia bloku, transakcja wprowadzana do łańcucha aktualizuje w sieci blockchain dane zasobu, które po zmianie wskazują na nowego nabywcę obrazu. Blok wymaga ponadto walidacji w ramach sieci, której wynik ostatecznie potwierdza, czy jest on prawidłowy, a transakcje w nim zawarte spełniają warunki (rysunek 5.9).

Uzyskany wynik głosowania węzłów (instytucji) potwierdza poprawność bloku. Proces biznesowy z udziałem blockchainu został przeprowadzony z sukcesem, udowadniając praktyczne funkcjonowanie sieci zdecentralizowanego rynku sztuki.

The screenshot shows the MongoDB Compass interface for the database `blockchain_bigchaindb`. The left sidebar lists databases and collections: `admin`, `bigchain` (which is expanded to show `assets`, `backlog`, `bigchain`, `metadata`, and `votes`), and `config`. The right panel is focused on the `bigchain.votes` collection, which is currently selected (indicated by a blue bar). The interface includes a toolbar with `Find`, `Insert`, `Update`, `Remove`, `Index`, and `Aggregation` buttons. Below the toolbar, a query builder shows the command `db.votes.find().sort({_id: 1}).skip(0).limit(30)`. The results table displays three documents with the following fields:

<code>_id</code>	<code>Objectid("5b08f0abe90f5b0042df5281")</code>	<code>Object id</code>
<code>_id</code>	<code>Objectid("5b08f2c7e90f5b0041516dcc")</code>	<code>Object id</code>
<code>node_pubkey</code>	<code>BL85xU5ihwkndJ7ZJVrbj4kecfiKqTri2P8vGTTv56HY</code>	<code>String</code>
<code>signature</code>	<code>4uPs7AiCzuvNbwmKnQhFfnqMUZFkgqViWydrdEGb...</code>	<code>String</code>
<code>vote</code>	<code>8df6d22a3bc385fd20ac0e5f8b451625bc73947e02f5...</code>	<code>Object, 5 items</code>
<code>voting_for_block</code>	<code>83ea54c44d4ce8881019e10dbf0c07e3ba78a8f86bf5...</code>	<code>String</code>
<code>previous_block</code>	<code>83ea54c44d4ce8881019e10dbf0c07e3ba78a8f86bf5...</code>	<code>String</code>
<code>is_block_valid</code>	<code>true</code>	<code>Boolean</code>
<code>invalid_reason</code>	<code>NULL</code>	<code>NULL</code>
<code>timestamp</code>	<code>1527313095</code>	<code>String</code>
<code>_id</code>	<code>Objectid("5b08f768e90f5b0042afc449")</code>	<code>Object id</code>
<code>_id</code>	<code>Objectid("5b08f768e90f5b0042afc449")</code>	<code>Object id</code>
<code>node_pubkey</code>	<code>2GNbgmnnCDExEVsE6NmFU56Uqm5J61h3tCdqw4FjxRe</code>	<code>String</code>
<code>signature</code>	<code>4A6Q21nxLBfuLoEotCsngxB4cff1W1GoI7IWuD4xbgM...</code>	<code>String</code>
<code>vote</code>	<code>8df6d22a3bc385fd20ac0e5f8b451625bc73947e02f5...</code>	<code>Object, 5 items</code>
<code>voting_for_block</code>	<code>83ea54c44d4ce8881019e10dbf0c07e3ba78a8f86bf5...</code>	<code>String</code>
<code>previous_block</code>	<code>83ea54c44d4ce8881019e10dbf0c07e3ba78a8f86bf5...</code>	<code>String</code>
<code>is_block_valid</code>	<code>true</code>	<code>Boolean</code>
<code>invalid_reason</code>	<code>NULL</code>	<code>NULL</code>
<code>timestamp</code>	<code>1527314281</code>	<code>String</code>

Total Results: 3 (0.00s)

Rysunek 5.9. Pozytywny wynik głosowania poprawności bloku zawierającym przeprowadzane transakcje.

Źródło: zrzut ekranu

5.8 Empiryczna próba falsyfikacji danych sieci

Proces biznesowy sieci blockchain może istnieć jedynie, gdy jest on **niepodatny na ataki**, czyli bezpieczny na dowolne próby zakłócenia funkcjonowania, kradzieży danych, czy falsyfikacji bloków itp.

Dowolna instytucja zdecentralizowanego rynku sztuki, mogłaby niepostrzeżenie ręcznie wprowadzić nowy blok bezpośrednio do bazy MongoDB, który zostałby rozprowadzony po całej sieci w ramach *replica-set* (rysunek 5.10) bez sprawdzenia warunków kryptograficznych. Sztucznie dodany blok transakcji może zawierać dowolne informacje, również przekazania własności obrazu, do którego dana instytucja nie posiada prawa, dopuszczając się więc **kradzieży zasobu**.

The screenshot shows the MongoDB Compass interface connected to a database named 'blockchain_bigchaindb'. On the left, the database structure is visible with collections: admin, bigchain (which is expanded to show assets, backlog, bigchain, metadata, and votes), config, and others. The 'bigchain' collection has a count of 5 documents. The right side of the interface shows the 'bigchain.bigchain Stats' tab selected. A query result table displays a single document from the 'bigchain' collection. The document structure is as follows:

	Value	Type
<code>_id</code>	ObjectId("5b08f0abe90f5b0077ef89b5")	Object id
<code>_id</code>	ObjectId("5b08f2c6e90f5b0019516dca")	Object id
<code>_id</code>	ObjectId("5b08fd5d620b7215ba29bd81")	Object id
<code>_id</code>	ObjectId("5b08fd5d620b7215ba29bd81")	Object id
<code>id</code>	8df6d22a3bc385fd20ac0e5f8b451625bc73947e02f5...	String
<code>block</code>		Object, 4
<code>timestamp</code>	1527313099	String
<code>transactions</code>		Array, 1
<code> 0</code>		Object, 5
<code> inputs</code>		Array, 1
<code> outputs</code>		Array, 1
<code> 0</code>		Object, 3
<code> public_keys</code>		Array, 1
<code> 0</code>	GW1ndZm4mbVC8ePeiGWz6DqHexqewqy5teURVHi3RG4	String
<code> condition</code>		Object, 2
<code> amount</code>	1	String
<code> operation</code>	CREATE	String
<code> version</code>	1.0	String
<code> id</code>	9732b86f89d75fce24c7a0073652d834db9553fa7ea...	String
<code> node_pubkey</code>	BL85xU5ihwkndJ7ZJvrbj4kecfiKqTrI2P8vGTTv56HY	String
<code> voters</code>		Array, 5
<code> signature</code>	4WE3JwX3jfTXCd1N7F5Arb7LwKtJa6g2RwQkWQ668S...	String

Total Results: 3 (0.00s)

Rysunek 5.10. Spreparowany blok transakcji, sztucznie wprowadzony do sieci blockchain w celu dokonania kradzieży.

Źródło: zrzut ekranu, opracowanie własne

Mechanizm głosowania sieci blockchain BigchainDB po przekazaniu tak spreparowanego bloku wywołuje proces weryfikacji, bezpośrednio dla każdego węzła (instytucji) (rysunek 5.11) w efekcie demaskując próbę ataku i unieważniając

dodany blok transakcji, a tym samym **udaremniając próbę kradzieży**.

_id	ObjectID("5b08f2c7e90f5b0041516dcc")	Object id
_id	ObjectID("5b08f768e90f5b0042afc449")	Object id
_id	ObjectID("5b08fd63e90f5b004131d154")	Object id
_id	ObjectID("5b08fd63e90f5b004131d154")	Object id
node_pubkey	BL85xU5ihwkndJ7ZJrbj4kecfiKqTri2P8vGTTv56HY	String
signature	3SuD742wwD32kn19rFD8BGoqgAvBov3MPA1u1V4Ydp...	String
vote	{...}	Object, 5 items
voting_for_block	8df6d22a3bc385fd20ac0e5f8b451625bc73947e02f5...	String
previous_block	8df6d22a3bc385fd20ac0e5f8b451625bc73947e02f5...	String
is_block_valid	false	Boolean
invalid_reason	NULL	NULL
timestamp	1527315812	String
_id	ObjectID("5b08fd3e90f5b0041643024")	Object id
_id	ObjectID("5b08fd3e90f5b0041643024")	Object id
node_pubkey	2GNbgmnnCDAExEvS6NmFu56Uqm5J61h3tCdqw4FjxRe	String
signature	5a1LTYijUZYNaZ1scuNenMNBUxU26sdXXDYuyFanYPE...	String
vote	{...}	Object, 5 items
voting_for_block	8df6d22a3bc385fd20ac0e5f8b451625bc73947e02f5...	String
previous_block	8df6d22a3bc385fd20ac0e5f8b451625bc73947e02f5...	String
is_block_valid	false	Boolean
invalid_reason	NULL	NULL
timestamp	1527315923	String

Rysunek 5.11. Negatywny wynik głosowania poprawności dla spreparowanego bloku.

Źródło: zrzut ekranu

5.9 Efekt końcowy oraz konkluzja wdrożenia

Przeprowadzone badania dowiodły możliwości realnego funkcjonowania blockchainu w przyjętej (oraz istniejącej) sieci kontrahentów rynku sztuki. Z proponowanym rozwiązańiem wiążą się istotne wartości dodane, które po wdrożeniu przełożą się na zyski instytucji, nie tylko finansowe, ale również niematerialne.

W aplikacji oraz przy jej opracowaniu skupiono się na ujęciu odpowiednio wysokiego poziomu szczegółowości tak, aby analiza kontekstu wdrożenia rozwijała wątpliwości podczas badania. Ujęte punkty krytyczne oraz pogłębione opisy służą wykazaniu, iż pomimo pozornych komplikacji, **technologia blockchain ma możliwość z powodzeniem funkcjonować w rzeczywistości biznesowej**.

Analiza przypadku naruszenia integralności danych w sieci wykazała precyzję i

niemal natychmiastową reakcję sieci blockchainu na próbę falsyfikacji, będąc dowodem gwarancji bezpieczeństwa danych biznesowych na pospolitą próbę ataku.

Kwestią złożoną pozostaje podjęcie wielu trudnych decyzji architektonicznych, a dotyczących bezpieczeństwa, rozwoju projektu w instytucjach oraz jego utrzymania i bezawaryjnego funkcjonowania przy ryzyku ograniczonym do minimum. Decyzje te powinny zostać podjęte w środowisku doświadczonych ekspertów przy wdrożeniu produkcyjnym sieci zdecentralizowanego zaufania.

Długi okres praktyki w technologiach blockchain ostatecznie pokaże czy podjęcie prób decentralizacji procesów z wykorzystaniem łańcucha bloków okaże się rewolucyjną innowacją czy kosztowną porażką. Niemniej w blockchain istnieje potencjał, który spożytkować może wiele współczesnych biznesów.

5.10 Podsumowanie

W rozdziale przeanalizowano praktyczny przypadek problemu biznesowego i wdrożono przykładową implementację dowodu poprawności, której wynik ewaluacji udowodnił skuteczne działanie blockchain w rzeczywistości biznesowej dla rozwiązania konkretnego problemu.

Wraz z funkcjonowaniem sieci zdecentralizowanego zaufania potwierdzono możliwość wykorzystania wartości dodanej, jaką oferuje dla podmiotów gospodarczych technologia blockchain, podjęto jej analizę pod kątem bezpieczeństwa oraz elementów krytycznych wdrożenia produkcyjnego.

Zwrócono również uwagę na kwestię rozwoju rozwiązania blockchain w dłuższym okresie, ponieważ postęp w dziedzinie nastąpić może jedynie, gdy pierwsze biznesy zaryzykują adaptację i na bazie błędów pionierskich wypracują sprawdzone i stabilne pochodne technologie blockchain, gotowe do masowego wdrażania w sieciach przedsiębiorstw.

Rozdział 6

Podsumowanie

Celem pracy była implementacja technologii blockchain w aplikacji dowodu poprawności (ang. proof of concept) dla wdrożenia w biznesowej sieci kontrahentów celem wypracowania nowej wartości dodanej. Cel ten został zrealizowany za pomocą następujących celów szczegółowych:

1. Przedstawienie oraz omówienie idei technologii blockchain, jej zasad funkcjonowania, wraz z oferowaną wartością dodaną dla przedsiębiorstw.
2. Omówienie architektury oprogramowania z propozycją technologii wartościowych w zastosowaniu implementacji blockchain.
3. Analiza wybranego rozwiązania blockchain, dostępnego do wykorzystania w procesie wdrażania dla podmiotów gospodarczych.
4. Wyekstrahowanie procesu biznesowego licytacji obrazu z komercyjnego rynku sztuki oraz opracowanie aplikacji blockchain ewaluującej proces z nową wartością dodaną.
5. Walidacja funkcjonowania zdecentralizowanej sieci na podstawie próby ataku.

Rozdział drugi zrealizował pierwszy cel szczegółowy pracy, omawiając istniejące rozwiązania blockchainowe funkcjonujące produkcyjnie dla rozwijającej się technologii. Ujęto w nim uzasadnioną krytykę rozwiązań oraz nakreślono potencjał rozwoju wraz z omówieniem potencjalnych zagrożeń.

W rozdziale trzecim omówiono koncepcję architektury sieci blockchain, spełniając drugi cel szczegółowy pracy, ze skupieniem się na uzyskaniu najlepszego rezultatu z punktu widzenia technologii oraz biznesu, obierając full-stack JavaScript jako technologię wiodącą aplikacji. Wybór ten został szczegółowo umotywowany w rozdziale.

Kolejny cel szczegółowy został zrealizowany w rozdziale czwartym. Wybrano w nim narzędzie BigchainDB do zaimplementowania technologii blockchain dla biznesowej sieci kontrahentów. Omówiono zasadę działania rozwiązania, replikację z wykorzystaniem MongoDB replica-set oraz podstawowe typy transakcji CREATE-/TRANSFER wraz z metodą walidacji, czyli głosowaniem węzłów na poprawność bloku (ang. voting).

Ostatnie cele szczegółowe, czwarty oraz piąty, zostały zrealizowane w ramach rozdziału piątego pracy. Przeprowadzono w nim analizę problemu biznesowego oraz z powodzeniem zaimplementowano go na blockchainie w aplikacji dowodu poprawności (zgodnie z zaprojektowaną architekturą). Sprawdzono funkcjonowanie w procesie ewaluacji oraz podjęto próbę ataku w postaci falsyfikacji danych bloku w sieci, któremu rozwiązanie BigchainDB się oparło dzięki systemowi głosowania.

Dzięki realizacji celów szczegółowych, z sukcesem zrealizowano cel główny pracy, a mianowicie zaimplementowano aplikację blockchain, która udowadnia, że z powodzeniem można wdrożyć tę technologię w istniejącej sieci kontrahentów zdecentralizowanego rynku sztuki oraz osiągnąć nową wartość dodaną, która wynika z zastosowania zunifikowanej sieci instytucji oraz śledzenia unikalnych zasobów sztuki (kwestię tę szeroko omówiono w rozdziale piątym).

Rozwiązanie ma istotną szansę funkcjonować, jeśli w procesie dalszych prac nad technologią, biznesy zaryzykują adaptację i na bazie błędów pionierskich wypracują sprawdzone i stabilne technologie blockchain, gotowe do masowego wdrażania w sieciach przedsiębiorstw. Ostatecznie to praktyka biznesowa potwierdzi, czy podjęcie prób decentralizacji procesów z wykorzystaniem łańcucha bloków okaże się rewolucyjną innowacją czy kosztowną porażką. Niemniej w blockchain istnieje potencjał, który spożytkować może wiele współczesnych biznesów.

Opracowana w procesie badawczym aplikacja blockchain stanowi dowód na potencjał oraz potwierdzone działanie technologii zdecentralizowanego zaufania w biz-

nesie. Do celów dalszych badań kod źródłowy aplikacji jest upublicznyony na portalu Github:

- aplikacja kliencka „*blockchain-client*”¹,
- aplikacja serwerowa „*blockchain-api*”².

Kody te sa dostępne na otwartej licencji MIT dla każdego zainteresowanego wynikami empirycznej ewaluacji procesu. Udostępniono także wyniki pracy badawczej do celów porównawczych „*blockchain-thesis*”³.

¹<https://github.com/Xarvalus/blockchain-client>

²<https://github.com/Xarvalus/blockchain-api>

³<https://github.com/Xarvalus/blockchain-thesis>

Bibliografia

- Bailey, J. (2018). Why Use Blockchain Provenance For Art? Dostęp from <https://www.artname.com/news/2018/1/26/why-use-blockchain-provenance-for-art>
- Bartosiewicz, A. (2016). Handel dziełami sztuki w Polsce. Dostęp from http://superbiz.se.pl/opinie-biz/handel-dzielami-sztuki-w-polsce_894948.html
- BigchainDB GmbH. (2018a). BigchainDB Server Documentation. Dostęp from <https://media.readthedocs.org/pdf/bigchaindb-server/latest/bigchaindb-server.pdf>
- BigchainDB GmbH. (2018b). Key concepts of BigchainDB. Dostęp from <https://www.bigchaindb.com/guides/key-concepts-of-bigchaindb/>
- BigchainDB GmbH. (2018c). Meet BigchainDB. The blockchain database. Dostęp from <https://www.bigchaindb.com/>
- BigchainDB GmbH. (2018d). Terminology. Dostęp from <https://docs.bigchaindb.com/en/latest/terminology.html>
- Bitcoin.org. (2018). How does Bitcoin work? Dostęp from <https://bitcoin.org/en/how-it-works>
- Bitcoin.org GitHub & GitHub Community. (2018). Bitcoin Core integration/staging tree. Dostęp from <https://github.com/bitcoin/bitcoin>
- Buchko, S. (2017). How long do Bitcoin transactions take? Dostęp from <https://coincentral.com/how-long-do-bitcoin-transfers-take/>
- CBINSIGHTS. (2017). What Is Blockchain Technology? Dostęp from <https://www.cbinsights.com/research/what-is-blockchain-technology/>
- CBINSIGHTS. (2018). Banking Is Only The Beginning: 36 Big Industries Blockchain Could Transform. Dostęp from <https://www.cbinsights.com/research/industries-disrupted-blockchain/>

- CoinMarketCap. (2018). Top 100 Cryptocurrencies by Market Capitalization. Dostęp from <https://coinmarketcap.com/>
- D'Aliessi, M. (2016). How Does the Blockchain Work? Dostęp from <https://medium.com/@micheledaliessi/how-does-the-blockchain-work-98c8cd01d2ae>
- DESA Unicum. (2015). "Fraucymer", 1959 r. Dostęp from <https://desa.pl/pl/auctions/265/object/30542/tadeusz-brzozowski-fraucymer-1959-r>
- DESA Unicum. (2018). H2, 1987 r. Dostęp from <https://desa.pl/pl/exhibitions/25/object/1217/zdzislaw-beksinski-h2-1987-r>
- Docker Inc. (2018). Docker - Build, Ship, and Run Any App, Anywhere. Dostęp from <https://www.docker.com/>
- Emerging Technology. (2017). Quantum Computers Pose Imminent Threat to Bitcoin Security. Dostęp from <https://www.technologyreview.com/s/609408/quantum-computers-pose-imminent-threat-to-bitcoin-security/>
- Ethereum Foundation. (2018). Ethereum Project. Dostęp from <https://www.ethereum.org/>
- Facebook Open Source. (2018a). Delightful JavaScript Testing. Dostęp from <https://facebook.github.io/jest/>
- Facebook Open Source. (2018b). Introduction to GraphQL. Dostęp from <http://graphql.org/learn/>
- Facebook Open Source. (2018c). Introduction to type checking with Flow. Dostęp from <https://flow.org/en/docs/getting-started/>
- Facebook Open Source. (2018d). React - A JavaScript library for building user interfaces. Dostęp from <https://reactjs.org/>
- Garner, B. (2017). What is Bitcoin Cash? Dostęp from <https://coincentral.com/what-is-bitcoin-cash-vs-bitcoin/>
- Iansiti, M. & Lakhani, K. R. (2017). The Truth About Blockchain. Dostęp from <https://hbr.org/2017/01/the-truth-about-blockchain>
- Ishizaki, T. (2018). Ethereum Introduction. Dostęp from <https://github.com/ethereum/wiki/wiki/Ethereum - introduction#proof-of-work--proof-of-stake--other-proving-methods>

- Kadivar, N. (2018). How Netflix and Paypal did product transformation using Node.js? Dostęp from <https://www.linkedin.com/pulse/how-netflix-paypal-did-product-transformation-using-nodejs-patel/>
- Lam, E. (2017). What the World's Central Banks Are Saying About Bitcoin. Dostęp from <https://www.bloomberg.com/news/articles/2017-12-15/what-the-world-s-central-banks-are-saying-about-cryptocurrencies>
- LearnBoost. (2018). elegant mongodb object modeling for node.js. Dostęp from <http://mongoosejs.com/docs/guide.html>
- Lisk Foundation. (2018). Lisk - Blockchain Application Platform. Dostęp from <https://lisk.io/>
- Mearian, L. (2018). What is blockchain? The most disruptive tech in decades. Dostęp from <https://www.computerworld.com/article/3191077/security/what-is-blockchain-the-most-disruptive-tech-in-decades.html>
- MongoDB, Inc. (2018a). MongoDB for GIANT Ideas. Dostęp from <https://www.mongodb.com/>
- MongoDB, Inc. (2018b). Replication. Dostęp from <https://docs.mongodb.com/manual/replication/>
- Murch, Satoshi & Community ☽. (2013). Can someone explain how the Bitcoin Blockchain works? Dostęp from <https://bitcoin.stackexchange.com/questions/12427/can-someone-explain-how-the-bitcoin-blockchain-works>
- nadzw. dr hab. Sylwia Sysko-Romańczuk, P., Roszkowska, D. P. & Niedźwiecka, A. (2012). Odpowiedzialność biznesu. Teoria i praktyka. Dostęp from http://www.kozminski.edu.pl/fileadmin/wspolne_elementy/Jednostki/Czasopismo_MBA/Sysko_Roszkowska_Niedzwiecka_mba_2012_02_026_1_01.pdf
- Narayanan, A., Bonneau, J., Felten, E., Miller, A. & Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton: Princeton University Press.
- Nath, D. S. (2017). 4 important points to know about Progressive Web Apps (PWA). Dostęp from <https://medium.com/@deepusnath/4-points-to-keep-in-mind-before-introducing-progressive-web-apps-pwa-to-your-team-8dc66bcf6011>

- Node.js Foundation. (2018). Fast, unopinionated, minimalist web framework for Node.js. Dostęp from <https://expressjs.com/>
- Norry, A. (2017). An In-depth Look at Bitcoin Laws & Future Regulation. Dostęp from <https://blockonomi.com/bitcoin-regulation/>
- Ramirez, C. (2017). Deploy Nodejs microservices to a Docker Swarm Cluster [Docker from zero to hero]. Dostęp from <https://towardsdatascience.com/deploy-a-nodejs-microservices-to-a-docker-swarm-cluster-docker-from-zero-to-hero-464fa1369ea0>
- Redux. (2018). Redux is a predictable state container for JavaScript apps. Dostęp from <https://redux.js.org/>
- Redux Saga. (2018). redux-saga is a library that aims to make application side effects easier. Dostęp from <https://redux-saga.js.org/>
- Schwartz, D., ODell, N. & Community ☽. (2013). How do Bitcoin clients find each other? Dostęp from <https://bitcoin.stackexchange.com/questions/3536/how-do-bitcoin-clients-find-each-other>
- Sherry, B. (2018). What is the Genesis Block in Bitcoin Terms? Dostęp from <https://www.investopedia.com/news/what-genesis-block-bitcoin-terms/>
- Socket.IO. (2018). Socket.IO enables real-time bidirectional event-based communication. Dostęp from <https://socket.io/>
- Stallings, W. (1999). *Cryptography and Network Security: Principles and Practice*. Prentice Hall.
- Statista. (2018). Size of the Bitcoin blockchain from 2010 to 2017, by quarter (in megabytes). Dostęp from <https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/>
- Styled Components. (2018). Visual primitives for the component age. Dostęp from <https://www.styled-components.com/>
- The Economist. (2015). The great chain of being sure about things. Dostęp from <https://www.economist.com/news/briefing/21677228-technology-behind-bitcoin-leaves-people-who-do-not-know-or-trust-each-other-build-dependable>

trottier & GitHub Community. (2018). This is a historical repository of Satoshi Nakamoto's original bitcoin sourcecode. Dostęp from <https://github.com/trottier/original-bitcoin>

Wats0ns, cdecker & Community ©. (2015). A complete bitcoin implementation in Python? Dostęp from <https://bitcoin.stackexchange.com/questions/41682/a-complete-bitcoin-implementation-in-python>

Zeev, N. (2015). "Bitcoin", "Block chain", "Proof of Work" and how they are all connected. Dostęp from <https://www.ness.com/bitcoin-block-chain-proof-of-work-and-how-they-are-all-connected/>

Zeit. (2018). To build server rendered JS web apps with React. Dostęp from <https://learnnextjs.com/>

Zimnoch, D. (2017). Blockchain - technology of the future or overhyped? W *Blockchain*.

Spis tabel

4.1 Przykładowe zasoby w przypadkach użycia blockchain dla biznesu. . . 58

Spis rysunków

2.1 Wizualizacja działania sieci transakcyjnej Bitcoin.	7
2.2 Struktura łańcucha bloków (na podst. Bitcoin).	12
2.3 Próba falsyfikacji bloku oraz jej następstwa.	18
2.4 Rozmiar fizyczny blockchain Bitcoin (w megabajtach).	21
3.1 Draft architektury implementacji.	31
3.2 Przykład wykorzystania Docker dla opracowywania aplikacji.	50
3.3 Kompletna architektura po procesie doboru technologii.	53
4.1 Model sieci blockchain oraz węzły w BigchainDB.	58
4.2 Opis transakcji <i>CREATE</i>	59
4.3 Opis transakcji <i>TRANSFER</i>	60
5.1 „ <i>Fraucymer</i> ” Tadeusza Brzozowskiego wraz z prowieniącją, zlicytowaną za ponad 800 tys. PLN w domu aukcyjnym DESA Unicum.	67
5.2 Zasób biznesowy w sieci powiązań komercyjnego rynku sztuki.	68
5.3 Współdzielony proces biznesowy w zdecentralizowanej sieci kontrahentów operujących zasobem kolektywnym.	70
5.4 Schemat rozproszenia węzłów w sieci z ujęciem architektury aplikacji oraz głównymi technologiami.	72
5.5 Diagram procesu obrotu dziełem sztuki w BPMN.	74
5.6 Formularz katalogowania obrazu wraz z danymi zasobu.	83
5.7 Stworzony blok łańcucha w efekcie procesu tworzenia zasobu obrazu. .	83
5.8 Przeniesienie własności obrazu oraz aktualizacja danych w formularzu.	84

5.9 Pozytywny wynik głosowania poprawności bloku zawierającym przeprowadzane transakcje.	85
5.10 Spreparowany blok transakcji, sztucznie wprowadzony do sieci blockchain w celu dokonania kradzieży.	86
5.11 Negatywny wynik głosowania poprawności dla spreparowanego bloku.	87

Spis kodów źródłowych

2.1	Fragment implementacji <i>chain.h</i> . Źródło: (Bitcoin.org GitHub i GitHub Community, 2018)	13
3.1	Standard ES5 (ECMAScript 5) od roku 2009 do czasów obecnych. Źródło: opracowanie własne	34
3.2	Najnowszy ES2017 (ECMAScript 8). Źródło: opracowanie własne	35
3.3	Przykładowy komponent formularza w React. Źródło: opracowanie własne	37
3.4	Redux + Redux Saga. Źródło: opracowanie własne	40
3.5	ImmutableJS oraz Reselect. Źródło: opracowanie własne	42
3.6	Styled component, CSS w JS. Źródło: opracowanie własne na podstawie (Styled Components, 2018)	43
3.7	Flow, analiza typowania składni. Źródło: opracowanie własne na podstawie (Facebook Open Source, 2018c)	43
3.8	Wymiana danych za pomocą GraphQL. Źródło: opracowanie własne na podstawie (Facebook Open Source, 2018b)	44
3.9	Reagowanie na zdarzenie WebSocket z Socket.IO. Źródło: opracowanie własne na podstawie (Socket.IO, 2018)	45
3.10	NodeJS Express z Async Await. Źródło: opracowanie własne na podstawie (Node.js Foundation, 2018)	46
3.11	Schemat Moongose kolekcji MongoDB. Źródło: opracowanie własne na podstawie (LearnBoost, 2018)	47
3.12	Flow w wykorzystaniu NodeJS. Źródło: opracowanie własne na podstawie (Facebook Open Source, 2018c).	47

3.13 Prosty pogladowy schemat oraz URL dla GraphQL. Źródło: opracowanie własne na podstawie (Facebook Open Source, 2018b)	47
3.14 Propagacja nowej wiadomości od nadawcy do obiorców w SocketIO. Źródło: opracowanie własne na podstawie (Socket.IO, 2018)	48
3.15 Przykładowy obiekt kolekcji <i>users</i> w MongoDB. Źródło: opracowanie własne	51
4.1 Poglądowa struktura głosu w blockchain. Źródło: (BigchainDB GmbH, 2018a)	60
5.1 Fragmenty <i>docker-compose.yml</i> dla aplikacji oraz sieci blockchain. Źródło: opracowanie własne	75
5.2 Rekonfiguracja węzła BigchainDB z jednej instancji w drugą, wraz z nadpisywany parametrami unikatowymi. Źródło: opracowanie własne	76
5.3 Opracowane strony aplikacji klienckiej z poziomu routingu w głównym komponencie <i>App</i> . Źródło: opracowanie własne	77
5.4 Formularz tworzenia zasobu dla sieci blockchain w aplikacji klienckiej. Źródło: opracowanie własne	78
5.5 Transakcja tworzenia zasobu umieszczonego w sieci blockchain po stronie aplikacji serwerowej. Źródło: opracowanie własne	79
5.6 Fragmenty transakcji transferu zasobu od pierwotnego właściciela do wtórnego w aplikacji serwerowej. Źródło: opracowanie własne	80
5.7 Fragmenty bloku transakcji umieszczonego w sieci blockchain. Źródło: opracowanie własne	81