

Praca inżynierska

Kamil Susek

September 2020

Rozdział 1

Wstęp

Rozwój technologii pozwala na tworzenie coraz większych i bezpieczniejszych systemów, co prowadzi do przenoszenia niektórych procesów do świata wirtualnego. Takie działania pozwalają na zwiększenie efektywności i zmniejszenie kosztów tych procesów. Dodatkowo wyeliminowanie wpływu czynnika ludzkiego zwiększa wiarygodność takiego procesu i poprawia bezpieczeństwo. Przykładem takiego procesu jest głosowanie, które w tradycyjnej formie jest czasochłonne i trudne w organizacji. Natomiast przykładem technologii, która doprowadziła do prawdziwej rewolucji jest Blockchain. Blockchain odpowiada za przełomowe rozwiązanie w sektorze finansów, jakim jest Bitcoin, który zapoczątkował rozwój dużych systemów odpowiedzialnych za istotne procesy życia społecznego.

Tematem niniejszej pracy jest E-voting, czyli system głosowania przez internet, który będzie wspierany przez technologię Blockchain. W ramach pracy został przedstawiony teoretyczny model systemu, którego część zaimplementowano. Do zaimplementowanej części systemu należą: aplikacje internetowe wyborcy i organizatora oraz grupa serwisów dostarczających REST-owe API. Aplikacje wyborcy pozwala na zalogowanie się, wysłanie głosu i uzyskanie wyników wyborów. Natomiast aplikacja organizatora dostarcza narzędzia do konfiguracji i nadzorowania procesu wyborczego. Dostarczono również możliwość konfigurowania sieci Blockchain, która jest odpowiedzialna za zapisanie dodawanych głosów. Przez konfigurację sieci Blockchain rozumiane jest tworzenie sieci Peer-to-Peer.

W rozdziale pierwszym zostały przybliżone cele projektu i jego założenia, a także zakres pracy. Rozdział drugi poświęcony jest analizie dziedziny pracy, w tym rozdziale znajdują się informacje teoretyczne dotyczące technologii Blockchain, przykłady jej zastosowania oraz analiza istniejących rozwiązań E-votingu. Rozdział trzeci dotyczy opisu systemu od strony inżynierii oprogramowania. W tym rozdziale znajdują się informacje o wymaganiach projektu, przypadkach użycia oraz użytych narzędziach i technologiach. Rozdział czwarty dotyczy specyfikacji zewnętrznej systemu. Główne treści tego rozdziału to sposób korzystania z aplikacji, instrukcje dotyczące użytkowników systemu i sposób instalacji. Rozdział piąty dotyczy opisu architektury systemu, opisu algorytmów i szczegółów implementacji ciekawych fragmentów systemu. W szóstym rozdziale umieszczono wszystkie informacje dotyczące testowania oraz wykrytych błędów. Ostatni rozdział, czyli rozdział siódmy poświęcony jest podsumowaniu całej pracy, w tym rozdziale znaj-

dużą się uzyskane wyniki i wnioski, a także możliwości dalszego rozwoju pracy.

1.1 Cele i założenia pracy

Celem pracy jest opracowanie oraz zaimplementowanie systemu, który umożliwia przeprowadzenie wyborów przez internet. Głównym założeniem projektu jest wykorzystanie technologii Blockchain. Zakres pracy obejmuje:

- Analizę dziedziny i przegląd literatury dotyczącej wyborów internetowych i technologii Blockchain;
- Opracowanie systemu wyborów internetowych od strony teoretycznej;
- Implementację wybranej części całego systemu;
- Przetestowanie aplikacji.

Rozdział 2

Wstępna analiza dziedziny

Elektroniczne systemy głosowania coraz bardziej zyskują na popularności. Wraz z postępem technologii oraz coraz większego znaczenia internetu w życiu społecznym, pojawiają się pomysły przeniesienia procesu głosowania do sieci. Pomysły i implementacje internetowych systemów wyborczych dotyczą wyborów na szczeblu państwowym (przykładowo wybory parlamentarne), jak i wyborów organizowanych na potrzeby prywatne. System e-votingu zazwyczaj sprowadza się do serwisu internetowego, który pozwala na oddanie głosu poprzez odpowiednią stronę internetową, bądź aplikację desktopową.

Przeniesienie głosowania do aplikacji internetowej, pozwala zminimalizować wpływ ewentualnego błędu ludzkiego podczas przeprowadzania wyborów. Jednakże takie rozwiązanie generuje nowe problemy, z którymi muszą się zmierzyć projektanci tych systemów. Największy problem stanowi zabezpieczenie aplikacji, przed zewnętrznymi próbami fałszowania wyników głosowania. Rozwój technologii prowadzi również do powstawania nowych odmian “złośliwego oprogramowania”, co prowadzi do ciągłego aktualizowania zabezpieczeń. Aplikacja odpowiadająca na potrzeby wyborów, na wysokim szczeblu powinna być tworzona “na potrzeby danych czasów”, bądź łatwa w płynnej aktualizacji.

Zabezpieczenie przed fałszowaniem głosów to nie jest jedyny problem, z jakim trzeba się zmierzyć podczas próby przeniesienia procesu głosowania do internetu. Kolejnym problemem jest sama logika systemu głosowania, niektóre wybory wymagają, aby informacje o wyborcach oraz ich głosach, były tajne. System powinien również umożliwiać weryfikację użytkownika, na podstawie dostarczonych przez organizatora wyborów danych logowania. Coraz głębsza analiza problemu generuje coraz więcej potrzeb z zakresu bezpieczeństwa systemu. Dodatkowo wyszczególniając elementy, które powinny podlegać szczególnej protekcji należy zadbać, aby architektura systemu pozwalała na łatwą aktualizację zabezpieczeń.

Wykorzystując komputery do obsługi głosowania, oczekuje się szybkiego i poprawnego uzyskania rezultatu głosowania. System taki powinien być zoptymalizowany, a czas uzyskania rezultatów powinien być zdeterminowany. Warto rozważyć także moduł generujący statystyki wyborcze.

System wyborczy to nie tylko serwer, który zbiera, przechowuje i liczy głosy. Klientka część aplikacji (widoczna dla wyborcy) powinna być responsywna, przejrzysta oraz przyjazna dla osób z pewnymi niepełnosprawnościami.

Dużą zaletą e-votingu jest przeniesienie procedur, które musi wykonać organizator do interaktywnej aplikacji przeglądarkowej. Aplikacja webowa powinna zapewniać możliwość zarządzania głosowaniem oraz kreator głosowania, ten element również powinien podlegać zabezpieczeniu danych i autentykacji użytkownika. Aplikacja wyposażona w takie funkcjonalności powinna spełniać wymagania elektronicznego systemu głosowania.

2.1 Wstępne wymagania niefunkcjonalne

Z powyższej analizy można wypunktować następujące wymagania:

- Dbłość o zabezpieczenie głosu wyborcy przed fałszerstwem - rezultat głosu nie może zostać zmieniony, po jego zatwierdzeniu.
- Zabezpieczenie danych wyborcy, poprzez utajnienie jego tożsamości.
- Sprawne i bezbłędne liczenie głosów - czas liczenia głosów powinien być deterministyczny.
- Dbłość o stronę wizualną aplikacji, strona kliencka powinna być przejrzysta i łatwa w obsłudze.
- System musi być konfigurowalny, z uwzględnieniem bezpieczeństwa konfiguracji.

Rozdział 3

Blockchain

Podstawy teoretyczne technologii Blockchain powstały już w roku 1991, a zostały opracowane przez dr Stuart'a Haber'a oraz dr W. Scott Stornetta. Naukowcy zajmowali się opracowaniem systemu, zabezpieczającego cyfrowe dokumenty przed podrobieniem, bądź podmianą. System opierał się na łańcuchach bloków, gdzie w każdym bloku znajdowały się cyfrowe dane dokumentu. Podczas dodawania nowego dokumentu do łańcucha, podpisywano go za pomocą tak zwanego stempla czasu (ang. timestamp), a następnie dokument był łączony z poprzednim dokumentem. Łączenie polega na przypisaniu nowemu blokowi, wskaźnika na poprzedni dokument. Wartością wskaźnika były określone dane poprzedniego dokumentu, co stanowiło zabezpieczenie dokumentów. Jeżeli zawartość dokumentu w łańcuchu uległaby zmianie, to należałoby zmienić również wskaźnik na ten dokument [1].

Kamieniem milowym w popularyzacji tego pomysłu był open-source'owy projekt o nazwie Bitcoin. Projekt ten powstał w roku 2009, a jego autorstwo przyznawane jest Satoshi'emu Nakamoto. Bitcoin to kryptowaluta, której podstawa działania oparta jest o wykorzystanie systemu blockchain. Transakcje wykonane za pomocą Bitcoina zapisywane są na tak zwanym arkuszu, który jest widoczny dla każdego uczestnika sieci. Blockchain odgrywa swoją rolę w rejestrowaniu nowych transakcji i zapisywaniu ich w arkuszu. Arkusz ma strukturę łańcucha bloków, a sama architektura Blockchain została ulepszona o działanie w rozproszonej sieci Peer-to-Peer. Peer-to-Peer pozwala na udostępnienie zawartości Bitcoina milionom użytkowników na świecie, a dodatkowo zabezpieczenie sieci. Bitcoin wykorzystuje algorytm konsensusu, w celu zapewnienia spójności zawartości węzłów sieci. Jest to algorytm Proof-of-Work, który odpowiada za silnie kojarzony z Bitcoinem proces "kopania Bitcoinów". Działanie Proof-of-Work polega na włożeniu odpowiedniej ilości mocy obliczeniowej przez węzły, poprzez rozwiązanie zagadki matematycznej, w celu utworzenia nowego bloku. Węzeł który rozwiąże zagadkę jako pierwszy nagradzany jest odpowiednią wartością Bitcoina. Węzeł, który ułoży najdłuższy łańcuch jest synchronizowany z innymi [2].

Kombinacja rozwiązań tworzących Blockchain, w sieci Bitcoin wywołała poruszenie na całym świecie. Zaczęto rozmyślać nad nowymi rozwiązaniami, których podstawą miał być Blockchain. Blockchain znalazł swoje zastosowanie w systemach dokumentowania transakcji, ze względu na redukcję kosztów utrzymania oraz wydajność, co jest spowodowane [3] brakiem pośredników. Dodatkowo tech-

nologia cały czas się rozwija. Kolejnym rewolucyjnym krokiem dla Blockchainu było utworzenie przez Vitalika Buterina nowej kryptowaluty, o nazwie Ethereum. Ethereum poza funkcjami płatniczymi kryptowaluty wprowadza smart kontrakty, czyli możliwość stworzenia kodu i uruchomienia go w systemie Ethereum. Programy mogą wchodzić w interakcje z węzłami w sieci, co pozwala na tworzenie ogromnych rozproszonych aplikacji.

Technologia cały czas podlega rozwojowi i znajduje zastosowanie w nowych dziedzinach, jednakże przedstawione powyżej koncepty są podstawą obecnego systemu Blockchain. W skrócie podstawę systemu Blockchain tworzą [4]:

- Wspólny arkusz - rozproszony łańcuch bloków.
- Ochrona dostępu.
- Smart kontrakty - kontrolowanie i programowanie transakcji.
- Konsensus - zawartość sieci jest bezpieczna, dzięki zgodzie pomiędzy jej uczestnikami.

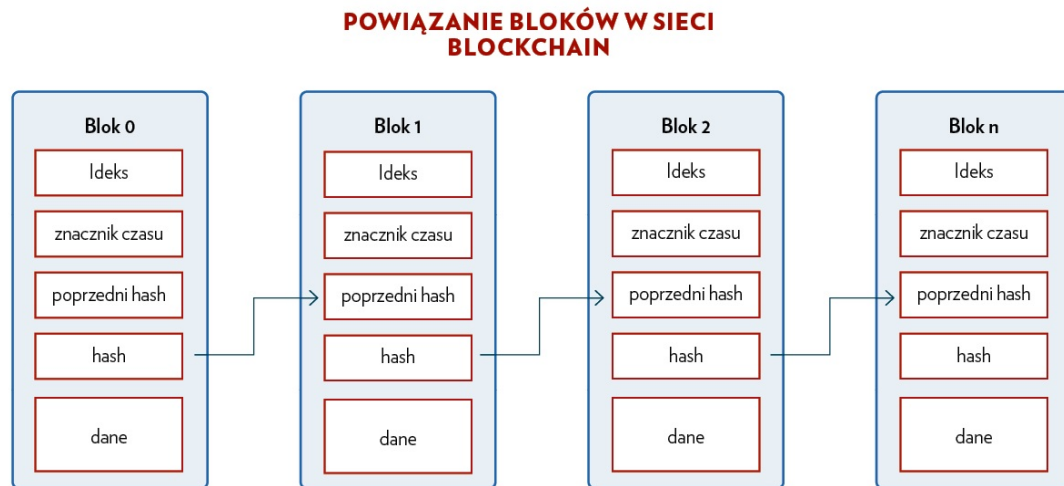
3.1 Architektura Blockchain

Obecnie na system Blockchain składa się wiele rozwiązań, co czasami prowadzi do rozmycia definicji Blockchainu. Mówiąc Blockchain można mieć na myśli sam łańcuch bloków lub rozproszoną sieć urządzeń. Zdarza się nawet że termin Blockchain jest używany zamiennie z terminem Bitcoin [5]. Mylenie Bitcoina, z Blockchainem jest błędem. Jednakże ten przypadek pokazuje, jak ważnym elementem Bitcoina jest Blockchain. W celu uściślenia konceptów charakterystycznych dla Blockchainu, w tej sekcji zajmę się opisaniem składników, które wchodzi w skład technologii Blockchain.

3.1.1 Łańcuch bloków

Łańcuch bloków to można powiedzieć jeden z filarów, na którym opiera się cała architektura współczesnego Blockchain'u. Jak już wspominałem, koncepcja łańcucha bloków pochodzi z pracy naukowej z 1991 roku. Wykorzystana w tym pomysł zasady, według której każdy blok wskazuje na pewne dane obecne w poprzednim, cały czas funkcjonuje w systemach blockchain.

Łańcuch bloków to struktura danych, w której każdy blok wskazuje na określoną wartość poprzedniego bloku. Wartość ta określana jest jako hash. Hash to ciąg znaków utworzony przez odpowiednią metodę kryptograficzną (zwaną funkcją skrótu). Ważną cechą hash'u jest nieodwracalność i bezkolizyjność. Nieodwracalność oznacza, że znając hash, nie można wywnioskować jakie dane zostały za-hash'owane. Natomiast bezkolizyjność oznacza, że nie istnieją dwa różne zestawy danych, które po hash'owaniu dają ten sam wynik. Dodatkowo drobna zmiana zawartości hash'owanych danych prowadzi do uzyskania nowego, kompletnie innego



Rysunek 3.1: Łańcuch bloków.

hash'a. Znalazienie takich danych wymaga dużej mocy obliczeniowej [6]. Doprecyzowanie działania hash'owania będzie miało miejsce w części o tytule Funkcja skrót.

Na rysunku 2.1 ukazany jest schemat łańcucha bloków. Jak widać dane przechowywane są w blokach, a bloki połączone są za pomocą wskaźników na hash poprzedniego bloku. Taka budowa stanowi zabezpieczenie łańcucha bloków przed modyfikacją lub usunięciem danych. Modyfikacja dowolnego bloku w sieci jest widoczna i łatwa do zweryfikowania. Wystarczy policzyć od początku hash dla każdego bloku i porównać z polem następnego bloku, które przechowuje poprzedni hash.

3.1.2 Sieci Peer to Peer

Zmodyfikowanie łańcucha bloków jest trudne, lecz nie jest niemożliwe. Wymaga jedynie wystarczająco dużej mocy obliczeniowej. Dodatkowo zamiast modyfikować łańcuch, można przejąć urządzenie, na którym uruchomiony jest łańcuch bloków, co daje możliwość zakłócenia pracy systemu opartego na działaniu łańcucha.

Twórcy Bitcoina rozwiązali podane problemy wykorzystując zdecentralizowaną sieć Peer-to-Peer. Sieć Peer-to-Peer składa się z węzłów, które komunikują się ze sobą, bez centralnego serwera. Jeżeli węzły chcą się ze sobą skontaktować, to wysyłają dane bezpośrednio do siebie. Takie rozwiązanie pozwala zwiększyć bezpieczeństwo sieci, ponieważ trudniej jest przejąć kontrolę nad wieloma maszynami tworzącymi całą sieć, niż nad centralnym systemem obsługującym działanie tej sieci. Dodatkowo P2P zapewnia zabezpieczenie danych zgromadzonych w sieci, przed usunięciem, ponieważ każde urządzenie przechowuje swoje egzemplarze danych, które można traktować jako kopie zapasowe. Dystrybucja sieci na całym świecie zabezpiecza sieć przed cenzurowaniem, z czego korzysta Bitcoin i inne kryptowaluty.

Peer-to-Peer jest przykładem ciekawego zastosowania architektury klient-serwer. Architektura klient serwer polega na podzieleniu systemu na stronę kliencką, która

zajmuje się wysłaniem żądań do serwera, oraz na stronę serwerową, której zadaniem jest odpowiadanie na żądania, przechowywanie danych i operowanie na tych danych. W sieci P2P węzeł pełni role zarówno serwera, jak i klienta. Przykładowo węzeł może żądać przesłania danych od innego węzła, a także odpowiedzieć na żądanie innego węzła.

3.1.3 Konsensus

Konsensus w sieci Blockchain oznacza, że każdy z jej uczestników zgadza się na stan danych przechowywanych w sieci. Każda aktywność zmieniająca stan przechowywanych danych powinna być autoryzowana, aby zapewnić bezpieczeństwo i spójność sieci. Konsensus pozwala na wprowadzenie do sieci zestawu reguł, według których działa cała sieć. Takie działanie pozwala na zweryfikowanie zaufanego źródła i wykluczenie wpływu węzłów działających na szkodę sieci. W kryptowalutach najczęściej wykorzystywane algorytmy konsensusu to Proof of Work i Proof of Stake. Natomiast Proof of Authority dotyczy rozwiązań prywatnych.

Proof of Work

Proof of work to popularny w kryptowalucie Bitcoin algorytm uzyskiwania konsensusu. Algorytm ten wykorzystywany jest podczas tworzenia nowego bloku. Nowy blok wysyłany jest do wszystkich węzłów w sieci, które w celu dodania bloku do swojego łańcucha muszą zostać zatwierdzone. Na tym etapie algorytmu mamy do czynienia z popularnymi "koparkami" Bitcoinów. Aby blok został zatwierdzony węzeł musi wykonać pracę, czyli dokonać wkładu mocy obliczeniowej. Moc obliczeniowa wykorzystywana jest do rozwiązania zagadki matematycznej. Praca ta wykonywana jest przez węzeł i jest określana jako mining (pol. wydobywanie). Ten węzeł, który jako pierwszy rozwiąże zagadkę, otrzyma nagrodę w postaci określonej ilości środków, w kryptowalucie Bitcoin. W typowych rozwiązaniach dla Bitcoinu poziom trudności dodania nowego bloku jest tak skonstruowany, aby nowe bloki były dodawane, w odstępach około 10 minutowych [7].

Takie rozwiązanie premiuje węzły, o większej mocy obliczeniowej. Dlatego w celu utrzymania spójności danych należy przeprowadzać synchronizację węzłów. Synchronizacja przebiega następująco: każdy węzeł rozsyła do całej sieci swój łańcuch, węzły porównują długość swojego łańcucha z nadesłanym łańcuchem i w węźle zapisywany jest dłuższy łańcuch.

Proof of work to rozwiązanie towarzyszące architekturze Blockchain, od momentu jej powstania. Bitcoin poza dokonywaniem transakcji, pozwala na zarabianie poprzez tworzenie tzw. węzłów górniczych. Węzeł górniczy zajmuje się realizacją algorytmu Proof of Work, co oznacza rozwiązywanie zagadki matematycznej. Fizycznie jest to urządzenie, ze specjalistycznymi podzespołami i oprogramowaniem, które pozwala na uzyskanie wysokiej efektywności w wydobywaniu Bitcoina [9]. Głównym problemem tego rozwiązania jest zużycie energii elektrycznej. W roku 2017 zużycie energii elektrycznej, przez wydobywanie Bitcoina było większe, niż w 159 krajach na świecie [8]. W 2019 naukowcy z University of Cambridge stworzyli specjalny indeks (CBECI), ukazujący dane dotyczące zużycia energii

elektrycznej, przez Bitcoin'a [10].

W tym algorytmie czynnik wkładu fizycznych zasobów, jest głównym motywem do uczciwego postępowania. Jednakże głównym czynnikiem sprawiającym, że ten algorytm działa jest pewność, że ponad 50% uczestników sieci działa uczciwie. Ogrom inwestowanych zasobów wpływa na bezpieczeństwo sieci. Duża zdecentralizowana sieć Blockchain dysponuje ogromną mocą obliczeniową, na jej moc składa się moc wszystkich węzłów w sieci. Węzły różnią się mocą obliczeniową, natomiast decentralizacja sieci sprawia, że żaden z węzłów nie może przejąć kontroli nad całą siecią. Im większa decentralizacja tym bardziej zmniejsza się wpływ pojedynczego węzła na całą sieć. Co innego, gdy do sieci dołączona zostanie grupa węzłów, które będą stanowiły 51% mocy obliczeniowej sieci oraz będą ze sobą współpracowały, działając na szkodę sieci. Taki scenariusz nazywany jest atakiem 51%. Dysponując 51% mocy obliczeniowej sieci, można bez wiedzy uczciwych uczestników, w szybszy sposób dodawać nowe bloki, co prowadzi do przejścia obsługi dodawania zawartości, przez szkodliwych uczestników sieci [11].

W przypadku algorytmu Proof of Work atak 51% jest bardzo rzadkim zjawiskiem, gdyż jest on bardzo drogi. Atak ten może być opłacalny jedynie wtedy, gdy atakujący chce zdestabilizować sieć. W przypadku Bitcoin'a destabilizacja sieci nie ma większego sensu, ponieważ dysponując taką mocą obliczeniową można uzyskać ogromne przychody z kopania Bitcoin'a, dodatkowo uzyskanie takiej mocy obliczeniowej w rozrastających się sieciach Blockchain dla Bitcoin'u jest na tyle drogie, że w teorii jest uznawane za nieosiągalne. Z tego powodu, w przypadku sieci Blockchain, bardzo ważne jest zadbanie o jak największe rozproszenie węzłów w sieci P2P.

Proof of Stake

Proof of Stake to kolejny algorytm używany do osiągnięcia konsensusu w sieciach Blockchain. Jego największe zalety, to rozwiązanie problemu wykorzystania ogromnych ilości zasobów energii elektrycznej przez algorytm Proof of Work.

Działanie algorytmu Proof of Stake polega na wybieraniu spośród dostępnych węzłów sieci Blockchain walidatora. Walidator to węzeł, którego zadaniem jest weryfikacja poprawności bloku dołączanego do łańcucha. Wybór walidatora może się odbywać na różne sposoby, od całkowicie losowego wyboru, po wybór z uwzględnieniem czynników takich jak np. wiek stawki węzła, wysokość stawki. Stawka to zablokowane na koncie danego węzła jednostki waluty (kryptowaluty). Stawka jest wymagana, aby węzeł mógł uczestniczyć w losowaniu.

W przypadku algorytmu Proof of Stake głównym motywem uczciwego uczestnictwa w sieci jest możliwość utracenia części stawki. W przypadku wykrycia przez sieć nieprawidłowości w bloku dodanym przez konkretny węzeł, traci on określoną część stawki.

Proof of Stake jest o wiele bardziej ekologiczny, jeśli chodzi o zużycie energii elektrycznej, niż Proof of Work. Oznacza to, że sieć jest dostępna dla większej ilości urządzeń, co wpływa na zwiększenie rozproszenia sieci. W Proof of Stake głównym czynnikiem determinującym moc sieci jest zgromadzony w niej wkład finansowy w postaci stawek. W przypadku tego algorytmu również występuje pro-

blem ataku 51%, jednakże tym razem atakujący musi dysponować odpowiednią ilością środków płatniczych, a dokładnie musi włożyć 51% ogólnej stawki, co w przypadku Bitcoin'a jest mało prawdopodobne.

Proof of Authority

Proof of Authority to rozwiązanie skłaniające się ku mniej zdecentralizowanym systemom, które są tworzone na prywatne potrzeby i wymagających dużej przepustowości. Proof of Authority jest w swoim działaniu nieco zbliżony do Proof of Stake. Zamiast stawki podanej w jednostkach odpowiedniej waluty, w PoA używana jest reputacja walidatora. Prowadzi to do skonstruowania sieci, w której uczestniczą tylko zaufani walidatorzy. Uzyskanie statusu walidatora wiąże się z dużym nakładem pracy, w celu spełnienia kryteriów i uzyskania odpowiednio wysokiej reputacji. Podczas projektowania takiego systemu należy zwrócić szczególną uwagę, na stworzenie kryteriów o wysokiej trudności do spełnienia, aby wykluczyć węzły o szkodliwym działaniu.

Trudność uzyskania statusu walidatora bardzo mocno wpływa na stopień zdecentralizowania. Decentralizacja porównywalna z rozwiązaniem Proof of Stake jest niemożliwa do osiągnięcia. Niski stopień decentralizacji wymaga pełnego zaufania do walidatorów, co może stanowić słaby punkt systemu, gdy zaufany walidator zostanie przekupiony i dokona szkodliwej manipulacji na zawartości systemu.

Algorytm Proof of Authority świetnie sprawdza się w prywatnych sieciach Blockchain, gdy ważna jest wysoka przepustowość łącza. Jednakże należy pamiętać o odpowiednim doborze walidatorów.

3.1.4 Funkcja skrótu

Funkcje skrótu w systemach informatycznych pełnią rolę zabezpieczenia integralności danych. Zabezpieczenie danych polega na wykryciu modyfikacji. Matematyczna definicja funkcji skrótu h to odwzorowanie wiadomości m o dowolnej, skończonej długości, w ciąg bitów o określonej, stałej długości n [6]:

$$h : \{0, 1\}^* \rightarrow \{0, 1\}^n, \text{ gdzie } : \{0, 1\}^* = \bigcup_{i \in N} \{0, 1\}^i, N = \{0, 1, \dots\}, n \in N \quad (3.1)$$

Funkcja skrótu (hash) cechuje się wysoką podatnością na zmiany. Zmiana jednego bitu w danych wejściowych powoduje otrzymanie zupełnie innych wyników. Przykładowo dla ciągu znaków hash0, dla funkcji skrótu SHA256 wynikiem jest ciąg znaków

6ac73742db534beebccc9af1453c5637ee5bb5d7c9628ec2f26cf9777c89e96d8. Natomiast

zmieniając ciąg znaków na hash1, otrzymanym wynikiem jest

af316ecb91a8ee7ae99210702b2d4758f30cdde3bf61e3d8e787d74681f90a6. Przykład obrazuje jak na wynikowy ciąg znaków wpływa zmiana jednego symbolu na wejściu.

Głównymi cechami funkcji skrótu są:

- Nieodwracalność - na podstawie otrzymanego wyniku nie można bezpośrednio odtworzyć danych.

- Bezkolizyjność - nie istnieją dwa różne ciągi znaków, które dają ten sam wynik.

Znalezienie danych wejściowych dla funkcji skrótu jest zadaniem wymagającym dużego nakładu mocy obliczeniowej. Dane można uzyskać losując wejściowy ciąg znaków i hashując go, a następnie porównując otrzymany hash z hashem, który ma być rozszyfrowany. Jest to tak zwany atak brutalny (ang. brute force).

Popularnym zastosowaniem funkcji skrótu jest podpis cyfrowy. Podpis cyfrowy to zaszyfrowany hash, wygenerowany na podstawie danych potrzebnych do podpisu. Głównym zaletą wykorzystania hash'owania w podpisie cyfrowym jest szybkość. Hashowanie wiadomości jest szybsze od szyfrowania, a zaszyfrować wystarczy sam ciąg znaków o stosunkowo małej długości, w porównaniu do tekstu wiadomości.

Funkcje skrótu można spotkać w bazach danych. Hasła są przykładem informacji, które w bazie danych są przechowywane w postaci funkcji skrótu. Takie rozwiązanie zabezpiecza hasła, w przypadku wycieku danych.

3.1.5 Smart kontrakty

Wraz z powstaniem kryptowaluty Ethereum do architektury Blockchain została dodana nowa funkcja. Dodano tak zwane Smart Kontrakty, które pozwalają na zaprogramowanie i uruchomienie kodu w sieci Blockchain. Takie programy można stosować, w celu kontrolowania sieci poprzez zdefiniowanie zestawu reguł dla sieci. Zdefiniowanie zestawu reguł pozwala na zwiększenie zaufania w sieci, a także automatyzację niektórych procesów.

Smart kontrakty pozwalają na tworzenie aplikacji, których podstawą jest Blockchain. Przykładem środowiska pozwalającego na tworzenie takich programów jest język Solidity, który został stworzony dla sieci Ethereum [16].

3.2 Technologia Blockchain na rynku

3.2.1 Kryptowaluty

Kryptowaluta to cyfrowy środek płatniczy. W przeciwieństwie do tradycyjnych walut, kryptowaluty nie są kontrolowane przez bank centralny. Środki płatnicze w kryptowalucie znajdują się w rozproszonej, po całym świecie sieci. Każda transakcja w tym środowisku jest zapisywana we współdzielonym rejestrze. Rejestr pozwala na ustalenie ile środków w danej kryptowalucie jest na koncie danego uczestnika sieci. Podstawą działania kryptowalut jest Blockchain. To Blockchain odpowiada za tworzenie rejestru transakcji, zabezpieczenie danych przed modyfikacją, współdzielenie danych i uzyskanie konsensusu. Przykładem takich kryptowalut są Bitcoin i Ethereum.

3.2.2 Finanse

Dynamiczny rozwój technologii zapoczątkował rozwój innych branż, w tym sektora finansów. Powstała osobna gałąź tego sektora o nazwie FinTech. FinTech zajmuje się opracowywaniem i wdrażaniem nowych technologii do systemów finansowych na całym świecie. Korzyści płynące z unowocześnienia sektora finansów mogą skupiają się głównie na zwiększeniu szybkości transakcji przez brak pośredników, zmniejszeniu kosztów przepływu transakcji i zwiększenie bezpieczeństwa. FinTech zajmuje się między innymi szukaniem zastosowań dla technologii Blockchain. Jako przykład rozwijania technologii blockchain przez FinTech może posłużyć przykład trwałego nośnika wdrożonego przez PKO Bank Polski. Trwały nośnik pozwala na dostarczanie klientom banku prywatnych dokumentów w formie cyfrowej [12]. Twórcy tego rozwiązania zastosowali również mechanizm Smart Kontraktów. Smart Kontrakty są wykorzystywane do zautomatyzowania procesu egzekwowania umów ubezpieczeniowych [13]. Pozwala to na zmniejszenie uczestnictwa osób trzecich i redukcję kosztów.

3.2.3 Internet of Things

Internet of Things to technologia pozwalająca na łączenie ze sobą wielu urządzeń. Tak skonstruowane sieci znajdują swoje przeznaczenie w systemach czasu rzeczywistego w przemyśle, bądź w inteligentnym domu. IoT wykorzystuje serwery internetowe, do przechowywania i pobierania danych. Blockchain w usługach IoT pomaga zabezpieczyć serwery z danymi, poprzez ich decentralizację.

3.3 Przegląd istniejących rozwiązań e-votingu

3.4 System estoński

Przykładem regularnego wykorzystania e-votingu jest Estonia. Estonia jest krajem, który pierwszy udostępnił możliwość głosowania elektronicznego w wyborach lokalnych, na szczeblu krajowym. W pierwszych wyborach wzięło udział około 1% wyborców. Od roku 2005 w Estonii postępował rozwój systemów e-votingu. Wraz z następnymi wyborami zwiększała się liczba uczestników. W roku 2019 liczba wyborców, którzy skorzystali z e-votingu wyniosła 43,8% wyborców.

Na system e-votingu w Estonii składa się kilka mniejszych rozwiązań, które można wylistować w następujący sposób:

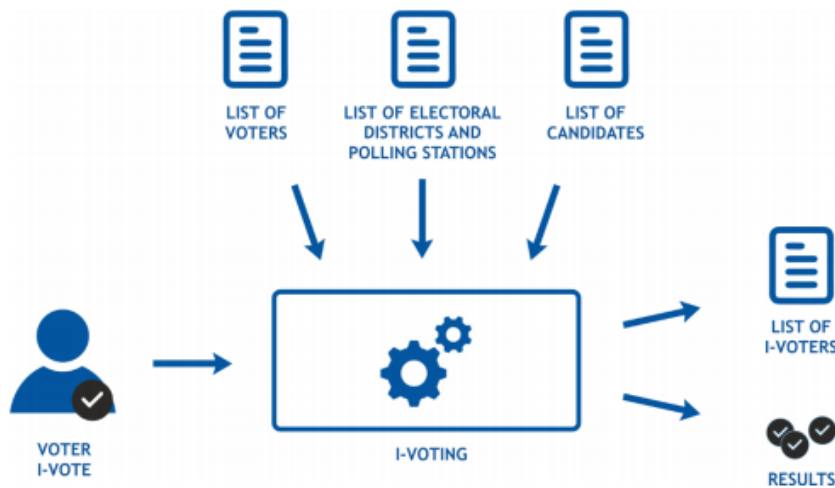
- Identyfikacja za pomocą karty e-obywatela lub profilu e-obywatela w telefonie.
- Architektura rozwiązania pozwala na wysłanie wielu głosów, a głosem wiążącym jest zawsze głos finalny.
- Serwery wykorzystywane podczas głosowania są pod szczególną ochroną i nie można uzyskać do nich dostępu z bezpośrednio z internetu (zabezpieczenie firewallem).
- Wykorzystywanie prywatnych kluczy i narzędzi kryptograficznych w celu zabezpieczenia dostępu do danych. Wykorzystywanie standardu SSL.

3.4.1 Przegląd rozwiązania

Przebieg całego procesu wyborczego w Estonii można podzielić na kilka etapów:

- ogłoszenie wyborów;
- zarejestrowanie kandydatów;
- przygotowanie list wyborczych;
- głosowanie;
- liczenie głosów;
- ogłoszenie wyników;

Wsparcie e-votingu (w Estonii określanego i-votingiem) obejmuje ostatnie trzy etapy procesu wyborczego.



Rysunek 3.2: Moduły systemu i-votingu. [17]

W skład systemu głosowania wchodzi bazy danych przechowujące:

- listę osób uprawnionych do głosowania;
- listę okręgów wyborczych;
- listę kandydatów lub opcji wyborczych;
- listę e-wyborców (i-voters);

Dostarczenie odpowiednich danych do systemu pozwala na walidację wyborcy, zgłoszenie przez niego głosu oraz zapisanie wyborcy, w bazie danych odpowiedzialnej za przechowanie listy e-wyborców. W skład logiki systemu wchodzi mechanizmy generujące wyniki. Wyniki e-votingu są scalane z wynikami wyborów tradycyjnych, przy czym scalanie nie pozwala na podwójne zliczanie głosu oddanego tradycyjnie i elektronicznie.



Rysunek 3.3: Proces wyborczy. [17]

Główne części systemu:

- Voter (wyborca) - za pośrednictwem aplikacji klienckiej (która w systemie estońskim jest aplikacją desktopową pobieraną przed wyborami) szyfruje i potwierdza swoim podpisem elektronicznym głos, który następnie wysyłany jest do Kolektora.
- Collector (Zbieracz głosów) - to aplikacja serwerowa wyposażona w logikę pozwalającą na utworzenie głosu. Aplikacja waliduje dane wprowadzone przez wyborcę i elektronicznie podpisuje dane, które następnie przesyła do Procesora.
- Processor (Procesor głosów) - zajmuje się przetwarzaniem głosów. Sprawdza poprawność danych otrzymanych z Kolektora, wraz z podpisem elektronicznym. Usuwa głosy, które się powtarzają, zarówno głosy oddane elektronicznie, jak i te oddane w lokalach wyborczych. Sortuje głosy według okręgów wyborczych i usuwa z nich podpis elektroniczny, w celu anonimizacji głosu. Tak przetworzone głosy są mieszane według odpowiedniego algorytmu i wysyłane do Licznika.
- Tallier (Licznik głosów) - rolą licznika jest odebranie głosu od Procesora, otwarcie go i dodanie przyporządkowanie do odpowiedniego wyniku.

Estoński system E-votingu jest spójny i silnie zabezpieczony. Dlatego jest on wykorzystywany w ważnych wyborach krajowych. Rozwiązanie jest stale udoskonalane, aby mogło dotrzymać kroku rozwojowi technologii. Omówiony system pochodzi z roku 2016 [17].

3.5 Evoting w Walnych Zgromadzeniach

Krajowy Depozyt Papierów Wartościowych to instytucja finansowa, której zadaniem jest nadzorowanie i prowadzenie rejestracji obrotu instrumentami finansowymi w Polsce. KDPW pełni również rolę depozytu papierów wartościowych. KDPW jest spółką akcyjną, której współwłaścicielami są akcjonariusze - państwo (Skarb Państwa) i państwowe spółki finansowe (Giełda Papierów Wartościowych, Narodowy Bank Polski).

Walne Zgromadzenie to organ mający najwyższe uprawnienia w spółce akcyjnej. Walne Zgromadzenie pozwala akcjonariuszom na zarządzanie spółką. Podczas Walnego Zgromadzenia podejmuje się uchwały dotyczące spółki akcyjnej.

Krajowy Depozyt Papierów Wartościowych udostępnia swoim akcjonariuszom aplikacje e-voting oraz Walne Zgromadzenia, które przenoszą proces przeprowadzania Walnego Zgromadzenia do internetu. Zgodnie z art. 402 Kodeksu Spółek Handlowych [14], Walne Zgromadzenie zwoływane jest przez ogłoszenie go, co najmniej trzy tygodnie przed zaplanowanym terminem. W celu zwołania Walnego Zgromadzenia w aplikacji Evoting i Walne Zgromadzenia należy zarejestrować Walne Zgromadzenie. KDPW przesyła ogłoszenie, o Walnym Zgromadzeniu do jego uczestników. Wymiana informacji odbywa się drogą elektroniczną. Uczestnicy tworzą listę osób uprawnionych do udziału w Walnym Zgromadzeniu, która

jest przesyłana do KDPW. KDPW generuje wykaz osób uprawnionych do uczestnictwa w Walnym Zgromadzeniu, a następnie dostarcza informacje o uczestnictwie w Walnym Zgromadzeniu indywidualnie, dla każdego uczestnika. Każdy uczestnik otrzymuje również kod autoryzacyjny, który wykorzystywany jest do potwierdzenia tożsamości uczestnika, w celu przyznania uprawnień do uczestnictwa w Walnym Zgromadzeniu. Przyznanie uprawnień jest równoznaczne z potwierdzeniem obecności na Walnym Zgromadzeniu i udostępnieniu uczestnikowi interfejsu do głosowania [15].

Za zapewnienie nieodwracalności wykonanych akcji i ogólnego bezpieczeństwa danych, w aplikacjach eVoting i Walne Zgromadzenia jest odpowiedzialny Blockchain.

Rozdział 4

Wymagania systemu i wykorzystane narzędzia

W tym rozdziale sformułowano i omówiono wymagania funkcjonalne i niefunkcjonalne. Dodatkowo przedstawiono przypadki użycia i aktorów systemu. Na koniec opisane zostały technologie i narzędzia wykorzystane podczas tworzenia systemu.

4.1 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne definiują jakość usług dostarczanych przez system. W systemie e-votingu ważnymi elementami są bezpieczeństwo i czytelność procedury głosowania. Dlatego w celu uściślenia oczekiwań jakościowych systemu sformułowano następujące wymagania niefunkcjonalne:

- dane logowania użytkownika są zabezpieczone,
- dane logowania administratora są zabezpieczone,
- przechowywane głosy są zabezpieczone przed modyfikacją,
- przejrzyste treści na stronie wyborcy,
- czytelna i łatwa w obsłudze procedura wyborcza na stronie wyborcy,
- przejrzyste treści na stronie administratora wyborów,
- czytelna i łatwa w obsłudze konfiguracja wyborów,
- aplikacje internetowe wyborcy i administratora działają na najpopularniejszych przeglądarkach internetowych.
- czas liczenia głosów jest zdeteminowany.

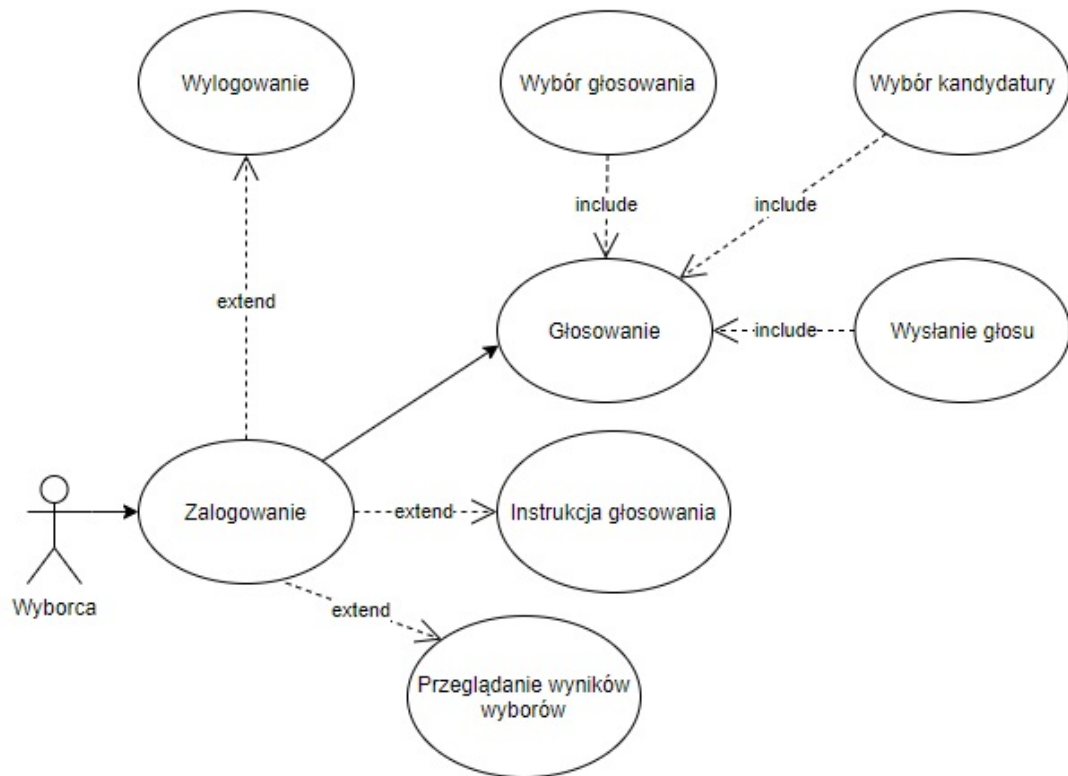
4.2 Wymagania funkcjonalne

W tej sekcji umieszczono wszystkie funkcjonalności i części systemu. Do głównych części systemu należą: aplikacja internetowa wyborcy, aplikacja internetowa administratora wyborów, serwis udostępniający REST (Representational State Transfer) API i serwery sieci Blockchain. Do wymagań funkcjonalnych systemu należą:

- hasła przechowywane w bazie danych są szyfrowane za pomocą funkcji skrótu SHA256,
- dostęp do zasobów wyborcy i administratora odbywa się poprzez dwie inne aplikacje internetowe,
- komunikacja pomiędzy aplikacjami przeglądarkowymi, a API odbywa się z wykorzystaniem konwencji REST API,
- dostęp do punktów końcowych API (endpointów) jest zabezpieczony za pomocą JWT (Json Web Token) oraz uwierzytelnienia na podstawie roli (Role Based Authentication),
- dostęp do zasobów posiadają tylko zalogowani użytkownicy,
- za bezpieczne przechowywanie głosów odpowiada sieć Blockchain,
- każdy węzeł sieci Blockchain to indywidualny program umożliwiający konfigurację sieci,
- dla każdego wyborów konfigurowana jest indywidualna sieć Blockchain,
- dostęp do sieci Blockchain uzyskiwany jest za pomocą serwisu,
- komunikacja pomiędzy serwisem, a siecią Blockchain odbywa się z udziałem jednego węzła, który odbiera informacje i rozsyła po całej sieci,
- konsensus w sieci może zostać uzyskany z wykorzystaniem algorytmu Proof of Work lub Proof of Authority.

4.3 Diagramy przypadków użycia

W systemie można wyróżnić dwóch aktorów. W tym podrozdziale zostały przedstawione i omówione diagramy przypadków użycia dla aktorów.



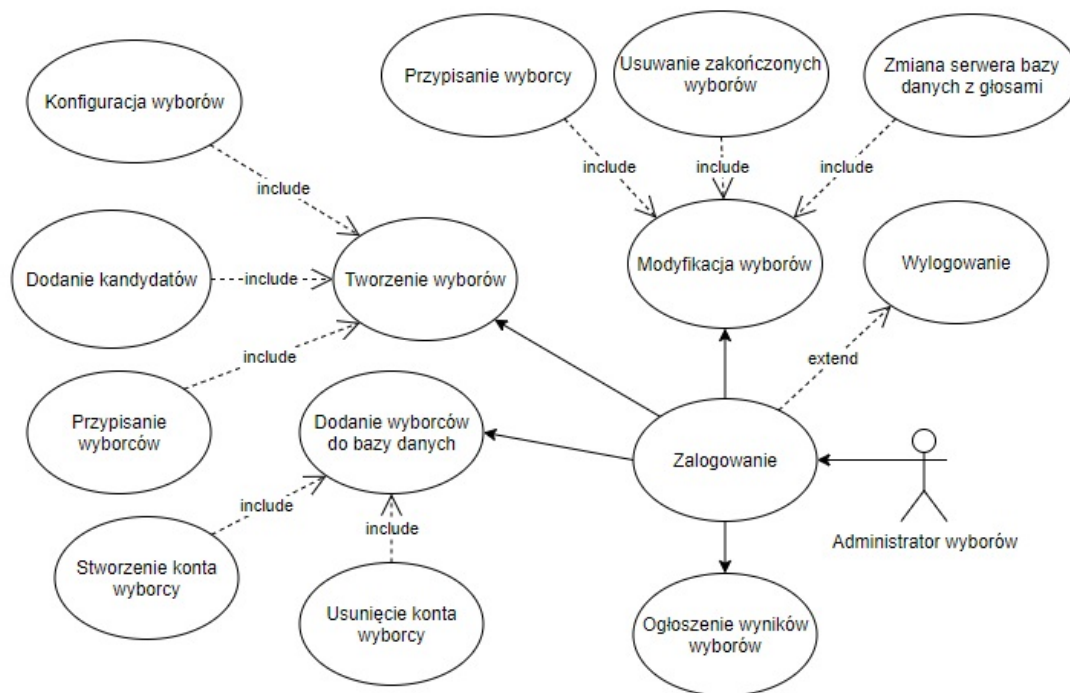
Rysunek 4.1: Diagram przypadków użycia wyborcy.

Pierwszym z wyszczególnionych aktorów jest użytkownik aplikacji, czyli wyborca. Przewidziane akcje dostępne dla wyborcy to zalogowanie, wylogowanie, dostęp do instrukcji głosowania, głosowanie i odbiór wyników wyborów. Rysunek 4.1 przedstawia diagram przypadków użycia dla wyborcy.

Drugim aktorem występującym w systemie jest administrator wyborów. Do zadań administratora należą:

- tworzenie wyborów - konfiguracja składników wyborów,
- modyfikacja wyborów - nadzorowanie procesu wyborczego,
- dodawanie wyborców do bazy danych - zarządzanie kontami wyborców,
- ogłoszenie wyników wyborów.

Dodatkowo administrator posiada swoje konto w systemie. Aby mógł korzystać z przydzielonych mu zasobów musi się zalogować. Rysunek 4.2 przedstawia diagram przypadków użycia dla administratora wyborów.



Rysunek 4.2: Diagram przypadków użycia administratora.

4.4 Wykorzystane narzędzia i technologie

System e-votingu jest składany z wielu modułów, których stworzenie obejmuje różne dziedziny programowania. W systemie są elementy odpowiedzialne za interfejs użytkownika, aplikację serwerową i bazę danych. Każdy z tych rodzajów oprogramowania wymaga innego podejścia. Są to popularne dziedziny programowania, dlatego znalezienie rozwiązań nie stanowi problemu. W tej sekcji przedstawiono wykorzystane narzędzia i technologie, a także ich zastosowania w projekcie.

4.4.1 Interfejs użytkownika

W celu stworzenia rozbudowanego interfejsu użytkownika wykorzystano bibliotekę javascript-ową - ReactJs. React pozwala na tworzenie aplikacji internetowych z wykorzystaniem języków javascript, html i css. Takie podejście do tworzenia aplikacji określane jest jako SPA (Single Page Application), taka aplikacja posiada jeden plik html, którego zawartość jest dynamicznie generowana w trakcie działania aplikacji. Podejście SPA pozwala na zredukowanie liczby zasobów potrzebnych do generowania strony, poprzez wykorzystanie tylko jednego pliku html. Strona typu SPA działa płynnie, ponieważ nie wymaga ona ładowania szablonów z plików html. Ważną cechą ReactJs jest podejście do interfejsu użytkownika, jak do grupy komponentów. Komponent może być dowolnym elementem interfejsu, od paska nawigacji do przycisku. Każdy komponent posiada swój stan, czyli zestaw zmiennych kontrolujących jego funkcjonalności. Takie podejście pozwala na tworzenie uniwersalnych komponentów, które można wykorzystywać w wielu miej-

scach aplikacji.

Biblioteka ReactJs to bardzo rozbudowane narzędzie, które dzięki swojej popularności posiada wiele dodatkowych narzędzi współpracujących z całym środowiskiem ReactJs. Przykładem takiego dodatku jest biblioteka Redux, która pozwala na zarządzanie stanem aplikacji. Redux pozwala przenieść stan z komponentu do osobnego kontekstu. Takie rozwiązanie pozwala na umieszczenie tych części stanu aplikacji, które odnoszą się do różnych komponentów w jednym miejscu. Przykładowo stan informujący o zalogowaniu do aplikacji powinien być dostępny dla całej aplikacji. Przeniesienie tego stanu do osobnego kontekstu pozwala na zdefiniowanie go w jednym miejscu i dostarczenie do komponentów, które wymagają tego stanu. W projekcie wykorzystano opartą na Redux, bibliotekę Redux-Toolkit. Głównym argumentem za wykorzystaniem Redux-Toolkit jest mniejsza objętość kodu, która powstaje do uzyskania rozwiązania.

Podstawowym narzędziem odpowiedzialnym za wygląd interfejsu aplikacji internetowej są kaskadowe arkusze stylów, czyli popularny CSS (Cascade Style Sheets). Język css jest jedynym językiem odpowiedzialnym za stylowanie wyglądu, który jest rozumiany przez przeglądarkę. Jednakże istnieją rozwiązania, które pozwalają na wprowadzenie nowej składni z zachowaniem kompatybilności z danym środowiskiem uruchomieniowym (w tym przypadku z przeglądarką). Są to tzw. preprocesory. Preprocesor pozwala na przetworzenie kodu źródłowego na określony kod wyjściowy. Przykładem preprocesora języka CSS, który został wykorzystany w projekcie jest Sass. Sass dostarcza funkcjonalności, które nie są dostępne w języku CSS. Przykładowo Sass pozwala na zagnieżdżanie instrukcji, deklarowanie i wywoływanie zmiennych oraz tzw. mixins, które pozwalają na zadeklarowanie powtarzających się fragmentów kodu w jednym miejscu i ich wywoływanie bez konieczności przepisywania kodu. Sass pozwala na lepsze wykorzystanie dobrych praktyk programowania, takich jak nie powtarzanie kodu i ogólna poprawa jakości kodu.

Jak już wspomniano w tej sekcji ReactJs jest popularną biblioteką, co owocuje dużą ilością rozszerzeń i dodatków. Dodatki obejmują również stylowanie aplikacji. W projekcie została wykorzystana biblioteka Material-ui, która dostarcza gotowe zestawy komponentów i narzędzia do ich stylowania. Zaletą tego rozwiązania jest wykorzystywanie komponentów, które stworzono z wykorzystaniem dobrych praktyk tworzenia interfejsu.

Aplikacje internetowe często wymagają komunikacji z zewnętrznym serwerem, w celu pobrania lub wysłania odpowiednich zasobów. Takie działanie to przykład architektury klient-serwer. Istnieje wiele rozwiązań, które pozwalają na wysyłanie żądań w standardzie HTTP. W projekcie wykorzystano javascript-ową bibliotekę Axios. Biblioteka dostarcza wszelkich mechanizmów do wysyłania żądań HTTP, a jej wielką zaletą jest łatwość obsługi.

Javascript to język programowania, który świetnie sprawdza się w środowisku aplikacji przeglądarkowych. Język umożliwia wieloaspektowe podejście do programowania. Można pisać programy z podejściem deklaratywnym lub obiektowym. Javascript spełnia najważniejsze standardy, dla nowoczesnych języków programowania, o czym świadczyć może ogromna popularność tego języka. Jednakże niektóre właściwości tego języka mogą utrudniać tworzenie oprogramowania. Przykładem jest

typowanie dynamiczne. W Javascript każda zmienna nie ma przypisanego typu na stałe. Takie rozwiązanie ma swoje wady i zalety, jednak programista powinien mieć wybór, kiedy użyć zmiennej typowanej dynamicznie, a kiedy nie. Rozwiązaniem tego problemu jest język Typescript. Typescript jest nadzbiorem javascript co oznacza, że dostarcza on wszystkie funkcje Javascriptu i jednocześnie rozszerza go o nowe. W Typescript można deklarować zmienne z określonym typem, a także używać typu `any`. Typ `any` określa zmienne, które mogą być typowane dynamicznie. Język Typescript dostarcza również możliwość definiowania własnych typów. W projekcie, w celu uniknięcia problemów związanych z dynamicznym typowaniem wykorzystano język Typescript.

4.4.2 Serwer

Za stronę serwerową aplikacji odpowiada silnik NodeJs. NodeJs pozwala na tworzenie aplikacji w Javascript, poza środowiskiem przeglądarkowym. Silnik stanowi również rozbudowane narzędzie pozwalające na tworzenie REST API. Zaletą NodeJs jest ogromna popularność, co jest zauważalne podczas szukania rozwiązań na portalu Stackoverflow. Popularność silnika przekłada się również na jego bogatą dokumentację i dodatkowe biblioteki, które usprawniają elementy tworzenia oprogramowania.

ExpressJs to biblioteka działająca na silniku NodeJs, która została wykorzystana w projekcie do tworzenia REST API i węzłów sieci Blockchain. ExpressJs dostarcza podstawowe narzędzia do tworzenia REST API, takie jak endpointy (punkty końcowe serwera) pozwalające na obsługiwanie żądań HTTP. Zaletą biblioteki ExpressJs jest kompatybilność z wieloma innymi bibliotekami przeznaczonymi dla silnika NodeJs.

Prisma to jedna z bibliotek NodeJs, które są odpowiedzialne za obsługę bazy danych. Prisma dostarcza mechanizm ORM (Object Relational Mapping), czyli możliwość odwzorowania relacyjnej bazy danych, na obiekty. Prisma generuje moduły odpowiedzialne za obsługę bazy danych i dostarcza je programiście. Takie rozwiązanie pozwala na utrzymanie całości aplikacji w jednym języku programowania, ponieważ nie wymaga korzystania z języka SQL, w celu zarządzania bazą danych z poziomu aplikacji. Prisma pozwala oddzielić implementację bazy danych, od innych funkcjonalności aplikacji.

4.4.3 Baza danych

Wykorzystana baza danych to SQLite. Baza danych znajduje się w pliku i jest umieszczona na serwerze wraz z REST API. SQLite to szybki, mały i łatwy w konfiguracji silnik baz danych. Jest to świetny silnik do prototypowania i tworzenia małych, a zarazem przenośnych baz danych.

Silnik SQLite można obsługiwać z poziomu konsoli lub za pomocą specjalnej aplikacji, która dostarcza interfejs graficzny prezentujący zawartość bazy danych. W projekcie wykorzystano aplikację SQLiteStudio, która pozwala na połączenie się z bazą danych, w celu zarządzania nią. SQLiteStudio pozwala na tworzenie tabel, dodawanie zawartości, usuwanie danych i przeszukiwanie bazy danych. Aplikacja

pozwała również na pisanie własnych zapytań w języku SQL. Tego rodzaju aplikacje są przydatne podczas tworzenia tabel w bazie danych i testowania aplikacji.

4.4.4 Testowanie

Testowanie strony serwerowej systemu odbywało się przez aplikację Postman, która pozwala na wysyłanie żądań HTTP każdego rodzaju. W aplikacji można skonfigurować żądanie HTTP, tak aby zasymulować odpowiednie scenariusze żądania. Pozwala to przetestować działanie wszystkich funkcjonalności API.

Testowanie aplikacji w środowisku lokalnym pozwala odpowiedzieć na pytanie, czy całość komponentów działa w kontrolowanym środowisku. Natomiast przeniesienie aplikacji do sieci pozwala na przetestowanie wydajności aplikacji. W projekcie, w celu udostępnienia aplikacji w internecie wykorzystano serwisy Netlify oraz Heroku. Netlify pozwala na udostępnienie aplikacji internetowej, pod adresem generowanym przez serwis. Heroku zajmuje się uruchomieniem aplikacji serwerowej, która również dostępna jest w sieci pod odpowiednim adresem.

4.4.5 Metodyka pracy

Do prowadzenia projektu wykorzystano repozytorium kontroli wersji GitHub. GitHub to narzędzie, które pozwala przechowywać kod źródłowy aplikacji, a także zapewnia jego archiwizację i wersjonowanie. Do przydatnych funkcji należy również tworzenie rozgałęzień na różnych etapach tworzenia aplikacji. Pozwala to na oddzielenie poszczególnych funkcji na odrębne gałęzie projektu. Takie podejścia pozwala na uporządkowanie pracy nad projektem i większą przejrzystość w procesie rozwoju modułów.

GitHub pozwala również na synchronizację ze wspomnianymi Heroku i Netlify, co pozwala na automatyzację procesu umieszczania nowych wersji aplikacji w środowisku internetowym. Wraz z umieszczeniem nowej wersji aplikacji w repozytorium, serwisy otrzymują kod źródłowy za pośrednictwem GitHuba, a następnie budują i uruchamiają aplikację.

Rozdział 5

Specyfikacja zewnętrzna

Ten rozdział zawiera szczegółowy opis funkcjonalności systemu. Zawarto w nim opis instalacji, uruchomienia i korzystania z systemu. Opis dotyczy pracy systemu w środowisku lokalnym, ponieważ uzyskania działającego systemu w środowisku internetowym opiera się o wykorzystanie odpowiednich serwisów internetowych. Przykładowo może to być serwis Heroku lub Netlify.

5.1 Instalacja

Przed rozpoczęciem instalacji poszczególnych modułów systemu należy zainstalować silnik NodeJs oraz środowisko NPM. Silnik NodeJs jest niezbędny do uruchomienia poszczególnych serwerów systemu. NPM (Node Packet Manager) to menadżer pakietów, który dostarcza mechanizmy pozwalające uruchamiać odpowiednie skrypty, które pozwalają na konfigurację serwera.

5.1.1 Instalacja dla Windows i MacOS

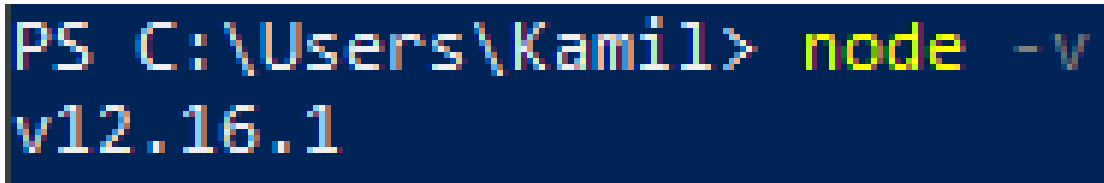
Dla systemów operacyjnych Windows i MacOS pakiet instalacyjny znajduje się na stronie NodeJs - <https://nodejs.org/en/download/>. Na stronie należy wybrać opcję *Windows Installer* lub *macOs Installer*, a pobieranie rozpocznie się automatycznie. Pobrany instalator przeprowadzi również instalację NPM.

5.1.2 Instalacja dla Linux

Instalacja wymaganego oprogramowania dla systemu Linux może zostać przeprowadzona z wykorzystaniem polecenia *sudo*. W terminalu należy wpisać następujące polecenia:

```
sudo apt install nodejs  
sudo apt install npm
```

W celu sprawdzenia poprawności instalacji dla każdego systemu można wykonać terminal (lub konsolę w przypadku Windows). W terminalu należy wpisać: *node -v*, co powinno skutkować wyświetleniem aktualnie zainstalowanej wersji NodeJs. Tak jak na rysunku 5.1.

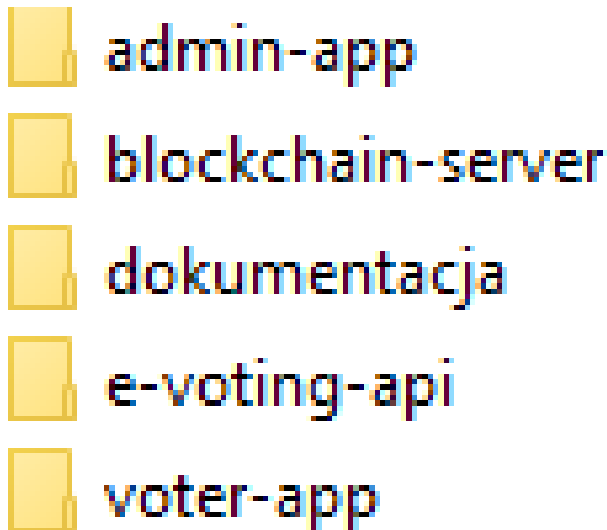


```
PS C:\Users\Kamil> node -v
v12.16.1
```

Rysunek 5.1: Rezultat polecenia `node -v`

5.2 Struktura oprogramowania

System składa się z kilku modułów, dlatego dostarczone oprogramowanie jest podzielone na kilka folderów. Struktura folderów została przedstawiona na rysunku 5.2. Folder dokumentacja jest w tym opisie nieistotny, ponieważ nie przechowuje on żadnego oprogramowania.



Rysunek 5.2: Struktura folderów

Foldery przechowują następujące moduły:

- `admin-app` przechowuje aplikację przeglądarkową administratora,
- `voter-app` przechowuje aplikację przeglądarkową użytkownika,
- `e-voting-api` przechowuje aplikację serwerową dostarczającą API systemu,
- `blockchain-server` przechowuje program, który realizuje działanie węzła w sieci Blockchain.

5.3 Uruchamianie modułów

Za pomocą menadżera pakietów NPM uruchamiane są poszczególne moduły systemu. Aby uruchomić dany moduł należy otworzyć katalog z tym modułem. W

katalogu znajduje się plik *package.json*, który zawiera konfigurację i skrypty uruchamiające program. Programy uruchamia się za pomocą poleceń dostarczanych przez pakiet NPM, które należy wpisać w konsoli. Konsola musi być uruchomiona w folderze z plikiem *package.json*.

Moduły *admin-app* i *voter-app* uruchamiane są za pomocą polecenia *npm-start*. Aplikacja *admin-app* w środowisku lokalnym uruchamiana jest pod adresem `http://localhost:4000`, a aplikacja *voter-app* pod adresem `http://localhost:3000`.

Podczas uruchamiania aplikacji serwerowych, należy określić port, na którym widoczny będzie określony serwer.

Aplikacja serwerowa *e-voting-api* jest domyślnie uruchamiana na porcie 8080, jednak można to zmienić za pomocą następującego polecenia:

```
$env:PORT=NUMER_PORTU; npm start.
```

Użycie tego polecenia jest obligatoryjne podczas uruchamiania węzła sieci Blockchain, ponieważ węzeł nie uruchamia się na domyślnym porcie. W celu uruchomienia węzła należy wybrać dowolny nieużywany port komputera.

5.4 Aplikacja administratora

5.4.1 Logowanie

5.4.2 Konfiguracja sieci Blockchain

5.4.3 Konfiguracja wyborów

5.4.4 Nadzorowanie wyborów

5.4.5 Ogłaszanie wyników

5.5 Aplikacja użytkownika

5.5.1 Logowanie

5.5.2 Instrukcja obsługi

5.5.3 Głosowanie

5.5.4 Wyniki

Rozdział 6

Specyfikacja wewnętrzna

W tym rozdziale umieszczono wszystkie informacje dotyczące architektury systemu, wykorzystanych struktur danych, algorytmów i wykorzystania wzorców projektowych.

6.1 Architektura systemu

Podrozdział architektura systemu zawiera przegląd komponentów systemu i pozwala nakreślić idee projektu. Podrozdział podzielono na dwie części. Pierwsza część dotyczy całego systemu, natomiast druga opisuje charakterystykę sieci Blockchain.

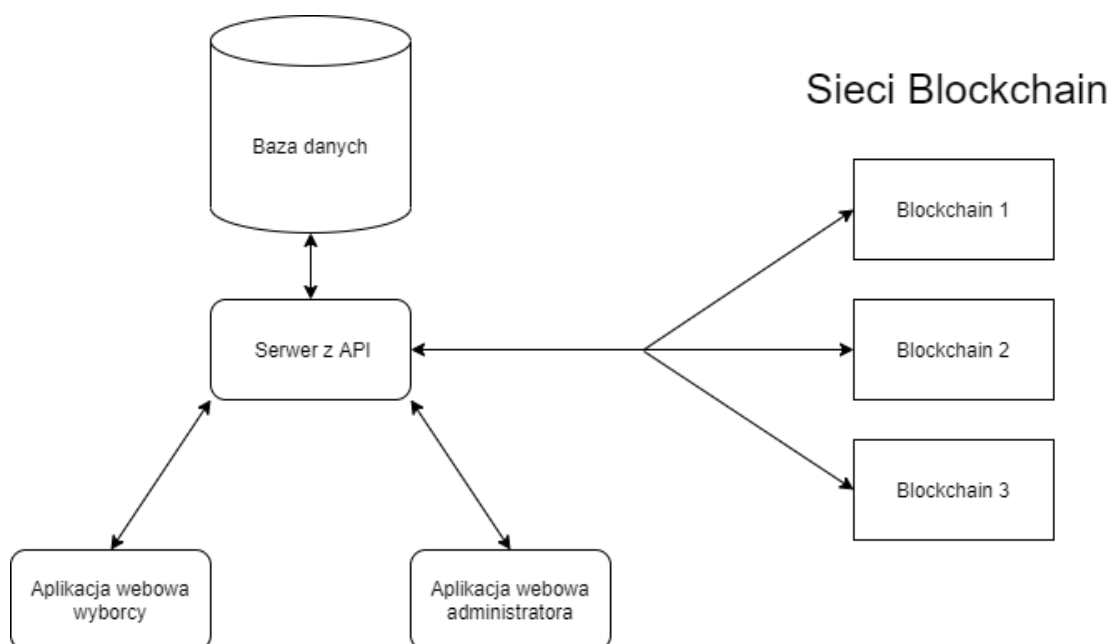
6.1.1 Architektura pełnego systemu

System składa się z pięciu głównych komponentów:

- Aplikacja wyborcy - pozwala na dostęp do zasobów wyborcy,
- Aplikacja administratora - umożliwia konfigurację wyborów, audytowanie wyborów, zarządzanie sieciami Blockchain i zarządzanie bazą danych,
- Serwer z API - serwer, na którym uruchomione jest API. API dostarcza odpowiednie zasoby określonym użytkownikom i posiada dostęp do bazy danych,
- Baza danych - przechowuje informacje dotyczące konfiguracji wyborów. Listy użytkowników, wyborów, kandydatów i określa relacje pomiędzy nimi. W bazie danych znajdują się również hasła użytkowników,
- Sieć Blockchain - odpowiada za przechowywanie głosów i udostępnianie ich na życzenie administratora. Blockchain pełni również funkcje zabezpieczenia głosów.

Komponenty są przedstawione na rysunku 5.1.

Taki podział systemu umożliwia wykorzystanie sieci Blockchain. Proces przeprowadzania wyborów jest zcentralizowany, a sieć Blockchain, która odpowiada za zapisanie wyników wyborów jest rozproszona.



Rysunek 6.1: Schemat przedstawiający komponenty systemu.

6.1.2 Architektura sieci Blockchain

Sieć Blockchain to grupa serwerów, które wymieniają się informacjami. Komunikacja głównie odbywa się w celu przesłania zawartości nowego bloku, synchronizacji danych i dodania nowego węzła do sieci.

6.2 Struktury danych

W systemie można wyróżnić dwie struktury danych, które w znaczny sposób wpływają na zachowanie systemu. W tym podrozdziale omówiono implementację i zastosowania tych struktur danych.

6.2.1 Łańcuch bloków

Za dostarczenie odpowiedniej struktury łańcucha bloków odpowiedzialne są klasy `Block` i `ChainRepository`. Listing 5.1 przedstawia klasę `Block`, która jest reprezentacją pojedynczego bloku. Każdy blok jest odpowiedzialny za przechowywanie głosu wyborczego. Głos jest przechowywany w polu `data`. Pole `timestamp` przechowuje datę utworzenia bloku w łańcuchu. W łańcuchu bloków, każdy blok posiada odniesienie do hasza poprzedniego bloku, w tej implementacji zajmuje się tym pole `prevHash`. Klasa `Block` posiada również pole `nonce`, którego zastosowanie zostało omówione w rozdziale dotyczącym algorytmów (5.3.1 Proof of Work).

```
1 class Block {
2     index: number
3     timestamp: string
4     nonce: number
5     prevHash: string
6     data: string
7
8     constructor (index: number,
9         timestamp: string,
10        nonce: number,
11        prevHash: string,
12        data: string) {
13         this.index = index
14         this.timestamp = timestamp
15         this.nonce = nonce
16         this.prevHash = prevHash
17         this.data = data
18     }
19 }
```

Listing 6.1: Klasa `Block`.

Klasa `ChainRepository` (przedstawiona w listingu 5.2) odpowiada za przechowywanie łańcucha bloków i jego udostępnianie. Aby zapewnić spójność danych, ułatwienie dostępu i ograniczenie zużycia pamięci, w programie istnieje tylko jedna instancja tej klasy. W tym celu wykorzystano wzorzec projektowy Singleton. Z punktu widzenia łańcucha bloków istotne jest pole `chain`, które w tablicy przechowuje bloki. Klasa udostępnia metodę `addBlock`, która pozwala na dodawanie nowego bloku na początek tablicy, zgodnie z regułą tworzenia łańcucha bloków. Dodatkowo klasa posiada prywatną metodę `createGenesisBlock`, która tworzy pierwszy blok łańcucha, którego zawartość nie przechowuje istotnych danych. Metoda wywoływana jest podczas tworzenia instancji `ChainRepository`.

```
1 class ChainRepository {
2     public static instance: ChainRepository
3     private chain: Block[]
4
5     private constructor () {
```

```

6   this.chain = new Array<Block>()
7   this.createGenesisBlock()
8 }
9
10 public static getInstance (): ChainRepository {
11     if (!ChainRepository.instance) {
12         ChainRepository.instance = new ChainRepository()
13     }
14     return ChainRepository.instance
15 }
16
17 public addBlock (block: Block) {
18     this.chain.push(block)
19 }
20
21 private createGenesisBlock () {
22     const block
23     = new Block(this.chain.length, '0', 0, '0', 'genesis')
24
25     this.chain.push(block)
26 }
27 /* ... Getters and setters */
28 }

```

Listing 6.2: Klasa ChainRepository.

Za zapewnienie poprawnego utworzenia nowego bloku odpowiedzialna jest metoda *createNewBlock* (Listing 5.3), która należy do klasy *Blockchain*. Metoda przyjmuje następujące parametry: *nonce*, *prevBlock* i *data*. Z parametru *prevBlock* wyliczany jest hash, za pomocą metody *getHash*. Jest to hash poprzedniego bloku, który przypisywany jest do pola *prevHash* tworzonego bloku.

```

1 abstract class Blockchain {
2     protected blockchain: ChainRepository
3
4     public createNewBlock (nonce: number, prevBlock: Block,
5         data: string): Block {
6         const blockchain = this.blockchain.getChain()
7         const chainLength = blockchain.length
8         const prevHash = this.getHash(prevBlock)
9         const date = Date.now().toString()
10
11         const block = new Block(chainLength, date, nonce,
12             prevHash, data)
13
14         return block
15     }
16 }

```

```

17 protected getHash (block: Block): string {
18     const inputWord = block.toString()
19     const hash = SHA256(inputWord).toString()
20     return hash
21 }
22 /* ... */
23 }

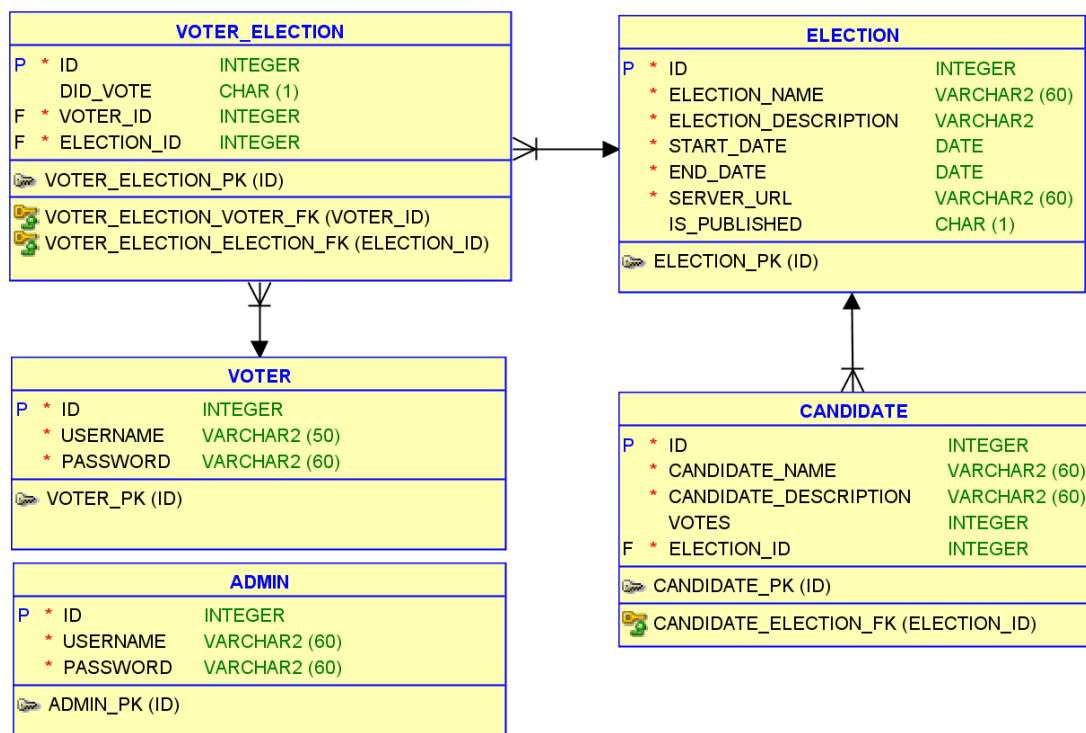
```

Listing 6.3: Fragment klasy Blockchain.

Tak skonstruowany blok jest gotowy do dodania do łańcucha. Dodawanie bloku jest realizowane za pomocą metody *addBlock*, z klasy *ChainRepository*.

6.2.2 Baza danych

W projekcie wykorzystano relacyjną bazę danych. Taka struktura danych pozwala na uporządkowanie danych w relacje pomiędzy tabelami. Rysunek 5.2 przedstawia schemat relacyjnego modelu wykorzystanej bazy danych.



Rysunek 6.2: Schemat przedstawiający relacyjny schemat bazy danych.

Utworzona baza danych dzieli się na część odpowiadającą za przeprowadzenie wyborów oraz bank haseł i ról użytkowników. Tabele *ADMIN* i *VOTER* odpowiadają aktorom systemu. Na podstawie tych tabel można dokonać autoryzacji użytkownika systemu i przydzielić mu odpowiednią rolę, która zapewnia dostęp do zasobów. Dodatkowo tabela *VOTER* jest połączona relacją wiele do wielu z tabelą *ELECTION*. W wyniku relacji wiele do wielu powstała tabela łącznikowa

VOTER_ELECTION, która pozwala podzielić relację wiele do wielu, na dwie relacje jeden do wielu. Tabela *VOTER_ELECTION* pozwala na przypisanie odpowiedniego wyborcy do odpowiednich wyborów. Dodatkowo tabela przechowuje informację o oddaniu głosu przez wyborcę, w polu *DID_VOTE*.

Pozostała część tabel odpowiada za przechowywanie danych dotyczących bezpośrednio wyborów. Zadaniem tabeli *ELECTION* jest składowanie danych dotyczących konfiguracji wyborów. Konfiguracji mogą podlegać:

- *ELECTION_NAME* - tytuł wyborów,
- *ELECTION_DESCRIPTION* - opis dotyczący wyborów,
- *START_DATE* - data rozpoczęcia wyborów,
- *END_DATE* - data zakończenia wyborów,
- *SERVER_URL* - adres zewnętrznego serwera sieci Blockchain, na który wysłane są głosy wyborców,
- kandydaci - zbiór kandydatów, którzy biorą udział w wyborach, jest reprezentowany jako relacja jeden do wielu z tabelą *CANDIDATE*.

Dodatkowo w tabeli znajduje się również pole *IS_PUBLISHED* typu *Boolean*, które odpowiada za oznaczenie stanu wyborów jako opublikowane lub nieopublikowane. System wykorzystuje tę informację do zakończenia wyborów i udostępnienia ich wyników.

Tabela *CANDIDATE* reprezentuje kandydata, którego dane i opis definiują pola *CANDIDATE_NAME* i *CANDIDATE_DESCRIPTION*. Tabela posiada również pole *VOTES*, które po zakończeniu wyborów uzupełniane jest o liczbę zdobytych głosów, przez danego kandydata.

6.3 Algorytmy

W podrozdziale algorytmy opisano implementacje algorytmów utrzymania konsensusu w sieci Blockchain. Węzeł sieci Blockchain może pracować w dwóch algorytmach konsensu: Proof of Work lub Proof of Authority. Każdy węzeł sieci Blockchain pozwala na wybranie algorytmu konsensusu, w którym będzie pracował. Algorytmy różnią się sposobem dodawania nowego bloku i synchronizacją sieci. Wybór algorytmu jest możliwy poprzez wykorzystanie wzorca projektowego strategia. Implementacja została opisana w rozdziale 5.4.

6.3.1 Proof of Work

Dodawanie nowego bloku w algorytmie Proof of Work poprzedzone jest rozwiązaniem zagadki matematycznej. W listingu 5.4 przedstawiono metodę *validateNonce*, która zajmuje się weryfikacją rozwiązania zagadki.

```
1 private validateNonce (  
2     lastNonce: number,  
3     nonce: number,  
4     prevHash: string  
5 ): boolean {  
6     const attempt = `${lastNonce}${nonce}${prevHash}`  
7     const hash = SHA256(attempt).toString()  
8  
9     const isValid = hash.startsWith('000')  
10    return isValid  
11 }
```

Listing 6.4: Metoda *validateNonce*.

Rozwiązaniem zagadki matematycznej jest znalezienie odpowiedniej liczby (*nonce*), która po dołączeniu do określonego stałego ciągu znaków i zahashowaniu go, daje ciąg zer na początku. W przypadku tej implementacji są to trzy zera.

Metoda *solveNonce* (listing 5.5) zajmuje się znalezieniem liczby *nonce*. W pętli wywołana jest metoda *validateNonce*, do momentu rozwiązania zagadki. Wraz z każdą iteracją pętli wykonywana jest inkrementacja zmiennej *nonce*.

```
1 private solveNonce (lastNonce: number, prevHash: string) {  
2     let nonce = 0  
3  
4     while (!this.validateNonce(lastNonce, nonce, prevHash)) {  
5         nonce++  
6     }  
7     return nonce  
8 }
```

Listing 6.5: Metoda *solveNonce*.

Dodawanie nowego bloku jest przeprowadzane w metodzie *mine*. Metodę przedstawiono w listingu 5.6.

```
1 public mine (data: any) {  
2     const blockchain = this.blockchain.getChain()  
3     const lastBlock = blockchain[blockchain.length - 1]  
4     const prevHash = this.getHash(lastBlock)  
5     const nonce = this.solveNonce(lastBlock.nonce, prevHash)  
6  
7     const newBlock = this.createNewBlock(nonce, lastBlock, data)  
8     this.blockchain.addBlock(newBlock)  
9 }
```

Listing 6.6: Dodawanie nowego bloku. Metoda *mine*.

Przed wywołaniem metody *createNewBlock* wywoływana jest metoda *solveNonce*, która rozwiązuje zagadkę matematyczną. Następnie zwracany jest wynik zagadki i tworzony jest nowy blok. Nowy blok jest dodawany do łańcucha.

Synchronizacja łańcucha odbywa się poprzez porównanie jego długości z innym łańcuchem. Listing 5.7 przedstawia metodę *synchronize*, która przyjmuje jako argument długość porównywanego łańcucha znaków. Następnie porównuje ją z łańcuchem bloków węzła.

```
1 public synchronize (syncValue: number): boolean {
2   if (this.blockchain.getChain().length < syncValue) {
3     return true
4   } else {
5     return false
6   }
7 }
```

Listing 6.7: Metoda *synchronize*.

Jeżeli metoda *synchronize* zwróci prawdę, to obecny łańcuch jest podmieniany, na łańcuch przysłany do synchronizacji.

6.3.2 Proof of Authority

Algorytm Proof of Authority pozwala zwiększyć przepustowość sieci Blockchain w porównaniu z algorytmem Proof of Work. Ten algorytm nie wymaga rozwiązywania żadnej zagadki logicznej, dlatego dodawanie nowego bloku sprowadza się tylko do umieszczenia nowego bloku na końcu tablicy. Proces dodawania nowego bloku przedstawia listing 5.8. Klasa odpowiedzialna za algorytm Proof of Authority posiada dodatkowe pole - *authorityFactor*. Jest to współczynnik określający zaufanie danego węzła w sieci. Im wyższy współczynnik, tym wyższe zaufanie. Zaufanie jest zwiększane, wraz z dodawaniem nowych bloków.

```
1 private authorityFactor: number
2
3 public mine (data: any) {
4   const blockchain = this.blockchain.getChain()
5   const lastBlock = blockchain[blockchain.length - 1]
6   const newBlock = this.createNewBlock(0, lastBlock, data)
7   this.blockchain.addBlock(newBlock)
8   this.authorityFactor++
9 }
```

Listing 6.8: Dodawanie nowego bloku w algorytmie Proof of Authority.

W przypadku algorytmu Proof of Authority synchronizacja łańcucha dokonywana jest poprzez porównanie wartości *authorityFactor* dwóch węzłów. Listing 5.9 przedstawia ten proces.

```
1 public synchronize (auth: number) {
2   if (this.authorityFactor < auth) {
3     return true
4   } else {
5     return false
```

```
6     }  
7 }
```

Listing 6.9: Synchronizacja w algorytmie Proof of Authority.

Podmiana węzłów odbywa się analogicznie, jak dla algorytmu Proof of Work.

6.4 Zastosowane wzorce projektowe

W projekcie zastosowano kilka wzorców projektowych dotyczących zarówno programowania obiektowego, jak i podejścia funkcyjnego towarzyszącego ReactJs. W tym rozdziale przedstawiono przykłady implementacji tych wzorców w projekcie.

6.4.1 Singleton

Najczęściej wykorzystywany wzorec projektowy w aplikacjach serwerowych. Jego zastosowaniem było bezpośrednie przechowywanie danych lub utrzymywanie połączenia i dostępu do bazy danych. W Listingu 5.10 przedstawiono klasę *Database*, która jest singletonem. Klasa zajmuje się udostępnianiem połączenia z bazą danych, za pośrednictwem *PrismaClient*. Prywatny konstruktor otwiera połączenie z bazą danych tylko raz, a statyczna metoda *getInstance* zwraca instancję klasy *Database*. Publiczna metoda *getDatabase* zwraca obiekt *PrismaClient*, który umożliwia dostęp do bazy danych.

```
1 class Database {  
2     private static instance: Database;  
3     private db: PrismaClient;  
4  
5     private constructor() {  
6         this.db = new PrismaClient();  
7     }  
8  
9     public static getInstance(): Database {  
10         if (!Database.instance) {  
11             Database.instance = new Database();  
12         }  
13  
14         return Database.instance;  
15     }  
16  
17     public getDatabase(): PrismaClient {  
18         return this.db;  
19     }  
20 }
```

Listing 6.10: Klasa *Database*. Singleton.

6.4.2 Repozytorium

Repozytorium pozwala na udzielenie dostępu do danych warstwom biznesowym aplikacji. Repozytorium posiada wszelkie metody wyszukiujące i zwracające żądane obiekty lub kolekcje obiektów. Listing 5.11 przedstawia klasę *CandidateRepository*. Jest to jedno z wielu repozytoriów, które dostarcza podstawowe metody służące do wyszukiwania i zapisywania danych. Klasa posiada pole *db*, które zawiera obiekt udzielający dostępu do bazy danych.

```
1 class CandidateRepository {
2     private db: PrismaClient
3
4     constructor () {
5         this.db = Database.getInstance().getDatabase()
6     }
7
8     public async findAll () { /* Implementacja*/ }
9
10    public async findByName (candidateName: string) {
11        /* Implementacja*/
12    }
13
14    public async findByElectionId (electionId: number) {
15        /* Implementacja*/
16    }
17
18    public async save (candidate: CandidateDTO, election: Election) {
19        /* Implementacja*/
20    }
21 }
```

Listing 6.11: Klasa *CandidateRepository*.

W projekcie repozytoria mają za zadanie dostarczać obiekty z bazy danych w klasach serwisowych, które zawierają implementację logiki biznesowej.

6.4.3 Strategia

Aplikacja serwerowa węzła sieci Blockchain pozwala na wybór jednego z dwóch algorytmów konsensusu. Algorytmy zajmują się rozwiązaniem tego samego problemu, dlatego można je dowolnie podstawiać. Wzorec projektowy strategia pozwala na dokonanie takiego podstawienia, bez konieczności zmiany struktury programu. Listing 5.12 przedstawia klasę bazową *Blockchain*. Jest to klasa abstrakcyjna, ponieważ posiada implementację metod uniwersalnych dla każdego algorytmu. Metody *mine*, *getSyncValue* i *synchronize* są abstrakcyjne.

```
1 abstract class Blockchain {
2     protected blockchain: ChainRepository
3 }
```



```
4   constructor () {
5       this.blockchain = ChainRepository.getInstance()
6   }
7
8   abstract mine (data: any): void
9   abstract getSyncValue (): number
10  abstract synchronize (syncValue: number): boolean
11
12  public createNewBlock (nonce: number, prevBlock: Block,
13  data: any): Block {
14      const blockchain = this.blockchain.getChain()
15      const chainLength = blockchain.length
16      const prevHash = this.getHash(prevBlock)
17      const date = Date.now().toString()
18
19      const block = new Block(chainLength, date, nonce,
20  prevHash, data)
21
22      return block
23  }
24
25  public setChain (chain: Block[]) {
26      this.blockchain.setChain(chain)
27  }
28
29  public getScore (): Block[] {
30      return this.blockchain.getChain()
31  }
32
33  protected getHash (block: Block): string {
34      const inputWord = block.toString()
35      const hash = SHA256(inputWord).toString()
36      return hash
37  }
38 }
```

Listing 6.12: Klasa *Blockchain*.

Implementacja metod zależy od wykorzystywanego algorytmu. Implementacje znajdują się w klasach *PoWAlgorithm* i *PoAAlgorithm*. Klasy rozszerzają klasę *Blockchain*. Szczegółowa implementacja metod abstrakcyjnych została omówiona w podrozdziale 5.3. Wykorzystanie mechanizmu dziedziczenia pozwala na podstawienie w miejsce klasy *Blockchain* dowolną klasę potomną, która posiada zdefiniowaną implementację metod odpowiedzialnych za realizację algorytmu.

6.4.4 Redux

Redux to specjalna architektura wykorzystywana w aplikacjach ReactJs. W projekcie wykorzystano bibliotekę Redux’a, o nazwie Redux-Toolkit, która upraszcza niektóre elementy Reduxa. Architektura pozwala na przeniesienie stanu aplikacji do jednego magazynu, który jest dostępny globalnie. Listingi 5.13 i 5.14 przedstawiają jeden z wycinków magazynu. Wycinek (ang. slice) to oznaczony odpowiednim identyfikatorem fragment magazynu. Każdy wycinek przechowuje określony stan (listing 5.13).

```
1 const initialState = {  
2   electionsList: [],  
3   candidatesList: [],  
4   selectedElection: '',  
5   selectedVote: '',  
6   isCandidatesShown: false,  
7   isResultsShown: false,  
8   electionResults: []  
9 }
```

Listing 6.13: Stan początkowy wycinka *electionsSlice*.

Wycinek posiada zestaw metod, które pozwalają na modyfikowanie stanu wycinka. W kodzie są to metody obiektu *reducers*, przedstawione w listingu 5.14.

```
1 const electionsSlice = createSlice({  
2   name: 'elections',  
3   initialState,  
4   reducers: {  
5     setElectionsList (state, action) {  
6       state.electionsList = action.payload  
7     },  
8     selectElection (state, action) {  
9       state.selectedElection = action.payload  
10    },  
11    setCandidates (state, action) {  
12      state.candidatesList = action.payload  
13    },  
14    setElectionResults (state, action) {  
15      state.electionResults = action.payload  
16    },  
17    showCandidates (state) {  
18      state.isCandidatesShown = true  
19      state.isResultsShown = false  
20    },  
21    showElections (state) {  
22      state.isCandidatesShown = false  
23      state.isResultsShown = false  
24    },
```

```
25   showResults (state) {
26     state.isCandidatesShown = false
27     state.isResultsShown = true
28   },
29   selectVote (state, action) {
30     state.selectedVote = action.payload
31   }
32 }
33 })
```

Listing 6.14: Wycinek *electionsSlice*.

Tak stworzony kontener ułatwia zarządzanie stanem aplikacji, ponieważ pozwala na odseparowanie implementacji stanu, od kodu odpowiedzialnego za użycie stanu. Architektura Redux posiada również specjalny mechanizm do wykonywania zadań asynchronicznych, takich jak np. pobranie danych z API. Takie zadania nazywane są Thunk'ami. Listing 5.15 przedstawia Thunk'a, który pobiera listę wyborów dostępnych dla określonego użytkownika.

```
1 export const fetchElections = (): AppThunk => async dispatch => {
2   try {
3     const username = localStorage.getItem('username')
4     const response = await axios.get(`/elections/${username}`, {
5       headers: {
6         role: localStorage.getItem('role'),
7         authorization: `Bearer ${localStorage.getItem('token')}`
8       }
9     })
10    const electionList = response.data
11
12    dispatch(setElectionsList(electionList))
13  } catch (error) {
14
15  }
16 }
```

Listing 6.15: Thunk *fetchElection*.

W Thunk'ach można wykorzystywać reducer'y, a następnie wywoływać Thunk'i w komponentach. Przykład znajduje się w listingu 5.16. Wywołanie każdego reducer'a zawsze znajduje się wewnątrz funkcji dispatch, która wysyła reducer do punktu obsługi reducer'ów.

```
1 function ElectionsPage () {
2   const dispatch = useDispatch()
3   const { electionsList, isCandidatesShown, isResultsShown }
4   = useSelector(
5     (state: RootState) => state.elections
6   )
```

```
7  
8   useEffect(() => {  
9     dispatch(fetchElections())  
10  }, [])  
11  
12  /* ... */  
13 }
```

Listing 6.16: Wykorzystanie Thunk’a, w komponencie *ElectionsPage*.

Rozdział 7

Testowanie

System został przetestowany zarówno w środowisku lokalnym jak i internetowym. Aplikacja przeglądarkowa wyborcy została przetestowana na komputerach i *smartfonach*.

Bibliografia

- [1] Ankit Patel: *English T_EX*, https://medium.com/@ankit_233/history-of-the-blockchain-1991-38d6d4c3420c
- [2] Academy Binance: Czym jest Bitcoin? *Polski T_EX*, <https://academy.binance.com/pl/articles/what-is-bitcoin#what-is-bitcoin>
- [3] Vadapalli, Ravindhar: *English T_EX*, BLOCKCHAIN FUNDAMENTALS TEXT BOOK Fundamentals of Blockchain s. 10
- [4] Vadapalli, Ravindhar: *English T_EX*, BLOCKCHAIN FUNDAMENTALS TEXT BOOK Fundamentals of Blockchain s. 19
- [5] Academy Binance: Jaka jest różnica między Blockchainem a Bitcoinem? *English T_EX*, <https://academy.binance.com/pl/articles/difference-between-blockchain-and-bitcoin>
- [6] ZESZYTY NAUKOWE AKADEMII MARYNARKI WOJENNEJ ROK LIV NR 2 (193) 2013 Przemysław Rodwald : KRYPTOGRAFICZNE FUNKCJE SKRÓTU *English T_EX*, s. 92
- [7] Andrew Tar: Proof-of-Work, Explained. 2018. *English T_EX*, <https://cointelegraph.com/explained/proof-of-work-explained>
- [8] Bitcoin Mining. 2017. *English T_EX*, <https://powercompare.co.uk/bitcoin/>
- [9] Academy Binance: Węzły (nodes) - czym są i jak działają? *Polski T_EX*, <https://academy.binance.com/pl/articles/what-are-nodes>
- [10] Cambridge Bitcoin Electricity Consumption Index *English T_EX*, <https://www.epe.admin.cam.ac.uk/cambridge-bitcoin-electricity-consumption-index-cbeci>
- [11] Sarwar Sayeed and Hector Marco-Gisbert: Assessing Blockchain Consensus and Security Mechanisms against the 51% Attack *English T_EX* <https://www.mdpi.com/2076-3417/9/9/1788/htm>
- [12] PKO BP wdraża nową wersję trwałego nośnika. To pierwsze takie rozwiązanie w Europie *Polski T_EX* <https://fintek.pl/pko-bp-wdraza-nowa-wersje-trwalego-nosnika-to-pierwsze-takie-rozwiazanie-w-europie/>

-
- [13] BLOCKCHAIN W PKO BANKU POLSKIM *Polski T_EX*
<https://fintech.pkobp.pl/blockchain-w-banku/>
- [14] <https://www.lexlege.pl/ksh/oddzial-3-walne-zgromadzenie/57/> *Polski T_EX*
Kodeks Spółek Handlowych Art. 402. Tryb zwoływania walnego zgromadzenia
- [15] https://www.kdpw.pl/pl/uslugi/Walne_zgromadzenia/Strony/Jak-to-dziala.aspx *Polski T_EX* Walne zgromadzenia. eVoting - Jak to działa?
- [16] <https://academy.binance.com/pl/articles/what-are-smart-contracts> *Polski T_EX*
Co to Smart Kontrakt?
- [17] General Framework of Electronic Voting and Implementation thereof at National Elections in Estonia *English*, rok 2016