Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY

Instytut Elektrotechniki Teoretycznej i Systemów Informacyjno-Pomiarowych

Praca dyplomowa inżynierska

na kierunku Informatyka stosowana

w specjalności Inżynieria Oprogramowania

Wykorzystanie technologii blockchain do podpisywania autentyczności prac dyplomowych

Wojciech Szade

numer albumu 391110

promotor

dr inż. Robert Szmurło

WARSZAWA 2025

Wykorzystanie technologii blockchain do podpisywania autentyczności prac dyplomowych

Streszczenie

Ta praca dyplomowa skupia się na analizie problemu autentyfikacji prawdziwości prac dyplomowych na uczelni i możliwym jego rozwiązaniu przy pomocy systemu opartego o technologię blockchain. Czytelnik pozna obecnie istniejące systemy, które zapewniają wiarygodność danych przy użyciu łańcucha blockchain. Przybliżone zostaną algorytmy i składowe systemu, który implementuje taką funkcjonalność. Przedstawiony zostanie utworzony na potrzeby projektu system, który przechowuje dane o pracach dyplomowych w strukturze blockchain, wraz z pełną implementacją wymiany informacji pomiędzy uczestnikami systemu.

Słowa kluczowe: A, B, C

Using blockchain technology to authenticate thesis works

Abstract

This is abstract. This one is a little too short as it should occupy the whole page.

Keywords: X, Y, Z

Spis treści

1 Wstęp **9**

2 Potrzeby systemu **11**

3 Analiza istniejących rozwiązań **15**

4 Proponowane rozwiązanie **17**

4.1 Słownik pojęć 17

4.2 Założenia projektu 18

4.3 Zarys rozwiązania 18

5 Technologie **19**

6 Architektura systemu **21**

6.1 Moduły systemu 21

6.2 Struktura bloku 22

6.3 Komunikacja w systemie 24

6.4 Protokoły komunikacji 26

7 Podsumowanie **31**

Bibliografia **33**

Wykaz skrótów i symboli **35**

Spis rysunków **37**

Spis tabel **39**

Spis załączników **41**

Rozdział 1

Wstęp

Fałszowanie dokumentów akademickich to poważny problem, który dotyka systemy edukacyjne na całym świecie [5]. Wzrost dostępności zaawansowanych technologii edycji i druku sprawia, że tworzenie fałszywych dyplomów i certyfikatów staje się coraz łatwiejsze. Brak łatwych metod sprawdzenia prawdziwości przedstawianych dokumentów, sprawia, że krok jest ten zupełnie pomijany. W odpowiedzi na to rosnące zagrożenie, instytucje edukacyjne i pracodawcy poszukują nowoczesnych rozwiązań, które zapewnią wiarygodne potwierdzenie autentyczności dokumentów.

Jedyne rozwiązania oferowane przez uczelnie w zakresie autentykacji prac dyplomowych w momencie publikowania pracy opierają się o udostępnianie informacji o pracach w formie repozytorium. Systemy te nie skupiają się na możliwości łatwej walidacji prawdziwości pracy i nie są dostosowane do takiego zastosowania. W dodatku systemy te nie są ujednolicone pomiędzy uczelniami, co utrudnia korzystanie z nich. Uczelnie nie gwarantują wiecznego utrzymywania tych systemów, a dane mogą zostać utracone na przestrzeni lat lub dostęp do nich może być znacząco utrudniony.

W tym kontekście technologia blockchain zyskuje coraz większe zainteresowanie jako potencjalne rozwiązanie. Blockchain oferuje unikalne cechy, które czynią go idealnym narzędziem do weryfikacji autentyczności dokumentów. Decentralizacja blockchaina, eliminująca potrzebę centralnego organu kontrolnego, zwiększa bezpieczeństwo i transparentność systemu, a także gwarantuje jego długowieczność.[4] Niezmienność danych zapisanych w blockchainie gwarantuje ich integralność i autentyczność.

Badania pokazują, że blockchain ma potencjał zrewolucjonizować sposób, w jaki dokumenty akademickie są wydawane i weryfikowane [12]. Wdrożenie systemu opartego na blockchainie może przynieść wiele korzyści: zwiększenie bezpieczeństwa i zaufania do dokumentów, uproszczenie procesu weryfikacji, zmniejszenie kosztów administracyjnych oraz stworzenie globalnie dostępnego systemu weryfikacji.

Takie rozwiązanie musiałoby jednak rozwiązać problemy z którymi boryka się blockchain - wysokie zużycie energii[6], czy ryzyko przejęcia kontroli nad łańcuchem przez niepożądanych użytkowników[11]. Kwestie te są możliwe do rozwiązania, a sam blockchain, jak i jego kluczowa część - mechanizmy konsensusu bardzo szybko się rozwijają[7].

Ujednolicony system, który zapewniałby niewymagające wsparcia żadnych z uczelni, sposób autentyfikacji pracy dyplomowej w wiarygodny sposób zmieniłby podejście do tej kwestii.

W dalszej części pracy zostaną poruszone i rozwinięte kwestie dokładnych potrzeb i ograniczeń takiego systemu. Czytelnikowi zostanie przybliżone pojęcie blockchaina i części z których się składa, zostanie przeprowadzona analiza zalet i wad zastosowania go dla naszego problemu. Autor przeprowadzi porównanie obecnych mechanizmów konsensusu i wskaże te, które nadają się do zastosowania w tym projekcie. Opisane zostaną istniejące rozwiązania stosujące blockchain w podobnych zastosowaniach.

Po zrozumieniu i opisaniu tych zagadnień Autor wskaże rozwiązania i technologie, których użyje do stworzenie rozwiązania opisywanego problemu i dokona ich implementacji, skupiając się w pracy na przedstawienie Czytelnikowi architektury i metody działania powstałego systemu.

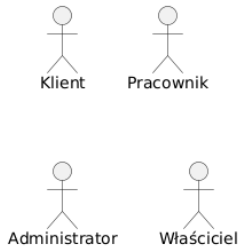
System który powstanie w ten sposób, będzie miał szansę służyć jako drogowskaz dla Uczelni wyższych, które we współpracy będą mogły utworzyć oparty na przedstawionym rozwiązaniu ulepszony system.

Rozdział 2

Potrzeby systemu

Po przeanalizowaniu problemu opisanego w rozdziale 1 należy ustalić wymagania systemu opartego o blockchain, który mógłby rozwiązać zadany problem. W tym celu należy w pierwszej kolejności określić rodzaje użytkowników systemu, którzy będą z niego korzystać - aktorów. Kolejnym krokiem będzie przeanalizowanie ich potrzeb.

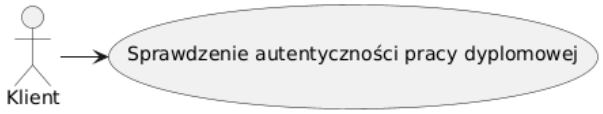
Oczywiście głównym użytkownikiem - dla którego niejako tworzony jest system - jest klient. Jest to każda osoba, która chce sprawdzić autentyczność określonej pracy dyplomowej. Mogą być to zarówno byli absolwenci - autorzy prac dyplomowych, którzy chcą udowodnić bycie twórcą takiego dzieła, jak i pracodawcy lub uczelnie wyższe, które dokonują takiego działania w procesie rekrutacyjnym - pracownika lub studenta.
Kolejnym użytkownikiem takiego systemu jest pracownik uczelni wyższej. To on będzie odpowiedzialny za wprowadzanie danych do systemu.
Nad płynną pracą systemu będzie czuwał administrator. Będzie on odpowiedzialny za wdrożenie systemu i utrzymywanie go. Ponadto, decyzję o wdrożeniu samego systemu będzie podejmować uczelnia wyższa - występować będzie ona w roli właściciela projektu. Z racji na to, że istotne jest wprowadzenie systemu w wersji ujednoliconej, współdziałającej pomiędzy wieloma uczelniami - będzie to więcej niż jedna instytucja. Na potrzeby projektu uznamy jednak, że ich potrzeby będą tożsame. Wymienieni użytkownicy - aktorzy, zostali przedstawieni na rysunku 1.



Rysunek 1. Diagram aktorów

Jedyną interakcją klienta z systemem będzie sprawdzenie autentyczności pracy dyplomowej w systemie. System udostępniający im informację o pracy powinien działać szybko, dostarczać

wystarczającą ilość informacji do potwierdzenia autentyczności danej pracy, a informacje przez niego przekazane powinny być precyzyjny i dokładne - przekazując użytkownikowi jasne komunikaty. Należy zwrócić szczególną uwagę na to, żeby niedoskonałość systemu wprowadziła użytkownika w błąd na temat prawdziwości lub fałszywości danej pracy. Diagram dla tego przypadku użycia został zaprezentowany na diagramie 2.



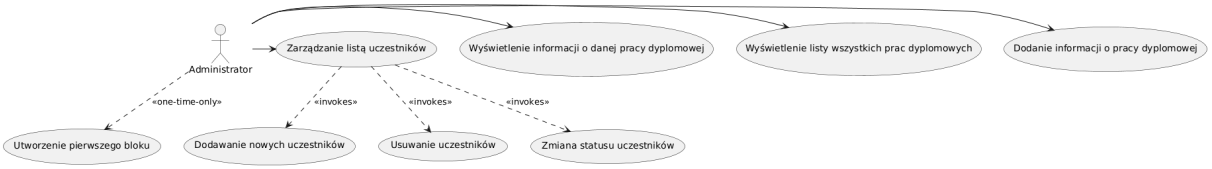
Rysunek 2. Diagram przypadków użycia klienta

Pracownik powinien móc dodawać informacje o nowych pracach dyplomowych do systemu. Tak naprawdę funkcjonalność ta mogłaby być także zautomatyzowana, a dodanie pracy do systemu odbywało by się przy pomocy integracji z innym systemem, który przechowuje takie informacje, ale na potrzeby projektu założymy, że dane te są wprowadzane przez pracownika ręcznie.
Pracownik powinien więc mieć możliwość wprowadzenia danych o nowej pracy i dodania ich do systemu, a także wyświetlenia listy wszystkich prac dyplomowych oraz wyświetlenia szczegółów dla konkretnej pracy. System powinien poinformować pracownika, jeżeli doszło do próby dodania pracy, która już znajduje się w systemie. Ze względu na specyfikę propagacji informacji o nowym bloku w sieci blockchain, operacja dodawania nowej pracy będzie czasochłonna. Żeby nie wprowadzać nadmiarowych opóźnień, należy przyjąć ograniczenie czasu dodawania nowej pracy na 24 godziny. Pracownik powinien móc w tym czasie dodawać kolejne prace. W przypadku niepowodzenia dodania pracy, pracownik i administrator powinni zostać o tym poinformowani. Diagram przypadków użycia pracownika został przedstawiony na rysunku 3.

Rysunek 3. Diagram przypadków użycia pracownika

Administrator będzie nadzorował pracę całego systemu. Tak jak pracownik, powinien on przeglądać i wyświetlać prace dyplomowe. W przeciwieństwie do wielu systemów - nie będzie miał on jednak możliwości jakiejkolwiek edycji dodanej już pracy dyplomowej. Wynika to ze specyfiki blockchainu - blok nie może być w żaden sposób edytowany, na tym opiera się spójność i bezpieczeństwo tej technologii. Administrator odpowiedzialny będzie za zarządzenie listą uczestników - systemów, z którymi komunikować się będzie nasz element zarządzający blockchainem. Będzie on miał możliwość dodawania, usuwania oraz zmiany ich statusu - ponieważ w systemie będą występować uczestnicy autoryzowani oraz nieautoryzowani. Zagadnienie zostanie to bliżej wytłumaczone w kolejnych rozdziałach. Kolejną zaawansowaną funkcjonalnością jest utworzenie przez administratora pierwszego

bloku w systemie. Czynność ta jest dokonywana tylko raz, w momencie wdrażania systemu. Te funkcjonalności są kluczowe dla podstawowego bezpieczeństwa systemu, ponieważ umożliwiają na wykluczenie nieprawidłowych uczestników, którzy mogliby dodawać fałszywe informacje do łańcucha. Opisane przypadki użycia zostały przedstawione na rysunku 4.



Rysunek 4. Diagram przypadków użycia administratora

Właściciel systemu (uczelnia wyższa) też ma wymagania dotyczące systemu. Jednym z kluczowych wymagań jest bezpieczeństwo systemu - nie może być on podatny na ataki z zewnątrz, próby dodania fałszywych danych, czy podszywania się pod jedną z uczelni. Możliwe powinno być zintegrowanie tego rozwiązania z obecnymi systemami stosowanymi na uczelni - tak, żeby rozpoczęcie korzystania z systemu było stosunkowo proste i niewymagające nadmiarów pracy. System nie powinien korzystać też z nieproporcjonalnej ilości zasobów - żeby nie powodować nadmiarowego zużycia prądu, ani nie wymagać niepotrzebnie wydajnych maszyn do działania.

Rozdział 3

Analiza istniejących rozwiązań

Należy zauważyć, że technologia blockchain jest wszechstronna i może być dostosowywana do różnorodnych zastosowań, a systemy oparte na niej, nawet te służące podobnym celom, mogą znacznie różnić się pod względem zalet i wad.
Najpopularniejszym systemem opartym o blockchain jest Bitcoin, który jest kryptowalutą opartą na rozproszonej księdze rachunkowej, umożliwiającą anonimowe płatności bez udziału rządów i banków [13]. Bitcoin, w przeciwieństwie do wielu innych blockchainów, wykorzystuje mechanizm konsensusu *Proof-of-Work* i skupia się na działaniu jako *system płatności*, a nie jako uniwersalna platforma dla aplikacji [8]. Jego idea polega na zapisywaniu w środku bloku transakcji dokonywanych z jego użyciem - dlatego nazywa się go cyfrowym rejestrem, księgą. Jest to już dosyć stara kryptowaluta, a jej mechanizm konsensusu powoduje nadmiarowe zużycie prądu [6], przez co często jest krytykowany - nowe popularne kryptowaluty takie jak *Etherum* rozwiązują ten problem zmieniając mechanizm konsensusu na bardziej efektywny.

Systemem, który implementuje podobną idee, co nasz system korzystając przy tym z blockchainu jest BigchainDB 2.0.
Jest to oprogramowanie, które łączy w sobie cechy blockchainu, takie jak: decentralizacja, niezmienność i aktywa kontrolowane przez właściciela, z cechami bazy danych, takimi jak: wysoka szybkość transakcji, niskie opóźnienia oraz możliwość indeksowania i przeszukiwania uporządkowanych danych. Głównym celem BigchainDB 2.0 było stworzenie systemu, który łączyłby zalety dwóch technologii - bazy danych i blockchainu, jednocześnie eliminując ich wady. W poprzednich wersjach istniały problemy z odpornością na awarie oraz scentralizowanym zarządzaniem zapisem danych, gdzie jeden węzeł pełnił rolę węzła głównego - czyli system nie był zdecentralizowany [1]. Wersja 2.0 projektu wyeliminowała te problemy. Głównymi cechami systemu w obecnej wersji jest:

- Pełna decentralizację: Każdy węzeł ma swoją lokalną bazę danych *MongoDB*, a komunikacja między węzłami odbywa się za pomocą protokołów *Tendermint*. Dzięki temu system jest odporny na awarie, a nawet jeśli 1/3 węzłów ulegnie awarii, pozostałe węzły będą kontynuować działanie[1].

- Niezmienność danych: Dane przechowywane w sieci BigchainDB nie mogą być zmieniane ani usuwane, a próby ich zmiany są wykrywalne[1]. Każda transakcja jest kryptograficznie podpisana. Są to tak naprawdę założenia każdego łańcucha blockchain.
- Wysoka szybkość transakcji: BigchainDB 2.0 jest zaprojektowany tak, aby obsługiwać dużą liczbę transakcji na sekundę. [1]
- Niskie opóźnienie i szybką finalizacja transakcji: Transakcje są zatwierdzane w ciągu kilku sekund, a po zatwierdzeniu nie mogą być cofnięte. [1]
- Indeksowanie i przeszukiwanie strukturalnych danych: wykorzystanie lokalnych baz umożliwia wykorzystanie zalet *MongoDB* do indeksowania i przeszukiwania danych. [1]
- Odporność na ataki Sybil: Atak Sybil opiera się na przejęciu większości węzłów w sieci, w wyniku czego przejmuje się nad nią kontrolę. W BigchainDB 2.0 węzłem systemu zostają tylko uczestnicy z odpowiednimi uprawnieniami, a nie dowolny komputer - więc jest na nie odporny[1].

BigchainDB 2.0 może być używany w wielu różnych zastosowaniach, w tym w zarządzaniu łańcuchem dostaw logistycznych, ochronie praw własności intelektualnej, zarządzaniu danymi [1]. Dzięki połączeniu cech blockchain i bazy danych, BigchainDB 2.0 stanowi wszechstronne narzędzie do budowy zdecentralizowanych i bezpiecznych systemów.

System ten korzysta z mechanizmu konsensusu *Raft*. Algorytm ten polega na wybieraniu lidera wśród uczestników systemu, który przyjmuje polecenia i zapisuje je w swoim logu. Następnie wysyła on te polecenia do pozostałych uczestników, którzy po zatwierdzeniu ich przez większość uczestników zostają zapisane. Jeżeli lider przestanie być dostępny - uczestnicy wybierają nowego lidera. Lider wybierany jest na podstawie głosowania wszystkich uczestników. [15]

Mimo, że BigchainDB 2.0 jest uniwersalnym systemem z szeregiem zalet nie spełnia on potrzeb naszego systemu, ponieważ brakuje mu kluczowych funkcjonalności, które są istotne dla naszego rozwiązania. System BigchainDB nie pozwala na kontrolę dostępu i eliminację uczestników systemu, ponieważ opiera się on na równorzędnych węzłach z identycznymi uprawnieniami [1]. Nasz system musi w jakiś sposób pozwalać na zarządzanie permisjami uczestników, najlepiej opierając się na wiarygodnych informacjach spoza systemu - wśród uczelni znalezienie takiej metody nie powinno być problematyczne. BigchainDB jest też kompleksowy i złożony - integracja z istniejącymi rozwiązaniami może być skomplikowana i kosztowna. Dostosowanie go do systemów uczelnianych może okazać się zbyt znaczącym problemem. Bigchain DB 2.0 to system, który implementuje wiele dobrych rozwiązań, a jego otwarto źródłowość ułatwia inspirowanie się tym systemem przy tworzeniu własnej implementacji blockchaina. Pomaga też w tym jego dobre udokumentowanie.

Rozdział 4

Proponowane rozwiązanie

W celu rozwiązania zadanego problemu, biorąc pod uwagę przybliżone w rozdziale 2 potrzeby, a także opierając się na rozwiązaniach opisanych w rozdziale 3, zauważając brak istniejącego odpowiedniego systemu, autor pracy zdecydował się utworzyć własne rozwiązanie. Zanim przejdziemy do opisania tego rozwiązania - przybliżmy pojęcia stosowane dalej w pracy i ich znaczenie w kontekście tego projektu.

4.1 Słownik pojęć

- Użytkownik - osoba korzystająca z systemu - zazwyczaj klient lub pracownik.
- Uczestnik - kopia systemu utrzymywana na maszynie, zdolna do komunikacji z innymi uczestnikami.
- Uczestnik autoryzowany - uczestnik, który ma możliwość dodawania bloków do łańcucha - tworzenia nowych prac dyplomowych. W naszym projekcie w formie takich uczestników występują jedynie uczelnie wyższe.
- Uczestnik nieautoryzowany - uczestnik, który nie ma możliwości dodawania bloków do łańcucha - może jedynie przechowywać wersję łańcucha i przekazywać o nim informacje do innych uczestników. W założeniu pracy - każda osoba, może zostać uczestnikiem systemu - jest to warunek zapewnienia projektu niezależności od uczelni.
- Blok - element łańcucha, który zawiera w sobie informacje o jednej pracy dyplomowej - struktura bloku została bliżej przybliżona w podrozdziale 6.2.
- Łańcuch - zbiór bloków, komplet danych o wielu pracach dyplomowych.
- Hasz pracy - identyfikator pracy dyplomowej, a dokładniej bloku w którym jest przechowywana.

4.2 Założenia projektu

Proponowane przez autora rozwiązanie opiera się na kilku poczynionych założeniach, zgodnych z potrzebami systemu.

1. Z systemu korzystać będzie wiele uczelni - dodawać one będą prace dyplomowe do jednego łańcucha blockchain.
2. Dodana do łańcucha praca nie będzie mogła być edytowana, ani usunięta.
3. Jednym z celów systemu jest zapewnienie długowieczności przechowywanych danych, a możliwość ich utrzymywania i udostępniania informacji o prawdziwości prac dyplomowych, powinna móc być utrzymywana nawet, jeżeli projekt utraci oficjalne wsparcie uczelni, poprzez utrzymywanie kopii łańcucha blockchain przez dowolnych użytkowników systemu.
4. Utworzona na potrzeby pracy inżynierskiej wersja rozwiązania nie powinna być traktowana jako pełnoprawna implementacja systemu, a jedynie wartościowy *'Proof of Concept'* - dowód koncepcji, na podstawie którego mogłyby powstać w przyszłości rzeczywiste rozwiązania. Mimo to, projekt utworzony został z największą starannością i z dbałością o detale, skupiając się na dokładnym odwzorowaniu blockchaina.
5. Autor pracy dyplomowej po jej dodaniu do systemu poznaje jej hasz.
6. Udostępnianie danych osobistych przez system jest dozwolone, ze względu na to, że wyświetlenie ich wymaga znajomości haszu pracy dyplomowej, który traktowany jest ze strony prawnej jak klucz dostępowy.

4.3 Zarys rozwiązania

Rozwiązanie dla klienta, zakłada że autor pracy dyplomowej udostępnia osobie, która ma sprawdzić prawdziwość jego pracy, hasz pracy dyplomowej. Osoba ta - klient - wprowadza hasz na odpowiedniej stronie. Jeżeli hasz jest poprawny i w systemie istnieje odpowiednia praca - udostępniane są o niej informacje. Klient może potwierdzić, że tytuł, dziedzina pracy - oraz jej autor są zgodne z przedstawionymi przez jej autora informacjami. Jeżeli został mu udostępniony dokument zawierający samą pracę dyplomową - może on sprawdzić, czy to ten dokument został załączony do pracy dyplomowej - załączając go na stronie pracy dyplomowej, system udostępni informację o zgodności utworzonego hasza.
Rozwiązanie dla pracownika umożliwia mu dodanie nowej pracy do systemu, poprzez wprowadzenia na odpowiednich danych i dodaniu pracy. Po tym jak praca zostanie rzeczywiście dodana - strona przedstawi odpowiednią informację. To jest proces, który może być czasochłonny, ale nie powinien zająć więcej niż kilka minut.

Rozdział 5

Technologie

W ramach realizacji projektu inżynierskiego zdecydowano się wykorzystać język Python, który jest odpowiednio dostosowany do potrzeb zadania. Umożliwia też efektywne rozwiązywanie problemów programistycznych, jednocześnie eliminując potrzebę skupiania się na kwestiach istotnych w niskopoziomowych językach - takich jak dokładne zarządzanie pamięcią. Dla Pythona istnieje też szereg bibliotek wspierających zarówno komunikację P2P, jak i tworzenie REST API, łączenie się z różnymi bazami danych i implementowania blockchaina.

Jako bazę danych do przechowywania bloków wybrane zostało nierelacyjne *MongoDB*. Baza ta przechowuje informacje w formie dokumentów w formacie *JSON*, pogrupowanych w kolekcje. Bazy nierelacyjne są lepsze w przechowywaniu bloków blockchain od tych opartych na SQL [9]. W dodatku mogą one umożliwić w dużo łatwiejszy sposób dodanie możliwości przechowywania tylko części łańcucha na wybranej maszynie. Przy użyciu bazy SQLowej zadanie jest to bardzo trudne. *MongoDB* pozwala na większą swobodę przy tworzeniu struktury bazy danych, jest szybkie i wydajne, a jego elastyczna architektura umożliwia łatwe skalowanie oraz integrację z innymi systemami, co sprawia, że jest idealnym rozwiązaniem dla aplikacji blockchain, gdzie szybki dostęp do danych oraz możliwość przechowywania wybranych fragmentów łańcucha mają kluczowe znaczenie [2]. *MongoDB* jest też używane przez opisywany w rozdziale 3 system BigchainDB 2.0 [1].

Dla API udostępnianego do zarządzania działaniami na łańcuchu i odczytywania z niego danych początkowo rozważano użycie biblioteki *Django*. Oferuje ona dużo wbudowanych rozwiązań ułatwiających tworzenie kompleksowych systemów. Biblioteka ta zawiera chociażby panel administratora umożliwiający zarządzanie danymi, ułatwia podział projektu na moduły oraz wspiera zarządzenie obiektami w sposób *ORM* (*Object Relational Mapping - rodzaj interfejsu stosowanego do implementacji operacji na obiektach w bazie relacyjnej*. Jednak wszystkie te rozwiązania w *Django* zostały stworzone z myślą o relacyjnych bazach danych wykorzystujących *SQL* [3]. Niewspierane jest połączenie w wbudowany sposób z *MongoDB*, które jest nierelacyjną bazą [3]. Wykorzystywanie jej z biblioteką *Django* sprawiałoby, że nie możliwe było by korzystanie z wbudowanych metod zarządzania modelami i bazą, a także z panelu administracyjnego - korzystając z prostego silnika połączenia z bazą, takiego jak *PyMongo*. Możliwe byłoby zastosowanie biblioteki *Djongo*, która integruje *Django*

z bazą *MongoDB* w „niezauważalny" sposób - mapując działania do konkretnych funkcji. Niestety ta biblioteka nie jest aktywnie wspierana i brakuje jej wielu funkcjonalności. Jej założenia zakładają tłumaczenie kwerend *SQL* na takie zrozumiałe dla Mongo - co nie jest optymalnym rozwiązaniem i może skutkować w problemach z wydajnością.
Alternatywnymi bibliotekami od *Django* stworzonymi dla *Pythona*, która także umożliwia tworzenie aplikacji udostępniających API są *Flask* i *FastAPI*. Ich porównanie zostało opisane w tabeli 1 .

Kryterium	**FastAPI**	**Flask**
Wydajność	Wysoka - dzięki natywnemu wsparciu dla programowania asynchronicznego; idealna dla aplikacji o dużej liczbie równoczesnych połączeń.	Niższa - przez domyślą synchroniczność; wsparcie dla asynchroniczności wymaga dodatkowych bibliotek, takich jak *Gevent*.
Dokumentacja API	Automatycznie generuje dokumentację API (*Swagger UI*, *ReDoc*).	Wymaga dodatkowych narzędzi, np. *Flask-RESTX*, do generowania dokumentacji.
Walidacja danych	Wbudowana walidacja dzięki *Pydantic* i podpowiedziom typów.	Brak wbudowanej walidacji; wymaga użycia bibliotek zewnętrznych, takich jak *Marshmallow*.
Przypadki użycia	Idealny dla API o wysokiej wydajności, mikrousług i aplikacji czasu rzeczywistego.	Wszechstronny; odpowiedni dla różnych typów aplikacji webowych.

Tabela 1. Porównanie bibliotek FastAPI i Flask [14]

Biblioteki nie są bardzo odmienne od siebie i obie mogłyby być zastosowane, ale autor zdecydował się skorzystać z *FastAPI*, które jest nowszym rozwiązaniem udostępniającym więcej rozwiązań w sposób wbudowany - bez potrzeby korzystania z kolejnych bibliotek. W dodatku biblioteka ta oferuje imponującą wydajność.

Do połączeń pomiędzy użytkownikami wykorzystana zostanie biblioteka *p2pd*, służąca do obsługi połączeń Peer2Peer. Jej zaletą jest zaawansowany system wyszukiwania połączeń między węzłami, obsługujący różne rodzaje połączeń i wyszukiwania ich. W tej chwili, nie istnieją alternatywne, podobnie rozbudowane biblioteki do zarządzania połączeniami Peer2Peer w Pythonie.

Do kryptograficznych operacji w systemie - podpisywania bloków, tworzenia haszy, szyfrowania i odszyfrowywania danych w systemie została użyta bardzo popularna biblioteka *pycryptodomex*, która nie ma w tej chwili rozsądnych alternatyw.

Jako wcześniej wspomniany silnik bazy danych zostało użyte oficjalnie zalecane przez MongoDB - *PyMongo*. Jest to klasyczna biblioteka umożliwiająca połączenie i wykonywanie operacji na bazie danych Mongo.

Rozdział 6

Architektura systemu

6.1 Moduły systemu

Na rysunku 5 przedstawiony został diagram architektury systemu w uproszczonej formie - skupiający się na modułach systemu. Pokazuje on podział systemu na moduły oraz komunikację pomiędzy nimi.

Komponent *Front-end server* przedstawia moduł systemu, który udostępnia interfejs do obsługi systemu, zarówno poprzez klientów, jak i pracowników i administratora. Jak widać komponent ten komunikuje się z resztą systemu poprzez połączenie z API back-endu, umożliwiając odczyt i wysyłanie danych przez poszczególne end-pointy. Ten komponent dzieli się na pomniejsze komponenty, które zostaną omówione w dalszej części pracy.

Zaplecze systemu składa się z wielu komponentów. Tym głównym, który zarządza pracą reszty jest Blockchain Service. Pracuje on równolegle z komponentem API i wykonuje przekazane przez nie polecenia. Udostępnia on pozostałym modułom dostęp do bazy danych - i innych modułów, poprzez wstrzykiwanie zależności.

Block service to komponent, który odpowiada za wszystkie operacje na blokach. Do jego zadań należy między innymi:

- tworzenie nowych bloków i ich podpisywanie,
- zapisywanie i odczytywanie bloków z bazy danych,
- sprawdzanie autentyczności bloków,
- tworzenie bloku początkowego.

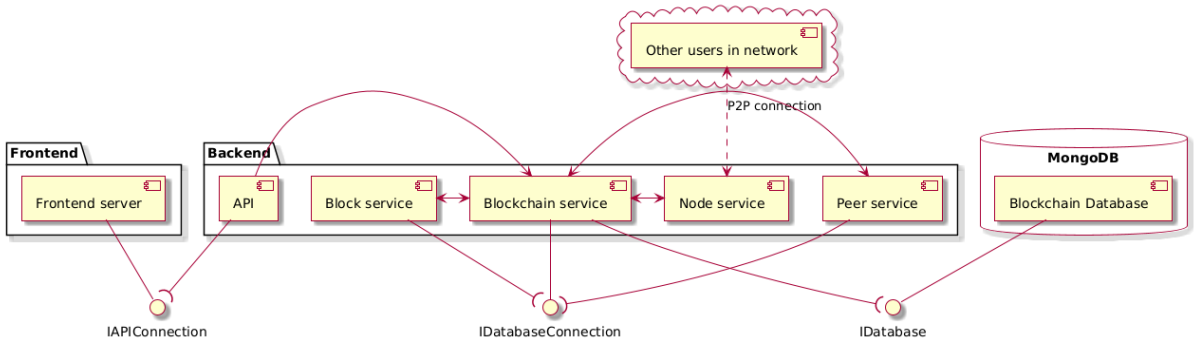
Wykorzystuje on przekazane przez *Blockchain service* połączenie z bazą danych oraz z *Peer service* - do uzyskiwania kluczy do autentyfikacji bloków stworzonych przez inne węzły.

Wspomniany *Peer service* to komponent odpowiadająca za zarządzanie uczestnikami łańcucha. Odpowiada on za zapisywanie nowych uczestników, usuwanie ich, aktualizowanie statusów uczestników, zapis i odczyt danych z bazy. Ten komponent korzysta z połączenia z bazą danych.

Node service to komponent, który zajmuje się komunikacją z innymi uczestnikami systemu. Pozyskuje on informacje zarówno z komponentu zarządzania uczestnikami, jak i blokami.
Jego główne zadanie to ciągłe oczekiwanie na wiadomości od innych uczestników i odpowiadanie na

nie, jak i wysyłanie własnych wiadomości.
Dokładniejszy opis rodzajów wymienianej komunikacji znajduje się w rozdziale 6.4.

Centralnym elementem systemu oczywiście jest nasza nierelacyjna baza danych *MongoDB*. Baza przechowuje bloki z informacjami o pracach dyplomowych. Służy ona także do przechowywania informacji o uczestnikach systemu. W strukturze *MongoDB* takie bloki będą występować jako dokumenty, a należeć będą do kolekcji łańcucha.



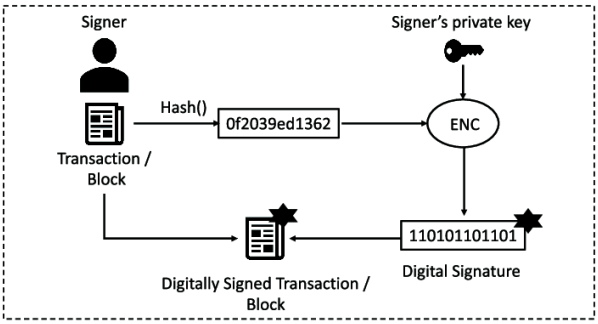
Rysunek 5. Diagram architektury systemu
Źródło: Opracowanie własne

6.2 Struktura bloku

Informacje o pracach dyplomowych będą przechowywane w naszym systemie w formie bloków. Jedna praca dyplomowa przechowywana jest w jednym bloku. Uporządkowane są one w łańcuchu w kolejności dodawania ich, a wszystkie przynależą do tego samego łańcucha. Każdemu blokowi przypisany jest hasz. Jest on wyliczany na podstawie zawartości bloku, czyli informacji, które on przechowuje, takich jak tytuł pracy, czy autorzy, ale też informacji o poprzednim haszu. To, co kluczowe - to że zmiana którejkolwiek wartości w bloku będzie oznaczała zmianę hasza. Każdy blok posiada też podpis. Jest to hasz podpisany przy użyciu prywatnego klucza uczestnika łańcucha. Proces podpisywania bloku został przedstawiony na rysunku 6. Dzięki temu, że zależny od zawartości hasz zostaje dodatkowo podpisany przez prywatny klucz, zagwarantowana zostaje niezmienność bloku. Modyfikacja go oznaczałaby zmianę hasza, co oznaczałoby potrzebę ponownego podpisu - który nie jest możliwy bez znajomości klucza prywatnego. Pozostali uczestnicy walidują podpis na podstawie przekazanego im hasza i posiadanego klucza publicznego autora bloku [10]. Zawarcie w bloku informacji o haszu bloku poprzedzającego gwarantuje ciągłość łańcucha i zapobiega dodaniu niepożądanego bloku do łańcucha - zmiana informacji o bloku poprzedzającym spowodowałaby to samo, co zmiana każdej innej informacji.

Blok musi mieć określoną strukturę określającą zawartość bloku. Zmiana struktury w trakcie funkcjonowania systemu jest możliwa, ale dobrze, jeżeli przewidujemy takie zmiany, wprowadzić pole z informacją o wersji łańcucha. Nasz blok będzie zawierał następujące pola:

- Hasz poprzedniego bloku,



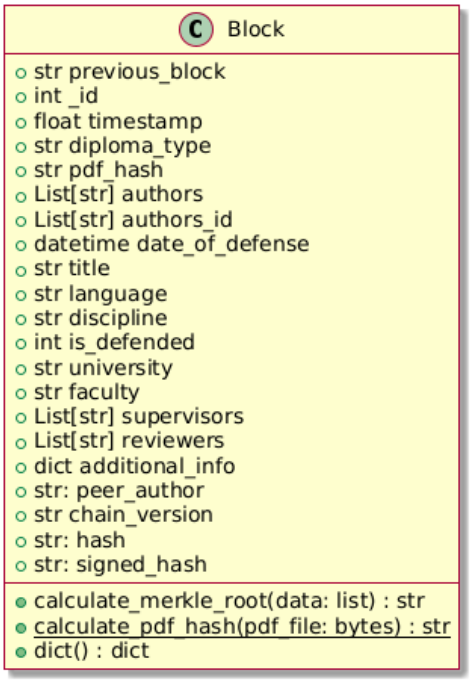
Rysunek 6. Proces podpisywania bloku [10]

- Identyfikator bloku,
- Znacznik czasu (czas utworzenia bloku w formie timestamp),
- Rodzaj dyplomu,
- Hasz PDF (hasz utworzony na podstawie pliku PDF z pracą dyplomową),
- Autora lub autorów pracy,
- Numer identyfikacyjny autorów pracy - numer albumu studenta
- Datę obronienia pracy,
- Tytuł pracy,
- Język,
- Dyscyplina nauki,
- Status pracy (czy została obroniona),
- Nazwę uczelni,
- Uczelnię wyższą,
- Wydział uczelni,
- Promotora lub promotorów,
- Recenzent lub recenzenci,
- Dodatkowe informacje,
- Węzła autora bloku,
- Wersję łańcucha,
- Hasz bloku,
- Podpisany hasz bloku.

Diagram klasy bloku został przedstawiony na rysunku 7. Przedstawione zostały na nim wymienione wyżej pola dostosowane do implementacji oraz metody tej klasy.

calculate_merkle_root(data: list) - metoda, która tworzy drzewo merkle dla podanych danych. Zwraca ona hasz korzenia drzewa. Jest to metoda uzyskiwania hasza dla bloku biorąca pod uwagę każdą przechowywaną daną. Lista danych dzielona jest na pary, które są ze sobą haszowane, poprzez zsumowanie ich i shaszowanie. Powstałe wyniki haszuje się ze sobą, aż do uzyskania tylko jednego wyniku. Dla tych samych danych powstanie zawsze dokładnie taki sam hasz - jednak, jeżeli zmieni się chociaż jedna informacja - hasz ten się zmieni.

Metoda *calculate_pdf_hash(pdf_file: bytes)* służy do obliczania hasza załączonego pliku pdf. Metoda *dict()* zwraca blok w formie *dictionary*.



Rysunek 7. Diagram klasy bloku

6.3 Komunikacja w systemie

Na rysunku 8 przedstawiony został diagram sekwencji dla uruchomienia systemu. Ilustruje on, co dzieje się i jak przepływa komunikacja środku systemu po uruchomieniu go. Jak widać na diagramie - pierwszą instrukcją wydawaną przez Admina bezpośrednio na samej maszynie - jest prośba o uruchomienie go. System jeszcze nie działa - więc niemożliwe jest komunikowanie się z nim poprzez panel administratora, czy nawet powiązane z nim API. Kolejno, system ustanowi połączenie z bazą danych (jak wspomniane w rozdziale 6 - połączenie to umożliwia zarówno odczyt jak i modyfikację zawartości bazy danych). Następnym krokiem jest uruchomienie API dla panelu administratora - od tego momentu będzie ono przyjmować dowolne zapytania - skutkujące wywoływaniem określonych funkcji, takich jak chociażby wymuszenie synchronizacji łańcucha. Następnie system stworzy instancję węzła sieci P2P - który otworzy się na połączenia od innych węzłów w sieci - każdy węzeł to uczestnik systemu. Węzeł ten będzie od uruchomienia dostępny na zapytania od innych użytkowników. Kolejnym krokiem będzie zsynchronizowanie łańcucha blockchain. W tym celu nasz węzeł połączy się z wszystkimi węzłami, które zapisane są na liście węzłów i zapyta je o długość ich łańcucha. Na podstawie tej informacji oraz zapisanych lokalnie danych o samym łańcuchu ustali on od którego węzła uzyskać informacje o łańcuchu. Na wybór węzła wpływa przekazana długość łańcucha oraz

wiarygodność węzła - autoryzowane węzły są bardziej wiarygodne. Wybór jest też częściowo losowy, żeby uniknąć sytuacji w której wszystkie węzły wybierają zawsze ten sam cel. Po ustanowieniu połączenia z celem nasz węzeł zacznie proces porównywania i synchronizacji łańcucha. Zostanie wysłane zapytanie dla każdego brakującego bloku, a sam blok będzie sprawdzany przed dodaniem go do naszej bazy. Sprawdzany będzie jego podpis - czy został wykonany przez autoryzowanego uczestnika oraz czy podpisany hasz nie uległ zmianie - a także to czy wskazuje on prawidłowo na poprzednika. Jeżeli blok zostanie uznany za nieprawidłowy - połączenie zostanie przerwane, a węzeł z którym wykonywaliśmy synchronizację zostanie uznany za niewiarygodny. Proces wybierania wiarygodnego węzła do wykonania synchronizacji ponowi się. Po zakończeniu synchronizacji system przygotuje listę wszystkich węzłów, które nie posiadały kompletnego łańcucha i wyśle im polecenie wykonania synchronizacji łańcucha. Uczestniczy który otrzymali takie polecenie nie wykonują go bezwarunkowo - rozważają wiarygodność węzła który wysłał im to polecenie, oraz sprawdzają czy rzeczywiście posiadają brakujące węzły. Finalnie system zwraca administratorowi informację o tym, że serwis jest zsynchronizowany i gotowy do dalszej pracy.

Na rysunku 9 przedstawiony został diagram sekwencji dla dodania nowego bloku (pracy dyplomowej do łańcucha. Jak widać na tym diagramie - panel admina razem z API już działają, więc to przez panel admin - lub pracownik posiadający odpowiednie uprawnienia dodaje nową pracę dyplomową uzupełniając wszystkie związane z nią informacje. Zapytanie to wraz z danymi trafia poprzez API do blockchain managera. Pierwszym krokiem który on wykonuje jest zsynchronizowanie łańcucha. Proces ten został pokazany dokładniej na rysunku 8. Po zsynchronizowaniu łańcucha przygotowywany jest blok z informacjami o pracy dyplomowej, który następnie jest podpisywany przy użyciu prywatnego klucza uczestnika. Blok ten jest następnie dodawany do bazy i udostępniany innych uczestnikom. Proces ten składa się z uzyskania od węzłów informacji o ich stanie łańcucha, posortowania ich na podstawie informacji o wiarygodności i obecnej ich długości bloku, a następnie przesyłana jest im prośba o zsynchronizowanie łańcucha. Następnie nasz węzeł wielokrotnie sprawdza czy nowy blok został już dodany do łańcucha innych uczestników. Robi to odpytując o niego węzły - o ile zwrócą mu informację, że ich obecna długość łańcucha wskazuje na to, że mogą one zawierać nowo dodany blok lub więcej nowych bloków. Po uzyskaniu odpowiedzi system sprawdza czy blok występujący u innych węzłów jest taki sam jak ten, który sam próbuje dodać. Jeżeli tak jest - proces zostaje zakończony. Jeżeli jednak uzyskany blok różni się od tego, który system próbuje dodać lub minęło już zbyt dużo czasu od dodania tego bloku - uznajemy, że dodawanie bloku nie powiodło się. W takiej sytuacji usuwamy z naszej bazy dodany blok i ponawiamy cały proces - zaczynając znowu od synchronizacji i stworzeniu nowego bloku. Blok który powstanie przy takiej kolejnej próbie będzie różnił się od poprzednika jedynie datą utworzenia. Jednak jego ponowne stworzenie jest potrzebne, żeby wyeliminować ryzyko błędu przy tworzeniu bloku, a także naprawić jego strukturę, w przypadku gdy w międzyczasie został utworzony inny blok - w takiej sytuacji utworzony przez nas blok stał się tak zwanym blokiem sierotą.

6.4 Protokoły komunikacji

Komunikacja pomiędzy uczestnikami sieci realizowana jest przy użyciu protokołów komunikacji. Wykorzystywana biblioteka *p2pd* opisana w rozdziale 5 umożliwia przesył informacji pomiędzy węzłami poprzez przesyłanie ciągów danych w formie binarnej. W celu prawidłowego przesyłu informacji należy zaprojektować protokoły przesyłu informacji. Podstawowy sposób przesyłania wiadomości między węzłami nie różni się dla wysyłanych wiadomości. Wiadomość tworzona jest poprzez utworzenie słownika z następującymi informacjami:

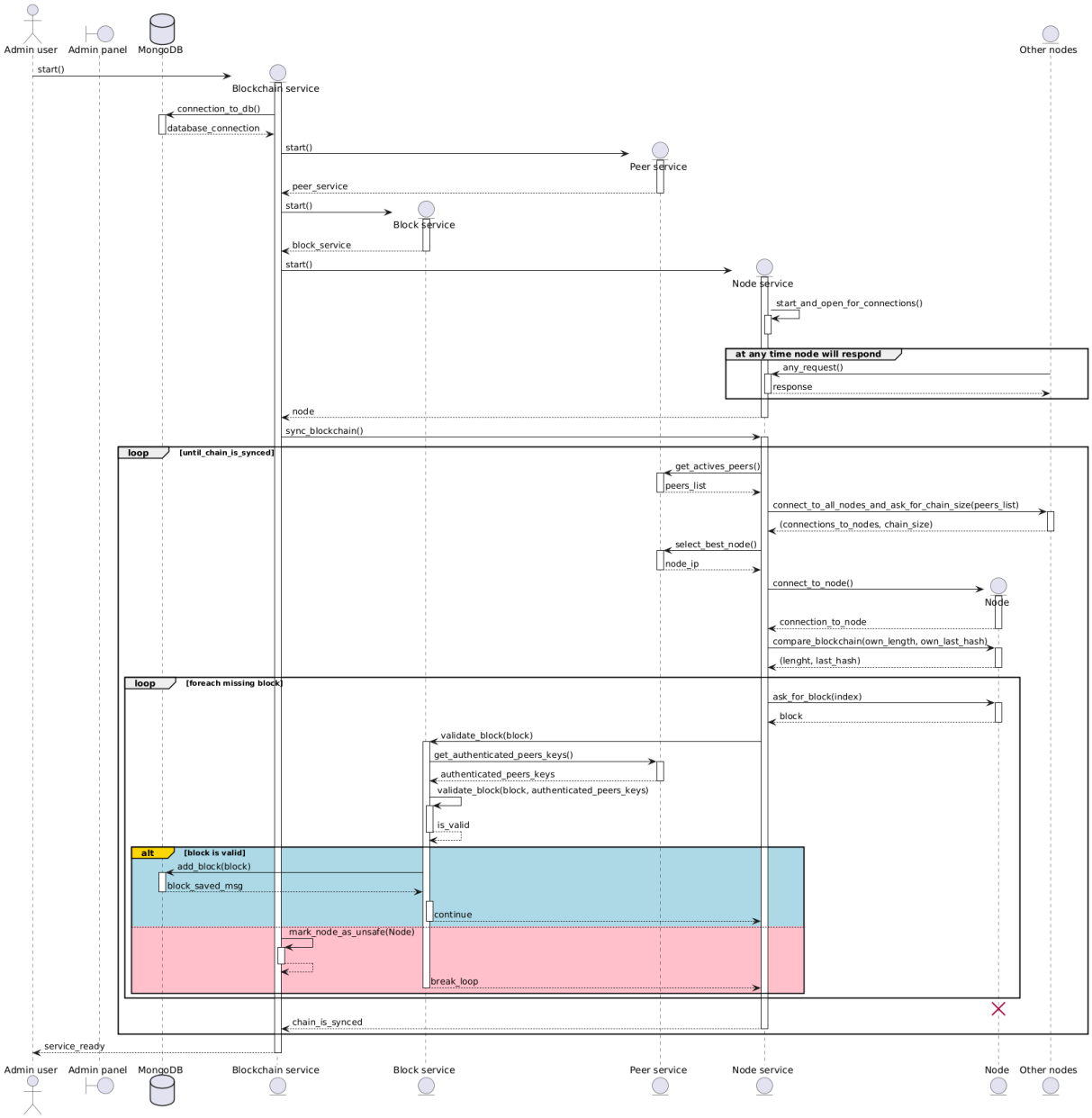
- protokół - wskazuje na rodzaj wysyłanej wiadomości - w jaki sposób powinna zostać obsłużona,
- autor - nazwa uczestnika, który wysłał wiadomość,
- podpis - podpis wiadomości utworzony przy użyciu klucza prywatnego uczestnika i treści wiadomości,
- ładunek - opcjonalne dane przesyłane w wiadomości, takie jak przesyłany blok.

Proces przygotowywania wiadomości do wysłania polega na przetworzenia jej w JSONa, a dokładnie *JSON Canonicalization Scheme (JCS)* - uporządkowaną wiadomość JSON (to ważne dla procesu podpisywania). Następnie tworzony jest podpis dla wiadomości, poprzez zahaszowanie JSONa i jego podpisanie przy użyciu publicznego klucza. Kolejno wiadomość jest zakodowywana w formacie binarnym, a do jej treści dodawany jest prefiks *"DPACHAIN"* - wskazujący na to, że jest to wiadomość zakodowana w naszym własnym protokole.

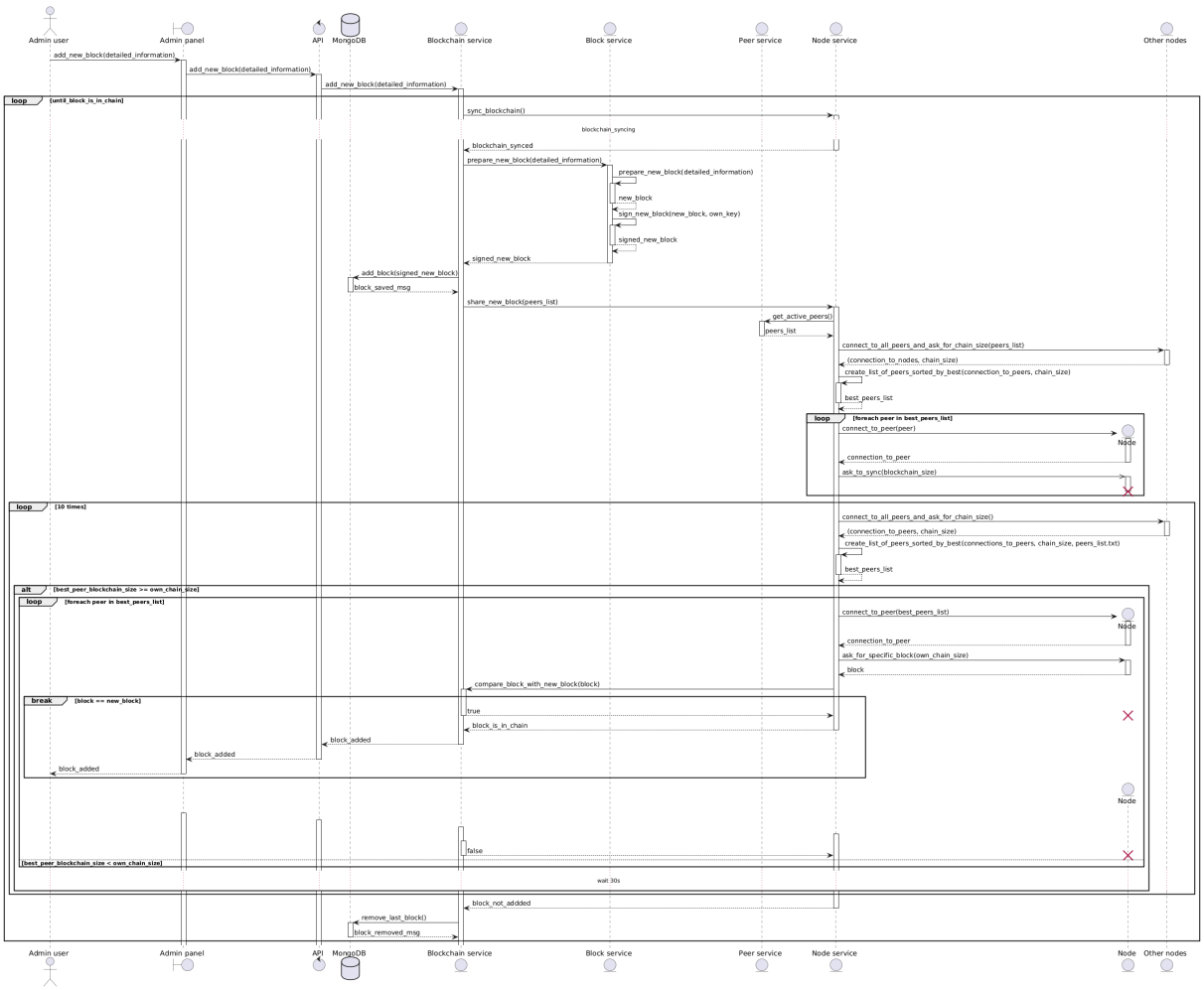
Każdy z protokołów ma określony format i zdefiniowane informacje jakie przesyła. Z założenia - każdy uczestnik powinien odpowiadać na każde zapytanie, które otrzymuje - jeżeli nadawca wiadomości znajduje się na jego liście uczestników i nie jest zablokowany. Nadawcy wiadomości muszą być znani, w celu zmniejszenia ilości złośliwych zapytań, na które odpowiadać będzie uczestnik systemu. Złośliwi nadawcy wiadomości mogą zostać zablokowani przez administratora systemu na podstawie logów - a w przyszłości można też wdrożyć system, który blokować ich będzie automatycznie. Specyfiką wykorzystywanej przez nas biblioteki do obsługi połączeń Peer2Peer jest to, że po ustanowieniu połączenia z węzłem, po przesłaniu wiadomości i ewentualnej uzyskaniu na nie odpowiedzi - połączenie powinno zostać zamknięte, tak żeby węzeł odpowiadający nie utrzymywał potencjalnie niepotrzebnego procesu. W wyniku tego, po przesłaniu każdej pary wiadomości należy na nowo otwierać połączenia. Skutkuje to dużą ilością połączeń i rozłączeń między węzłami.

W obecnej wersji projekt zakłada dwie metody dodawania nowych uczestników do listy. Pierwsza metoda zakłada ręczne wprowadzanie danych o nowych uczestnikach przez administratora systemu. Tylko w ten sposób mogą też być dodawani nowi uczestnicy autoryzowani systemu. Drugi sposób jest bardziej przyjazny nieautoryzowanym uczestnikom. Mogą oni wysłać wiadomość specjalnym protokołem do "przedstawiania się" innym węzłom. Polityka akceptacji takich uczestników będzie mogła być w przyszłości zmieniona na bardziej restrykcyjną, jeżeli system będzie cierpiał na zbyt dużej ilości złośliwych uczestników - dołączenie do systemu może być dozwolone tylko za zgodą lub ograniczone w całości do autoryzowanych uczestników.

Większość protokołów komunikacji tworzona jest w parach - istnieją protokoły pytające i odpowiadające na pytania. Protokoły pytające w większości nie przesyłają ładunku. Podstawowym protokołem jest na przykład protokół *"ask_chain_size"* - proszący odbiorcę o przesłanie swojego rozmiaru łańcucha bloków przy użyciu protokołu *"response_chain_size"*.
Podobną parą protokołów jest *"ask_compare_blockchain"* i *"response_compare_blockchain"*, które służą do porównywania łańcucha pomiędzy uczestnikami. Odbiorca przesyła nadawcy wiadomości rozmiar swojego łańcucha i hasz ostatniego bloku. Oba te protokoły wykorzystywane są przy synchronizacji łańcucha oraz przy dodawaniu nowego bloku do łańcucha.
Protokoły *"ask_block"* oraz *"response_block"* działają w zbliżony sposób, ale tym razem nadawca pyta o przesłanie mu określonego bloku definiując w ładunku wiadomości indeks bloku o który pyta.
Omawianym wyżej specjalnym protokołem do "przedstawiania się" węzłowi jest *"new_peer"*. Protokół ten jest wyjątkowy, bo nie wysyła podpisu w treści wiadomości - oraz jest jedynym, który nie powinien być odrzucany w przypadku nieznalezienia autora wiadomości na liście - przeczyło by to jego celowi. Nie posiada on także odpowiadającego protokołu "odpowiadającego". Także inaczej niż reszta protokołów "proszących" protokół ten nie czeka na odpowiedź węzła, ponieważ obsłużenie tej wiadomości nie inicjuje żadnej odpowiedzi - węzeł jest po prostu dodawany do listy węzłów i od teraz uzyska odpowiedzi na wiadomości innych protokołów.
Kolejnych protokołem - który także nie oczekuje na odpowiedź - jest protokół *"ask_sync_chain"*. Jego celem jest poproszenie odbiorcy wiadomości o dokonanie synchronizacji łańcucha. Jego głównym zastosowaniem jest propagowanie nowego bloku - w formie przekazania informacji o tym, że został dodany nowy blok i uczestnik powinien zsynchronizować swój łańcuch do najnowszej wersji. Wiadomość z tym protokołem mogła by także zostać wysłana po tym, jak węzeł w trakcie odpotywania znanych mu węzłów o ich rozmiar - zauważy mniejszy niż reszty rozmiar pewnego węzła.



Rysunek 8. Diagram sekwencji dla uruchomienia aplikacji



Rysunek 9. Diagram sekwencji dla dodania nowego bloku

Rozdział 7

Podsumowanie

Bibliografia

[1] BigchainDB GmbH, „BigchainDB 2.0 The Blockchain Database", BigchainDB GmbH, Berlin, Germany, spraw. tech., maj 2018, Paper version 1.0.

[2] Chauhan, A., „A Review on Various Aspects of MongoDb Databases", *International Journal of Engineering Research & Technology (IJERT)*, t. 8, nr. 05, maj 2019, IJERTV8IS050031. Published by: http://www.ijert.org. Licensed under a Creative Commons Attribution 4.0 International License., ISSN: 2278-0181.

[3] Django Software Foundation, *Django: The web framework for perfectionists with deadlines — Databases*, Documentation version 5.1, accessed 3 February 2025, 2022. adr.: https://docs.djangoproject.com/en/5.1/ref/databases/#.

[4] Golosova, J. i Romanovs, A., „The Advantages and Disadvantages of the Blockchain Technology", w *2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, 2018, s. 1–6. DOI: 10.1109/AIEEE.2018.8592253.

[5] Grolleau, G., Lakhal, L. i Mzoughi, N., „An introduction to the economics of fake degrees", *Journal of Economic Issues*, t. 42, nr. 3, s. 673–693, 2008.

[6] Hayes, A. S., „A Cost of Production Model for Bitcoin", Department of Economics The New School for Social Research, New York, NY, spraw. tech., lut. 2015.

[7] Hussein, Z., Salama, M. i El-Rahman, S., „Evolution of blockchain consensus algorithms: a review on the latest milestones of blockchain consensus algorithms", *Cybersecurity*, t. 6, s. 30, 2023. DOI: 10.1186/s42400-023-00163-y. adr.: https://doi.org/10.1186/s42400-023-00163-y.

[8] John, K., O'Hara, M. i Saleh, F., „Bitcoin and Beyond", *Annual Review of Financial Economics*, t. 14, s. 95–115, 2022.

[9] Kalajdjieski, J., Raikwar, M., Arsov, N., Velinov, G. i Gligoroski, D., „Databases fit for blockchain technology: A complete overview", *Blockchain: Research and Applications*, t. 4, nr. 1, s. 100 116, 2023, ISSN: 2096-7209. DOI: https://doi.org/10.1016/j.bcra.2022.100116. adr.: https://www.sciencedirect.com/science/article/pii/S2096720922000574.

[10] Raikwar, M., Gligoroski, D. i Kralevska, K., „SoK of Used Cryptography in Blockchain", *IEEE Access*, t. 7, s. 1–1, paź. 2019. DOI: 10.1109/ACCESS.2019.2946983.

[11] Saad, M., Spaulding, J., Njilla, L., Kamhoua, C. A., Nyang, D. i Mohaisen, A., „Overview of attack surfaces in blockchain", w *Blockchain for Distributed Systems Security*, Shetty, S. S., Kamhoua, C. A. i Njilla, L. L., red., Wiley-IEEE Computer Society Pr, 2019, s. 51–62, ISBN: 978-1-119-51960-7.

[12] Sayed, R. H., „Potential of Blockchain Technology to Solve Fake Diploma Problem", prac. mag., University of Jyväskylä, Department of Computer Science i Information Systems, 2019.

[13] Segendorf, B., „Have virtual currencies affected the retail payments market?", *Economic Commentaries, no. 2, Sveriges Riksbank*, 2014.

[14] Turing, *Python FastAPI vs Flask: A Detailed Comparison*. adr.: https://www.turing.com/kb/fastapi-vs-flask-a-detailed-comparison.

[15] Wikipedia contributors, *Raft (algorithm) — Wikipedia, The Free Encyclopedia*, [Online; accessed 3-February-2025], 2025. adr.: https://en.wikipedia.org/w/index.php?title=Raft_(algorithm)&oldid=1270154233.

Wykaz skrótów i symboli

Spis rysunków

1 Diagram aktorów 11
2 Diagram przypadków użycia klienta 12
3 Diagram przypadków użycia pracownika 12
4 Diagram przypadków użycia administratora 13

5 Diagram architektury systemu 22
6 Proces podpisywania bloku [10] 23
7 Diagram klasy bloku 24
8 Diagram sekwencji dla uruchomienia aplikacji 28
9 Diagram sekwencji dla dodania nowego bloku 29

Spis tabel

1 Porównanie bibliotek FastAPI i Flask [14] 20

Spis załączników