

UNIWERSYTET W BIAŁYMSTOKU
WYDZIAŁ INFORMATYKI



**Opracowanie Webowego Serwisu Archiwizacji
Danych Eksperimentalnych**

Dawid Jacek Metelski

Promotor:
dr hab. Witold Rudnicki, prof. UwB

BIAŁYSTOK 2024

Spis treści

Wstęp	5
1 Wprowadzenie teoretyczne	6
1.1 Analiza Metod Śledzenia i Poświadczania Pochodzenia Obiektów Cyfrowych (Data Lineage and Provenance) w Kontekście Archiwizacji Danych Doświadczalnych	6
1.1.1 Data Lineage - Śledzenie Życiorysu Danych	6
1.1.2 Data Provenance - Poświadczenie Pochodzenia Danych	7
1.1.3 Technologie Wspierające Data Lineage i Provenance	8
1.1.4 Wyzwania i Rozwiązania	13
1.1.5 Wnioski Analizy	14
1.2 Bezpieczeństwo Danych w Blockchain	15
1.2.1 Blockchain - Łańcuch Bloków Informacji	15
1.2.2 Historia Blockchain	16
1.2.3 Kluczowe Cechy Gwarantujące Bezpieczeństwo Danych	17
1.2.4 Składowe Technologii Blockchain	18
1.2.5 Mechanizmy Konsensusu	19
1.2.6 Skalowalność w Blockchain	20
1.2.7 Wydajność w Blockchain	20
1.3 Nierelacyjne Bazy Danych	21
1.3.1 Charakterystyka Nierelacyjnych Baz Danych	21
1.3.2 Typy Nierelacyjnych Baz Danych	22
1.3.3 Zastosowania Nierelacyjnych Baz Danych	23
1.3.4 Nierelacyjne Bazy Danych a Blockchain	24
1.4 Przechowywanie Danych Medycznych	25
1.4.1 Znaczenie Przechowywania Danych Medycznych	25
1.4.2 Regulacje Prawne w Polsce i Europie	26
1.4.3 Procedury Zgodności z Regulacjami Prawnymi	27
1.4.4 Metody Przechowywania Danych Medycznych	28
1.4.5 Wyzwania i Rozwiązania Przechowywania Danych Medycznych .	28
1.5 Wprowadzenie Nowych Procedur Medycznych na Rynek	29
1.5.1 ISO 8000 - Jakość Danych	30
1.5.2 ISO 27001 - Zarządzanie Bezpieczeństwem Informacji	30
1.5.3 ISO 13485 - System Zarządzania Jakością dla Wyrobów Medycznych	31
1.5.4 ISO 27040 - Bezpieczeństwo Przechowywania Danych	31
1.5.5 IEEE 802 - Zarządzanie Metadanymi	32

1.5.6	GxP (Good Practice)	32
1.5.7	Podsumowanie	33
2	Technologie	34
2.1	Hyperledger Fabric	34
2.1.1	Architektura Hyperledger Fabric	34
2.1.2	Dlaczego Hyperledger Fabric?	35
2.1.3	Zastosowania	36
2.2	Flask	37
2.2.1	Charakterystyczne Cechy Flaska	37
2.2.2	Dlaczego Flask?	37
2.2.3	Zastosowania Flaska	38
2.2.4	Przykład Użycia	38
3	Serwis Archiwizacji Danych Eksperimentalnych	40
3.1	Diagram Przypadków Użycia	40
3.1.1	Aktorzy Aplikacji	40
3.2	Architektura Systemu	42
3.3	Implementacja Systemu Hyperledger Fabric	43
3.3.1	Konfiguracja Projektu	44
3.3.2	Struktura Systemu	46
3.4	Implementacja w Technologii Flask	49
3.4.1	Architektura Aplikacji	49
3.4.2	Konfiguracja Środowiska	50
3.4.3	Zarządzanie Użytkownikami	50
3.4.4	Zarządzanie Danymi	51
3.5	Graficzny Interfejs Użytkownika	54
3.6	Testy Wydajnościowe	59
3.6.1	Scenariusze i Założenia Testów Wydajnościowych	59
3.6.2	Pobieranie Listy Danych Medycznych	60
3.6.3	Przesłanie Nowych Danych Medycznych	61
3.6.4	Pobieranie Historii Danych	63
3.6.5	Pobieranie Listy Ról Użytkowników	65
3.6.6	Tworzenie Nowych Ról Użytkowników	67
3.6.7	Pobieranie Listy Szpitali	69
3.6.8	Tworzenie Nowych Szpitali	71
3.6.9	Pobieranie Listy Użytkowników	73
3.6.10	Tworzenie Nowych Użytkowników	75
3.6.11	Skalowanie Rozmiaru Danych	77
3.6.12	Podsumowanie	80
3.7	Wdrożenie Systemu	80
3.7.1	System Operacyjny	80
3.7.2	System Kontroli Wersji	81
3.7.3	Licencja	81
Podsumowanie		83
Bibliografia		85

Wstęp

W dzisiejszych czasach epoki cyfryzacji, gdzie ilość generowanych danych rośnie w dynamicznym tempie, kluczowe staje się zapewnienie ich bezpieczeństwa, optymalnego zarządzania i archiwizacji. Obecnie ilość danych cyfrowych podwaja się co około dwa lata, co stanowi ogromne wyzwanie logistyczne i technologiczne. W 2020 roku globalna liczba danych wynosiła około 64,2 zettabajtów i przewiduje się, że do 2025 roku osiągnie 175 zettabajtów [2].

Celem niniejszej pracy jest stworzenie serwisu do archiwizacji danych eksperymentalnych, który będzie służył gromadzeniu, a także analizie danych medycznych, wykorzystującej nowe procedury medyczne. Wprowadzanie takich procedur regulowane jest przepisami prawnymi, z których wynikają pewne wymagania dotyczące zarządzania danymi. Warto podkreślić, że niniejszego projektu nie dotyczą wymagania RODO, ponieważ wykorzystywane w nim dane nie zawierają danych osobowych.

Do wykonania projektu zostaną wykorzystane technologie, które zapewnią bezpieczeństwo przechowywanych danych, a także spełnienie wymagań regulacji dotyczących weryfikacji nowych procedur medycznych. Założenia obejmują technologię blockchain, nierelacyjne bazy danych i proste frameworki internetowe.

Rozdziały pracy przedstawiają teoretyczne podstawy bezpieczeństwa danych w technologii blockchain, przepisy regulujące wprowadzanie nowych procedur medycznych oraz praktyczne aspekty implementacji systemu archiwizacji danych. Przeprowadzona analiza pozwoli na dogłębne zrozumienie wyzwań i możliwości wykorzystania nowoczesnych technologii w zarządzaniu danymi naukowymi. W rezultacie, praca ta wniesie wartościowy wkład dla analizy danych medycznych, dostarczając nowe perspektywy i narzędzie do efektywnego zarządzania danymi.

1. Wprowadzenie teoretyczne

1.1 Analiza Metod Śledzenia i Poświadczania Pochodzenia Obiektów Cyfrowych (Data Lineage and Provenance) w Kontekście Archiwizacji Danych Doświadczalnych

Dane ekspermentalne, na przykład gromadzone w celu walidacji skuteczności procedur medycznych są gromadzone w postaci zapisów w systemach komputerowych. Zapewnienie przejrzystości, wiarygodności oraz możliwości weryfikacji dużych zbiorów danych ekspermentalnych, jest poważnym wyzwaniem. Główną rolę w tym kontekście odgrywają metody śledzenia linii życia danych (Data Lineage) i poświadczania ich pochodzenia (Data Provenance). W związku z tym niezbędne jest zrozumienie pochodzenia danych oraz procesów, przez które przechodzą.

1.1.1 Data Lineage - Śledzenie Życiorysu Danych

Znaczenie Data Lineage wzrasta wraz ze zwiększeniem się złożoności procesów badawczych i ekspermentalnych. Precyzyjne śledzenie pochodzenia danych umożliwia lepsze zrozumienie i weryfikację procesów badawczych. Jest to szczególnie istotne w przypadku danych będących podstawą do dopuszczenia stosowania procedur medycznych.

Definicja i Podstawowa Koncepcja

Data Lineage, czyli pochodzenie danych odnosi się do procesu przepływu danych. Polega na śledzeniu wszystkich zmian przeprowadzonych w określonych danych, od źródła po finalne użycie. Jest ważnym aspektem zarządzania danymi w badaniach naukowych, umożliwiający zrozumienie przetwarzania i wykorzystania danych. Jak zaznaczyli Woodruff i Stonebraker "Zrozumienie pochodzenia danych może prowadzić do lepszego zaufania do danych i ich analiz"[44].

W przeszłości śledzenie danych prowadzono ręcznie, co wymagało dużego nakładu czasu i było podatne na błędy. Dziś zautomatyzowane narzędzia ułatwiają ten proces, optymalizując tym samym pracę wielu naukowcom. Przykładami mogą być chociażby

programy do tworzenia dokumentów, takie jak Word czy Latex, które automatycznie śledzą zmiany, liczą słowa i wiele innych.

Znaczenie w Archiwizacji Danych Eksperymentalnych

Dokładne śledzenie cyklu życia danych eksperymentalnych zapewnia możliwości powtarzania eksperymentów i weryfikacji wyników. W archiwizacji danych niezbędne jest nie tylko zrozumienie transformacji danych, ale także ich bezpieczne przechowywanie i archiwizacja. Dobrze zaprojektowane systemy archiwizacji danych pozwalają na szybkie odzyskiwanie i analizę danych, co przyspiesza procesy badawcze. Ponadto, zaawansowane techniki szyfrowania i autoryzacji dostępu zapewniają bezpieczeństwo i poufność danych, chroniąc je przed nieautoryzowanym dostępem i utratą. Dzięki wyeliminowaniu błędów ludzkich zautomatyzowane systemy zwiększą wiarygodność danych, co skutecznie wpływa na precyzję badań.

1.1.2 Data Provenance - Poświadczenie Pochodzenia Danych

Dokumentowanie historii i pochodzenia danych, znane jako Data Provenance, jest niezbędnym elementem zarządzania danymi naukowymi. Pełne zrozumienie pochodzenia danych jest fundamentem do ich skutecznego wykorzystania i interpretacji, umożliwiając weryfikację autentyczności oraz ocenę jakości i wiarygodności danych.

Definicja i Kluczowe Aspekty

Data Provenance odnosi się do szczegółowego dokumentowania historii danych, w tym ich źródeł, zmian oraz odpowiedzialnych podmiotów. Zapewnia widoczność błędów i upraszcza ich śledzenie, umożliwiając reprodukcję aż do głównej przyczyny awarii. Dzięki temu procesowi możliwe jest odtwarzanie całego przepływu danych lub jego części, na przykład w celu debugowania. Odnosi się do zapisów danych wejściowych, systemów, procesów i jednostek, które wpływają na historię danych. Data provenance jest niezbędne do oceny wiarygodności danych, szczególnie w kontekście danych eksperymentalnych. Jak stwierdził Peter Buneman "Dokumentowanie pochodzenia danych jest kluczowe w naukach eksperymentalnych, gdzie interpretacja danych zależy od ich kontekstu" [?].

Rola w Weryfikacji Danych

Data Provenance odgrywa znaczącą rolę w weryfikacji danych w różnych dziedzinach, zapewniając szczegółową dokumentację pochodzenia danych, co jest niezbędne do oceny ich integralności i autentyczności. Oto główne aspekty:

- **Weryfikacja autentyczności danych:** Data provenance wykorzystywane do śledzenia historii danych pacjentów w elektronicznych rejestrach zdrowotnych umożliwiając identyfikację i naprawę błędów w danych [1].
- **Automatyzacja procesów i redukcja błędów ludzkich:** Automatyzacja procesu śledzenia pochodzenia danych eliminuje błędy ludzkie i znaczco zwiększa wiarygodność danych. Na przykład w badaniach naukowych zautomatyzowane narzędzia mogą śledzić każdy etap przetwarzania danych, zapewniając tym samym dokładność i przejrzystość. Narzędzia te mogą również monitorować zmiany w danych w czasie rzeczywistym, co pozwala zachować integralność danych.

1.1.3 Technologie Wspierające Data Lineage i Provenance

Rozwój nowych technologii, takich jak blockchain, zaawansowane systemy bazodanowe oraz inne innowacyjne rozwiązania, otwiera nowe możliwości w zakresie śledzenia i poświadczania pochodzenia danych. Wykorzystanie tych innowacji może znacząco ułatwić zarządzanie danymi eksperymentalnymi, ponieważ zapewniają jednocześnie większą transparentność i bezpieczeństwo przechowywanych informacji.

Blockchain

Blockchain jest zdecentralizowaną technologią rozproszonego rejestru, która umożliwia przechowywanie danych w sposób zapewniający ich niezmiennosć i transparentność. Każdy blok łańcucha zawiera kryptograficznie zabezpieczone informacje o transakcjach. Uniemożliwia to ich modyfikację bez wykrycia. Blockchain działa na zasadzie rozproszonej sieci węzłów, gdzie każdy węzeł przechowuje pełną kopię łańcucha bloków. Transakcje są zatwierdzane przez sieć węzłów, które muszą osiągnąć konsensus co do ich ważności, co dodatkowo zwiększa bezpieczeństwo i integralność danych.

Aby dokładniej zrozumieć działanie technologii blockchain w przedstawianym kontekście, należy przyjrzeć się jej najważniejszym elementom:

- **Schematy i struktury bloków:** Każdy blok w łańcuchu zawiera nagłówek oraz listę transakcji. Nagłówek zawiera m.in. skrót kryptograficzny poprzedniego bloku, znacznik czasu oraz tzw. nonce, czyli liczbę używaną jednorazowo w procesie tworzenia bloku. Jak opisuje Antonopoulos w książce *Mastering Bitcoin* [5], struktura bloków zapewnia integralność danych oraz ich chronologię.
- **Mechanizmy szyfrowania i deszyfrowania:** Blockchain wykorzystuje zaawansowane techniki kryptograficzne do zabezpieczania danych, między innymi algorytm hashujący (np. SHA-256), który przekształca dane wejściowe w unikalny skrót o stałej długości. Arvind Narayanan i jego współautorzy w książce *Bitcoin and Cryptocurrency Technologies* [35] omawiają techniczne aspekty kriptografii używanej w blockchainie.
- **Mechanizmy poświadczania:** Każda transakcja w sieci jest podpisywana cyfrowo, co umożliwia weryfikację tożsamości nadawcy oraz integralności danych. W publikacji *Blockchain Basics: A Non-Technical Introduction in 25 Steps* [18] Daniel Drescher wyjaśnia, jak mechanizmy poświadczania przyczyniają się do bezpieczeństwa i transparentności systemu.

Blockchain jest szeroko uznawany za jedną z najbardziej innowacyjnych technologii w zakresie zarządzania danymi. Jak opisano w książce *Blockchain: Blueprint for a New Economy* autorstwa Melanie Swan, "blockchain oferuje niespotykane dotąd poziomy bezpieczeństwa i integralności danych dzięki swoim kryptograficznym podstawom i decentralizacji" [38].

Zalety Blockchain

Blockchain oferuje wiele zalet, które czynią go atrakcyjnym rozwiązaniem do śledzenia i poświadczania pochodzenia danych:

- **Niezmiennosć zapisów:** Raz zapisane dane w blockchainie nie mogą być zmienione bez zmiany wszystkich kolejnych bloków, co jest praktycznie niemożliwe do przeprowadzenia bez wykrycia [43].

- **Wysoka odporność na manipulacje:** Dzięki zdecentralizowanej naturze, każda próba manipulacji danymi jest natychmiast wykrywana przez sieć [9].
- **Transparentność:** Publiczne blockchainy umożliwiają wgląd w pełną historię transakcji, co zwiększa zaufanie i przejrzystość [42].
- **Decentralizacja:** Brak centralnego punktu awarii sprawia, że system jest bardziej odporny na ataki i awarie [35].

Jak zauważa Paul Vigna, "blockchain oferuje niezrównany poziom bezpieczeństwa i wiarygodności danych, co czyni go idealnym rozwiązaniem dla wielu aplikacji, w tym archiwizacji danych naukowych"[42].

Wady Blockchain

Mimo licznych zalet, blockchain ma również pewne wady:

- **Wysokie koszty energii:** Mechanizmy konsensusu, takie jak Proof of Work (PoW), wymagają dużej mocy obliczeniowej, co przekłada się na wysokie zużycie energii [16].
- **Problemy z przepustowością:** Wysokie koszty transakcji i długi czas przetwarzania mogą być problematyczne w systemach o dużym obciążeniu [15].

Zastosowanie Blockchain

Blockchain znajduje zastosowanie w wielu dziedzinach:

- **Kryptowaluty:** Bitcoin i Ethereum są najpopularniejszymi przykładami wykorzystania blockchaina do zarządzania cyfrowymi walutami [34, 43].
- **Inteligentne kontrakty:** Platformy takie jak Ethereum umożliwiają tworzenie i wykonywanie inteligentnych kontraktów, które automatycznie realizują określone warunki umowy [11].
- **Archiwizacja danych naukowych:** Blockchain może być wykorzystywany do zapewnienia transparentności i wiarygodności pochodzenia danych w badaniach naukowych, umożliwiając śledzenie całej historii danych od ich zebrania po końcowe analizy [7].

Systemy Chmurowe

Systemy chmurowe, takie jak Amazon Web Services (AWS), Google Cloud Platform (GCP) i Microsoft Azure, oferują skalowalne i elastyczne rozwiązania do przechowywania i przetwarzania dużych ilości danych. Te platformy zapewniają zaawansowane usługi bazodanowe i analityczne, które mogą wspierać zarządzanie Data Lineage i Provenance. Dzięki rozproszonej infrastrukturze, dane są przechowywane w różnych centrach danych, co zapewnia ich dostępność i niezawodność. Usługi takie jak AWS Glue, BigQuery w GCP czy Azure Data Factory posiadają narzędzia do integracji, przetwarzania i analizowania danych, umożliwiając śledzenie całego cyklu życia danych [39].

Systemy chmurowe działają na zasadzie wirtualizacji zasobów, gdzie fizyczne serwery są podzielone na wiele wirtualnych maszyn, które mogą być dynamicznie przydzielane w zależności od obciążenia. Korzystają one z zaawansowanych technologii konteneryzacji, takich jak Docker oraz systemów orkiestracji kontenerów jak Kubernetes, aby zapewnić elastyczność i skalowalność aplikacji [33].

Zalety Systemów Chmurowych

Zalety systemów chmurowych prezentują się następująco:

- **Skalowalność:** Możliwość dynamicznego dostosowywania zasobów w zależności od potrzeb, co jest szczególnie przydatne w kontekście dużych zbiorów danych [45].
- **Elastyczność:** Szybkie dostosowanie się do zmieniających się wymagań biznesowych i technologicznych.
- **Wysoka dostępność:** Redundancja danych i zaawansowane mechanizmy odzyskiwania po awarii zapewniają ciągłość działania [12].
- **Zaawansowane narzędzia bezpieczeństwa i zgodności:** Systemy chmurowe zawierają kompleksowe rozwiązania zabezpieczające, takie jak szyfrowanie danych, kontrola dostępu i monitorowanie aktywności [39].

Jak zauważa Thomas Erl w książce *Cloud Computing: Concepts, Technology & Architecture*, "chmura obliczeniowa oferuje niespotykane możliwości skalowania i elastyczności, co czyni ją idealnym rozwiązaniem dla zarządzania dużymi zbiorami danych" [39].

Wady Systemów Chmurowych

Pomimo licznych zalet, systemy chmurowe mają również pewne wady:

- **Zależność od dostawcy usług:** Ryzyko vendor lock-in, czyli uzależnienie od jednego dostawcy usług chmurowych, co może utrudniać migrację danych do innej platformy [6].
- **Problemy z prywatnością danych:** Potencjalne obawy dotyczące bezpieczeństwa danych przechowywanych w chmurze, zwłaszcza w kontekście zgodności z przepisami o ochronie danych osobowych [45].
- **Koszty:** Wysokie koszty długoterminowego przechowywania i przetwarzania dużych ilości danych, które mogą rosnąć w miarę wzrostu wykorzystania zasobów [12].

Zastosowania Systemów Chmurowych

Systemy chmurowe są szeroko stosowane w różnych branżach:

- **Przechowywanie danych big data:** Firmy takie jak Netflix, Airbnb i Spotify wykorzystują chmurę do przechowywania i analizowania ogromnych zbiorów danych użytkowników [6].
- **Analiza danych:** Narzędzia takie jak AWS Glue, BigQuery w GCP i Azure Data Factory umożliwiają integrację, przetwarzanie i analizę danych na dużą skalę [45].
- **Zarządzanie Data Lineage i Provenance:** Systemy chmurowe dostarczają narzędzia do śledzenia pochodzenia danych i zarządzania ich historią w sposób skalowalny i elastyczny [12].

Zaawansowane Systemy Zarządzania Bazami Danych (DBMS)

Zaawansowane systemy zarządzania bazami danych (DBMS), takie jak Oracle, SQL Server i MySQL, dostarczają funkcje do zarządzania danymi, w tym śledzenie i poświadczanie ich pochodzenia. Są wyposażone w narzędzia do zarządzania metadanymi, logowania zmian i monitorowania aktywności użytkowników. DBMS umożliwiają przechowywanie, zarządzanie i przetwarzanie danych w sposób uporządkowany i bezpieczny, co jest niezbędne w archiwizacji danych medycznych [25].

Systemy DBMS działają na zasadzie relacyjnych modeli danych, gdzie dane są przechowywane w tabelach powiązanych za pomocą kluczy obcych. Relacyjne bazy danych wykorzystują język SQL do zarządzania i manipulowania danymi. Oprócz tradycyjnych relacyjnych DBMS istnieją również nowoczesne bazy NoSQL, takie jak MongoDB czy Cassandra, które oferują elastyczność w przechowywaniu niestrukturyzowanych danych [13].

Zalety DBMS

Systemy DBMS mają następujące zalety:

- **Dojrzałość technologii:** Długie doświadczenie i stabilność sprawiają, że systemy te są niezawodne i sprawdzone.
- **Szeroki zakres funkcji:** Kompleksowe narzędzia do zarządzania danymi, w tym zaawansowane mechanizmy zapytań, optymalizacji wydajności i zarządzania transakcjami.
- **Wsparcie techniczne:** Rozbudowana pomoc techniczna i dokumentacja, które ułatwiają implementację i zarządzanie systemami.

Jak zauważa Hector Garcia-Molina w książce *Database Systems: The Complete Book*, zaawansowane systemy zarządzania bazami danych oferują szeroki wachlarz funkcji, które zapewniają efektywne zarządzanie dużymi zbiorami danych [23].

Wady DBMS

Wady systemów DBMS mogą obejmować:

- **Koszt licencji:** Wysokie koszty zakupu i utrzymania licencji, szczególnie w przypadku dużych przedsiębiorstw [21].
- **Złożoność konfiguracji i zarządzania:** Wymaga specjalistycznej wiedzy i doświadczenia, co może generować dodatkowe koszty związane z zatrudnieniem wykwalifikowanego personelu [21].

Zastosowania DBMS

Systemy DBMS znajdują swoje zastosowanie w:

- **Zarządzanie dużymi zbiorami danych:** Firmy takie jak Bank of America, Walmart i NASA korzystają z zaawansowanych systemów DBMS do zarządzania swoimi danymi.
- **Śledzenie metadanych i logów aktywności:** DBMS umożliwiają dokładne monitorowanie i dokumentowanie zmian w danych.

- **Wsparcie dla procesów biznesowych wymagających wysokiej niezawodności:** Systemy DBMS są wykorzystywane w sektorach, gdzie niezawodność i bezpieczeństwo danych są na pierwszym miejscu, na przykład finanse, zdrowie i telekomunikacja.

Apache Kafka

Apache Kafka to rozproszona platforma strumieniowa, która umożliwia śledzenie i zarządzanie strumieniami danych w czasie rzeczywistym. Jest przydatna w kontekście danych eksperymentalnych, gdzie są one generowane i przetwarzane w sposób ciągły [31]. Pozwala na rejestrowanie i przechowywanie strumieni danych oraz zapewnia narzędzia do analizy i przetwarzania tych danych na bieżąco [24].

Kafka działa jako rozproszony system, który składa się z brokerów, tematów i partycji. Dane są publikowane do tematów, a każdy temat jest podzielony na partycje, które są replikowane na różnych brokerach w celu zapewnienia wysokiej dostępności i niezawodności [31]. Konsumenti subskrybują tematy i przetwarzają dane w czasie rzeczywistym.

Zalety Apache Kafka

Apache Kafka oferuje wiele zalet:

- **Przetwarzanie danych w czasie rzeczywistym:** Umożliwia natychmiastową analizę i reakcję na dane, co jest kluczowe w wielu zastosowaniach biznesowych i naukowych [31].
- **Wysoka wydajność:** Kafka zapewnia szybkie i niezawodne przetwarzanie dużych wolumenów danych, co jest niezbędne w kontekście danych big data [31].
- **Skalowalność:** Możliwość rozbudowy infrastruktury w miarę wzrostu potrzeb, co pozwala na efektywne zarządzanie rosnącymi zbiorami danych.
- **Redundancja i replikacja:** Dane są replikowane na wielu brokerach, co zapewnia wysoką dostępność i odporność na awarie.

Jak opisano w książce *Kafka: The Definitive Guide* autorstwa Neha Narkhede, Gwen Shapira i Todd Palino, "Apache Kafka jest niezwykle potężnym narzędziem do przetwarzania strumieni danych w czasie rzeczywistym, oferującym wysoką wydajność i skalowalność" [24].

Zastosowania Apache Kafka

Apache Kafka jest stosowana w wielu scenariuszach:

- **Zarządzanie strumieniami danych w czasie rzeczywistym:** Firmy takie jak LinkedIn, Netflix i Uber wykorzystują Kafka do zarządzania danymi strumieniowymi.
- **Analiza danych big data:** Kafka jest wykorzystywana do analizy dużych zbiorów danych w czasie rzeczywistym, co pozwala na szybkie podejmowanie decyzji biznesowych.
- **Integracja systemów o dużej przepustowości:** Kafka umożliwia integrację różnych systemów IT, zapewniając wysoką wydajność i niezawodność przesyłu danych [31].

1.1.4 Wyzwania i Rozwiązania

W procesie zarządzania danymi medycznymi, zwłaszcza w kontekście ich śledzenia i poświadczania pochodzenia, pojawia się szereg wyzwań. Wymagają one często złożonych rozwiązań, które są w stanie sprostać dynamicznemu charakterowi danych naukowych oraz ich różnorodności.

Skalowalność i Wydajność

W miarę wzrostu ilości generowanych i przetwarzanych danych, systemy zarządzania danymi muszą być w stanie skalować swoje zasoby i możliwości przetwarzania, aby zachować wysoką wydajność. W naszym przypadku wyzwaniem jest skalowanie systemów przy zapisie, odczytcie i modyfikacji danych. Konieczne jest, aby systemy te mogły efektywnie obsługiwać zarówno małe, jak i bardzo duże ilości rekordów (np. od 1 do 100 000) oraz różnorodne rozmiary rekordów (od 10 do nawet 1 000 000 pól w rekordzie).

Bezpieczeństwo i Prywatność

Zapewnienie bezpieczeństwa i integralności danych medycznych jest niezbędną składową w ich zarządzaniu, zwłaszcza w kontekście ich przechowywania i przetwarzania. Dyrektywa 2001/20/WE Parlamentu Europejskiego dotycząca prowadzenia badań klinicznych nakłada obowiązek stosowania odpowiednich środków technicznych i organizacyjnych w celu zapewnienia, że dane są zbierane, przechowywane i przetwarzane zgodnie z zasadami dobrej praktyki klinicznej oraz w sposób zapewniający ich bezpieczeństwo i poufność [40]. W Polsce, Ustawa z dnia 6 września 2001 r., czyli prawo farmaceutyczne, określa zasady przeprowadzania badań klinicznych oraz przechowywania danych medycznych w taki sposób, aby gwarantował ich integralność i ochronę przed nieuprawnionym dostępem [36].

Historia Zmian Danych

Śledzenie i poświadczanie pochodzenia danych oraz ich zmian jest potrzebne do zapewnienia reprodukowalności i rzetelności badań. Zwiększa to wiarygodność naukowców oraz umożliwia im dokładne odtworzenie eksperymentów i analiz, co jest niezbędne dla weryfikacji wyników i budowania zaufania do badań naukowych.

Rozwiązania

Aby sprostać tym wyzwaniom, konieczne jest zastosowanie zaawansowanych technologii i narzędzi wspierających zarządzanie Data Lineage i Provenance. Zaprezentowane wyżej technologie takie oferują rozwiązania, które mogą wspierać śledzenie pochodzenia danych, zapewniać ich integralność i bezpieczeństwo oraz ułatwiać reprodukowalność badań.

- **Blockchain:** Blockchain zapewnia niezmiennosć i transparentność danych dzięki zdecentralizowanemu rejestrowi, co pozwala na śledzenie historii każdej transakcji i zapewnia integralność danych [35]. Przykład zastosowania blockchaina w śledzeniu pochodzenia danych medycznych to projekt MedRec, który wykorzystuje technologię blockchain do zarządzania rejestrami medycznymi [20].
- **Systemy Chmurowe:** Systemy chmurowe, takie jak AWS, GCP i Azure, oferują skalowalne i elastyczne zasoby obliczeniowe, które mogą być dynamicznie dostosowywane do zmieniających się potrzeb. Dzięki technologii konteneryzacji, jak

Docker, oraz systemom orkiestracji, takim jak Kubernetes, możliwe jest efektywne zarządzanie i skalowanie aplikacji oraz danych [33, 10].

- **Zaawansowane DBMS:** Relacyjne bazy danych, takie jak Oracle i SQL Server, oraz nowoczesne bazy NoSQL, takie jak MongoDB i Cassandra, oferują zaawansowane funkcje zarządzania danymi, w tym skalowanie poziome i pionowe, replikację danych oraz mechanizmy zapewniające wysoką dostępność i niezawodność [13].
- **Narzędzia ETL:** Narzędzia do ekstrakcji, transformacji i ładowania danych (ETL), takie jak Apache NiFi, Talend i Informatica, umożliwiają automatyzację procesów integracji danych, co pozwala efektywnie zarządzać dużymi zbiorami danych i przekształcać je na potrzeby analiz [46].
- **Apache Kafka:** Kafka to rozproszona platforma strumieniowa, która umożliwia przetwarzanie i analizę danych w czasie rzeczywistym. Dzięki mechanizmom replikacji i skalowania, Kafka zapewnia wysoką dostępność i wydajność nawet dla dużych zbiorów danych medycznych [31, 24].

1.1.5 Wnioski Analizy

Analiza metod śledzenia i poświadczania pochodzenia obiektów cyfrowych w kontekście archiwizacji danych doświadczalnych ukazuje znaczenie tych procesów dla zapewnienia integralności i wiarygodności danych. Technologie takie jak blockchain, systemy chmurowe, zaawansowane DBMS oraz platformy strumieniowe, takie jak Apache Kafka, dostarczają rozwiązania wspierające śledzenie i poświadczanie pochodzenia danych. Każda z tych technologii ma swoje zalety i wady, które rozważono w kontekście serwisu do archiwizacji danych eksperymentalnych.

Blockchain wyróżnia się na tle innych metod swoją zdolnością do zapewnienia niezmienności zapisów, wysokiej odporności na manipulacje oraz transparentności. Dzięki zdecentralizowanej strukturze i kryptograficznym zabezpieczeniom, blockchain oferuje wyjątkowo wysoki poziom bezpieczeństwa i wiarygodności danych.

Systemy chmurowe, choć oferują skalowalność i elastyczność, mogą wiązać się z problemami dotyczącymi prywatności danych i zależnością od dostawcy usług. Zaawansowane DBMS zapewniają szeroki zakres funkcji do zarządzania danymi, ale ich wdrożenie może być kosztowne i złożone. Platformy strumieniowe są niezbędne do integracji i przetwarzania danych w czasie rzeczywistym, ale wymagają zaawansowanej konfiguracji i odpowiedniej infrastruktury.

W niniejszej pracy zdecydowano się na wykorzystanie technologii blockchain. Wybór ten uzasadnia wysoka niezawodność, bezpieczeństwo oraz transparentność technologii. Blockchain zapewnia integralność danych, ich wiarygodność oraz pełną weryfikowalność historii danych. Umożliwia dokładne śledzenie pochodzenia danych i zapewnia, że każda zmiana będzie rejestrowana i niezmienna, co jest niezbędne dla zachowania wysokiej jakości i rzetelności danych eksperymentalnych.

1.2 Bezpieczeństwo Danych w Blockchain

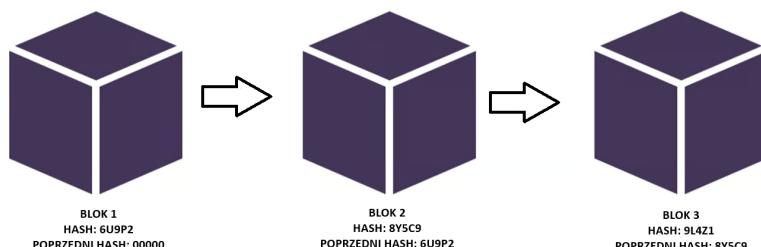
W większości przypadków, dla osób spoza dziedziny technologii, termin "blockchain" kojarzy się głównie z kryptowalutami. To wrażenie jest zrozumiałe, ponieważ początkowo został on stworzony jako infrastruktura dla kryptowalut. Niemniej jednak, obecnie blockchain jest kluczowym narzędziem w obszarze finansów cyfrowych i kontekście zabezpieczania danych. Jak opisuje Don Tapscott w 'Blockchain Revolution' [17], technologia ta ma potencjał do przekształcenia wielu branż, zapewniając bezpieczeństwo, przejrzystość i odporność systemów zarządzania danymi. Niniejsza sekcja przedstawia fundamentalne aspekty i zasady, które definiują tę technologię.

1.2.1 Blockchain - Łańcuch Bloków Informacji

Blockchain, tłumaczony dosłownie jako łańcuch bloków informacji to system, który bez wątpienia można określić jako jedną z najbardziej istotnych technologii dotyczących bezpieczeństwa danych. To nie tylko narzędzie, ale też rewolucyjna koncepcja, która zmienia sposób, w jaki przechowujemy dane, zarządzamy transakcjami i budujemy zaufanie w świecie cyfrowym.

Struktura Blockchain

Blockchain jest zdecentralizowanym systemem bazowanym, który umożliwia bezpieczne przechowywanie i szyfrowanie danych w postaci połączonych ze sobą bloków. Każdy blok zawiera unikalny identyfikator, zwany haszem, który generowany jest na podstawie danych zawartych w danym bloku oraz hasza poprzedniego bloku.



Idea Decentralizacji

Jednym z najważniejszych założeń technologii blockchain jest decentralizacja. Oznacza to, że przetwarzanie danych nie odbywa się w jednym systemie, ale na wielu pojedynczych jednostkach. Jednostki te nazywane są węzłami. Każdy węzeł przechowuje kopię łańcucha bloków. W ten sposób, awaria pojedynczego węzła nie wpływa na całą sieć. Dzięki temu blockchain jest bardziej odporny na awarie i ataki, co zwiększa jego niezawodność.

Zapewnienie Bezpieczeństwa Danych

Zapewnienie bezpieczeństwa danych jest głównym aspektem technologii blockchain. Każdy blok chroniony jest przez kryptograficzny hasz, który jest generowany na podsta-

wie zawartości oraz hasza poprzedniego bloku. Każda najmniejsza zmiana danych spowoduje zmianę hasza, co zostanie wykryte przez wszystkie węzły w sieci. W ten sposób sieć uniemożliwia modyfikację danych, co stanowi fundament dla bezpieczeństwa w blockchainie.

Podsumowanie

Blockchain to nie tylko łańcuch bloków informacji, ale także koncepcja zdecentralizowanego, niezmienialnego i bezpiecznego systemu, który ma duży potencjał w różnych dziedzinach. Jego struktura, oparta na blokach i unikalnych haszach, sprawia, że jest on odporny na ataki i awarie, co czyni go jednym z najważniejszych osiągnięć w dziedzinie technologii.

1.2.2 Historia Blockchain

Historia technologii blockchain sięga korzeniami do końca XX wieku. Poniżej zostały przedstawione kluczowe etapy i wydarzenia, które ukształtowały rozwój tej technologii.

Początek Idei

Blockchain jako podstawa kryptowalut, kojarzony jest przede wszystkim z Bitcoinem. Fakt ten tworzy przekonanie, że obie technologie powstały w tym samym czasie, w 2008 roku. Jednak pionier świata kryptowalut miał swój początek parę lat wcześniej, w 1982 roku. Amerykański fotograf David Chaum opublikował pracę "Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups" [14], w której przedstawił koncepcję tzw. "e-cash" i wraz z nim zarys bazy danych blockchain. W oparciu o swoją ideę Chaum w kolejnych latach założył firmę i stworzył własną kryptowalutę. Pomimo obiecujących efektów nie udało mu się przekonać do swojej koncepcji banków i w związku z tym projekt upadł.

Narodziny Bitcoinia

W 2008 roku na światło dziennie wyszedł dokument opisujący koncepcję blockchain w kontekście kryptowaluty Bitcoin. W 2009 roku Bitcoin został uruchomiony, co stanowiło początek ery blockchain jako rozproszonej i bezpiecznej technologii rejestru transakcji. Twórca kryptowaluty znany pod pseudonimem Satoshi Nakamoto pozostaje anonimowy do dziś.

Rozwój Zastosowań

W 2014 roku Ethereum zmieniło podejście do technologii blockchain, wprowadzając smart kontrakty. Blockchain przestał służyć już tylko do przesyłania kryptowalut i zaczął umożliwiać programistom pisanie autonomicznych programów, które automatycznie wykonywały określone działania na podstawie warunków. Skutkowało to początkiem wykorzystania blockchain w dziedzinach takich jak finanse, ubezpieczenia i zarządzanie łańcuchem dostaw. Dzięki Ethereum, blockchain przestał być jedynie narzędziem dla kryptowalut, a również potężnym narzędziem do tworzenia różnorodnych aplikacji.

Nowe Kierunki

Kolejne lata przyniosły reformę blockchaina poza obszar finansów. Technologia ta jest teraz badana i wdrażana w wielu dziedzinach takich jak logistyka, opieka zdrowotna, usługi publiczne czy nawet sztuka. W obecnej chwili, najbardziej istotne obszary rozwoju technologii blockchain to DeFi i NFT. DeFi to Finanse Zdecentralizowane, które przekształcają tradycyjne usługi finansowe, a NFT czyli Non-Fungible Tokens, rewolucjonizują sposób posiadania cyfrowych aktywów w dziedzinie sztuki i rozrywki.

1.2.3 Kluczowe Cechy Gwarantujące Bezpieczeństwo Danych

Technologia blockchain to nie tylko narzędzie finansowe, lecz kompleksowy system, który łączy w sobie elementy kryptografii, rozproszonej technologii bazodanowej i konsensusu. Jego celem jest stworzyć bezpieczny i niezmienny w swojej strukturze danych system. To rewolucyjne podejście do zarządzania informacjami, które wprowadza kilka kluczowych cech gwarantujących bezpieczeństwo i niemodyfikowalność informacji. Oto one:

Rozproszone Przechowywanie Informacji

Jedną z najważniejszych cech bezpieczeństwa danych w blockchain jest rozproszenie informacji pomiędzy wszystkimi uczestnikami sieci. Każdy z węzłów przechowuje kopię łańcucha bloków, eliminując konieczność centralnego przechowywania danych. To działanie sprawia, że pojedyncza awaria lub atak nie spowoduje awarii całej sieci.

Niemodyfikowalność Danych

Kolejnym filarem bezpieczeństwa blockchain jest niemodyfikowalność istniejących już danych. Jest to osiągane poprzez zastosowanie funkcji kryptograficznych, które generują *hash* dla każdego bloku, bazując na jego zawartości i identyfikatorze poprzedniego bloku. Nawet najdrobniejsza zmiana danych skutkuje zmianą hasza, co natychmiastowo jest wykrywane przez wszystkie węzły sieci.

Bezpieczeństwo Transakcji

W celu zachowania bezpieczeństwa transakcji, blockchain wykorzystuje mechanizmy kryptograficzne nazywane podpisami cyfrowymi. Każda transakcja musi być potwierdzona przez właściwy podpis. Takie działanie gwarantuje autentyczność użytkowników i zabezpiecza dane przed podrobieniem. Mechanizmy konsensusu, takie jak *Proof of Work* czy *Proof of Stake*, działają jako zabezpieczenie przed fałszowaniem transakcji i zabezpieczają sieć przed atakami.

Descentralizacja

Brak centralnej jednostki sprawia, że każdy węzeł w sieci ma równe sobie prawa i obowiązki. Descentralizacja to nie tylko zabezpieczenie przed monopolizacją zasobów

czy decyzji, ale także fundament demokratycznego funkcjonowania systemu. Każdy węzeł przyczynia się do procesu decyzyjnego, co zapewnia równowagę procesów przetwarzających.

1.2.4 Składowe Technologii Blockchain

Technologia blockchain jest niezawodna w swojej dziedzinie przez innowacyjną, strategiczną infrastrukturę wewnętrzną. Jej fundamentalne komponenty współpracują ze sobą, tworząc bezpieczną i niezmienialną strukturę danych.

Bloki

Podstawowym elementem w blockchainie jest blok. Blok to zestaw danych, który może zawierać informacje o wielu transakcjach. Każdy blok połączony jest z poprzednim i następnym blokiem za pomocą unikalnego identyfikatora. Tworzy to chronologiczną sekwencję bloków. W ramach bloku przechowywane są informacje, takie jak dane transakcyjne, znacznik czasu i hasz poprzedniego bloku.

Transakcje

Transakcje są fundamentalnym elementem blockchain. To właśnie dzięki nim następuje wymiana danych między uczestnikami sieci. Każda transakcja jest zapisywana w bloku, co zapewnia możliwość weryfikacji każdej operacji, przez całą sieć. Transakcje są również poddawane procesowi potwierdzania, zanim zostaną dodane do łańcucha bloków.

Łańcuch Bloków

Łańcuch bloków to sekwencja połączonych ze sobą bloków, które razem tworzą chronologiczną strukturę. Każdy nowy blok zawiera hasz poprzedniego bloku, co zapewnia spójność i niemożliwość modyfikacji danych w poprzednich blokach. Łańcuch bloków stanowi podstawę dla niezmienności danych w blockchainie.

Węzły

Węzeł w kontekście technologii blockchain pełni ważną rolę w funkcjonowaniu tej struktury. Jest to urządzenie podłączone do sieci blockchain, mające na celu uczestniczenie w procesie utrzymania, weryfikacji i zabezpieczania danych zawartych w łańcuchu bloków. Jego główne zadania obejmują przechowywanie kopii całego łańcucha bloków, weryfikowanie transakcji, a także udział w procesie tworzenia nowych bloków. Odgrywa istotną rolę w zapewnieniu bezpieczeństwa sieci. Każdy węzeł zawiera łańcuch bloków, co pozwala na zachowanie spójności danych oraz weryfikację każdej transakcji. Istnieją trzy typy węzłów w blockchainie, są to węzły pełne, lekkie oraz węzły masternode. Z każdym z nich związane są specyficzne funkcje i zadania.

Węzły Pełne

Pełne węzły przechowują cały łańcuch bloków, co umożliwia im pełną weryfikację transakcji i bloków. Uruchamiając pełny węzeł, uczestnicy mogą niezależnie zweryfikować

każdą transakcję w historii łańcucha bloków, bez potrzeby polegania na referencjach zewnętrznych. Ta cecha zapewnia, że sieć pozostaje zdecentralizowana oraz odporna na ataki i awarie.

Węzły Lekkie

Węzły lekkie są mniej wymagające od pełnych węzłów pod względem zasobów. Posiadają kopię niepełnego łańcucha bloków i są w stanie przeglądać transakcje oraz stan sieci, ale nie biorą udziału w procesie tworzenia nowych bloków ani w weryfikacji wszystkich transakcji.

Węzły Masternode

Węzły masternode to rodzaj węzła wykorzystywany głównie w sieciach blockchain, które obsługują smart kontrakty i różne zaawansowane funkcje. Posiadają one specjalne uprawnienia, takie jak udział w procesie głosowania w celu wprowadzenia zmian w protokole sieci lub zarządzanie zdeponowanymi środkami w smart kontraktach. Węzły masternode często wymagają większej ilości kryptowaluty jako zabezpieczenia.

1.2.5 Mechanizmy Konsensusu

Mechanizmy konsensusu pozwalają zapewnić spójność i bezpieczeństwo danych w blockchainie. Wybór konkretnego mechanizmu zależy od specyfiki danego projektu i jego wymagań. Oto kilka głównych mechanizmów konsensusu:

Proof of Work (PoW)

W przypadku PoW, uczestnicy sieci (górnicy) konkurują ze sobą, rozwiązyując skomplikowane matematyczne problemy. Pierwszy, który znajdzie poprawne rozwiązanie, uzyskuje prawo do dodania nowego bloku do łańcucha. Ten mechanizm wymaga dużej ilości mocy obliczeniowej i energii, co sprawia, że fałszowanie danych staje się praktycznie niemożliwe.

Proof of Stake (PoS)

Mechanizm Proof of Stake wykorzystywany jest głównie w sieciach publicznych, czyli kryptowalut. W PoS prawa do dodania nowego bloku do łańcucha przyznawane są na podstawie ilości kryptowaluty, jaką posiadają uczestnicy. Im większa jest ilość posiadanej kryptowaluty, tym większe jest prawdopodobieństwo zostania wybranym do utworzenia bloku. Mechanizm PoS potrzebuje mniej energii niż PoW, ale wymaga kryptowaluty jako zabezpieczenia.

Delegated Proof of Stake (DPoS)

W mechanizmie Delegated Proof of Stake właściciele kryptowaluty wybierają węzły, które reprezentują ich w procesie tworzenia bloków. Ten system jest bardziej skalowalny niż PoW czy PoS, ponieważ liczba uczestników jest ograniczona do wybranych węzłów.

Proof of Authority (PoA)

W mechanizmie Proof of Authority prawa do tworzenia bloków są przyznawane autorytetom lub instytucjom. Ze względu na fakt, że uczestnicy sieci muszą posiadać odpowiednie uprawnienia, mechanizm jest wykorzystywany szczególnie w prywatnych blockchainach.

1.2.6 Skalowalność w Blockchain

W blockchainie skalowalność oznacza, że sieć może zarządzać dużą ilością transakcji i użytkowników, zachowując jednocześnie stabilność i efektywność. Gwarantuje, że blockchain sprosta wymaganiom, szczególnie dla dużej liczby użytkowników. Istnieją różne sposoby na poprawę skalowalności w blockchainie:

Skalowalność Pozioma

Skalowalność pozioma to kosztowny i skomplikowany proces, który pozwala na elastyczne dostosowanie się do wzrostu transakcji. Polega na dodaniu nowych serwerów lub węzłów do sieci w celu zwiększenia jej przepustowości i mocy obliczeniowej.

Skalowalność Pionowa

Skalowalność pionowa jest kosztowna, ale bardzo skuteczna. Proces polega na poprawie wydajności poprzez zwiększenie mocy obliczeniowej jednego węzła lub serwera. Podejście to jest kosztowne, ale bardzo skuteczne, szczególnie w sytuacjach nagłego wzrostu obciążenia.

Skalowalność Warstwowa

Skalowalność warstwowa jest stosowana przeważnie w kryptowalutach, między innymi Bitcoin i Ethereum. Polega na dodawaniu dodatkowych warstw i protokołów do istniejącej sieci blockchain. Pozwala to na przetwarzanie transakcji poza głównym łańcuchem bloków.

1.2.7 Wydajność w Blockchain

Wydajność w blockchain oznacza czas, który jest potrzebny na przetwarzanie i potwierdzenie transakcji oraz tworzenie nowych bloków. Jest szczególnie ważna w przypadkach gdzie ważny jest czas, na przykład płatności lub dostawa danych w czasie rzeczywistym. Oto najważniejsze czynniki wpływające na wydajność w blockchain:

Mechanizm Konsensusu

Mechanizmy konsensusu mają bezpośredni wpływ na wydajność. Jej poziom zależy od wybranego mechanizmu. Na przykład mechanizmy takie jak Proof of Work (PoW) mogą być bardziej czasochłonne niż Proof of Stake (PoS) czy Delegated Proof of Stake (DPoS).

Rozmiar Bloków

Wydajność w blockchain zależy od rozmiaru bloków. Im blok jest większy tym więcej transakcji może pomieścić. Odbywa się to jednak większym kosztem mocy obliczeniowej

potrzebnej do przetwarzania transakcji. Optymalizacja rozmiaru bloków jest jednym z najważniejszych elementów poprawy wydajności.

Technologie Skalowania

Wydajność można również poprawić wprowadzając nowe technologie. Przykładami takowymi mogą być warstwy skrótu skracające weryfikację transakcji czy technologie strumieniowe i algorytmy kompresji danych, które zmniejszają rozmiar danych.

1.3 Nierelacyjne Bazy Danych

Nierelacyjne bazy danych, znane także jako bazy NoSQL, stanowią alternatywę dla tradycyjnych systemów bazodanowych opartych na modelu relacyjnym. Ich główną cechą jest elastyczność schematu, wysoka skalowalność oraz zdolność do efektywnego przechowywania i przetwarzania różnorodnych typów danych. Zostały zaprojektowane z myślą o nowoczesnych aplikacjach internetowych, które generują i przetwarzają ogromne ilości danych, często w formatach niestandardowych lub nieuporządkowanych.

1.3.1 Charakterystyka Nierelacyjnych Baz Danych

W nierelacyjnych bazach danych dane nie są przechowywane w ścisłe zdefiniowanych tabelach, a relacje między danymi nie są ustalane na poziomie struktury bazy. Pozwala to na łatwe dostosowanie się do dynamicznych wymagań aplikacji oraz umożliwia efektywne zarządzanie danymi niestrukturalnymi, takimi jak dokumenty, grafy, strumienie danych czy proste pary klucz-wartość.

Elastyczność Schematu

Nierelacyjne bazy danych oferują dynamikę schematu, co oznacza, że struktura danych może być modyfikowana "w locie", bez potrzeby przeprowadzania kosztownych i czasochłonnych operacji na bazie. Umożliwia to szybką adaptację aplikacji do zmieniających się wymagań biznesowych oraz łatwe zarządzanie różnorodnymi typami danych, co jest szczególnie istotne w aplikacjach webowych, mediach społecznościowych czy systemach IoT.

Skalowalność

Nierelacyjne systemy bazodanowe zaprojektowane są z myślą o łatwej skalowalności, zarówno poziomej, poprzez dodawanie nowych węzłów do klastra, jak i pionowej, poprzez zwiększanie zasobów pojedynczych węzłów. Taka architektura sprawia, że bazy NoSQL są idealnym rozwiązaniem dla aplikacji o dużym obciążeniu, wymagających przetwarzania wielkich zbiorów danych w rozproszonych środowiskach.

Wysoka Wydajność

Dzięki specjalizacji w obsłudze określonych modeli danych (np. klucz-wartość, dokumenty, grafy) oraz zastosowaniu zaawansowanych technik optymalizacji, takich jak indeksowanie czy cache'owanie, bazy NoSQL często oferują wyższą wydajność w specyficznych scenariuszach użytkowania. Jest to szczególnie widoczne w aplikacjach wyma-

gających szybkiego odczytu i zapisu dużej ilości danych, analizy dużych zbiorów danych w czasie rzeczywistym czy efektywnego przeszukiwania złożonych struktur danych.

1.3.2 Typy Nierelacyjnych Baz Danych

Nierelacyjne bazy danych, ze względu na swoją różnorodność i specyfikację, są podzielone na kilka głównych typów. Każdy z nich został zaprojektowany do efektywnego rozwiązywania konkretnych problemów i wyzwań związanych z przetwarzaniem oraz przechowywaniem danych w nowoczesnych aplikacjach. W niniejszym podrozdziale przedstawione są najpopularniejsze typy baz nierelacyjnych.

Bazy Dokumentowe

Bazy dokumentowe, takie jak MongoDB czy Couchbase, przechowują dane w formacie dokumentów, najczęściej JSON lub BSON, co pozwala na przechowywanie strukturyzowanych danych w sposób hierarchiczny. Taka organizacja danych sprzyja elastyczności i dynamice, umożliwiając łatwe dostosowanie struktury danych do bieżących potrzeb aplikacji. Bazy dokumentowe są szczególnie przydatne w aplikacjach webowych, systemach CMS czy platformach e-commerce, gdzie wymagana jest wysoka elastyczność schematu oraz efektywna praca z złożonymi, hierarchicznymi strukturami danych.

Bazy Klucz-Wartość

Bazy typu klucz-wartość, takie jak Redis czy DynamoDB, przechowują dane w postaci prostych par klucz-wartość. Charakteryzują się one bardzo wysoką wydajnością operacji odczytu i zapisu, co czyni je idealnym rozwiązaniem dla aplikacji wymagających szybkiego dostępu do danych, takich jak systemy cache, sesje użytkowników czy systemy kolejkowania. Są one również wykorzystywane w środowiskach, gdzie potrzebna jest szybka i efektywna obsługa dużych wolumenów danych, często w czasie rzeczywistym.

Bazy Grafowe

Bazy grafowe, np. Neo4j czy Amazon Neptune, specjalizują się w przechowywaniu danych w formie grafów, składających się z wierzchołków i krawędzi. Są one szczególnie przydatne w modelowaniu złożonych relacji między danymi, takich jak sieci społecznościowe, systemy rekomendacji, analizy sieciowe czy aplikacje, które wykorzystują algorytmy grafowe. Bazy grafowe umożliwiają efektywne przeszukiwanie, analizowanie i wizualizowanie złożonych struktur danych.

Bazy Kolumnowe

Bazy danych kolumnowe, takie jak Cassandra czy HBase, zostały zaprojektowane do efektywnego przechowywania i przetwarzania dużych zbiorów danych. Organizacja danych w kolumnach, a nie w wierszach, umożliwia szybkie wykonywanie operacji agregujących i analitycznych na dużych wolumenach danych, co jest niezwykle przydatne w systemach Big Data, hurtowniach danych czy aplikacjach wymagających efektywnego przetwarzania analiz statystycznych i raportowania.

1.3.3 Zastosowania Nierelacyjnych Baz Danych

Nierelacyjne bazy danych znalazły szerokie zastosowanie w różnych dziedzinach technologii informatycznych. Ich adaptacyjność do zmieniających się wymagań oraz możliwość szybkiego skalowania sprawiają, że są one idealnym wyborem dla wielu nowoczesnych aplikacji.

Big Data i Analiza Danych

W systemach gdzie ilość generowanych danych jest bardzo duża, nierelacyjne bazy danych stały się nieodzownym narzędziem do zarządzania złożonością Big Data. Umożliwiają one efektywne przetwarzanie ogromnych zbiorów nieuszkodzonych danych, co jest wykorzystywane w wielu sektorach - od finansów po ochronę zdrowia. Przetwarzanie takich danych wymaga technologii, które mogą sprostać wymaganiom związanych z objętością, prędkością i różnorodnością danych. Nierelacyjne bazy danych, z ich niezwykłą skalowalnością i elastycznością, pozwalają na dynamiczne adaptowanie się do zmiennych wymagań analizy danych, zapewniając jednocześnie szybki dostęp do informacji, które są niezbędne do podejmowania decyzji biznesowych.

Aplikacje Webowe i Mobilne

W dynamicznie rozwijającym się świecie aplikacji webowych i mobilnych, gdzie użytkownicy oczekują nie tylko intuicyjnych i estetycznych zaprojektowanych interfejsów, ale również błyskawicznej reakcji na swoje działania, nierelacyjne bazy danych oferują rozwiązanie. Dzięki ich elastyczności schematu, programiści mogą szybko adaptować aplikacje do zmieniających się wymagań, co jak podkreślają Fowler i Sadalage w swojej książce "*NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*" [22], jest kluczowe w utrzymaniu konkurencyjności i zadowolenia użytkowników. Nierelacyjne bazy danych, dzięki swojej optymalizacji pod kątem szybkich operacji odczytu i zapisu, umożliwiają aplikacjom mobilnym i webowym oferowanie płynnego doświadczenia użytkownika, nawet przy dużym obciążeniu systemu.

Systemy Rekomendacji i Sieci Społecznościowe

Bazy grafowe stały się nieocenionym narzędziem w sektorze mediów społecznościowych i systemów rekomendacji, gdzie zachodzą złożone relacje między użytkownikami, treściami i produktami. Ich unikatowa struktura, idealnie odwzorowująca sieci relacji, umożliwia tworzenie zaawansowanych algorytmów rekomendacji, które są sercem personalizacji treści. Jak wskazują Aggarwal i Zhai [3], zdolność do analizy złożonych sieci społecznościowych i personalizacji treści na podstawie interakcji użytkowników stanowi o sile systemów rekomendacji i jest niezbędna w utrzymaniu zaangażowania i satysfakcji użytkowników w dzisiejszych sieciach społecznościowych.

W kontekście zastosowań nierelacyjnych baz danych, od Big Data po systemy rekomendacji, istotne jest, aby pamiętać o ich specyficznych cechach i potencjalnych ograniczeniach. Wybierając odpowiednią technologię, należy kierować się charakterystyką projektu i specyficznymi wymaganiami danych, aby w pełni wykorzystać potencjał nierelacyjnych baz danych w danym kontekście.

1.3.4 Nierelacyjne Bazy Danych a Blockchain

Rozwój technologii blockchain często wiązany jest z koncepcjami nierelacyjnych baz danych ze względu na ich rozproszone, elastyczne i skalowalne charakterystyki. Jednakże, blockchain posiada unikalne cechy i mechanizmy, które odróżniają go od tradycyjnych systemów bazodanowych, nierelacyjnych włącznie. W tym podrozdziale zostaną przedstawione główne podobieństwa i różnice między nierelacyjnymi bazami danych a blockchainem, podkreślając specyficzną rolę i zastosowania tej technologii.

Podobieństwa Blockchain do Nierelacyjnych Baz Danych

Technologie blockchain i nierelacyjne bazy danych dzielą wiele wspólnych cech, które sprzyjają ich wykorzystaniu w różnorodnych środowiskach aplikacyjnych:

Rozproszenie Danych

Zarówno blockchain, jak i niektóre typy nierelacyjnych baz danych (np. bazy danych klucz-wartość, bazy danych kolumnowe) są zaprojektowane do działania w rozproszonych środowiskach. Dzięki temu zapewniają one większą odporność na awarie oraz możliwość skalowania poziomego, co jest najważniejszym aspektem systemów wymagających wysokiej dostępności.

Elastyczność Schematu

Blockchain, podobnie jak nierelacyjne bazy danych, może oferować pewną elastyczność w strukturze przechowywanych danych. Szczególnie widoczne jest to w zastosowaniach takich jak inteligentne kontrakty na platformach blockchain, które mogą manipułować różnymi rodzajami danych, podobnie do nierelacyjnych baz danych, które często nie wymagają ścisłego schematu.

Szybkość i Wydajność

W niektórych zastosowaniach, gdzie kluczowe jest szybkie przetwarzanie transakcji, zarówno blockchain jak i nierelacyjne bazy danych mogą oferować odpowiednią wydajność, choć w przypadku blockchaina może to zależeć od konkretnego mechanizmu konsensusu i innych czynników sieciowych.

Różnice Względem Nierelacyjnych Baz Danych

Pomimo znaczących podobieństw, obie technologie mają swoje unikatowe cechy, które podkreślają ich specyficzną rolę i możliwości.

Niezmienność Danych

Główna cecha, która wyróżnia blockchain, jest niezmienność zapisanych danych. Po zapisaniu, informacje przechowywane na blockchainie nie mogą być modyfikowane ani usuwane, co stanowi podstawę bezpieczeństwa i transparentności systemu. W nierelacyjnych bazach danych, dane mogą być zazwyczaj aktualizowane lub usuwane przez uprawnionych użytkowników.

Mechanizmy Konsensusu

Blockchain wykorzystuje zaawansowane protokoły bezpieczeństwa. Imran Bashir w 'Mastering Blockchain' [8] dokładnie wyjaśnia, jak znacząco różne mechanizmy konsensusu, w tym Proof of Work (PoW) i Proof of Stake (PoS), wpływają na bezpieczeństwo i wydajność sieci blockchain, podkreślając ich rolę w utrzymaniu niezmienności i zaufania między uczestnikami.

Zastosowanie Kryptografii

Bezpieczeństwo i integralność danych na blockchainie są zapewnione przez intensywne wykorzystanie technik kryptograficznych, włączając w to kryptografię klucza publicznego i funkcje haszujące. Chociaż nierelacyjne bazy danych również mogą implementować rozwiązania kryptograficzne, to jednak nie stanowią one tak integralnej części systemu, jak ma to miejsce w blockchainie.

Publiczne i Prywatne Sieci

Blockchain może funkcjonować zarówno jako sieć publiczna, dostępna dla każdego (np. Bitcoin, Ethereum), jak i jako sieć prywatna, ograniczona do określonej grupy uczestników. Nierelacyjne bazy danych są zazwyczaj kontrolowane przez pojedynczą organizację lub ograniczoną grupę użytkowników.

Podsumowując, chociaż blockchain i nierelacyjne bazy danych dzielą kilka istotnych dla baz danych cech, to jednak blockchain wyróżnia się unikalnymi mechanizmami, które przeważają nad nierelacyjnymi bazami danych zapewniając bezpieczeństwo, niezmienność i transparentność danych. Te cechy czynią go odpowiednim rozwiązaniem dla zastosowań wymagających wysokiego poziomu zaufania i weryfikalności, takich jak systemy rejestrów, audytu czy niniejsza praca polegająca na archiwizacji danych eksperymentalnych.

1.4 Przechowywanie Danych Medycznych

Przechowywanie danych medycznych odgrywa bardzo ważną rolę w diagnostyce i analizie danych zdrowotnych. Dobre zarządzanie danymi umożliwia przeprowadzanie skutecznych analiz, które pozwalają identyfikować wzorce chorób i efektywność leczenia. W diagnostyce, dostęp do wyników badań i historii medycznej pacjenta jest niezbędny. Dane medyczne często stanowią źródło do prowadzenia badań epidemiologicznych i klinicznych.

1.4.1 Znaczenie Przechowywania Danych Medycznych

Przechowywanie danych medycznych ma ogromne znaczenie. Wykorzystywane jest między innymi w opiece zdrowotnej i analizie danych medycznych. Ich archiwizacja jest ważna dla badań naukowych, ponieważ umożliwia analizę dużych zbiorów danych.

Rola Danych Medycznych w Diagnozowaniu i Leczeniu

Systematyczne gromadzenie i zarządzanie danymi umożliwia tworzenie szczegółowych i kompleksowych profili zdrowotnych pacjentów. Dostęp do tych danych pozwala

lekierzom na dokładną analizę stanu zdrowia pacjenta, co umożliwia skuteczne planowanie i monitoring leczenia.

Wpływ Dokładności i Dostępności Danych na Jakość Diagnostyki i Analizy

Dla zapewnienia wysokiej jakości diagnostyki i analizy niezbędna jest dokładność i dostępność przechowywanych danych. Błędy lub ograniczony do nich dostęp mogą prowadzić do nieprawidłowej diagnozy lub niewłaściwego leczenia. Właściwe przechowywanie danych zapewnia ich integralność, co zwiększa rzetelność analizy i interpretacji. Zastosowanie blockchain znacznie przyczynia się do zwiększenia bezpieczeństwa danych, co jest istotne w dziedzinie badań medycznych.

1.4.2 Regulacje Prawne w Polsce i Europie

Regulacje prawne dotyczące przechowywania danych medycznych mają szczególne znaczenie w sektorze opieki zdrowotnej, tym bardziej w czasach dynamicznego rozwoju technologii. Ochrona danych medycznych stanowi fundament zaufania systemów stosowanych w medycynie.

Rozporządzenie Ogólne o Ochronie Danych (RODO)

Wprowadzone w Unii Europejskiej w 2018 roku Rozporządzenie Ogólne o Ochronie Danych stanowi podstawę w ochronie danych osobowych obywateli UE. RODO zbiera przepisy dotyczące ochrony danych osobowych i zapewnia mieszkańcom UE większą kontrolę nad ich danymi osobowymi. Dane medyczne w RODO uznawane są za kategorię szczególnych danych osobowych. Oznacza to, że ich przetwarzanie podlega surowym ograniczeniom.

W kontekście niniejszego projektu RODO nie ma bezpośredniego zastosowania, ponieważ archiwizowane dane nie zawierają danych osobowych. Należy jednak pamiętać, że każde przetwarzanie danych medycznych powinno przestrzegać prywatności danych i podejmować odpowiednie środki techniczne i organizacyjne, by zapewnić ich bezpieczeństwo.

Ustawa o Prawach Pacjenta i Rzeczniku Praw Pacjenta

Ustawa o Prawach Pacjenta i Rzeczniku Praw Pacjenta jest aktem prawnym regulującym kwestie związane z ochroną danych medycznych w Polsce. Ustawa ta określa prawa pacjentów, w tym prawo do informacji o stanie zdrowia, dostępu do dokumentacji medycznej oraz ochrony danych osobowych. Choć ustawa ta koncentruje się głównie na prawach pacjentów, ma również znaczenie dla systemów przechowywania danych medycznych, wymagając, aby były one traktowane z należytą starannością i zgodnie z obowiązującymi przepisami o ochronie danych osobowych.

W kontekście projektu, ustawa ta może mieć ograniczone zastosowanie. Jednakże, zrozumienie jej postanowień jest istotne, aby zapewnić, że wszelkie przyszłe modyfikacje lub usprawnienia systemu będą zgodne z prawnymi wymogami ochrony praw pacjentów.

Specyfika Regulacji Sektorowych w Ochronie Zdrowia

Specyfika Regulacji Sektorowych w Ochronie Zdrowia zawiera szereg regulacji sektorowych w prawie polskim, które wprowadzają szczegółowe wymagania dotyczące przechowywania i przetwarzania danych w sektorze zdrowia. Przykładowo, Rozporządzenie

Ministra Zdrowia w sprawie dokumentacji medycznej określa rodzaje, zakres i wzory dokumentacji medycznej oraz sposoby jej przetwarzania. Dla projektu skoncentrowanego na archiwizacji danych medycznych bez danych osobowych, zrozumienie tych wymogów pozwoli zapewnić zgodność z najlepszymi praktykami i standardami sektorowymi.

1.4.3 Procedury Zgodności z Regulacjami Prawnymi

W celu zapewnienia zgodności z regulacjami prawnymi dotyczącymi przechowywania danych medycznych, należy wdrożyć następujące procedury:

Zgodność z Polskim Prawem

W polskim prawie zasady przetwarzania i przechowywania danych medycznych reguluje ustanowiona o systemie informacji w ochronie zdrowia. Określa ona wymagania dotyczące systemów informatycznych w ochronie zdrowia. Oto normy techniczne i organizacyjne według tej ustawy:

- **Bezpieczeństwo fizyczne i środowiskowe:** Systemy informatyczne muszą być zabezpieczone przed fizycznym dostępem osób nieupoważnionych oraz przed zagrożeniami środowiskowymi, takimi jak pożar lub zalanie.
- **Kontrola dostępu:** Zastosowanie odpowiednich mechanizmów uwierzytelniania w celu ograniczenia dostępu do systemów informatycznych tylko dla upoważnionych pracowników.
- **Zarządzanie incydentami:** Opracowanie i wdrożenie procedur reagowania na incydenty bezpieczeństwa, w tym ich rejestracja, analiza oraz podejmowanie działań naprawczych.
- **Szyfrowanie danych:** Dane medyczne muszą być szyfrowane, aby zapewnić ich poufność i integralność.
- **Kopie zapasowe:** Tworzenie kopii zapasowych danych, aby zapewnić możliwość ich odzyskania w przypadku awarii systemu.
- **Szkolenia i świadomość:** Regularne szkolenia dla personelu dotyczące bezpieczeństwa danych.

Więcej informacji na temat ustawy można znaleźć na stronie Sejmu RP [37].

Audyty i Kontrola Zgodności

W celu utrzymania wysokich standardów bezpieczeństwa należy przeprowadzać regularne audyty i kontrole zgodności systemu z regulacjami prawnymi. Procedury te mogą obejmować:

- **Audyty wewnętrzne:** Regularne sprawdzanie systemu przez wewnętrzny zespół ds. zgodności. Mają na celu znalezienie potencjalnych luk w zabezpieczeniach oraz ocenę skuteczności ochrony danych.
- **Audyty zewnętrzne:** Przeprowadzane przez niezależne jednostki certyfikujące. Potwierdzają zgodność z regulacjami i standardami. Dostarczają obiektywną ocenę systemu oraz rekomendacje dotyczące dalszych usprawnień.

1.4.4 Metody Przechowywania Danych Medycznych

Przechowywanie danych medycznych jest ważnym elementem zarządzania informacjami w sektorze opieki zdrowotnej. Zastosowanie odpowiednich metod przechowywania ma fundamentalne znaczenie dla zachowania integralności, bezpieczeństwa oraz dostępności danych. Współczesne podejścia do przechowywania danych medycznych obejmują zarówno tradycyjne systemy papierowe, jak i zaawansowane rozwiązania cyfrowe.

Systemy Papierowe

Mimo postępu technologicznego, w wielu obszarach opieki zdrowotnej nadal stosowane są tradycyjne systemy papierowe. Ich główną zaletą jest prostota i bezpośredni dostęp bez potrzeby stosowania urządzeń elektronicznych. Jednakże, mają liczne ograniczenia, w tym ryzyko uszkodzenia fizycznego, trudności w szybkim wyszukiwaniu informacji oraz problemy z przechowywaniem dużych ilości dokumentów.

Elektroniczne Zbiory Danych (EHR/EMR)

Elektroniczne Rejestry Zdrowia (EHR) i Elektroniczne Rejestry Medyczne (EMR) to cyfrowe systemy przechowywania danych, które umożliwiają gromadzenie, zarządzanie i udostępnianie informacji zdrowotnych pacjentów w bezpieczny i efektywny sposób. Systemy te oferują liczne korzyści, w tym łatwy dostęp do danych, możliwość szybkiego udostępniania informacji między specjalistami, a także lepszą ochronę danych dzięki za stosowaniu zaawansowanych technologii szyfrowania.

Chmura Obliczeniowa i Przechowywanie Hybrydowe

Rozwiązania oparte na chmurze obliczeniowej i przechowywanie hybrydowe stają się coraz bardziej popularne w sektorze opieki zdrowotnej. Chmura oferuje skalowalność, elastyczność oraz możliwość dostępu do danych z dowolnego miejsca, co jest szczególnie ważne dla placówek o rozproszonej strukturze. Przechowywanie hybrydowe łączy zalety lokalnych systemów przechowywania danych z elastycznością i skalowalnością chmury, oferując zrównoważone rozwiązanie dla zróżnicowanych potrzeb sektora zdrowia.

Systemy Blockchain

Zastosowanie blockchaina w sektorze zdrowia umożliwia stworzenie bezpiecznego i transparentnego systemu zarządzania danymi medycznymi, gdzie udostępnianie danych między uprawnionymi podmiotami odbywa się w sposób szyfrowany i w pełni zabezpieczony. Jak podaje Daniel Drescher [19], zastosowanie blockchain może zrewolucjonizować przechowywanie danych medycznych, nie tylko poprzez zwiększenie bezpieczeństwa i dostępności, ale również poprzez zapewnienie pełnej przejrzystości historii danych medycznych. Technologia ta wspiera automatyzację procesów opieki zdrowotnej za pomocą inteligentnych kontraktów, które uruchamiają określone działania, gdy spełnione są zdefiniowane warunki, co przyczynia się do zwiększenia efektywności i ograniczenia ryzyka błędów.

1.4.5 Wyzwania i Rozwiązania Przechowywania Danych Medycznych

Przechowywanie danych medycznych wiąże się z szeregiem wyzwań, które wymagają kompleksowych rozwiązań w celu zapewnienia bezpieczeństwa, prywatności i do-

stępności danych.

Bezpieczeństwo i Prywatność Danych

Bezpieczeństwo i ochrona prywatności danych medycznych są głównymi wyzwaniem w kontekście przechowywania informacji zdrowotnych. Ryzyko naruszenia danych, cyberataków oraz nieautoryzowanego dostępu wymaga zastosowania zaawansowanych środków bezpieczeństwa, takich jak szyfrowanie danych, uwierzytelnianie wieloskładnikowe oraz regularne audyty bezpieczeństwa.

Zgodność z Regulacjami Prawnymi

Zapewnienie zgodności z lokalnymi i międzynarodowymi regulacjami prawnymi, takimi jak RODO, jest kolejnym wyzwaniem. Organizacje muszą wdrożyć odpowiednie polityki i procedury, aby spełnić wymagania prawne dotyczące ochrony danych medycznych, w tym zarządzanie zgodnością, rejestrowanie zgód i zarządzanie dostępem do danych.

Integracja Systemów i Interoperacyjność

Integracja różnorodnych systemów przechowywania danych oraz zapewnienie ich interoperacyjności to kluczowe wyzwanie dla zapewnienia ciągłości opieki i efektywnej współpracy między różnymi podmiotami sektora zdrowia. Rozwiązania takie jak ujednolicone standardy wymiany danych, otwarte interfejsy API oraz współpraca między dostawcami technologii są niezbędne dla osiągnięcia pełnej integralności.

Skalowalność i Zarządzanie Dużymi Zbiorami Danych

Wzrost ilości generowanych danych medycznych wymaga rozwiązań, które są skalowalne i mogą efektywnie zarządzać dużymi zbiorami danych. Technologie takie jak chmura obliczeniowa, big data i sztuczna inteligencja oferują nowe możliwości w zakresie przetwarzania, analizy i przechowywania danych, umożliwiając lepsze wykorzystanie informacji zdrowotnych w celach diagnostycznych, terapeutycznych i badawczych.

1.5 Wprowadzenie Nowych Procedur Medycznych na Rynek

Ze względu na możliwość komercjalizacji systemu, który w przyszłości zostanie rozwinięty o innowacyjny sposób analizy przechowywanych danych, należy przyjrzeć się wprowadzaniu nowych procedur medycznych na rynek. Takie przedsięwzięcie wymaga rygorystycznego podejścia do śledzenia i poświadczania pochodzenia danych. Oba te aspekty są fundamentem dla podejmowania decyzji klinicznych oraz zatwierdzania nowych procedur medycznych. Zgodność ze związanymi z tym regulacjami prawnymi zapewnia nie tylko wiarygodność produktu, ale również zabezpieczenie przed błędem ludzkim czy niedopatrzeniem, które mogłyby wywołać poważne konsekwencje prawne.

1.5.1 ISO 8000 - Jakość Danych

ISO 8000 jest międzynarodowym standardem dotyczącym jakości danych. Skupia się na aspektach dotyczących dokładności danych. Standard ten wymaga, aby dane były gromadzone, przechowywane i przetwarzane w sposób zapewniający ich najwyższą jakość [28].

Wymogi Standardu

- **Integralność Danych:** Dane muszą być dokładne i niezmienne od momentu ich zebrania do momentu ich użycia.
- **Kompletność Danych:** Wszystkie niezbędne informacje muszą być dostępne i możliwe do zweryfikowania.
- **Dokładność Danych:** Dane muszą być precyzyjne i wolne od błędów.
- **Aktualność Danych:** Dane muszą być aktualizowane w odpowiednich odstępach czasu, aby odzwierciedlać najnowszy stan rzeczy.

Jak Aplikacja Radzi Sobie z Wymogami ISO 8000

Aplikacja rejestruje wszystkie operacje na danych i prowadzi ich historię, co zapewnia ich integralność i możliwość audytu. Pozwala na precyzyjne śledzenie każdej zmiany danych od momentu ich stworzenia po ostatnią przeprowadzoną operację. Aktualność danych i ich stan zależy od użytkowników, ponieważ to oni będą te dane wprowadzać, aplikacja umożliwia w tej kwestii jedynie sprawdzenie dokładnej daty ich utworzenia i zmian.

1.5.2 ISO 27001 - Zarządzanie Bezpieczeństwem Informacji

ISO 27001 jest międzynarodowym standardem zarządzania bezpieczeństwem informacji. Skupia się na ochronie informacji przed utratą, kradzieżą i nieautoryzowanym dostępem poprzez wdrożenie odpowiednich środków zabezpieczających [27].

Wymogi Standardu

- **Ocena Ryzyka:** Identyfikacja i ocena ryzyk związanych z przetwarzaniem danych.
- **Środki Zabezpieczające:** Wdrożenie odpowiednich środków technicznych i organizacyjnych w celu ochrony danych.
- **Kontrola Dostępu:** Zapewnienie, że tylko uprawnione osoby mają dostęp do danych.
- **Audyt i Monitorowanie:** Regularne audyty i monitorowanie systemów w celu wykrycia i zapobiegania naruszeniom bezpieczeństwa.

Jak Aplikacja Radzi Sobie z Wymogami ISO 27001

Aplikacja spełnia wymogi ISO 27001 dzięki zastosowaniu technologii blockchain, która z natury jest bardzo bezpieczna. Umożliwia przechowywanie danych w sposób zdecentralizowany, co znacząco utrudnia ich nieautoryzowaną modyfikację lub usunięcie. Wszelkie operacje na danych są chronione za pomocą zaawansowanych mechanizmów szyfrowania, a każda zmiana jest rejestrowana i weryfikowana. Kontrola dostępu jest realizowana poprzez przypisywanie ról i uprawnień autoryzowanym użytkownikom.

1.5.3 ISO 13485 - System Zarządzania Jakością dla Wyrobów Medycznych

ISO 13485 jest międzynarodowym standardem dotyczącym systemów zarządzania jakością w wyrobach medycznych. Określa wymagania dotyczące procesów, które muszą być spełnione przez organizacje produkujące wyroby medyczne [30].

Wymogi Standardu

- **Dokumentacja Procesów:** Szczegółowa dokumentacja wszystkich procesów związanych z projektowaniem, produkcją i dystrybucją wyrobów medycznych.
- **Śledzenie i Kontrola:** Mechanizmy śledzenia i kontroli wszystkich etapów produkcji i dystrybucji.
- **Zarządzanie Ryzykiem:** Identyfikacja i zarządzanie ryzykiem związanym z produkcją wyrobów medycznych.
- **Ocena Kliniczna:** Przeprowadzanie badań klinicznych w celu potwierdzenia bezpieczeństwa i skuteczności wyrobu.

Jak Aplikacja Radzi Sobie z Wymogami ISO 13485

Poprzez zastosowanie blockchain każdy etap przetwarzania danych jest rejestrowany i możliwy do odtworzenia, co zapewnia pełne śledzenie pochodzenia danych. Dane z badań klinicznych są przechowywane w systemie, co umożliwia łatwe udostępnianie wyników i ich weryfikację przez organy regulacyjne lub inne zainteresowane strony. W ramach zarządzania ryzykiem, aplikacja automatyzuje procesy przechowywania i dodawania danych, czym zastępuje tradycyjne procesy, takie jak prowadzenie dokumentacji papierowej (zmniejszenie ryzyka błędu ludzkiego). Warto jednak wziąć pod uwagę implementację gotowych narzędzi do identyfikacji i minimalizacji ryzyka dostępnych na rynku. Wiele z nich tworzonych jest przez doświadczonych specjalistów, którzy cieszą się w branży nienaganną renomą, co może być równoznaczne ze zmniejszeniem potencjalnego ryzyka do poziomu bliskiego zera.

1.5.4 ISO 27040 - Bezpieczeństwo Przechowywania Danych

ISO 27040 jest standardem, który określa wytyczne dotyczące bezpiecznego przechowywania danych. Skupia się na ochronie danych przed nieautoryzowanym dostępem i utratą [29].

Wymogi Standardu

- **Bezpieczne Przechowywanie:** Dane muszą być przechowywane w sposób zabezpieczony przed nieautoryzowanym dostępem i utratą.
- **Szyfrowanie Danych:** Stosowanie odpowiednich metod szyfrowania w celu ochrony danych w spoczynku i podczas przesyłania.
- **Odzyskiwanie Danych:** Mechanizmy zapewniające szybkie i skuteczne odzyskiwanie danych w przypadku awarii.

Jak Aplikacja Radzi Sobie z Wymogami ISO 27040

Blockchain zapewnia bezpieczne przechowywanie danych. Są one przechowywane w zdecentralizowanych bazach danych, co eliminuje ryzyko związane z centralnym punktem awarii. Dodatkowo, dane są szyfrowane zarówno w spoczynku, jak i podczas przesyłania, co zapewnia ich bezpieczeństwo przed nieautoryzowanym dostępem. System przechowuje kopie zapasowe danych, co umożliwia ich szybkie odzyskanie w przypadku awarii.

1.5.5 IEEE 802 - Zarządzanie Metadanymi

IEEE 802 jest standardem dotyczącym zarządzania metadanymi, które są jednym z najważniejszych aspektów śledzenia pochodzenia i zmian w danych [26].

Wymogi Standardu

- **Dokumentacja Metadanych:** Wszystkie dane muszą być opatrzone odpowiednimi metadanymi, które umożliwiają ich identyfikację i śledzenie.
- **Aktualizacja Metadanych:** Metadane muszą być aktualizowane w czasie rzeczywistym, aby odzwierciedlać wszelkie zmiany w danych.
- **Dostępność Metadanych:** Metadane muszą być dostępne dla uprawnionych użytkowników w celu weryfikacji i audytu.

Jak Aplikacja Radzi Sobie z Wymogami IEEE 802

Aplikacja stosuje kompleksowy system zarządzania metadanymi. Każdy wpis w systemie jest opatrzony danymi, które zawierają informacje o źródle danych, czasie ich stworzenia i szczegółach wersji. Dostęp do tych danych mają tylko uprawnieni użytkownicy, co zapewnia ich bezpieczeństwo i możliwość audytu.

1.5.6 GxP (Good Practice)

GxP to ogólny termin obejmujący wiele wytycznych dotyczących dobrych praktyk w przemyśle farmaceutycznym i opiece zdrowotnej. Dotyczą one wszystkich aspektów produkcji i kontroli jakości, w tym także zarządzania danymi [41].

Wymogi Standardu

- **Dobra Praktyka Dokumentacyjna (GDP):** Wszystkie procesy i dane muszą być dokładnie dokumentowane.
- **Dobra Praktyka Laboratoryjna (GLP):** Standardy dotyczące jakości i integralności badań laboratoryjnych.
- **Dobra Praktyka Kliniczna (GCP):** Standardy dotyczące jakości badań klinicznych, w tym zarządzania danymi pacjentów.

Jak Aplikacja Radzi Sobie z Wymogami GxP

Aplikacja spełnia wymogi GxP poprzez dokładną dokumentację wszystkich procesów i operacji na danych. Dzięki technologii blockchain, dane są zabezpieczone, a ich integralność gwarantowana. Dane pacjentów są bezpieczne i poufne, ponieważ system sam w sobie ich nie przechowuje. Dane zawierają jedynie identyfikator pacjenta, który umożliwia jego identyfikację tylko poprzez sprawdzenie dokumentacji dostawcy danych, co jest niemożliwe z poziomu niniejszego systemu.

1.5.7 Podsumowanie

Proces wprowadzania nowych procedur medycznych na rynek wymaga spełnienia wymogów regulacyjnych dotyczących śledzenia i poświadczania pochodzenia danych. Niniejszy rozdział przedstawia, że stworzona aplikacja spełnia te wymogi. Jednakże, z pewnością nie robi tego nad wyraz i warto rozważyć zastosowanie specjalistycznych narzędzi, które zapewniają stuprocentową zgodność z poszczególnymi standardami. Na rynku możemy znaleźć do tego wiele oprogramowań, na przykład Vanta, który monituruje zgodność z ISO 27001 lub Veeva Systems sprawdzający praktyki GxP.

2. Technologie

2.1 Hyperledger Fabric

Hyperledger Fabric, stworzony przez IBM i Digital Assets z ramienia projektu Hyperledger, jest projektem open-source, zaprojektowanym dla zastosowań biznesowych. Jego elastyczna architektura umożliwia implementację komponentów dostosowanych do specyficznych potrzeb danego przedsiębiorstwa. Ważnym aspektem jest możliwość używania różnych języków programowania do tworzenia smart kontraktów dzięki wykorzystaniu technologii kontenerowej Docker [32].

2.1.1 Architektura Hyperledger Fabric

Architektura Hyperledger Fabric jest złożona i modułowa, co pozwala na łatwiejsze dostosowanie systemów do specyficznych potrzeb. Jest to system zbudowany w celu wspierania aplikacji biznesowych. Oferuje unikalne funkcje, takie jak zarządzanie tożsamościami, prywatność transakcji oraz wydajne mechanizmy konsensusu. Modułowa konstrukcja Fabric pozwala na niezależne rozwijanie komponentów systemu, takich jak mechanizm konsensusu, system członkostwa czy logika smart kontraktów. Taka struktura zapewnia elastyczność, skalowalność i bezpieczeństwo wymagane w zastosowaniach korporacyjnych.

Bardzo klarowny opis biblioteki Hyperledger Fabric został przedstawiony przez Androulakiego i współpracowników [4], poniższy opis opiera się na tej pracy. Wprowadza ona i szczegółowo opisuje najważniejsze komponenty oraz mechanizmy tej technologii.

Podstawowe Komponenty

Podstawowe komponenty Hyperledger Fabric obejmują Peers, Ordering Service i Membership Service Provider (MSP). Peers są węzłami sieci, które zatwierdzają transakcje, utrzymującą stan sieci i przechowującą historię transakcji w postaci łańcucha bloków. Ordering Service zarządza kolejnością transakcji w sieci i gwarantuje ich dostarczenie do wszystkich peerów w odpowiedniej kolejności. MSP zarządza tożsamościami użytkowników i ich uprawnieniami. Chaincode, zwany również smart kontraktem, to kod aplikacji uruchamiany na blockchainie, który definiuje zasady biznesowe dla systemu.

Kanały

Kanały są prywatnymi, ukrytymi sieciami komunikacyjnymi między wybranymi uczestnikami sieci. Pozwalają one na ograniczenie dostępu do pewnych transakcji tylko dla

określonych uczestników. Jest to szczególnie przydatne w przypadku, gdy w sieci blockchain uczestniczą konkurencyjne podmioty biznesowe, które nie chcą ujawniać wszystkich swoich transakcji wszystkim innym uczestnikom sieci. Kanały zapewniają także zwiększoną prywatność i bezpieczeństwo danych.

Zarządzanie Tożsamościami

Ważną częścią architektury Hyperledger Fabric jest mechanizm zarządzania tożsamościami, który jest zaimplementowany poprzez dostawców usług członkostwa (Membership Service Providers - MSPs). MSPs kontrolują proces uwierzytelniania i autoryzacji węzłów sieci oraz użytkowników. Dzięki temu mechanizmowi możliwe jest nadanie odpowiednich uprawnień do wykonywania określonych operacji na blockchainie, co zapewnia bezpieczeństwo i integralność danych.

Zarządzanie Transakcjami

Hyperledger Fabric oferuje zaawansowane mechanizmy zarządzania transakcjami, w tym możliwość prywatnych transakcji. Dzięki temu, nawet jeśli dane transakcje są zapisywane na blockchainie, mogą być dostępne tylko dla wybranych uczestników sieci. To podejście zwiększa prywatność uczestników oraz pozwala na implementację bardziej złożonych przypadków użycia, które wymagają ograniczonego udostępniania danych.

Modułowość i Elastyczność

Cała architektura Hyperledger Fabric została zaprojektowana z myślą o modułowości i elastyczności. Dzięki temu każdy element systemu, począwszy od mechanizmu konsensusu, poprzez zarządzanie tożsamościami, aż po logikę smart kontraktów, może być łatwo dostosowany do konkretnych wymagań biznesowych. To sprawia, że Hyperledger Fabric jest atrakcyjnym rozwiązaniem dla różnorodnych przypadków użycia, zwłaszcza w sektorze korporacyjnym, gdzie wymagana jest wysoka elastyczność i skalowalność systemu blockchain.

2.1.2 Dlaczego Hyperledger Fabric?

Hyperledger Fabric wyróżnia się na tle innych technologii blockchain ze względu na szereg cech, które są istotne w kontekście tworzenia serwisu do archiwizacji danych medycznych. Poniżej przedstawiono najważniejsze z nich:

Modułowa Architektura

Modułowa architektura Hyperledger Fabric pozwala na elastyczne dostosowanie różnych komponentów do konkretnych potrzeb biznesowych. Każdy element systemu, od mechanizmu konsensusu po logikę smart kontraktów, może być łatwo modyfikowany i rozwijany niezależnie. Dzięki temu przedsiębiorstwa mogą budować rozwiązania blockchainowe, które idealnie odpowiadają ich wymaganiom.

Zarządzanie Tożsamościami

Mechanizm zarządzania tożsamościami w Hyperledger Fabric pozwala na precyzyjną kontrolę dostępu do sieci blockchainowej. Dzięki systemowi MSP (Membership Service Provider), administratorzy mogą nadawać różne uprawnienia użytkownikom i węzłom sieci, co zapewnia bezpieczeństwo i poufność danych.

Prywatność Transakcji

Hyperledger Fabric umożliwia implementację prywatnych transakcji, co oznacza, że pewne dane transakcyjne mogą być dostępne tylko dla wybranych uczestników sieci. To szczególnie istotne w przypadku współpracy różnych podmiotów, które chcą zachować poufność swoich operacji.

Wsparcie dla Różnych Języków Programowania

Hyperledger Fabric umożliwia pisanie smart kontraktów w różnych językach programowania, co ułatwia integrację z istniejącymi systemami oraz dostosowanie platformy do umiejętności programistycznych zespołu. Dzięki temu przedsiębiorstwa mogą korzystać z istniejących zasobów i doświadczenia programistycznego.

Wsparcie Dokumentacyjne

Hyperledger Fabric oferuje bogatą dokumentację, zawierającą przykłady użycia, tutoriale oraz szczegółowe instrukcje dotyczące konfiguracji i zarządzania platformą. Dostępność kompleksowych materiałów edukacyjnych ułatwia pracę z platformą dla osób zarówno początkujących, jak i doświadczonych deweloperów.

2.1.3 Zastosowania

Hyperledger Fabric znajduje zastosowanie w różnych dziedzinach, co potwierdza jej wszechstronność i uniwersalność. Możliwość dostosowania architektury, wysoki poziom prywatności i bezpieczeństwa oraz elastyczność inteligentnych kontraktów sprawiają, że platforma jest atrakcyjnym rozwiązaniem dla projektów opartych na technologii blockchain, w tym dla serwisów do archiwizacji danych medycznych.

Łańcuch Dostaw

W sektorze zarządzania łańcuchem dostaw Hyperledger Fabric pomaga w śledzeniu produktów od producenta do konsumenta, co pozwala na lepsze zarządzanie zapasami, poprawę autentyczności produktów oraz szybsze reagowanie na ewentualne problemy w łańcuchu dostaw.

Finanse

Sektor finansowy wykorzystuje Hyperledger Fabric do rozliczeń międzybankowych, zarządzania aktywami, weryfikacji tożsamości klientów oraz do wydawania cyfrowych walut i tokenów.

Medycyna

W sektorze medycznym Hyperledger Fabric zapewnia bezpieczeństwo danych medycznych i wyników badań. Dzięki technologii blockchain możliwe jest śledzenie historii operacji, zapisywanie wyników badań i udostępnianie ich tylko uprawnionym osobom, co zapewnia wysoką prywatność danych medycznych.

Nieruchomości

W branży nieruchomości, Hyperledger Fabric jest wykorzystywany do rejestracji transakcji kupna-sprzedaży nieruchomości, zarządzania umowami najmu oraz do śledzenia historii nieruchomości.

2.2 Flask

Flask to lekki, minimalistyczny framework internetowy dla języka Python, który umożliwia szybkie tworzenie aplikacji webowych. Jego prostota i elastyczność sprawiają, że jest popularnym wyborem zarówno dla początkujących, jak i doświadczonych programistów.

2.2.1 Charakterystyczne Cechy Flaska

Flask wyróżnia się kilkoma cechami, które czynią go atrakcyjnym wyborem do tworzenia aplikacji internetowych, w tym również w niniejszym projekcie:

Minimalistyczna Architektura

Flask oferuje minimalistyczną architekturę, co sprawia, że jest łatwy w nauce i zrozumieniu. Nie posiada zbędnych zależności i zapewnia tylko niezbędne narzędzia do budowy aplikacji.

Elastyczność

Framework umożliwia elastyczne dostosowywanie aplikacji do konkretnych potrzeb poprzez modułową strukturę i bogaty system rozszerzeń. Do rozbudowania funkcjonalności aplikacji można zarówno wybrać je spośród wielu dostępnych rozszerzeń jak i tworzyć własne.

Integracja z Językiem Python

Flask doskonale integruje się z językiem Python, wykorzystując jego zalety, takie jak czytelność kodu, bogate biblioteki i wsparcie społeczności. Znajomość Pythona powoduje łatwe przyswojenie składni Flaska i pozwala szybko rozpocząć pracę nad projektem.

2.2.2 Dlaczego Flask?

Flask wyróżnia się na tle innych frameworków internetowych z kilku powodów:

Czytelność Kodu

Flask oferuje czytelną i zwięzłą składnię, co przyspiesza proces tworzenia aplikacji i ułatwia utrzymanie kodu. Dzięki minimalistycznej architekturze, programista może skupić się na rozwiązywaniu problemów biznesowych, zamiast tracić czas na konfigurację skomplikowanych struktur.

Modułowość

Framework ten umożliwia łatwe dodawanie i usuwanie funkcjonalności poprzez moduły i rozszerzenia, co daje programistom pełną kontrolę nad aplikacją. Nie ma potrzeby korzystania z całego zestawu funkcji dostarczanych przez framework - programista mogą wybierać tylko te elementy, które są mu potrzebne.

Wsparcie dla Zewnętrznych API

Flask umożliwia łatwą integrację z zewnętrznymi API poprzez wbudowane narzędzia do obsługi żądań HTTP. Możliwe jest budowanie aplikacji, które komunikują się z innymi serwisami internetowymi, takimi jak płatności online, serwisy społecznościowe itp.

2.2.3 Zastosowania Flaska

Flask znajduje zastosowanie w różnych obszarach tworzenia aplikacji internetowych:

Strony Internetowe i Blogi

Flask jest często wybierany do budowy prostych stron internetowych, blogów i portfólio online. Dzięki swojej prostocie i elastyczności, jest doskonałym narzędziem dla osób, które chcą publikować swoje treści w internecie.

Aplikacje Internetowe

Flask nadaje się również do budowy bardziej zaawansowanych aplikacji internetowych, takich jak platformy e-commerce, aplikacje społecznościowe czy systemy zarządzania treścią. Dzięki bogatemu systemowi rozszerzeń, programista może rozbudowywać funkcjonalność swojej aplikacji według własnych potrzeb.

Interfejsy API

Flask jest popularnym wyborem dla tworzenia interfejsów API dla aplikacji mobilnych i webowych. Dzięki wbudowanym narzędziom do obsługi żądań HTTP, możliwe jest szybkie i łatwe budowanie API, które umożliwiają komunikację między różnymi serwisami internetowymi.

Mikroserwisy

W architekturze mikroserwisów, Flask może być używany do budowy pojedynczych usług, które wykonują określone zadania lub funkcje. Dzięki swojej lekkości i elastyczności, Flask jest doskonałym wyborem dla mikroserwisów, które wymagają szybkiego rozwoju i iteracji.

2.2.4 Przykład Użycia

Poniżej przedstawiony jest prosty przykład aplikacji Flask:

```
from flask import Flask
app = Flask(name)

@app.route('/')
def hello_world():
    return 'Hello, World!'

if name == 'main':
    app.run()
```

W powyższym przykładzie definiujemy prostą aplikację Flask, która zwraca napis "Hello, World!" po wejściu na główną stronę.

3. Serwis Archiwizacji Danych Eksperimentalnych

3.1 Diagram Przypadków Użycia

Diagram przypadków użycia przedstawiony na Rysunku 2.1 ilustruje interakcje pomiędzy użytkownikami a systemem zarządzania danymi eksperimentalnymi w kontekście szpitala. Zamierzonymi użytkownikami są tutaj: Pracownik szpitala, Gość, Administrator oraz Analityk, który w niniejszym projekcie nie będzie miał określonych zadań i uprawnień, ponieważ jest to kwestia do ustalenia przy integracji serwisu z projektem odpowiadającym za analizę archiwizowanych danych.

3.1.1 Aktorzy Aplikacji

Pracownik Szpitala

Pracownik szpitala ma możliwość wykonywania następujących czynności:

- Dodawanie danych eksperimentalnych – Pracownik może wprowadzać nowe dane do systemu.
- Edycja danych eksperimentalnych – Pracownik ma możliwość modyfikacji już istniejących danych.
- Eksportowanie danych eksperimentalnych – Możliwość eksportowania danych do różnych formatów.
- Wyłączenie danych eksperimentalnych – Pracownik może wyłączyć dane z dalszego użycia.
- Zarządzanie kontem – Obejmuje zmianę hasła oraz zarządzanie własnym profilem użytkownika.
- Wyświetlanie danych eksperimentalnych – Pracownik ma dostęp do przeglądania danych.

Gość

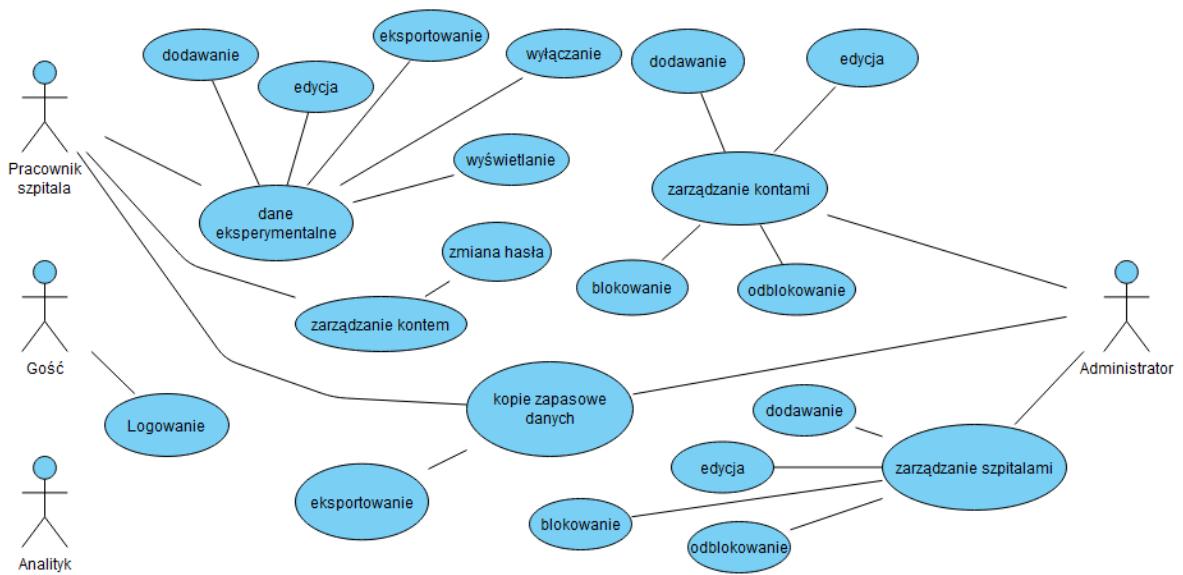
Gość, czyli użytkownik nieposiadający konta w systemie, ma ograniczone możliwości:

- Logowanie – Może jedynie zalogować się do systemu, jeśli posiada odpowiednie dane dostępowe.

Administrator

Administrator systemu posiada najszerzy zakres uprawnień, w tym:

- Zarządzanie kontami – Może tworzyć, modyfikować oraz blokować konta użytkowników.
- Dodawanie danych eksperymentalnych – Tak jak pracownik szpitala, może dodawać nowe dane.
- Edycja danych eksperymentalnych – Ma możliwość edytowania wszystkich danych w systemie.
- Zarządzanie szpitalami – Administrator może zarządzać listą szpitali współpracujących z systemem.
- Odblokowywanie danych eksperymentalnych – Ma możliwość odblokowania danych zablokowanych przez innych użytkowników.



Rysunek 3.1: Diagram przypadków użycia dla systemu zarządzania danymi eksperymentalnymi.

3.2 Architektura Systemu

System składa się z dwóch komponentów technologicznych: Hyperledger Fabric, który funkcjonuje jako rozproszona baza danych zabezpieczająca dane eksperymentalne, oraz Flask, który stanowi interfejs użytkownika, umożliwiając zarządzanie danymi i administrację.

Architektura Hyperledger Fabric

Hyperledger Fabric w systemie wykorzystywany jest jako REST API. Nie posiada graficznego interfejsu, oferuje endpointy do dodawania, zmieniania, pobierania i usuwania danych, które w praktyce nie jest trwałym usunięciem, a jedynie zmianą statusu, pozwalającą wykluczyć dane z analizy. Te endpointy współpracują z interfejsem Flask, umożliwiając pełną funkcjonalność systemu. Wszelkie funkcjonalności logiczne zostały zaimplementowane z wykorzystaniem języka TypeScript, który jest dedykowany do prac związanych z sieciową integracją systemów. Zastosowanie prywatnej sieci blockchain pozwala na utrzymanie wysokiego poziomu bezpieczeństwa i kontroli nad dostępem do danych, co jest szczególnie ważne w obszarze medycznym, gdzie ochrona informacji ma zasadnicze znaczenie. System pozwala na efektywne wdrażanie smart kontraktów, które automatyzują procesy biznesowe i zabezpieczają operacje zgodnie z wewnętrznymi regulaminami i procedurami. Dzięki izolacji danych w obrębie prywatnych kanałów, Hyperledger Fabric zapewnia pełną ochronę i prywatność przetwarzanych informacji.

Architektura Flask

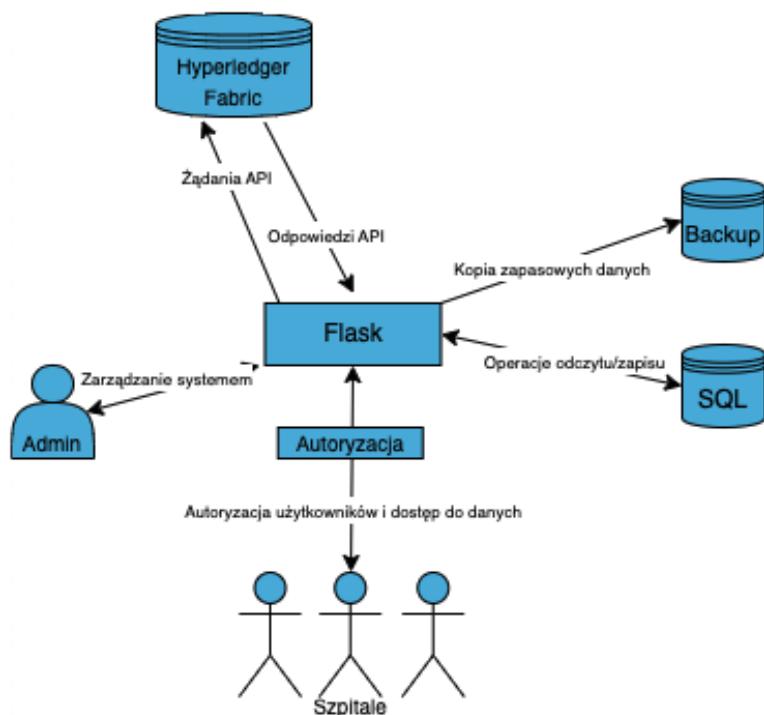
Flask pełni rolę interfejsu użytkownika i zarządzania danymi, zapewniając niezbędne funkcjonalności dla użytkowników aplikacji i jej administratorów. Flask jest lekkim frameworkiem webowym w języku Python, który umożliwia szybkie tworzenie aplikacji webowych.

- **Interakcja z Hyperledger Fabric:** Flask komunikuje się z Hyperledger Fabric poprzez REST API, umożliwiając użytkownikom wykonywanie operacji na danych zapisanych w blockchain. Flask realizuje zapytania do API, które są następnie przetwarzane przez Hyperledger Fabric.
- **Zarządzanie użytkownikami i rolami:** Flask oferuje kompleksowe zarządzanie użytkownikami, rolami i szpitalami. Administratorzy systemu mają możliwość tworzenia nowych użytkowników, przypisywania im określonych szpitali oraz ról w ramach systemu. Informacje te są przechowywane w bazie danych SQL.
- **Backup danych:** Flask spełnia rolę warstwy backupu, przechowując lokalne kopie danych. Lokalne przechowywanie danych pozwala na szybką reakcję w przypadku awarii lub potrzeby przywrócenia danych, zwiększając odporność systemu na potencjalne problemy.

Architektura ta umożliwia efektywne zarządzanie danymi medycznymi, zapewniając nie tylko integralność i prywatność danych, ale również intuicyjny w użytkowaniu interfejs graficzny. Dzięki połączeniu możliwości Hyperledger Fabric i Flask, system jest w stanie sprostać wymaganiom związanym z przetwarzaniem danych medycznych oraz zapewniać zgodność z odpowiednimi standardami bezpieczeństwa.

Diagram Architektury Systemu

Zamieszczony diagram przedstawia zależności między komponentami systemu, połączenia między modelem Hyperledger Fabric a aplikacją Flask oraz sposób przepływu danych. Diagram pokazuje, jak te elementy współpracują, aby efektywnie zarządzać danymi medycznymi. Hyperledger Fabric odgrywa centralną rolę w zabezpieczaniu transakcji i danych, wykorzystując technologię blockchain do ich niezmiennego rejestrowania. Flask, służący jako interfejs użytkownika, umożliwia administratorom zarządzanie użytkownikami, szpitalami i rolami, a także obsługuje zapytania i operacje danych. Diagram podkreśla również zastosowane mechanizmy bezpieczeństwa, w tym szyfrowanie danych i autoryzację, które są niezbędne do ochrony informacji przed nieautoryzowanym dostępem.



Rysunek 3.2: Diagram Architektury Systemu

3.3 Implementacja Systemu Hyperledger Fabric

Hyperledger Fabric umożliwia implementację smart kontraktów w kilku językach programowania:

- Go.
- Java
- JavaScript

- Typescript

Warstwa REST API, która również jest dostarczona przez framework dostępne są języki Go i TypeScript. Ze względu na czytelność, możliwość wykorzystania narzędzi deweloperskich JavaScript i typowanie danych wybrany został język TypeScript. Umożliwia on integrację z istniejącym kodem aplikacji oraz zapewnia dużą wydajność i bezpieczeństwo podczas wykonywania operacji.

3.3.1 Konfiguracja Projektu

Konfiguracja sieci Hyperledger Fabric jest głównym etapem w procesie tworzenia rozproszonego rejestru blockchain. Proces ten obejmuje przygotowanie środowiska zarówno dla sieci blockchain jak i dla serwera REST API zastosowanego do interakcji z siecią. W niniejszym projekcie wykorzystana została wersja oprogramowania 2.5.

Instalacja Środowiska

Hyperledger Fabric jest dedykowany dla użytkowników systemów macOS i Linux. Pierwszym krokiem do stworzenia środowiska była instalacja niezbędnych narzędzi, które pozwoliły na dalszą pracę z frameworkm. Są to:

- Git
- Curl
- Docker
- Jq
- Node.js
- NPM

Po zainstalowaniu narzędzi należało pobrać plik instalacyjny projektu i wykorzystać go do utworzenia środowiska pracy. Odbyło się to w następujący sposób:

```
mkdir fabric
cd fabric
curl -sSL0 https://raw.githubusercontent.com/hyperledger/fabric/main/scripts
/install-fabric.sh
chmod +x install-fabric.sh
./install-fabric.sh
```

Uruchomienie Sieci Blockchain

Łańcuch bloków jest instalowany i wdrażany do sieci przy pomocy skryptów dostarczonych przez platformę Hyperledger Fabric. Proces ten obejmuje:

- Uruchomienie sieci Hyperledger Fabric
- Stworzenie kanału komunikacyjnego

- Wdrożenie chaincode do sieci

Odbiera się to w następujący sposób:

```
./network.sh up
./network.sh up createChannel
./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-typescript
-ccl typescript
```

Konfiguracja Węzłów

W konfiguracji węzłów dla Hyperledger Fabric zdefiniowane są zmienne środowiskowe, które umożliwiają komunikację i zarządzanie węzłem peer w sieci blockchain. Oto opis ustawień, które są konfigurowane przez poszczególne komendy:

```
export CORE_PEER_TLS_ENABLED=true
```

Ta komenda aktywuje użycie TLS (Transport Layer Security) dla węzła peer, co zwiększa bezpieczeństwo komunikacji poprzez szyfrowanie wszystkich przesyłanych danych.

```
export CORE_PEER_LOCALMSPID="Org1MSP"
```

Ta zmienna definiuje lokalny identyfikator dostawcy usług członkowskich (MSP ID) dla węzła, co umożliwia jednoznaczne określenie, do której organizacji należy dany węzeł w sieci.

```
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations
/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
```

Określa lokalizację certyfikatu root CA (Certificate Authority) używanego do weryfikacji połączeń TLS. To zapewnia, że połączenia są akceptowane tylko jeśli pochodzą od zaufanych źródeł z odpowiednim certyfikatem.

```
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations
/org1.example.com/users/Admin@org1.example.com/msp
```

Ustawia ścieżkę do folderu MSP, który zawiera wszystkie istotne certyfikaty i klucze konieczne do uwierzytelnienia i autoryzacji węzła w sieci.

```
export CORE_PEER_ADDRESS=localhost:7051
```

Definiuje adres IP i port, na którym węzeł peer nasłuchuje połączeń. To umożliwia innym węzłom i aplikacjom klienta łączenie się z tym peerem.

Uruchomienie REST API

W celu uruchomienia REST API w projekcie Hyperledger Fabric, konieczne jest przygotowanie środowiska, zainstalowanie zależności oraz odpowiednia konfiguracja serwera. Poniżej opisano kroki, które należy wykonać, aby pomyślnie uruchomić REST API:

```
npm install
```

Ta komenda instaluje wszystkie zależności wymienione w pliku package.json projektu, które są niezbędne do jego działania. Instalacja zależności jest pierwszym krokiem do uruchomienia aplikacji.

```
npm run build
```

Komenda ta uruchamia proces budowania aplikacji, który kompliuje kod źródłowy do formatu, który może być wykonywany na serwerze.

```
TEST_NETWORK_HOME=$HOME/fabric/fabric-samples/test-network npm run generateEnv
```

Zmienna środowiskowa TEST_NETWORK_HOME jest ustawiana na ścieżkę do katalogu testowej sieci w instalacji Hyperledger Fabric. Następnie, uruchamia się skrypt generateEnv, który generuje pliki konfiguracyjne potrzebne do prawidłowego działania REST API w kontekście danej sieci.

```
export REDIS_PASSWORD=$(uuidgen)
```

Generuje unikalne hasło dla bazy danych Redis, używanej jako część infrastruktury backendowej, i eksportuje je jako zmienną środowiskową. Użycie unikalnego hasła zwiększa bezpieczeństwo aplikacji.

```
npm run start:redis
```

Uruchamia serwer Redis lokalnie, który jest wykorzystywany przez REST API do przechowywania danych sesji i cache'owania.

```
npm run start:dev
```

Uruchamia serwer developerski REST API, umożliwiając testowanie i rozwijanie aplikacji w środowisku developerskim.

3.3.2 Struktura Systemu

Projekt oparty na Hyperledger Fabric skupia się na stworzeniu zdecentralizowanego systemu do zarządzania danymi medycznymi. W ramach systemu zdefiniowano klasę Asset, które reprezentują zasoby medyczne przechowywane w blockchainie, klasę AssetTransferContract, która zawiera logikę transakcji i plik assets.router.ts odpowiadający za routing.

Model Danych

Model danych reprezentowany jest przez klasę Asset. Definiuje ona strukturę obiektu, który odzwierciedla przekazywane dane medyczne. Każde pole deklarowane jest dekoratorem @Property(), co jest niezbędne dla integracji z Hyperledger Fabric i jego systemem typów:

```
@Object()
export class Asset {
    @Property()
    public docType?: string;

    @Property()
    public ID: string;

    @Property()
    public UserId: number;

    @Property()
    public FirstName: string;

    @Property()
    public LastName: string;

    @Property()
    public Hospital: string;

    @Property()
    public Data: string;

    @Property()
    public IsDeleted: boolean;

    @Property()
    public IsEdited: number;

    @Property()
    public ParentId?: string;

    @Property()
```

```

    public Version: number;

    @Property()
    public Description: string;

    @Property()
    public CreatedAt: string;
}

```

Dzięki precyzyjnie zdefiniowanej strukturze, system jest w stanie efektywnie zarządzać danymi. Struktura ta umożliwia łatwe rozszerzanie modelu danych o nowe pola, bez zakłócania działania istniejącego systemu, co jest istotne w przypadku rozwoju.

Logika Transakcji

Implementacja smart kontraktów w aplikacji zawiera metody zarządzające cyklem życia zasobów medycznych. Każda transakcja jest zaimplementowana w sposób zapewniający integralność i audytowalność danych.

```

@Transaction()
public async UpdateAsset(ctx: Context, id: string, newData: string): Promise<void>
{
    const asset: Asset = JSON.parse(await this.ReadAsset(ctx, id));
    asset.Data = newData;
    await ctx.stub.putState(id, Buffer.from(stringify(sortKeysRecursive(asset))));
}

```

Ta metoda aktualizacji zasobu pokazuje, jak system radzi sobie z modyfikacjami danych, zachowując przy tym historię zmian.

```

@Transaction()
public async DeleteAsset(ctx: Context, id: string): Promise<void> {
    const assetString = await this.ReadAsset(ctx, id);
    const asset = JSON.parse(assetString);
    asset.IsDeleted = !asset.IsDeleted;
    await ctx.stub.putState(id, Buffer.from(stringify(sortKeysRecursive(asset))));
}

```

Metoda DeleteAsset przedstawia mechanizm miękkiego usuwania danych z systemu. Zamiast trwałego usuwania zasobów, funkcja ta zmienia status zasobu na "usunięty". To podejście pozwala na zachowanie historii danych dla celów audytu i przeglądów regulacyjnych.

Te przykłady funkcji transakcyjnych pokazują, jak system zarządza danymi medycznymi, umożliwiając ich bezpieczne przechowywanie, aktualizację i odpowiednie zarządzanie dostępem do informacji. Implementacja takich funkcji w ramach systemu Hyperledger Fabric gwarantuje, że wszystkie operacje na danych są śledzone, co zapewnia zgodność z zasadami Data Lineage i Provenance.

Bezpieczeństwo i Prywatność

Mechanizmy bezpieczeństwa, takie jak kryptograficzne funkcje hashujące używane w blockchainie, zapewniają ochronę przed nieautoryzowanymi zmianami danych. Kod odpowiedzialny za umieszczanie danych w blockchainie zawsze używa funkcji stringify oraz sortKeysRecursive, co gwarantuje deterministyczność i niezmienność danych:

```
await ctx.stub.putState(asset.ID, Buffer.from(stringify(sortKeysRecursive(asset))))
```

3.4 Implementacja w Technologii Flask

Implementacja systemu do zarządzania danymi w oparciu o framework Flask umożliwia skuteczne i elastyczne zarządzanie komponentami aplikacji webowej. Flask jest lekkim narzędziem do tworzenia aplikacji internetowych w języku Python, które ułatwia zarządzanie danymi, autoryzację użytkowników oraz interakcje z bazami danych i zewnętrznymi serwisami.

3.4.1 Architektura Aplikacji

Aplikacja została zaprojektowana z uwzględnieniem modułowości, bezpieczeństwa i łatwości utrzymania. Wykorzystany został wzorzec MVC, który jest typowy dla nowoczesnych aplikacji webowych.

Interfejs Użytkownika

Do implementacji interfejsu użytkownika wykorzystane zostały szablony Flask (Jinja2), pozwalające na renderowanie widoków, które interaktywnie komunikują się z backendem za pomocą formularzy HTML i żądań AJAX. Frontend odpowiada za prezentację danych użytkownikom i zbieranie danych wejściowych za pomocą formularzy i interfejsów interaktywnych.

Logika Aplikacji

Logika aplikacji została zaimplementowana w języku Python, z wykorzystaniem frameworka Flask. Backend obsługuje wszystkie żądania HTTP, zarządzanie sesjami, autoryzację użytkowników oraz interakcje z bazami danych. Stanowi podstawę systemu, jest w nim zdefiniowany routing, który odpowiada na żądania klienta takie jak logowanie czy operacje na danych.

Zarządzanie Stanem Sesji

Zarządzanie sesją w Flask jest realizowane za pomocą Flask-Login dla autoryzacji użytkowników i zarządzania aktywnymi sesjami. Moduł ten pozwala na łatwe śledzenie zalogowanych użytkowników i ogranicza dostęp do zasobów na podstawie ich statusu uwierzytelnienia.

Baza Danych

Aplikacja wykorzystuje SQLAlchemy jako ORM (Object-Relational Mapping), co umożliwia abstrakcyjne i efektywne zarządzanie relacyjną bazą danych. Przechowuje obiekty modeli danych, takie jak użytkownicy, role, szpitale i historia transakcji.

Bezpieczeństwo

Bezpieczeństwo jest kluczowym aspektem systemu, zarówno na poziomie komunikacji (HTTPS), jak i aplikacji. Flask-WTF i WTForms są wykorzystywane do walidacji danych wejściowych i zapobiegania atakom typu Cross-Site Scripting (XSS) oraz Cross-Site Request Forgery (CSRF). Ponadto, system posiada własne funkcje haszowania i weryfikacji haseł.

Elastyczność

System jest zaprojektowany modularnie, co ułatwia dodawanie nowych funkcjonalności oraz utrzymanie istniejącego kodu. Podział na pliki i moduły (np. forms.py, models.py, imports.py) pozwala na czytelne zarządzanie zależnościami i funkcjonalnościami.

3.4.2 Konfiguracja Środowiska

Konfiguracja aplikacji w Flask jest określona w pliku konfiguracyjnym, który zawiera parametry, takie jak URI bazy danych, sekretny klucz aplikacji oraz dane dostępu do API.

```
SQLALCHEMY_DATABASE_URI='sqlite:///Users\\dawidmetelski\\Desktop\\medical-data-store-management-flask\\data\\database.db'
SQLALCHEMY_TRACK_MODIFICATIONS=False
SECRET_KEY = 'nk5Z] [!ZNd0InuI'
API_URL='http://localhost:3000/api'
API_KEY='62C3F8F0-4D87-4F3B-ACBE-8B1B0C7DAD95'
```

Proces inicjalizacji aplikacji obejmuje przygotowanie struktury bazy danych za pomocą SQLAlchemy, co umożliwia zarządzanie modelami danych oraz ich relacjami.

```
app = Flask(__name__)
app.config.from_pyfile('config.cfg')
db = SQLAlchemy(app)
```

3.4.3 Zarządzanie Użytkownikami

System umożliwia zarządzanie użytkownikami, w tym logowanie, rejestrację, edycję i blokowanie kont. Formularze do tych operacji są zaimplementowane przy użyciu rozszerzenia Flask-WTF, które ułatwia walidację danych wejściowych oraz zwiększa bezpieczeństwo poprzez zapobieganie atakom typu CSRF.

Rejestracja Użytkowników

Proces rejestracji użytkowników jest pierwszym krokiem do umożliwienia interakcji z aplikacją. Implementacja rejestracji obejmuje formularz rejestracyjny, validację wprowadzonych danych oraz dodanie nowego konta do bazy danych. Tworzenie użytkownika możliwe jest tylko z poziomu administratora.

```
@app.route('/new_user', methods=['GET', 'POST'])  
@login_required  
@is_admin_required  
...  
...
```

Logowanie Użytkowników

Proces logowania umożliwia użytkownikom uzyskanie dostępu do swoich kont. Jest to jedyna funkcjonalność udostępniona dla nieautoryzowanych użytkowników. Po pozwolonym uwierzytelnieniu użytkownik otrzymuje dostęp do chronionych komponentów aplikacji takich jak profil i zarządzanie danymi.

```
class LoginForm(FlaskForm):  
    name = StringField('Nazwa użytkownika')  
    password = PasswordField('Hasło')  
    remember = BooleanField('Remember me')
```

Edycja Użytkowników

Proces edycji danych użytkowników obejmuje operacje na danych konta, takie jak przypisywanie roli, szpitala oraz podstawowych danych osobowych. Umożliwia administratorom systemu aktualizację informacji o użytkownikach w celu zapewnienia dokładności oraz spójności danych. Administratorzy posiadają uprawnienia do edycji danych użytkowników, co umożliwia im efektywne zarządzanie kontami oraz dostosowanie uprawnień do zmieniających się potrzeb. Właściciel konta ma jedynie możliwość zmiany hasła.

Blokowanie Użytkowników

Blokowanie użytkowników jest alternatywą dla usuwania kont z systemu. Polega na dezaktywacji konta użytkownika, co uniemożliwia mu dostęp do aplikacji i jej zasobów. Jest to istotne w przypadku wystąpienia sytuacji, w których konieczne jest zawieszenie dostępu do konta, np. w przypadku naruszenia regulaminu, podejrzenia nieautoryzowanej aktywności lub zakończenia współpracy z danym pracownikiem. Administratorzy posiadają uprawnienia do blokowania i aktywacji kont, co umożliwia im skuteczne zarządzanie bezpieczeństwem systemu oraz minimalizowanie ryzyka potencjalnych zagrożeń.

3.4.4 Zarządzanie Danymi

Zarządzanie danymi w systemie zrealizowane jest za pomocą interfejsu API Hyperledger Fabric, co pozwala na bezpieczne i efektywne operacje na danych zapisanych w blockchain. Dzięki temu podejęciu, aplikacja Flask może korzystać z rozproszonych i niezmienialnych cech blockchainu, zwiększając tym samym bezpieczeństwo i niezawodność systemu.

Komunikacja z Hyperledger Fabric

Aplikacja komunikuje się z Hyperledger Fabric poprzez API REST, wykorzystując bibliotekę requests do wysyłania i odbierania żądań HTTP.

```
def get_assets_from_api():
    api_url = app.config.get('API_URL')
    api_key = app.config.get('API_KEY')

    headers = {'X-Api-Key': api_key}
    response = requests.get(api_url + '/assets', headers = headers)

    if response.status_code == 200:
        return response.json()
    else:
        return None
```

Przetwarzanie i Przechowywanie Danych

Przetwarzanie i przechowywanie danych odbywa się poprzez serię operacji zapisu i odczytu z blockchain, co zapewnia ich trwałość i niezmiennosć. Kod aplikacji odpowiedzialny za przesyłanie danych do blockchaina wykorzystuje struktury danych JSON do organizacji informacji przed ich zapisem. Przykład funkcji przetwarzającej i wysyłającej dane:

```
def new_data():
    form = CSVUploadForm()
    if form.validate_on_submit():
        file = form.csv_file.data
        timestamp = datetime.now().strftime('%Y%m%d%H%M%S')
        filename = secure_filename(f"{timestamp}_{file.filename}")
        file.save(os.path.join('files', filename))

        # Odczyt i konwersja pliku CSV do JSON
        csv_file_path = os.path.join('files', filename)
        with open(csv_file_path, 'r', encoding='utf-8') as csv_file:
            csv_reader = csv.DictReader(csv_file)
            json_data = json.dumps([row for row in csv_reader])

        # Wysyłanie danych do API
        api_url = app.config.get('API_URL')
        api_key = app.config.get('API_KEY')
```

```

        headers = {'X-Api-Key': api_key}
        payload = {
            "UserID": current_user.id,
            "Data": json_data,
            "Description": form.description.data
        }
        response = requests.post(api_url + '/assets', data=payload,
                                 headers=headers)

    if response.status_code == 200:
        flash('Dane zostały zapisane w blockchainie.', 'success')
    else:
        flash('Wystąpił błąd podczas zapisu danych.', 'error')

    return render_template('data/new_data.html', form=form)

```

Historia Transakcji

Historia transakcji w systemie jest realizowana poprzez zapisywanie każdej operacji CRUD wykonanej na danych w Hyperledger Fabric, co gwarantuje niezmienność i nie-zaprzecalność zapisanych informacji. Dla każdej transakcji zapisywane są następujące informacje:

- Unikalny identyfikator rekordu historii.
- Klucz obcy identyfikujący użytkownika, który dokonał zmiany.
- Nazwa pliku, który został dodany lub zmodyfikowany.
- Data i czas, kiedy zmiana została dokonana.
- Klucz obcy wskazujący szpital, w którym użytkownik był zalogowany podczas dokonywania zmiany.

```

class History(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    filename = db.Column(db.String(100), unique=True)
    date = db.Column(db.DateTime)
    hospital_id = db.Column(db.Integer, db.ForeignKey('hospital.id'))

```

Obsługa błędów i Bezpieczeństwo

Aplikacja zapewnia obsługę błędów związanych z żądaniami HTTP, reagując na różne kody odpowiedzi od API. Ponadto, wszystkie żądania do API są zabezpieczone za pomocą klucza API, co ogranicza nieautoryzowany dostęp. Bezpieczeństwo transmisji danych jest dodatkowo wzmacniane przez stosowanie protokołu HTTPS, który zapewnia

szyfrowanie komunikacji między aplikacją Flask a API Hyperledger Fabric. Dzięki tym mechanizmom, zarządzanie danymi w systemie opartym na Hyperledger Fabric jest nie tylko wydajne, ale również bezpieczne.

3.5 Graficzny Interfejs Użytkownika

Graficzny interfejs użytkownika (GUI) systemu umożliwia interakcję pomiędzy systemem a jego użytkownikami. Jest on zaprojektowany tak, aby zapewnić intuicyjność obsługi oraz efektywne zarządzanie i przeglądanie danych. Aplikacja umożliwia dostęp do komponentów tylko dla zalogowanych użytkowników.

Logowanie

Ekran logowania jest pierwszym kontaktem użytkownika z systemem. Umożliwia wprowadzenie nazwy użytkownika oraz hasła i oferuje opcję "Zapamiętaj mnie". Wszystkie pozostałe funkcjonalności dostępne są tylko po zalogowaniu.

Logowanie

Nazwa użytkownika

Hasło

Remember me

Zaloguj

Lista Danych

Widok wszystkich danych jest ważnym elementem interfejsu, który umożliwia użytkownikom przeglądanie zgromadzonych danych medycznych. Użytkownicy mogą korzystać z filtrów, takich jak filtrowanie po imieniu, nazwisku, szpitalu, co ułatwia szybkie odnalezienie potrzebnych informacji. Każdy rekord danych można edytować, usunąć, lub przejrzeć jego historię zmian, co pozwala na dynamiczne zarządzanie informacjami. Dodatkowo użytkownik może wyświetlić modal z tabelą przedstawiającą znormalizowane dane każdego rekordu. Widok umożliwia nie tylko przegląd, ale również eksport do pliku CSV. Każdy z użytkowników, z wyjątkiem administratora widzi tylko dane szpitala, do którego jest przypisany.

Lista danych

[Eksportuj do CSV](#)
 Pokaż usunięte

#	Imię	Nazwisko	Szpital	Usunięto	Wersja	Data dodania	Opis	Dane	Akcje
202405061918031185	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Nie	2	2024-05-07 1:18	Poprawka danych z badania xyz	[{"pacjent": "daw", "pacjent_id": "2323", "imie": "dawid", "białko": "933"}]	Usuń Dane Edytuj Historia
202405061917451160	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Nie	1	2024-05-07 1:17	Dane dodane w wyniku badania xyz	[{"participant_id": "10822", "participant_id.1": "11450", "obecno": "015b1u0107_błaszek": "0", "gender": "1", "age": "23", "height": "165.7", "weight": "63.9", "rs3825807": "", "rs1801274": "\n", "rs4977574": "\n", "rs17398575": "\n", "rs17045031": "G/G", "rs6511720": "\n", "rs9369640": "A/G", "rs9349379": "\n", "rs8000449": "\n", "rs10455872": "T/C", "rs6952610": "\n"}]	Usuń Dane Edytuj
202405061917451290	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Nie	1	2024-05-07 1:17	Dane dodane w wyniku badania xyz	[{"participant_id": "10233", "participant_id.1": "10307", "obecno": "015b1u0107_błaszek": "0", "gender": "2", "age": "77", "height": "175.5", "weight": "81.2", "rs3825807": "", "rs1801274": "\n", "rs4977574": "G/G", "rs17398575": "\n", "rs17045031": "G/G", "rs6511720": "\n", "rs9369640": "G/G", "rs9349379": "T/T", "rs8000449": "\n", "rs10455872": "T/C", "rs6952610": "G/G"}]	Usuń Dane Edytuj
202405061917451338	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Nie	1	2024-05-07 1:17	Dane dodane w wyniku badania xyz	[{"participant_id": "10309", "participant_id.1": "10480", "obecno": "015b1u0107_błaszek": "1", "gender": "1", "age": "53", "height": "163.9", "weight": "91.1", "rs3825807": "", "rs1801274": "A/A", "rs4977574": "\n", "rs17398575": "\n", "rs17045031": "\n", "rs6511720": "C/A", "rs9369640": "\n", "rs9349379": "C/T", "rs8000449": "\n", "rs10455872": "C/T", "rs6952610": "C/C"}]	Usuń Dane Edytuj

Szczegóły Danych

Po wybraniu konkretnego rekordu z listy, użytkownik może wyświetlić modal, gdzie prezentowana jest tabela znormalizowanych danych konkretnego rekordu. Nie jest to możliwe w widoku głównym danych, ponieważ mogą one różnić się ilością kolumn.

Szczegóły danych																
participant_id	participant_id.1	obecność_błaszek	gender	age	height	weight	rs3825807	rs1801274	rs4977574	rs17398575	rs17045031	rs6511720	rs9369640	rs9349379	rs8000449	rs10455872
10421	10948	0	2	71	159.2	92.8				G/G	G/G	A/C	C/T	A/G	C/C	
										badania xyz	"A/G", "rs4977574": "G/G", "rs17398575": "G/G", "rs17045031": "G/G", "rs6511720": "A/C", "rs9369640": "\n", "rs9349379": "T/T", "rs8000449": "G/G", "rs10455872": "C/T", "rs6952610": "\n"}]		Dane Edytuj			
202405061917451609	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Nie	1	2024-05-07 1:17	Dane dodane w wyniku badania xyz	[{"participant_id": "11631", "participant_id.1": "10068", "obecno": "015b1u0107_błaszek": "0", "gender": "2", "age": "31", "height": "172.6", "weight": "91.3", "rs3825807": "", "rs1801274": "G/A", "rs4977574": "G/G", "rs17398575": "\n", "rs17045031": "G/G", "rs6511720": "A/A", "rs9369640": "A/A", "rs9349379": "C/T", "rs8000449": "G/A", "rs10455872": "T/C", "rs6952610": "A/A"}]		Usuń Dane Edytuj						
202405061917451821	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Nie	1	2024-05-07 1:17	Dane dodane w wyniku badania xyz	[{"participant_id": "10654", "participant_id.1": "10240", "obecno": "015b1u0107_błaszek": "0", "gender": "2", "age": "38", "height": "174.7", "weight": "90.1", "rs3825807": "", "rs1801274": "G/G", "rs4977574": "A/O", "rs17398575": "\n", "rs17045031": "G/G", "rs6511720": "C/A", "rs9369640": "G/A", "rs9349379": "\n", "rs8000449": "G/O", "rs10455872": "G/O", "rs6952610": "A/O"}]		Usuń Dane Edytuj						
202405061917451839	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Nie	1	2024-05-07 1:17	Dane dodane w wyniku badania xyz	[{"participant_id": "10421", "participant_id.1": "10948", "obecno": "015b1u0107_błaszek": "0", "gender": "2", "age": "71", "height": "159.2", "weight": "92.8", "rs3825807": "", "rs1801274": "\n", "rs4977574": "\n", "rs17398575": "G/G", "rs17045031": "G/G", "rs6511720": "A/C", "rs9369640": "\n", "rs9349379": "C/T", "rs8000449": "A/G", "rs10455872": "C/C", "rs6952610": "A/G"}]		Usuń Dane Edytuj						

Rysunek 3.3: Szczegóły danych

Historia Danych

Widok historii danych umożliwia użytkownikom śledzenie wszystkich zmian dokonanych na konkretnym rekordzie danych, jeżeli takowe zaszły. W modalu prezentowana jest chronologiczna lista wszystkich operacji. Każda pozycja historii zawiera datę zmiany,

opis wykonanych operacji oraz szczegółowe dane, co jest istotne dla zachowania zgodności z przepisami o ochronie danych.

#	Imię	Nazwisko	Szpital	Usunięto	Wersja	Data dodania	Opis	Dane
202405061917451017	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Nie	1	2024-05-07 1:17	Dane dodane w wyniku badania xyz	[{"participant_id": "10516", "participant_id.1": "10518", "obecnoju015bju0107_blašek": "0", "gender": "2", "age": "62", "height": "172.7", "weight": "131.6", "rs3825807": "", "rs1801274": "G/A", "rs4977574": "G/G", "rs17398575": "G/G", "rs17045031": "G/G", "rs6511720": "C/A", "rs9369640": "G/A", "rs9349379": "\n", "rs8000449": "\n", "rs10455872": "C/C", "rs6952610": "A/A"}]
202405061918031185	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Tak	2	2024-05-07 1:18	Poprawka danych z badania xyz	[{"pacjent": "daw", "pacjent_id": "2323", "imie": "dawid", "białko": "933"}]
202405061917451160	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Nie	1	2024-05-07 1:17	Dane dodane w wyniku badania xyz	[{"participant_id": "10822", "participant_id.1": "11450", "obecnoju015bju0107_blašek": "0", "gender": "1", "age": "23", "height": "165.7", "weight": "63.9", "rs3825807": "", "rs1801274": "G/G", "rs4977574": "\n", "rs17398575": "G/G", "rs17045031": "G/G", "rs6511720": "\n", "rs9369640": "A/G", "rs9349379": "\n", "rs8000449": "\n", "rs10455872": "T/J/C", "rs6952610": "\n"}]
202405061917451290	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Nie	1	2024-05-07 1:17	Dane dodane w wyniku badania xyz	[{"participant_id": "10309", "participant_id.1": "10233", "obecnoju015bju0107_blašek": "0", "gender": "2", "age": "77", "height": "175.5", "weight": "81.2", "rs3825807": "", "rs1801274": "G/G", "rs4977574": "G/G", "rs17398575": "\n", "rs17045031": "G/G", "rs6511720": "\n", "rs9369640": "G/G", "rs9349379": "T/T", "rs8000449": "\n", "rs10455872": "T/J/C", "rs6952610": "G/G"}]

Rysunek 3.4: Historia danych

Dodawanie Danych

Dodawanie danych pozwala na import dużych zestawów danych do systemu. Użytkownicy mogą załadować plik CSV, a system automatycznie zintegruje przesłane dane. Ten sposób ułatwia masowe wprowadzanie nowych zestawów danych do systemu.

Zarządzanie Danymi ▾ Użytkownicy ▾ Szpitale ▾ Role ▾ Profil Wyloguj

Dodaj plik CSV

Wybierz plik CSV

Wybierz plik Nie wybrano pliku

Opis

Dodaj plik CSV

Rysunek 3.5: Dodawanie danych

Zarządzanie Użytkownikami

Zarządzanie użytkownikami pozwala administratorom systemu na zarządzanie kontami użytkowników, w tym na przypisywanie ról, edycję danych oraz ich aktywację lub blokowanie. Możliwość zarządzania uprawnieniami zapewnia utrzymanie odpowiednich poziomów dostępu do wrażliwych danych i funkcji systemu.

#	Nazwa użytkownika	Email	Imię	Nazwisko	Szpital	Aktywny	Rola	Akcje
1	admin	dawidmetelski00@gmail.com	Dawid	Metelski	Uniwersytecki Szpital Kliniczny w Białymostku	Tak	Administrator	<button>Blokuj</button> <button>Edytuj</button> <button>Usuń</button>
2	pracownik	pracownik@pracownik.pl	pracownik	pracownik	Szpital Pro-Medica w Ełku	Tak	Pracownik	<button>Blokuj</button> <button>Edytuj</button> <button>Usuń</button>
3	fffff	fffff@fffff.pl	fffff123	fffff	Szpital Pro-Medica w Ełku	Nie	Pracownik	<button>Aktywuj</button> <button>Edytuj</button> <button>Usuń</button>

Rysunek 3.6: Lista użytkowników

Twoje konto jest zablokowane.

Logowanie

Nazwa użytkownika
pracownik

Hasło
.....

Remember me

Zaloguj

Rysunek 3.7: Blokada użytkownika

Edytuj

Nazwa użytkownika	<input type="text" value="pracownik"/>
Imię	<input type="text" value="pracownik"/>
Nazwisko	<input type="text" value="pracownik"/>
Email	<input type="text" value="pracownik@pracownik.pl"/>
Szpital	<input type="text" value="Szpital Pro-Medica w Ełku"/>
Rola	<input type="text" value="Pracownik"/>
<button>Zapisz</button>	

Rysunek 3.8: Edycja użytkownika

Zarządzanie Szpitalami

W ramach zarządzania szpitalami administratorzy mogą dodawać, edytować oraz usuwać szpitale z listy dostępnych instytucji.

Zarządzanie Danymi		Użytkownicy	Szpitalne	Role	Profil	Wyloguj
#	Nazwa	Akcje				
1	Szpital Wojewódzki w Łomży		Edytuj	Usuń		
2	Szpital Pro-Medica w Ełku		Edytuj	Usuń		
3	Uniwersytecki Szpital Kliniczny w Białymostku		Edytuj	Usuń		

Rysunek 3.9: Lista szpitali

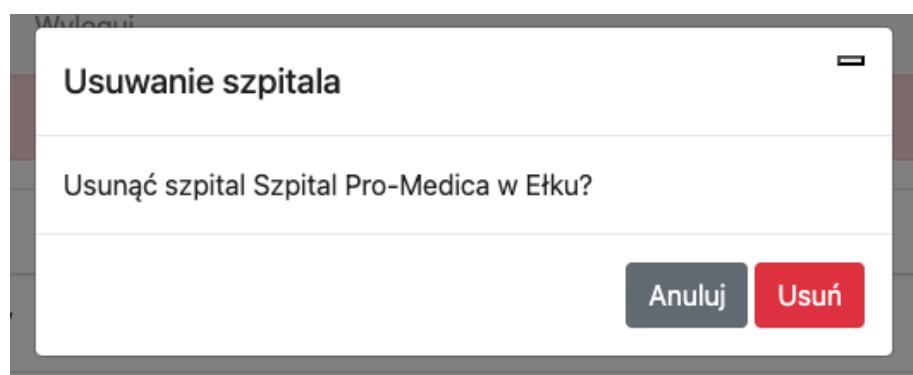
HOME Zarządzanie Danymi Użytkownicy Szpitale Role Profil Wyloguj

Nowy szpital

Nazwa szpitala

[Utwórz](#)

Rysunek 3.10: Dodawanie szpitali



Rysunek 3.11: Usunięcie szpitala

HOME Zarządzanie Darami ▾ Użytkownicy ▾ Szpitale ▾ Role ▾ Profil Wyloguj		
Szpital w użyciu, usunięcie niemożliwe.		
#	Nazwa	Akcje
1	Szpital Wojewódzki w Łomży	<button>Edytuj</button> <button>Usuń</button>
2	Szpital Pro-Medica w Ełku	<button>Edytuj</button> <button>Usuń</button>
3	Uniwersytecki Szpital Kliniczny w Białymostku	<button>Edytuj</button> <button>Usuń</button>

Rysunek 3.12: Błąd usunięcia używanego szpitala

Zarządzanie Rolami

Widok zarządzania rolami w systemie przeznaczony jest do tworzenia i modyfikacji ról użytkowników, co ma istotne znaczenie dla zarządzania uprawnieniami dostępu w ramach systemu.

Zarządzanie Darami ▾ Użytkownicy ▾ Szpitale ▾ Role ▾ Profil Wyloguj		
#	Nazwa	Actions
1	Administrator	<button>Edytuj</button> <button>Usuń</button>
2	Pracownik	<button>Edytuj</button> <button>Usuń</button>
3	Analityk	<button>Edytuj</button> <button>Usuń</button>

Rysunek 3.13: Lista szpitali

3.6 Testy Wydajnościowe

Testy wydajnościowe są ważnym elementem monitorowania wydajności tworzonych systemów informatycznych, szczególnie w kontekście aplikacji internetowych. W niniejszym projekcie testy mierzą czas odpowiedzi na zapytania do aplikacji oraz wskaźnik błędów przy obciążeniu systemu. Celem jest zapewnienie, że system archiwizacji danych eksperymentalnych działa niezawodnie nawet przy dużym natężeniu ruchu.

3.6.1 Scenariusze i Założenia Testów Wydajnościowych

Testy zostały przeprowadzone za pomocą narzędzia K6. Wykorzystane do tego zostały metryki, takie jak czas trwania zapytań (Trend) oraz wskaźnik błędów (Rate). Scenariusz testowy obejmuje różne etapy obciążenia systemu, począwszy od niskiego do wysokiego ruchu i obejmuje następujące etapy:

- Powolny wzrost do 20 użytkowników w ciągu 1 minuty symulujący wstępne obciążenie aplikacji.
- Szybszy wzrost do 50 użytkowników w ciągu 2 minut zwiększający obciążenie, aby sprawdzić, jak system radzi sobie ze średnim natężeniem ruchu.

- Szybszy wzrost do 100 użytkowników w ciągu 3 minut symulujący szczytowe obciążenie systemu.
- Utrzymanie 100 użytkowników przez 3 minuty sprawdzający stabilność aplikacji pod wysokim obciążeniem.
- Powolne zmniejszenie do 0 użytkowników w ciągu 3 minut symulujący zmniejszenie obciążenia i sprawdza, jak system wraca do normalnego stanu.

3.6.2 Pobieranie Listy Danych Medycznych

Test ma na celu zmierzenie wydajności endpointu odpowiedzialnego za pobieranie listy danych medycznych z API Hyperledger Fabric. Funkcjonalność ta umożliwia użytkownikom dostęp do zgromadzonych danych.

Kod Źródłowy Fragment kodu odpowiedzialny za test:

```
export default function () {
  // Testowanie endpointu /assets (API)
  const assetsParams = {
    headers: {
      'X-Api-Key': API_KEY,
    },
  };

  const assetsRes = http.get(`${API_URL}/assets`, assetsParams);
  check(assetsRes, {
    'assets success': (r) => r.status === 200 && r.json().length > 0,
  });
  assetsTrend.add(assetsRes.timings.duration);
  assetsErrorRate.add(assetsRes.status !== 200);
}
```

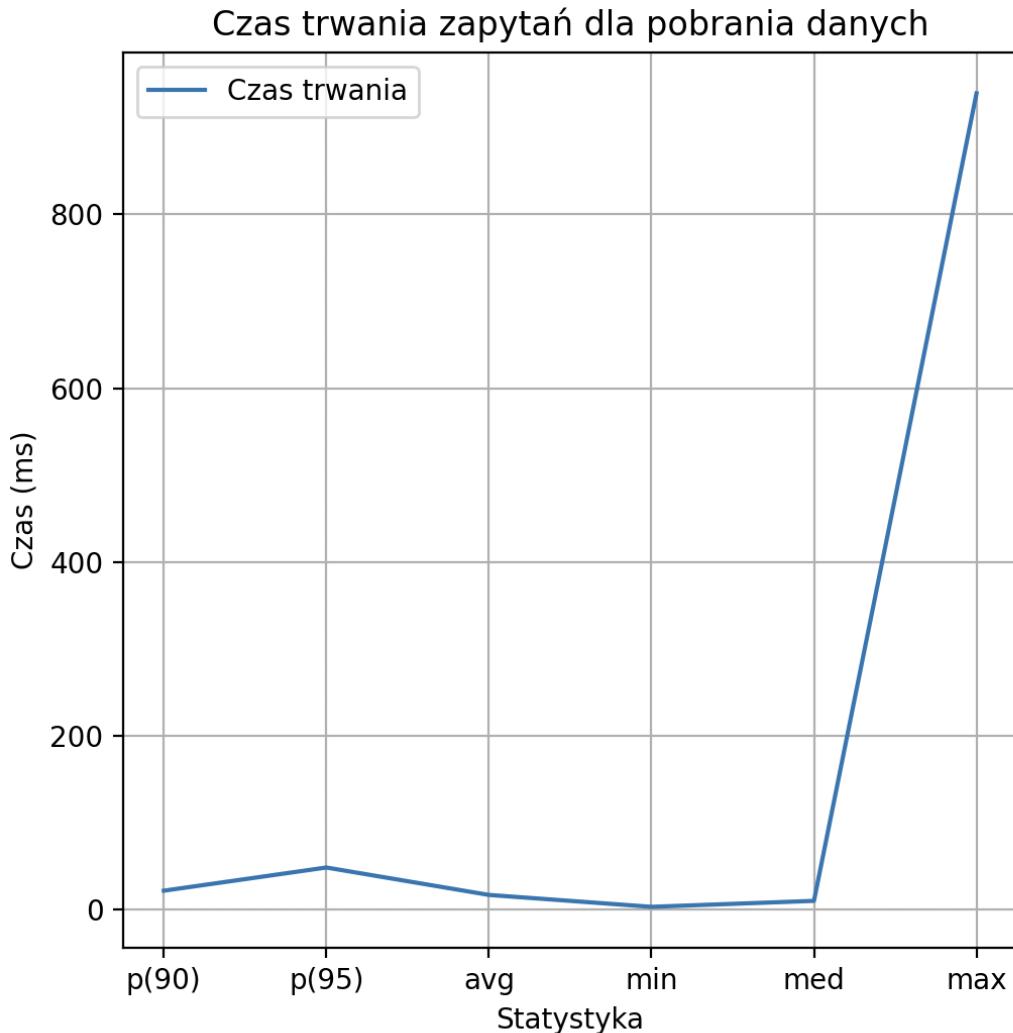
Powyższy kod wysyła zapytanie GET do odpowiedniego endpointu sprawdzając, czy odpowiedź jest poprawna i czy zwrocone dane nie są puste. Metryki czasu trwania i błędów są aktualizowane na podstawie wyników tego testu.

Wyniki

Test uzyskał następujące wyniki:

- Średni czas (avg): 40.76 ms
- Mediana (med): 29.34 ms
- Minimalny czas (min): 2.11 ms

- Maksymalny czas (max): 1543.09 ms
- 90-ty percentyl (p(90)): 79.56 ms
- 95-ty percentyl (p(95)): 100.77 ms



Rysunek 3.14: Test Wydajnościowy Listy Danych

Średni czas pobierania listy danych medycznych 40.76 ms oraz mediana 29.34 ms wskazują na dobrą wydajność systemu w scenariuszu testowym. Minimalny czas 2.11 ms sugeruje, że w warunkach testowych operacja ta jest wykonywana bardzo szybko, co jest korzystne dla systemu. Maksymalny czas wyniósł 1543.09 ms, jednak pozostałe miary wskazują, że jest to sporadyczne. Taka wartość może być wynikiem nieoczekiwanych zakłóceń. Wartości percentyli sugerują, że operacje mieszczą się w akceptowalnych granicach czasowych.

3.6.3 Przesłanie Nowych Danych Medycznych

Test ma na celu zmierzenie wydajności endpointu odpowiedzialnego za tworzenie danych medycznych w API Hyperledger Fabric. Funkcjonalność ta umożliwia użytkownikom

dodawanie danych do systemu.

Kod Źródłowy

Kod testujący przesyłanie nowych danych medycznych:

```
export default function () {
    // Testowanie endpointu /new_data (API)
    const newDataPayload = {
        csv_file: 'permutacja_example.csv',
        description: 'Test data',
    };

    const newDataParams = {
        headers: {
            'Content-Type': 'application/json',
            'X-Api-Key': API_KEY,
        },
    };

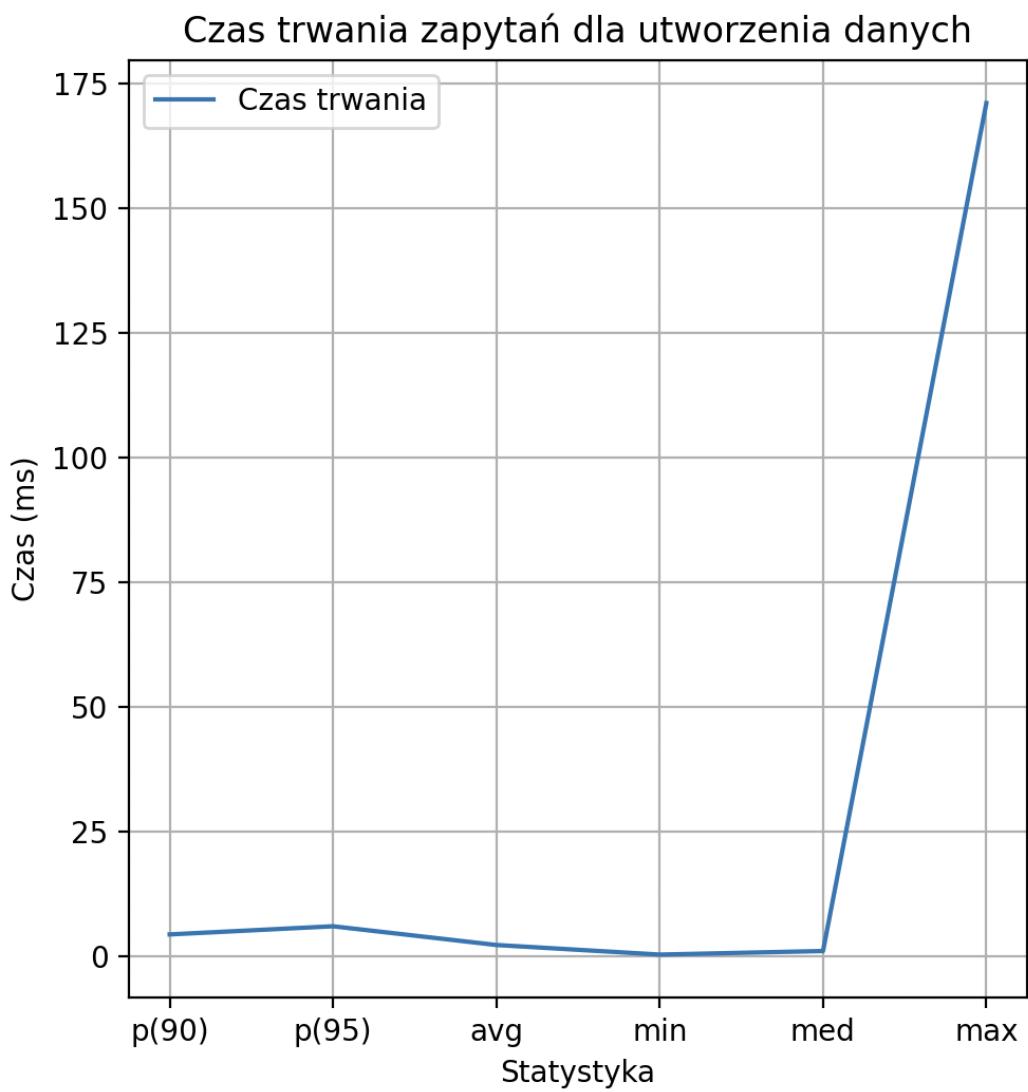
    const newDataRes = http.post(`${API_URL}/assets`, newDataPayload, newDataParams);
    check(newDataRes, {
        'new_data success': (r) => r.status === 202,
    });
    newDataTrend.add(newDataRes.timings.duration);
    newDataErrorRate.add(newDataRes.status !== 202);
}
```

Kod wysyła zapytanie POST z danymi pliku CSV do endpointu odpowiadającego za tworzenie danych. Kod sprawdza czy odpowiedź ma status 202, co oznacza, że dane zostały poprawnie przesłane i przetworzone.

Wyniki

Test uzyskał następujące wyniki:

- Średni czas (avg): 2.16 ms
- Mediana (med): 0.96 ms
- Minimalny czas (min): 0.24 ms
- Maksymalny czas (max): 171.08 ms
- 90-ty percentyl (p(90)): 4.29 ms
- 95-ty percentyl (p(95)): 5.92 ms



Rysunek 3.15: Test Wydajnościowy Tworzenia Danych

Operacja przesyłania danych medycznych osiągnęła bardzo niskie średnie czasy (2.16 ms) i mediany (0.96 ms). Wskazują one, że system jest bardzo efektywny w realizacji tej operacji. Minimalny czas 0.24 ms sugeruje, że w optymalnych warunkach operacja ta jest wykonywana niezwykle szybko. Maksymalny czas 171.08 ms, choć znacznie wyższy niż średnia, nadal jest stosunkowo niski, co wskazuje na sporadyczne problemy z wydajnością. Niskie wartości percentylów potwierdzają, że operacja ta jest wykonywana szybko i sprawnie w większości przypadków.

3.6.4 Pobieranie Historii Danych

Celem tego testu jest ocena wydajności endpointu służącego do pobierania historii wprowadzania danych. Test zapewnia, że użytkownicy mogą szybko i efektywnie uzyskać dostęp do historii transakcji.

Kod Źródłowy

Kod odpowiedzialny za testowanie pobierania historii danych:

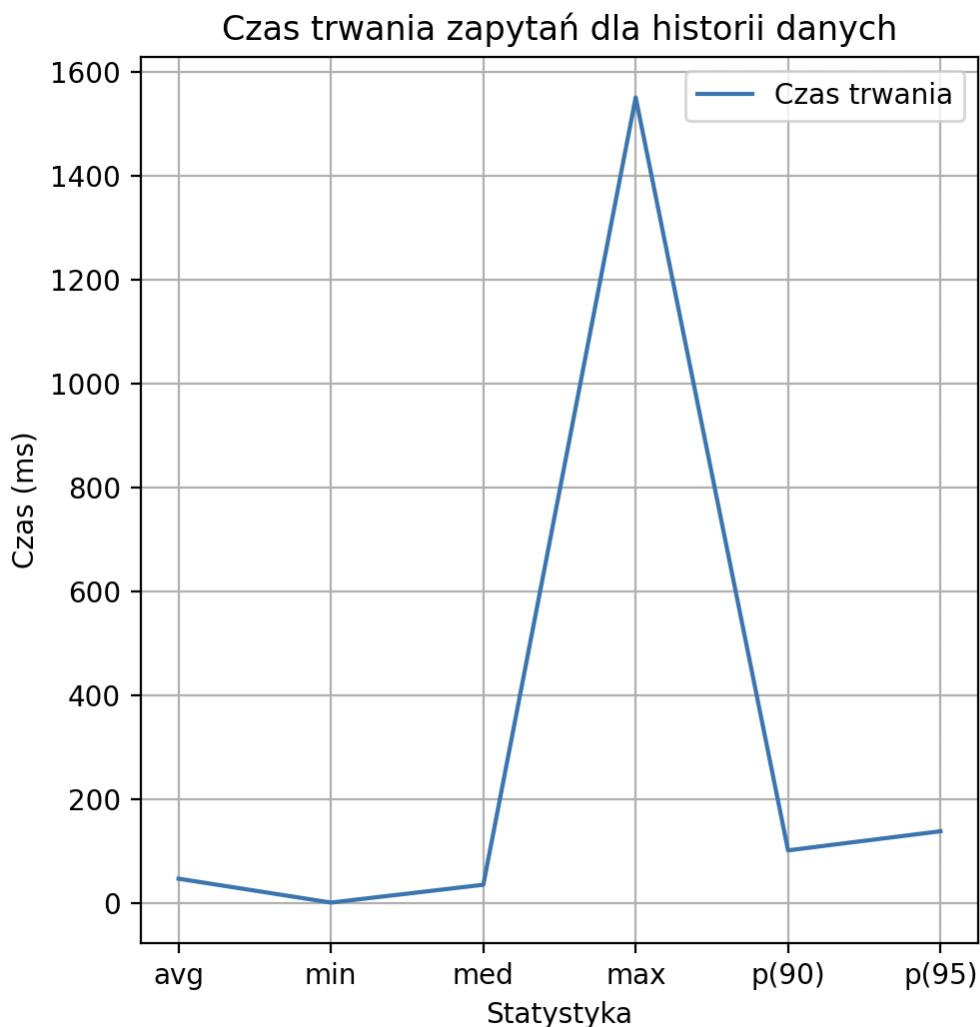
```
export default function () {
  // Testowanie endpointu /data_history
  const dataHistoryRes = http.get(`${APP_URL}/data_history`);
  check(dataHistoryRes, {
    'data history success': (r) => r.status === 200,
  });
  dataHistoryTrend.add(dataHistoryRes.timings.duration);
  dataHistoryErrorRate.add(dataHistoryRes.status !== 200);
}
```

Kod wysyła zapytanie GET do endpointu /data_history i sprawdza, czy odpowiedź ma status 200. Metryki czasu trwania i błędów są aktualizowane na podstawie wyników testu.

Wyniki

Test uzyskał następujące wyniki:

- Średni czas (avg): 47.75 ms
- Mediana (med): 36.09 ms
- Minimalny czas (min): 1.84 ms
- Maksymalny czas (max): 1551.73 ms
- 90-ty percentyl (p(90)): 102.32 ms
- 95-ty percentyl (p(95)): 139.09 ms



Rysunek 3.16: Test Wydajnościowy Historii Danych

Pobieranie historii danych osiąga wyższe czasy trwania. Średni czas wynoszący 47.75 ms i mediana 36.09 ms są akceptowalne, lecz wyższe, niż dla poprzednich operacji. Minimalny czas 1.84 ms sugeruje, że operacja ta może być wykonana bardzo szybko. Wysoki czas maksymalny 1551.73 ms sugeruje natomiast, że są sytuacje, w których operacja ta jest wydłużona i wpływa na użytkowników.

3.6.5 Pobieranie Listy Ról Użytkowników

Test ma na celu ocenę wydajności endpointu odpowiedzialnego za pobieranie listy ról użytkowników. Zapewnia, że system może szybko zwrócić informacje o rolach.

Kod Źródłowy Kod testujący pobieranie listy ról użytkowników:

```
export default function () {
  // Testowanie endpointu /roles
  const rolesRes = http.get(`${APP_URL}/roles`);
  check(rolesRes, {
```

```
    'roles list success': (r) => r.status === 200,
  });
  rolesListTrend.add(rolesRes.timings.duration);
  rolesListErrorRate.add(rolesRes.status !== 200);
}

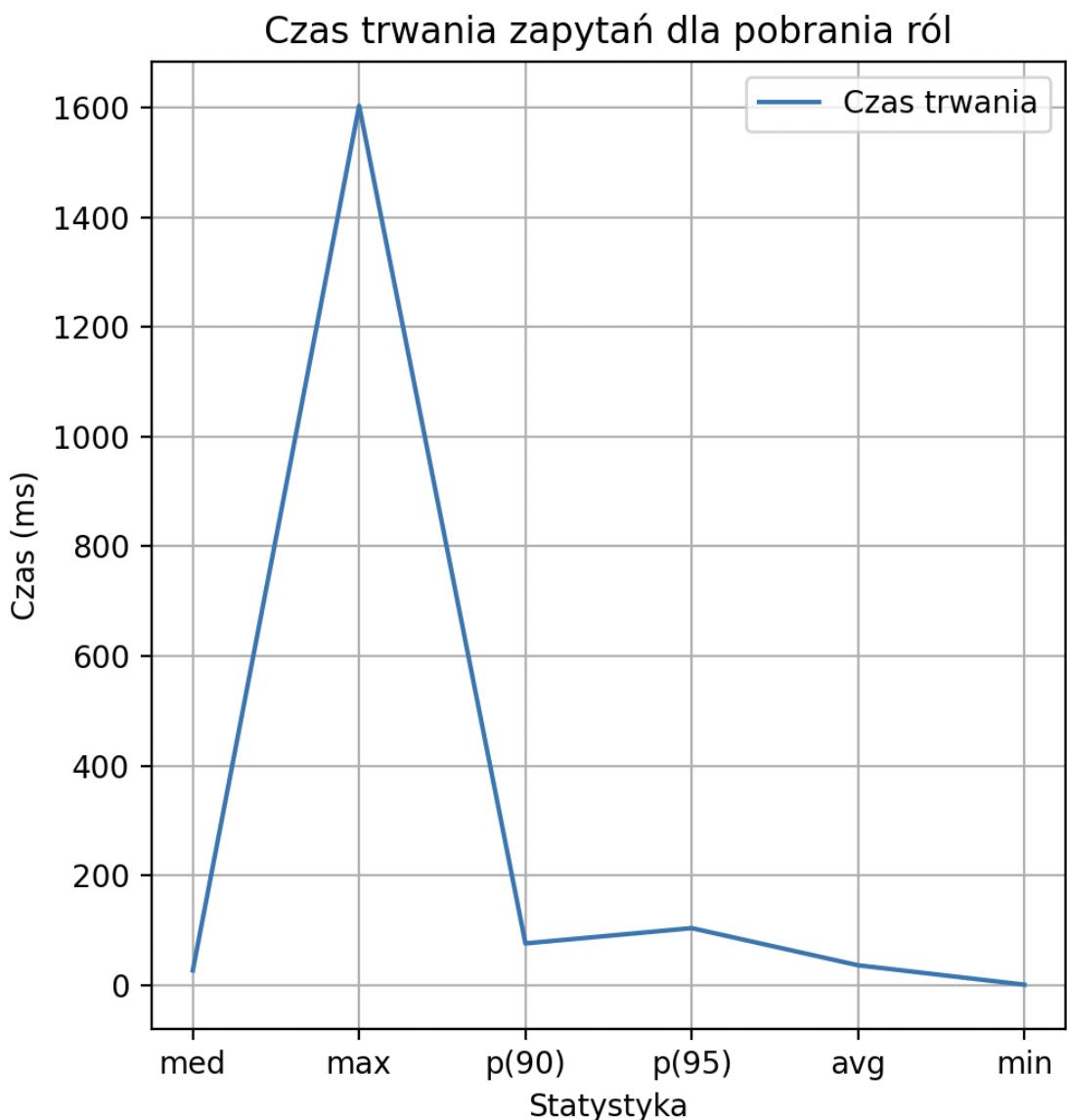
```

Kod wysyła zapytanie GET do endpointu /roles i sprawdza, czy odpowiedź jest poprawna. Metryki czasu trwania i błędów są aktualizowane na podstawie wyników testu.

Wyniki

Test uzyskał następujące wyniki:

- Średni czas (avg): 36.47 ms
- Mediana (med): 27.13 ms
- Minimalny czas (min): 1.06 ms
- Maksymalny czas (max): 1602.97 ms
- 90-ty percentyl (p(90)): 76.30 ms
- 95-ty percentyl (p(95)): 104.17 ms



Rysunek 3.17: Test Wydajnościowy Listy Ról

Operacja pobierania listy ról użytkowników wykazuje dobry poziom wydajności. Średni czas wynoszący 36.47 ms oraz mediana na poziomie 27.13 ms sugerują, że operacja ta jest stabilna. Minimalny czas 1.06 ms wskazuje na to, że operacja może być wykonywana bardzo szybko. Maksymalny czas 1602.97 ms wskazuje na sporadycznie dłuższe czasy wykonania. Wartości percentylu prezentują stabilny czas wykonania operacji, jednak może to ulec zmianie w najgorszych scenariuszach. Oznacza to, że istnieją przypadki, w których operacja ta działa znacznie wolniej.

3.6.6 Tworzenie Nowych Ról Użytkowników

Test ma na celu zmierzenie wydajności tworzenia ról. Funkcjonalność ta umożliwia dodawanie roli dla użytkowników.

Kod Źródłowy

Kod testujący tworzenie nowych ról użytkowników:

```

export default function () {
  // Testowanie endpointu /new_role
  const newRolePayload = {
    name: 'Test Role',
  };

  const newRoleRes = http.post(`${APP_URL}/new_role`, newRolePayload);
  check(newRoleRes, {
    'new role success': (r) => r.status === 200,
  });
  newRoleTrend.add(newRoleRes.timings.duration);
  newRoleErrorRate.add(newRoleRes.status !== 200);
}

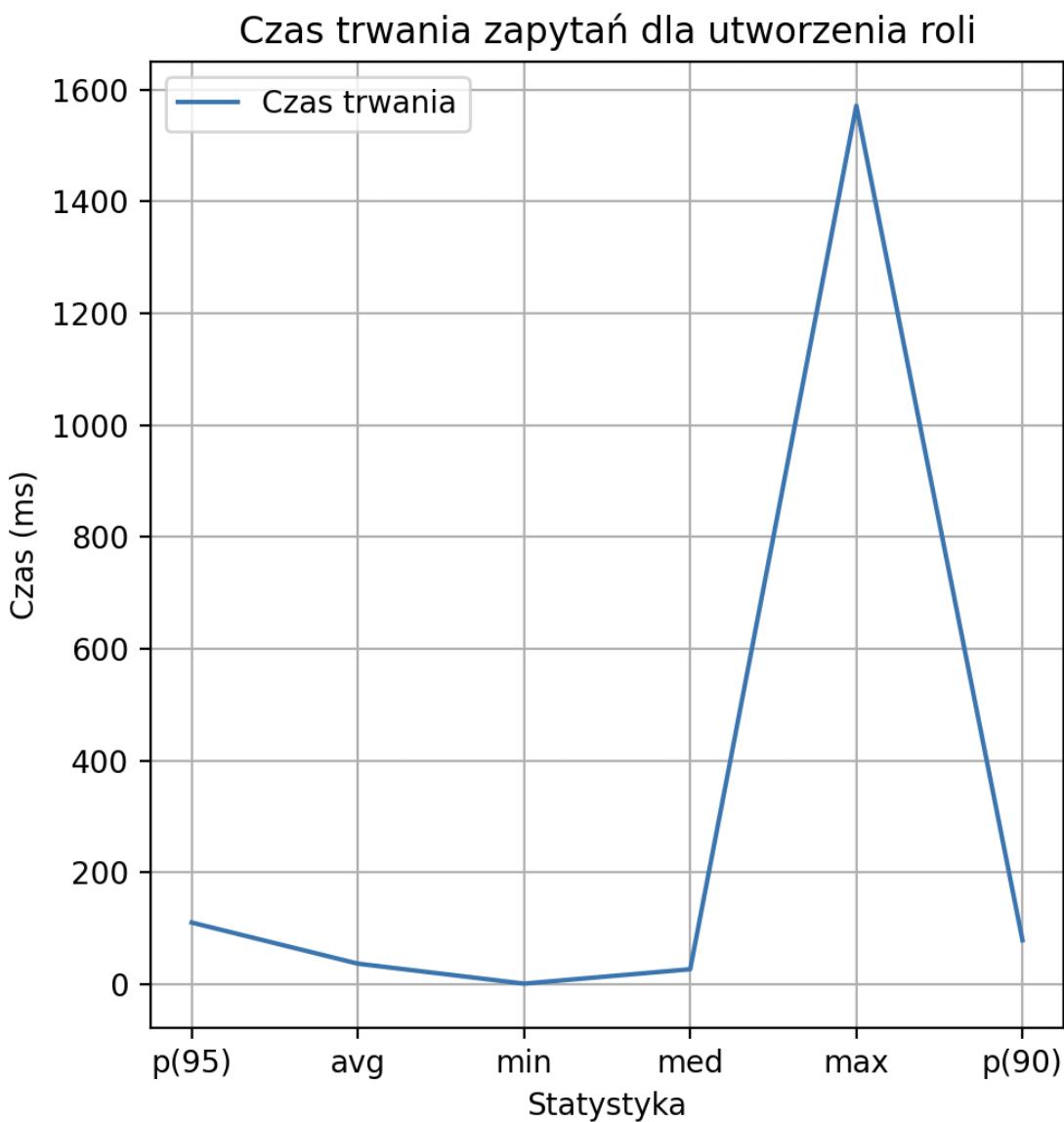
```

Kod wysyła zapytanie POST z danymi do endpointu /new_role. Sprawdzane jest, czy odpowiedź ma status 200, co oznacza, że rola została poprawnie dodana.

Wyniki

Test uzyskał następujące wyniki:

- Średni czas (avg): 36.63 ms
- Mediana (med): 26.66 ms
- Minimalny czas (min): 0.89 ms
- Maksymalny czas (max): 1571.41 ms
- 90-ty percentyl (p(90)): 78.63 ms
- 95-ty percentyl (p(95)): 110.17 ms



Rysunek 3.18: Test Wydajnościowy Tworzenia Ról

Średni czas tworzenia nowych ról użytkowników wynosił 36.63 ms, a mediana 26.66 ms. Wartości te wskazują na optymalne działanie operacji. Minimalny czas 0.89 ms świadczy o możliwości bardzo szybkiego wykonania tej operacji. Maksymalny czas wyniósł w skrajnym przypadku 1571.41 ms. Wysokie wartości percentylów wskazują, że w niektórych przypadkach tworzenie nowych ról użytkowników może być czasochłonne względem pozostałych operacji.

3.6.7 Pobieranie Listy Szpitali

Celem tego testu jest ocena wydajności endpointu służącego do pobierania listy szpitali. Jest to ważne dla zapewnienia, że system może szybko zwrócić informacje o szpitalach.

Kod Źródłowy

Kod odpowiedzialny za testowanie pobierania listy szpitali:

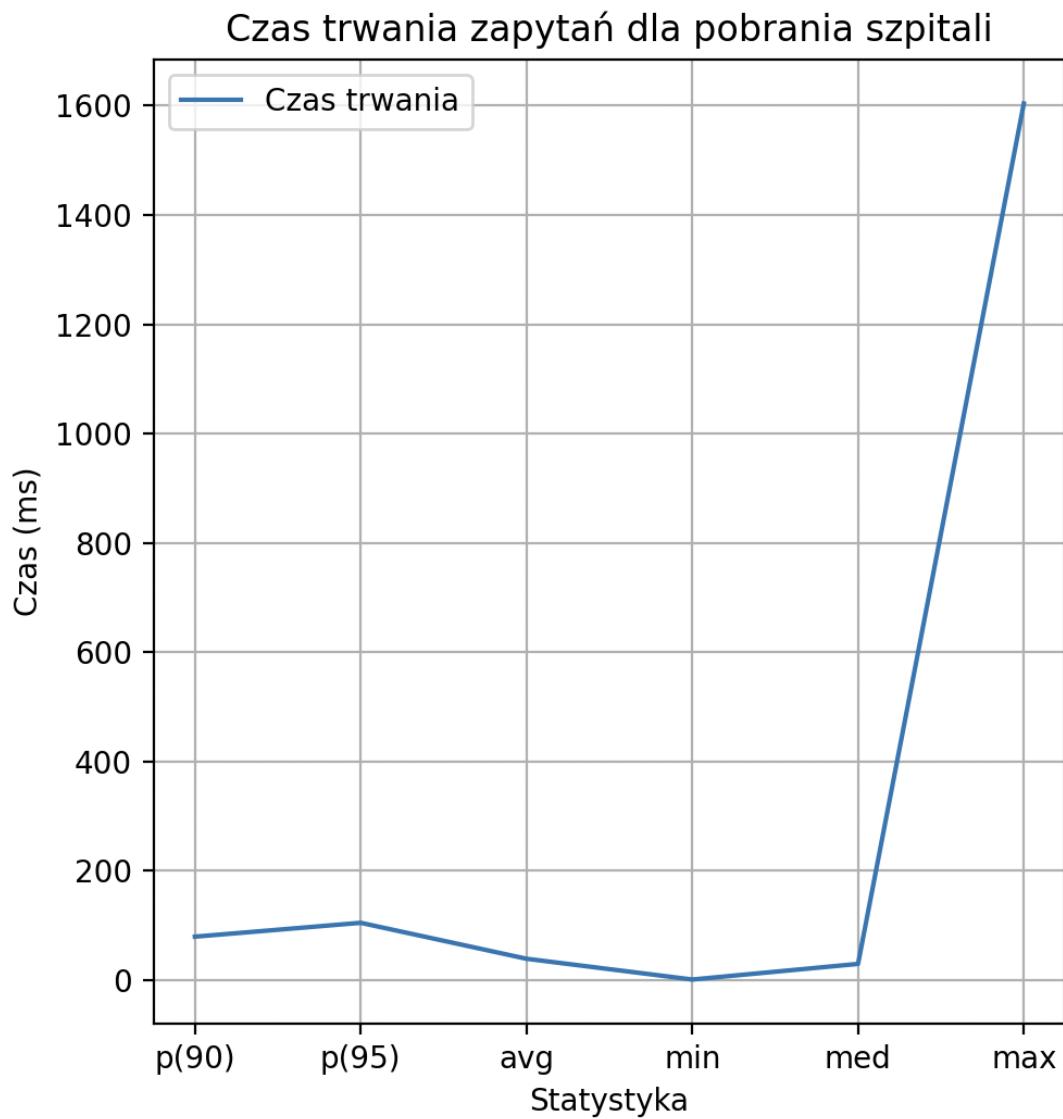
```
export default function () {
  // Testowanie endpointu /hospitals
  const hospitalsRes = http.get(`${APP_URL}/hospitals`);
  check(hospitalsRes, {
    'hospitals list success': (r) => r.status === 200,
  });
  hospitalsListTrend.add(hospitalsRes.timings.duration);
  hospitalsListErrorRate.add(hospitalsRes.status !== 200);
}
```

Kod ten wysyła zapytanie GET do endpointu /hospitals i sprawdza, czy odpowiedź ma status 200. Metryki czasu trwania i błędów są aktualizowane na podstawie wyników testu.

Wyniki

Test uzyskał następujące wyniki:

- Średni czas (avg): 39.12 ms
- Mediana (med): 29.72 ms
- Minimalny czas (min): 1.09 ms
- Maksymalny czas (max): 1603.85 ms
- 90-ty percentyl (p(90)): 79.55 ms
- 95-ty percentyl (p(95)): 104.88 ms



Rysunek 3.19: Test Wydajnościowy Listy Szpitali

Operacja pobierania listy szpitali jest podobna do innych operacji pobierania danych z bazy SQL. Średni czas 39.12 ms oraz mediana 29.72 ms wskazują na stabilne działanie systemu. Minimalny czas 1.09 ms jest bardzo niski. Maksymalny czas 1603.85 ms oraz wysokie wartości percentylów oznaczają, że mogą występować sporadyczne opóźnienia w działaniu systemu.

3.6.8 Tworzenie Nowych Szpitali

Test ocenia wydajność endpointu odpowiedzialnego za tworzenie nowych szpitali. Jest on istotny dla zapewnienia, że system może sprawnie dodawać nowe szpitale.

Kod Źródłowy

Kod testujący tworzenie nowych szpitali:

```
export default function () {
```

```
// Testowanie endpointu /new_hospital
const newHospitalPayload = {
    name: 'Test Hospital',
};

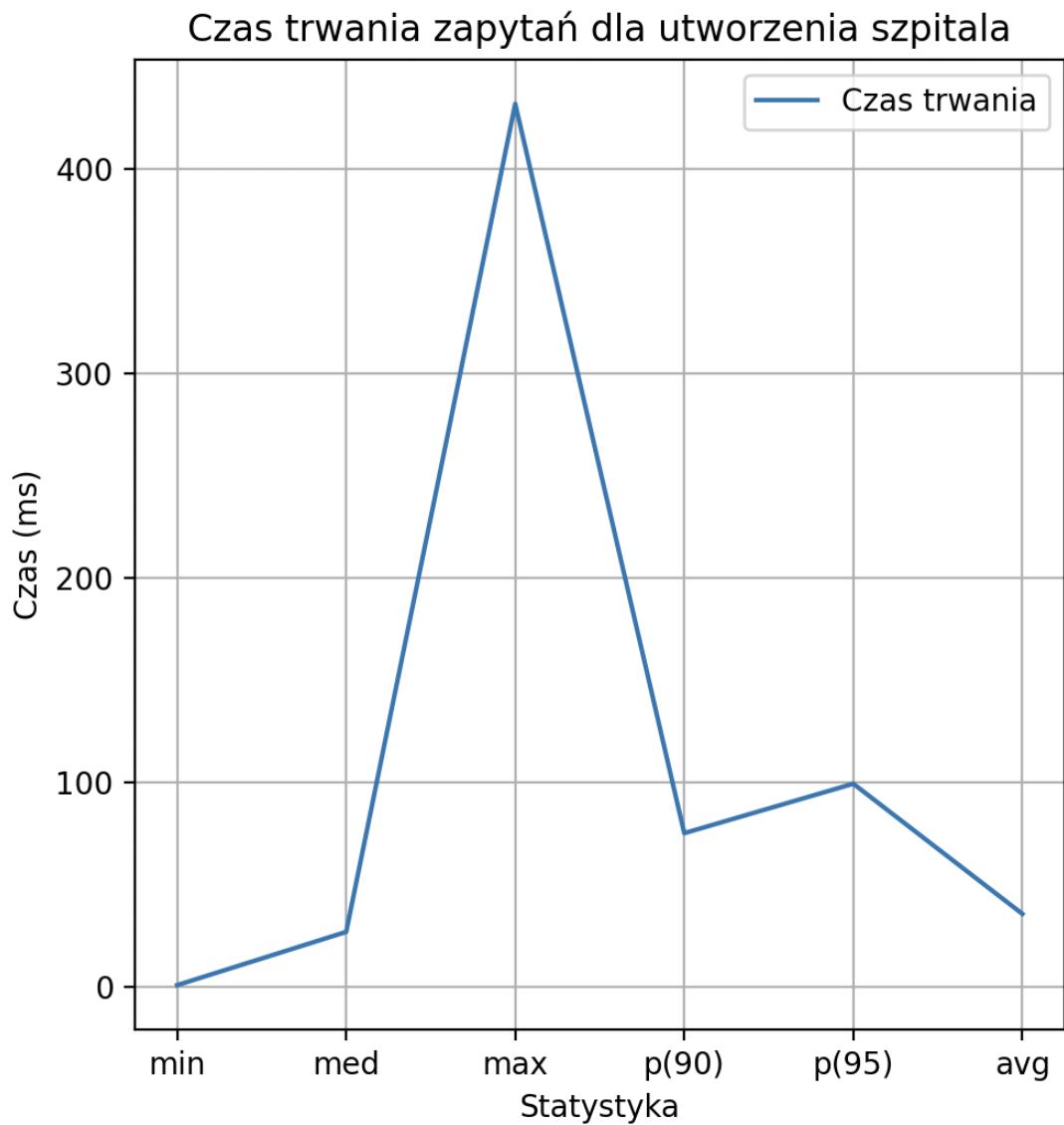
const newHospitalRes = http.post(`${APP_URL}/new_hospital`, newHospitalPayload);
check(newHospitalRes, {
    'new hospital success': (r) => r.status === 200,
});
newHospitalTrend.add(newHospitalRes.timings.duration);
newHospitalErrorRate.add(newHospitalRes.status !== 200);
}
```

Kod wysyła zapytanie POST z danymi nowego szpitala do endpointu /new_hospital. Sprawdzane jest, czy odpowiedź ma status 200, co oznacza, że szpital został poprawnie dodany.

Wyniki

Test uzyskał następujące wyniki:

- Średni czas (avg): 35.74 ms
- Mediana (med): 26.93 ms
- Minimalny czas (min): 0.91 ms
- Maksymalny czas (max): 431.57 ms
- 90-ty percentyl (p(90)): 75.18 ms
- 95-ty percentyl (p(95)): 99.34 ms



Rysunek 3.20: Test Wydajnościowy Tworzenia Szpitali

Tworzenie nowych szpitali cechuje się stabilnością, niskimi wartościami średnimi i medianą czasu wykonania operacji. Minimalny czas wynoszący 0,91 ms i maksymalny 431,57 ms wskazują na ogólną efektywność procesu, mimo że zdarzają się sporadyczne przypadki dłuższych czasów realizacji. Niskie wartości percentylu dodatkowo sugerują, że większość operacji przebiega szybko i sprawnie, co podkreśla efektywność przeprowadzanych działań.

3.6.9 Pobieranie Listy Użytkowników

Test ma na celu ocenę wydajności pobierania listy użytkowników. Jest to istotne dla zapewnienia, że system szybko zwraca informacje o użytkownikach.

Kod Źródłowy

Kod testujący pobieranie listy użytkowników:

```
export default function () {
```

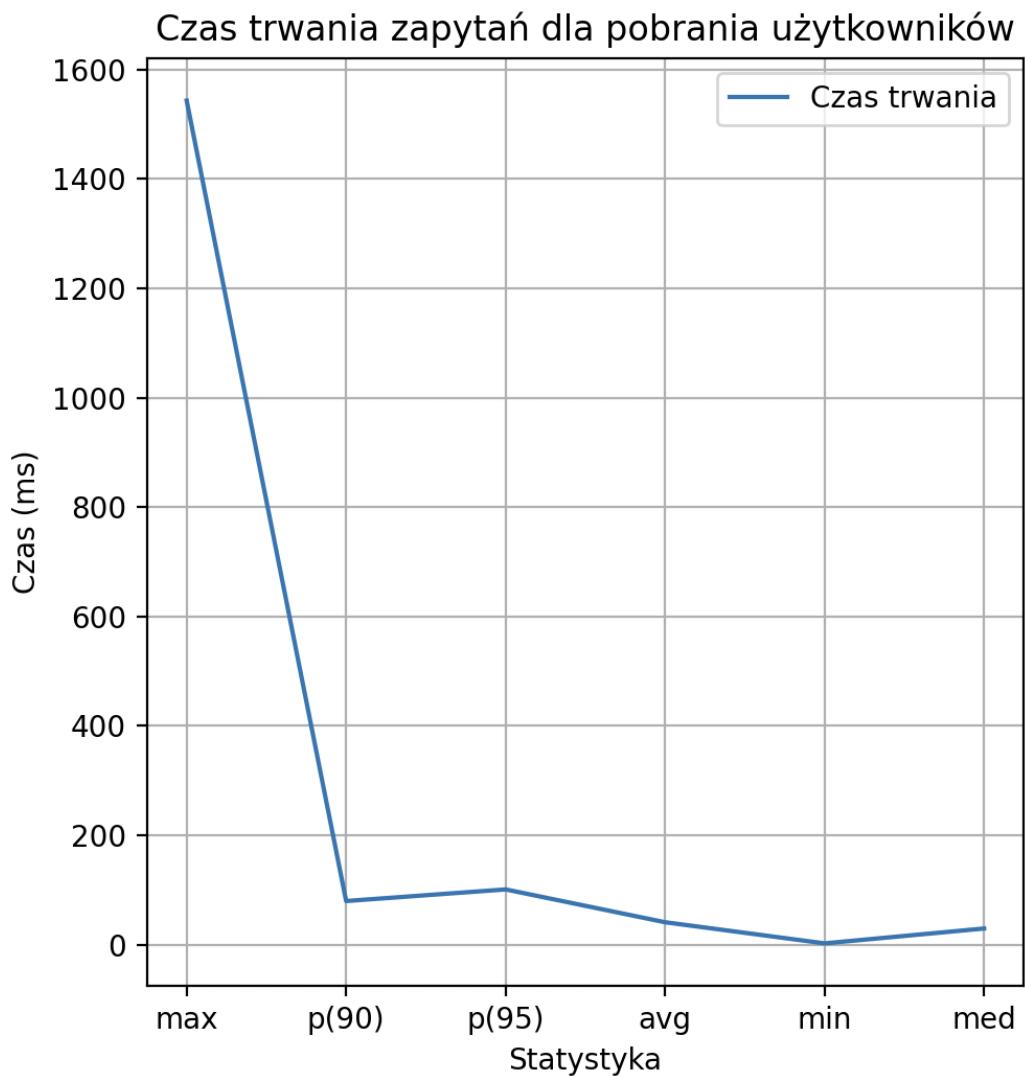
```
// Testowanie endpointu /users
const usersRes = http.get(`${APP_URL}/users`);
check(usersRes, {
  'users list success': (r) => r.status === 200,
});
usersListTrend.add(usersRes.timings.duration);
usersListErrorRate.add(usersRes.status !== 200);
}
```

Kod wysyła zapytanie GET do endpointu /users i sprawdza, czy odpowiedź jest poprawna. Metryki czasu trwania i błędów są aktualizowane na podstawie wyników testu.

Wyniki

Test uzyskał następujące wyniki:

- Średni czas (avg): 40.76 ms
- Mediana (med): 29.34 ms
- Minimalny czas (min): 2.11 ms
- Maksymalny czas (max): 1543.09 ms
- 90-ty percentyl (p(90)): 79.56 ms
- 95-ty percentyl (p(95)): 100.77 ms



Rysunek 3.21: Test Wydajnościowy Listy Użytkowników

Operacja pobierania listy użytkowników osiągnęła średni czas wynoszący 40.76 ms oraz medianę na poziomie 29.34 ms. Wartości wskazują na ogólnie stabilne i szybkie działanie tej operacji. Minimalny czas 2.11 ms sugeruje, że operacja ta może być wykonywana bardzo szybko. Jednak maksymalny czas wynoszący 1543.09 ms jest znacznie wyższy i wskazuje na sporadyczne przypadki znacznego opóźnienia. Takie skrajne wartości mogą być wynikiem chwilowego przeciążenia systemu lub innych nieoczekiwanych zakłóceń. Wartości percentylu wynoszące 79.56 ms i 100.77 ms wskazują, że większość operacji mieści się w rozsądnych granicach czasowych. To oznacza, że w 90% przypadków czas trwania operacji nie przekracza 79.56 ms, a w 95% przypadków nie przekracza 100.77 ms. Jest to dobry wynik, sugerujący, że tylko w nielicznych przypadkach czas trwania operacji jest wydłużony.

3.6.10 Tworzenie Nowych Użytkowników

Test ocenia wydajność endpointu odpowiedzialnego za tworzenie nowych użytkowników. Zapewnia, że system może sprawnie dodawać nowych użytkowników.

Kod Źródłowy

Kod testujący tworzenie nowych użytkowników:

```
export default function () {
    // Testowanie endpointu /new_user
    const newUserPayload = {
        name: 'testuser',
        first_name: 'Test',
        last_name: 'User',
        email: 'testuser@example.com',
        password: 'password',
        hospital_id: 1,
        role_id: 2,
    };

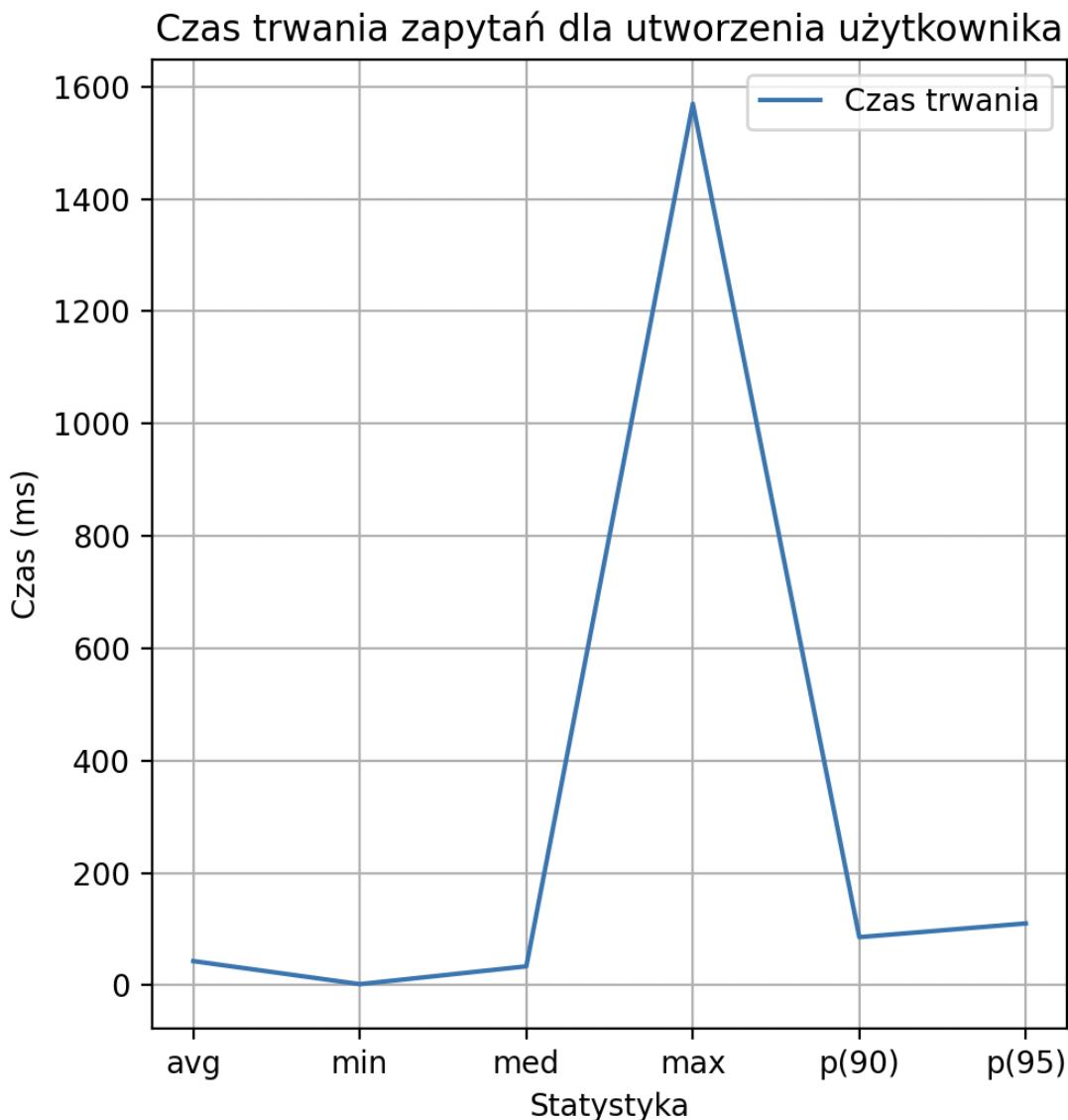
    const newUserRes = http.post(`${APP_URL}/new_user`, newUserPayload);
    check(newUserRes, {
        'new user success': (r) => r.status === 200,
    });
    newUserTrend.add(newUserRes.timings.duration);
    newUserErrorRate.add(newUserRes.status !== 200);
}
```

Kod wysyła zapytanie POST z danymi nowego użytkownika do endpointu /new_user. Sprawdza, czy odpowiedź ma status 200, co oznacza, że użytkownik został poprawnie dodany.

Wyniki

Test uzyskał następujące wyniki:

- Średni czas (avg): 43.03 ms
- Mediana (med): 33.69 ms
- Minimalny czas (min): 1.97 ms
- Maksymalny czas (max): 1569.40 ms
- 90-ty percentyl (p(90)): 85.68 ms
- 95-ty percentyl (p(95)): 110.06 ms



Rysunek 3.22: Test Wydajnościowy Tworzenia Użytkownika

Operacja tworzenia nowych użytkowników wykazuje stabilność z niskimi wartościami średnimi i medianą czasu wykonania. Wysokie wartości maksymalne i percentylu sugerują sporadyczne przypadki, które mogą prowadzić do wydłużonych czasów realizacji.

3.6.11 Skalowanie Rozmiaru Danych

Testy wydajności względem rozmiaru danych badają jego wpływ na szybkość wykonywania określonych działań. Celem testów było zbadanie, jak różne konfiguracje danych wpływają na wydajność operacji wykonywanych poprzez integrację z systemem blockchain. Testy zostały przeprowadzone z wykorzystaniem zróżnicowanych liczb danych i kolumn.

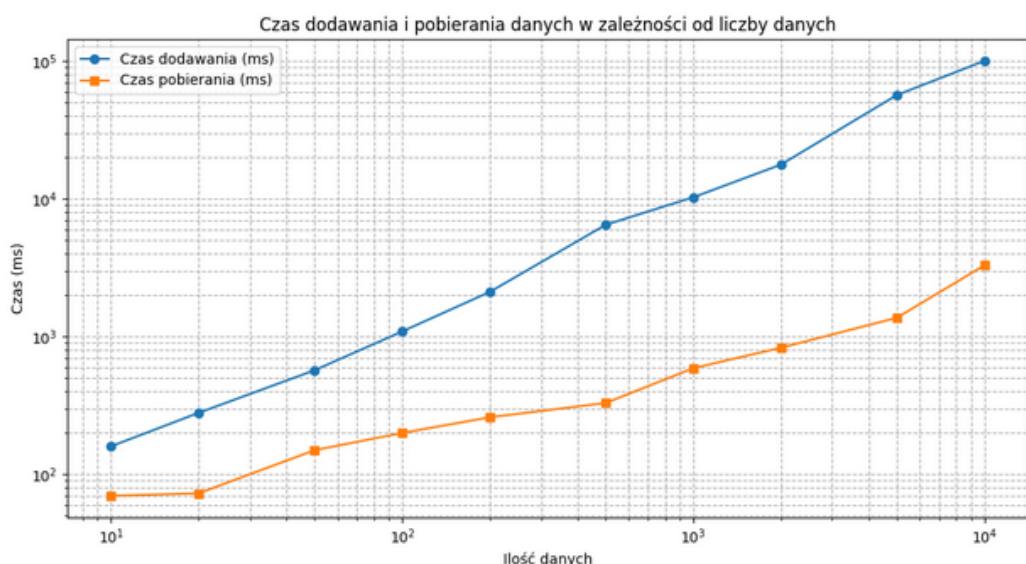
Skalowanie Względem Liczby Danych

Skalowanie względem liczby danych odnosi się do zdolności systemu do efektywnego

zarządzania rosnącą liczbą rekordów. System musi być w stanie dodawać, przechowywać oraz pobierać dane w sposób, który utrzymuje wysoką wydajność niezależnie od rosnącej liczby przechowywanych rekordów. Poniższa tabela przedstawia wyniki:

Liczba danych	Liczba kolumn	Czas dodawania	Czas pobierania
10	845	160 ms	70 ms
20	845	280 ms	73 ms
50	845	570 ms	150 ms
100	845	1,090 ms	200 ms
200	845	2,110 ms	260 ms
500	845	6,470 ms	330 ms
1,000	845	10,250 ms	590 ms
2,000	845	17,690 ms	830 ms
5,000	845	56,520 ms	1,370 ms
10,000	845	100,010 ms	3,290 ms

Tabela 3.1: Czasy tworzenia i pobierania zasobów w blockchain dla różnej liczby danych



Rysunek 3.23: Test Wydajnościowy dla Różnych Liczb Danych

Przeprowadzone testy wykazały, że system blockchain doskonale radzi sobie z rosnącą liczbą danych, zapewniając płynne i stabilne operacje zarówno w zakresie dodawania, jak i pobierania danych. Minimalny czas dodawania danych wyniósł 160 ms dla 10 rekordów, a maksymalny, dla 10 000 rekordów wyniósł 100,010 ms, co pokazuje proporcjonalny wzrost czasu dodawania wraz z liczbą danych, świadcząc o stabilności i przewidywalności systemu. Podobnie, minimalny czas pobierania danych wyniósł 70 ms dla 10 rekordów, podczas gdy maksymalny, dla 10 000 rekordów wyniósł 3,290 ms. Taki wzrost wskazuje na efektywne zarządzanie dużymi zbiorami danych, ponieważ nieznacznie wpływa na pobieranie danych, co wskazuje na brak problemu z ich wyświetleniem.

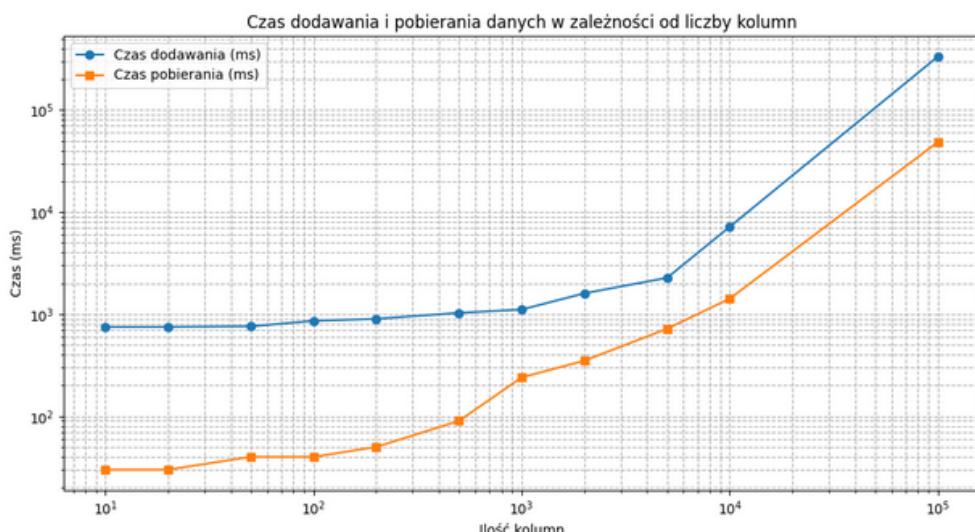
Skalowanie Względem Liczby Kolumn

Skalowanie względem liczby kolumn odnosi się do zdolności systemu do zarządzania

rekordami o różnej liczbie atrybutów. System musi być w stanie utrzymać wysoką wydajność podczas ich przechowywania i przetwarzania. Wyniki zostały przedstawione w poniższej tabeli:

Liczba kolumn	liczba danych	Czas dodawania	Czas pobierania
10	100	750 ms	30 ms
20	100	750 ms	30 ms
50	100	760 ms	40 ms
100	100	860 ms	40 ms
200	100	900 ms	50 ms
500	100	1,030 ms	90 ms
1,000	100	1,110 ms	240 ms
2,000	100	1,600 ms	350 ms
5,000	100	2,270 ms	720 ms
10,000	100	7,180 ms	1,420 ms
100,000	100	333,720 ms	48,510 ms

Tabela 3.2: Czasy tworzenia i pobierania zasobów w blockchain dla różnej liczby kolumn



Rysunek 3.24: Test Wydajnościowy dla Różnej Liczby Kolumn

Testy wydajnościowe wykazały, że system blockchain jest wysoce efektywny w zarządzaniu rekordami o różnej liczbie kolumn. Minimalny czas dodawania danych wyniósł 750 ms dla rekordów zawierających 10 kolumn, natomiast czas dodawania dla rekordów z 10 000 kolumn wyniósł 7,180 ms. Pokazuje to, że system zachowuje wysoką wydajność nawet przy znacznym wzroście liczby kolumn. Warto również zwrócić uwagę na czas pobierania danych. Minimalny wyniósł 30 ms dla rekordów z 10 kolumnami, podczas gdy czas dla danych z 10 000 kolumn osiągnął 1,420 ms. Wzrost liczby kolumn wpływa nieznacznie na czas operacji, co świadczy o doskonałej skalowalności systemu.

Dodatkowo, w celu sprawdzenia możliwości systemu oraz identyfikacji potencjalnego punktu krytycznego, przeprowadzone zostały testy z rekordami zawierającymi 100,000 kolumn. Wyniki wykazały, że czas dodawania danych wyniósł 333,720 ms, a czas pobierania danych osiągnął 48,510 ms. Pomimo znacznego wzrostu liczby kolumn, system nadal był w stanie przetwarzać dane. Czas operacji uległ znacznemu wydłużeniu, co wskazuje na limit efektywności przy ekstremalnym obciążeniu.

3.6.12 Podsumowanie

Przeprowadzone testy wydajnościowe potwierdziły, że stworzony system do archiwizacji danych eksperymentalnych jest wysoce wydajny. Średnie czasy odpowiedzi dla większości operacji interfejsu wyniosły poniżej 50 ms, co zapewnia użytkownikom niemal natychmiastowy dostęp do danych. Nawet przy maksymalnym obciążeniu system wykazywał stabilność i niskie opóźnienia, co świadczy o jego solidności i gotowości do pracy w wymagających warunkach.

Testy skalowania wykazały, że system skutecznie zarządza dużą liczbą danych oraz intensywnymi operacjami użytkowników. Uzyskane wyniki pokazały, że wzrost czasu tworzenia i pobierania danych jest proporcjonalny do wzrostu ich liczby oraz rozmiaru, co świadczy o stabilności i przewidywalności systemu. Efektywność systemu potwierdza także niski czas operacji przy wzroście liczby kolumn danych, co dodatkowo podkreśla jego zaawansowane możliwości w kontekście zarządzania rozbudowanymi zestawami danych.

Dodatkowo, testy na takich rozmiarach danych wskazują na dobrą konfigurację serwera, ponieważ API sprawnie obsługuje duże żądania. Uzyskane wyniki świadczą o tym, że integracja aplikacji stworzonych w Hyperledger Fabric i Flask przebiegła pomyślnie.

Podsumowując, system jest nie tylko gotowy do obsługi obecnych potrzeb, ale również dobrze przygotowany na przyszłe wyzwania związane ze zwiększym obciążeniem i skalowaniem danych, co czyni go efektywnym narzędziem do zarządzania danymi eksperymentalnymi.

3.7 Wdrożenie Systemu

Po pomyślnych testach obie aplikacje składowe zostały wdrożone na serwer udostępniony przez Uniwersytet w Białymostku. Decyzja o wykorzystaniu tego serwera była podkutowana potrzebą zapewnienia stabilnej i bezpiecznej infrastruktury, niezbędnej do działania aplikacji o wysokiej dostępności i niezawodności.

3.7.1 System Operacyjny

Aplikacja została wdrożona na systemie operacyjnym Linux - Ubuntu LTS 19. Wybór tego systemu operacyjnego wynikał z jego długoterminowego wsparcia (LTS) oraz szerokiej kompatybilności z narzędziami deweloperskimi i środowiskami produkcyjnymi. Ubuntu LTS 19 zapewnia stabilność i regularne aktualizacje bezpieczeństwa, co umożliwia utrzymanie ciągłości działania aplikacji oraz zapewnienie ochronę przechowywanych danych.

Ubuntu 19 jest dedykowanym i zalecanym systemem operacyjnym przez Hyperledger Fabric. System ten wspiera narzędzia takie jak Docker i Git, które pomagają efektywnie zarządzać konteneryzacją i system wersji aplikacji. Konteneryzacja w Dockerze umożliwia nie tylko izolację środowiska, ale też jego konfigurację, co pozwala zachować spójność środowiska uruchomieniowego.

3.7.2 System Kontroli Wersji

System kontroli wersji to narzędzie, które umożliwia śledzenie zmian w kodzie źródłowym oprogramowania. Pozwala on deweloperom na zarządzanie różnymi wersjami plików, zachowanie historii zmian, a także na współpracę w zespole poprzez synchronizację i integrację kodu. Dzięki systemowi kontroli wersji możliwe jest:

- **Słedzenie historii zmian:** Każda zmiana w kodzie jest zapisywana z informacją o autorze, dacie i opisie zmiany, co pozwala na łatwe śledzenie i odtworzenie poprzednich wersji.
- **Praca zespołowa:** Wiele osób może pracować nad tym samym projektem jednocześnie, a system kontroli wersji pomaga w zarządzaniu i łączeniu ich pracy. Pomimo, że system został stworzony samodzielnie to możliwe jest, że będzie on rozwijany w przyszłości przez zespół developerów.
- **Bezpieczeństwo:** Kod źródłowy jest przechowywany w centralnym repozytorium, co zapewnia jego bezpieczeństwo i minimalizuje ryzyko utraty danych.

W celu zapewnienia powyższych aspektów, w trakcie tworzenia systemu wykorzystany został system kontroli wersji Git, a kod źródłowy przechowywany jest w zdalnych repozytoriach na platformie Github. Obie aplikacje składowe zostały umieszczone w osobnych, publicznych repozytoriach, które są dostępne pod następującymi adresami:

- **Hyperledger Fabric:** github.com/dawid628/Master-Thesis-Blockchain
- **Flask:** github.com/dawid628/Master-Thesis-Flask

3.7.3 Licencja

Licencja oprogramowania to formalne upoważnienie, które określa, jak oprogramowanie może być używane i dystrybuowane. Licencje open-source, takie jak licencja MIT, pozwalają użytkownikom na szerokie możliwości korzystania z oprogramowania, jednocześnie chroniąc prawa autora.

Stworzony system udostępniony jest na licencji MIT. Jest to jedna z najbardziej popularnych i liberalnych licencji open-source, co czyni ją atrakcyjną dla wielu projektów programistycznych. Elementy tej licencji obejmują:

- **Swobodne Użycie:** Każda osoba, która uzyska kopię oprogramowania i powiązanej dokumentacji, może korzystać z oprogramowania bez żadnych ograniczeń, w tym prawo do używania, kopowania, modyfikowania, łączenia, publikowania, dystrybuowania, sublicencjonowania i/lub sprzedaży kopii oprogramowania.
- **Warunki Dystrybucji:** Warunkiem jest dołączenie informacji o prawach autorskich oraz powiadomienia o zezwoleniu w każdej kopii lub istotnej części oprogramowania.
- **Brak Gwarancji:** Oprogramowanie jest dostarczane "tak jak jest", bez jakiegokolwiek gwarancji, w tym domyślnych gwarancji przydatności handlowej, przydatności do określonego celu oraz nienaruszalności praw. Autorzy lub posiadacze praw autorskich nie ponoszą odpowiedzialności za jakiekolwiek roszczenia, szkody lub inne zobowiązania wynikające z korzystania z oprogramowania.

Treść licencji umieszczonej w repozytoriach:

MIT License Copyright (c) 2024 Dawid Metelski

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Podsumowanie

Niniejsza praca miała na celu stworzenie platformy do archiwizacji danych eksperymentalnych zapewniającej bezpieczeństwo i niezmienność przechowywanych danych oraz zgodność z regulacjami prawnymi dotyczącymi wprowadzania nowych procedur medycznych. W projekcie wykorzystano zaawansowane rozwiązania takie jak blockchain, nierelacyjne bazy danych i framework internetowy, które zapewniły solidną podstawę dla rozwoju aplikacji.

Blockchain został zastosowany jako narzędzie do zapewnienia niezmienności i weryfikowalności danych. Jest to szczególnie istotne w kontekście danych medycznych, gdzie ważna jest możliwość śledzenia ich historii i autentyczności. Wykorzystanie nierelacyjnych baz danych umożliwiło efektywne zarządzanie dużymi ilościami danych, co przyczyniło się do wysokiej wydajności systemu, a framework internetowy pozwolił na stworzenie intuicyjnego interfejsu dla użytkowników.

W rozdziale dotyczącym analizy skupiono się na metodach śledzenia i poświadczania pochodzenia danych, czyli Data Lineage i Data Provenance. Te techniki pozwoliły na pełne zrozumienie ścieżki, jaką dane przechodzą od momentu ich generacji do wykorzystania w badaniach naukowych. Wprowadzenie tych metod umożliwiło dostosowanie technologii do systemu oraz weryfikację i potwierdzenie wiarygodności danych, co umożliwia dokładne prowadzenie badań.

Wnioski płynące z analizy potwierdzają, że integracja nowoczesnych technologii z procesami zarządzania danymi eksperymentalnymi może znaczco zwiększyć efektywność i jakość badań naukowych. Przez transparentność i dostępność danych, naukowcy mogą szybciej weryfikować hipotezy i prowadzić dalsze badania.

Przeprowadzone testy wydajnościowe potwierdziły, że stworzony system do archiwizacji danych eksperymentalnych jest wysoce wydajny. Średnie czasy odpowiedzi dla większości operacji interfejsu wyniosły poniżej 50 ms, co zapewnia użytkownikom niemal natychmiastowy dostęp do danych. Nawet przy maksymalnym obciążeniu system wykazywał stabilność i niskie opóźnienia, co świadczy o jego solidności i gotowości do pracy w wymagających warunkach.

Testy skalowania wykazały, że system skutecznie zarządza dużymi ilościami danych oraz intensywnymi operacjami użytkowników. Uzyskane wyniki pokazały, że wzrost czasu tworzenia i pobierania danych jest proporcjonalny do wzrostu ich ilości oraz rozmiaru, co świadczy o stabilności i przewidywalności systemu.

Ostateczne wyniki projektu potwierdzają, że zamierzony cel został osiągnięty. Opracowany serwis może skutecznie wspierać gromadzenie danych i ich analizę, również w przypadku wprowadzania nowych procedur medycznych. Projekt stanowi wartościowy wkład w dziedzinie analizy danych, dostarczając narzędzie, które nie tylko spełnia regulacyjne wymagania, ale także usprawnia zarządzanie danymi eksperymentalnymi. Platforma może być dostosowana do specyficznych wymogów różnych dziedzin naukowych,

oferując personalizowane narzędzia i interfejsy dla użytkowników z różnorodnymi potrzebami badawczymi.

Bibliografia

- [1] Data provenance in healthcare: Approaches, challenges, and future directions, 2023.
- [2] Doubling time, 2024.
- [3] Charu C. Aggarwal and ChengXiang Zhai. *Mining Text Data*. 2012.
- [4] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.
- [5] Andreas M. Antonopoulos. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. O'Reilly Media, 2017.
- [6] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Communications of the ACM*, 2010.
- [7] Sönke Bartling and Sascha Friesike. *Blockchain for Science and Knowledge Creation*. 2018.
- [8] Imran Bashir. *Mastering Blockchain - Third Edition*. 2020.
- [9] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. *2015 IEEE Symposium on Security and Privacy*, 2015.
- [10] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, omega, and kubernetes. 2016.
- [11] Vitalik Buterin. A next-generation smart contract and decentralized application platform. 2014.
- [12] Rajkumar Buyya, James Broberg, and Andrzej Goscinski. *Cloud Computing: Principles and Paradigms*. Wiley, 2013.
- [13] Rick Cattell. Scalable sql and nosql data stores. *ACM SIGMOD Record*, 2011.
- [14] David L. Chaum. Computer systems established, maintained and trusted by mutually suspicious groups. 1982.

- [15] Kyle Croman, Christian Decker, Ignacio Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gun Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains. 2016.
- [16] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. 2013.
- [17] Alex Tapscott Don Tapscott. *Blockchain Revolution*. 2016.
- [18] Daniel Drescher. *Blockchain Basics: A Non-Technical Introduction in 25 Steps*. Apress, 2017.
- [19] Daniel Drescher. *Blockchain. Podstawy technologii łańcucha bloków w 25 krokach*. 2018.
- [20] Ariel Ekblaw, Asaph Azaria, John D. Halamka, and Andrew Lippman. A case study for blockchain in healthcare: "medrec" prototype for electronic health records and medical research data. 2016.
- [21] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, 2010.
- [22] Martin Fowler and Pramod Sadalage. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley Professional, 2012.
- [23] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Blockchain Basic-Database Systems: The Complete Book*. Prentice Hall, 2008.
- [24] Rajini Sivaram Gwen Shapira, Todd Palino. *Kafka: The Definitive Guide*. 2021.
- [25] Jennifer Widom Hector Garcia-Molina, Jeffrey Ullman. *Database Systems: The Complete Book*. 2008.
- [26] Institute of Electrical and Electronics Engineers. *IEEE 802: Standards for local and metropolitan area networks*. IEEE, New York, USA, 2014.
- [27] International Organization for Standardization. *ISO 27001: Information Security Management*. ISO, Geneva, Switzerland, 2013.
- [28] International Organization for Standardization. *ISO 8000: Data quality*. ISO, Geneva, Switzerland, 2014.
- [29] International Organization for Standardization. *ISO 27040: Information technology — Security techniques — Storage security*. ISO, Geneva, Switzerland, 2015.
- [30] International Organization for Standardization. *ISO 13485: Medical devices — Quality management systems — Requirements for regulatory purposes*. ISO, Geneva, Switzerland, 2016.
- [31] Jay Kreps, Neha Narkhede, and Jun Rao. 2011.
- [32] Mark Anthony Morris Matt Zand, Xun (Brian) Wu. *Hands-On Smart Contract Development with Hyperledger Fabric V2*. 2021.

- [33] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014.
- [34] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008.
- [35] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.
- [36] Sejm Rzeczypospolitej Polskiej. Ustawa z dnia 6 września 2001 r. prawo farmaceutyczne. *Dziennik Ustaw*, 2001.
- [37] Sejm Rzeczypospolitej Polskiej. Ustawa o systemie informacji w ochronie zdrowia. <https://isap.sejm.gov.pl/>, 2011.
- [38] Melanie Swan. *Blockchain. Blueprint for a New Economy*. 2015.
- [39] Zaigham Mahmood Thomas Erl, Ricardo Puttini. *Cloud Computing: Concepts, Technology Architecture*. 2013.
- [40] European Union. Directive 2001/20/ec of the european parliament and of the council of 4 april 2001 on the approximation of the laws, regulations and administrative provisions of the member states relating to the implementation of good clinical practice in the conduct of clinical trials on medicinal products for human use. *Official Journal of the European Communities*, 2001.
- [41] Various. *GxP Guidelines: Good Practices for the Life Sciences Industry*. Regulatory agencies and industry standards, Global, 2018.
- [42] Paul Vigna and Michael J. Casey. *The Truth Machine: The Blockchain and the Future of Everything*. St. Martin's Press, 2018.
- [43] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *ethereum.github.io*, 2014.
- [44] A. Woodruff and M. Stonebraker. Supporting fine-grained data lineage in a database visualization environment. *Journal Name*, 1997.
- [45] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 2010.
- [46] Tomasz Żurawski and Bartosz Kosko. Apache nifi: A versatile platform for data ingestion and processing. 2018.

Spis tabel

3.1 Czasy tworzenia i pobierania zasobów w blockchain dla różnej liczby danych	78
3.2 Czasy tworzenia i pobierania zasobów w blockchain dla różnej liczby kolumn	79

Spis rysunków

3.1	Diagram przypadków użycia dla systemu zarządzania danymi eksperymentalnymi	41
3.2	Diagram Architektury Systemu	43
3.3	Szczegóły danych	55
3.4	Historia danych	56
3.5	Dodawanie danych	56
3.6	Lista użytkowników	57
3.7	Blokada użytkownika	57
3.8	Edycja użytkownika	57
3.9	Lista szpitali	58
3.10	Dodawanie szpitali	58
3.11	Usunięcie szpitala	58
3.12	Błąd usunięcia używanego szpitala	59
3.13	Lista szpitali	59
3.14	Test Wydajnościowy Listy Danych	61
3.15	Test Wydajnościowy Tworzenia Danych	63
3.16	Test Wydajnościowy Historii Danych	65
3.17	Test Wydajnościowy Listy Ról	67
3.18	Test Wydajnościowy Tworzenia Ról	69
3.19	Test Wydajnościowy Listy Szpitali	71
3.20	Test Wydajnościowy Tworzenia Szpitali	73
3.21	Test Wydajnościowy Listy Użytkowników	75
3.22	Test Wydajnościowy Tworzenia Użytkownika	77
3.23	Test Wydajnościowy dla Różnych Liczb Danych	78
3.24	Test Wydajnościowy dla Różnej Liczby Kolumn	79