**AKADEMIA NAUK STOSOWANYCH**

**W NOWYM SĄCZU**

**WYDZIAŁ NAUK INŻYNIERYJNYCH**

**PRACA DYPLOMOWA**

**APLIKACJA WEBOWA PRZECHOWUJĄCA PLIKI Z DOSTĘPEM CHRONIONYM**

**Autor: Bartłomiej Cetera**

**Kierunek: Informatyka stosowana**

**Nr albumu: 31230**

**Promotor: dr. Grzegorz Litawa**

**Akceptacja promotora: ……………………………………………………**  data i podpis

**NOWY SĄCZ 2024**

SPIS TREŚCI

[1. WPROWADZENIE 5](#_Toc185442150)

[1.1. WSTĘP 5](#_Toc185442151)

[1.2. ISTNIEJĄCE ROZWIĄZANIA 6](#_Toc185442152)

[1.3. PODSUMOWANIE ISTNIEJĄCYCH ROZWIĄZAŃ 7](#_Toc185442153)

[2. PROJEKT 8](#_Toc185442154)

[2.1. CEL I ZAKRES PRACY 8](#_Toc185442155)

[2.2. WYMAGANIA FUNKCJONALNE 9](#_Toc185442156)

[2.3. WYMAGANIA NIEFUNKCJONALNE 13](#_Toc185442157)

[2.4. TECHNOLOGIE 14](#_Toc185442158)

[2.5. DIAGRAMY 16](#_Toc185442159)

[2.5.1. DIAGRAM PRZYPADKÓW UŻYCIA 16](#_Toc185442160)

[2.5.1.1. DIAGRAM PRZYPADKÓW UŻYCIA - UŻYTKOWNIK 17](#_Toc185442161)

[2.5.1.2. DIAGRAM PRZYPADKÓW UŻYCIA - MAGAZYNY 18](#_Toc185442162)

[2.5.2. DIAGRAM ERD 19](#_Toc185442163)

[2.5.2.1. DIAGRAM ERD - BAZA SDRIVE\_LOGINS 20](#_Toc185442164)

[2.5.2.2. DIAGRAM ERD - BAZA SDRIVE\_FILES 22](#_Toc185442165)

[2.5.3. DIAGRAMY AKTYWNOŚCI 23](#_Toc185442166)

[2.5.3.1. DIAGRAM AKTYWNOŚCI - LOGOWANIE 23](#_Toc185442167)

[2.5.3.2. DIAGRAM AKTYWNOŚCI - WYSŁANIE WIADOMOŚCI DO NOWEGO ZNAJOMEGO 24](#_Toc185442168)

[2.5.3.4. DIAGRAM AKTYWNOŚCI - PRACA NA NOWYM MAGAZYNIE 25](#_Toc185442169)

[2.5.3.5. DIAGRAM AKTYWNOŚCI - PRACA NA WSPÓLNYM PLIKU W MAGAZYNIE 27](#_Toc185442170)

[2.6. SCENARIUSZE UŻYCIA 28](#_Toc185442171)

[2.6.1. SCENARIUSZ UŻYCIA - LOGOWANIE 28](#_Toc185442172)

[2.6.2. SCENARIUSZ UŻYCIA - REJESTRACJA 28](#_Toc185442173)

[2.6.3. SCENARIUSZ UŻYCIA - PRZESYŁ PLIKÓW DO SYSTEMU 29](#_Toc185442174)

[2.6.4. SCENARIUSZ UŻYCIA - UDOSTĘPNIANIE PLIKÓW W SYSTEMIE 29](#_Toc185442175)

[2.6.5. SCENARIUSZ UŻYCIA - ZAPRASZANIE NOWYCH ZNAJOMYCH 30](#_Toc185442176)

[2.6.6. SCENARIUSZ UŻYCIA - KONWERSACJA ZE ZNAJOMYM 30](#_Toc185442177)

[2.6.7. SCENARIUSZ UŻYCIA - TWORZENIE MAGAZYNU 31](#_Toc185442178)

[2.6.8. SCENARIUSZ UŻYCIA - EDYCJA PLIKU W MAGAZYNIE 31](#_Toc185442179)

[2.6.9. SCENARIUSZ UŻYCIA - PRZYPISYWANIE ZNAJOMYCH DO MAGAZYNU 32](#_Toc185442180)

[2.6.10. SCENARIUSZ UŻYCIA - PRZYPISYWANIE NOWYCH PLIKÓW DO MAGAZYNU 32](#_Toc185442181)

[3. IMPLEMENTACJA 33](#_Toc185442182)

[3.1. IMPLEMENTACJA - SYSTEM LOGOWANIA I REJESTRACJI 33](#_Toc185442183)

[3.1.1. SYSTEM LOGOWANIA I REJESTRACJI - OBRAZY 33](#_Toc185442184)

[3.1.1. SYSTEM LOGOWANIA I REJESTRACJI - LISTINGI 34](#_Toc185442185)

[3.2. IMPLEMENTACJA - SYSTEM ZARZĄDZANIA PLIKAMI 36](#_Toc185442186)

[3.2.1. SYSTEM ZARZĄDZANIA PLIKAMI - OBRAZY 36](#_Toc185442187)

[3.2.1. SYSTEM ZARZĄDZANIA PLIKAMI - LISTINGI 37](#_Toc185442188)

[3.3. IMPLEMENTACJA - SYSTEM ZNAJOMYCH 40](#_Toc185442189)

[3.3.1. SYSTEM ZNAJOMYCH - OBRAZY 40](#_Toc185442190)

[3.3.2. SYSTEM ZNAJOMYCH - LISTINGI 41](#_Toc185442191)

[3.4. IMPLEMENTACJA - SYSTEM MAGAZYNÓW 43](#_Toc185442192)

[3.4.1. SYSTEM MAGAZYNÓW - OBRAZY 43](#_Toc185442193)

[3.4.2. SYSTEM MAGAZYNÓW - LISTINGI 46](#_Toc185442194)

[3.5. IMPLEMENTACJA - SYSTEM DWUSTOPNIWEJ AUTENTYFIKACJI 51](#_Toc185442195)

[3.5.1. SYSTEM DWUSTOPNIWEJ AUTENTYFIKACJI - OBRAZY 51](#_Toc185442196)

[3.5.2. SYSTEM DWUSTOPNIWEJ AUTENTYFIKACJI - LISTINGI 51](#_Toc185442197)

[3.6. IMPLEMENTACJA - SYSTEM SZYFROWANIA 51](#_Toc185442198)

[3.6.1. SYSTEM SZYFROWANIA - LISTINGI 51](#_Toc185442199)

[3.7. IMPLEMENTACJA - SYSTEM POWIADOMIEŃ I WIADOMOŚCI 51](#_Toc185442200)

[3.7.1. SYSTEM POWIADOMIEŃ I WIADOMOŚCI - OBRAZY 51](#_Toc185442201)

[3.7.2. SYSTEM POWIADOMIEŃ I WIADOMOŚCI - LISTINGI 52](#_Toc185442202)

[4. PODSUMOWANIE 53](#_Toc185442203)

[5. BIBLIOGRAFIA 54](#_Toc185442204)

[5.1. BIBLIOGRAFIA - LINKI 54](#_Toc185442205)

[5.1. BIBLIOGRAFIA - KSIĄŻKI 54](#_Toc185442206)

[6. ZAŁĄCZNIKI 54](#_Toc185442207)

# 1. WPROWADZENIE

## 1.1. WSTĘP

Wraz z dynamicznym rozwojem technologii oraz coraz powszechniejszym dostępem   
do internetu, przechowywanie plików w chmurze stało się istotnym elementem codziennego życia zarówno użytkowników indywidualnych, jak i firm. Chmura, czyli system umożliwiający przechowywanie plików na serwerach dostępnych przez internet, pozwala na korzystanie z danych niezależnie od miejsca i urządzenia. Jest to nowoczesne rozwiązanie, które eliminuje konieczność fizycznego przechowywania plików na dyskach lokalnych, zapewniając użytkownikom wygodę, elastyczność i dostępność danych w dowolnym momencie. Podstawową zaletą chmury jest możliwość synchronizacji plików pomiędzy różnymi urządzeniami. Użytkownik może rozpocząć pracę na jednym urządzeniu, a następnie kontynuować ją na innym, mając pewność, że wszystkie zmiany są zapisywane na bieżąco. Dodatkowo, chmura umożliwia łatwe udostępnianie plików innym osobom, co znacząco ułatwia współpracę zarówno w środowisku zawodowym, jak   
i w życiu codziennym. Rozwiązania chmurowe oferują również wiele korzyści związanych   
z bezpieczeństwem danych. Przechowywanie plików w chmurze eliminuje ryzyko ich utraty   
w wyniku awarii urządzenia lokalnego, takich jak uszkodzenie dysku twardego czy przypadkowe skasowanie. Serwery chmurowe często są wyposażone w zaawansowane mechanizmy zabezpieczające, takie jak szyfrowanie danych czy regularne tworzenie kopii zapasowych,   
co zwiększa ich odporność na ataki zewnętrzne i inne zagrożenia. Chociaż korzystanie z chmury jest niezwykle wygodne, wiąże się także z pewnymi wyzwaniami. Wielu użytkowników   
ma obawy dotyczące prywatności i bezpieczeństwa przechowywanych danych. Popularne usługi chmurowe, takie jak Google Drive, OneDrive czy Dropbox, często zbierają i analizują dane użytkowników, co rodzi pytania o ochronę prywatności. W przypadku naruszenia bezpieczeństwa serwera dane przechowywane w chmurze mogą zostać ujawnione lub skradzione. Te zagrożenia sprawiają, że użytkownicy coraz częściej poszukują alternatywnych rozwiązań, które zapewnią im większą kontrolę nad przechowywanymi plikami. Rozwój technologii chmurowych otworzył nowe możliwości w zakresie przechowywania danych, ale jednocześnie postawił przed nami szereg wyzwań związanych z bezpieczeństwem, prywatnością oraz efektywnością zarządzania zasobami cyfrowymi. Zrozumienie tych aspektów jest kluczowe dla każdego, kto chce świadomie korzystać z chmury jako narzędzia ułatwiającego życie i pracę w środowisku cyfrowym.

## 1.2. ISTNIEJĄCE ROZWIĄZANIA

Obecnie na rynku istnieje wiele usług pozwalających na przechowywanie plików w chmurze, które zdobyły popularność ze względu na swoją dostępność, wygodę i funkcjonalność. Platformy te są wykorzystywane zarówno w życiu codziennym, jak i w środowisku zawodowym. Niestety, mimo licznych zalet, wiele   
z tych usług posiada istotne ograniczenia i wyzwania, które mogą zniechęcać użytkowników lub wpływać na ich zaufanie do tych platform.

* **Dysk Google** – *usługa do przechowywania, synchronizacji i udostępniania plików, stworzona przez Google. Uruchomiona 24 kwietnia 2012 r. Dysk Google pozwala użytkownikom przechowywać pliki na swoich serwerach, synchronizować   
  je na różnych urządzeniach i udostępniać. Oprócz strony internetowej Dysk Google oferuje aplikacje z funkcjami offline dla komputerów z systemami Windows i MacOS oraz smartfonów i tabletów z Androidem i iOS. Dysk Google obejmuje Dokumenty Google, Arkusze i Prezentacje Google, pakiet biurowy umożliwiający wspólne edytowanie dokumentów, arkuszy kalkulacyjnych, prezentacji, rysunków, formularzy   
  i innych elementów. Pliki utworzone i edytowane za pośrednictwem pakietu biurowego są zapisywane na Dysku Google.*

[źródło: https://pl.wikipedia.org/wiki/Dysk\_Google]

* **Dropbox** - *usługa świadczona przez Dropbox, Inc. polegająca na udostępnieniu przestrzeni dyskowej na serwerach tej firmy. Wysyłanie, przeglądanie i pobieranie danych jest możliwe poprzez przeglądarkę internetową lub poprzez dedykowaną aplikację zainstalowaną na komputerze. W wersji darmowej dostępne jest 2 GB miejsca na serwerach, a zwiększenie tej przestrzeni wymaga uiszczenia cyklicznej opłaty (jednak korzystając z różnych ofert można powiększyć swoją przestrzeń dyskową za darmo – zaproszenie nowego użytkownika, instalacja oficjalnej aplikacji etc.). Zapraszając nowych użytkowników można zwiększyć przestrzeń dyskową maksymalnie o 16 GB.*

[źródło: https://pl.wikipedia.org/wiki/Dropbox]

* **Microsoft One Drive** - *OneDrive udostępnia 5 GB darmowego miejsca z możliwością rozszerzenia pojemności do 100 GB. Posiadacze subskrypcji Microsoft 365 otrzymują dodatkowo 1 TB miejsca na swoje pliki. Ponadto pojedynczy plik może mieć maksymalnie 100 GB. Oficjalnie wspierane są przeglądarki Internet Explorer i Mozilla Firefox, jednak możliwe jest także korzystanie z usługi za pomocą innych. Wyjątkiem jest funkcja przeciągania plików, która wymaga instalacji kontrolki ActiveX, działa więc tylko w Internet Explorerze 6+ i Mozilli Firefox 1.5+.*

[źródło: https://pl.wikipedia.org/wiki/OneDrive]

* **iCloud** - *zestaw usług online oraz oprogramowania przedsiębiorstwa Apple zaprezentowany na WWDC 6 czerwca 2011 roku, zastępujący wcześniejsze MobileMe. iCloud dostarcza i synchronizuje usługi sieciowe dla telefonu iPhone, tabletu iPad, odtwarzacza iPod touch oraz dla komputerów Windows i Mac. Można go obsługiwać poprzez dwa interfejsy: przeglądarkowy oraz oprogramowanie zainstalowane   
  na komputerze.*

[źródło: https://pl.wikipedia.org/wiki/ICloud]

## 1.3. PODSUMOWANIE ISTNIEJĄCYCH ROZWIĄZAŃ

Jedną z głównych zalet popularnych usług takich jak Google Drive i OneDrive jest ich konkurencyjna cena, co czyni je atrakcyjnymi dla użytkowników końcowych. Niestety,   
te rozwiązania wiążą się z istotnymi kompromisami, które często nie są w pełni zrozumiane przez użytkowników. W zamian za niskie koszty użytkownicy tracą prywatność i kontrolę nad przechowywanymi treściami, co wynika z polityki tych firm. Google i Microsoft mają pełen dostęp do plików przechowywanych na ich platformach i rutynowo skanują je w poszukiwaniu potencjalnie niebezpiecznych treści. Co więcej, te korporacje podejmują działania takie jak przeszukiwanie plików tekstowych w celu identyfikowania haseł do zabezpieczonych archiwów, tłumacząc te działania troską o bezpieczeństwo użytkownika. Takie praktyki wzbudzają uzasadnione obawy o inwazyjność i brak prywatności. Dodatkowo, stabilność usług tych platform bywa problematyczna. Wielokrotnie zgłaszano przypadki znikania plików, nawet przez użytkowników posiadających konta premium. Tego rodzaju sytuacje podważają zaufanie do usług oferowanych przez gigantów technologicznych, a użytkownicy nie mają pełnej pewności, że ich dane są bezpieczne i dostępne w każdej chwili. Z kolei Dropbox Inc. prezentuje odmienną strategię, która również budzi kontrowersje. Darmowy plan tej usługi oferuje zaledwie   
2 GB miejsca, co w dzisiejszych czasach jest wartością bardzo ograniczoną, szczególnie   
w porównaniu z darmowymi ofertami konkurencji. Co więcej, liczba urządzeń, które mogą mieć dostęp do konta, jest ograniczona do dwóch, co znacznie ogranicza elastyczność użytkowania. Plany płatne Dropboxa są znacznie droższe niż u konkurencji, zaczynając od około 900 zł rocznie za 2 TB miejsca. Takie podejście zmusza użytkowników do zakupu znacznie większej przestrzeni, niż faktycznie potrzebują, co może być odbierane jako brak dostosowania oferty   
do indywidualnych wymagań klientów. W rezultacie brak jest opcji pośrednich między darmowym planem a kosztownymi subskrypcjami premium, co ogranicza dostępność tej usługi dla wielu potencjalnych użytkowników. Niepokojący jest również fakt, że Dropbox dzieli się danymi telemetrycznymi swoich użytkowników z partnerami biznesowymi, co rodzi pytania   
o prywatność i przejrzystość polityki firmy. Co gorsza, firma była wielokrotnie krytykowana   
za brak konsekwencji w realizacji własnych zasad. Mimo zapewnień, że usunięte pliki są trwale usuwane z serwerów, użytkownicy zgłaszali przypadki, w których dawno usunięte dane pojawiały się ponownie. Sugeruje to możliwość monitorowania i przechowywania treści nawet po ich usunięciu, co może być postrzegane jako naruszenie zasad etycznych oraz prywatności użytkowników. Podsumowując, zarówno Google Drive, OneDrive, jak i Dropbox prezentują modele biznesowe, które wiążą się z ukrytymi kosztami – czy to w postaci utraty prywatności, braku przejrzystości zasad, czy wysokich cen za usługi premium. Te praktyki podkreślają potrzebę poszukiwania alternatyw, które oferują lepszy balans między kosztem, funkcjonalnością a ochroną danych użytkownika. Wzrasta zapotrzebowanie na usługi, które nie tylko są przystępne cenowo, ale także respektują prywatność i zapewniają bezpieczeństwo danych.

# 2. PROJEKT

## 2.1. CEL I ZAKRES PRACY

Temat pracy dyplomowej skupia się na zaprojektowaniu oraz implementacji internetowej aplikacji do przechowywania danych, która w odpowie na współczesne potrzeby użytkowników w zakresie bezpieczeństwa, prywatności i zarządzania plikami w środowisku online. Głównym celem aplikacji jest stworzenie narzędzia, które nie tylko zapewni użytkownikom możliwość przechowywania plików, ale także pozwoli im na pełną kontrolę nad dostępem do swoich zasobów, a jednocześnie ułatwi współpracę i interakcję z innymi użytkownikami. Projektowana aplikacja powinna być przystosowana do funkcjonowania jako prywatny magazyn internetowy, który stawia na ochronę prywatności użytkownika. System będzie w pełni szyfrować dane, eliminując ryzyko nieautoryzowanego dostępu, nawet w przypadku naruszenia bezpieczeństwa serwera. Wykorzystanie nowoczesnych technologii, takich jak algorytmy szyfrowania AES, zapewni bezpieczeństwo przechowywanych informacji, dając użytkownikom pewność, że ich dane są chronione przed zagrożeniami zewnętrznymi. Kolejnym kluczowym aspektem aplikacji jest wsparcie współpracy między użytkownikami. Aplikacja powinna umożliwiać tworzenie dedykowanych przestrzeni współdzielonych, nazywanych magazynami, w których użytkownicy będą mogli gromadzić, organizować i wspólnie edytować pliki. System kontroli wersji plików pozwoli na jednoczesną pracę kilku osób nad jednym dokumentem, a w razie wystąpienia konfliktów umożliwi ich rozwiązanie poprzez porównanie różnic między wersjami i wybór najbardziej odpowiedniej. Projektowana aplikacja powinna również odpowiadać na potrzeby związane z zarządzaniem plikami w sposób przyjazny użytkownikowi. Użytkownicy powinni mieć możliwość łatwego przesyłania, pobierania, zmieniania nazw oraz usuwania plików. Szczególny nacisk należy położyć na przejrzysty interfejs, który pozwoli na intuicyjne korzystanie z dostępnych funkcji, takich jak zarządzanie znajomymi, wysyłanie wiadomości oraz przeglądanie udostępnionych zasobów. Ważnym celem aplikacji jest również zapewnienie użytkownikom możliwości edycji wybranych typów plików bez konieczności ich pobierania. Dzięki integracji   
z odpowiednimi narzędziami użytkownicy będą mogli edytować pliki tekstowe i dokumenty Worda bezpośrednio w przeglądarce, co zwiększy wygodę i efektywność korzystania z systemu. Podsumowując, celem projektu jest stworzenie aplikacji, która stanie się alternatywą dla istniejących rozwiązań, takich jak Google Drive czy Dropbox, oferując przy tym większe bezpieczeństwo, prywatność oraz bardziej elastyczne plany użytkowania. Projektowana aplikacja ma na celu zbudowanie zaufania użytkowników poprzez wdrożenie zaawansowanych funkcji zarządzania plikami, wysokiego poziomu ochrony danych oraz łatwego udostępniania   
i współpracy w ramach środowiska cyfrowego.

## 2.2. WYMAGANIA FUNKCJONALNE

1. **System rejestracji** - Aplikacja powinna oferować funkcjonalność rejestracji, która pozwoli nowym użytkownikom na utworzenie konta. W procesie rejestracji użytkownicy powinni podawać podstawowe informacje, takie jak adres e-mail, hasło oraz imię   
   i nazwisko. Hasła powinny być przechowywane w sposób bezpieczny, poprzez ich haszowanie przy użyciu algorytmów, takich jak bcrypt, co uniemożliwi ich odczytanie   
   w przypadku naruszenia bezpieczeństwa bazy danych. Logowanie do aplikacji powinno odbywać się za pomocą podanego podczas rejestracji adresu e-mail i hasła. Dla zwiększenia poziomu bezpieczeństwa system powinien wspierać opcjonalne dwuetapowe uwierzytelnianie (2FA). W tej funkcji użytkownik, oprócz standardowego hasła, musi podać jednorazowy kod weryfikacyjny, który może być wysyłany na adres e-mail. Autoryzacja użytkowników powinna opierać się na wykorzystaniu tokenów JWT (JSON Web Token), które umożliwiają uwierzytelnianie i autoryzowanie użytkowników bez konieczności przechowywania ich danych w sesji serwera. Po zalogowaniu użytkownika, aplikacja powinna generować unikalny token JWT, który będzie zawierał zaszyfrowane informacje o użytkowniku, takie jak jego identyfikator. Tokeny mogą być przechowywane w pamięci przeglądarki (np. cookies) i wykorzystywane do autoryzacji przy każdym żądaniu do serwera. System powinien również umożliwiać użytkownikom edycję ich danych osobowych po zalogowaniu. Użytkownicy powinni mieć możliwość zmiany swojego imienia, nazwiska, adresu e-mail czy hasła.
2. **Zarządzanie plikami –** Projektowany system zarządzania plikami powinien zapewniać użytkownikom pełen zakres funkcjonalności związanych z obsługą ich plików w ramach dedykowanego, indywidualnego obszaru przechowywania. Użytkownicy powinni mieć możliwość przesyłania plików za pomocą interfejsu, który pozwoli im na dodanie plików z urządzenia. Podczas przesyłania plików aplikacja powinna przechwytywać i zapisywać metadane, takie jak oryginalna nazwa pliku, data przesłania, typ pliku oraz identyfikator użytkownika będącego jego właścicielem. Metadane te powinny być przechowywane   
   w tabeli w bazie danych, co umożliwi zarządzanie i przeszukiwanie zasobów. Każdy użytkownik powinien mieć dostęp do dedykowanego widoku, w którym będą widoczne wszystkie pliki znajdujące się w jego obszarze przechowywania. Użytkownicy powinni mieć możliwość przeglądania szczegółowych informacji o każdym pliku oraz szybkiego wykonywania operacji, takich jak pobieranie, zmiana nazwy, usuwanie lub udostępnianie. Funkcja zmiany nazwy pliku powinna być dostępna z poziomu interfejsu użytkownika   
   i umożliwiać edytowanie nazwy wybranego pliku bez konieczności jego ponownego przesyłania. Zmiana nazwy powinna być zapisywana zarówno w bazie danych, jak   
   i na poziomie przechowywanego pliku, a aplikacja powinna zapewniać spójność tych danych. System powinien również umożliwiać użytkownikom usuwanie plików z ich przestrzeni przechowywania. Po potwierdzeniu decyzji o usunięciu plik powinien być usuwany po przez zmianę jego statusu w bazie danych „*soft delete*”. Funkcjonalność udostępniania plików powinna pozwalać użytkownikom na dzielenie się wybranymi zasobami z innymi osobami. System powinien umożliwiać wybór odbiorców z listy znajomych oraz definiowanie poziomu dostępu, takiego jak możliwość odczytu, edycji lub dalszego udostępniania. Udostępnione pliki powinny być widoczne w dedykowanej sekcji aplikacji, zarówno odbiorców, co zapewni dostęp do współdzielonych zasobów.
3. **Szyfrowanie danych** - Projektowany system powinien zapewniać pełne bezpieczeństwo   
   i prywatność danych użytkowników, dlatego kluczowym elementem funkcjonalności aplikacji musi być implementacja szyfrowania danych. Wszystkie pliki przechowywane w aplikacji powinny być szyfrowane na serwerze. Proces ten ma na celu ochronę zawartości plików przed nieautoryzowanym dostępem, nawet w przypadku naruszenia bezpieczeństwa serwera. Szyfrowanie powinno być realizowane za pomocą nowoczesnych i uznanych algorytmów kryptograficznych, takich jak AES (Advanced Encryption Standard) lub DES (Data Encryption Standard). AES, ze względu na swoją wyższą wydajność i odporność na ataki, powinien być preferowanym algorytmem. System powinien umożliwiać szyfrowanie plików za pomocą kluczy kryptograficznych generowanych indywidualnie dla każdego użytkownika lub magazynu plików. Klucze   
   te powinny być przechowywane w sposób bezpieczny, np. w zaszyfrowanej bazie danych lub dedykowanym menedżerze kluczy. Aplikacja powinna również umożliwiać użytkownikom odzyskanie dostępu do zaszyfrowanych plików w przypadku utraty kluczy szyfrujących. Może to być realizowane za pomocą funkcji odzyskiwania danych, która będzie opierać się na przechowywaniu kopii kluczy w bezpiecznym środowisku   
   (np. za pomocą techniki podziału klucza – tzw. key escrow). Dzięki wdrożeniu takiego systemu szyfrowania aplikacja zapewni pełne bezpieczeństwo danych, budując zaufanie użytkowników oraz zgodność z regulacjami dotyczącymi ochrony danych, takimi jak RODO (GDPR). W rezultacie użytkownicy będą mieli pewność, że ich dane są bezpieczne i prywatne, niezależnie od okoliczności.
4. **Edycja plików Word i TXT** - Aplikacja powinna wspierać przeglądanie i edycję plików tekstowych, takich jak .txt oraz dokumentów .docx (Word). Dzięki integracji   
   z odpowiednimi narzędziami do konwersji i edycji plików użytkownicy mogliby modyfikować dokumenty bez konieczności pobierania ich na urządzenie lokalne, a zmiany mogłyby być zapisywane bezpośrednio w systemie.
5. **System znajomych -** Projektowany system znajomych w aplikacji powinien zapewniać użytkownikom intuicyjne i rozbudowane narzędzia do zarządzania relacjami z innymi użytkownikami, wspierając zarówno współpracę, jak i komunikację. Użytkownicy powinni mieć możliwość wyszukiwania innych osób w systemie za pomocą wyszukiwarki, która umożliwi odnalezienie znajomych na podstawie ich adresu e-mail lub innych dostępnych identyfikatorów. Po odnalezieniu odpowiedniej osoby, użytkownik powinien mieć możliwość wysłania zaproszenia do znajomych, które druga osoba może zaakceptować lub odrzucić. W przypadku akceptacji zaproszenia, osoba ta zostaje dodana do listy znajomych, która powinna być widoczna w specjalnej sekcji w aplikacji. Użytkownicy powinni mieć możliwość przeglądania podstawowych informacji o swoich znajomych, takich jak ich imię, nazwisko. Powinna istnieć możliwość szybkiego usuwania znajomych z listy, jeśli relacja z daną osobą nie jest już potrzebna. System znajomych powinien wspierać udostępnianie plików oraz magazynów w sposób szybki i intuicyjny. Właściciel pliku lub magazynu powinien mieć możliwość wybrania osoby z listy znajomych, aby udostępnić im wybrane zasoby. Udostępnione pliki i magazyny powinny być dostępne w specjalnej sekcji aplikacji dedykowanej współdzielonym zasobom, dzięki czemu użytkownicy mogą odnaleźć to, co zostało im udostępnione. Komunikacja między użytkownikami powinna odbywać się za pomocą wbudowanego systemu wiadomości. Aplikacja powinna umożliwiać użytkownikom wysyłanie prywatnych wiadomości   
   do swoich znajomych. Użytkownicy powinni otrzymywać informacje o nowych zaproszeniach do znajomych, wiadomościach.
6. **System magazynów** – System magazynów w projektowanej aplikacji powinien oferować kompleksowe zarządzanie plikami w dedykowanych zbiorach, które będą mogły być współdzielone z innymi użytkownikami. Użytkownicy powinni mieć możliwość tworzenia magazynów, które będą pełniły rolę kontenerów dla plików, pozwalając na ich przejrzystą organizację. Każdy magazyn powinien posiadać możliwość nadania mu nazwy, aby ułatwić identyfikację jego zawartości. Aplikacja powinna umożliwiać udostępnianie magazynów innym użytkownikom, aby wspierać współpracę nad przechowywanymi plikami. System powinien pozwalać właścicielowi magazynu na bieżące zarządzanie listą osób mających dostęp, umożliwiając dodawanie użytkowników, usuwanie dostępu. Kluczowym elementem systemu magazynów powinna być kontrola wersji plików, która umożliwi wspólną pracę wielu osób na danym pliku. W przypadku, gdy kilku użytkowników edytuje ten sam plik jednocześnie, system powinien automatycznie wykrywać konflikty wersji i umożliwiać ich rozwiązanie poprzez porównanie różnic   
   i wybór właściwej wersji.

## 2.3. WYMAGANIA NIEFUNKCJONALNE

1. **Interfejs** **-** Aplikacja powinna być zaprojektowana w taki sposób, aby interfejs użytkownika był prosty, przejrzysty i intuicyjny, umożliwiając użytkownikom szybki dostęp do wszystkich kluczowych funkcji w maksymalnie pięciu kliknięciach. Oznacza to, że użytkownik, korzystając z aplikacji, nie powinien być zmuszony do wykonywania zbędnych czynności lub przechodzenia przez skomplikowane struktury menu, aby osiągnąć zamierzony cel. Interfejs powinien być responsywny, dostosowany do różnych rozdzielczości ekranów. Dzięki temu użytkownicy będą mogli łatwo korzystać z aplikacji niezależnie od miejsca i urządzenia, z którego korzystają. Kluczowe funkcje, takie jak przesyłanie plików, zarządzanie magazynami, udostępnianie zasobów czy komunikacja ze znajomymi, powinny być widoczne i dostępne z poziomu głównego ekranu lub intuicyjnych ikon nawigacyjnych.
2. **Skalowalność** - Aplikacja powinna być zaprojektowana z myślą o skalowalności, co oznacza, że jej architektura i infrastruktura muszą umożliwiać łatwe rozszerzanie zarówno pod kątem funkcjonalności, jak i obsługi rosnącej liczby użytkowników, bez negatywnego wpływu na wydajność systemu. Skalowalność jest kluczowym aspektem nowoczesnych aplikacji, szczególnie w kontekście systemów, które mają być rozwijane i użytkowane w długiej perspektywie czasowej. Elastyczna architektura aplikacji powinna opierać się na modularności, dzięki której nowe funkcje mogą być dodawane w postaci niezależnych modułów. Każdy moduł powinien być odpowiedzialny za określoną funkcję, taką jak zarządzanie plikami, kontrola dostępu czy system wiadomości. Taki podejście pozwala na rozwój poszczególnych elementów aplikacji bez wpływu na istniejące funkcjonalności. Na przykład, jeżeli zajdzie potrzeba wprowadzenia nowego modułu do obsługi wersjonowania plików, powinno być możliwe jego wdrożenie bez ingerencji w działanie podstawowych funkcji aplikacji, takich jak rejestracja czy zarządzanie znajomymi.
3. **Dostępność** - System powinien być zaprojektowany w sposób zapewniający pełną kompatybilność z wszystkimi popularnymi przeglądarkami internetowymi, takimi jak Google Chrome, Mozilla Firefox, Microsoft Edge oraz Safari, w ich najnowszych stabilnych wersjach. Zapewnienie zgodności z szeroką gamą przeglądarek jest kluczowe, aby użytkownicy mieli dostęp do aplikacji niezależnie od preferowanego środowiska pracy. Dzięki temu aplikacja może dotrzeć do szerszego grona odbiorców i zagwarantować spójne doświadczenie użytkownika bez względu na wybór przeglądarki.Tokeny autoryzacyjne JWT powinny być ważne maksymalnie przez 60 minut od ostatniego żądania użytkownika.
4. **Spójność** - W projektowanej aplikacji proces usuwania plików przez użytkowników powinien opierać się na mechanizmie „soft delete”. Zamiast natychmiastowego, fizycznego usuwania plików z bazy danych lub systemu plików, system powinien jedynie zmieniać status danego pliku w bazie danych. Dzięki temu aplikacja zapewni dodatkowy poziom bezpieczeństwa oraz pozwoli użytkownikom na ewentualne przywrócenie plików w przyszłości, minimalizując ryzyko przypadkowej utraty danych.

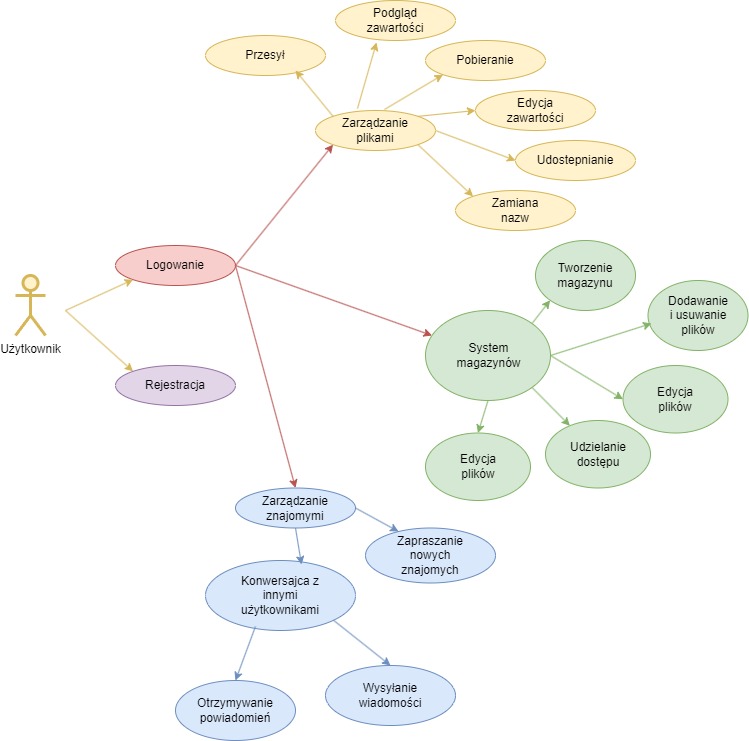
## 2.4. TECHNOLOGIE

* + **JavaScript** - to język programowania wysokiego poziomu używany głównie w tworzeniu dynamicznych stron internetowych. Początkowo zaprojektowany jako język skryptowy   
    do przeglądarek, dziś jest jednym z najpopularniejszych języków wykorzystywanych nie tylko po stronie klienta, ale również po stronie serwera (dzięki Node.js). JavaScript pozwala na tworzenie interaktywnych aplikacji internetowych, manipulowanie strukturą HTML   
    i CSS w czasie rzeczywistym oraz integrację z backendem aplikacji. Język wspiera różne paradygmaty programowania, w tym programowanie obiektowe i funkcjonalne.
  + **Node.js -** to środowisko uruchomieniowe JavaScript, które umożliwia wykonywanie kodu JavaScript po stronie serwera. Dzięki Node.js programiści mogą tworzyć aplikacje serwerowe w JavaScript, co pozwala na pełne wykorzystanie tego języka zarówno po stronie klienta, jak i serwera. Node.js jest oparty na silniku V8 Google, co sprawia, że jest bardzo wydajny. Obsługuje asynchroniczne operacje wejścia/wyjścia, co czyni go idealnym   
    do obsługi dużych obciążeń i jednoczesnych żądań. Node.js wspiera moduły, co pozwala   
    na łatwe zarządzanie i organizowanie kodu.
  + **Express.js -** to minimalistyczny framework dla Node.js, który umożliwia tworzenie aplikacji sieciowych i API w szybki i przejrzysty sposób. Zapewnia elastyczność i prostotę, umożliwiając programistom łatwe tworzenie aplikacji sieciowych dzięki gotowym funkcjom, takim jak routowanie, obsługa żądań HTTP i middleware. Express pozwala   
    na rozwój zarówno prostych aplikacji, jak i złożonych aplikacji typu SPA i mikroserwisów, dzięki czemu jest popularny wśród programistów Node.js.
  + **JWT (JSON Web Token) -** to otwarty standard (RFC 7519) używany do bezpiecznego przekazywania informacji między stronami jako obiekt JSON. JWT jest szeroko stosowany do uwierzytelniania w aplikacjach internetowych, gdzie po pomyślnym zalogowaniu użytkownik otrzymuje token, który jest następnie wysyłany w każdym kolejnym żądaniu jako dowód autentyczności. Tokeny są bezpieczne, ponieważ są podpisane kryptograficznie, dzięki czemu nie można ich łatwo podrobić, co sprawia, że są one często wykorzystywane do tworzenia bezpiecznych sesji w aplikacjach sieciowych.
  + **MySQL** - to dobrze znany system zarządzania relacyjnymi bazami danych (RDBMS), który wykorzystuje Structured Query Language (SQL) do manipulowania danymi. Został zaprojektowany przez szwedzką firmę MySQL AB w 1995 roku, a jego rozwój jest obecnie kontynuowany przez Oracle Corporation, która przejęła firmę w 2010 roku. MySQL jest znany ze swojej wydajności, niezawodności i łatwości użytkowania. MySQL jest dostępny na licencji GNU General Public License, co oznacza, że użytkownicy mogą go swobodnie pobierać, używać i modyfikować bez ponoszenia kosztów licencyjnych. Dostępna jest również komercyjna edycja MySQL, która zapewnia dodatkowe funkcje i wsparcie, idealne dla firm poszukujących zaawansowanych rozwiązań i bezpieczeństwa operacyjnego. System obsługuje różne typy tabel, które wykorzystują różne silniki pamięci masowej, takie jak InnoDB, który zapewnia obsługę transakcji, oraz MyISAM, znany z wydajnego indeksowania danych. MySQL oferuje również funkcje replikacji, które są kluczowe dla ciągłości biznesowej i skalowalności systemu, a także partycjonowanie tabel, które ułatwia zarządzanie dużymi zbiorami danych.
  + **MySQL Workbench -** to narzędzie graficzne do zarządzania bazami danych MySQL. Zapewnia użytkownikom interfejs do tworzenia, edytowania, zarządzania i przeglądania struktury baz danych, a także pozwala na pisanie i wykonywanie zapytań SQL. Workbench jest narzędziem wszechstronnym, umożliwiającym tworzenie kopii zapasowych, importowanie danych i administrowanie serwerem. Jest szczególnie przydatny dla programistów i administratorów baz danych, którzy mogą w prosty sposób   
    wizualizować struktury baz danych oraz optymalizować ich wydajność.
  + **Git -** to rozproszony system kontroli wersji zaprojektowany w celu umożliwienia programistom efektywnego zarządzania zmianami w kodzie źródłowym podczas pracy nad projektami. Stworzony przez Linusa Torvaldsa w 2005 roku, Git pozwala   
    na przechowywanie historii zmian lokalnie, umożliwiając szybką pracę nad kodem nawet   
    w trybie offline. Jego główną cechą jest to, że każdy, kto pobierze kopię repozytorium, otrzymuje pełną historię zmian, co oznacza, że każde repozytorium może służyć jako pełna kopia zapasowa. Git jest niezwykle elastyczny w zarządzaniu wieloma gałęziami kodu, umożliwiając eksperymentowanie, rozwój funkcji i naprawianie błędów niezależnie   
    od głównej linii kodu. afsafsfdsafsafdsafsafdfssssssssssssssssssssssssssssssssssssssssaa
  + **GitHub -** to internetowa platforma hostingowa wykorzystująca Git do zarządzania projektami oprogramowania. Uruchomiony w 2008 roku przez Toma Prestona-Wernera, Chrisa Wanstratha i PJ Hyetta, GitHub pozwala programistom i firmom przechowywać, zarządzać i kontrolować ich projekty kodu źródłowego. Jest to również społeczność,   
    w ramach której deweloperzy mogą współpracować nad projektami, udostępniając kod, pracując nad poprawkami i opracowując nowe funkcje. GitHub oferuje różne funkcje, takie jak śledzenie zgłoszeń, pull requesty, zarządzanie projektami i wiki dla każdego repozytorium.
  + **Draw.io -** to narzędzie do tworzenia diagramów online, które oferuje szeroki zakres możliwości wizualizacji danych i procesów. Jest to aplikacja internetowa, która umożliwia użytkownikom łatwe tworzenie diagramów, takich jak diagramy przepływu pracy, schematy blokowe, organigramy, diagramy sieciowe, mapy myśli i wiele innych.

## 2.5. DIAGRAMY

### 2.5.1. DIAGRAM PRZYPADKÓW UŻYCIA

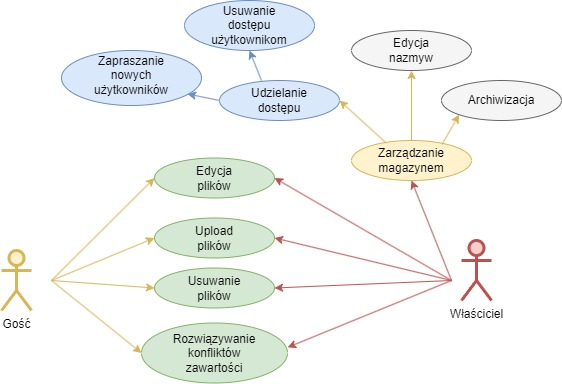
#### 2.5.1.1. DIAGRAM PRZYPADKÓW UŻYCIA - UŻYTKOWNIK



*Diagram przypadków użycia aplikacji sdrive - użytkownik*

Diagram przypadków użycia użytkownika przedstawia kluczowe funkcjonalności, które powinna oferować projektowana aplikacja webowa, oraz relacje pomiędzy użytkownikiem a poszczególnymi modułami systemu. Centralnym elementem tego diagramu jest logowanie, które będzie stanowić punkt wyjścia do korzystania z pozostałych funkcji aplikacji. Proces ten powinien zapewnić użytkownikowi bezpieczny dostęp do systemu po wcześniejszej rejestracji, podczas której użytkownik będzie miał możliwość utworzenia konta. Rejestracja będzie niezbędnym krokiem, aby zidentyfikować użytkownika i przypisać mu dostęp do zasobów aplikacji. Po zalogowaniu użytkownik powinien zyskać dostęp do modułu zarządzania plikami, który będzie podstawowym obszarem funkcjonalnym aplikacji. W ramach tego modułu użytkownik będzie mógł przesyłać pliki do systemu, pobierać je na swoje urządzenie oraz przeglądać ich zawartość. Aplikacja powinna również umożliwić edytowanie plików tekstowych, zmianę ich nazw oraz udostępnianie plików innym użytkownikom. Kolejnym modułem będzie system magazynów, który umożliwi organizację plików. System magazynów powinien oferować funkcje dodawania oraz usuwania plików, a także edycji ich zawartości. Ważnym elementem tej funkcjonalności będzie zarządzanie dostępem do magazynów. Właściciel magazynu będzie mógł udzielać dostępu innym użytkownikom. Ważną częścią projektowanego systemu będzie również moduł zarządzania znajomymi, który będzie wspierał nawiązywanie relacji między użytkownikami aplikacji. Użytkownicy powinni mieć możliwość wyszukiwania innych osób w systemie, zapraszania ich do listy znajomych oraz zarządzania swoimi kontaktami. System powinien pozwalać na prowadzenie konwersacji z innymi użytkownikami, co będzie istotnym elementem komunikacji w kontekście współdzielenia plików i magazynów. Funkcjonalność ta ułatwi nawiązywanie współpracy i usprawni komunikację pomiędzy użytkownikami systemu.

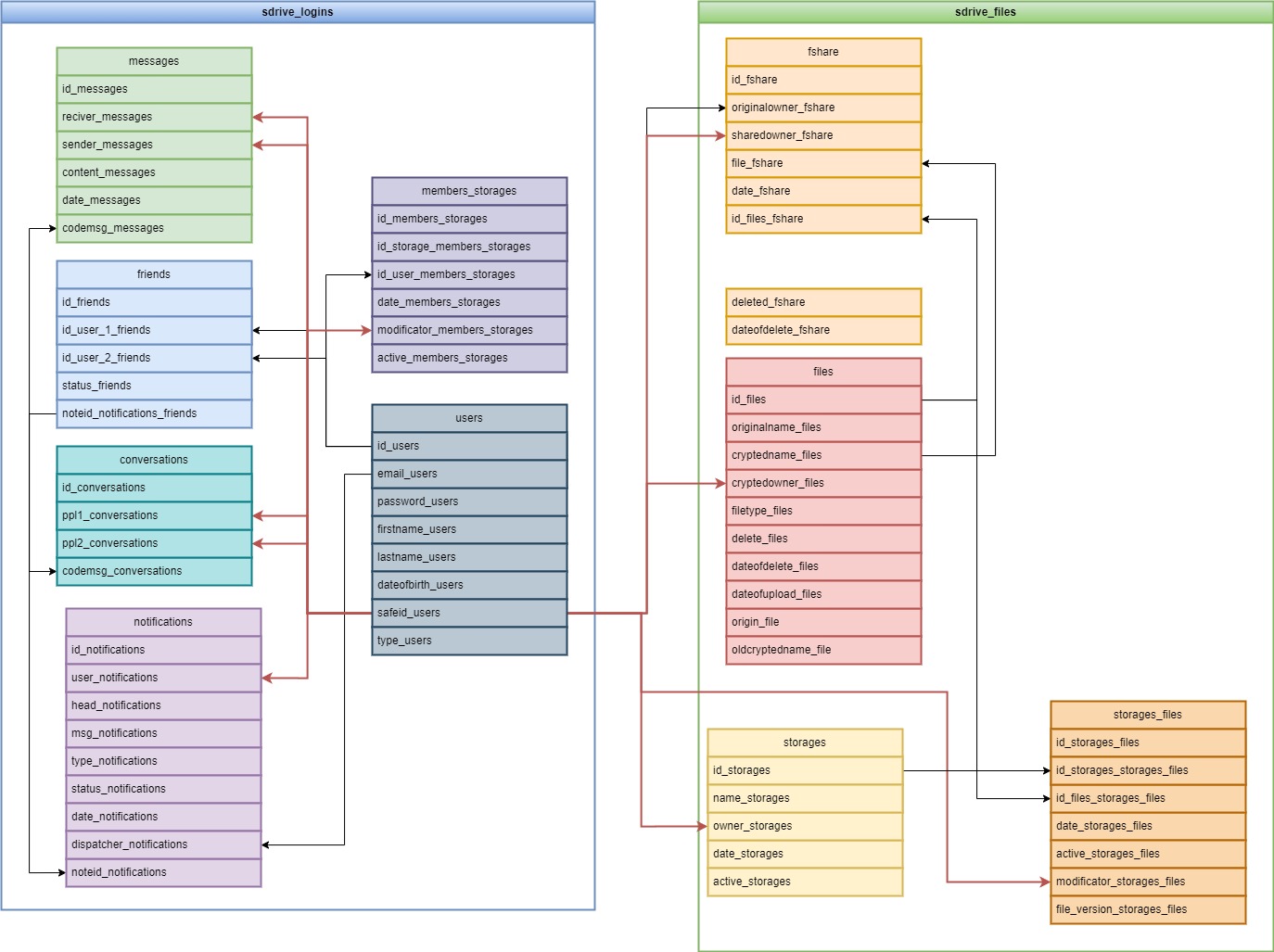
#### 2.5.1.2. DIAGRAM PRZYPADKÓW UŻYCIA - MAGAZYNY



*Diagram przypadków użycia aplikacji sdrive - magazyny*

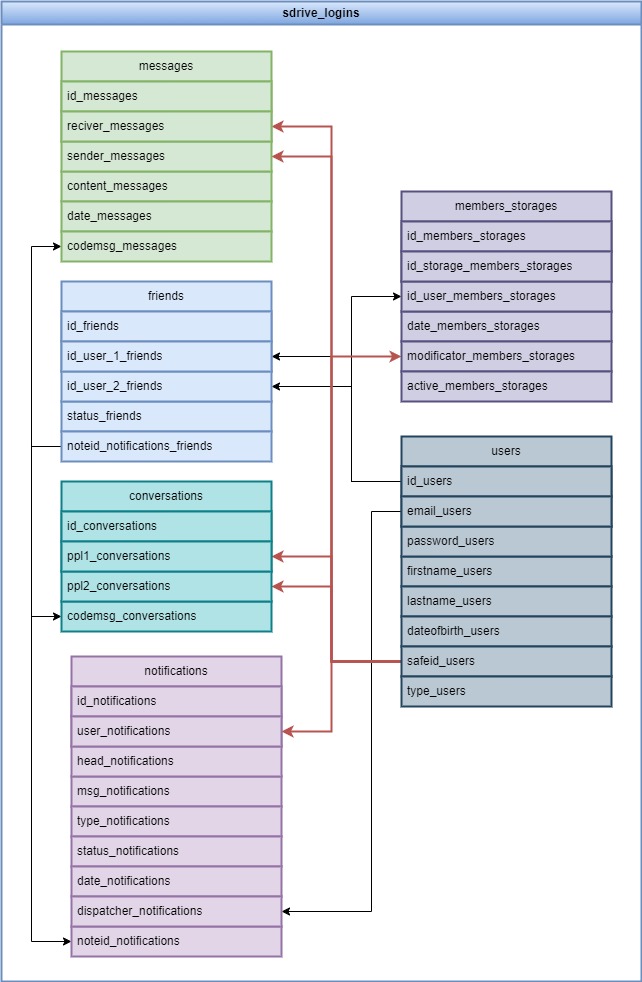
Diagram przypadków użycia magazynów przedstawia funkcjonalności związane z systemem magazynów oraz role użytkowników, którzy mogą z niego korzystać. W systemie wyróżniono dwie główne role: Właściciela magazynu oraz Gościa, z których każda posiada określone uprawnienia i zakres działań. Właściciel magazynu otrzyma pełną kontrolę nad zarządzaniem magazynem i jego zawartością. Podstawową funkcjonalnością dostępną dla właściciela będzie zarządzanie magazynem, które obejmuje takie operacje jak edycja nazwy magazynu oraz archiwizacja, co pozwali na porządkowanie i długoterminowe przechowywanie plików. Dodatkowo właściciel powinien mieć możliwość udzielania dostępu innym użytkownikom oraz usuwania dostępu w razie potrzeby, co da mu pełną kontrolę nad tym, kto może korzystać z magazynu. Po udzieleniu dostępu, użytkownicy posiadający rolę Gościa zyskują uprawnienia do pracy z plikami w magazynie. Goście mogą wykonywać operacje takie jak edycja plików, upload plików (przesyłanie nowych plików do magazynu) oraz usuwanie plików. Kluczową funkcjonalnością będzie także rozwiązywanie konfliktów zawartości, które może wystąpić w sytuacji, gdy kilka osób jednocześnie pracuje nad tym samym plikiem. Mechanizm ten umożliwi porównywanie różnic w wersjach plików i wybór odpowiedniej zawartości, co zapewni spójność danych. Obie role użytkowników – Właściciel i Gość – otrzymają dostęp do operacji związanych z plikami, jednak to Właściciel posiada nadrzędne uprawnienia do zarządzania dostępem, edycji magazynu oraz jego konfiguracji.

### 2.5.2. DIAGRAM ERD



*Diagram ERD aplikacji sdrive*

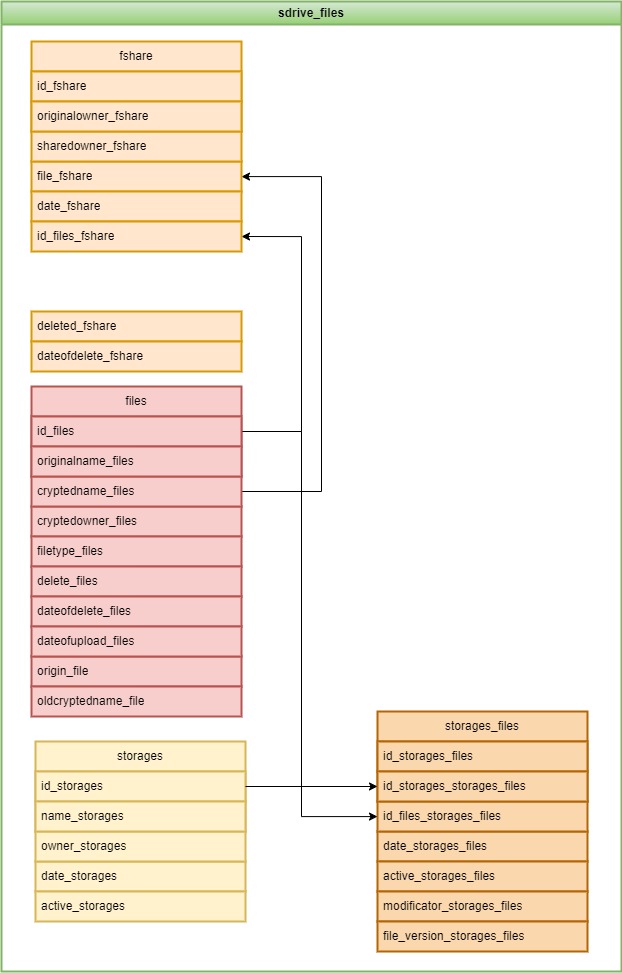
#### 2.5.2.1. DIAGRAM ERD - BAZA SDRIVE\_LOGINS



*Diagram ERD aplikacji sdrive - sdrive\_logins*

1. **messages** – przechowuje informacje o wiadomościach pomiędzy użytkownikami, takie jak:
   * id\_messages - identyfikator wiadomości
   * receiver\_messages - odbiorca
   * sender\_messages - nadawca
   * content\_messages - treść wiadomości
   * date\_messages - data wysłania
   * codemsg\_messages - kod wiadomości
2. **friends** – zawiera dane o relacjach znajomych między użytkownikami:
   * id\_friends - identyfikator znajomego
   * id\_user\_1\_friends i id\_user\_2\_friends - użytkownicy powiązani relacją
   * status\_friends - status znajomości
   * noteid\_notifications\_friends - powiązanie z wiadomościami
3. **conversations** – przechowuje dane o konwersacjach:
   * id\_conversations - identyfikator konwersacji
   * ppl1\_conversations i ppl2\_conversations - uczestnicy rozmowy
   * codemsg\_conversations - kod wiadomości
4. **notifications** – zarządza informacjami o powiadomieniach:
   * id\_notifications - identyfikator wiadomości
   * user\_notifications - użytkownik otrzymujący powiadomienie
   * msg\_notifications - treść powiadomienia
   * type\_notifications - typ powiadomienia
   * status\_notifications - status wiadomości
   * date\_notifications - data
   * dispatcher\_notifications - nadawca
5. **members\_storages** – tabela związana z przechowywaniem członków:
   * id\_members\_storages - identyfikator relacji członka magazynu
   * id\_storage\_members\_storages - identyfikator użutkownika
   * id\_user\_members\_storages - użytkownik związany z magazynem
   * date\_members\_storages - data
   * modificator\_members\_storages - identyfikator edytującego
   * active\_members\_storages - status członka
6. **users** – główna tabela przechowująca dane użytkowników:
   * id\_users - identyfikator użytkownika
   * email\_users - email
   * password\_users - hasło
   * firstname\_users - imię
   * lastname\_users - nazwisko
   * dateofbirth\_users - data urodzenia
   * safeid\_users - specjalny identyfikator użytkownika
   * type\_users - typ użytkownika

#### 2.5.2.2. DIAGRAM ERD - BAZA SDRIVE\_FILES

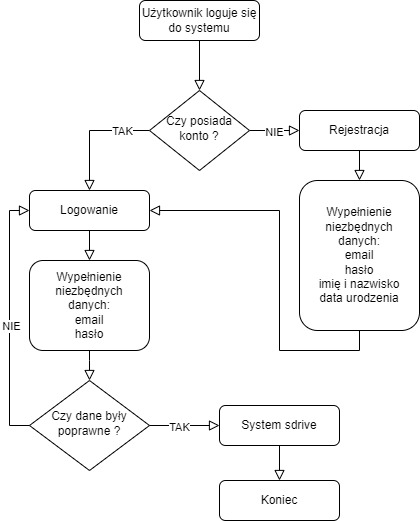


*Diagram ERD aplikacji sdrive - sdrive\_files*

1. **fshare** – przechowuje informacje o udostępnieniach plików:
   * id\_fshare - identyfikator udostępnienia
   * originalowner\_fshare - oryginalny właściciel
   * sharedowner\_fshare - współwłaściciel
   * file\_fshare - zaszyfrowana nazwa pliku
   * date\_fshare - data
   * id\_files\_fshare - identyfikator pliku
   * deleted\_fshare - status pliku
   * datedelete\_fshare - data usunięcia
2. **files** – główna tabela zawierająca informacje o plikach:
   * id\_files - identyfikator pliku
   * originalname\_files - oryginalna nazwa pliku
   * cryptedname\_files - szyfrowana nazwa
   * cryptedowner\_files - szyfrowany użytkownik
   * filetype\_files - typ pliku
   * delete\_files oraz dateofdelete\_files - status usunięcia pliku i data
   * dateofupload\_files - data przesłania
   * origin\_file - pochodzenie pliku
   * oldcryptedname\_file - stara nazwa szyfrowana
3. **storages** – tabela związana z magazynami danych:
   * id\_storages - identyfikator magazynu
   * name\_storages - nazwa magazynu
   * owner\_storages - właściciel
   * date\_storages - data
   * active\_storages - status magazynu
4. **storages\_files** – tabela łącząca pliki z magazynami:
   * id\_storages\_files - identyfikator powiązania pliku z magazynem
   * id\_storages\_storages\_files - powiązanie z tabelą magazynów
   * id\_files\_storages\_files - powiązanie z tabelą plików (identyfikator)
   * date\_storages\_files - data
   * active\_storages\_files - status powiązania
   * modificator\_storages\_files - identyfikator edytującego
   * file\_version\_storages\_files - wersja pliku

### 2.5.3. DIAGRAMY AKTYWNOŚCI

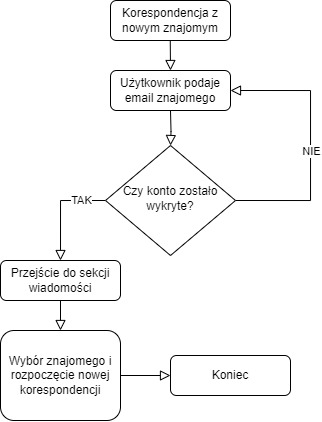
#### 2.5.3.1. DIAGRAM AKTYWNOŚCI - LOGOWANIE



*Diagram aktywności aplikacji sdrive - logowanie*

Diagram przedstawia proces logowania użytkownika do systemu, który rozpoczyna się od próby uzyskania dostępu przez użytkownika. Pierwszym krokiem jest weryfikacja, czy użytkownik posiada już konto w systemie. W przypadku odpowiedzi twierdzącej, użytkownik przechodzi do etapu logowania, gdzie wprowadza niezbędne dane, takie jak adres e-mail oraz hasło. Jeżeli podane dane są poprawne, użytkownik uzyskuje dostęp do systemu sdrive, co kończy proces logowania. Jeśli dane okażą się niepoprawne, system ponownie kieruje użytkownika do etapu logowania, umożliwiając ponowne wprowadzenie informacji. W sytuacji, gdy użytkownik nie posiada konta, przechodzi on przez proces rejestracji, który polega na wypełnieniu formularza rejestracyjnego wymagającego podania takich informacji jak adres e-mail, hasło, imię, nazwisko oraz data urodzenia. Po zakończeniu rejestracji użytkownik może przystąpić do logowania, aby uzyskać dostęp do systemu.

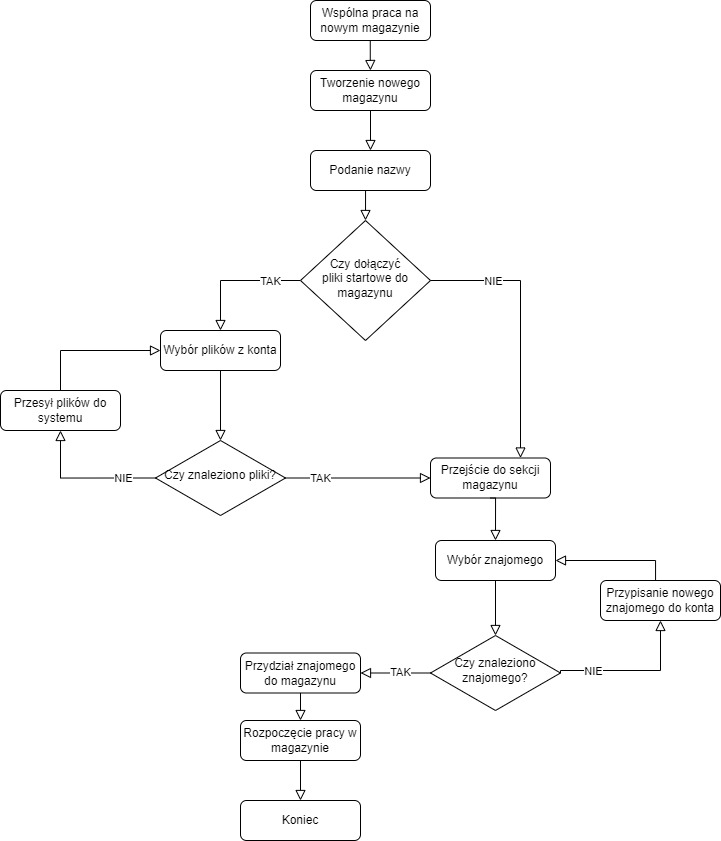
#### 2.5.3.2. DIAGRAM AKTYWNOŚCI - WYSŁANIE WIADOMOŚCI DO NOWEGO ZNAJOMEGO



*Diagramy aktywności aplikacji sdrive - wysłanie wiadomości do nowego znajomego*

Diagram przedstawia proces nawiązania korespondencji z nowym znajomym w systemie. Całość rozpoczyna się od podania przez użytkownika adresu e-mail znajomego. System weryfikuje, czy konto odpowiadające podanemu e-mailowi istnieje. W przypadku, gdy konto zostanie wykryte, użytkownik przechodzi do sekcji wiadomości, gdzie może wybrać znajomego z listy i rozpocząć nową korespondencję. Jeśli system nie znajdzie konta odpowiadającego podanemu adresowi e-mail, proces zostaje przerwany, a użytkownik ma możliwość ponownego podania danych znajomego.

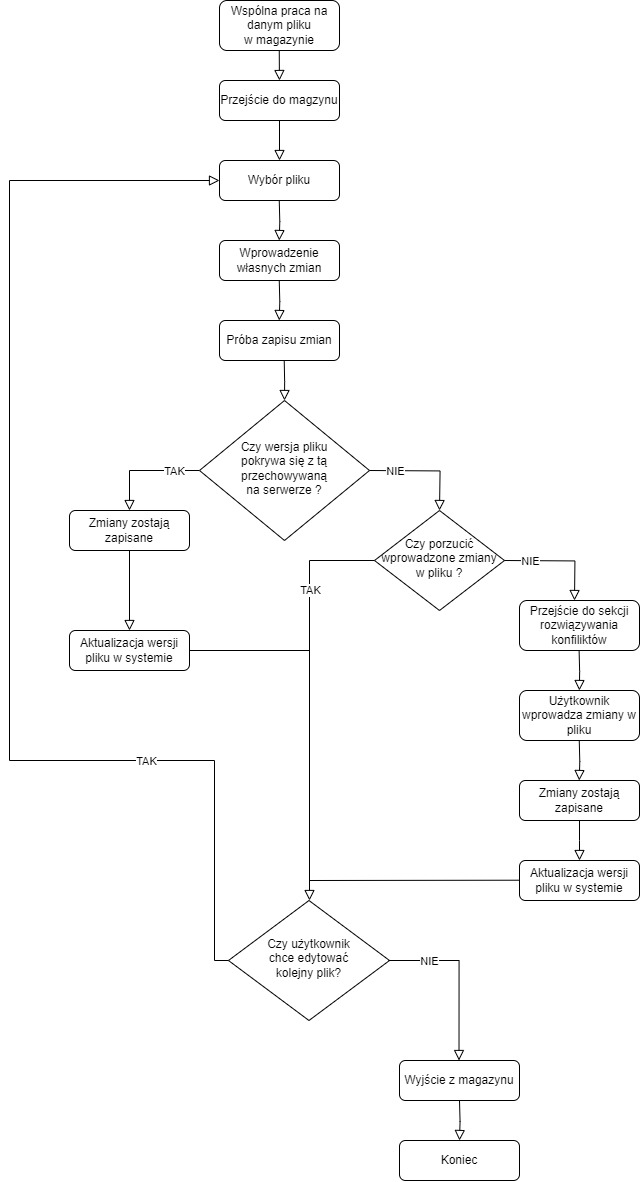
#### 2.5.3.4. DIAGRAM AKTYWNOŚCI - PRACA NA NOWYM MAGAZYNIE



*Diagramy aktywności aplikacji sdrive - praca na nowym magazynie*

Diagram przedstawia proces tworzenia nowego magazynu w kontekście współpracy z innymi użytkownikami. Całość rozpoczyna się od podania nazwy nowego magazynu. Następnie użytkownik decyduje, czy chce dołączyć pliki startowe do magazynu. Jeśli odpowiedź jest twierdząca, system przechodzi do etapu wyboru plików z konta użytkownika, po czym następuje ich przesył do systemu. W przypadku braku plików użytkownik ma możliwość ponownej próby wyboru. Jeśli użytkownik rezygnuje z dodania plików, system przechodzi do sekcji magazynu, gdzie można przypisać znajomego do współpracy. W dalszym etapie użytkownik wybiera znajomego. System weryfikuje, czy dany znajomy został znaleziony w bazie danych. Jeśli znajomy istnieje, zostaje on przypisany do magazynu, co umożliwia rozpoczęcie wspólnej pracy. W przeciwnym wypadku użytkownik może przypisać nowego znajomego do swojego konta. Proces kończy się, gdy znajomy zostanie przydzielony, a praca w magazynie może się rozpocząć. Diagram w sposób przejrzysty ukazuje zarówno proces organizacji zasobów, jak i integrację funkcji współpracy między użytkownikami systemu.

#### 2.5.3.5. DIAGRAM AKTYWNOŚCI - PRACA NA WSPÓLNYM PLIKU W MAGAZYNIE



*Diagramy aktywności aplikacji sdrive - praca na wspólnym pliku w magazynie*

Diagram przedstawia proces wspólnej pracy nad plikiem w magazynie, uwzględniający zarówno edycję, jak i rozwiązywanie potencjalnych konfliktów. Proces rozpoczyna się od przejścia użytkownika do magazynu, gdzie dokonuje on wyboru pliku i wprowadza własne zmiany. W momencie próby zapisu system sprawdza, czy wersja pliku pokrywa się z aktualną wersją na serwerze. Jeśli wersje są zgodne, zmiany zostają zapisane, a system dokonuje aktualizacji wersji pliku. W przypadku wykrycia niezgodności użytkownik podejmuje decyzję, czy porzucić wprowadzone zmiany. Jeśli decyduje się na ich odrzucenie, wraca do wcześniejszego etapu. Jeśli natomiast wybierze kontynuację pracy, system przenosi go do sekcji rozwiązywania konfliktów. Tam użytkownik może wprowadzić odpowiednie modyfikacje, które następnie zostają zapisane i zaktualizowane w systemie. Po zakończeniu operacji użytkownik ma możliwość wyboru kolejnego pliku do edycji. Jeśli zdecyduje się na kontynuację, proces powtarza się od etapu wyboru pliku. W przeciwnym przypadku użytkownik wychodzi z magazynu, kończąc pracę.

## 2.6. SCENARIUSZE UŻYCIA

### 2.6.1. SCENARIUSZ UŻYCIA - LOGOWANIE

|  |  |
| --- | --- |
| **Element** | **Opis** |
| **Nazwa scenariusza** | Logowanie do systemu |
| **Nr. Scenariusza** | 001 |
| **Aktorzy** | Użytkownik, system |
| **Warunek początkowy** | Użytkownik posiada konto w systemie sdrive. |
| **Przebieg główny** | Użytkownik podaje email i hasło. |
| **Alternatywny przebieg** | Hasło lub email zostaje odrzucone, podjęcie ponownej próby. |
| **Wynik** | Użytkownik zostaje zalogowany do systemu sdrive. |

### 2.6.2. SCENARIUSZ UŻYCIA - REJESTRACJA

|  |  |
| --- | --- |
| **Element** | **Opis** |
| **Nazwa scenariusza** | Rrejestracja w systemie sdrive |
| **Nr. Scenariusza** | 002 |
| **Aktorzy** | Użytkownik, system |
| **Warunek początkowy** | Brak --- |
| **Przebieg główny** | Użytkownik przechodzi do sekcji rejestracji, podaje email, imię i nazwisko oraz datę urodzenia. |
| **Alternatywny przebieg** | Wprowadzono błędne dane, należy ponowić próbę rejestracji. |
| **Wynik** | Użytkownik tworzy konto w systemie sdrive. |

#### 

### 2.6.3. SCENARIUSZ UŻYCIA - PRZESYŁ PLIKÓW DO SYSTEMU

|  |  |
| --- | --- |
| **Element** | **Opis** |
| **Nazwa scenariusza** | Przesył plików do systemu sdrive |
| **Nr. Scenariusza** | 003 |
| **Aktorzy** | Użytkownik, system |
| **Warunek początkowy** | Użytkownik ma konto w systemie i jest zalogowany. |
| **Przebieg główny** | Użytkownik w sekcji przesyłu plików wybiera plik z komputera i wysyła go do swojego konta. |
| **Alternatywny przebieg** | Nie udało się przesłać pliku, podjęcie ponownej próby. |
| **Wynik** | Udało się użytkownikowi przesłać plik z komputera do systemu. |

### 2.6.4. SCENARIUSZ UŻYCIA - UDOSTĘPNIANIE PLIKÓW W SYSTEMIE

|  |  |
| --- | --- |
| **Element** | **Opis** |
| **Nazwa scenariusza** | Udostępnianie plików w systemie |
| **Nr. Scenariusza** | 004 |
| **Aktorzy** | Użytkownik 1, użytkownik 2, system |
| **Warunek początkowy** | Użytkownicy 1 i 2 mają konto w systemie i są znajomymi, użytkownik 1 musi być zalogowany. |
| **Przebieg główny** | Użytkownik 1 z listy swoich plików lub udostępnianych mu plików wybiera plik i opcję „udostępnij”, następnie z listy znajomych wybiera użytkownika 2 i klika przycisk udostepnij. |
| **Alternatywny przebieg** | Nie udało się udostępnić pliku, podjęcie ponownej próby. |
| **Wynik** | Użytkownik 1 udostępnia plik użytkownikowi 2 |

### 2.6.5. SCENARIUSZ UŻYCIA - ZAPRASZANIE NOWYCH ZNAJOMYCH

|  |  |
| --- | --- |
| **Element** | **Opis** |
| **Nazwa scenariusza** | Zapraszanie nowych znajomych |
| **Nr. Scenariusza** | 005 |
| **Aktorzy** | Użytkownik 1, użytkownik 2, system |
| **Warunek początkowy** | Użytkownicy 1 i 2 mają konto w systemie, użytkownik 1 musi być zalogowany. |
| **Przebieg główny** | Użytkownik 1 w sekcji znajomych wpisuje email użytkownika 2, wyświetla się okno ze szczegółowymi danymi użytkownika 2, użytkownik 1 wybiera opcję „zaproś do znajomych”. |
| **Alternatywny przebieg** | Nie udało się zaprosić użytkownika 2 do znajomych (najprawdopodobniej podano zły adres email), podjęcie ponownej próby. |
| **Wynik** | Użytkownik 2 i użytkownik 1 zostają znajomymi. |

### 2.6.6. SCENARIUSZ UŻYCIA - KONWERSACJA ZE ZNAJOMYM

|  |  |
| --- | --- |
| **Element** | **Opis** |
| **Nazwa scenariusza** | Konwersacja ze znajomym |
| **Nr. Scenariusza** | 006 |
| **Aktorzy** | Użytkownik 1, użytkownik 2, system |
| **Warunek początkowy** | Użytkownicy 1 i 2 mają konto w systemie i są znajomymi, użytkownik 1 musi być zalogowany. |
| **Przebieg główny** | Użytkownik 1 po zalogowaniu przechodzi do sekcji wiadomości, wybiera opcję rozpoczęcia nowej konwersacji, wybiera z listy znajomych użytkownika 2, wpisuje treść wiadomości i wybiera opcję „wyślij wiadomość”. |
| **Alternatywny przebieg** | Nie udało się przesłać wiadomości, podjęcie ponownej próby. |
| **Wynik** | Użytkownik 2 i użytkownik 1 wymieniają między sobą korespondencję. |

### 2.6.7. SCENARIUSZ UŻYCIA - TWORZENIE MAGAZYNU

|  |  |
| --- | --- |
| **Element** | **Opis** |
| **Nazwa scenariusza** | Tworzenie magazynu |
| **Nr. Scenariusza** | 007 |
| **Aktorzy** | Użytkownik 1, system |
| **Warunek początkowy** | Użytkownicy ma konto w systemie, użytkownik 1 musi być zalogowany, użytkownik 1 musi mieć przesłany przynajmniej jeden plik do systemu sdrive. |
| **Przebieg główny** | Użytkownik 1 po zalogowaniu wybiera opcję „utwórz” w sekcji magazynów, podaje nazwę magazynu, wybiera pliki które mają być załączone do danego magazynu i na koniec klika „Stwórz magazyn”. |
| **Alternatywny przebieg** | Nie udało się stworzenie magazynu, podjęcie ponownej próby. |
| **Wynik** | Użytkownik 1 tworzy magazyn |

### 2.6.8. SCENARIUSZ UŻYCIA - EDYCJA PLIKU W MAGAZYNIE

|  |  |
| --- | --- |
| **Element** | **Opis** |
| **Nazwa scenariusza** | Edycja pliku w magazynie |
| **Nr. Scenariusza** | 008 |
| **Aktorzy** | Użytkownik 1, użytkownik 2, system |
| **Warunek początkowy** | Użytkownicy 1 i 2 mają konto w systemie i są znajomymi, użytkownik 1 musi być zalogowany, użytkownik 1 musi być przypisany przynajmniej do jednego magazynu. |
| **Przebieg główny** | Użytkownik 1 przechodzi do sekcji magazynów, wybiera konkretny magazyn, w magazynie wybiera plik, dokonuje zmian, zapisuje zmiany. |
| **Alternatywny przebieg** | Wykryto rozbieżność wersji pliku podczas zapisu (najprawdopodobniej użytkownik 2 dokonał zmian i zapisał je w pliku zanim użytkownik 1 zdołał zapisać swoje), użytkowink wybiera opcję rozwiązania konfliktu, przechodzi do sekcji konfliktów, dokonuje odpowiednich zmian w pliku i go zapisuje. |
| **Wynik** | Użytkownik 1 edytuje plik w magazynie |

### 2.6.9. SCENARIUSZ UŻYCIA - PRZYPISYWANIE ZNAJOMYCH DO MAGAZYNU

|  |  |
| --- | --- |
| **Element** | **Opis** |
| **Nazwa scenariusza** | Przypisanie znajomego do magazynu |
| **Nr. Scenariusza** | 009 |
| **Aktorzy** | Użytkownik 1, użytkownik 2, system |
| **Warunek początkowy** | Użytkownicy 1 i 2 mają konto w systemie i są znajomymi, użytkownik 1 musi być zalogowany, użytkownik 1 musi być właścicielem przynajmniej jednego magazynu. |
| **Przebieg główny** | Użytkownik 1 po zalogowaniu przechodzi do swojego magazynu, wybiera opcję udostępniania, z listy znajomych wybiera użytkownika 2 i klika opcję „Udostępnij” |
| **Alternatywny przebieg** | Nie udało się udostępnić dostępu do magazynu, podjęcie ponownej próby. |
| **Wynik** | Użytkownik 1 udziela dostępu do swojego magazynu |

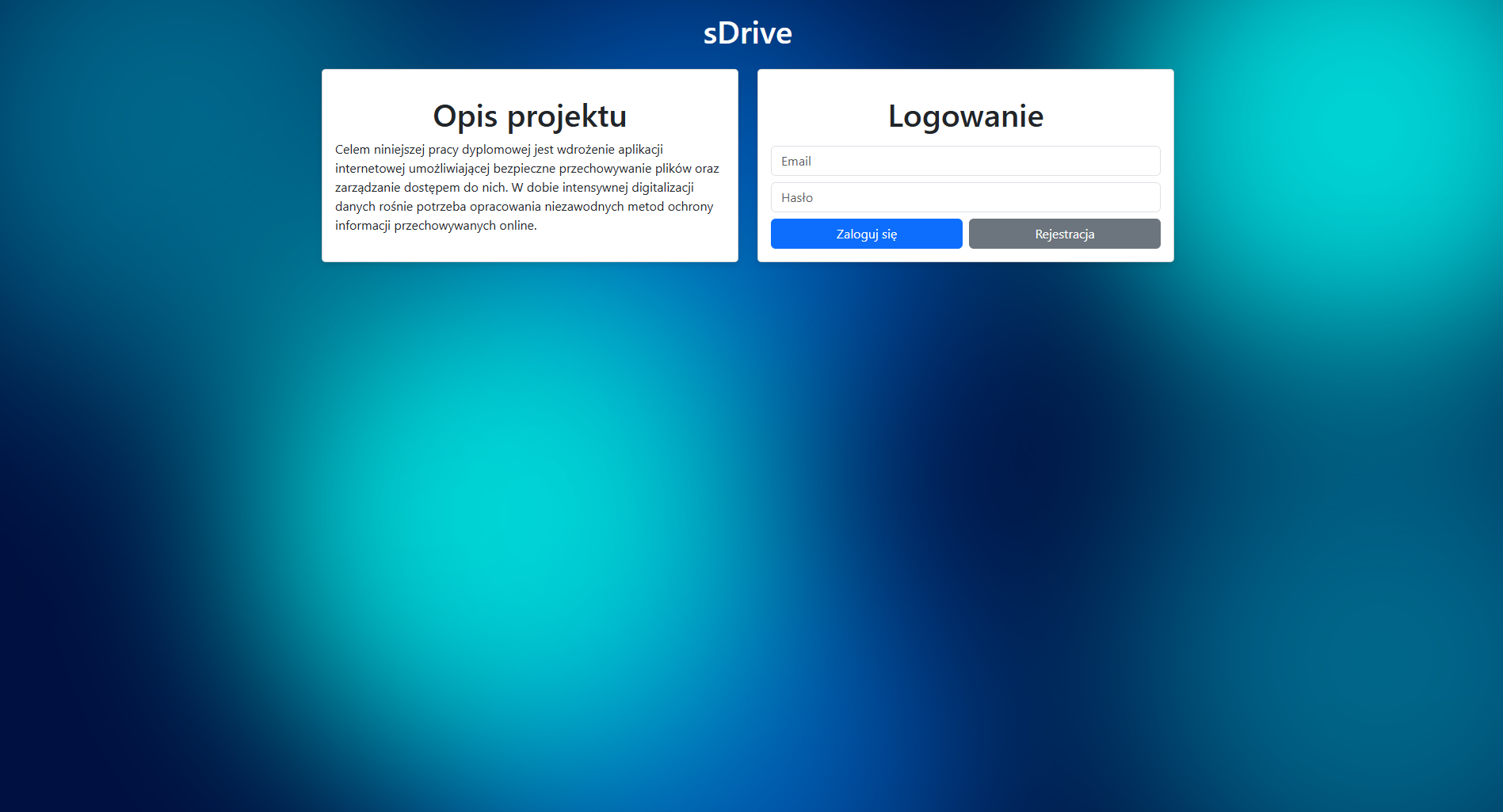
### 2.6.10. SCENARIUSZ UŻYCIA - PRZYPISYWANIE NOWYCH PLIKÓW DO MAGAZYNU

|  |  |
| --- | --- |
| **Element** | **Opis** |
| **Nazwa scenariusza** | Przypisywanie nowych plików do magazynu |
| **Nr. Scenariusza** | 010 |
| **Aktorzy** | Użytkownik 1, użytkownik 2, system |
| **Warunek początkowy** | Użytkownicy 1 ma konto w systemie, użytkownik 1 musi być zalogowany, użytkownik 1 musi być przypisany przynajmniej do jednego magazynu. |
| **Przebieg główny** | Użytkownik 1 po zalogowaniu się przechodzi do magazynu, wybiera opcję dodania pliku, z listy wybiera plik i klika opcję „Dodaj do magazynu”. |
| **Alternatywny przebieg** | Nie udało się dodać pliku do magazynu, podjęcie ponownej próby. |
| **Wynik** | Użytkownik 1 dodaje plik do magazynu. |

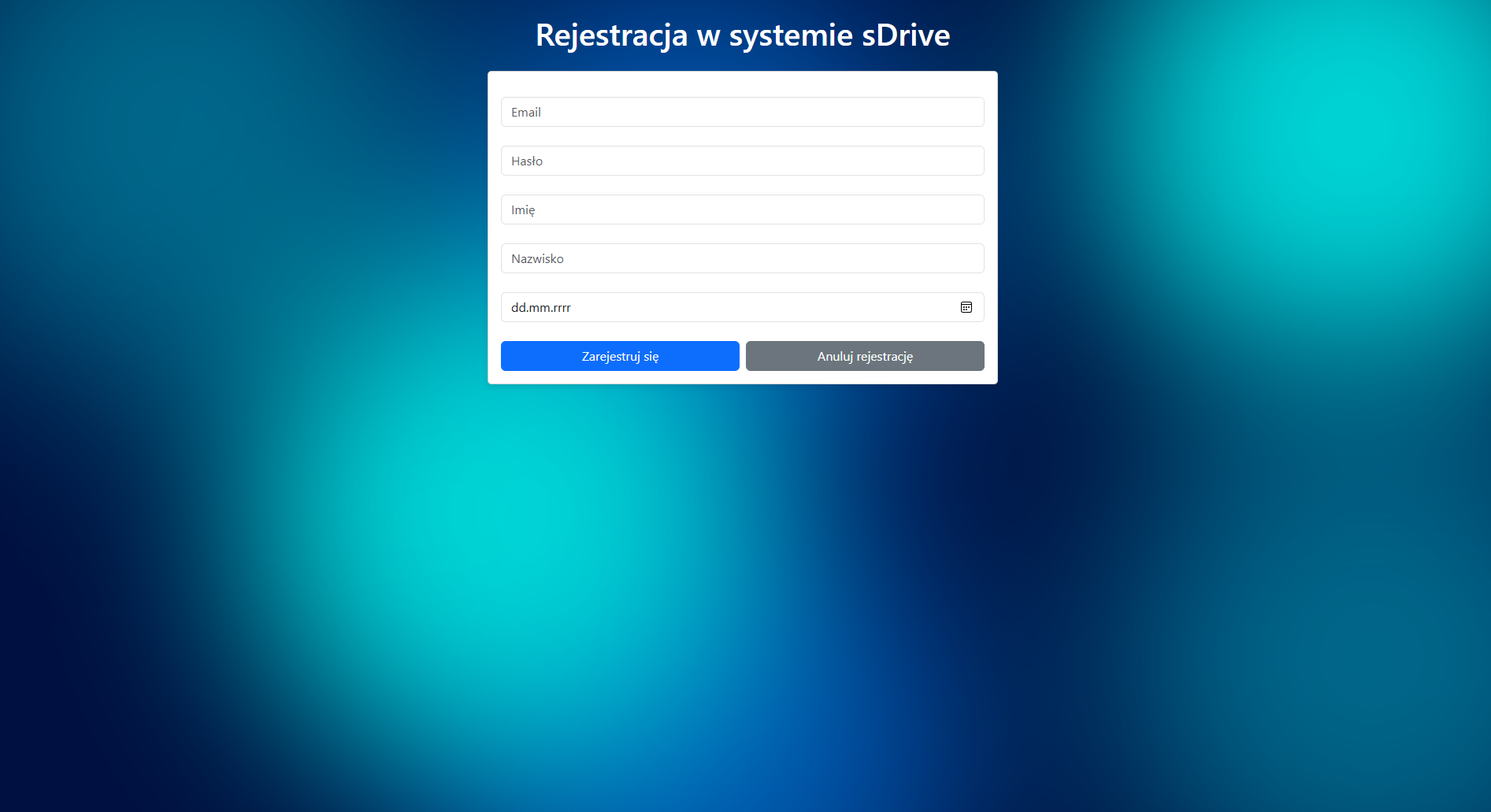
# 3. IMPLEMENTACJA

## 3.1. IMPLEMENTACJA - SYSTEM LOGOWANIA I REJESTRACJI

### 3.1.1. SYSTEM LOGOWANIA I REJESTRACJI - OBRAZY

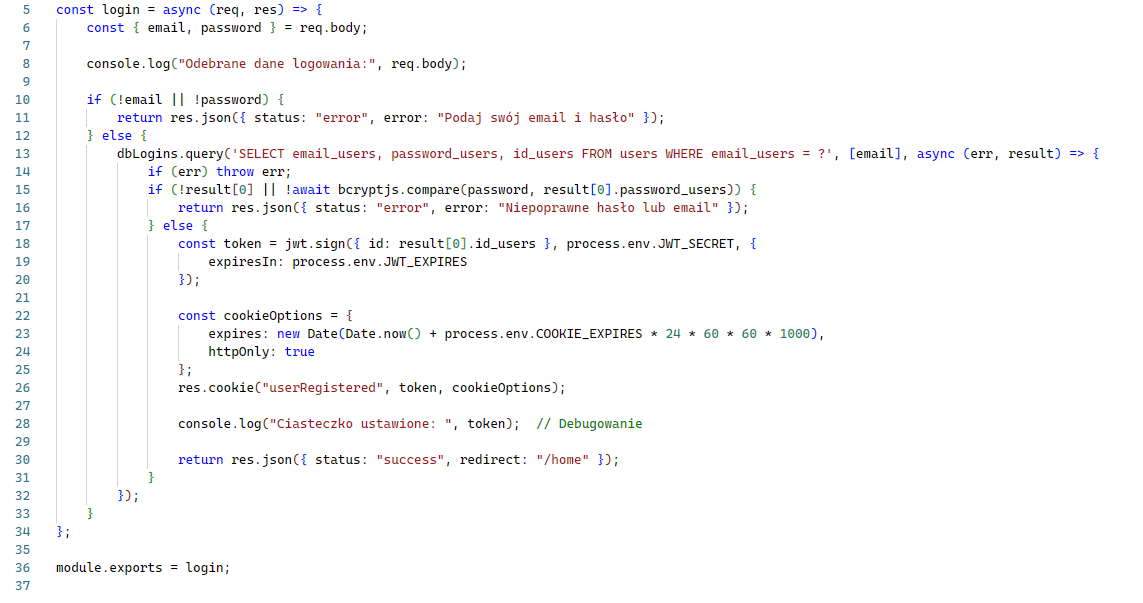


*Implementacja - system logowania i rejestracji - ekran logowania*

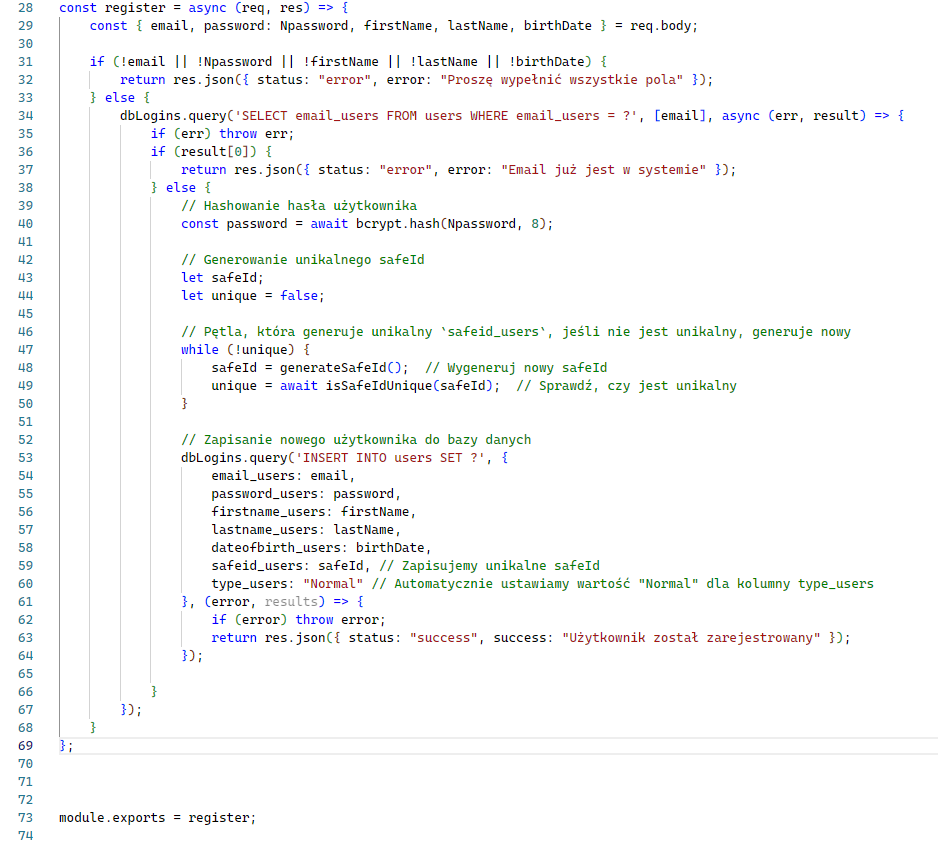


*Implementacja - system logowania i rejestracji - ekran rejestracji*

### 3.1.1. SYSTEM LOGOWANIA I REJESTRACJI - LISTINGI



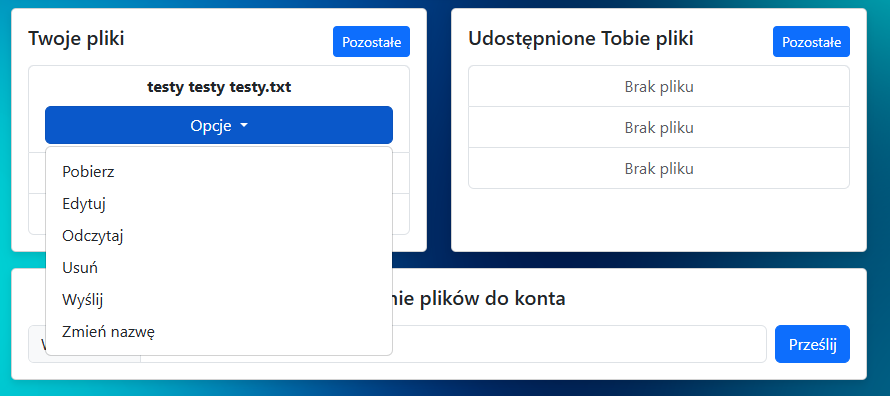
Listing przedstawia funkcję odpowiedzialną za obsługę logowania użytkownika w aplikacji webowej. Funkcja asynchroniczna login przyjmuje żądanie z danymi logowania (e-mail i hasło) przesyłanymi przez użytkownika i weryfikuje je w bazie danych. Najpierw sprawdza, czy dane zostały podane, a następnie wykonuje zapytanie SQL, aby pobrać odpowiadający użytkownika rekord z bazy danych. W przypadku poprawnych danych funkcja porównuje hasło z zahaszowanym hasłem w bazie za pomocą biblioteki bcryptjs. Jeśli logowanie się powiedzie, generowany jest token JWT z identyfikatorem użytkownika, który zostaje zapisany w ciasteczku z ustawionymi parametrami, takimi jak czas wygaśnięcia i dostępność jedynie po stronie serwera (httpOnly). Ostatecznie funkcja zwraca odpowiedź o sukcesie wraz z informacją o przekierowaniu na stronę domową, a w przypadku błędnych danych zwracany jest odpowiedni komunikat o błędzie.



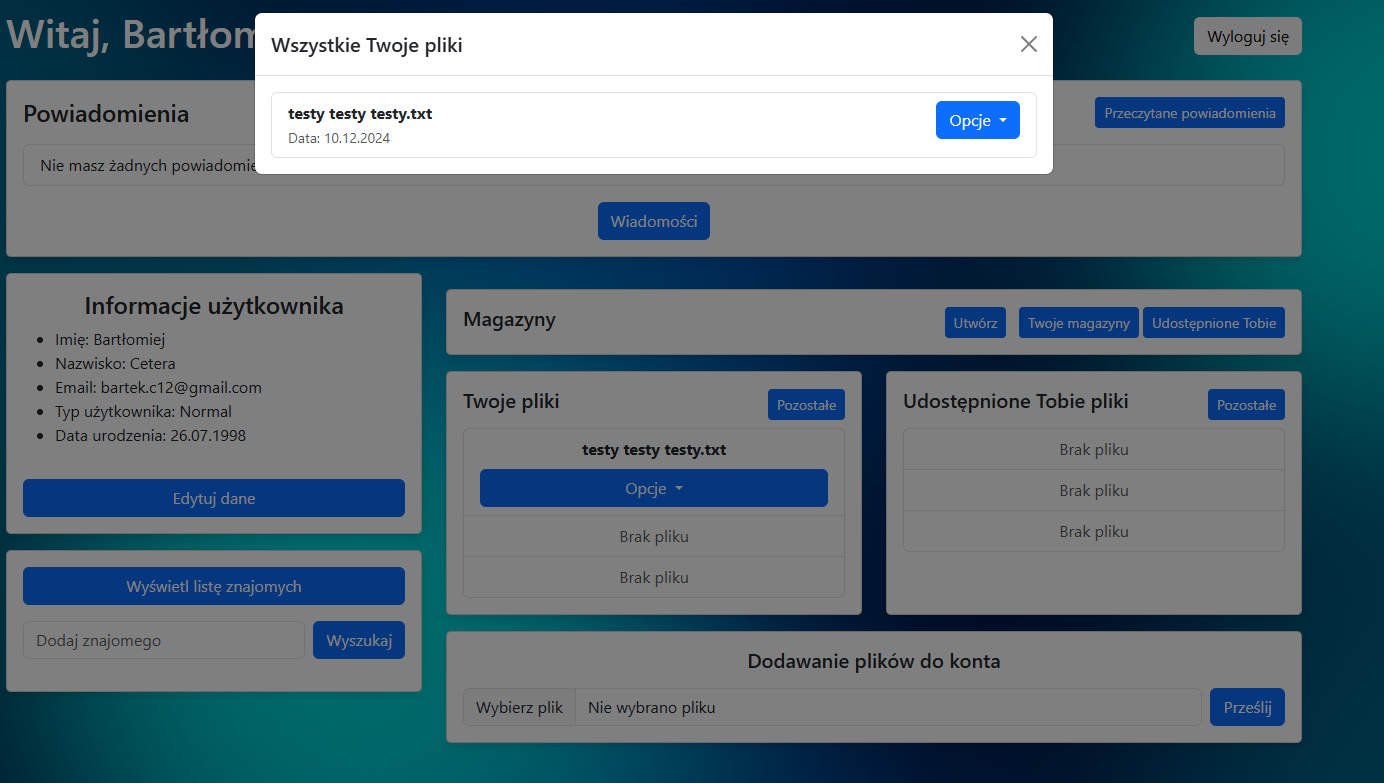
Funkcja register odpowiada za rejestrację nowego użytkownika w systemie, zapewniając walidację oraz bezpieczne przechowywanie danych. Najpierw sprawdzane jest, czy wszystkie wymagane pola zostały wypełnione (e-mail, hasło, imię, nazwisko, data urodzenia), a następnie system weryfikuje, czy podany adres e-mail już istnieje w bazie danych. Jeśli e-mail nie występuje, hasło użytkownika jest hashowane za pomocą biblioteki bcrypt, co zwiększa bezpieczeństwo przechowywanych danych. Dodatkowo generowany jest unikalny identyfikator safeId, który jest sprawdzany pod kątem unikalności za pomocą pętli, aby upewnić się, że nie występuje on już w systemie. W końcu nowy użytkownik zostaje zapisany do bazy danych z odpowiednimi danymi, a jego typ konta domyślnie ustawiany jest na „Normal”. Funkcja zwraca komunikat o sukcesie rejestracji lub odpowiedni błąd, jeśli wystąpią nieprawidłowości, takie jak duplikacja e-maila lub błąd serwera.

## 3.2. IMPLEMENTACJA - SYSTEM ZARZĄDZANIA PLIKAMI

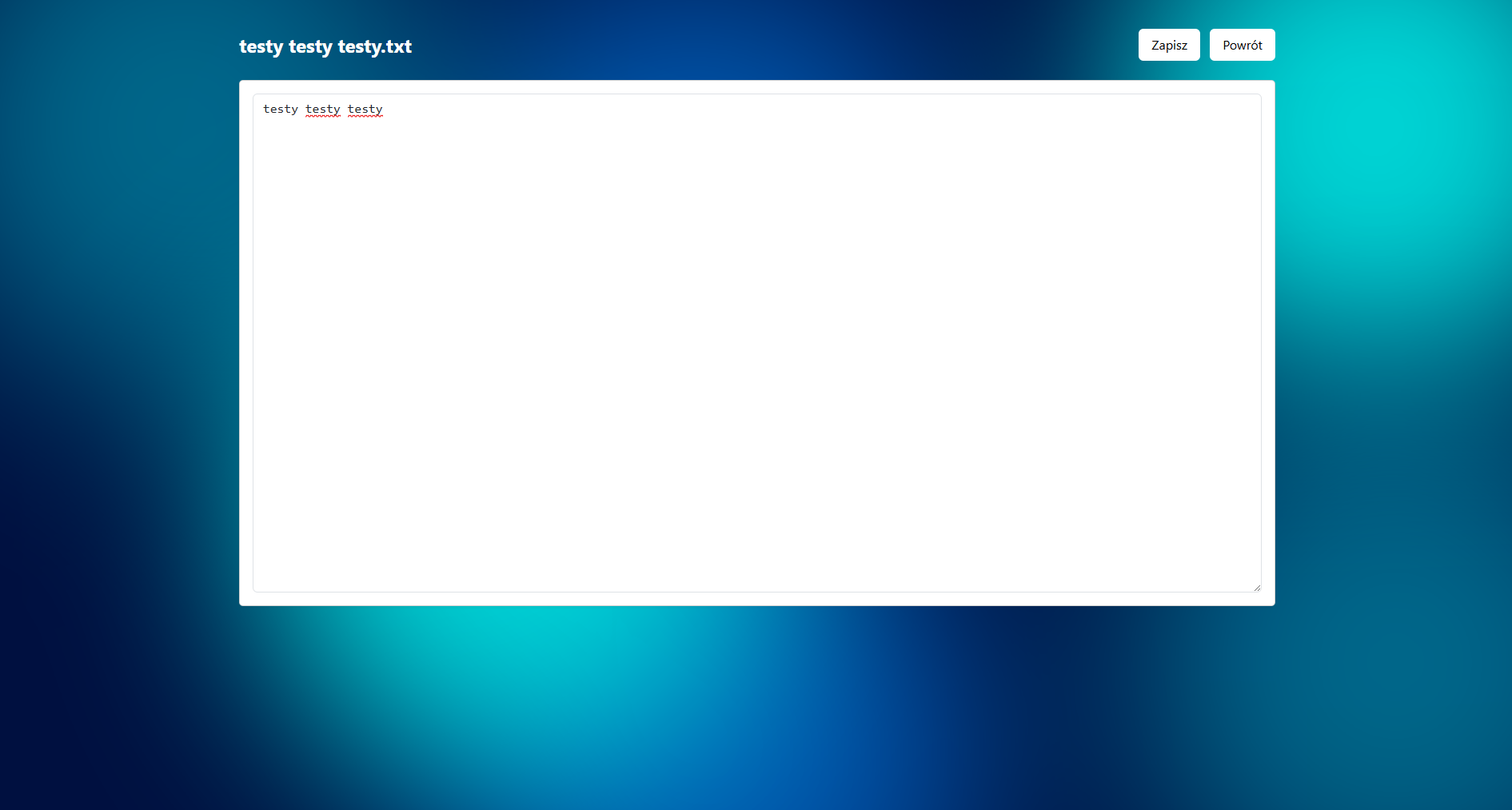
### 3.2.1. SYSTEM ZARZĄDZANIA PLIKAMI - OBRAZY



*Implementacja - system zarządzania plikami - sekcja plików*

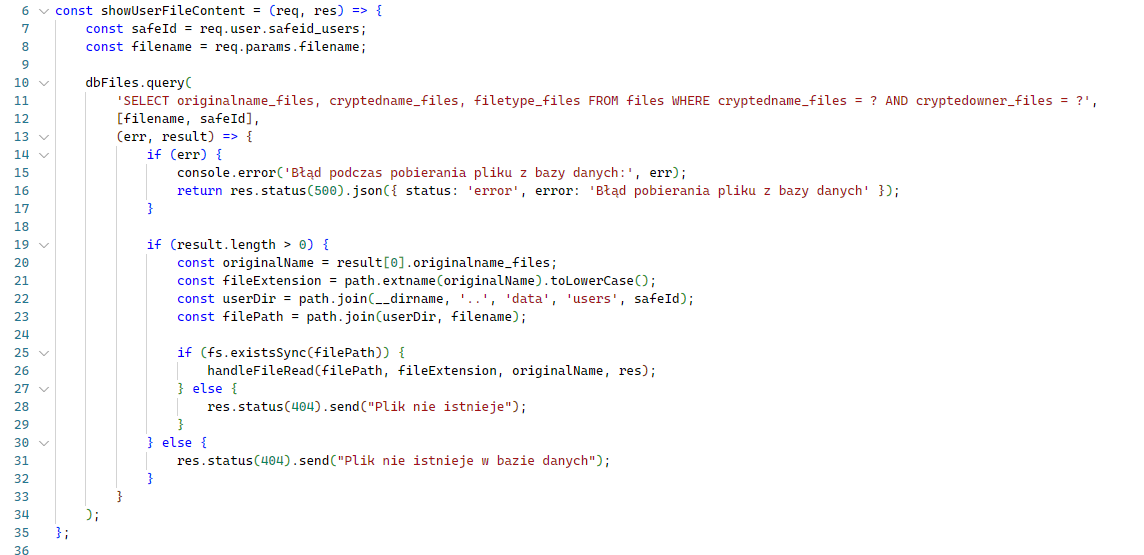


*Implementacja - system zarządzania plikami - sekcja wszystkich plików*

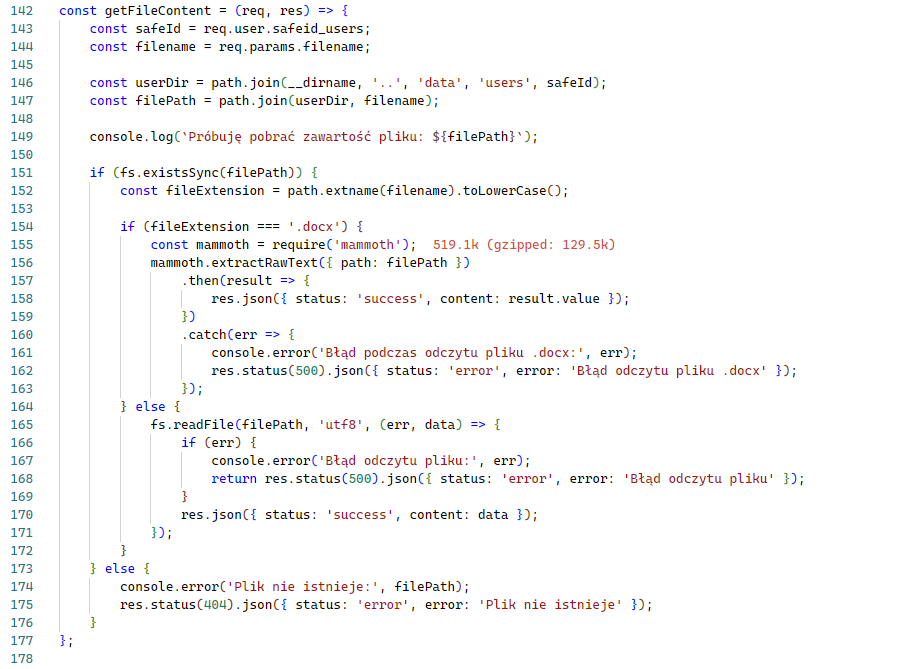


*Implementacja - system zarządzania plikami - edycja pliku*

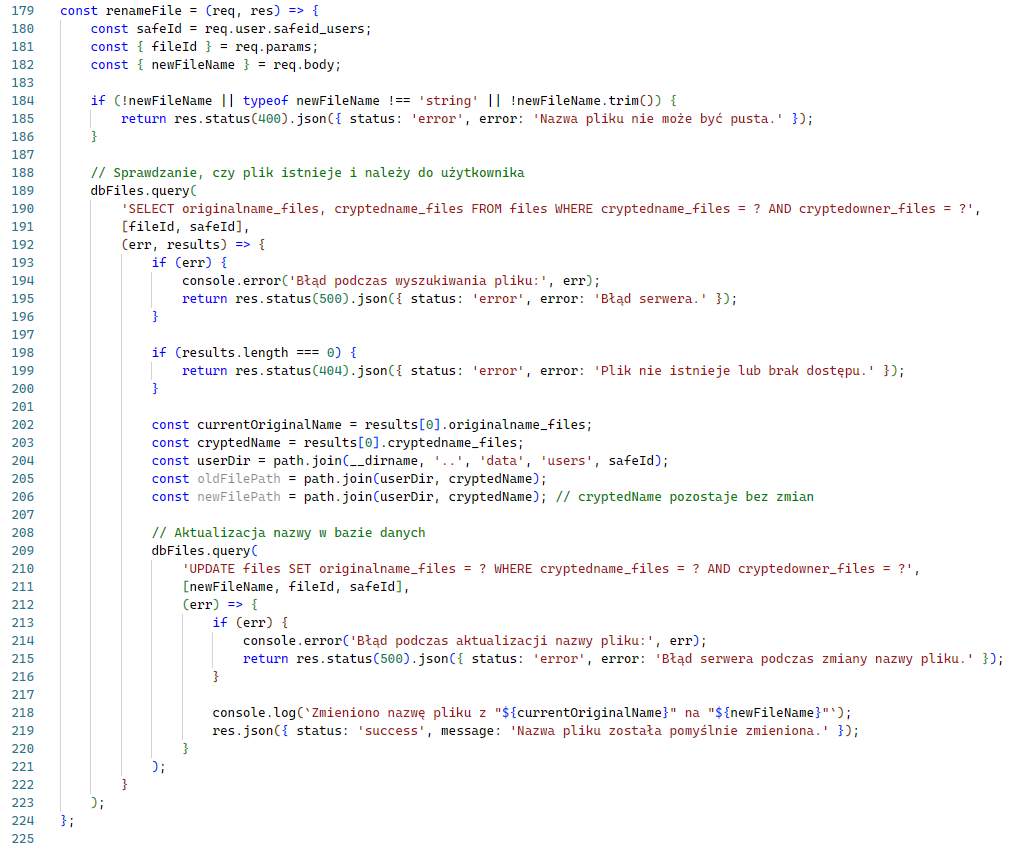
### 3.2.1. SYSTEM ZARZĄDZANIA PLIKAMI - LISTINGI



Funkcja showUserFileContent odpowiada za wyświetlanie zawartości pliku użytkownika na podstawie jego unikalnego identyfikatora safeId oraz nazwy pliku przesłanej w parametrach żądania. Najpierw wykonuje zapytanie do bazy danych, aby sprawdzić, czy plik o podanej zaszyfrowanej nazwie (cryptedname\_files) należy do użytkownika. Jeśli plik istnieje, funkcja pobiera jego oryginalną nazwę, określa rozszerzenie pliku oraz buduje ścieżkę do pliku w katalogu użytkownika. Następnie sprawdza, czy plik fizycznie istnieje na serwerze, i jeśli tak, wywołuje funkcję handleFileRead, która odpowiednio odczytuje plik i przesyła jego zawartość do klienta. W przypadku braku pliku w systemie plików lub bazie danych funkcja zwraca odpowiedź o błędzie z kodem 404. Funkcja zawiera także obsługę błędów serwera oraz informuje użytkownika o problemach z dostępem do pliku lub jego brakiem.



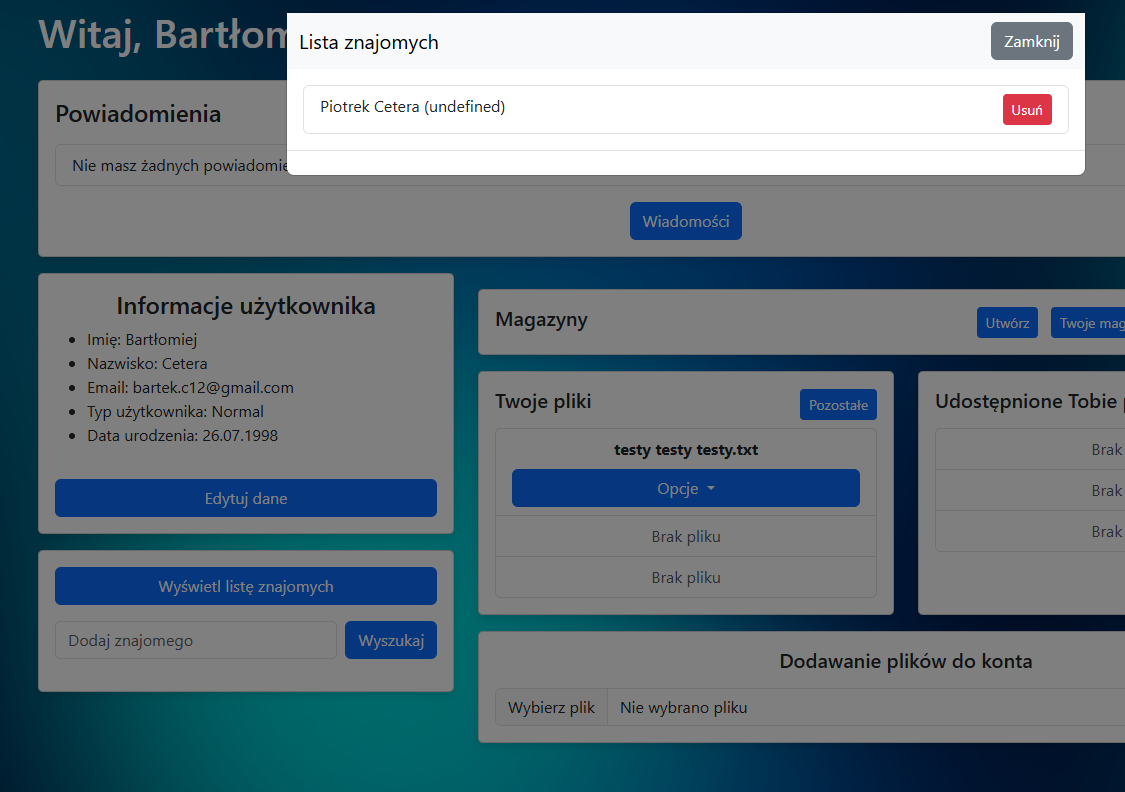
Funkcja getFileContent jest odpowiedzialna za pobieranie zawartości pliku użytkownika z serwera na podstawie jego unikalnego identyfikatora safeId oraz nazwy pliku przekazanej jako parametr żądania. Na początku generowana jest ścieżka do katalogu użytkownika oraz docelowego pliku, po czym system sprawdza, czy plik istnieje na serwerze. Jeśli plik ma rozszerzenie .docx, jego zawartość jest odczytywana przy użyciu biblioteki mammoth, która przekształca dokument Worda na tekst w formacie surowym. Dla plików o innych rozszerzeniach funkcja korzysta z metody fs.readFile do odczytania zawartości pliku jako tekstu w formacie UTF-8. W przypadku powodzenia zawartość pliku zostaje przesłana do klienta w formacie JSON z odpowiednim statusem. Jeśli plik nie istnieje lub wystąpią błędy podczas odczytu, system zwraca komunikaty błędów odpowiednio z kodami 404 lub 500, informując o problemie z dostępem lub odczytem pliku.



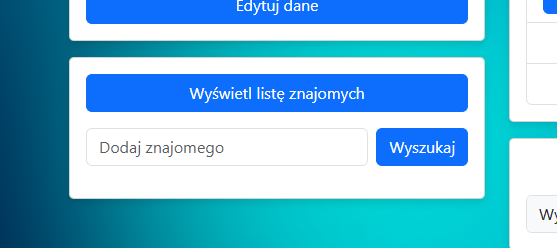
Funkcja renameFile odpowiada za zmianę nazwy pliku należącego do konkretnego użytkownika. Proces rozpoczyna się od walidacji nowej nazwy pliku, aby upewnić się, że jest ona poprawnym, niepustym ciągiem znaków. Następnie funkcja wykonuje zapytanie do bazy danych w celu sprawdzenia, czy plik o podanym identyfikatorze (cryptedname\_files) istnieje i należy do użytkownika na podstawie jego unikalnego identyfikatora safeId. Jeżeli plik zostanie odnaleziony, system aktualizuje jego oryginalną nazwę (originalname\_files) w bazie danych, pozostawiając zaszyfrowaną nazwę pliku bez zmian. Po pomyślnym zakończeniu operacji funkcja zwraca odpowiedź potwierdzającą zmianę nazwy pliku. W przypadku wystąpienia błędów, takich jak brak pliku, brak dostępu lub problem z serwerem, funkcja zwraca odpowiedni komunikat błędu z właściwym kodem HTTP.

## 3.3. IMPLEMENTACJA - SYSTEM ZNAJOMYCH

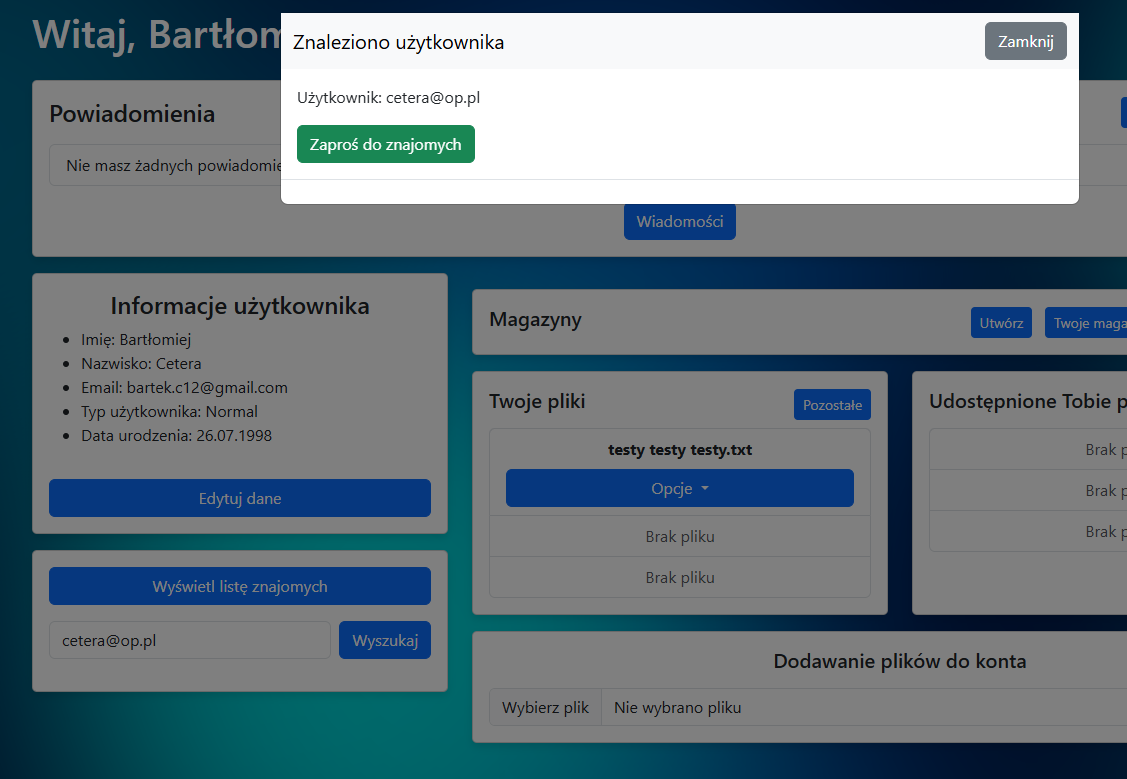
### 3.3.1. SYSTEM ZNAJOMYCH - OBRAZY



*Implementacja - system znajomych - lista znajomych*

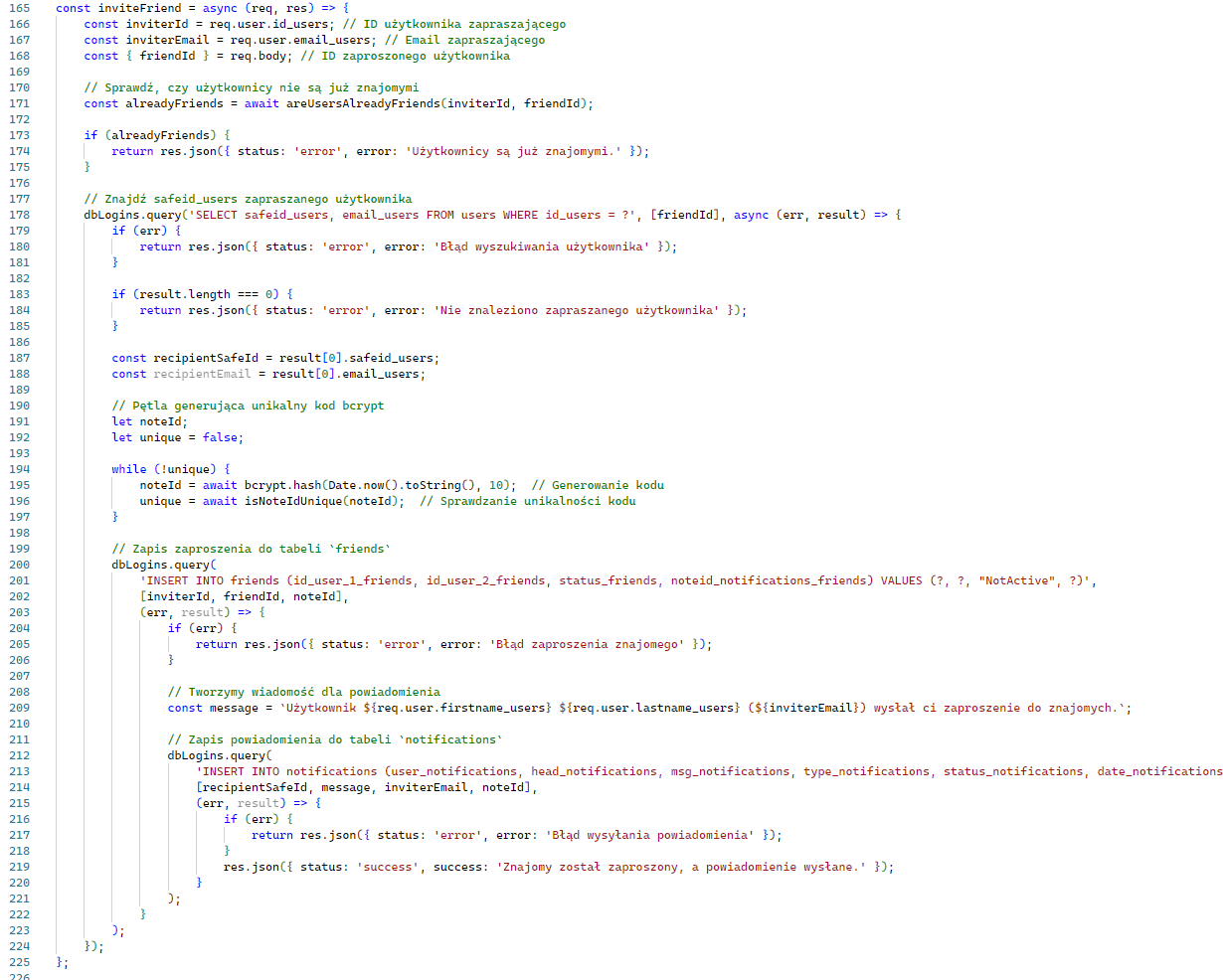


*Implementacja - system znajomych - sekcja znajomych*

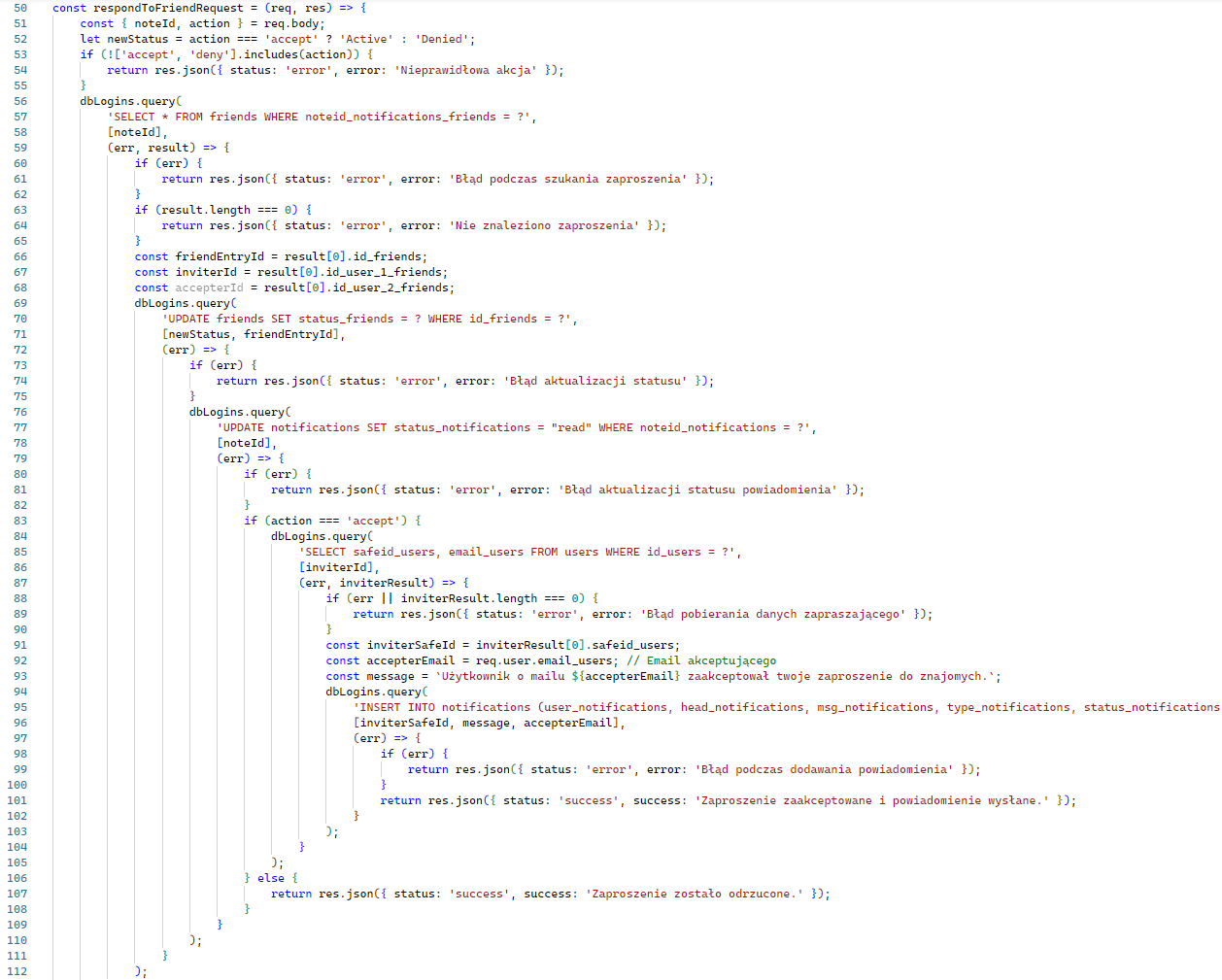


*Implementacja - system znajomych - wyszukiwanie nowych znajomych*

### 3.3.2. SYSTEM ZNAJOMYCH - LISTINGI



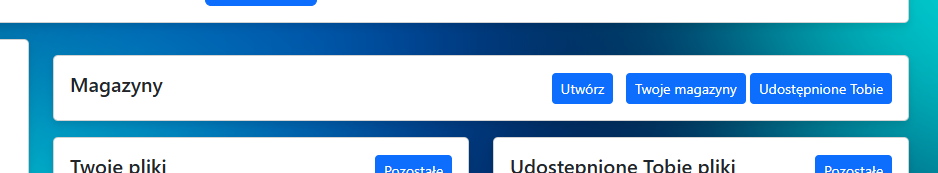
Funkcja inviteFriend jest odpowiedzialna za zapraszanie nowego znajomego w systemie oraz wysyłanie powiadomienia o zaproszeniu. Proces rozpoczyna się od pobrania identyfikatora oraz e-maila użytkownika zapraszającego, a także identyfikatora użytkownika, który ma zostać zaproszony. Funkcja sprawdza, czy użytkownicy nie są już znajomymi za pomocą funkcji areUsersAlreadyFriends. Jeśli relacja nie istnieje, wykonywane jest zapytanie do bazy danych w celu znalezienia unikalnego identyfikatora (safeid\_users) oraz e-maila zapraszanego użytkownika. Następnie generowany jest unikalny kod identyfikujący zaproszenie (noteId) przy użyciu funkcji bcrypt i pętli sprawdzającej unikalność kodu. Po wygenerowaniu kodu zaproszenie zostaje zapisane w tabeli friends z odpowiednim statusem ("NotActive"), co oznacza oczekiwanie na akceptację. Równocześnie tworzona jest wiadomość zawierająca szczegóły zaproszenia, która zostaje zapisana w tabeli notifications jako powiadomienie typu friend\_request. W przypadku powodzenia funkcja zwraca komunikat o sukcesie, informując, że znajomy został zaproszony i powiadomienie zostało wysłane. W razie wystąpienia błędów na którymkolwiek etapie (wyszukiwania użytkownika, zapisu zaproszenia lub powiadomienia), funkcja zwraca odpowiedni komunikat błędu w formacie JSON.



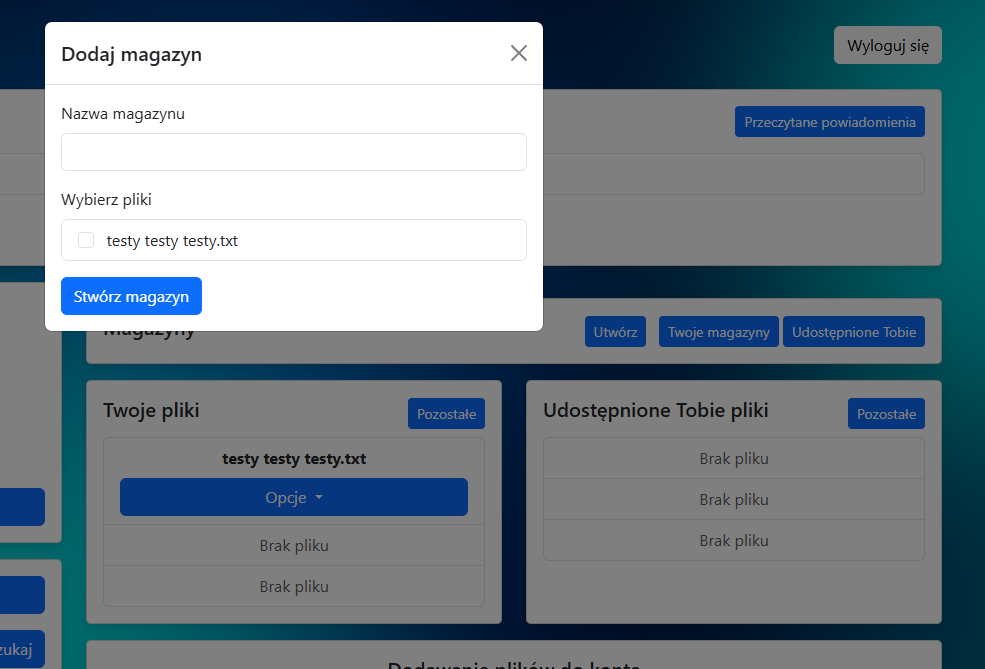
Funkcja respondToFriendRequest odpowiada za obsługę odpowiedzi na zaproszenie do znajomych, umożliwiając użytkownikowi jego zaakceptowanie lub odrzucenie. Proces rozpoczyna się od walidacji parametru action, który musi przyjmować wartość „accept” lub „deny”. Na podstawie tej akcji system ustala nowy status relacji („Active” dla akceptacji, „Denied” dla odrzucenia) i sprawdza w bazie danych, czy istnieje zaproszenie odpowiadające przesłanemu identyfikatorowi noteId. Jeśli zaproszenie zostanie znalezione, jego status jest aktualizowany w tabeli friends, a odpowiadające mu powiadomienie w tabeli notifications zostaje oznaczone jako „read”. W przypadku akceptacji system dodatkowo pobiera dane użytkownika zapraszającego (safeid\_users) i generuje powiadomienie informujące go, że jego zaproszenie zostało zaakceptowane. Powiadomienie to zawiera szczegóły akcji i jest zapisywane w tabeli notifications z odpowiednim statusem („unread”). Ostatecznie funkcja zwraca komunikat o sukcesie, informując, czy zaproszenie zostało zaakceptowane, czy odrzucone. W razie wystąpienia błędów na dowolnym etapie, takich jak problem z bazą danych lub brak zaproszenia, funkcja zwraca odpowiedni komunikat błędu.

## 3.4. IMPLEMENTACJA - SYSTEM MAGAZYNÓW

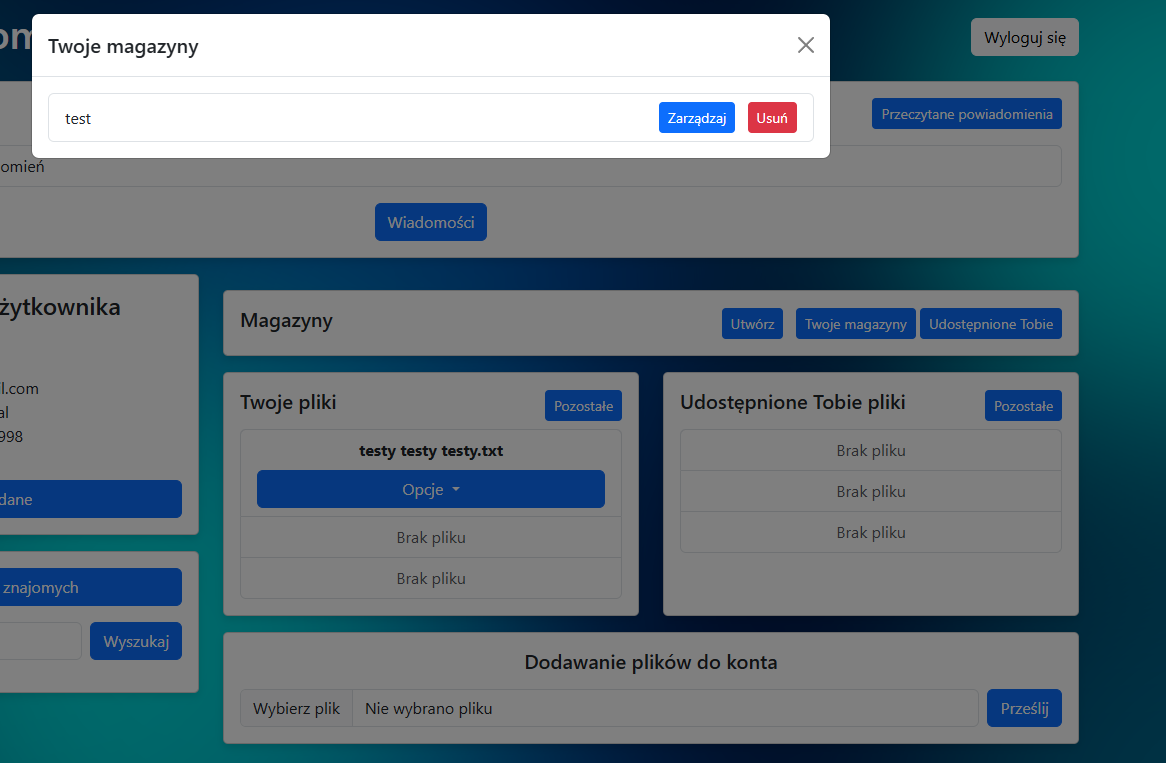
### 3.4.1. SYSTEM MAGAZYNÓW - OBRAZY



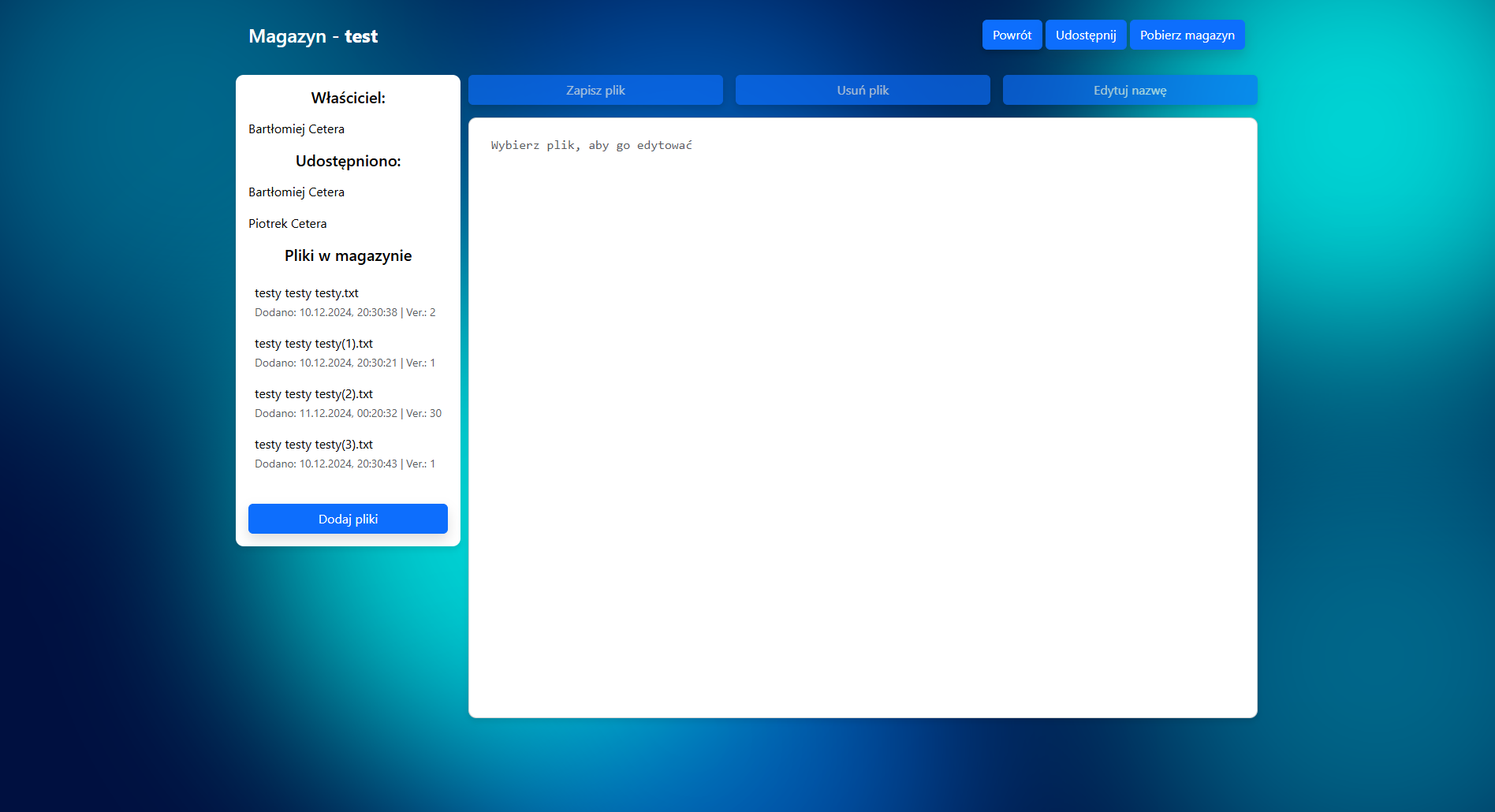
*Implementacja - system magazynów - sekcja magazynów*



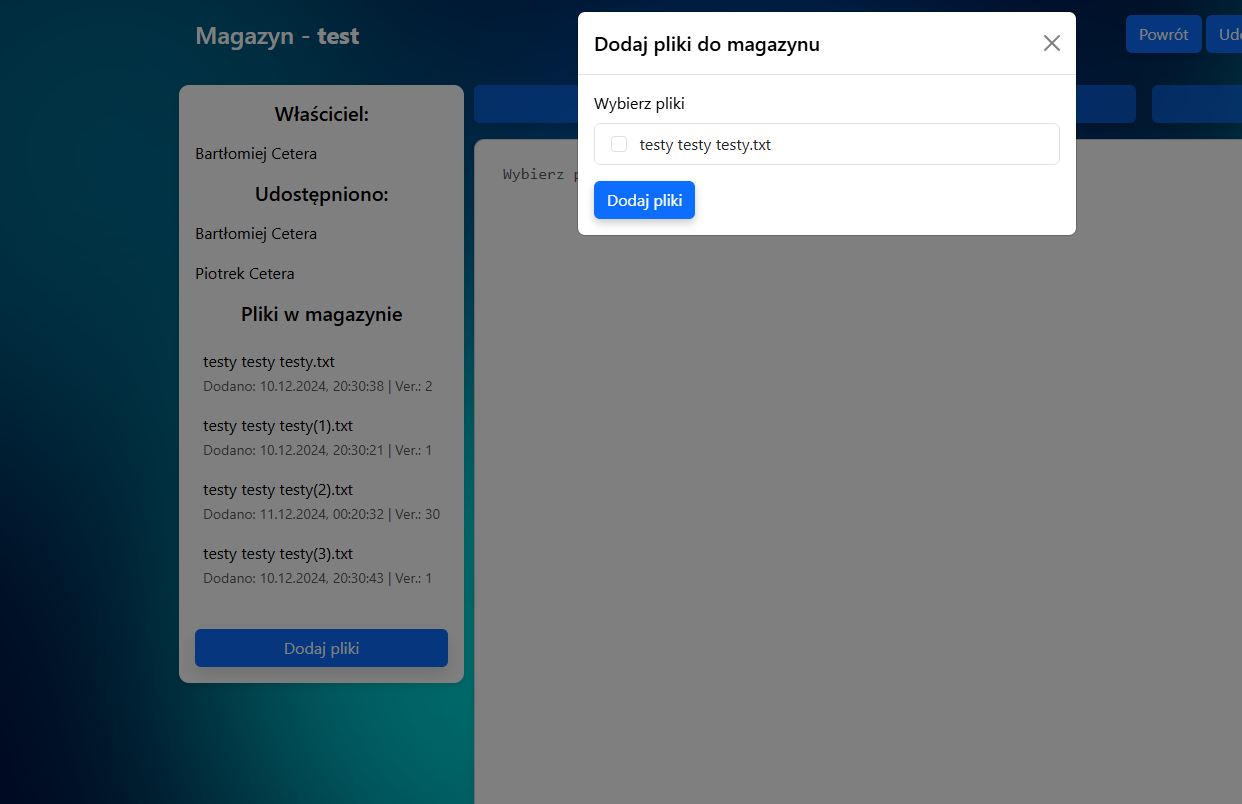
*Implementacja - system magazynów - tworzenie magazynu*



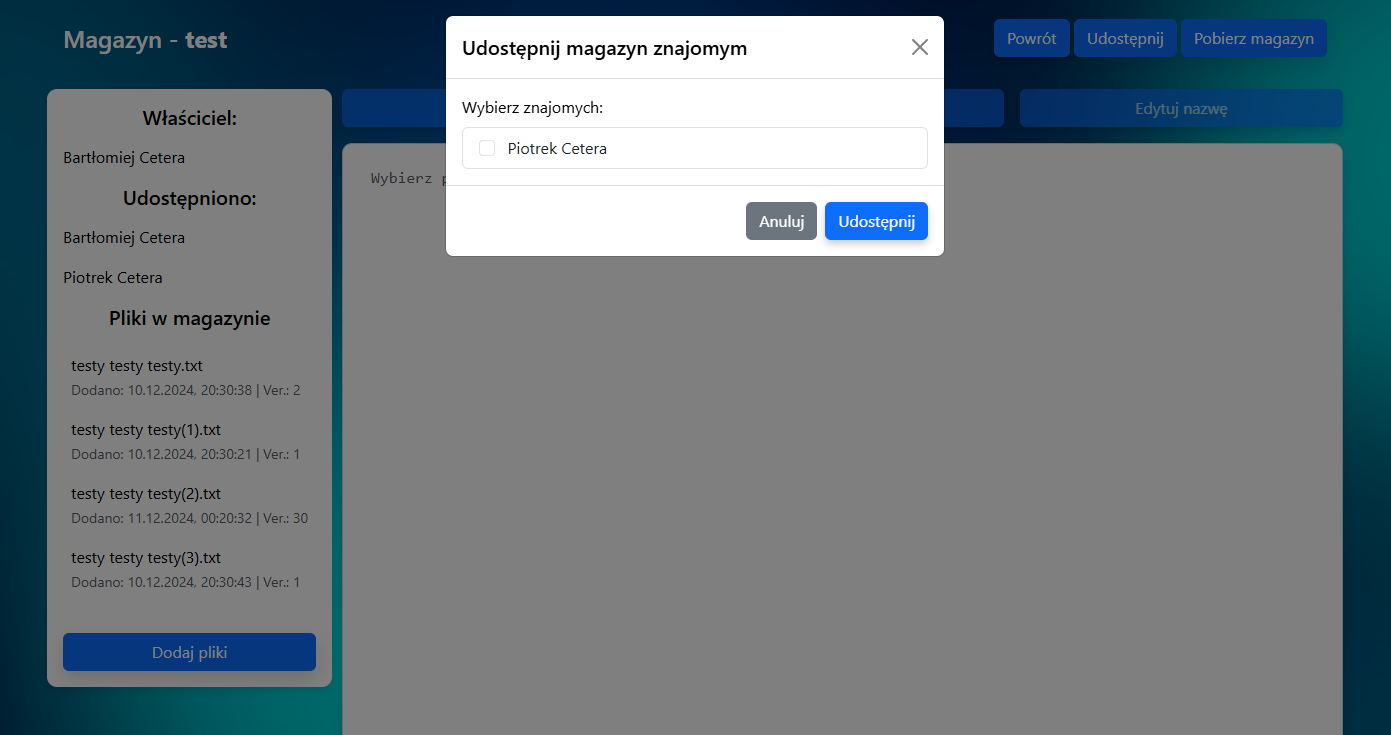
*Implementacja - system magazynów - sekcja moich magazynów*



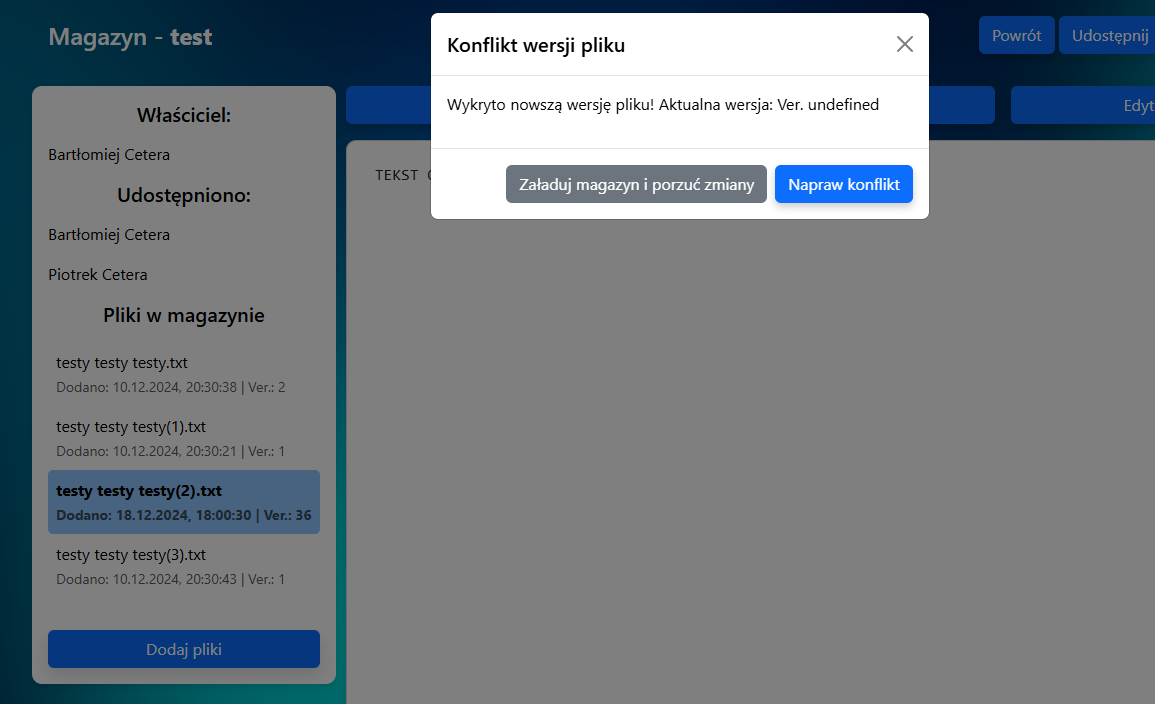
*Implementacja - system magazynów - magazyn*



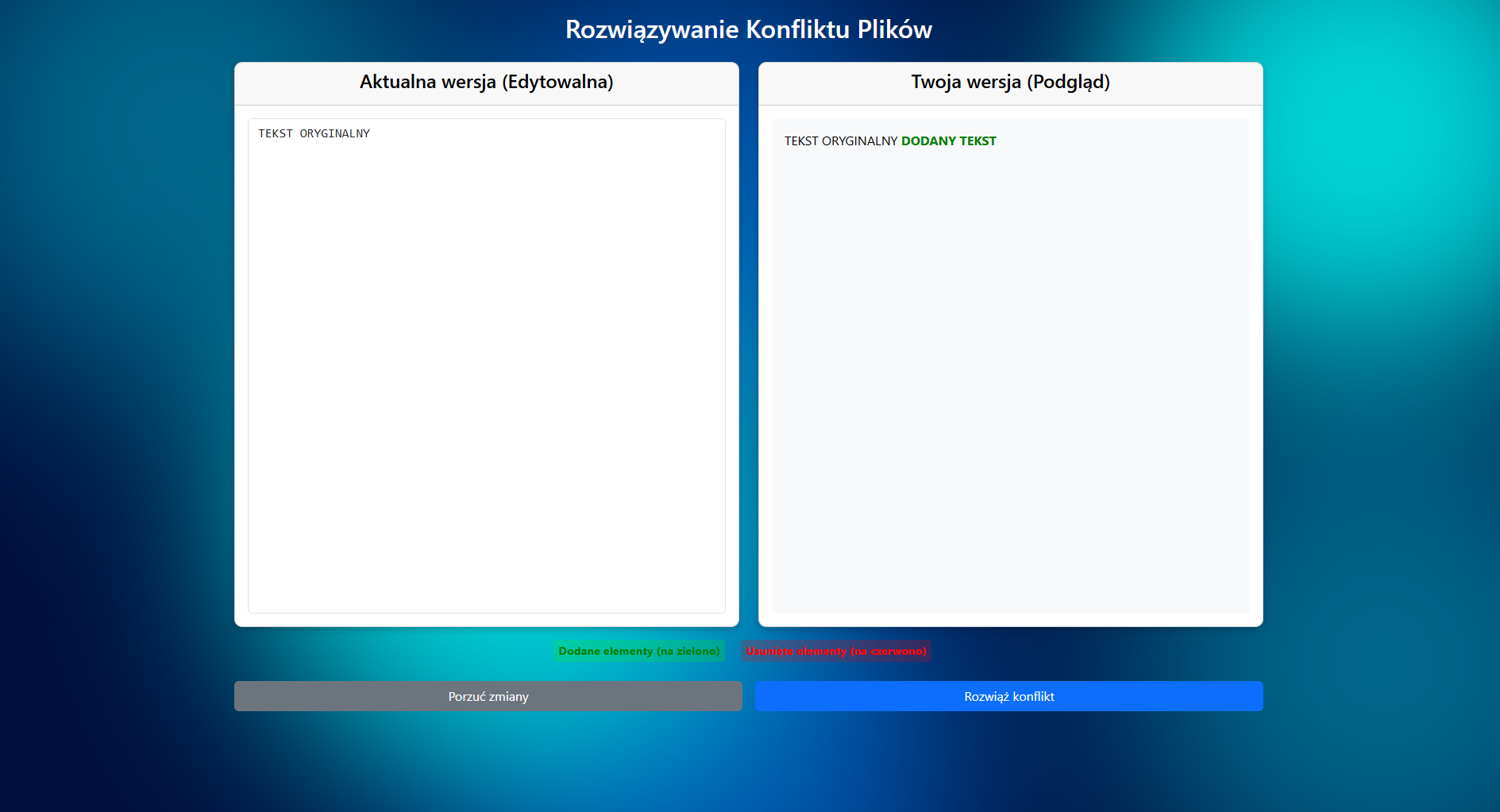
*Implementacja - system magazynów - dodawanie pliku do magazynu*



*Implementacja - system magazynów - udzielanie dostępu do magazynu*

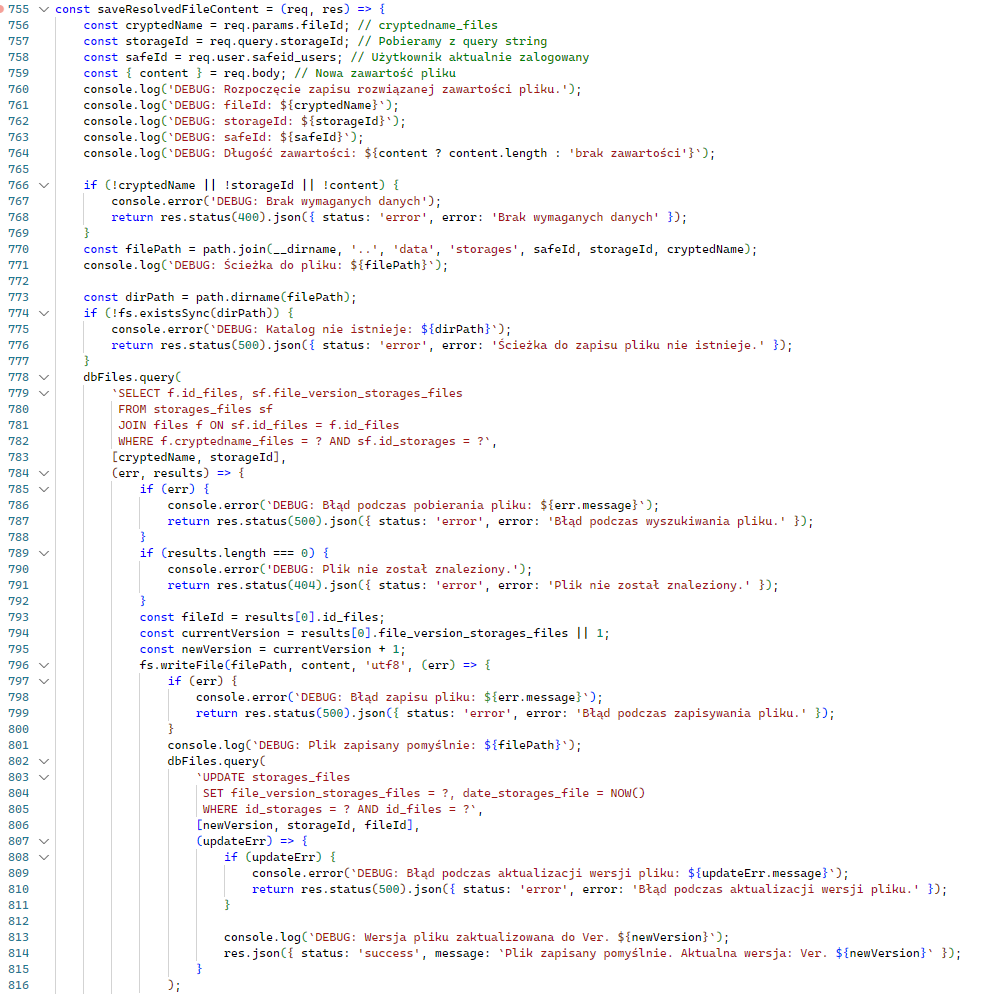


*Implementacja - system magazynów - wykrycie konfliktu w wersjonowaniu pliku w magazynie*



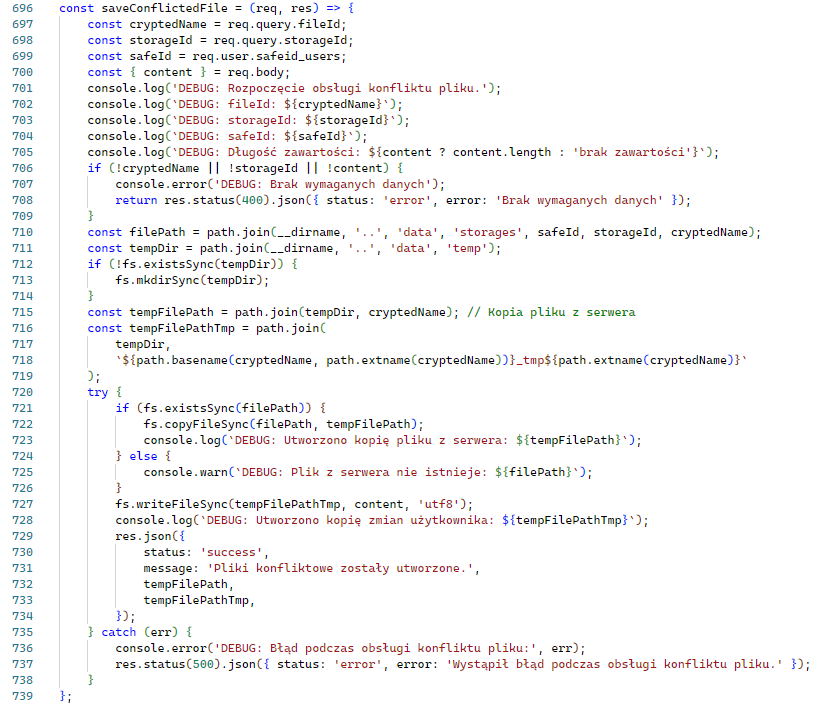
*Implementacja - system magazynów - rozwiązywanie konfliktu zawartości pliku w magazynie*

### 3.4.2. SYSTEM MAGAZYNÓW - LISTINGI



Funkcja saveResolvedFileContent odpowiada za zapisywanie rozwiązanej zawartości pliku oraz aktualizację wersji pliku w systemie magazynów. Na początku pobierane są niezbędne dane, takie jak zaszyfrowana nazwa pliku (cryptedName), identyfikator magazynu (storageId), identyfikator użytkownika (safeId) oraz nowa zawartość pliku. Funkcja sprawdza, czy wszystkie dane zostały przesłane i czy ścieżka zapisu istnieje na serwerze. Następnie wykonywane jest zapytanie do bazy danych, aby zweryfikować istnienie pliku w danym magazynie oraz pobrać jego bieżącą wersję.

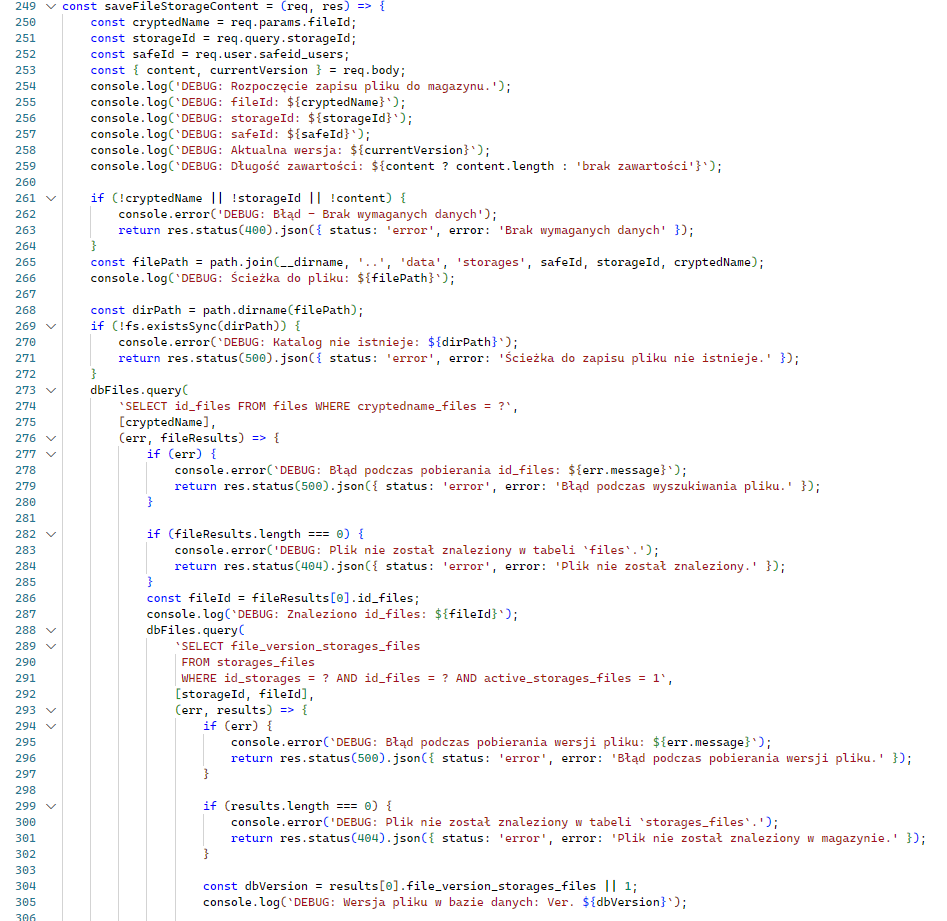
Jeżeli plik istnieje, jego zawartość zostaje nadpisana na serwerze przy użyciu funkcji fs.writeFile, a wersja pliku jest zwiększana o jeden. Nowa wersja oraz aktualna data modyfikacji są zapisywane w bazie danych poprzez aktualizację rekordu w tabeli storages\_files. W przypadku powodzenia funkcja zwraca komunikat potwierdzający zapis pliku oraz informację o jego nowej wersji. W przypadku błędów, takich jak brak wymaganych danych, problem z odczytem z bazy lub zapisem pliku na serwerze, funkcja zwraca odpowiednie komunikaty błędów z kodami statusu. Logowanie szczegółowych informacji (np. ścieżek, długości zawartości) pomaga w debugowaniu procesu.



Funkcja saveConflictedFile obsługuje sytuacje konfliktu plików w systemie, tworząc oddzielne kopie aktualnej wersji pliku oraz zmian wprowadzonych przez użytkownika. Na początku pobierane są wymagane dane, takie jak zaszyfrowana nazwa pliku (cryptedName), identyfikator magazynu (storageId), identyfikator użytkownika (safeId) oraz treść zmian. Funkcja weryfikuje, czy wszystkie niezbędne dane zostały przesłane, a następnie określa ścieżkę do oryginalnego pliku na serwerze oraz do katalogu tymczasowego, w którym będą przechowywane pliki konfliktowe. Jeśli katalog tymczasowy nie istnieje, zostaje on utworzony. Następnie funkcja sprawdza, czy oryginalny plik znajduje się na serwerze. Jeśli plik istnieje, jego kopia zostaje zapisana w katalogu tymczasowym. Równocześnie zmiany wprowadzone przez użytkownika są zapisywane jako osobny plik w tym samym katalogu, z dopiskiem \_tmp w nazwie. Na koniec funkcja zwraca komunikat o powodzeniu wraz ze ścieżkami do obu plików konfliktowych. W przypadku błędów, takich jak problem z dostępem do pliku lub operacjami na systemie plików, funkcja zwraca odpowiedni komunikat o błędzie i loguje szczegóły do debugowania.



Funkcja addFilesToStorage odpowiada za dodawanie wybranych plików użytkownika do określonego magazynu danych. Na początku pobierane są informacje o identyfikatorze magazynu (storageId), identyfikatorze użytkownika (safeId) oraz liście wybranych plików (selectedFiles). Funkcja sprawdza, czy wszystkie wymagane dane zostały przekazane i czy magazyn istnieje na serwerze. Następnie wykonuje zapytanie do bazy danych, aby zweryfikować, czy pliki należą do użytkownika i nie zostały usunięte. Dla każdego pliku funkcja sprawdza unikalność nazwy w obrębie magazynu, a w przypadku konfliktu modyfikuje nazwę pliku, dodając odpowiedni sufiks. Pliki są następnie kopiowane do katalogu magazynu z nadaniem nowej zaszyfrowanej nazwy, a informacje o plikach są zapisywane w bazie danych w tabelach files i storages\_files. Funkcja zapewnia obsługę błędów na każdym etapie procesu, takich jak problemy z dostępem do plików czy konflikty nazw, a w przypadku powodzenia zwraca komunikat potwierdzający dodanie plików do magazynu. Dzięki szczegółowemu logowaniu debugowania administratorzy mogą łatwo monitorować i rozwiązywać ewentualne problemy.



Funkcja saveFileStorageContent zajmuje się zapisywaniem zawartości pliku w magazynie danych i aktualizacją jego wersji. Na początku pobierane są kluczowe dane, takie jak zaszyfrowana nazwa pliku (cryptedName), identyfikator magazynu (storageId), identyfikator użytkownika (safeId) oraz nowa zawartość pliku i jego aktualna wersja. Funkcja sprawdza, czy wszystkie wymagane dane zostały przekazane, a także weryfikuje, czy ścieżka do magazynu istnieje na serwerze. Następnie system wykonuje zapytania do bazy danych, aby zweryfikować, czy plik istnieje w tabelach files i storages\_files. Po uzyskaniu informacji o wersji pliku zapisanej w bazie danych (dbVersion), następuje porównanie z wersją przesłaną w żądaniu (currentVersion). Jeśli wersje się różnią, funkcja zgłasza konflikt wersji i odrzuca żądanie z odpowiednim komunikatem błędu. W przypadku zgodności wersji zawartość pliku zostaje zapisana na serwerze, a wersja pliku w tabeli storages\_files jest zwiększana o jeden. Po pomyślnym zakończeniu operacji funkcja zwraca odpowiedź potwierdzającą zapis pliku oraz aktualną wersję. Jeśli w dowolnym etapie wystąpią błędy, takie jak problem z dostępem do pliku, zapisaniem zawartości czy aktualizacją wersji, funkcja zwraca odpowiedni komunikat błędu z kodem statusu. Debugowanie funkcji jest wspierane przez szczegółowe logowanie, co ułatwia rozwiązywanie potencjalnych problemów.

## 3.5. IMPLEMENTACJA - SYSTEM DWUSTOPNIWEJ AUTENTYFIKACJI

### 3.5.1. SYSTEM DWUSTOPNIWEJ AUTENTYFIKACJI - OBRAZY

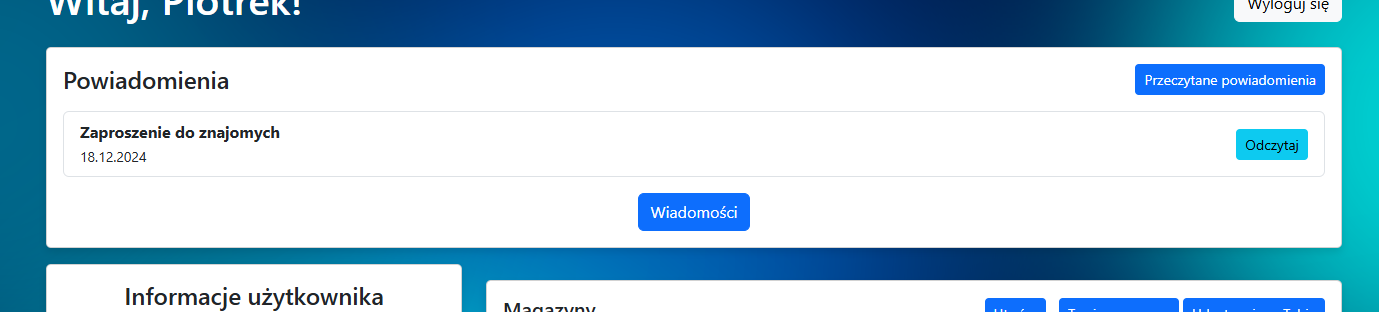
### 3.5.2. SYSTEM DWUSTOPNIWEJ AUTENTYFIKACJI - LISTINGI

## 3.6. IMPLEMENTACJA - SYSTEM SZYFROWANIA

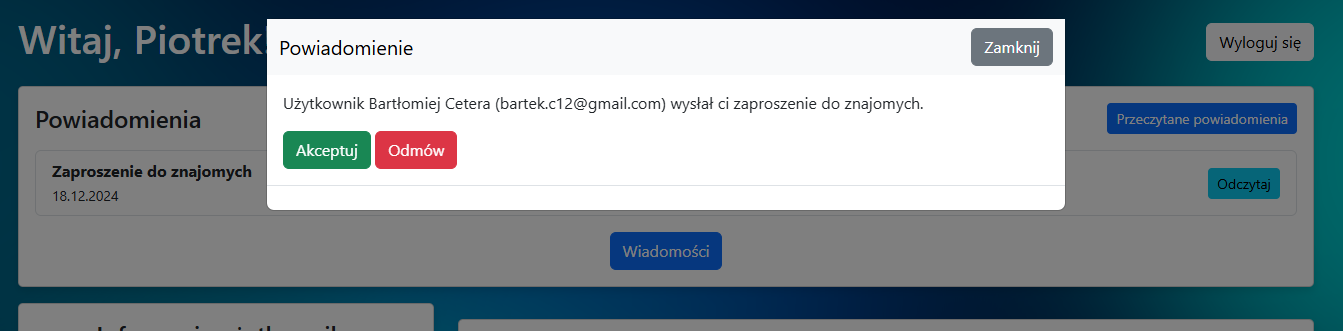
### 3.6.1. SYSTEM SZYFROWANIA - LISTINGI

## 3.7. IMPLEMENTACJA - SYSTEM POWIADOMIEŃ I WIADOMOŚCI

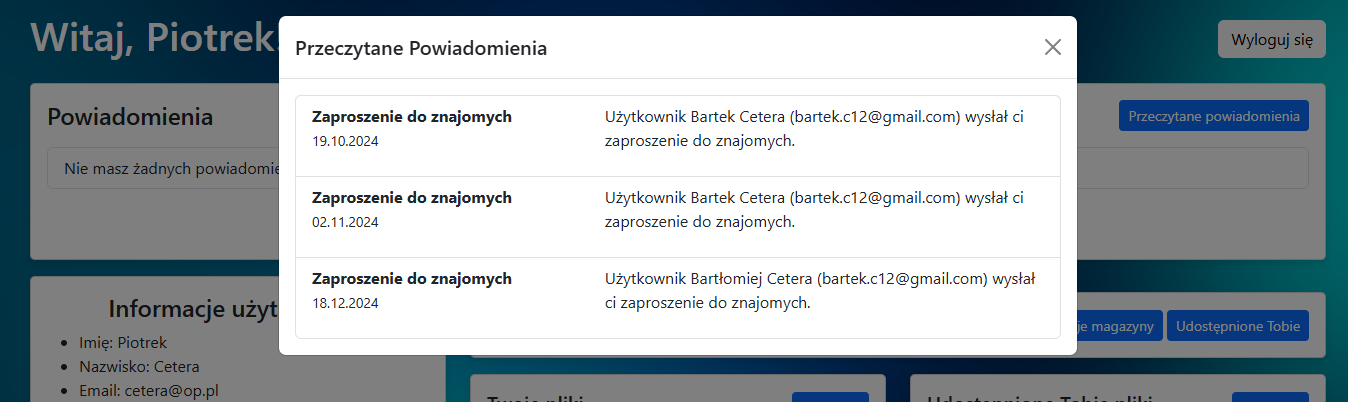
### 3.7.1. SYSTEM POWIADOMIEŃ I WIADOMOŚCI - OBRAZY



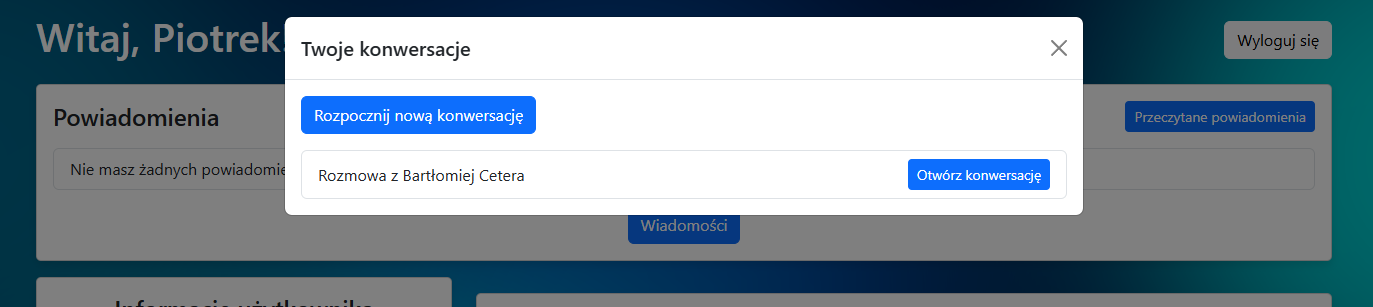
*Implementacja - system powiadomień i wiadomości - sekcja powiadomień*



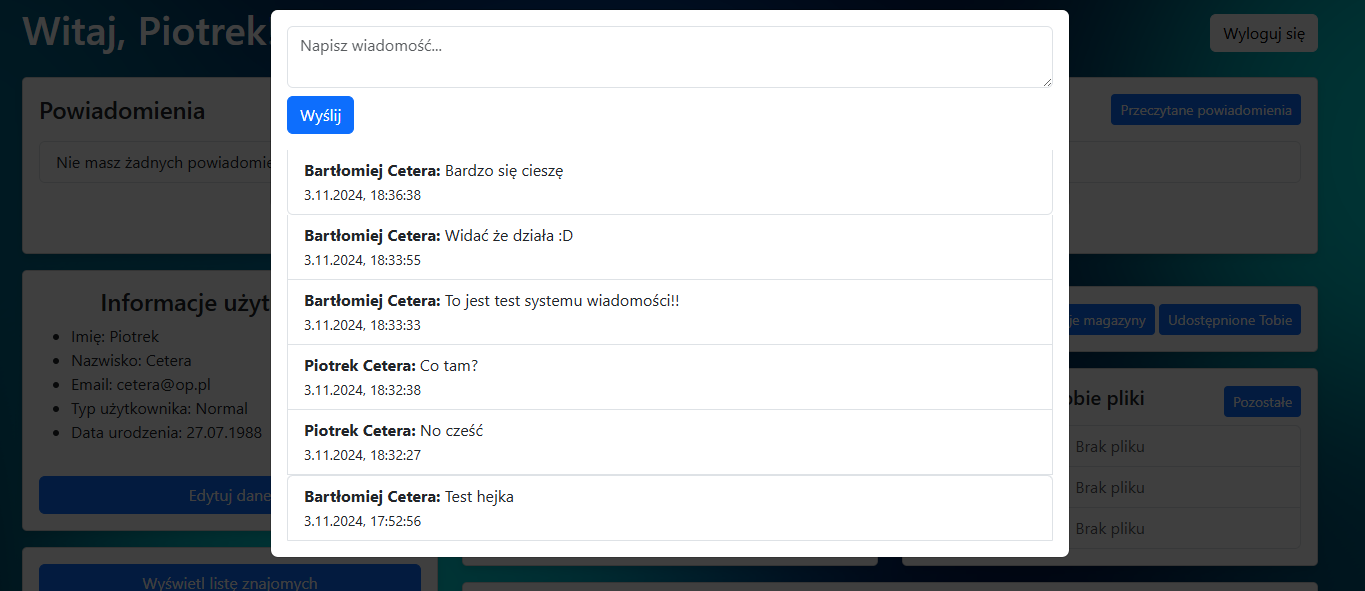
*Implementacja - system powiadomień i wiadomości - podgląd powiadomienia*



*Implementacja - system powiadomień i wiadomości - sekcja przeczytanych powiadomień*

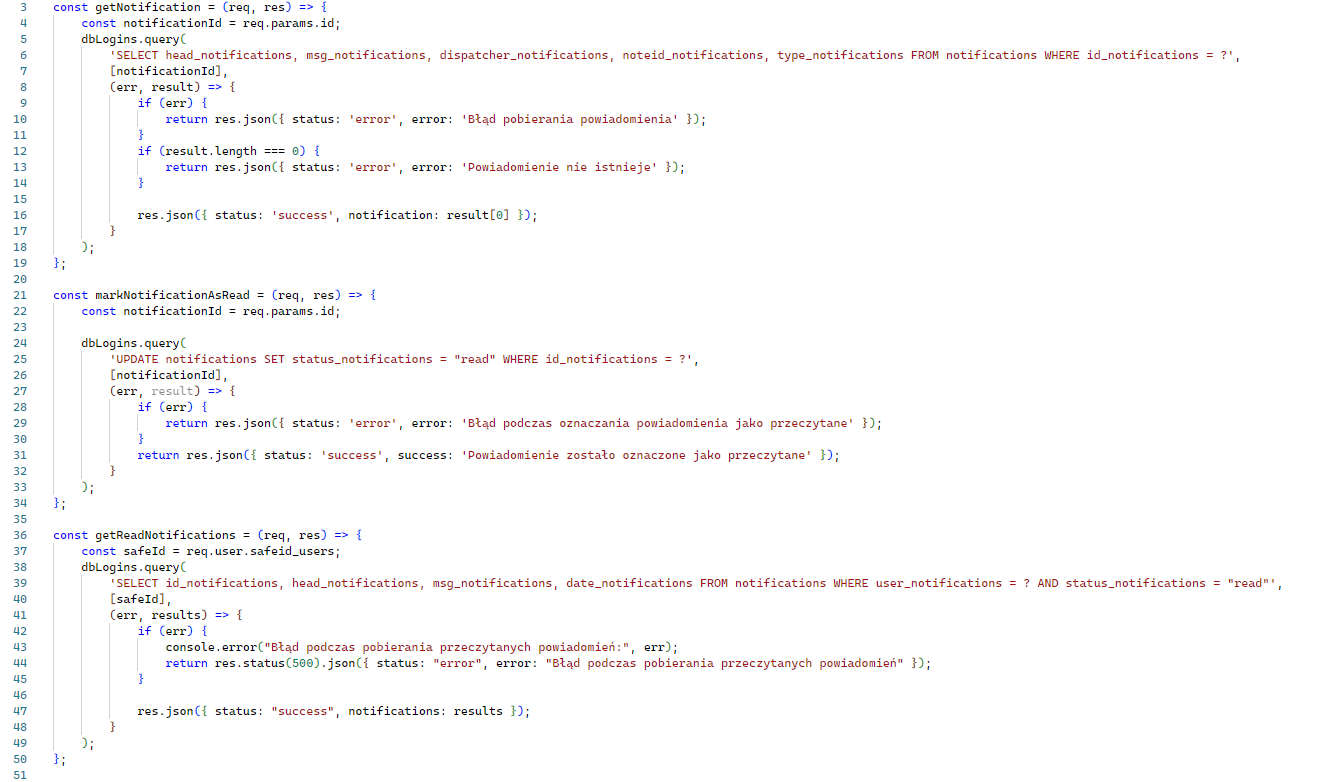


*Implementacja - system powiadomień i wiadomości - sekcja wiadomości*



*Implementacja - system powiadomień i wiadomości - konwersacja*

### 3.7.2. SYSTEM POWIADOMIEŃ I WIADOMOŚCI - LISTINGI



Ten fragment kodu zawiera trzy funkcje obsługujące powiadomienia w systemie. Pierwsza z nich, getNotification, umożliwia pobranie szczegółowych informacji o konkretnym powiadomieniu na podstawie jego identyfikatora (notificationId). Wykonywane jest zapytanie do bazy danych, aby uzyskać dane takie jak tytuł, treść, nadawca, identyfikator notyfikacji oraz typ powiadomienia. Jeśli powiadomienie istnieje, zwracane są jego szczegóły w odpowiedzi JSON. W przeciwnym wypadku funkcja zwraca komunikat o błędzie. Druga funkcja, markNotificationAsRead, służy do oznaczania konkretnego powiadomienia jako przeczytane. Na podstawie identyfikatora powiadomienia (notificationId) wykonywane jest zapytanie aktualizujące w bazie danych pole status\_notifications, zmieniając jego wartość na „read”. W przypadku powodzenia funkcja zwraca komunikat o sukcesie, a w razie błędów zwraca odpowiedni komunikat o błędzie. Trzecia funkcja, getReadNotifications, pozwala na pobranie listy wszystkich przeczytanych powiadomień dla danego użytkownika. Na podstawie identyfikatora użytkownika (safeId) system wyszukuje w bazie danych powiadomienia oznaczone jako „read” i zwraca ich listę w odpowiedzi JSON, zawierając szczegóły takie jak tytuł, treść i data. W razie błędów funkcja zwraca komunikat o błędzie z kodem statusu 500. Kod ten umożliwia zarządzanie powiadomieniami w sposób przejrzysty i efektywny, a dzięki zastosowaniu odpowiednich zapytań SQL zapewnia spójność danych w systemie.

# 4. PODSUMOWANIE

W niniejszej pracy dyplomowej zrealizowano projekt aplikacji webowej do przechowywania plików, która łączy w sobie nowoczesne technologie z priorytetem na ochronę danych użytkowników. System oferuje szereg funkcjonalności, takich jak zarządzanie plikami, tworzenie i współdzielenie magazynów, dwustopniowa autoryzacja, czy moduł znajomych z funkcją wysyłania powiadomień. Kluczowym elementem aplikacji jest wdrożenie zaawansowanego szyfrowania danych, które zapewnia pełne bezpieczeństwo przechowywanych plików, nawet w przypadku naruszenia serwera. Do realizacji projektu wykorzystano technologie takie jak Node.js, Express.js, MySQL oraz JWT, co umożliwiło stworzenie skalowalnej i wydajnej architektury aplikacji. Implementacja algorytmu AES zwiększyła bezpieczeństwo danych, a zastosowanie tokenów JWT zapewniło wygodne uwierzytelnianie użytkowników. Efektem końcowym jest aplikacja, która nie tylko stanowi alternatywę dla istniejących rozwiązań, ale także wyróżnia się przejrzystym interfejsem i możliwością intuicyjnego zarządzania danymi. Stworzony system odpowiada na współczesne potrzeby użytkowników, zapewniając im pełną kontrolę nad przechowywanymi plikami, jednocześnie ułatwiając współpracę w środowisku cyfrowym.

# 5. BIBLIOGRAFIA

## 5.1. BIBLIOGRAFIA - LINKI

* <https://cyberdefence24.pl/> - [*portal o cyberbezpieczeństwie CyberDefence24*](https://cyberdefence24.pl/)
* <https://instytutcyber.pl/> - [*portal o tym, jak być bezpiecznym w sieci*](https://instytutcyber.pl/) *instytutcyber*
* <https://www.youtube.com/@MateuszChrobok> - *Mateusz Chrobok*
* https://www.youtube.com/@pawelhordynski - [*Paweł Hordyński*](https://www.youtube.com/@pawelhordynski)
* <https://www.youtube.com/@JerzyAndrzejSurma> - [*Jerzy Surma*](https://www.youtube.com/@JerzyAndrzejSurma)
* <https://pl.wikipedia.org> - [*Wikipedia, wolna encyklopedia*](https://pl.wikipedia.org/wiki/Wikipedia:Strona_g%C5%82%C3%B3wna)

## 5.1. BIBLIOGRAFIA - KSIĄŻKI

* Bezpieczeństwo w chmurze *Autor:* [*Chris Dotson*](https://helion.pl/autorzy/chris-dotson)
* Internet Fenomen społeczeństwa informacyjnego - *Autor: Don Heath  Marek Robak Andrzej Dzięga Kosma Złotowski Radosław Chmura*
* Twoje bezpieczeństwo w świecie cyber i sztucznej inteligencji *Autor: Gołębiowski Dariusz*
* Node.js w akcji - *Autorzy:* [*Mike Cantelon*](https://helion.pl/autorzy/mike-cantelon)*, [Marc Harter](https://helion.pl/autorzy/marc-harter),*[*TJ Holowaychuk*](https://helion.pl/autorzy/tj-holowaychuk)*,*[*Nathan Rajlich*](https://helion.pl/autorzy/nathan-rajlich)
* JavaScript. Aplikacje WWW - *Autorzy:* [*Alex MacCaw*](https://helion.pl/autorzy/alex-maccaw)
* MySQL. Jak zaprojektować i wdrożyć wydajną bazę danych. - *Autorzy: Vinicius M. Grippa, Sergey Kuzmichev*
* UML 2.0 *Autor:* [*Russ Miles*](https://helion.pl/autorzy/russ-miles)*,*[*Kim Hamilton*](https://helion.pl/autorzy/kim-hamilton)

# 6. ZAŁĄCZNIKI

* Aplikacja nagrana na płytę CD
* Instrukcja instalacyjna
* Wniosek o temat pracy dyplomowej