



Uniwersytet Rzeszowski  
Kolegium Nauk Przyrodniczych  
Instytut Informatyki

# **Praca projektowa programowanie obiektowe**

## *System dla Komunikacji Connect*

Prowadzący:

mgr inż. Ewa Żesławska

Autor:

*Dmytro Gnatyk*

nr albumu: 120488

Kierunek: Informatyka, grupa lab I

Rzeszów 2024

## Spis treści

1. Opis założeń projektu .....	3
2. Opis struktury projektu .....	5
3. Harmonogram realizacji projektu.....	9
4. Prezentacja warstwy użytkowej projektu .....	11
5. Podsumowanie .....	19
6. Literatura .....	20

## **1. Opis założeń projektu**

Connect to prosta i intuicyjna aplikacja, która umożliwia bezpośrednie i szybkie połączenia między użytkownikami. Niezależnie od tego, czy chcesz rozmawiać z przyjaciółmi, czy kontaktować się z rodziną, Connect zapewnia wygodne narzędzie do komunikacji online.

Dzięki prostemu interfejsowi możesz łatwo wysłać wiadomości tekstowe. Rejestracja i logowanie są szybkie i bezproblemowe, dzięki czemu można natychmiast zacząć korzystać z wszystkich funkcji Connect.

### **• Cele i założenia projektu**

Celem projektu jest opracowanie nowoczesnej aplikacji umożliwiającej szybką i łatwą komunikację pomiędzy użytkownikami. Głównym problemem do rozwiązania jest potrzeba efektywnego wysyłania powiadomień w celu optymalizacji czasu użytkowników.

Problem ten staje się coraz bardziej istotny, co potwierdzają liczne badania wykazujące spadek tradycyjnych form komunikacji i rosnące zainteresowanie cyfrowymi narzędziami. Rozwiązaniem jest stworzenie aplikacji, która umożliwia użytkownikom natychmiastowe wysyłanie wiadomości oraz zarządzanie nimi w sposób przejrzysty i wygodny.

Realizacja projektu obejmuje etapy projektowania, implementacji, testowania i wdrożenia aplikacji. Kluczowy aspekt projektu to zaprojektowanie intuicyjnego interfejsu, który będzie łatwy w nawigacji i obsłudze dla wszystkich użytkowników, niezależnie od ich doświadczenia z technologią.

Projekt koncentruje się na stworzeniu kompleksowego systemu, który umożliwi szybką wymianę informacji oraz efektywne zarządzanie komunikacją. Wynikiem pracy będzie aplikacja, która integruje użytkowników w cyfrowym środowisku, eliminując tradycyjne bariery związane z komunikacją i wymianą informacji.

## **2. Specyfikacja wymagań**

W tym rozdziale przedstawiono szczegółowe wymagania funkcjonalne oraz niefunkcjonalne projektowanej aplikacji do komunikacji.

### **Wymagania Funkcjonalne**

#### **Rejestracja i logowanie**

- Aplikacja musi umożliwiać użytkownikom rejestrację konta, wymagając podania danych takich jak imię, adres e-mail, i hasło.
- Aplikacja musi umożliwiać logowanie do konta jest możliwe po wprowadzeniu odpowiednich danych uwierzytniających.

#### **Edycja danych użytkownika**

- Aplikacja musi umożliwiać edycję danych użytkownika. Takich jak: nazwa użytkownika, adres email oraz hasło

#### **Wysyłanie powiadomień**

- Aplikacja musi umożliwiać przesyłanie powiadomień.

#### **Zarządzanie kontem**

- Aplikacja musi umożliwiać rejestrację oraz logowanie dla każdego użytkownika.

#### **Edycja danych użytkownika**

- Aplikacja musi umożliwiać edycję danych użytkownika. Takich jak: nazwa użytkownika, adres email oraz hasło

#### **Dodawanie innych użytkowników**

- Aplikacja musi umożliwiać dodawanie użytkowników dla rozmowy.

### **Wymagania Niefunkcjonalne**

#### **Użyteczność**

- Interfejs aplikacji jest intuicyjny i przyjazny dla użytkowników o różnym stopniu zaawansowania technicznego.

#### **Bezpieczeństwo**

- Dane użytkowników, w tym dane logowania, są przechowywane i przesyłane w sposób bezpieczny, zgodnie z aktualnymi standardami ochrony danych.

### **Wydajność**

- Czasy odpowiedzi na żądania użytkowników są krótkie, zapewniając płynne doświadczenie użytkownika bez opóźnień.

### **Dostępność**

- System powinien być dostępny 24/7, minimalizując przerwy techniczne i konieczność przestoju.

### **Skalowalność**

- Architektura systemu powinna być elastyczna, umożliwiając łatwe dostosowanie do rosnącej liczby użytkowników i danych.

### **Utrzymanie i wsparcie**

- System powinien być łatwy w utrzymaniu, a wszelkie aktualizacje i naprawy błędów powinny być przeprowadzane sprawnie.

### **Podsumowanie**

Niniejszy projekt aplikacji Connect kładzie nacisk na usprawnienie komunikacji między użytkownikami poprzez wprowadzenie nowoczesnych rozwiązań cyfrowych. Aplikacja umożliwia szybkie i intuicyjne wysyłanie wiadomości tekstowych, zapewniając jednocześnie wysoki poziom bezpieczeństwa, wydajności i użyteczności. Dzięki elastycznej architekturze systemu, aplikacja jest przygotowana do obsługi rosnącej liczby użytkowników i danych, zapewniając nieprzerwaną dostępność i minimalne czasy odpowiedzi, co sprawia, że jest idealnym narzędziem do szybkiej i efektywnej komunikacji w erze cyfrowej.

### **3. Opis struktury projektu**

- Środowisko programistyczne Javy: Java JDK Kit 22.0.1 / JavaFX
- Środowisko programistyczne Javy: Java JDK Kit 17.0.0 / Spring Boot
- Web serwer: na przykład Apache Tomcat 10.1
- System zarządzania relacyjną bazą danych: PostgreSQL
- Wykorzystywane narzędzia: IntelliJ IDEA 2023.3.4
- Rekomendowane wymagania sprzętowe:
  - Wolne miejsce na dysku: Minimum 5GB
  - Procesor zgodny z architekturą x86\_64
  - Pamięć RAM: Minimum 4GB
  - System operacyjny: Windows 10

## Struktura bazy danych

### Tabela users

- Przechowuje dane użytkowników, takie jak unikalny identyfikator, nazwa użytkownika, e-mail, hasło oraz daty utworzenia i aktualizacji. Jest podstawą rejestracji i zarządzania użytkownikami.

### Tabela chatrooms

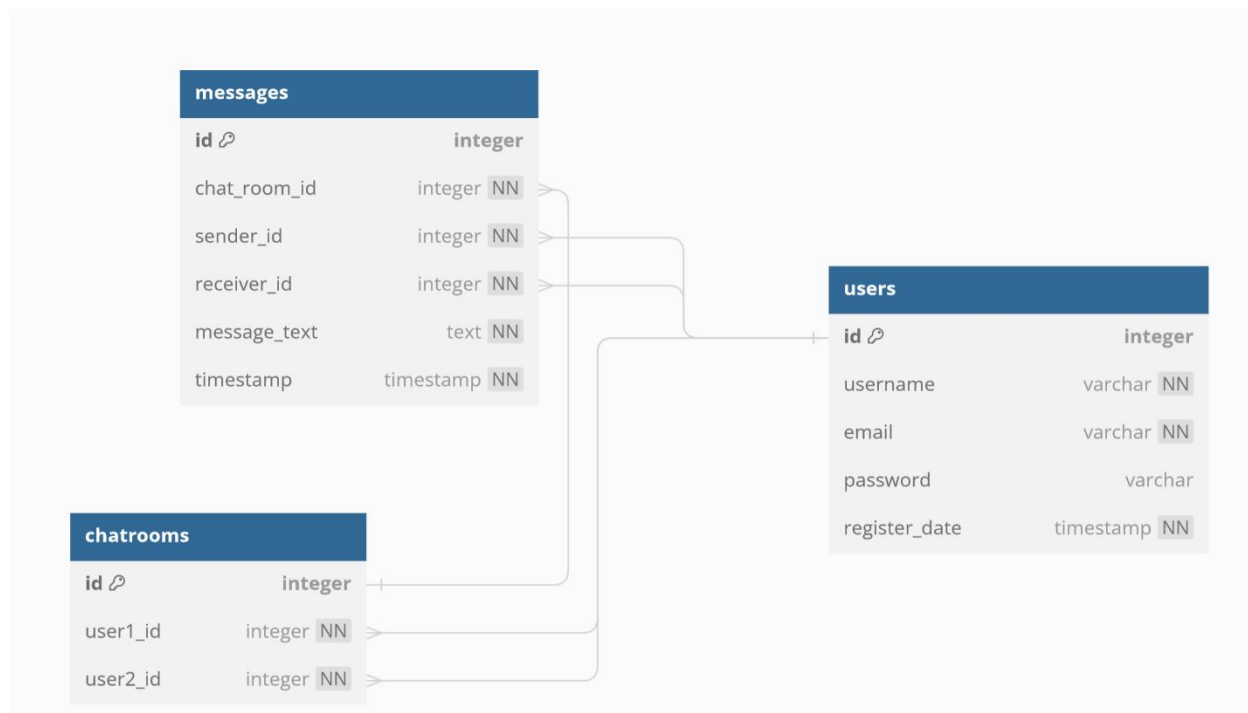
- Przechowuje informacje o pokojach czatowych, w tym unikalny ID, identyfikatory użytkowników (User1ID, User2ID) oraz daty utworzenia i aktualizacji pokoju. Umożliwia zarządzanie pokojami i komunikacją między użytkownikami.

### Tabela messages

- Przechowuje wiadomości z unikalnym ID, identyfikatorem pokoju czatowego nadawcy odbiorcy treścią wiadomości oraz datą wysłania. Śledzi historię komunikacji między użytkownikami.

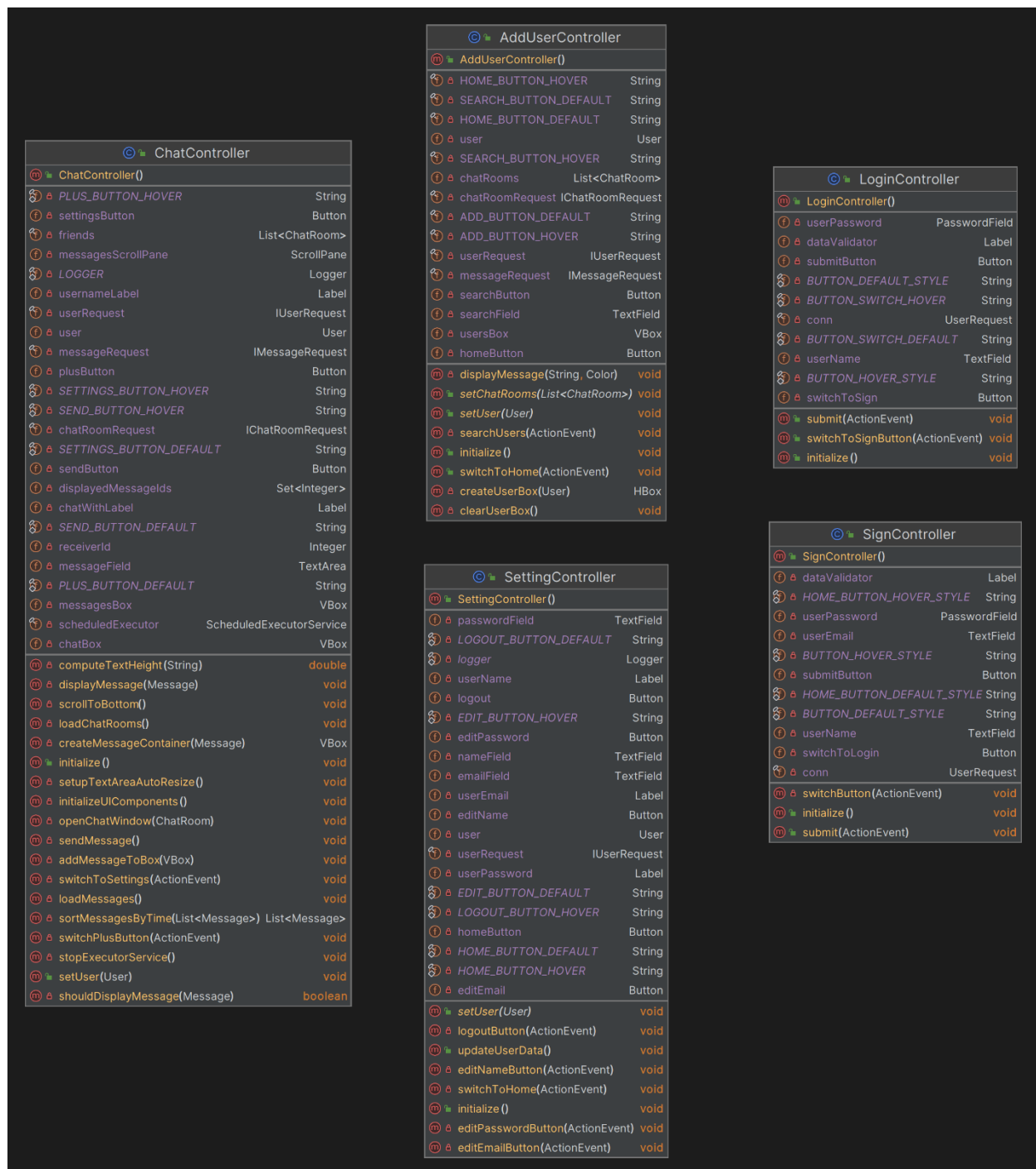
### Integracja z Spring Data JPA

- Umożliwia zarządzanie bazą danych i operacjami CRUD. Zapewnia efektywne mapowanie obiektów Javy na rekordy w bazie danych, wspierając zarządzanie użytkownikami, pokojami czatowymi i wiadomościami w systemie.



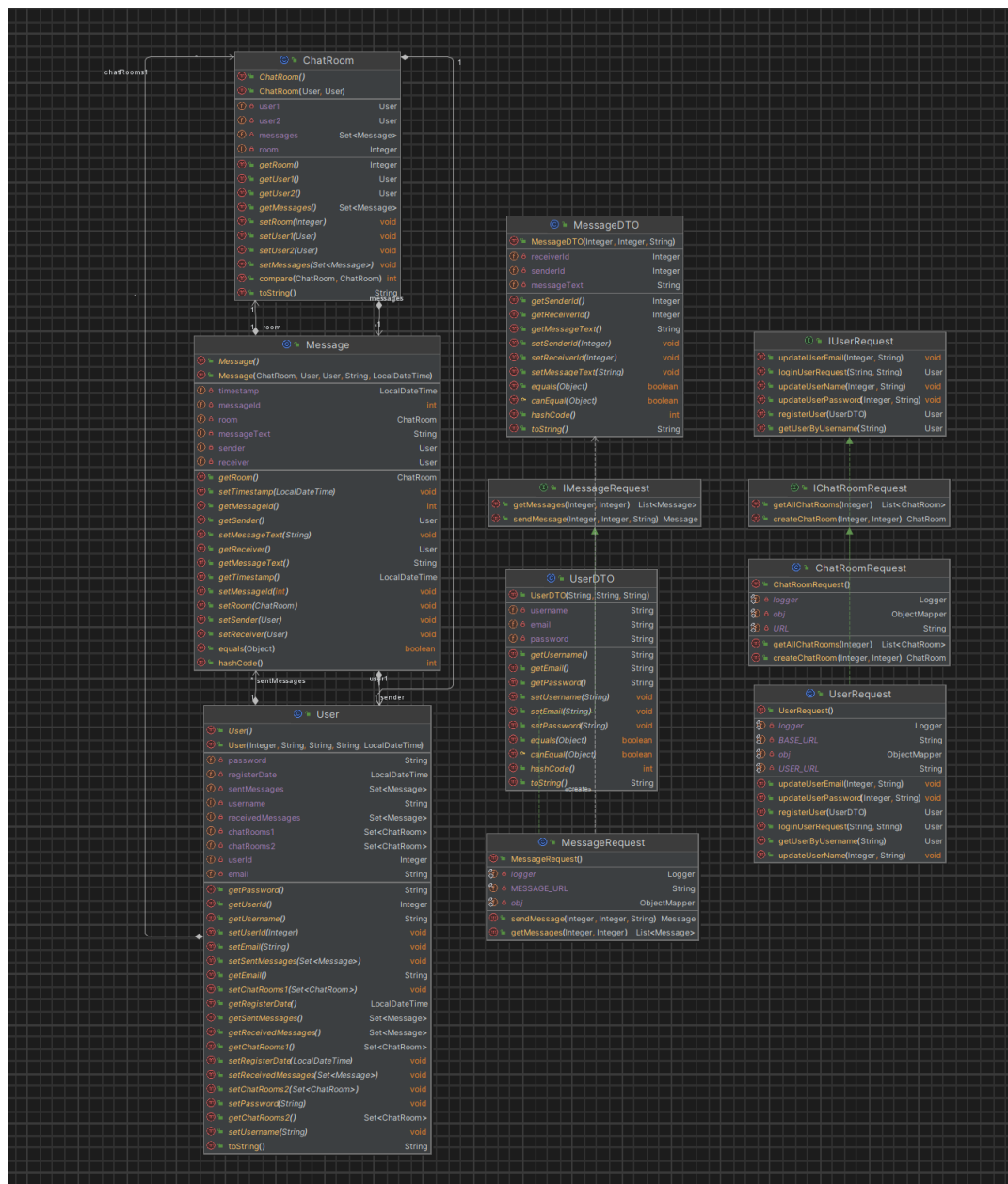
Rysunek 1. Diagram ERD

## Diagramy klas zostały podzielone na części Klient oraz Serwer



Rysunek 2. Diagram klas pakietu Controller (Część Klient)

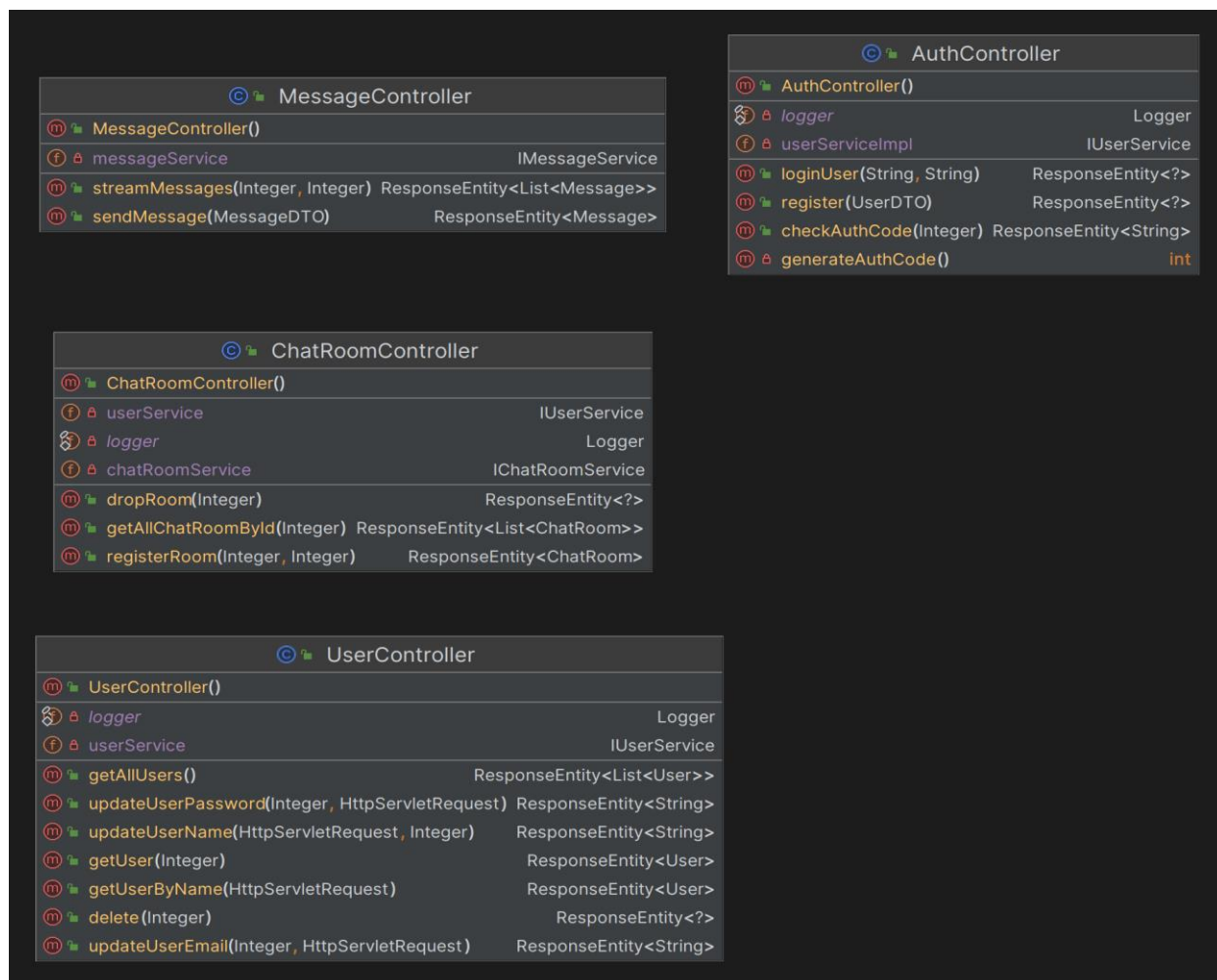
Na tym diagramie przedstawiono klasy kontrolerów, które zarządzają różnorodnymi operacjami w aplikacji klienckiej. Każdy kontroler odpowiada za specyficzne funkcje związane z interakcją użytkownika oraz obsługą logiki aplikacji. Kontrolery te pośredniczą między warstwą prezentacji a warstwą logiki biznesowej, przetwarzając dane wejściowe użytkownika, zarządzając stanem aplikacji i obsługując odpowiednie akcje.



### Rysunek 3. Diagram klas pakietu Model (Część Klient)

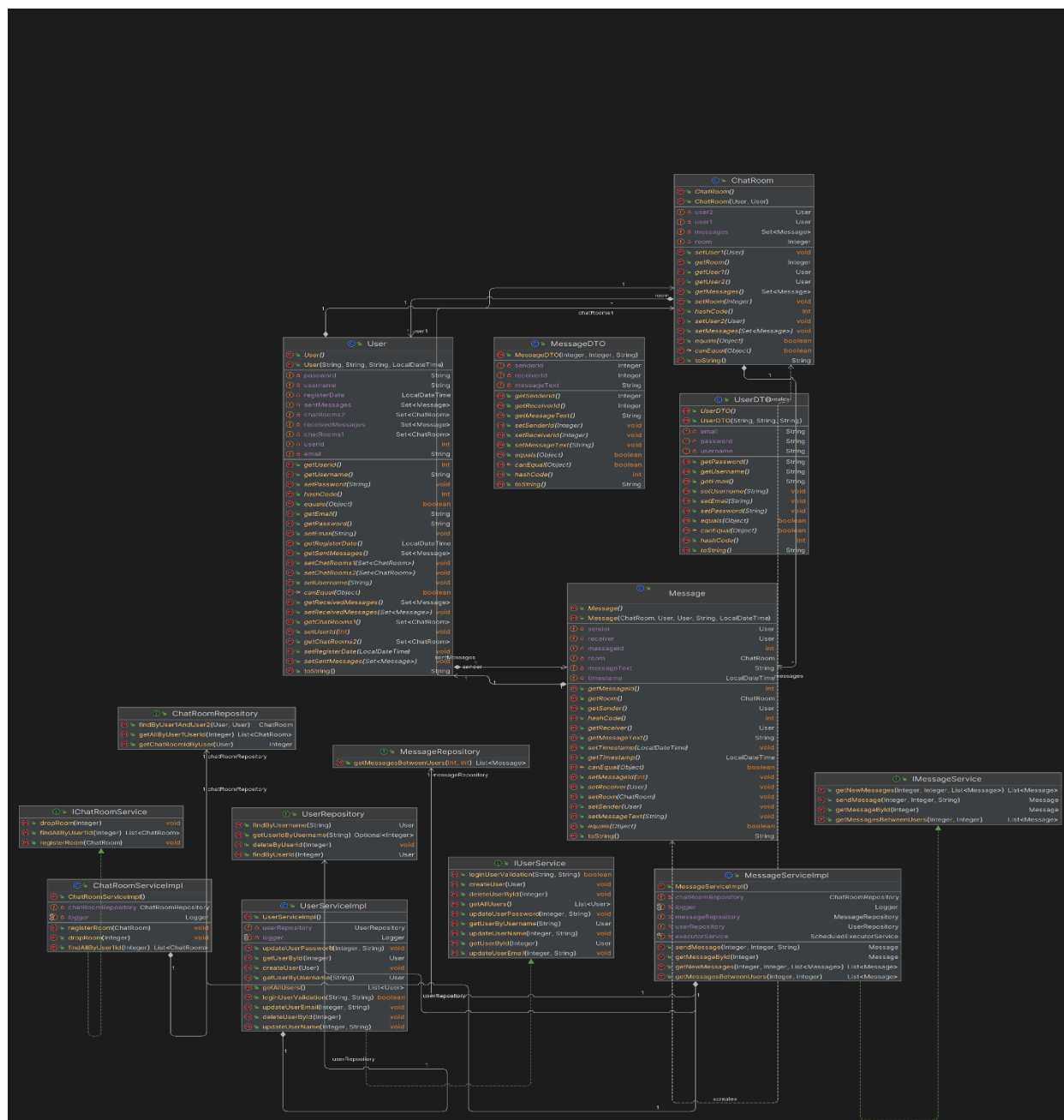
Drugi diagram przedstawia klasy modelu, które definiują strukturę danych i logikę biznesową aplikacji po stronie klienta. Klasy modelu reprezentują kluczowe obiekty domeny, takie jak użytkownicy, pokoje czatu i wiadomości. Model ten obsługuje operacje związane z manipulacją danymi, relacjami między obiektami oraz zachowaniem integralności i spójności danych w aplikacji.





Rysunek 4. Diagram klas pakietu Controller (Część Serwerowa)

Diagram przedstawia klasy kontrolerów w części serwerowej aplikacji, które zarządzają logiką biznesową dotyczącą użytkowników, pokoiów czatu, wiadomości oraz procesów autoryzacji. Kontrolery te obsługują żądania HTTP, przekazując dane do odpowiednich usług oraz zwracając odpowiedzi do klienta. Ich zadaniem jest zapewnienie, że serwer przetwarza dane zgodnie z wymogami logiki biznesowej aplikacji, a także utrzymuje bezpieczną i efektywną komunikację z aplikacją kliencką.



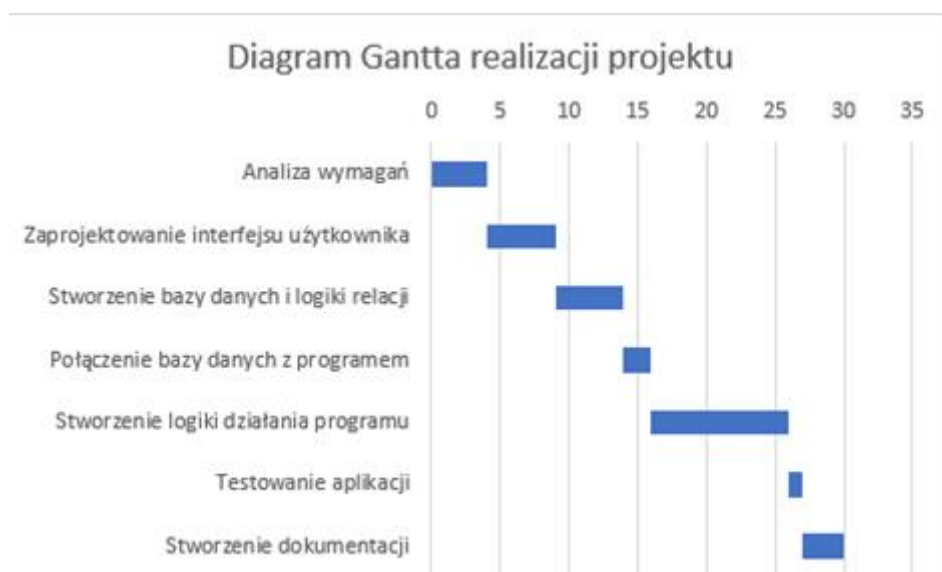
Rysunek 5. Diagram klas pakietu Model (Część Serwerowa)

W pakiecie Model w części serwerowej znajdują się klasy reprezentujące dane i logikę biznesową specyficzną dla strony serwera. Klasy te definiują obiekty domeny, takie jak użytkownicy, pokoje czatu, wiadomości oraz związane z nimi procesy biznesowe. Model ten odgrywa kluczową rolę w przechowywaniu, przetwarzaniu i manipulacji danymi, zapewniając integralność informacji i spójność w całym systemie. Dodatkowo, pakiet ten może obejmować DTO (Data Transfer Objects), które ułatwiają transfer danych między warstwą serwera a klientem, oraz encje danych.

#### 4. Harmonogram realizacji projektu

Podczas realizacji projektu napotkano kilka wyzwań, w tym optymalizację zapytań na serwer, dostosowanie wyglądu do innych okien oraz dynamicznego wyświetlenia powiadomień z pokojami.

Dzięki regularnej pracy udało się skutecznie przezwyciężyć te trudności, co przyczyniło się do sukcesu końcowej realizacji projektu. Poniżej przedstawiono diagram Gantta, który ilustruje czas poświęcony na poszczególne etapy projektu. Najwięcej czasu zajęło tworzenie logiki działania programu, a najmniej testowanie aplikacji.



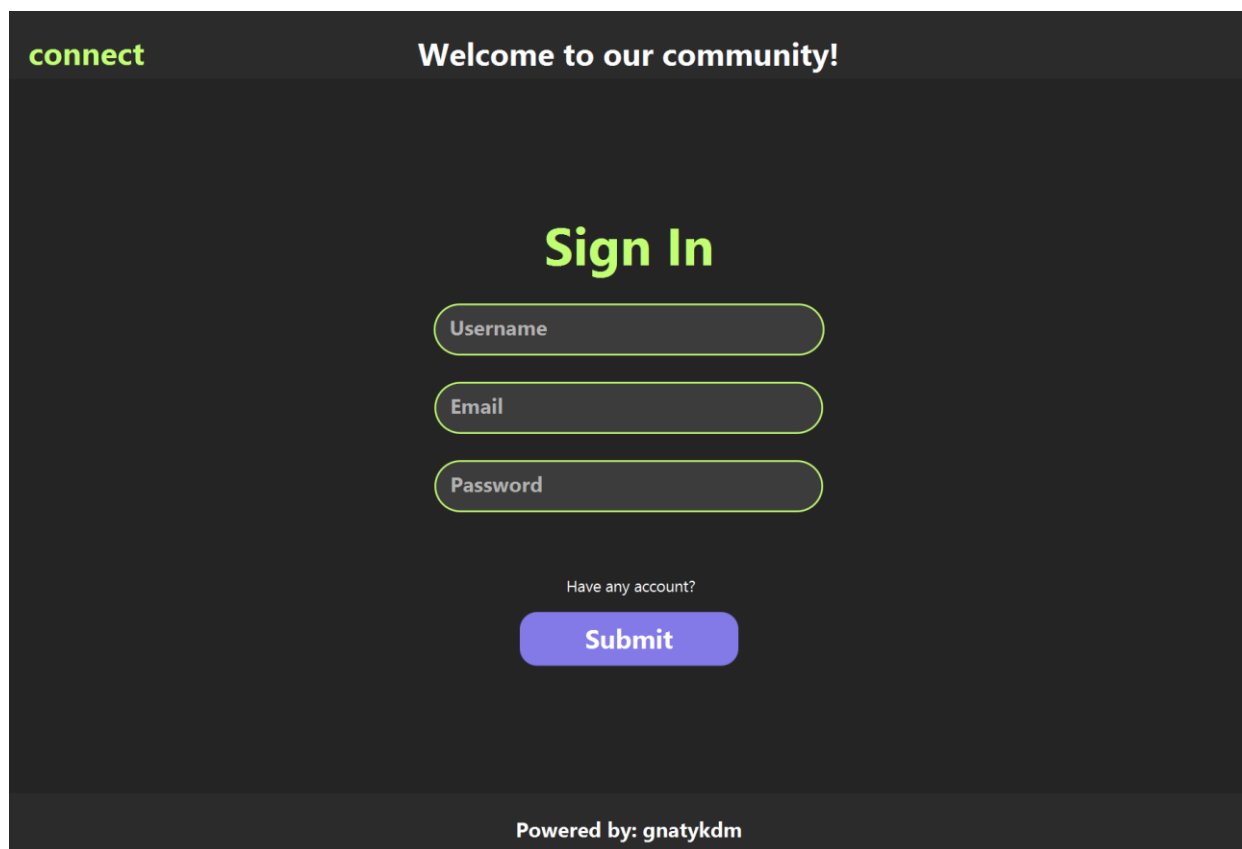
Rysunek 6. Diagram Gantta.

Projekt realizowany był z wykorzystaniem systemu kontroli wersji Git. Wszystkie pliki źródłowe projektu znajdują się pod adresem: <https://github.com/gnatykdm/connect> i będą dostępne do 31.07.2025. Zrzuty ekranu pokazujące historię kommitów znajdują się pod adresem: <https://github.com/gnatykdm/connect/tree/3e8e035e9471838016fc7afa5b79dc622f5fec31/commits>.

## 5. Prezentacja warstwy użytkowej projektu

Na rysunku 6 przedstawiono ekran rejestracji. Z tego miejsca użytkownik może stworzyć nowe konto.

Wymagane jest podanie: imienia użytkownika, adres email oraz hasło. Przyciskiem „Have any account?” użytkownik może cofnąć się do ekranu logowania. Po poprawnym wypełnieniu formularza rejestracji użytkownik zostanie przeniesiony do ekranu głównego ekranu aplikacji.

A dark-themed registration screen. At the top left is the word "connect" in green. At the top center is the text "Welcome to our community!". In the center, the text "Sign In" is displayed in large green font. Below it are three rounded rectangular input fields with green borders, labeled "Username", "Email", and "Password" in white text. Below the input fields is the text "Have any account?" in small white font. Below that is a blue rounded rectangular button with the word "Submit" in white. At the bottom center, the text "Powered by: gnatykd" is displayed in white.

Rysunek 6. Okno rejestracji do systemu.

Na rysunku 7 przedstawiono ekran logowania. Z tego miejsca użytkownik może zalogować się do konta

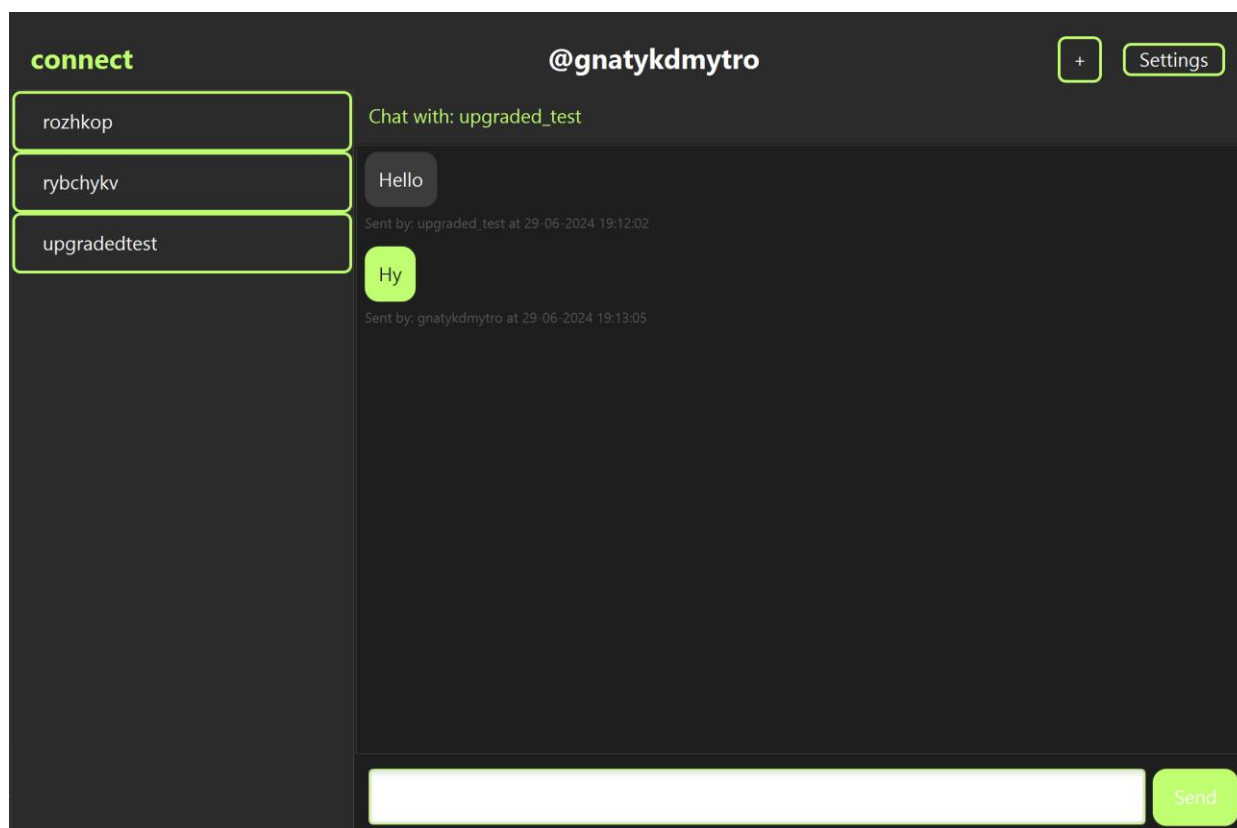
Wymagane jest podanie: imienia użytkownika oraz hasła. Przyciskiem „Do not have any account?” użytkownik może cofnąć się do ekranu rejestracji. Po poprawnym wypełnieniu formularza logowania użytkownik zostanie przeniesiony do głównego ekranu.

The image shows a login interface with a dark background. At the top left is the 'connect' logo in green. At the top center is the text 'Welcome Back!' in white. Below this is a large green 'Log in' heading. Under the heading are two rounded rectangular input fields: the first is labeled 'Username' and the second is labeled 'Password'. Below the input fields is a link that says 'Do not have any account?'. Below the link is a blue 'Submit' button. At the bottom of the interface, centered, is the text 'Powered by: gnatykdM'.

Rysunek 7. Okno logowania do systemu.

Na rysunku 8 przedstawiono główne okno aplikacji po zalogowaniu do systemu, każdy użytkownik biblioteki na poziomie tego okna może:

- Przycisk „Settings” przenosi do panelu ustawień.
- Przycisk „+” przenosi do panelu dodania użytkownika dla komunikacji.
- Po lewej stronie są przyciski dla przełączenia pomiędzy użytkownikami.
- W prawej paneli są widoczne powiadomienia które są przesyłane do użytkownika o zielonym kolorze, powiadomienia od użytkownika o kolorze szarym. Po nimi są następujące informacje: Kto powiadomienie wysłał, godzina o której wysłano.
- W samym dole jest pole dla wpisania tekstu oraz przycisk „Send” dla wysłania.



Rysunek 8. Okno główne aplikacji.

Na rysunku 9 przedstawiono okno Ustawień, w którym można zmienić nazwę, email oraz hasło użytkownika. Znajdują się tam również:

- Przycisk „Logout”, który wyłącza użytkownika i zamyka aplikację.
- Przycisk „Home”, cofa do głównego okna aplikacji.

The screenshot shows a settings window titled "Personal Data" with a dark background. At the top, there are two tabs: "Connect" and "Personal Data", with "Personal Data" being the active tab. A "Home" button is located in the top right corner. The main content area displays three user attributes, each with a "Change" button and a "Change [Attribute]..." button:

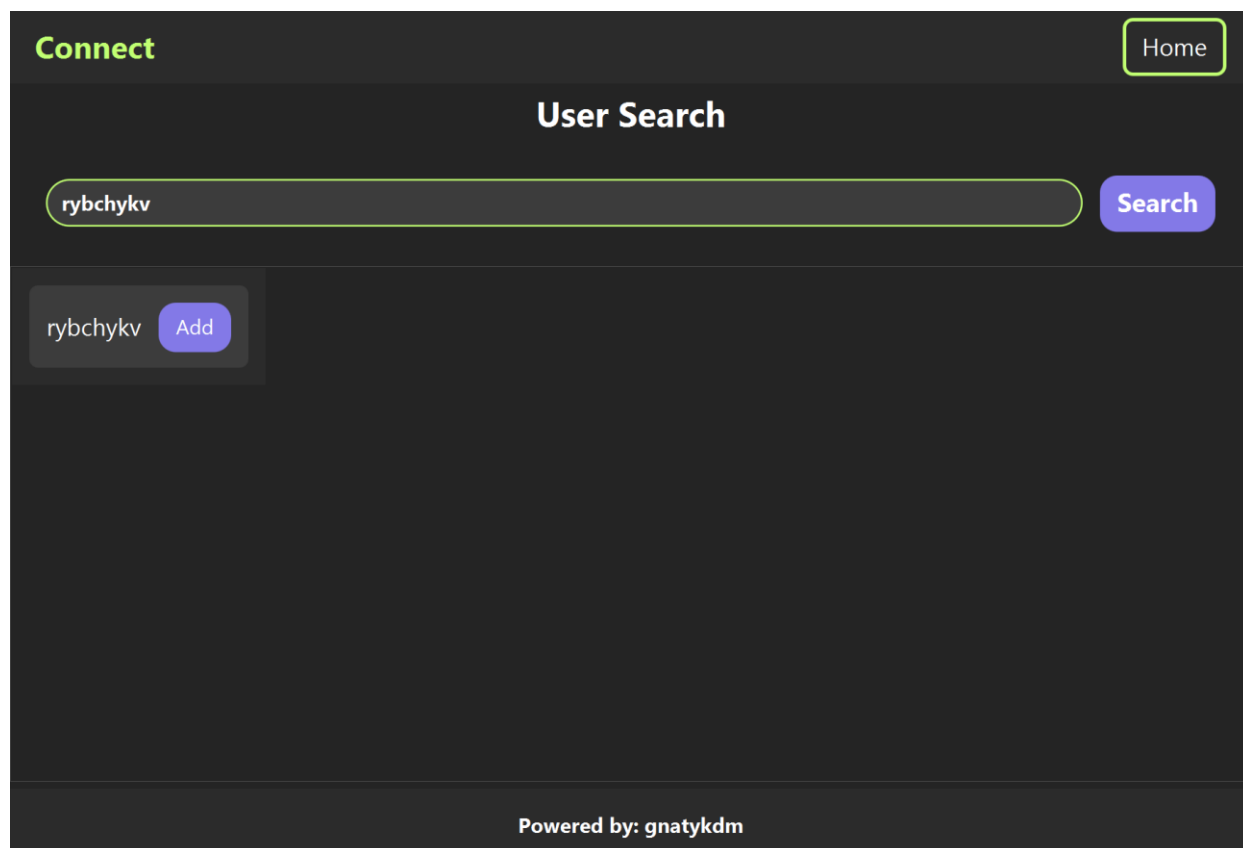
- UserName:** gnatykdmytro (Change Name... button)
- UserEmail:** gnatykwork@gmail.com (Change Email... button)
- UserPassword:** Gnatyk\_2004 (Change Password... button)

A red "Logout" button is positioned to the right of the email field. At the bottom of the screen, it says "Powered by: gnatykdm".

Rysunek 9. Okno ustawień użytkownika.

Na rysunku 10 przedstawiono okno dodawania użytkowników. Użytkownik może znaleźć osobę dla komunikacji po nazwie użytkownika.

- Przycisk „Search ” wyszukuje użytkownika po Imieniu.
- Przyciskiem „Add ” dodaje użytkownika Głównego Ekranu.
- Przycisk „Home” cofa do głównego okna aplikacji.



Rysunek 10. Okno dodawania użytkownika



## **6. Podsumowanie**

W ramach realizacji projektu skupiliśmy się na opracowaniu nowatorskiej aplikacji do komunikacji, której głównym celem jest efektywne zarządzanie i wysyłanie powiadomień. Dotychczasowe prace koncentrowaliśmy się na stworzeniu intuicyjnego interfejsu użytkownika, który zapewnia wygodę i łatwość obsługi.

### **Zrealizowano prace:**

**Analiza wymagań:** Dokonano wszechstronnej analizy zarówno funkcjonalnej, jak i niefunkcjonalnej, aby zidentyfikować kluczowe elementy do wdrożenia.

**Projektowanie struktury:** Stworzono elastyczną architekturę systemu opartą na wzorcu projektowym MVC, z oddzielnymi częściami klienta i serwera, co umożliwia łatwe dostosowanie do, skalowania programu i rosnącej liczby użytkowników.

**Implementacja aplikacji:** Pomyślnie wdrożono funkcje rejestracji, logowania, wysłania powiadomień, dodanie nowych użytkowników, edycji danych.

**Testowanie i optymalizacja:** Przeprowadzono szczegółowe testy, które umożliwiły zidentyfikowanie i usunięcie błędów, a także optymalizację wydajności aplikacji.

### **Planowane dalsze prace rozwojowe:**

**Rozwój funkcjonalności:** Planowane jest dodanie nowych funkcji, takich jak, przesyłanie plików binarnych, możliwość tworzenia grup.

**Udoskonalenie interfejsu:** Dalsze prace nad interfejsem, aby lepiej odpowiadał na aktualne trendy i potrzeby użytkowników.

**Wzmocnienie bezpieczeństwa:** Prace nad zaawansowanym szyfrowaniem danych użytkowników.

### **Podsumowanie końcowe:**

Zrealizowany projekt stanowi istotny krok w kierunku innowacyjnych rozwiązań w zakresie komunikacji. Priorytetem jest ciągły rozwój aplikacji, z uwzględnieniem potrzeb użytkowników oraz utrzymanie wysokiego poziomu funkcjonalności i bezpieczeństwa. Projekt przewiduje regularne aktualizacje i doskonalenie, co zapewni jego długoterminową użyteczność oraz dostosowanie do zmieniających się potrzeb użytkowników.

## 7. Literatura

1. JavaFX. <https://openjfx.io/openjfx-docs/>
2. Java Documentation. <https://docs.oracle.com/en/java/>
3. Spring Framework. <https://spring.io/projects/spring-framework/>
4. Spring Data JPA. <https://spring.io/projects/spring-data-jpa/>
5. Spring Boot. <https://spring.io/projects/spring-boot/>
6. Mockito Testing. <https://www.baeldung.com/mockito-series/>
7. Connection Pooling. <https://www.baeldung.com/java-connection-pooling/>
8. Rest API. <https://www.ibm.com/docs/en/intelligent-promising?topic=reference-rest-api-documentati/>
9. MVC Design Pattern. <https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/>