# Projekt Chatbota

### Wstęp

Niniejsza dokumentacja opisuje strukturę oraz funkcjonalność aplikacji opartej na frameworku **NestJS**, której głównym celem jest stworzenie systemu umożliwiającego użytkownikom rejestrację, logowanie oraz interakcję z zewnętrznym API (Mistral). Aplikacja została zaprojektowana w taki sposób, aby zapewnić pełną funkcjonalność, bezpieczeństwo i wydajność.

Moduły aplikacji zostały zaprojektowane w sposób modułowy, co pozwala na łatwą rozbudowę oraz modyfikację systemu. Każdy moduł odpowiada za konkretną funkcjonalność, taką jak:

- **User** – zarządzanie użytkownikami,
- **Auth** – logowanie oraz autentykacja,
- **Message** – zarządzanie wiadomościami użytkowników,
- **Mistral** – komunikacja z zewnętrznym API bota.

W aplikacji wykorzystano **JWT (JSON Web Tokens)** do zapewnienia bezpieczeństwa i autentykacji użytkowników, a także zintegrowano zewnętrzne API, które umożliwia automatyczne odpowiedzi na wiadomości użytkowników. Aplikacja jest przygotowana na łatwą rozbudowę o nowe funkcje, które mogą być dodawane w przyszłości.

Dokumentacja ta szczegółowo omawia poszczególne moduły, ich strukturę, funkcjonalności oraz wzajemne zależności, a także testy jednostkowe, które zapewniają poprawność działania systemu.

# Moduł User

Moduł **User** odpowiada za zarządzanie użytkownikami w systemie. Wspiera operacje takie jak tworzenie użytkowników, wyszukiwanie użytkowników po nazwie oraz identyfikatorze, a także przechowywanie informacji o użytkownikach w bazie danych.

**Komponenty Modułu User**

1. **User Controller**:
    - Obsługuje żądania HTTP związane z użytkownikami, np. rejestrację, edytowanie danych użytkownika.
2. **User Entity**:
    - Definiuje strukturę tabeli użytkowników w bazie danych. Zawiera atrybuty takie jak: identyfikator, nazwa użytkownika, hasło, relacja z tabelą **Chat**.
3. **User Module**:
    - Grupa wszystkich komponentów związanych z użytkownikami. Importuje **TypeOrmModule** do pracy z bazą danych.
4. **User Service**:
    - Zawiera logikę biznesową dla operacji na użytkownikach, takich jak tworzenie użytkowników, wyszukiwanie ich po nazwie i ID.

**Funkcjonalności**

- Tworzenie nowych użytkowników.
- Wyszukiwanie użytkowników po nazwie lub ID.
- Relacja z czatami utworzonymi przez użytkowników.

# Moduł Auth

Moduł **Auth** zapewnia mechanizm logowania oraz rejestracji użytkowników. Wykorzystuje tokeny JWT do autentykacji i ochrony dostępu do chronionych zasobów w systemie.

**Komponenty Modułu Auth**

1. **Auth Controller**:
    - Obsługuje żądania HTTP związane z logowaniem i rejestracją użytkowników.
2. **Auth Service**:
    - Odpowiada za logikę rejestracji, logowania oraz weryfikacji użytkowników.
3. **JwtAuthGuard**:
    - Chroni dostęp do chronionych tras, sprawdzając poprawność tokenu JWT.
4. **Jwt Strategy**:
    - Definiuje strategię używaną do weryfikacji tokenu JWT.

**Funkcjonalności**

- Rejestracja użytkowników.
- Logowanie użytkowników i generowanie tokenów JWT.
- Walidacja tokenów JWT w celu zapewnienia bezpiecznego dostępu do zasobów.

# Moduł Message

Moduł **Message** odpowiada za zarządzanie wiadomościami w systemie. Umożliwia użytkownikom wysyłanie i odbieranie wiadomości w ramach czatów. Zawiera także interakcje z botem, który odpowiada na wiadomości użytkowników.

**Komponenty Modułu Message**

1. **Message Controller**:
    - Obsługuje żądania HTTP związane z tworzeniem nowych wiadomości.
2. **Message Entity**:
    - Zawiera strukturę tabeli wiadomości w bazie danych. Przechowuje informacje o treści wiadomości, identyfikatorze czatu oraz autorze wiadomości.
3. **Message Service**:
    - Odpowiada za logikę tworzenia nowych wiadomości, w tym komunikację z botem, który odpowiada na wiadomości użytkowników.

**Funkcjonalności**

- Tworzenie wiadomości przez użytkowników.
- Automatyczne odpowiedzi bota na wiadomości użytkowników.
- Powiązanie wiadomości z odpowiednimi czatami.

# Moduł Mistral

Moduł **Mistral** integruje aplikację z API Mistral, które umożliwia generowanie odpowiedzi bota na wiadomości użytkowników. Moduł wykorzystuje specjalistyczną usługę do komunikacji z API.

**Komponenty Modułu Mistral**

1. **Mistral Controller**:
    - Obsługuje żądania związane z komunikacją z API Mistral. Może zostać użyty do testowania odpowiedzi generowanych przez API.
2. **Mistral Service**:
    - Komunikuje się z API Mistral i zarządza generowaniem odpowiedzi na podstawie historii wiadomości.
3. **MistralModule**:

- Moduł, który grupuje **MistralController** i **MistralService** w jeden logiczny blok. Dzięki niemu możliwe jest łatwe zarządzanie tymi komponentami oraz ich zależnościami.

**Funkcjonalności**

- Wysyłanie zapytań do API Mistral w celu uzyskania odpowiedzi na podstawie rozmowy.
- Obsługuje mechanizm ponawiania zapytań w przypadku problemów z dostępem do API (np. limit przekroczenia).

# Zależności Pomiędzy Modułami

Moduły w aplikacji są ze sobą powiązane w sposób umożliwiający pełną interakcję między użytkownikami, wiadomościami, czatami oraz odpowiedziami generowanymi przez bota. Moduł **User** współpracuje z modułami **Auth** i **Message**, zapewniając dostęp do danych użytkowników i ich wiadomości. Moduł **Message** łączy się z modułem **Mistral**, aby dostarczyć automatyczne odpowiedzi na wiadomości użytkowników.