1. Opis klas i metod

Modele danych

- **AppUser** - klasa reprezentująca użytkownika systemu, dziedzicząca po `IdentityUser`
- **Category** - model kategorii kontaktów
- **ContactModel** - główny model reprezentujący kontakt
- **SubCategory** - model podkategorii kontaktów

StrongPasswordAttribute

- Klasa atrybutu do walidacji złożoności hasła
- Metody:
  - `IsValid()` - sprawdza czy hasło spełnia wymagania (długość ≥8, wielkie i małe litery, cyfry, znaki specjalne)
  - `FormatErrorMessage()` - zwraca komunikat błędu

QueryObject

- Klasa pomocnicza do filtrowania kontaktów
- Właściwości:
  - `LastName` - filtrowanie po nazwisku
  - `PhoneNumber` - filtrowanie po numerze telefonu

Interfejsy

- **ITokenService**
  - Interfejs serwisu tokenów JWT
  - Metody:
    - `CreateToken()` - generowanie tokenu dla użytkownika

- ***IContactRepository***
  - Interfejs repozytorium kontaktów
  - Metody:
    - `GetAllAsync()` - pobieranie wszystkich kontaktów (z filtrowaniem i bez)

        - `GetByIdAsync()` - pobieranie pojedynczego kontaktu
        - `CreateAsync()` - tworzenie kontaktu
        - `UpdateAsync()` - aktualizacja kontaktu
        - `DeleteAsync()` - usuwanie kontaktu
        - `ExistsByEmail()` - sprawdzanie istnienia emaila

Klasy bazodanowe

ApplicationDBContext

- Kontekst bazy danych dziedziczący po `IdentityDbContext`
- Konfiguracja:
    - Definicja ról (Administrator, User)
    - Relacje między modelami
    - Unikalne indeksy
- DbSety:
    - `Contacts`
    - `Categories`
    - `SubCategories`

Klasy mapujące

MappingContact

- Statyczna klasa z metodami rozszerzającymi do mapowania
- Metody:
    - `ToContactFromCreateDto()` - mapowanie z DTO na model
    - `ToContactDetailDto()` - mapowanie na DTO szczegółów
    - `ToContactListDto()` - mapowanie na DTO listy

Klasy DTO

- **ContactCreateAndUpdateDto** - Klasa służąca do tworzenia i aktualizacji kontaktów
- **ContactListDto** - Klasa reprezentująca uproszczoną listę kontaktów
- **ContactDetailDto** - Klasa zawierająca szczegółowe informacje o kontakcie
- **LoginDto** - Klasa do obsługi logowania

- **RegisterDto** - Klasa do obsługi rejestracji użytkowników
- **NewUserDto** - Klasa zwracająca informacje o nowo utworzonym użytkowniku

Kontrolery

- **AccountController** - obsługa operacji związanych z kontem użytkownika
  - `Login()` - logowanie użytkownika
  - `Register()` - rejestracja nowego użytkownika
- **ContactController** - zarządzanie kontaktami
  - `GetAll()` - pobieranie wszystkich kontaktów
  - `GetById()` - pobieranie kontaktu po ID
  - `Create()` - tworzenie nowego kontaktu
  - `Update()` - aktualizacja kontaktu
  - `Delete()` - usuwanie kontaktu
  - `GetAllWithQuery()` - filtrowane pobieranie kontaktów

Serwisy

- **TokenService** - generowanie tokenów JWT
  - `CreateToken()` - tworzenie tokenu dla użytkownika

Repozytoria

- **ContactRepository** - operacje CRUD na kontaktach
  - `CreateAsync()` - dodawanie kontaktu
  - `DeleteAsync()` - usuwanie kontaktu
  - `GetByIdAsync()` - pobieranie po ID
  - `UpdateAsync()` - aktualizacja kontaktu
  - `GetAllAsync()` - pobieranie wszystkich/filtrowanych kontaktów
  - `ExistsByEmail()` - sprawdzanie istnienia emaila
  -

2. Wykorzystane biblioteki

Microsoft.AspNetCore

- Identity - zarządzanie użytkownikami
- Authentication.JwtBearer - autentykacja JWT
- EntityFrameworkCore - ORM do bazy danych

Inne pakiety

- Microsoft.IdentityModel.Tokens - obsługa tokenów JWT
- System.IdentityModel.Tokens.Jwt - generowanie tokenów JWT
- Microsoft.OpenApi - dokumentacja Swagger
- System.ComponentModel.DataAnnotations - atrybuty walidacyjne
- System.Text.RegularExpressions - wyrażenia regularne do walidacji

3.Uruchamianie

1 .Backend (Kontakty.Server)

Kompilacja i uruchomienie:

- Wejdź do folderu:
  cd kontakty.Server
- Zbuduj aplikację:
  dotnet build
- Uruchom aplikację:
  dotnet wattch run

aplikacja .NET będzie działać na http://localhost:5274/

2.Frontend (Kontakty.Client)

- Wejdź do katalogu:
  cd kontakty.Client
- Zainstaluj zależności (jeśli jeszcze nie zrobione):
  npm install
- Zbuduj aplikację:
  npm run dev

aplikacja dostępna pod portem: http://localhost:5173/