Nazwa i akronim projektu: Aplikacja wspomagająca zakładanie kont na serwerze baz danych.	**Zleceniodawca:** WFTiMS	**Zleceniobiorca:** PG, WFTiMS, zespół projektowy IO nr 1
Numer zlecenia: PG-WFTiMS-IO-2022-1	**Kierownik projektu:** Zofia Zinkowska	**Opiekun projektu:** *dr inż. Bartosz Reichel*

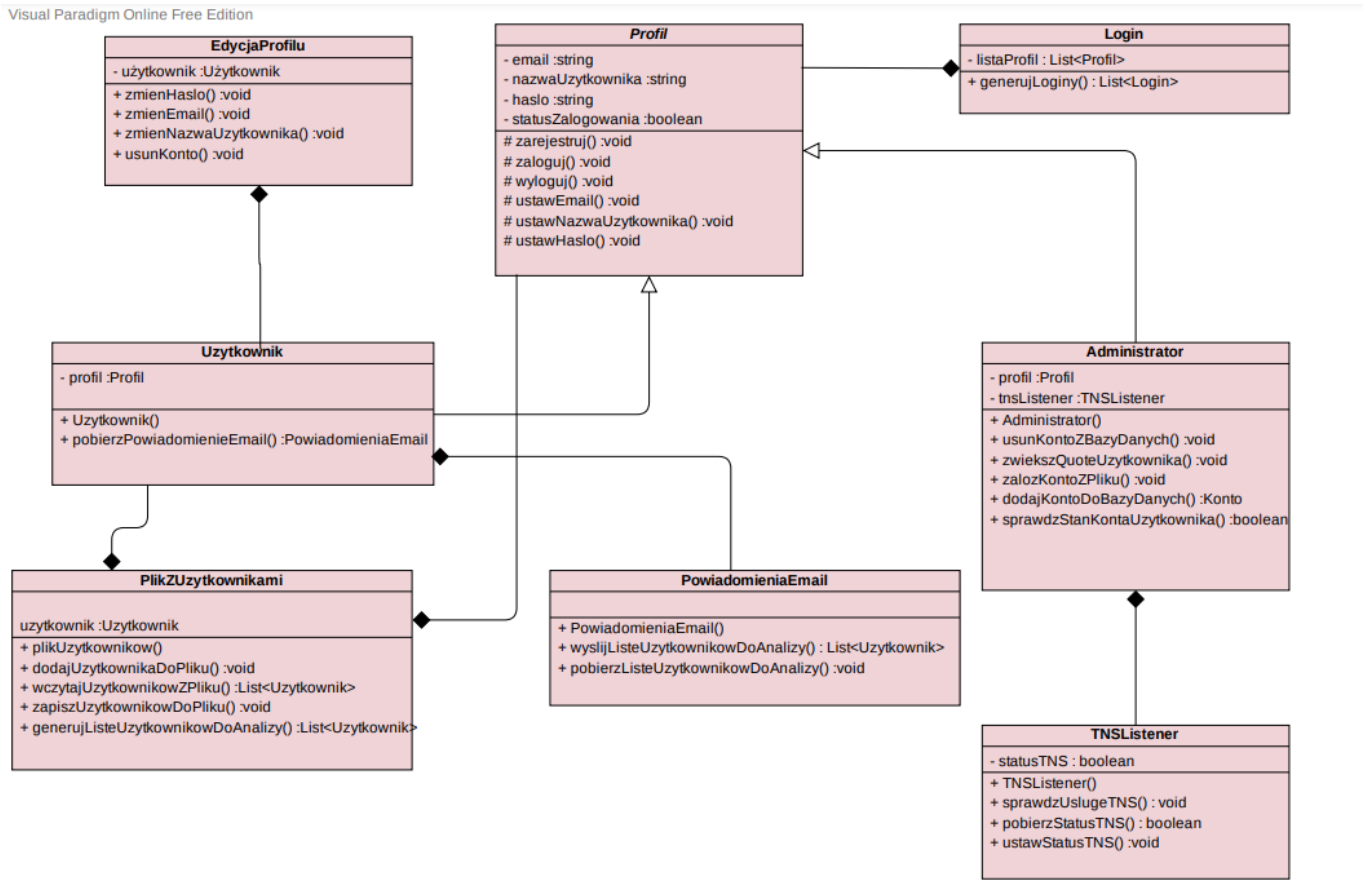
Diagram klas i jego opis	**Nr wersji:** 1
Odpowiedzialny za dokument: Zofia Zinkowska	**Data pierwszego sporządzenia:** 20.04.2022r.
	Data ostatniej aktualizacji: 20.04.2022r.

Historia dokumentu

Wersja	Opis modyfikacji	Rozdział / strona	Autor modyfikacji	Data
1	Wersja wstępna	całość	Zofia Zinkowska, Karol Majchrzak, Adam Kaczmarkiewicz, Beniamin Tyński	20.04.2022r.

1 Model obiektowy systemu

1.1. Diagram(y) klas



KLASA	**Edycja profilu**
OPIS KLASY	Klasa Edycja profilu odpowiada za funkcje służące do edytowania profilu użytkownika.
ATRYBUTY	• **uzytkownik :Uzytkownik** – klasa przyjmuje pole użytkownik w celu operacji mających na celu zmiany właściwości danych użytkownika.
METODY	• **zmienHaslo() :void** – metoda służy do zmiany hasła. • **zmienEmail() :void** – metoda służy do zmiany maila. • **zmienNazwaUzytkownika() :void** – metoda służy do zmiany nazwy użytkownika. • **usunKonto() :void** – użytkownik będzie miał możliwość usunięcia konta.
ZWIĄZKI	- Uzytkownik - kompozycja

KLASA	**Profil**
OPIS KLASY	Klasa Profil jest klasą abstrakcyjną i powstaje w celu uniknięcia powtórzeń pól i metod w klasach Uzytkownik i Administrator.
ATRYBUTY	• **email :string –** przechowuje email. • **nazwaUzytkownika :string –** przechowuje nazwę użytkownika. • **haslo :string –** przechowuje hasło. • **statusZalogowania :boolean –** ustawia status zalogowania (przechowuje status zalogowania).
METODY	• **zarejestruj() :void –** metoda służy do rejestracji. • **zaloguj() :void –** metoda służy do logowania. • **wyloguj() :void –** metoda służy do wylogowania. • **ustawEmail() :void –** metoda ustawia email. • **ustawNazwaUzytkownika() :void –** metoda ustawia nazwę użytkownika. • **ustawHaslo() :void –** metoda ustawia hasło.
ZWIĄZKI	- Login - Administrator - Uzytkownik - PlikZUzytkownikami

KLASA	**Login**
OPIS KLASY	Klasa Login służy do odebrania listy profili i wygenerowania z nich loginów.
ATRYBUTY	• **listaProfil :List<Profil> -** przechowuje kolekcję lista, w której każdy indeks jest obiektem typu profil.
METODY	• **generujeLoginy() :List<Login> -** metoda służy do generowania loginów.
ZWIĄZKI	- Profil

KLASA	**Użytkownik**
OPIS KLASY	Klasa Użytkownik dziedziczy elementy klasy abstrakcyjnej Profil oraz przechowuje dane konta użytkownika, pozwala na wykonywanie operacji z klasy Profil.
ATRYBUTY	• **profil :Profil –** atrybut służy do możliwości odwoływania się do klasy Profil.
METODY	• **Uzytkownik() –** konstruktor. • **pobierzPowiadomienieEmail() :PowiadomienieEmail –** metoda służy do pobierania powiadomień email z klasy PowiadomienieEmail.
ZWIĄZKI	- EdycjaProfilu - Profil - PlikZUzytkownikami - PowiadomieniaEmail

KLASA	**PlikZUzytkownikami**
OPIS KLASY	Klasa PlikZUzytkownikami odpowiada za operacje wykonywane na plikach.
ATRYBUTY	• **uzytkownik: Uzytkownik –** klasa przyjmuje pole użytkownik w celu operacji mających na celu zmiany właściwości danych użytkownika.
METODY	• **PlikZUzytkownikami() –** konstruktor. • **wczytajUzytkownikowZPliku() :List<Uzytkownik> -** metoda służy do wczytania listy użytkowników z pliku. • **zapiszUzytkownikowDoPliku() :void –** metoda służy do zapisu listy użytkowników do pliku. • **generujeListeUzytkownikowDoAnalizy() :List<Uzytkownik>** - metoda służy do generowania listy użytkowników, która posłuży administratorowi do analizy.
ZWIĄZKI	- Uzytkownik - Profil

KLASA	**Administrator**
OPIS KLASY	Klasa Administrator odpowiada za operacje, które będzie mógł wykonywać administrator.
ATRYBUTY	• **profil :Profil –** atrybut służy do możliwości odwoływania się do klasy Profil. • **tnsListener :TNSListener** – administrator będzie miał możliwość korzystania z metod oraz atrybutów tej klasy.
METODY	• **Administrator() –** konstruktor. • **usunKontoZBazyDanych() :void –** metoda służy do usuwania konta z bazy danych. • **zwiekszQuoteUzytkownika() :void –** metoda służy do zwiększania quote użytkownika. • **zalozKontoZPliku() :void –** metoda służy do zakładania konta użytkownika z pliku. • **dodajKontoDoBazyDanych() :Konto –** metoda służy do dodawania konta do bazy danych. • **sprawdzStanKontaUzytkownika() :boolean –** metoda służy do sprawdzania stanu konta użytkownika.
ZWIĄZKI	- TNSListener - Profil

KLASA	**PowiadomieniaEmail**
OPIS KLASY	Klasa PowiadomieniaEmail służy do obsługi operacji związanych z pracą na emailach.
ATRYBUTY	
METODY	• **PowiadomieniaEmail() –** konstruktor. • **wyslijListeUzytkownikowDoAnalizy() :List<Uzytkownik> -** metoda służy do wysyłania listy użytkowników do analizy. • **pobierzListeUzytkownikowDoAnalizy() :void –** metoda służy do pobierania listy użytkowników do analizy.
ZWIĄZKI	- Uzytkownik

KLASA	**TNSListener**
OPIS KLASY	Klasa TNSListener służy do operacji związanych z podsłuchaniem usługi TNS, aby administrator mógł ją kontrolować.
ATRYBUTY	• **statusTNS : boolean –** przechowuje status TNS.
METODY	• **TNSListener() –** konstruktor. • **sprawdzUslugeTNS() :void –** metoda służy do sprawdzania usługi TNS. • **pobierzStatusTNS() :boolean –** metoda służy do pobierania statusu TNS. • **ustawStatusTNS() :void –** metoda służy do ustawiania statusu TNS.
ZWIĄZKI	- Administrator