Temat projektu: Komunikator Internetowy

Skład grupy: Andrzej Świderski, Bartosz Prusaczyk, Tomasz Milianowicz, Bartosz Giełdon, Maksymilian Głowacki

Opis projektu:

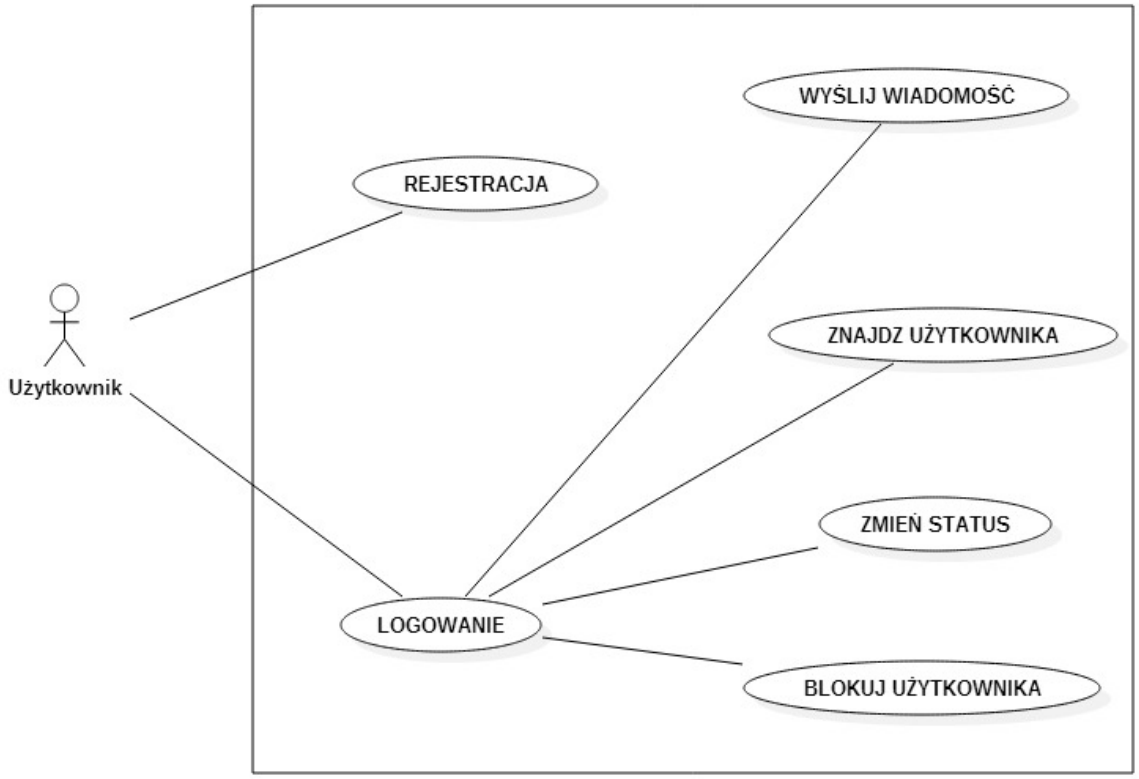
Aplikacja na komputery osobiste i urządzenia mobilne z systemem android przeznaczona do komunikacji z innymi użytkownikami.

1. **Założenia funkcjonalne:**
   - Użytkownik ma możliwość założenia konta,
   - wyszukiwanie użytkowników po nazwie konta (nick),
   - wysyłanie wiadomości tekstowych,
   - wyświetlanie statusu użytkownika (np. dostępny, zajęty, niedostępny),
   - wyświetlanie daty i godziny wysłanych wiadomości,
   - blokowanie użytkowników,

2. **Założenia niefunkcjonalne:**

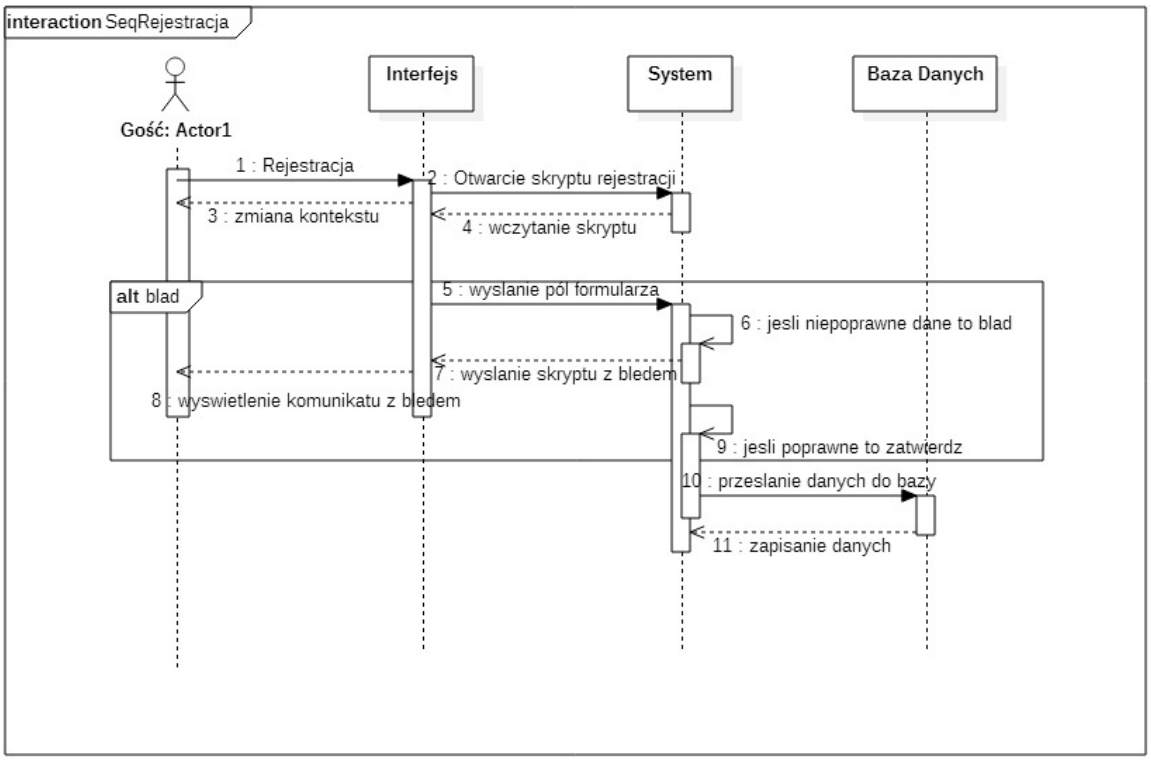
a) Użyteczność – Widok wiadomości będzie wyświetlany od góry w dół (na dole będą znajdować się najnowsze wiadomości), wybór użytkownika z wyświetloną nazwą i statusem dostępności.
b) Niezawodność - Czas awarii nie może być dłuższy niż 3 godziny. Czas wysyłania wiadomości nie może być dłuższy niż 3 minuty.
c) Wydajność - Czas wysyłania wiadomości nie może być dłuższy niż 3 minuty. Czas zmiany statusu nie może być dłuższy niż 5 minut. Czas logowania do aplikacji nie może trwać dłużej niż 3 minuty.
d) Wspieralność – Aplikacja na telefon będzie obsługiwana przez system Android KitKat 4.4 lub nowszy, a aplikacja na komputery osobiste będzie wspierana i obsługiwana przez system Windows 7 lub nowszy.

3. Diagram przypadków użycia:

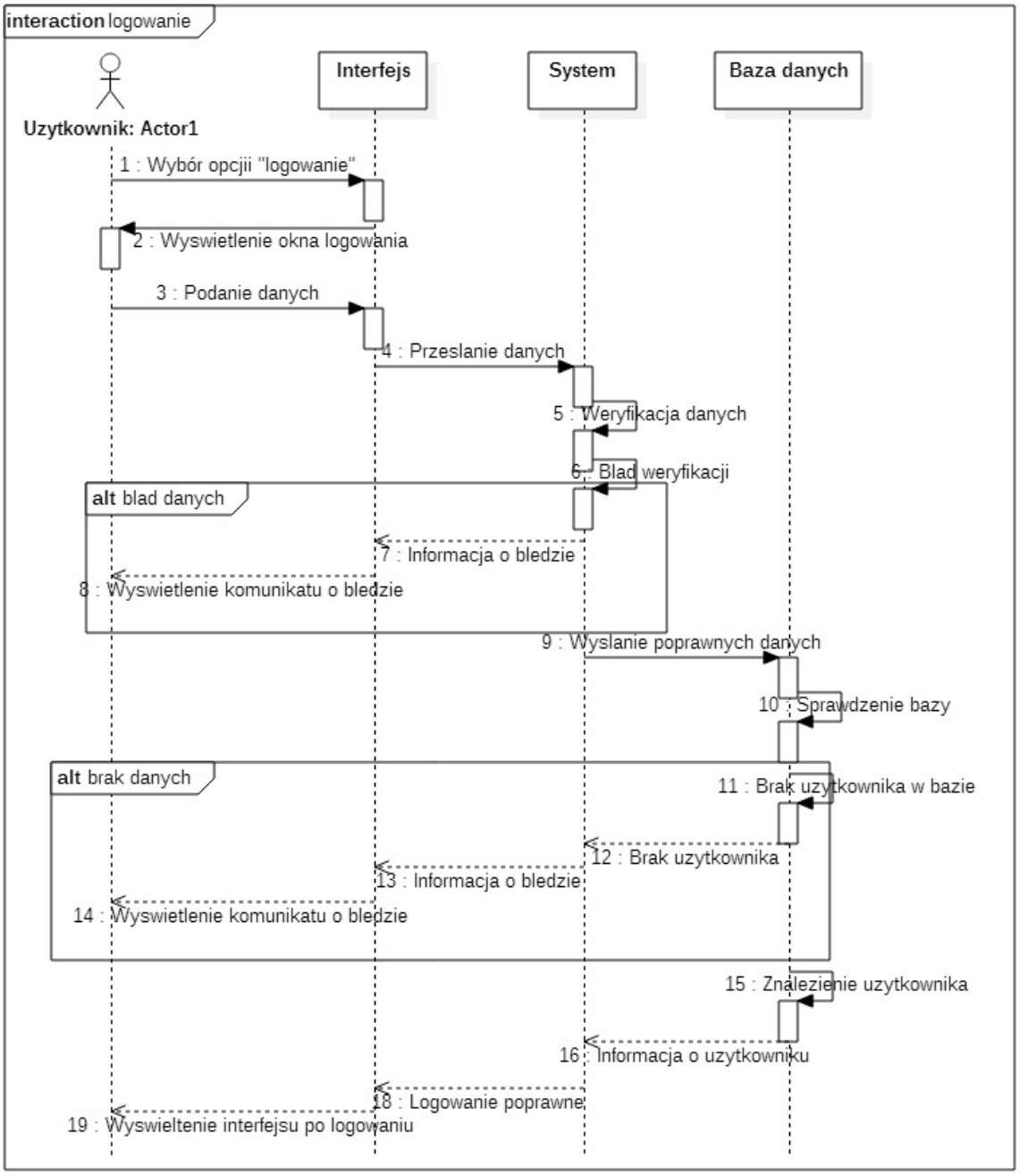


4. Diagramy sekwencji:

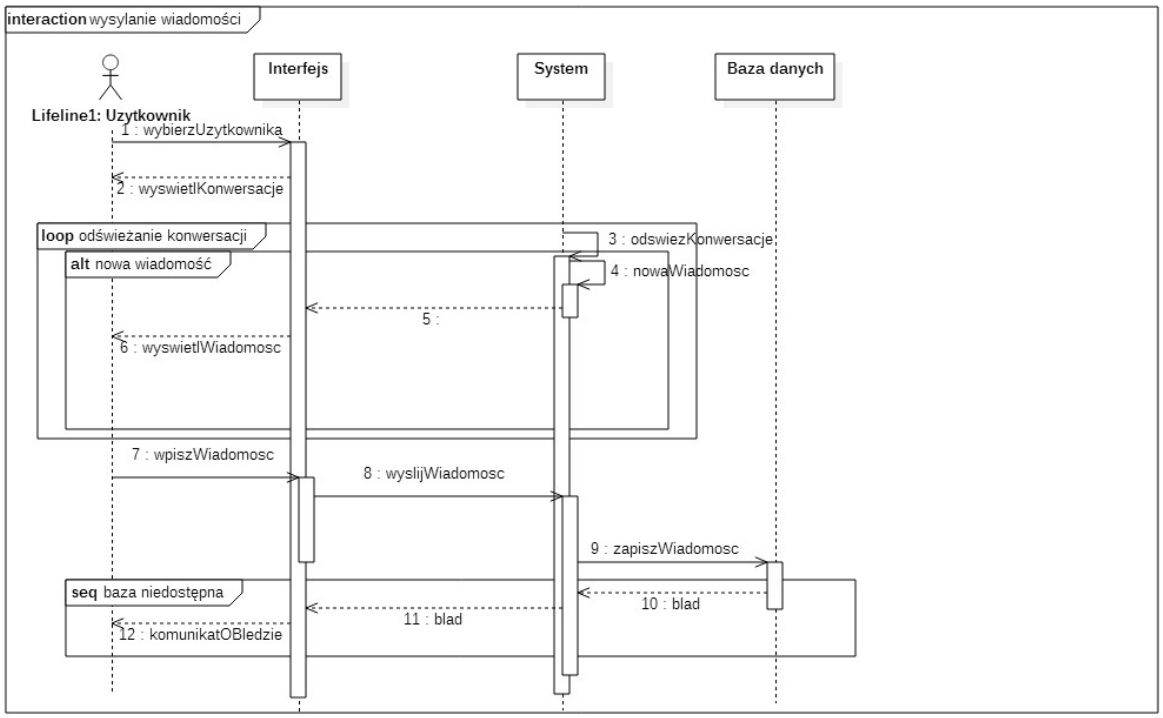
a) Rejestracja



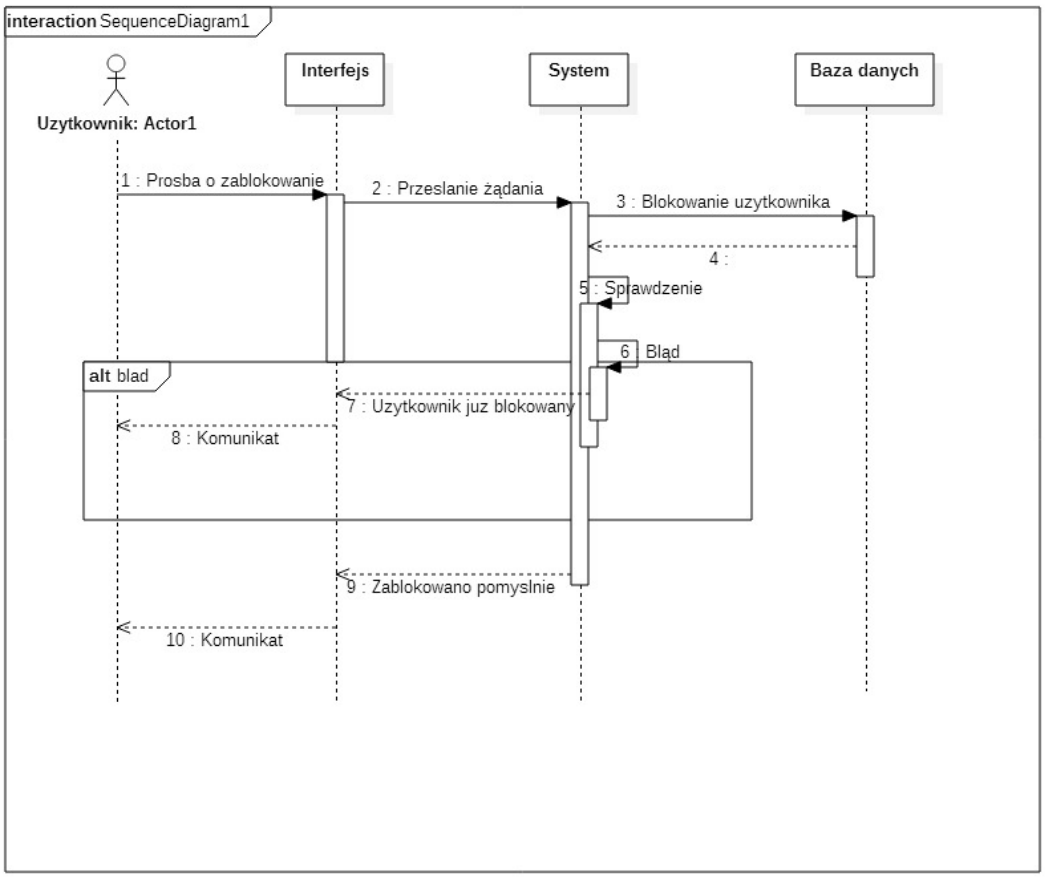
b) Logowanie


interaction logowanie
Interfejs
System
Baza danych
Uzytkownik: Actor1
1 : Wybór opcjii "logowanie"
2 : Wyswietlenie okna logowania
3 : Podanie danych
4 : Przeslanie danych
5 : Weryfikacja danych
6 : Blad weryfikacji
alt blad danych
7 : Informacja o bledzie
8 : Wyswietlenie komunikatu o bledzie
9 : Wyslanie poprawnych danych
10 : Sprawdzenie bazy
alt brak danych
11 : Brak uzytkownika w bazie
12 : Brak uzytkownika
13 : Informacja o bledzie
14 : Wyswietlenie komunikatu o bledzie
15 : Znalezienie uzytkownika
16 : Informacja o uzytkowniku
18 : Logowanie poprawne
19 : Wyswieltenie interfejsu po logowaniu

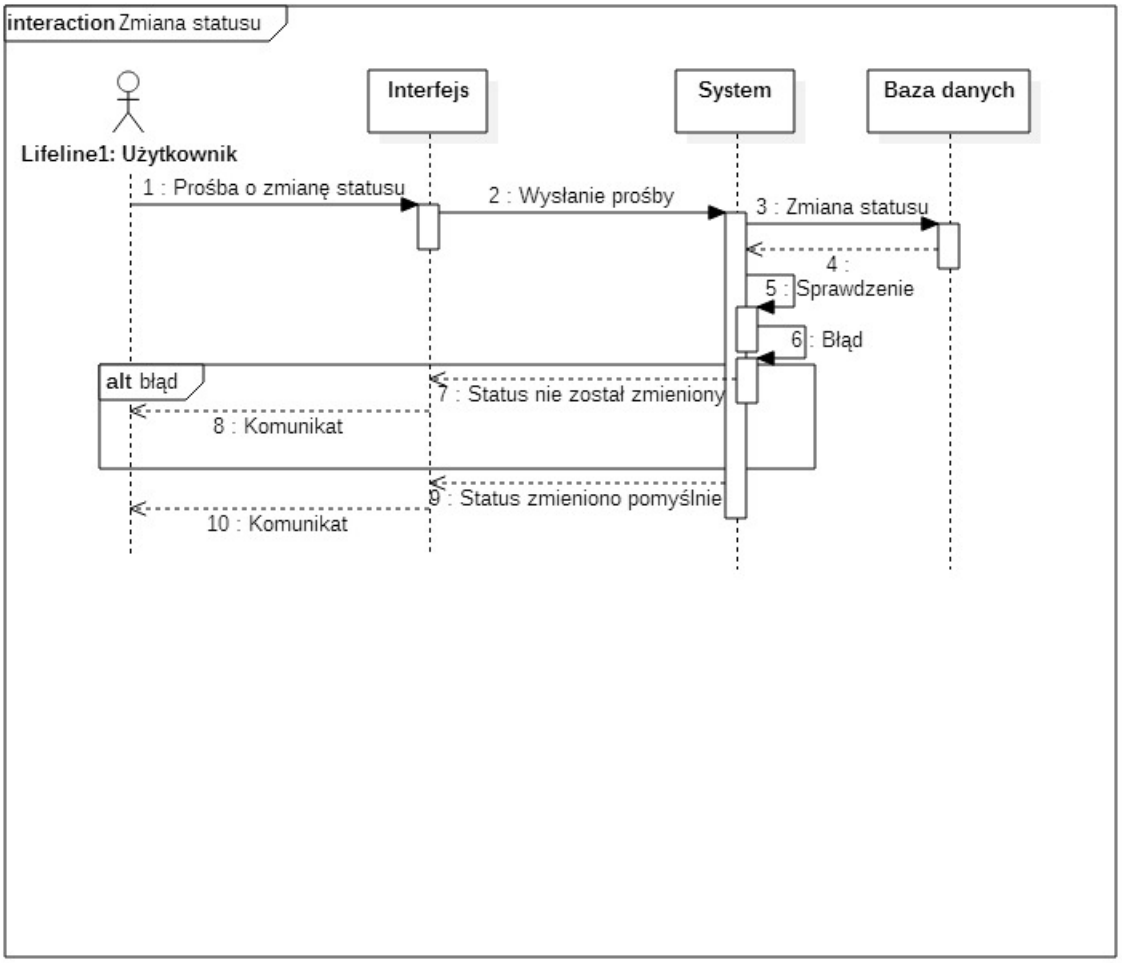
c) Wysyłanie wiadomości



d) Blokowanie użytkownika



e) Zmiana statusu



5. Technologie i metodyka:

Środowiska programistyczne użyte w projekcie:

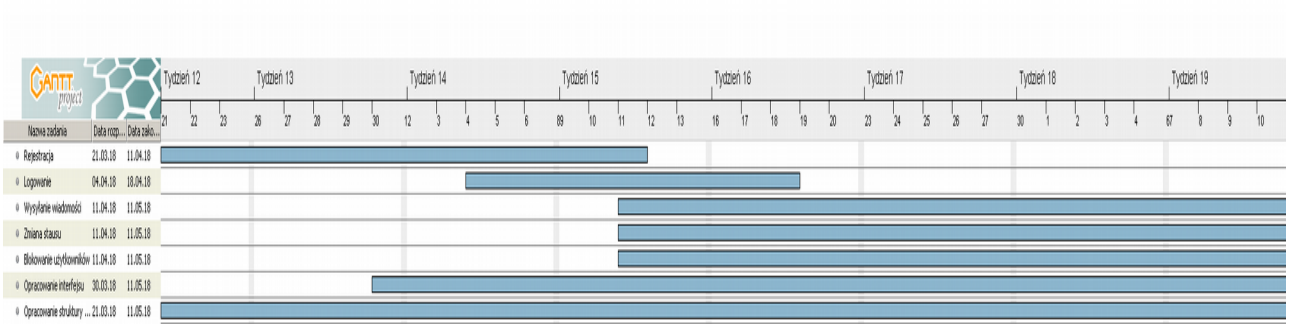
Visual studio 2015 – środowisko te posłuży nam do stworzenia aplikacji desktopowej w języku C#

Android Studio - środowisko te posłuży nam do stworzenia aplikacji desktopowej w języku JAVA

StarUML - program ten posłuży nam do stworzenia diagramów w języku UML.

GanttProject – program do wykonania harmonogramu zadań.

6. **Podział zadań i harmonogram**



Rejestracja (Andrzej Świderski) – 21.03 – 11.04
Logowanie (Maksymilian Głowacki) – 4.04 – 18.04
Wysyłanie wiadomości (Bartosz Prusaczyk) – 11.04 – 11.05
Zmiana statusu (Bartosz Giełdon) – 11.04 – 11.05
Blokowanie użytkowników (Tomasz Milianowicz) – 11.04-11.05
Opracowanie interfejsu (wszyscy) – 30.03 – 11.05
Opracowanie struktury bazy danych (wszyscy) -21.03 – 11.05

7. Opis kodu

Aplikacja desktopowa

Klasa Konwersacja

Przestrzeń nazw: komunikator.WysylanieWiadomosci

Klasa reprezentująca konwersację pomiędzy użytkownikami.

Pola:

public string ***login*** – login zalogowanego użytkownika.

public string ***adresat*** – login użytkownika, z którym jest prowadzona konwersacja.

private const string ***daneBazy*** – dane serwera MySQL.

Konstruktory:

public ***Konwersacja****(string login, string adresat)* – tworzy egzemplarz klasy.

Argumenty:

login – login zalogowanego użytkownika.

adresat – login użytkownika, z którym prowadzona jest konwersacja.

Metody:

private static string ***znajdzIdUzytkownika****(string login)* – zwraca ID w bazie danych danego użytkownika.

Argumenty:

login – login użytkownika, którego ID ma zostać znalezione.

private static string ***znajdzUzytkownikaPoId****(string id)* – zwraca login użytkownika, do którego jest przypisane dane ID w bazie danych.

Argumenty:

id – ID w bazie danych, do którego jest przypisany szukany użytkownik.

public string ***wyslijWiadomosc****(string tresc)* – wysyła wiadomość do adresata. Zwraca czas serwera, w którym wiadomość została dostarczona.

Argumenty:

tresc – treść wysyłanej wiadomości.

public List<Wiadomosc> ***wczytajWiadomosci****()* - zwraca wszystkie wiadomości, które zostały wysłane pomiędzy zalogowanym użytkownikiem i adresatem.

public bool ***sprawdzCzySaNoweWiadomosci****()* - zwraca **true**, jeśli zalogowany użytkownik ma nieodczytane wiadomości, w przeciwnym przypadku **false**.

public static bool ***sprawdzCzySaNoweWiadomosci****(string login)* – zwraca **true**, jeśli dany użytkownik ma nieodczytane wiadomości, w przeciwnym przypadku **false**.

Argumenty:

login – login użytkownika, którego wiadomości są sprawdzane.

public List<Wiadomosc> ***odswiezKonwersacje****()* - zwraca nieodczytane wiadomości od adresata.

public static bool ***znajdzUzytkownika****(string szukanyLogin)* – zwraca **true**, jeśli użytkownik o podanym loginie istnieje, w przeciwnym wypadku **false**.

Argumenty:

szukanyLogin – login szukanego uzytkownika.

public static void ***dodajKontakt****(string login1, string login2)* – dodaje nowy kontakt.

Argumenty:

login1, ***login2*** – loginy użytkowników z nowego kontaktu.

public static List<string> ***zaladujKontakty****(string login)* – zwraca kontakty danego użytkownika.

Argumenty:

login – login użytkownika, którego kontakty mają być zwrócone.

public static void ***usunKontakt****(string login1, string login2)* – usuwa kontakt.

login1, ***login2*** – loginy użytkowników z kontaktu, który ma być usunięty.

public static Dictionary<string, int> ***wyswietlPowiadomieniaONowychWiadomosciach****(string login)* – zwraca odwzorowanie zawierające kontakty danego użytkownika i liczbę odpowiadających im nieodczytanych wiadomości (zwraca tylko kontakty, od których są nieodczytane wiadomości)

Argumenty:

login – login użytkownika, którego liczba nieodczytanych wiadomości ma być zwrócona.

Klasa Konwersacja.Wiadomosc

Przestrzeń nazw: komunikator.WysylanieWiadomosci

Reprezentuje wiadomość.

Właściwości:

public string ***tresc*** – treść wiadomości.

public string ***uzytkownik*** – użytkownik, który wysłał wiadomość.

public string ***data*** – data i czas wysłania wiadomości.

Klasa Konwersacja.Kontakt

Przestrzeń nazw: komunikator.WysylanieWiadomosci

Reprezentuje kontakt.

Właściwości:

public string ***login*** – login użytkownika, do którego należy kontakt.

public int ***nieodczytaneWiadomosci*** – liczba nieodczytanych wiadomości, od danego kontaktu.

Metody:

public override string ***ToString****()* - zwraca login użytkownika, do którego należy kontakt, jeśli nie ma od tego kontaktu żadnych nieodczytanych wiadomości, w przeciwnym przypadku zwraca login użytkownika, do którego należy kontakt, i liczbę nieodczytanych wiadomości w nawiasie.

Klasa KonwersacjaOkno

Przestrzeń nazw: komunikator.WysylanieWiadomosci

Dziedziczy po: ***Window***

Reprezentuje okno, w którym jest wyświetlona konwersacja.

Pola:

*private Konwersacja **k*** – konwersacja, która jest wyświetlana.

*private Timer **odswiezacz*** – obiekt odpowiedzialny za odświeżanie konwersacji

Konstruktory:

*public **KonwersacjaOkno**(string loginZalogowanego, string loginAdresata)* – tworzy nowe okno.

Argumenty:

loginZalogowanego – login zalogowanego użytkownika.

loginAdresata – login użytkownika, z którym odbywa się konwersacja.

Metody:

*private void **OnOdswiezEvent**(object o)* – jeśli to konieczne, odświeża konwersację.

Argumenty:

o – konwersacja, którą należy odświeżyć.

*private void **dodajNoweWiadomosci**()* - wyświetla nadesłane wiadomości.

Metody powiązane ze zdarzeniami:

*private void **wyslijPrzycisk_Click**(object sender, RoutedEventArgs e)* – wysyła wpisaną wiadomość, aktywowane po kliknięciu na przycisk do wysyłania wiadomości.

*private void **wiadomoscTekst_KeyUp**(object sender, KeyEventArgs e)* - wysyła wpisaną wiadomość, aktywowane po wciśnięciu klawisza Enter.

Klasa WyborRozmowcy

Przestrzeń nazw: komunikator.WysylanieWiadomosci

Dziedziczy po: ***Window***

Reprezentuje okno, w którym są wyświetlone kontakty.

Pola:

*private string **zalogowanyUzytkownik*** – login zalogowanego użytkownika.

*private Timer **odswiezacz*** – obiekt odpowiadający za odświeżanie liczby nieodczytanych wiadomości.

Konstruktory:

public ***WyborRozmowcy****()* - tworzy nowe okno.

Metody:

private void ***OnOdswiezEvent****(object o)* – jeśli to konieczne, odświeża liczbę nieodczytanych wiadomości.

Argumenty:

o – dowolny obiekt (zalecane przekazanie wartości **null**).

private void ***odswiezKontaktyIWyzerujNoweWiadomosci****()* - zeruje liczbę nieodczytanych wiadomości dla każdego kontaktu.

private void ***poinformujONowychWiadomosciach****()* - wyświetla liczbę nieodczytanych wiadomości.

Metody powiązane ze zdarzeniami:

private void ***dodajUzytkownika_Click****(object sender, RoutedEventArgs e)* – dodaje nowy kontakt, aktywowane po kliknięciu na przycisk dodania nowego kontaktu.

private void ***szukanyUzytkownik_KeyDown****(object sender, KeyEventArgs e)* - dodaje nowy kontakt, aktywowane po wciśnięciu klawisza Enter.

private void ***otworz_Click****(object sender, RoutedEventArgs e)* – otwiera okno z konwersacją, aktywowane po kliknięciu na przycisk otwarcia konwersacji.

private ***void usun_Click(****object sender, RoutedEventArgs e)* – usuwa kontakt, aktywowane po kliknięciu na przycisk usunięcia kontaktu.

Testy jednostkowe

znajdzIdUzytkownikaTest – testuje metodę ***znajdzIdUzytkownika(string login)***

znajdzUzytkownikaPoIdTest – testuje metodę ***znajdzUzytkownikaPoId(string id)***

sprawdzCzySaNoweWiadomosciTest – testuje metodę ***sprawdzCzySaNoweWiadomosci()***

sprawdzCzySaNoweWiadomosciTest2 – testuje metodę ***sprawdzCzySaNoweWiadomosci(string login)***

znajdzUzytkownikaTest – testuje metodę ***znajdzUzytkownika(string szukanyLogin)***

wyslijWiadomoscTest – testuje metodę ***wyslijWiadomosc(string tresc)*** (sprawdza, czy wiadomości są zapisane w bazie danych)

wyslijWiadomoscTest2 – testuje metodę ***wyslijWiadomosc(string tresc)*** (sprawdza, czy metoda zwraca poprawny czas serwera)

Aplikacja mobilna

Klasa Konwersacja

Pakiet: gr2.pz2.pwsip2018.komunikator.wysylanieWiadomosci

Klasa reprezentująca konwersację pomiędzy użytkownikami.

Pola:

public String ***login*** – login zalogowanego użytkownika.

public String ***adresat*** – login użytkownika, z którym jest prowadzona konwersacja.

private static final String ***DANE_BAZY*** – dane serwera MySQL.

private static final String ***UZYTKOWNIK_BAZY*** – login użytkowika serwera MySQL

private static final String ***HASLO_BAZY*** – hasło użytkownika serwera MySQL

Konstruktory:

public ***Konwersacja****(String login, String adresat)* – tworzy egzemplarz klasy.

Argumenty:

login – login zalogowanego użytkownika.

adresat – login użytkownika, z którym prowadzona jest konwersacja.

Metody:

private static void ***przygotujDoPolaczeniaZBaza****()* - przygotowuje urządzenie do połączenia z serwerem MySQL.

public static String ***znajdzIdUzytkownika****(String login)* – zwraca ID w bazie danych danego użytkownika.

Argumenty:

login – login użytkownika, którego ID ma zostać znalezione.

public static String ***znajdzUzytkownikaPoId****(String id)* – zwraca login użytkownika, do którego jest

przypisane dane ID w bazie danych.

Argumenty:

id – ID w bazie danych, do którego jest przypisany szukany użytkownik.

public static ArrayList<Kontakt> ***zaladujKontakty****(String login)*– zwraca kontakty danego użytkownika.

Argumenty:

login – login użytkownika, którego kontakty mają być zwrócone.

public ArrayList<Wiadomosc> ***wczytajWiadomosci****()* - zwraca wszystkie wiadomości, które zostały wysłane pomiędzy zalogowanym użytkownikiem i adresatem.

public String ***wyslijWiadomosc****(String tresc)* – wysyła wiadomość do adresata. Zwraca czas serwera, w

którym wiadomość została dostarczona.

Argumenty:

tresc – treść wysyłanej wiadomości

public ArrayList<Wiadomosc> ***odswiezKonwersacje****()* - zwraca nieodczytane wiadomości od adresata.

public static Boolean ***znajdzUzytkownika****(String szukanyLogin)* – zwraca **true**, jeśli użytkownik o podanym loginie istnieje, w przeciwnym wypadku **false**.

Argumenty:

szukanyLogin – login szukanego uzytkownika.

public static void ***dodajKontakt****(String login1, String login2)* – dodaje nowy kontakt.

Argumenty:

login1, ***login2*** – loginy użytkowników z nowego kontaktu.

public static Boolean ***sprawdzCzySaNoweWiadomosci****(String login)* – zwraca **true**, jeśli dany użytkownik ma nieodczytane wiadomości, w przeciwnym przypadku **false**.

Argumenty:

login – login użytkownika, którego wiadomości są sprawdzane.

public static HashMap<String, Integer> ***wyswietlPowiadomieniaONowychWiadomosciach****(String login)* – zwraca odwzorowanie zawierające kontakty danego użytkownika i liczbę odpowiadających im nieodczytanych wiadomości (zwraca tylko kontakty, od których są nieodczytane wiadomości)

Argumenty:

login – login użytkownika, którego liczba nieodczytanych wiadomości ma być zwrócona.

Klasa Konwersacja.Wiadomosc

Pakiet: gr2.pz2.pwsip2018.komunikator.wysylanieWiadomosci

Reprezentuje wiadomość.

Pola:

public String ***tresc*** – treść wiadomości.

public String ***uzytkownik*** – użytkownik, który wysłał wiadomość.

public String ***data*** – data i czas wysłania wiadomości.

Konstruktory:

public ***Wiadomosc****(String tresc, String uzytkownik, String data)* – tworzy egzemplarz klasy.

Argumenty:

tresc – treść wiadomości.

uzytkownik – użytkownik, który wysłał wiadomość.

data – data i czas wysłania wiadomości.

Klasa Konwersacja.Kontakt

Pakiet: gr2.pz2.pwsip2018.komunikator.wysylanieWiadomosci

Reprezentuje kontakt.

Pola:

public String ***login*** – login użytkownika, do którego należy kontakt.

public int ***nieodczytaneWiadomosci*** – liczba nieodczytanych wiadomości, od danego kontaktu.

Konstruktory:

public ***Kontakt****(String login)* – tworzy egzemplarz klasy.

Argumenty:

login - login użytkownika, do którego należy kontakt.

Metody:

public String ***toString****()* - zwraca login użytkownika, do którego należy kontakt, jeśli nie ma od tego kontaktu żadnych nieodczytanych wiadomości, w przeciwnym przypadku zwraca login użytkownika, do którego należy kontakt, i liczbę nieodczytanych wiadomości w nawiasie.

Klasa WiadomoscAdapter

Pakiet: gr2.pz2.pwsip2018.komunikator.wysylanieWiadomosci

Dziedziczy po: ***BaseAdapter***

Adapter dla obiektów typu Konwersacja.Wiadomosc.

Pola:

private ArrayList<Konwersacja.Wiadomosc> ***wiadomosci*** – wiadomości przechowywane przez adapter.

private LayoutInflater ***li*** – obiekt do obsługi layoutu.

Konstruktory:

public ***WiadomoscAdapter****(Context kontekst, ArrayList<Konwersacja.Wiadomosc> wiadomosci)* – tworzy egzemplarz klasy.

Argumenty:

kontekst – kontekst, w którym ma działać adapter.

wiadomości – kolekcja wiadomości, które mają być obsługiwane przez adapter.

Metody:

public int ***getCount****()* - zwraca liczbę wiadomości obsługiwanych przez adapter.

public Object ***getItem****(int position)* – zwraca obiekt z określonej pozycji.

Argumenty:

position – pozycja żądanego obiektu.

public View ***getView****(int position, View convertView, ViewGroup parent)* – zwraca widok wiadomości.

Klasa KonwersacjaOkno

Pakiet: gr2.pz2.pwsip2018.komunikator

Dziedziczy po: ***AppCompatActivity***

Reprezentuje interfejs czatu.

Pola:

private Konwersacja ***k*** – konwersacja, która ma być wyświetlona.

private ArrayList<Konwersacja.Wiadomosc> ***wczytaneWiadomosci*** – wyświetlane wiadomości.

private WiadomoscAdapter ***adapter*** – adapter do obsługi wyświetlania wiadomości.

private Handler ***odswiezacz*** – obiekt odpowiedzialny za odświeżanie konwersacji.

private Runnable ***dzialanie*** – obiekt odpowiedzialny za akcje wywoływane podczas odświeżania.

private ListView ***czat*** – kontrolka wyświetlająca konwersację.

private EditText ***wiadomoscTekst*** – kontrolka do wpisywania wiadomości.

Metody:

private void ***dodajNoweWiadomosci****()* - wyświetla nadesłane wiadomości.

Metody powiązane ze zdarzeniami:

public void ***onClickWyslij****(View v)* - wysyła wpisaną wiadomość, aktywowane po kliknięciu na przycisk do wysyłania wiadomości.

public void ***onBackPressed****()* - zatrzymuje odświeżanie konwersacji, aktywowane po kliknięciu na przycisk cofania.

Klasa WyborRozmowcy

Pakiet: gr2.pz2.pwsip2018.komunikator

Dziedziczy po: ***AppCompatActivity***

Reprezentuje interfejs wyboru użytkownika, z którym ma się odbyć konwersacja.

Pola:

private String ***zalogowanyUzytkownik*** – login zalogowanego użytkownika

private ListView ***kontakty*** – kontrolka wyświetlająca kontakty.

private ArrayList<Konwersacja.Kontakt> ***kontaktyUzytkownika*** – kolekcja kontaktów użytkownika

private EditText ***szukanyUzytkownik*** – kontrolka do wpisywania loginu szukanego użytkownika

private ArrayAdapter<Konwersacja.Kontakt> ***adapter*** – adapter do obsługi wyświetlania kontaktów

private Handler ***odswiezacz*** – obiekt odpowiedzialny za odświeżanie liczby nieodczytanych wiadomości.

private Runnable ***dzialanie*** – obiekt odpowiedzialny za akcje wykonywane podczas odświeżania.

Metody:

private void ***poinformujONowychWiadomosciach****()* - wyświetla liczbę nieodczytanych wiadomości.

private void ***odswiezKontaktyIWyzerujNoweWiadomosci****() - zeruje liczbę nieodczytanych wiadomości dla każdego kontaktu.*

Metody powiązane ze zdarzeniami:

public void ***onClickDodajUzytkownika****(View v)* - dodaje nowy kontakt, aktywowane po kliknięciu na przycisk dodania nowego kontaktu.

Testy jednostkowe

znajdzIdUzytkownikaTest – testuje metodę ***znajdzIdUzytkownika(String login)***

znajdzUzytkownikaPoIdTest – testuje metodę ***znajdzUzytkownikaPoId(String id)***

znajdzUzytkownikaTest – testuje metodę ***znajdzUzytkownika(string szukanyLogin)***

sprawdzCzySaNoweWiadomosciTest – testuje metodę ***sprawdzCzySaNoweWiadomosci(String login)***

8. Instrukcja dla użytkownika

Aplikacja desktopowa

Rejestracja

Logowanie

Dodawanie kontaktów

Aby dodać kontakt, należy wpisać login szukanego użytkownika w polu w górnej części okna programu, następnie wcisnąć Enter lub kliknąć na przycisk „Dodaj użytkownika”. Jeśli wpisany login, jest poprawny, nowy kontakt zostanie dodany do listy.

Usuwanie kontaktów

Aby usunąć kontakt z listy, należy wybrać go, klikając na niego, następnie kliknąć na przycisk „Usuń” w dolnej części okna programu.

Zmiana statusu

Wysyłanie wiadomości

Aby wysłać wiadomość innemu użytkownikowi, należy wybrać go z listy kontaktów, klikając na jego loginie, następnie kliknąć na przycisk „Otwórz” znajdujący się w dolnej części okna programu (jeśli przycisk jest nieaktywny należy zmienić status). Otworzy się nowe okno, w którym wyświetlona jest dotychczasowa konwersacja z wybranym użytkownikiem. Aby wysłać wiadomość, należy wprowadzić jej treść w polu w dolnej części okna programu i wcisnąć Enter lub kliknąć na przycisk „Wyślij”.

Blokowanie

Aplikacja mobilna

Rejestracja

Logowanie

Dodawanie kontaktów

Aby dodać kontakt, należy wpisać login szukanego użytkownika w polu w górnej części programu, następnie kliknąć na przycisk „DODAJ”. Jeśli wpisany login, jest poprawny, nowy kontakt zostanie dodany do listy.

Zmiana statusu

Wysyłanie wiadomości

Aby wysłać wiadomość innemu użytkownikowi, należy wybrać go z listy kontaktów, klikając na jego loginie. Pojawi się nowy ekran, w którym wyświetlona jest dotychczasowa konwersacja z wybranym użytkownikiem. Aby wysłać wiadomość, należy wprowadzić jej treść w polu w dolnej części programu i kliknąć na przycisk „WYŚLIJ”.

Blokowanie

Numer wersji	Tytuł	Uczestnicy	Data

0.1	Opis projektu, założenia funkcjonalne i niefunkcjonalne.	Andrzej Świderski Bartosz Prusaczyk Tomasz Milianowicz Bartosz Giełdon Maksymilian Głowacki	14.03.2018r.
0.2	Diagram przypadków użycia, diagram sekwencji. Technologia, metodyka, podział pracy, harmonogram	Andrzej Świderski Bartosz Prusaczyk Tomasz Milianowicz Bartosz Giełdon Maksymilian Głowacki	21.03.2018r.
0.3	Dodanie opisu kodu programu – klasy Konwersacja, KonwersacjaOkno i WyborRozmowcy	Bartosz Prusaczyk	9.04.2018r.
0.4	Uzupełnienie dokumentacji o opis testów jednostkowych	Bartosz Prusaczyk	20.04.2018r.
0.5	Dodanie opisu kodu aplikacji mobilnej (klasy Konwersacja, Konwersacja.Wiadomosc, Konwersacja.Kontakt, WiadomoscAdapter, KonwersacjaOkno, WyborRozmowcy, testy jednostkowe), dodanie instrukcji dla użytkownika	Bartosz Prusaczyk	21.04.2018r.

Repozytorium: PWSIP_2018_PZ2_GR2