POLITECHNIKA KOSZALIŃSKA WYDZIAŁ ELEKTRONIKI I INFORMATYKI TiIM

Komunikator internetowy dokumentacja

Tomasz Trzciński

Mariusz Pawłowski

Maciel Chwaleba

# Spis treści

[Spis treści 2](#_Toc262456725)

[Protokół załóżycielski 3](#_Toc262456726)

[Opis funkcjonalny komunikatora internetowego (MODUŁ – funkcja) 4](#_Toc262456727)

[Wykaż narzędzi użytych do budowy projektu 6](#_Toc262456728)

[Harmonogram zadań 7](#_Toc262456729)

[Mapy konceptualne 8](#_Toc262456730)

[Mapy myśli 10](#_Toc262456731)

[Diagramy przypadków użycia 13](#_Toc262456732)

[Diagramy sekwencyjne 15](#_Toc262456733)

[Diagramy klas 16](#_Toc262456734)

[Opis klas 19](#_Toc262456735)

[Słowniczek pojęć 24](#_Toc262456736)

# Protokół załóżycielski

Protokół ze spotkania założycielskiego Grupy 6 rocznika 2009/2010 Informatyki specjalność "Technologie Internetowe i Mobilne", które odbyło się w dniu 26 lutego 2010.

Czas realizacji: od 26 luty 2010 r. do 30 czerwca 2010 r.

Obecni na spotkaniu członkowie grupy:

Tomasz Trzciński

Mariusz Pawłowski

Maciej Chwaleba

Po zapoznaniu się z celami i zakresem zadań ustanowiono:

1. Głosem większości funkcję kierownika projektu obejmuje Tomasz Trzciński wybrany demokratycznie spośród grupy.
2. Uczestnicy zobowiązują się do pełnej współpracy w ramach realizowanego projektu zespołowego.
3. Każdemu z uczestników przydzielone zostanie przez prowadzącego określone zadanie projektowe.
4. Poza przydzielonymi zadaniami indywidualnymi grupa ma wspierać pozostałe części projektu.
5. Całość jest koordynowana przez Kierownika Projektu.
6. Kierownik projektu może nakładać kare nagany lub karę obniżenia oceny o 0.5 stopnia lub w skrajnym wypadku usunąć uczestnika z grupy.
7. Kierownik może nałożyć nagrodę w postaci podwyższenia o 0.5 stopnia  
   w razie ukończenia projektu wysoce przed terminem lub za wyjątkowe zasługi.
8. Kierownik projektu odpowiedzialny jest za współdziałanie wszystkich modułów i komponentów projektu.
9. Ponadto uczestnicy nie mogą odstąpić od współpracy w określonym terminie (czas realizacji).
10. Kierownik projektu ma wyznaczać i kontrolować cele rozwojowe projektu.
11. Każdy z uczestników ma prawo do zgłaszania swoich pomysłów i poprawek odnośnie wykonywanego projektu do kierownika projektu.
12. Wszelkie trudności związane z porozumieniem uczestników rozstrzyga kierownik projektu.
13. Za całokształt projektu odpowiedzialni są jednakowo wszyscy członkowie.
14. Każdy współprojektant ma obowiązek aktywnego uczestniczenia w rozwoju projektu w czasie jego realizacji.

Podpis kier. projektu Podpisy członków zespołu

# Opis funkcjonalny komunikatora internetowego (MODUŁ – funkcja)

KLIENT

* wysyłanie i odbiór ciągów tekstowych o długości do 5000 znaków
* szyfrowanie tekstu za pomocą komponentu szyfrującego
* opcje zalogowania i wylogowania do i z serwera
* ustawianie IP, portu, klucza do protokołu komunikacyjnego za pomocą interfejsu panelowego
* zapis wszystkich wykonywanych działań do logu przez komponent logu
* wysyłanie prośby o utworzenie konta za pomocą formularza w interfejsie panelowym

SERWER

* włączanie, wyłączanie i resetowanie procesu pracy serwera
* odbiór oraz wysyłanie pakietów o długości do 5000 znaków
  + dodawanie, edycja i usuwanie kont użytkowników za pomocą formularzy panelu interfejsowego
  + zalogowanie i wylogowywanie użytkowników za pomocą interfejsu panelowego
  + blokowanie konta przez administratora w interfejsowym panelu
  + zapis wszystkich wykonywanych działań do logu przez komponent logu
  + zapis wszystkich otrzymywanych i wysyłanych ciągów tekstowych do logu
* ustawianie IP, portu, klucza do protokołu komunikacyjnego za pomocą interfejsu panelowego
* rozsyłanie informacji o aktywności serwera, zalogowaniu i wylogowaniu użytkowników, informacji i reklam wysyłanych przez administratora
* szyfrowanie tekstu za pomocą komponentu szyfrującego

SZYFROWANIE

* szyfrowanie tekstu i deszyfrowanie tekstu do 5000 znaków długości.
* ustawianie klucza szyfru podstawowego tekstowego przez okna dialogowe interfejsu panelowego
* informowanie użytkownika o pracy przez ikonę panelu komponentu
* generowanie klucza szyfru ukrytego na podstawie wprowadzonego ciągu tekstowego
* ustawianie klucza szyfru przesuwającego
* przetwarzanie danych w czasie rzeczywistym
* ustawianie kluczy i parametrów za pomocą procedur ustawiających dla zachowania niejawności klucza

LOGOPISARZ

* zapis odbieranych ciągów tekstowych do 5500 znaków jako wpis do pliku logu
* utworzenie nowego pliku logu w razie nie znalezienia pliku pod zadeklarowanym adresem
* odczyt pliku logu w celu dopisania doń nowego wpisu
* ustawianie ścieżki dostępu do pliku logu za pomocą interfejsu panelowego
* ustawianie nazwy pliku logu za pomocą interfejsu panelowego
* ustawanie ścieżki dostępu do pliku oraz nazwy pliku logu za pomocą procedur ustawiających w celu zachowania niejawności danych
* informowanie użytkownika o pracy komponentu za pomocą ikony na panelu komponentu

# Wykaż narzędzi użytych do budowy projektu

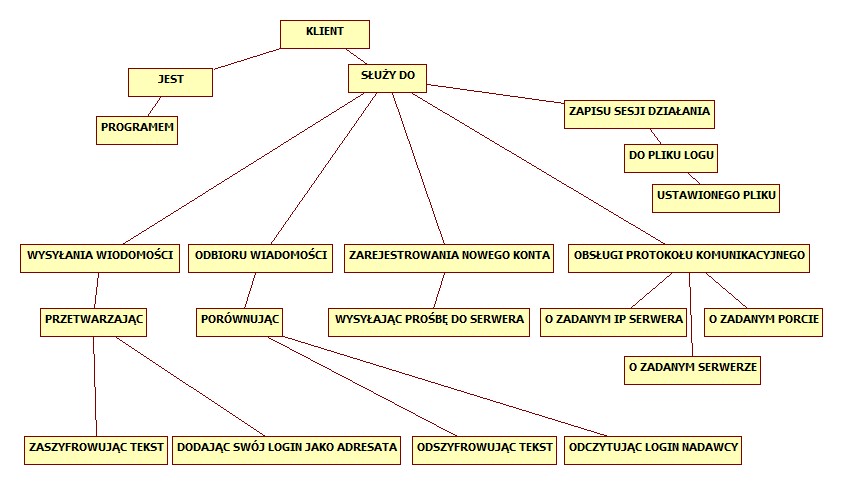
* Windows 7 Professional 64-bit (6.1, Build 7600) (7600.win7\_rtm.090713-1255)
* Microsoft Office Word 2007 (12.0.6514.5000) SP2 MSO (12.0.6521.5000)
* Microsoft Office Excel 2007 (12.0.6524.5003) SP2 MSO (12.0.6521.5000)
* StarUML v5.0.2.1570
* ArgoUML v0.28.1
* NetBeans IDE 6.9.1 Milestone 1 dev (Build 201002152000)
* 1.6.0\_16; Java HotSpot(TM) Client VM 14.2-b01

# Harmonogram zadań

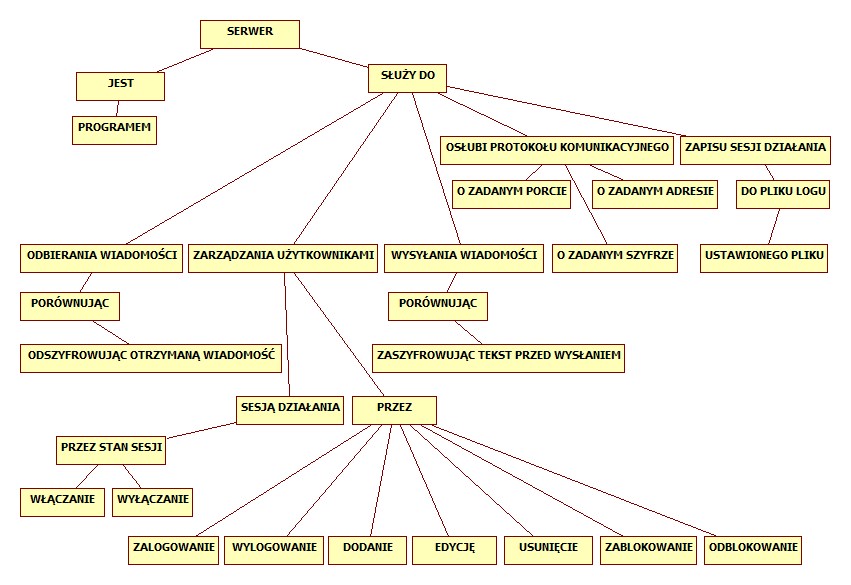
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **Skompletowanie grupy** (wybór tematu projektu, wybór kierownika projektu grupy, sporządzenie protokołu założycielskiego) | **√** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Wyodrębnienie komponentów** (przydział zagadnień projektowych członkom grupy) |  | **√** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Sporządzenie diagramów** (mapy myśli poszczególnych elementów, opis funkcjonalny projektu, diagramy przypadków użycia poszczególnych elementów oraz całości projektu) |  |  | **√** |  |  |  |  |  |  |  |  |  |  |  |
| **Sporządzenie słowniczka pojęć** |  |  |  | **√** |  |  |  |  |  |  |  |  |  |  |
| **Sporządzenie harmonogramu zadań** |  |  |  |  | **√** |  |  |  |  |  |  |  |  |  |
| **Sporządzenie diagramu klas** |  |  |  |  |  | **√** |  |  |  |  |  |  |  |  |
| **Sporządzenie wykazu narzędzi użytych do budowy projektu** |  |  |  |  |  |  | **√** |  |  |  |  |  |  |  |
| **Skompletowanie dokumentacji** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Implementacja komponentów projektu** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Testowanie poprawności działania komponentów** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Integracja komponentów oraz implementacja projektu** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Testowanie projektu** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Weryfikacja zgodności projektu z założeniami** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Sfinalizowanie całości projektu** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# Mapy konceptualne

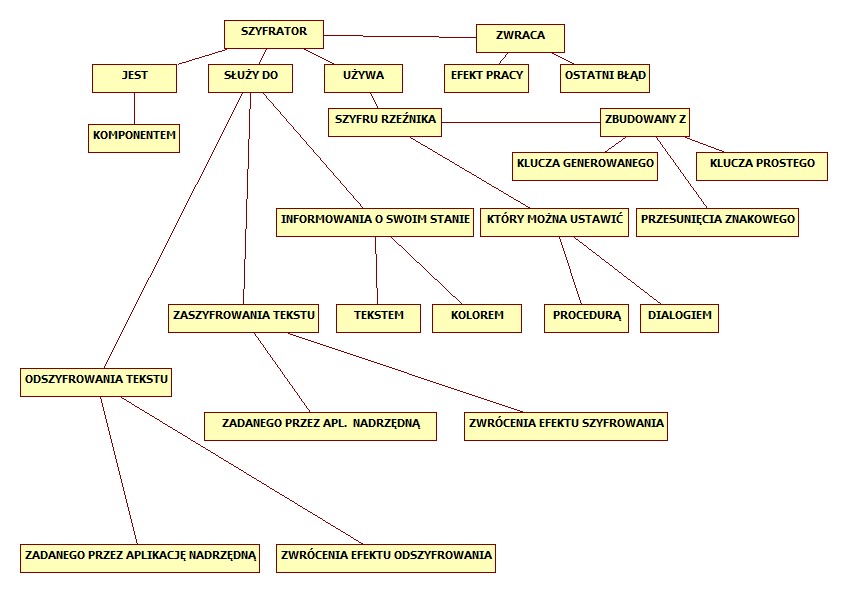
1. Klient



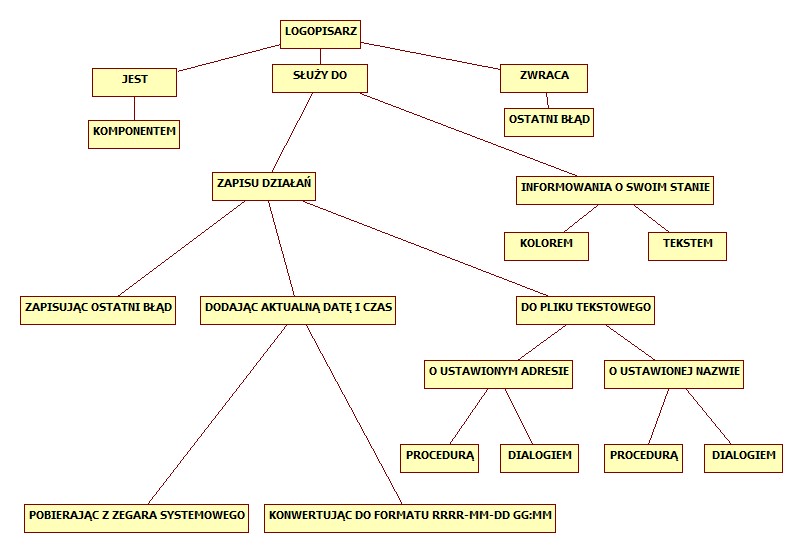
1. Serwer



1. Szyfrator

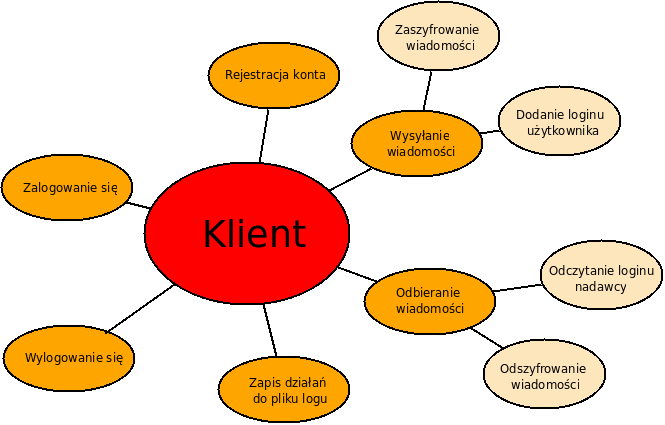


1. Logopisarz

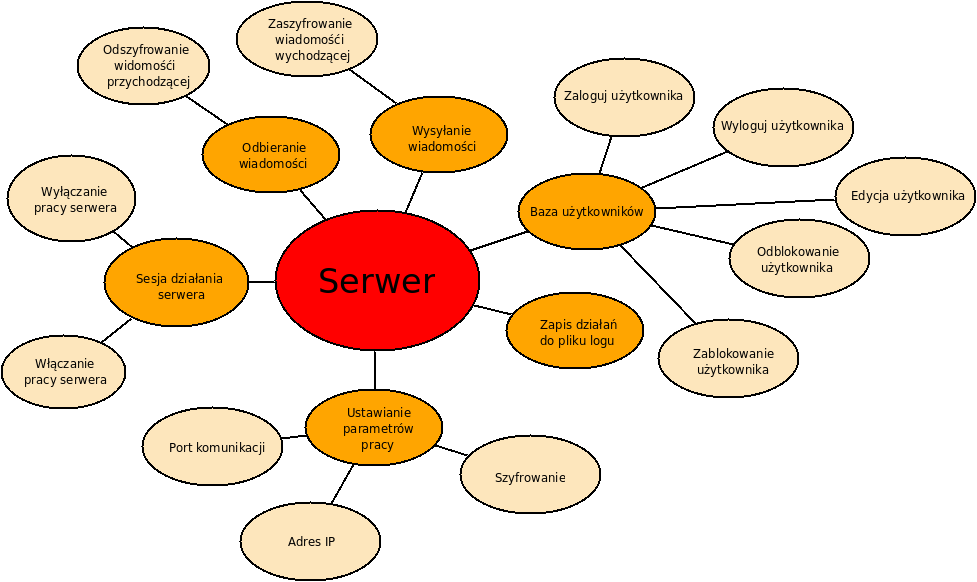


# Mapy myśli

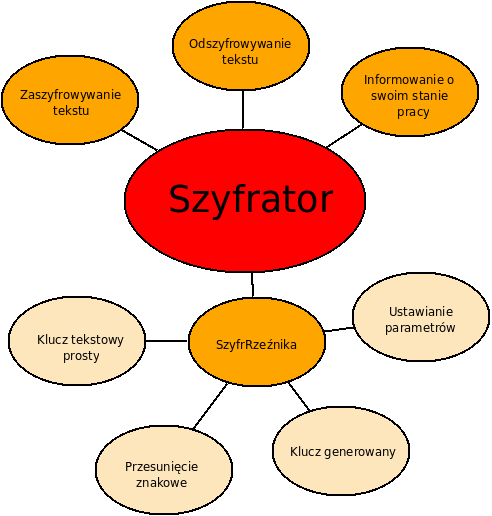
1. Klient



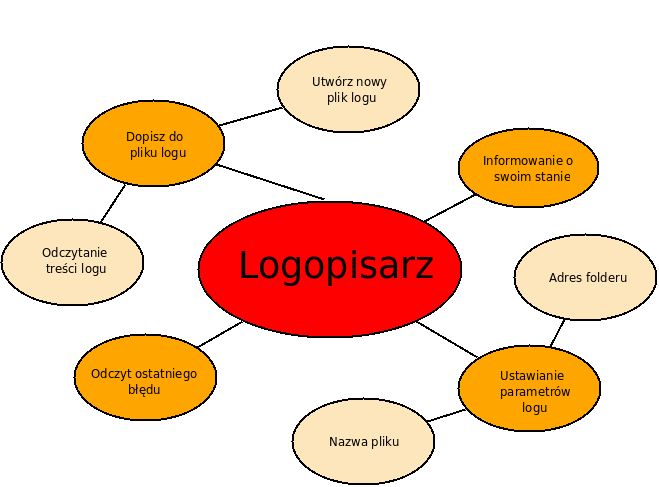
1. Serwer



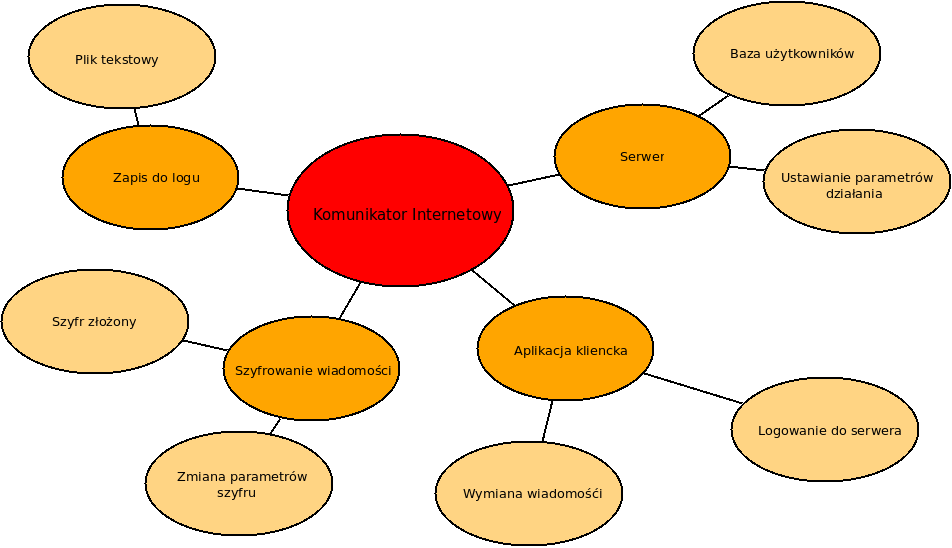
1. Szyfrator



1. Logopisarz

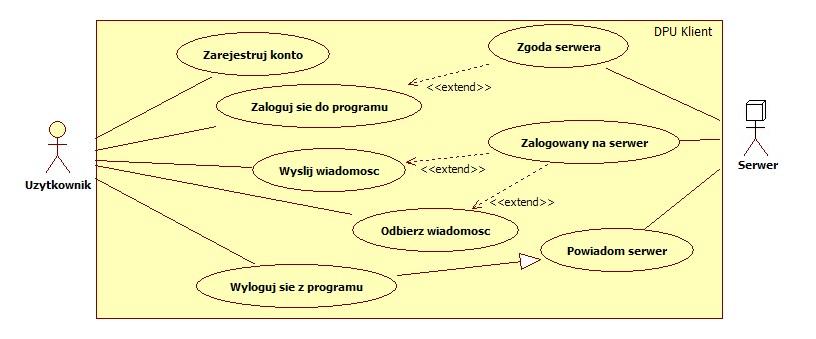


1. Mapa zbiorcza

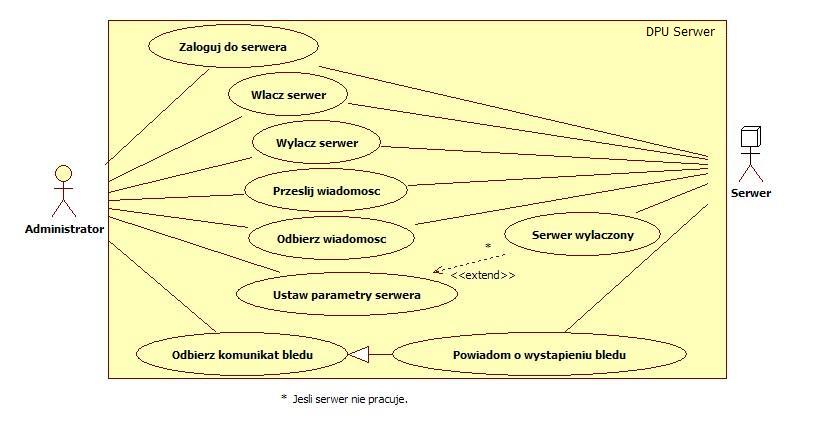


# Diagramy przypadków użycia

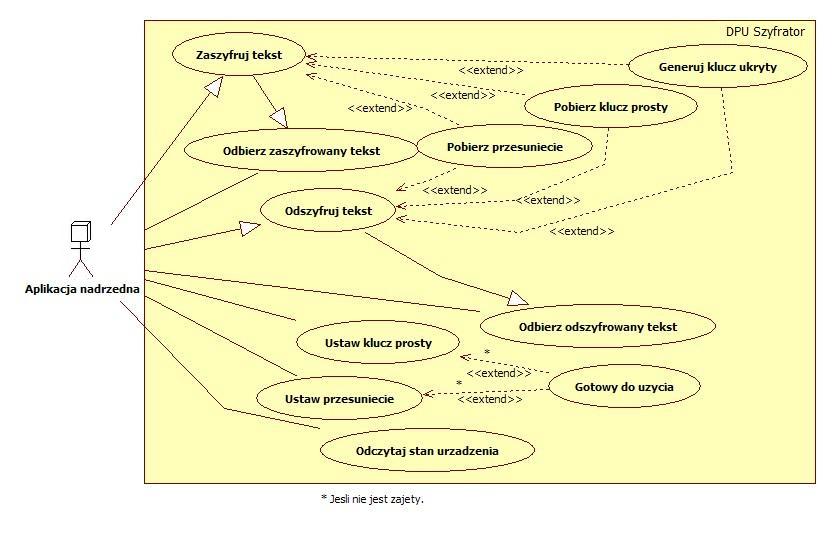
1. Klient



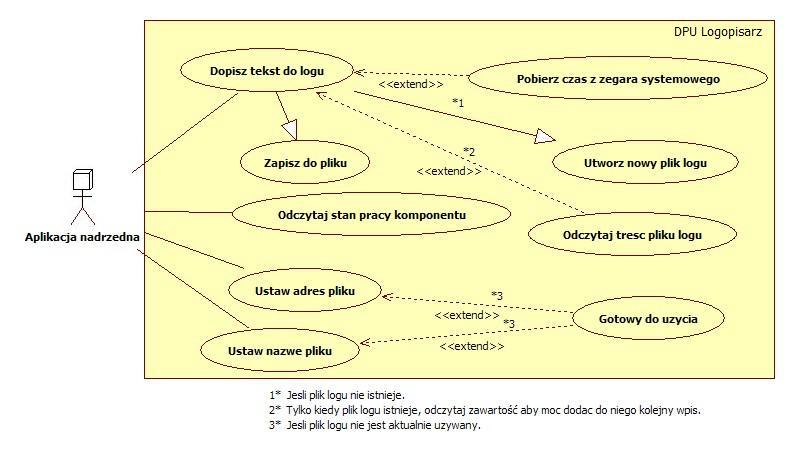
1. Serwer



1. Szyfrator

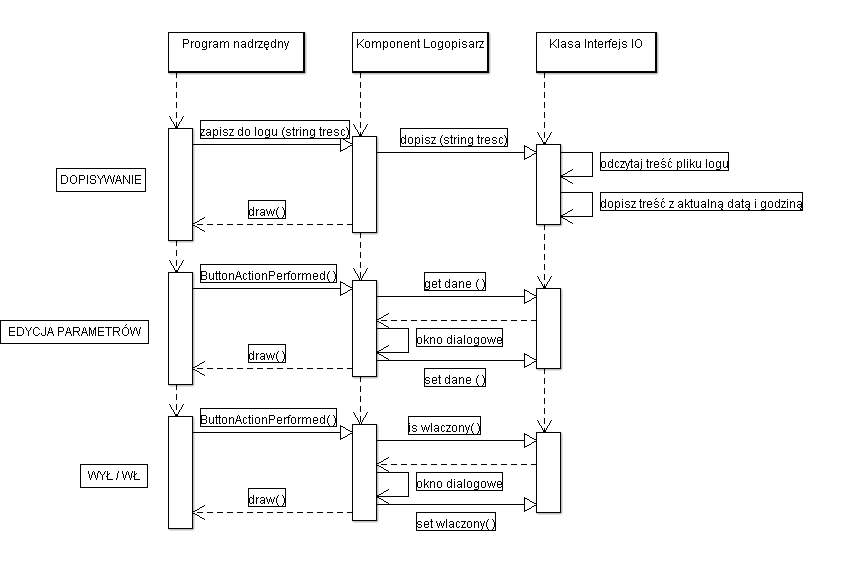


1. Logopisarz

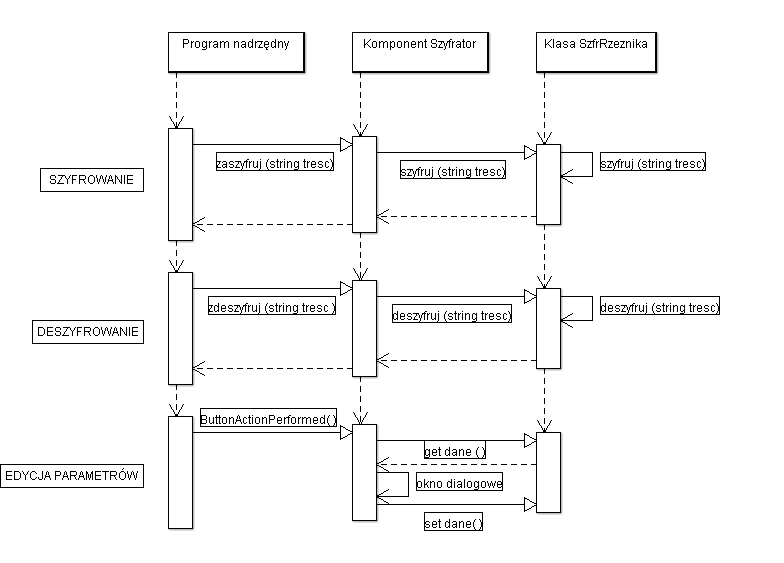


# Diagramy sekwencyjne

1. Logopisarz

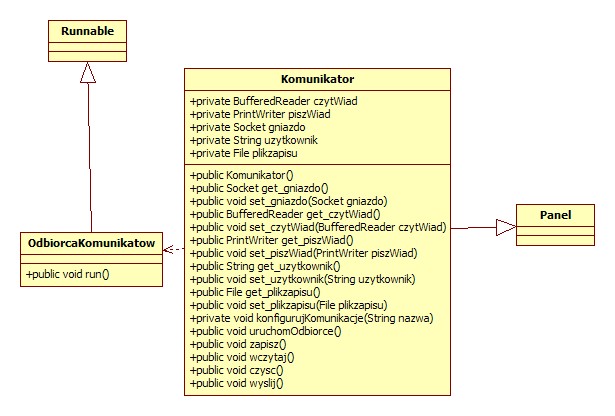


1. Szyfrator

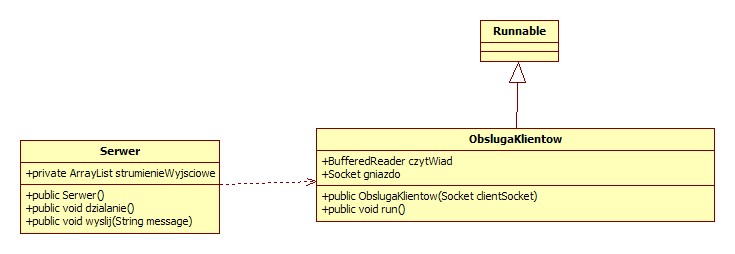


# Diagramy klas

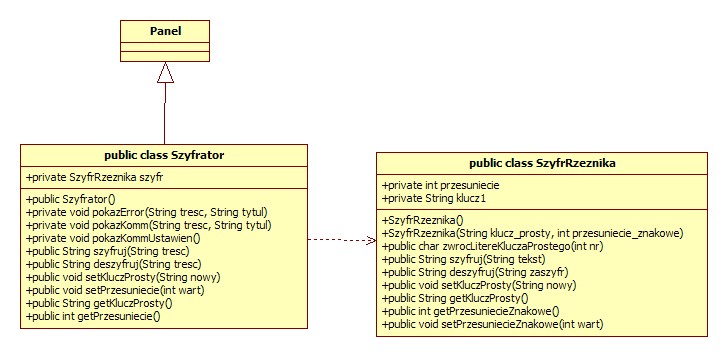
1. Komunikator



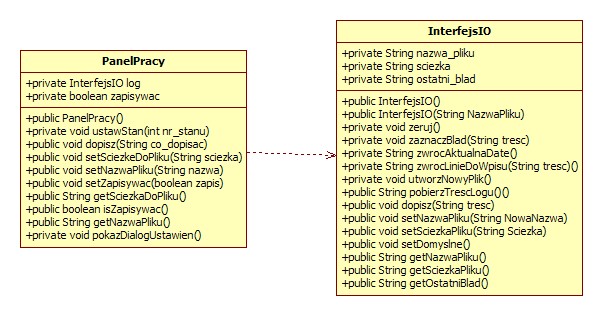
1. Serwer



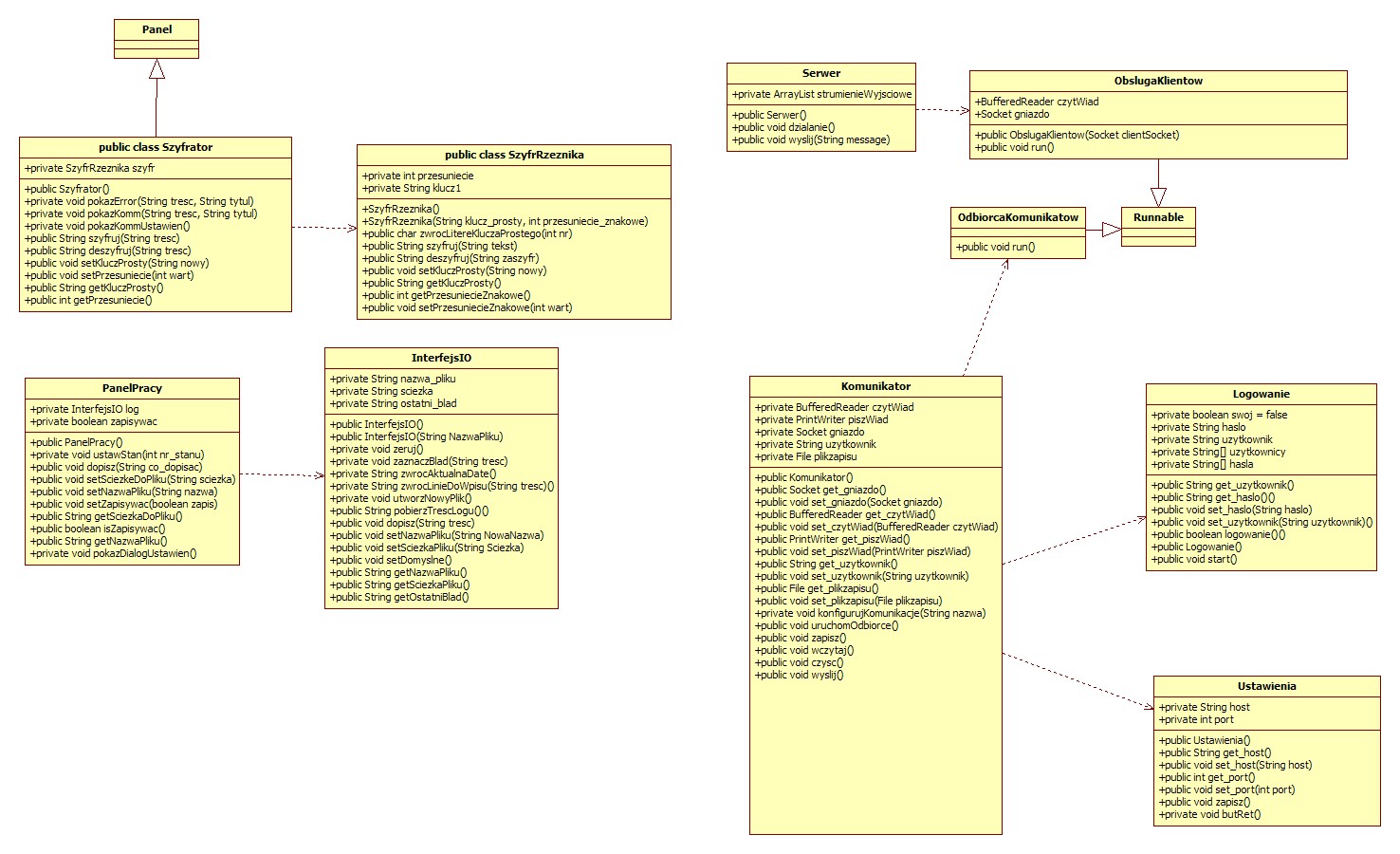
1. Szyfrator



1. Logopisarz



1. Diagram zbiorczy



# Opis klas

ZMIENNE – OPIS

Komunikator::Komunikator

* private BufferedReader czytWiad – zmienna strumieniowa służąca do odczytu nadchodzących wiadomości.
* private PrintWriter piszWiad – zmienna strumieniowa służąca do wysyłania wiadomości wychodzących.
* private Socket gniazdo – przechowuje informację o adresie IP i porcie, po którym komunikator ma się komunikować z serwerem.
* private String uzytkownik – łańcuch przechowujący nazwę użytkownika pod jaka jest on widoczny dla czytających.
* private File plikzapisu – zawiera nazwę pliku, do którego program odnosi się przy zapisywaniu treści rozmów.

Komunikator::Ramka

* private Komunikator komunkiator1 – egzemplarz komponentu opartego o klasę Komunikator znajdujący się w oknie widocznym dla użytkownika.

Logopisarz::PanelPracy

* private InterfejsIO log – obiekt klasy InterfejsIO służący do przechowywania informacji o nazwie i ścieżce dostępu pliku oraz prowadzący zapis.
* private boolean zapisywac – zmienna określająca, czy prowadzić zapis do pliku logu.

Logopisarz::InterfejsIO

* private String nazwa\_pliku – przechowuje nazwę pliku logu, do którego obiekt prowadzi zapis.
* private String sciezka – przechowuje adres pliku, gdzie znajduje się plik logu.
* private String ostatni\_blad – przechowuje treść ostatniego błędu, jaki wystąpił ww czasie działania programu.

Serwer::Serwer

* ArrayList strumienieWyjsciowe – przechowuje strumienie określająca adresy ip oraz port komunikacyjny do aktualnie zalogowanych użytkowników.

Serwer::ObslugaKlientow

* BufferedReader czytWiad - zmienna strumieniowa służąca do odczytu nadchodzących wiadomości
* Socket gniazdo - przechowuje informację o adresie IP i porcie, po którym komunikator ma się komunikować z klientem.

Szyfrator::Szyfrator:

* private SzyfrRzeznika szyfr – obiekt szyfru służący do przetwarzania tekstu i przechowujący parametry kluczy.

Szyfrator::SzyfrRzeznika

* private int przesuniecie – przechowuje wartość przesunięcia znakowego
* private String klucz1 – przechowuje treść klucza podstawowego prostego.

PROCEDURY - OPIS

Komunikator::Komunikator

* public Komunikator() - konstruktor bezparametrowy do klasy Komunikator.
* public Socket get\_gniazdo() - zwraca adres ip i port, po którym następuje komunikacja.
* public void set\_gniazdo(Socket gniazdo) – ustawia adres ip i port komunikacji na zadane w obiekcie gniazdo.
* public BufferedReader get\_czytWiad() - zwraca zawartość strumienia czytWiad.
* public void set\_czytWiad(BufferedReader czytWiad) – ustawia zawartość strumienia czytWiad na zadaną.
* public PrintWriter get\_piszWiad() - zwraca zawartość strumienia zapisu do zapisania na polu tekstowym.
* public void set\_piszWiad(PrintWriter piszWiad) – ustawia strumień zapisu dla pola tekstowego na zadany.
* public String get\_uzytkownik() - zwraca nazwę użytkownika.
* public void set\_uzytkownik(String uzytkownik) – ustawia nazwę użytkownika na zadaną.
* private void konfigurujKomunikacje(String nazwa) – ustawia podstawowe wartości na zmiennych w obiekcie.
* public File get\_plikzapisu() - zwraca nazwę pliku z zapisanymi rozmowami w postaci obiektu klasy File.
* public void set\_plikzapisu(File plikzapisu) – ustawia plik do zapisu rozmów na następujący.
* public void zapisz() - metoda zapisująca aktualną rozmowę do pliku tekstowego.
* public void wczytaj() - metoda wczytująca odbytą rozmowę zapisaną w pliku tekstowym i wypisująca ją na pole tekstowe.
* private void czysc() - procedura czyszcząca pole tekstowe z rozmową.
* public void wyslij() - procedura nadpisująca prowadzoną rozmowę o kolejną wypowiedź użytkownika.

Komunikator::OdbiorcaKomunikatow

* run() - metoda uruchamiająca wątek czytający nadchodzące wiadomości i wypisujący je na pole tekstowe.

Logopisarz::PanelPracy

* public PanelPracy() - konstruktor bezparametrowy tworzący obiekt klasy PanelPracy.
* private void ustawStan(int nr\_stanu) – procedura ustawiająca stan pracy komponentu na panelu tak, aby wyświetlany był odpowiedni kolor i treść. Pracuje na trzech stanach: Czeka, Zapis i Wyłączony.
* public void dopisz(String co\_dopisac) – procedura odwołujący się do obiektu log w celu dopisania podanego tekstu do pliku logu.
* public void setSciezkeDoPliku(String sciezka) – procedura odwołująca się do obiektu log w celu ustawienia ścieżki do pliku logu.
* public void setNazwaPliku(String nazwa) – procedura odwołująca się do obiektu log w celu ustawienia nazwy pliku logu na podaną.
* public void setZapisywac(boolean zapis) – procedura ustawiająca zmienną zapisywac na podany stan.
* public String getSciezkaDoPliku() - procedura zwracająca ścieżkę dostępu do pliku z obiektu log.
* public String getNazwaPliku() - procedura zwracająca nazwę pliku logu z obiektu log.
* public boolean isZapisywac() - procedura zwracająca stan zmiennej zapisywac.
* private void pokazDialogUstawien() - procedura wywołująca pokazanie komunikatu ustawiania parametrów zapisu do logu, czyli nazwę pliku, ścieżkę dostępu i zaznaczenie zapisu.

Logopisarz::InterfejsIO

* InterfejsIO() - konstruktor bezparametrowy tworzący obiekt klasy InterfejsIO.
* InterfejsIO(String NazwaPliku) – konstruktor parametrowy tworzący obiekt klasy InterfejsIO z podaną nazwą pliku logu.
* private void zeruj() - procedura wprowadzająca do wszystkich parametrów logu ich domyślne wartości. Jest używana w konstruktorze przy deklaracji zmiennych.
* private void zaznaczBlad(String tresc) – procedura wprowadzająca podaną treść do zmiennej ostatni\_blad.
* private String zwrocAktualnaDate() - procedura zwracająca aktualną datę w formacie HH:mm dd-MM-yyyy jako łańcuch tekstowy.
* private String zwrocLinieDoWpisu(String tresc) – procedura dodająca do podanej treści aktualną datę i nawiasy, po czym zwraca powstały łańcuch tekstowy.
* private void utworzNowyPlik() - procedura tworząca nowy plik według podanych parametrów ścieżki i nazwy w obiekcie jednocześnie usuwając treść z pliku, jeśli taki już istnieje.
* public String pobierzTrescLogu() - zwraca treść logu z odczytanego pliku logu zgodnie z parametrami ścieżki do pliku i nazwy pliku.
* public void dopisz(String tresc) – procedura wymuszająca zapis podanej treści do pliku, przy czym dodaje do treści z aktualną godziną na końcu odczytanej treści pliku lub tworzy nowy w razie jego braku.
* public void setNazwaPliku(String NowaNazwa) – ustawia nazwę pliku na podaną.
* public void setSciezkaPliku(String Sciezka) – ustawia ścieżkę dostępu do pliku na podaną.
* public void setDomyslne() - wywołuję procedurę zeruj() na obiekcie.
* public String getNazwaPliku() - zwraca nazwę pliku logu.
* public String getSciezkaPliku() - zwraca ścieżkę dostępu do pliku logu.
* public String getOstatniBlad() - zwraca treść ostatniego błędu, jaki wystąpił w czasie działania obiektu.

Serwer::ObslugaKlientow

* ObslugaKlientow(Socket clientSocket) - konstruktor obiektu tworzonego w celu obsługi jednego z zalogowanych użytkowników.
* run() - działanie obiektu wymuszająca nasłuchiwanie nadchodzących wiadomości od zalogowanych użytkowników i wysyłanie odebranych dalej.

Serwer::Serwer

* public void dzialanie() - główna metoda programu zawierająca wywołania wątków dotyczących odczytu i wysyłania wiadomości.
* public void wyslij(String message) – metoda rozsyłająca po kolei do wszystkich zalogowanych użytkowników zadaną wiadomość.

Szyfrator::Szyfrator

* public Szyfrator() - konstruktor bezparametrowy tworzący obiekt klasy Szyfrator.
* private void pokazError(String tresc, String tytul) – pokazuje okno dialogowe informujące o błędzie z zadaną treścią i tytułem.
* private void pokazKomm(String tresc, String tytul) – pokazuje okno dialogowe informujące o informacjach i zmianach o zadanej treści i tytule.
* private void pokazKommUstawien() - pokazuje okno dialogowe stanowiące formularz ustawiania parametrów szyfrowania.
* public String szyfruj(String tresc) – tworzy zaszyfrowany ciąg tekstowy na podstawie zadanej treści.
* public String deszyfruj(String tresc) – tworzy odszyfrowany ciąg tekstu na podstawie zadanej zaszyfrowanej treści.
* public void setKluczProsty(String nowy) – ustawia klucz prosty tekstowy na zadany.
* public void setPrzesuniecie(int wart) – ustawia przesunięcie znakowe na zadane.
* public String getKluczProsty() - zwraca treść klucz prostego.
* public int getPrzesuniecie() - zwraca wartość przesunięcia znakowego.

Szyfrator::SzyfrRzeznika

* SzyfrRzeznika() - konstruktor bezparametrowy tworzący obiekt klasy SzyfrRzeznika.
* SzyfrRzeznika(String klucz\_prosty, int przesuniecie\_znakowe) – konstruktor parametrowy tworzący obiekt klasy SzyfrRzeznika z parametrami kluczy podanymi jako argumenty.
* public char zwrocLitereKluczaProstego(int nr) – zwraca pojedynczą literę klucza prostego o numerze nr
* public String szyfruj(String tekst) – procedura, która szyfruje podany tekst z użyciem kluczy szyfrów używanych w programie metodą litera po literze.
* public String deszyfruj(String zaszyfr) – procedura odszyfrowująca podany zakodowany fragment tekstu z użyciem przechowywanych w obiekcie kluczy metodą litera po literze.
* public void setKluczProsty(String nowy) – ustawia wartość klucza prostego (tekstowego) na podany łańcuch znaków.
* public void setPrzesuniecieZnakowe(int wart) – ustawia wartość przesunięcia znakowego na podaną wartość.
* public String getKluczProsty() - zwraca wartość klucza prostego.
* public int getPrzesuniecieZnakowe() - zwraca wartość przesunięcia znakowego.

# Słowniczek pojęć

***KLIENT*** - Jest to aplikacja użytkownika po stronie dołączającego się.

***SERWER*** - Jest to program służący do przesyłania dalej wiadomości otrzymanych od użytkowników.

***WIADOMOŚĆ*** - Pojedyńczy fragment tekstu, wysłany z aplikacji klienckiej do serwera i odwrotnie.

***LOGOPISARZ*** - Jest to komponent tworzący i nadpisujący plik tekstowy logu.

***LOG***- Jest to wykaz zdarzeń obsługiwany przez komunikator.

***SZYFRATOR*** - Jest to komponent zabezpieczający wiadomości przed odczytem z zewnątrz.

***UŻYTKOWNIK*** - Osoba logująca się do serwera.

***ŚCIEŻKA*** - Jest to pełny adres bezwzględny do folderu zawierającego wybrany plik.

***SZYFRRZEŹNIKA*** - Jest to autorski szyfr złożony operujący na wektorach znków alfanumerycznych.

Składa się z szyfru prostego, przesunięcia i szyfru uktytego.

***SZYFR PROSTY*** - Jest to łańcuch teksotwy przechowujący klucz do przyrostu znaku o wartość odpowiadającego indeksem znaku.

***SZYFR UKRYTY*** - Jest to generowana zmiana znaku względem jego indeksu i długości klucza prostego.

***PRZESUNIĘCIE*** - Jest to przyrost wartoci znaku w fragmencie szyfrowanym w tablicy znaków UNICODE.

***SZYFROWANIE*** - Proces mający na celu takie przetworzenie fragmentu tekstu aby bez pewnych znanych prametrów nie był czytelny.

***DESZYFROWANIE*** - Proces mający na celu przetworzenie zaszyfrowaniego fragmentu tekstu z użyciem pewnych parametrów by uczynić go ponownie czytelnym.

***PROGRAM NADRZĘDNY*** - Jest to aplikacja posiadająca egzemplarz komponentu, który jest podrzędny.

***DIALOG***- Jest o komunikat wyświetlający informacje lub pytanie z prośbą wyboru rozwiązania.