

WYDZIAŁ PODSTAWOWYCH PROBLEMÓW TECHNIKI
POLITECHNIKA WROCŁAWSKA

MEMORY-HARD FUNCTIONS

KONRAD ŚWIERCZYŃSKI
NR INDEKSU: 229818

Promotor
dr Filip Zagórski



Politechnika
Wrocławska
WROCŁAW 2019

Spis treści

1	Wstęp	1
2	Analiza problemu	3
3	Projekt systemu	7
3.1	Grupy użytkowników i założenia	7
3.2	Przypadki użycia i scenariusze	7
3.3	Diagramy klas	7
3.4	Diagramy aktywności	7
3.5	Diagramy sekwencji	7
3.6	Diagramy stanów	8
3.7	Projekt bazy danych	8
3.8	Opis protokołów	8
3.9	Opis algorytmów	9
4	Implementacja systemu	11
4.1	Opis technologii	11
4.2	Omówienie kodów źródłowych	11
5	Instalacja i wdrożenie	13
6	Podsumowanie	15
	Bibliografia	17
A	Zawartość płyty CD	19

Wstęp

Moderately hard functions. Funkcje umiarkowanie ciężkie do obliczenia mają wiele zastosowań takich jak dowody pracy (ang. *proofs of work*), funkcje wyprowadzenia klucza oraz password hashing. Przy przechowywaniu haseł ważne jest, aby zminimalizować skutki wycieknienia pliku z hasłami. Zamiast przechowywać krotki (*login, password*) tekstem jawnym, dodaje się losową sól i przechowuje w postaci (*login, f(password, salt), salt*), gdzie f jest moderately hard function. Oznacza to, że funkcja ta musi być obliczana podczas każdego uwierzytelniania w celu sprawdzenia poprawności hasła. Nie może być ona zatem zbyt ciężka do obliczenia dla aplikacji uwierzytelniającej. Z drugiej strony, gdy krotka (*login, y, salt*) wycieknie, adversarz może przeprowadzać atak słownikowy obliczając funkcję f przy każdej próbie, co powinno być kosztowne. W tym celu zaczęto stosować funkcje, które obliczają wiele razy kryptograficzną funkcję skrótu. Popularnym przykładem takiej funkcji jest PBKDF2 (ang. *Password-Based Key Derivation Function 2*), dla której zalecanym parametrem bezpieczeństwa w 2000 roku było 1024 iteracji, a już w 2005 zaczęto zalecać 4096 iteracji, z powodu wzrostu wydajności CPU. Niestety takie podejście nie gwarantuje zabezpieczenia przed adversarzem używającym specjalizowany układ scalony (ang. *ASIC - Application-Specific Integrated Circuit*). Układy takie są znacznie bardziej wydajne pod względem szybkości obliczania funkcji skrótu takich jak SHA256 czy MD5 niż tradycyjne architektury.

– Porównanie Antminer - GPU - CPU –

Zauważono jednak, że na różnych architekturach koszt dostępu do pamięci jest dużo bardziej zrównoważony niż koszt obliczeń. [Percival [16]] Zaproponowano więc memory-hard functions (MHF), które wywołują podczas obliczania wiele kosztownych czasowo odwołań do pamięci.

scrypt - pierwsza taka funkcja

O MHF można myśleć jako o pewnej kolejności dostępu do komórek pamięci. Odwołania następują do już wcześniej obliczonych wartości w komórkach. Zatem kolejność tę można opisać jako acykliczny graf skierowany (DAG).

.....



Analiza problemu

Definicja 2.1 (*Parallel/Sequential Graph Pebbling*). Niech $G = (V, E)$ będzie grafem skierowanym grafem acyklicznym i niech $T \subset V$ będzie zbiorem wierzchołków do oetykietowania. T będzie nazywane celem. Stanem etykietowania G jest zbiór $P_i \subset V$. Poprawnym etykietowaniem równoległym jest ciąg $P = (P_0, \dots, P_t)$ stanów etykietowania G , gdzie $P_0 = \emptyset$ oraz gdzie spełnione są warunki 1 oraz 2 poniżej. Etykietowanie sekwencyjne musi dodatkowo spełniać warunek 3.

1. Każdy wierzchołek z celu jest w pewnej konfiguracji oetykietowany (nie koniecznie wszystkie jednocześnie).

$$\forall x \in T \exists x \leq t : x \in P_x$$

2. Oetykietować wierzchołek można tylko wtedy, gdy wszyscy jego rodzice są oetykietowani w poprzednim kroku.

$$\forall i \in [t] : x \in (P_i \setminus P_{i-1}) \Rightarrow \text{parents}(x) \subset P_{i-1}$$

3. W każdym kroku można oetykietować co najwyżej jeden wierzchołek.

$$\forall i \in [t] : |P_i \setminus P_{i-1}| \leq 1$$

Zbiory poprawnych etykietowań sekwencyjnych i równoległych grafu G z celem T oznaczamy odpowiednio jako $\mathcal{P}_{G,T}$ oraz $\mathcal{P}_{G,T}^{\parallel}$. Etykietowania najbardziej interesujących przypadków, gdy $T = \text{sinks}(G)$, oznaczamy \mathcal{P}_G oraz $\mathcal{P}_G^{\parallel}$.

Można zauważyć, że $\mathcal{P}_{G,T} \subset \mathcal{P}_{G,T}^{\parallel}$.

Definicja 2.2 Złożoność czasową, pamięciową, pamięciowo-czasową oraz łączną etykietowania $P = (P_0, \dots, P_t) \in \mathcal{P}_G^{\parallel}$ są zdefiniowane jako

$$\Pi_t(P) = t, \Pi_s(P) = \max_{y \in [t]} |P_y|, \Pi_{st}(P) = \Pi_t(P) * \Pi_s(P), \Pi_{cc}(P) = \sum_{i \in [t]} |P_i|$$

Dla $\alpha \in s, t, st, cc$ oraz celu $T \subset V$, złożoności sekwencyjnego oraz równoległego etykietowania grafu G definiujemy jako

$$\Pi_{\alpha}(G, T) = \min_{P \in \mathcal{P}_{G,T}} \Pi_{\alpha}(P)$$

$$\Pi_{\alpha}^{\parallel}(G, T) = \min_{P \in \mathcal{P}_{G,T}^{\parallel}} \Pi_{\alpha}(P)$$

Kiedy $T = \text{sinks}(G)$, piszemy $\Pi_{\alpha}^{\parallel}(G)$ oraz $\Pi_{\alpha}(G)$.

Definicja 2.3 (*N-Superconcentrator*) Skierowany graf acykliczny $G = (V, E)$ o ustalonym stopniu, N wejściach i N wyjściach nazywany jest *N-Superkoncentratorem* gdy dla każdego $k \in [N]$ oraz dla każdej pary podzbiorów $V_1 \subset V$ k wejść i $V_2 \subset V$ k wyjść istnieje k wierzchołkowo-rozłącznych ścieżek łączących wierzchołki ze zbioru V_1 z wierzchołkami w V_2 .

Definicja 2.4 (*(N, λ)-Superconcentrator*) Niech $G_i, i = 0, \dots, \lambda - 1$ będą *N-Superkoncentratorami*. Niech graf G będzie połączeniem wyjść G_i do odpowiadających wejść w G_{i+1} dla $i = 0, \dots, \lambda - 2$. Graf G jest nazywany *(N, λ)-Superkoncentratorem*.



Twierdzenie 2.1 (*Ograniczenie dolne dla (N, λ) -Superkoncentratora*) *Pebbling a (N, λ) -Superconcentrator using $S \leq N/20$ pebbles requires T placements such that*

$$T \geq N \left(\frac{\lambda N}{64S} \right)^\lambda.$$

Twierdzenie 2.2 *Niech $\lambda, n \in \mathbb{N}^+$ takie, że $n = \bar{n}(2\lambda c + 1)$, gdzie $c \in \mathbb{N}$ i $\bar{n} = 2^c$. Wtedy dla $g = \lfloor \sqrt{\bar{n}} \rfloor$ $RSG_\lambda^{\bar{n}} \in \mathbb{D}_{1,g}^{\lambda, \bar{n}}$ oraz $\Pi_{cc}^\parallel(RSG_\lambda^{\bar{n}}) = \Omega\left(\frac{n^{1.5}}{c\sqrt{c\lambda}}\right)$.*

Dowód. Niech $G = RSG_\lambda^{\bar{n}}$, niech $G_1, G_2, \dots, G_\lambda$ będą podgrafami G opisanymi w DEF[...]. Pokażemy, że każdy G_i jest (g, \bar{n}) -dispersed dla $g = \lfloor \sqrt{\bar{n}} \rfloor$.

Wybermy $i \in [\lambda]$ niech L_1 będzie ostatnimi \bar{n} wierzchołkami w porządku topologicznym grafu G_i . Oznaczamy wierzchołki zbioru L_1 poprzez $1 \times [\bar{n}]$, gdzie druga pozycja odpowiada kolejności wierzchołków w porządku topologicznym. Niech $\bar{g} = \lfloor \bar{n}/g \rfloor$, dla każdego $j \in [\bar{g}]$ $L_{1,j} = \{< 1, jg + x > : x \in [0, g - 1]\}$. Pokażemy, że wszystkie $L_{1,j}$ mają (g, g) -dependency.

Niech L_0 będzie \bar{n} pierwszymi wierzchołkami G_i , które oznaczamy $0 \times [\bar{n}]$ (ponownie druga pozycja odpowiada porządkowi topograficznemu). Zauważmy, że dla $n > 1$ i $g = \lfloor \sqrt{\bar{n}} \rfloor$ prawdą jest, że $g(g - 2c + 1) \leq n$. Zatem zbiór $S = \{< 0, i(g - 2c + 1) > : i \in [g]\}$ jest całkowicie zawarty w L_0 .

Z własności RSG [Superconcentrator- ...] wynika, że skoro zbiory S oraz $L_{1,j}$ mają po g wierzchołków, to istnieje g wierzchołkowo-rozłącznych ścieżek o długości $2c$ między wierzchołkami tych zbiorów. Zatem $L_{1,j}$ ma $(g, 2c)$ -dependency.

Rozszerzmy to do (g, g) -dependency. Niech ścieżka p zaczynająca się w wierzchołku $< 0, v > \in S$ będzie ścieżką w $(g, 2c)$ -dependency $L_{1,j}$. Zauważmy, że istnieje ścieżka przechodząca przez wszystkie wierzchołki L_0 oraz, że wierzchołki zbioru S są oddzielone między sobą o $g - 2c$ wierzchołków. Możemy dodać na początek ścieżki p ścieżkę $< 0, v - (g - 2c - 1) >, < 0, v - (g - 2c - 2) >, \dots, < 0, v >$. Otrzymujemy w ten sposób ścieżkę p_+ o długości $2c + g - 2c = g$. Ponieważ każda para ścieżek $p \neq q$ w $(g, 2c)$ -dependency $L_{1,j}$ jest wierzchołkowo-rozłączna, to w szczególności zaczynają się muszą w różnych wierzchołkach $< 0, v_p > \neq < 0, v_q >$. Ponieważ wierzchołki w S są od siebie oddalone o $g - 2c$ wierzchołków, zatem ścieżki p_+ i q_+ nadal pozostają rozłączne. Rozszerzając w ten sposób wszystkie ścieżki z $(g, 2c)$ -dependency otrzymujemy ścieżki wierzchołkowo-rozłączne długości g . Z tego wynika, że $L_{1,j}$ ma (g, g) -dependency, co dowodzi, że $RSG_\lambda^{\bar{n}} \in \mathbb{D}_{1,g}^{\lambda, \bar{n}}$. Pozostaje obliczyć górne ograniczenie używając [Theorem 6 ABP2017].

$$\Pi_{cc}^\parallel(RSG_\lambda^{\bar{n}}) = \lambda g \left(\frac{\bar{n}}{2} - g \right) \geq \lambda \lfloor \sqrt{\bar{n}} \rfloor \left(\frac{\bar{n}}{2} - \lfloor \sqrt{\bar{n}} \rfloor \right) = \lambda \sqrt{\bar{n}} \left(\frac{\bar{n}}{2} - \sqrt{\bar{n}} \right) - O(\bar{n}) = \Omega(\lambda \bar{n}) = \Omega\left(\frac{n^{1.5}}{c\sqrt{c\lambda}}\right) \quad \square$$

Twierdzenie 2.3 *Niech $\lambda, g \in \mathbb{N}^+$, $N = 2^g$, $n = N(2\lambda g + 1)$, wtedy*

$$\Pi_{cc}^\parallel(RSG_\lambda^{\bar{n}}) = O\left(n^{1.6}\right)$$

Dowód. Kozystając z [...], mamy, że RSG_λ^N jest λ -Stacked Sandwich Graph. Z [Lemma 4.2 AB16] wynika więc, że RSG_λ^N jest $(n/t, \lambda + t - \lambda - 1)$ -reducible dla dowolnego $t \geq 1$. Z twierdzenia 10 [ABP17] wynika, że $\Pi_{cc}^\parallel(RSG_\lambda^{\bar{n}}) = O\left(n \left(\sqrt{(\lambda t + t - \lambda - 1)n\delta} + \frac{n}{t} \right)\right)$. Aby dostać najdokładniejsze (najmniejsze) ograniczenie górne trzeba zminimalizować $n \left(\sqrt{(\lambda t + t - \lambda - 1)n\delta} + \frac{n}{t} \right)$. Zanim jednak przejdziemy do minimalizowania, uprośmy nieco to wyrażenie

$$n \left(\sqrt{(\lambda t + t - \lambda - 1)n\delta} + \frac{n}{t} \right) \leq n \left(\sqrt{2\lambda t n\delta} + \frac{n}{t} \right).$$

Teraz możemy znaleźć minimum względem naszego parametru t .

$$\frac{\partial}{\partial t} n \left(\sqrt{2\lambda t n\delta} + \frac{n}{t} \right) = \frac{\sqrt{2\lambda n^3}}{2t} - \frac{n^2}{t^2}$$

Minimum znajduje się w punkcie, gdzie pochodna ma wartość zero.

$$\frac{\sqrt{2\delta\lambda n^3}}{2t} - \frac{n^2}{t^2} = 0$$
$$t = \frac{2n^2}{\sqrt{2\delta\lambda n^3}} = O\left(n^{\frac{1}{3}}\right)$$

Zatem podstawiając t minimalizujące ograniczenie górne do wzoru z twierdzenie 10 [ABP17] otrzymujemy

$$\Pi_{cc}^{\parallel}(RSG_{\lambda}^{\bar{n}}) = O\left(n\left(\sqrt{(\lambda n^{\frac{1}{3}} + n^{\frac{1}{3}} - \lambda - 1)n\delta + \frac{n}{n^{\frac{1}{3}}}}\right)\right) = O\left(n^{1.\bar{6}}\right)$$

□



Projekt systemu

W tym rozdziale przedstawiono szczegółowy projekt systemu w notacji UML uwzględniający wymagania funkcjonalne opisane w rozdziale 2. Do opisu relacji pomiędzy składowymi systemu wykorzystano diagramy Przedstawiono w pseudokodzie i omówiono algorytmy generowania

3.1 Grupy użytkowników i założenia

Architektura systemu ... jest wielowarstwowa i rozproszona, przy czym Podsystem ... jest systemem zbiorczym dla danych ... wysyłanych do serwera

Taka architektura jest zgodna z wzorcem projektowym MVC¹ (ang. Model-View-Controller). Przetwarzanie danych odbywa się

3.2 Przypadki użycia i scenariusze

W tej sekcji należy przedstawić przypadki użycia oraz odpowiadające im scenariusze dla poszczególnych grup użytkowników

3.3 Diagramy klas

W tej sekcji należy przedstawić diagramy klas dla odpowiednich elementów systemu zidentyfikowane na podstawie wcześniejszych rozważań

3.4 Diagramy aktywności

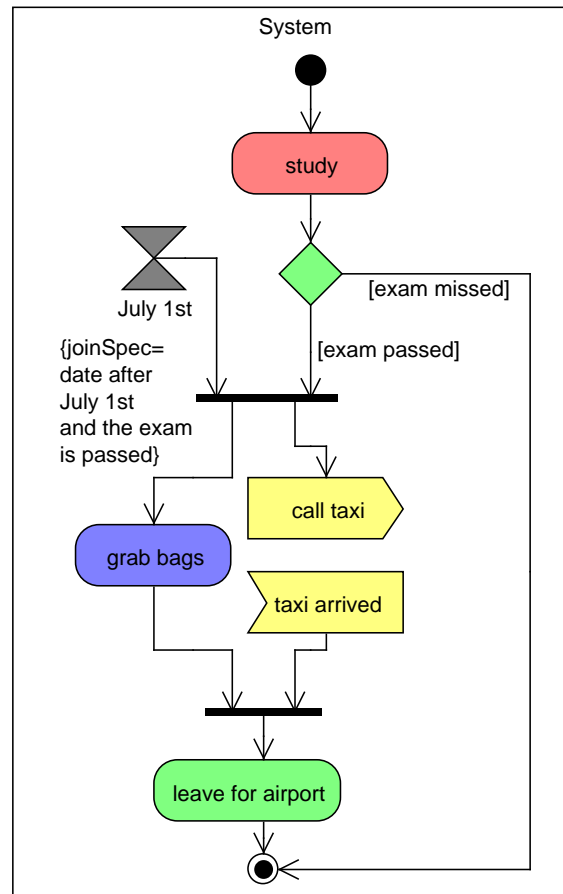
W tej sekcji należy przedstawić diagramy aktywności dla elementów systemu i odpowiednich procesów wynikające z wcześniejszej analizy.

W niniejszym rozdziale przedstawiono diagramy aktywności Diagram na rysunku 3.1 przedstawia

3.5 Diagramy sekwencji

W tej sekcji należy przedstawić diagramy sekwencji dla obiektów systemu zidentyfikowanych na podstawie wcześniejszych rozważań. Należy wykorzystać nazewnictwo wprowadzone w poprzednich rozdziałach, w szczególności odpowiadające definicjom wprowadzonych klas.

¹Należy odnieść się do wykorzystywanych wzorców projektowych



Rysunek 3.1: Diagram aktywności związany z procesem rejestracji dokumentu.

3.6 Diagramy stanów

W tej sekcji należy przedstawić diagramy stanów w których może znaleźć się system. Diagramy te są szczególnie istotne przy projektowaniu systemów czasu rzeczywistego.

3.7 Projekt bazy danych

W tej sekcji należy przedstawić projekt bazy danych. Należy omówić wycinek rzeczywistości i odpowiadające mu zidentyfikowane elementy systemu, których wartości będą podlegać utrwalaniu. Należy przedyskutować wybór typów danych dla atrybutów poszczególnych obiektów. Należy uzasadnić wybór platformy DBMS. Dla relacyjnych baz danych należy przedyskutować jej normalizację.

3.8 Opis protokołów

W tej sekcji należy omówić protokoły wykorzystywane przez komponenty systemu. Omówić formaty komunikatów i zilustrować je przykładami.

3.9 Opis algorytmów

W tej sekcji należy wymienić i przedyskutować algorytmy wykorzystywane w systemie. Algorytmy należy przedstawić w pseudokodzie (wykorzystać pakiet `algorithm2e`). Omówienia poszczególnych kroków algorytmów powinny zawierać odwołania do odpowiednich linii pseudokodu. Dla zaproponowanych autorskich algorytmów należy przeprowadzić analizę ich złożoności czasowej i pamięciowej.

Algorytm bąblowania jest przedstawiony w Pseudokodzie [3.1](#).

Pseudokod 3.1: Wyporność przez bąblowanie

Input: Zbiór bąbli B

Output: Wyporność W

```
1 foreach  $b \in B$  do
2   Process( $b$ );
3   for  $i \leftarrow 1$  to  $|B|$  do
4     if Calculate( $EW(i, b)$ )  $\leq 0$  then
5        $b \leftarrow 2 * b$ ;
6 while  $B \neq \emptyset$  do
7   for  $j \leftarrow 1$  to  $|B|$  do
8     if Calculate( $FT(j, \hat{b})$ )  $\leq 0$  then
9        $w \leftarrow 2 * \hat{b}$ ;
10       $W \leftarrow W \cup \{w\}$ ;
11       $B \leftarrow B \setminus \{b\}$ ;
```



Implementacja systemu

4.1 Opis technologii

Należy tutaj zamieścić krótki opis (z referencjami) do technologii użytych przy implementacji systemu.

Do implementacji systemu użyto języka JAVA w wersji . . . , szczegółowy opis można znaleźć w [1]. Interfejs zaprojektowano w oparciu o HTML5 i CSS3 [3].

4.2 Omówienie kodów źródłowych

Kod źródłowy 4.1 przedstawia opisy poszczególnych metod interfejsu: WSPodmiotRejestracjaIF. Kompletne kody źródłowe znajdują się na płycie CD dołączonej do niniejszej pracy w katalogu Kody (patrz Dodatek A).

Kod źródłowy 4.1: Interfejs usługi Web Service: WSPodmiotRejestracjaIF.

```
package erejestracja.podmiot;
import java.rmi.RemoteException;
// Interfejs web serwisu dotyczącego obsługi podmiotów i rejestracji.
public interface WSPodmiotRejestracjaIF extends java.rmi.Remote{
// Pokazuje informacje o danym podmiocie.
// parametr: nrPeselRegon – numer PESEL podmiotu lub numer REGON firmy.
// return: Podmiot – obiekt transportowy: informacje o danym podmiocie.
public Podmiot pokazPodmiot(long nrPeselRegon) throws RemoteException;
// Dodaje nowy podmiot.
// parametr: nowyPodmiot – obiekt transportowy: informacje o nowym podmiocie.
// return: true – jeśli podmiot dodano, false – jeśli nie dodano.
public boolean dodajPodmiot(Podmiot nowyPodmiot) throws RemoteException;
// Usuwa dany podmiot.
// parametr: nrPeselRegon – numer PESEL osoby fizycznej lub numer REGON firmy.
// return: true – jeśli podmiot usunięto, false – jeśli nie usunięto.
public boolean usunPodmiot(long nrPeselRegon) throws RemoteException;
// Modyfikuje dany podmiot.
// parametr: podmiot – obiekt transportowy: informacje o modyfikowanym podmiocie.
// return: true – jeśli podmiot zmodyfikowano, false – jeśli nie zmodyfikowano.
public boolean modyfikujPodmiot(Podmiot podmiot) throws RemoteException;
// Pokazuje zarejestrowane podmioty na dany dowód rejestracyjny.
// parametr: nrDowoduRejestracyjnego – numer dowodu rejestracyjnego.
// return: PodmiotRejestracja[] – tablica obiektów transportowych: informacje o
// wszystkich zarejestrowanych podmiotach.
public PodmiotRejestracja[] pokazZarejestrowanePodmioty(
String nrDowoduRejestracyjnego) throws RemoteException;
// Nowa rejestracja podmiotu na dany dowód rejestracyjny.
// parametr: nrDowoduRejestracyjnego – numer dowodu rejestracyjnego.
// parametr: nrPeselRegon – numer PESEL podmiotu lub numer REGON firmy.
// parametr: czyWlasciciel – czy dany podmiot jest właścicielem pojazdu.
// return: true – jeśli zarejestrowano podmiot, false – jeśli nie zarejestrowano.
```



```

public boolean zarejestrujNowyPodmiot(String nrDowoduRejestracyjnego ,
long nrPeselRegon , boolean czyWlasciciel) throws RemoteException;
// Usuwa wiązanie pomiędzy danym podmiotem, a dowodem rejestracyjnym.
// parametr: nrDowoduRejestracyjnego – numer dowodu rejestracyjnego.
// parametr: nrPeselRegon – numer PESEL podmiotu lub numer REGON firmy.
// return: true – jeśli podmiot wyrejestrowano, false – jeśli nie wyrejestrowano.
public boolean wyrejestrujPodmiot(String nrDowoduRejestracyjnego ,
long nrPeselRegon) throws RemoteException;

```

Kod źródłowy 4.2 przedstawia procedurę przetwarzającą żądanie. Hasz utrwalany %granulacja wykorzystany jest do komunikacji międzyprocesowej.

Kod źródłowy 4.2: Przetwarzanie żądania - procedura `process_req()`.

```

sub process_req(){
  my($r) = @_;
  $wyn = "";
  if ($r =~ /get/i) {
    @request = split("_", $r);
    $zad = $request[0];
    $ts1 = $request[1];
    $ts2 = $request[2];
    @date1 = split("/\D/", $ts1);
    @date2 = split("/\D/", $ts2);
    print "odebralem:_"$r;
    $wyn = $wyn."zadanie:_"$zad"\n";
    $wyn = $wyn."czas_od:_"$date1[0]."-"$date1[1]."-"$date1[2]."_"$date1[3]."":"$";
    $wyn = $wyn."czas_do:_"$date2[0]."-"$date2[1]."-"$date2[2]."_"$date2[3]."":"$";
    $wyn = $wyn.&sym_sens($ts1, $ts2);
    return $wyn;
  }
  if ($r =~ /set gt/i) {
    @request = split("_", $r);
    $zad = $request[0];
    $ts1 = $request[1];
    $ts2 = $request[2];
    $gt = $request[2];
    dbmopen(%granulacja, "granulacja_baza", 0644);
    $granulacja{"gt"}=$gt;
    dbmclose(%granulacja);
    $wyn = "'GT\'_zmienione_na:_"$gt;
  }
}

```


Instalacja i wdrożenie

W tym rozdziale należy omówić zawartość pakietu instalacyjnego oraz założenia co do środowiska, w którym realizowany system będzie instalowany. Należy przedstawić procedurę instalacji i wdrożenia systemu. Czynności instalacyjne powinny być szczegółowo rozpisane na kroki. Procedura wdrożenia powinna obejmować konfigurację platformy sprzętowej, OS (np. konfiguracje niezbędnych sterowników) oraz konfigurację wdrażanego systemu, m.in. tworzenia niezbędnych kont użytkowników. Procedura instalacji powinna prowadzić od stanu, w którym nie są zainstalowane żadne składniki systemu, do stanu w którym system jest gotowy do pracy i oczekuje na akcje typowego użytkownika.



Podsumowanie

W podsumowanie należy określić stan zakończonych prac projektowych i implementacyjnych. Zaznaczyć, które z zakładanych funkcjonalności systemu udało się zrealizować. Omówić aspekty pielęgnacji systemu w środowisku wdrożeniowym. Wskazać dalsze możliwe kierunki rozwoju systemu, np. dodawanie nowych komponentów realizujących nowe funkcje.

W podsumowaniu należy podkreślić nowatorskie rozwiązania zastosowane w projekcie i implementacji (niebanalne algorytmy, nowe technologie, itp.).



Bibliografia

- [1] Java technology. Web pages: <http://www.oracle.com/technetwork/java/>.
- [2] J. Cichoń, M. Klonowski, Łukasz Krzywiecki, B. Różański, P. Zieliński. On the number of nodes and their distribution in chord. 2006.
- [3] B. Frain. *Responsive Web Design with HTML5 and CSS3*. Packt Publishing, 2012.



Zawartość płyty CD

W tym rozdziale należy krótko omówić zawartość dołączonej płyty CD.

