

WOJSKOWA AKADEMIA TECHNICZNA

im. Jarosława Dąbrowskiego

WYDZIAŁ CYBERNETYKI



Technologie JEE

SPRAWOZDANIE

Autor:
**Karol
Baranowski**

Grupa:
K6C3S1

Prowadzący:
mgr inż. Łukasz Laszko

Warszawa 2019

1 Treść zadanie

TJEE, Projekt indywidualny, Zestaw 7.

Zadanie

- Wykonać aplikację do obliczania funkcji skrótu łańcucha znaków algorytmem SHAKE128.

Wymagania funkcjonalne:

- wyświetlanie historii do tej pory wygenerowanych skrótów (zapisanych w bazie danych),
- formularz do przeglądania, usuwania, dodawania oraz generowania skrótów.

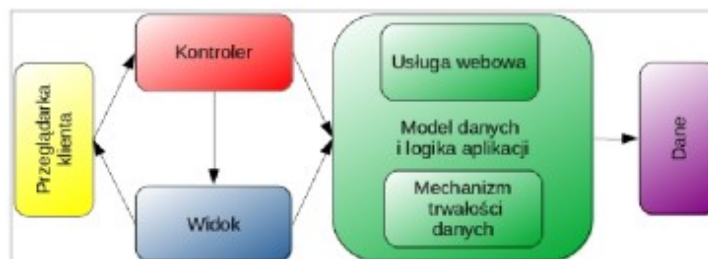
Wymagania pozafunkcjonalne:

- należy wykorzystać 2 usługi webowe,
- do obsługi bazy danych przygotować usługę webową nr 1 (dodawanie, usuwanie, pobieranie),
- do generowania nowych skrótów przygotować usługę webową nr 2,
- wygenerowany skrót należy przedstawić klientowi,
- klient może żądać zapisania wygenerowanego skrótu w bazie danych.


Wymagania technologiczne:

- za pomocą generatora udostępnionego na stronie przedmiotu należy wygenerować sobie zestaw technologii do wykorzystania w projekcie,
- w przypadku, gdy któryś z komponentów nie będzie mógł spełnić swojej funkcji, należy samodzielnie zastosować technologię wspomagającą (np. serwet),
- rozwiązanie należy przygotować do rozproszenia (uruchomienia w ramach różnych JVM, lub serwerów webowych),
- kod poszczególnych komponentów należy logicznie odseparować od siebie (np. różne projekty, różne pakiety, itp.).

Ogólny schemat rozwiązania



Rozliczenie

- Należy przygotować krótkie sprawozdanie. W sprawozdaniu ma się znaleźć między innymi informacja na temat: treści zadania, kodu projektu, kto wykonał zadanie, sposobu rozwiązania, wynikach uruchomienia, wykorzystanych specyficznych technologii i bibliotekach wraz z ich wersjami, strukturze repozytorium projektu (np. na diagramie wdrożenia) oraz pozostałe spostrzeżenia i uwagi. Repozytorium projektu i sprawozdanie należy umieścić w środowisku GitLab i przesłać na adres e-mail prowadzącego (lukasz.laszko@wat.edu.pl) informację o zakończeniu pracy na 2 dni przed ostatnimi zajęciami. Jednocześnie należy dodać prowadzącego  do projektu w GitLabie, w funkcji Reporter.

- Podczas ostatnich zajęć laboratoryjnych odbędzie się krótka obrona projektu.

- Za terminowe wykonanie zadania indywidualnego można uzyskać 10 pkt. Wyznaczenie każdego kolejnego, poprawkowego terminu zaliczenia zadania powoduje zmniejszenie o 2 maksymalnej, możliwej do uzyskania liczby punktów. Sposób oceniania jest następujący:

- 10 pkt. - realizacja wszystkich wymagań,
- niespełnienie jednego wymagania dotyczącego technologii komponentów (z generatora) – 1 pkt mniej,
- uzasadnione zastosowanie w projekcie nieomawianej technologii wytwórczej – 1 pkt więcej,
- pozostałe braki – 0,5 pkt mniej.

2 Wstęp

Parametry:

```
Technologia widoku: HTML+Servlet  
Technologia kontrolera: EJB Stateful  
Technologia usługi webowej: SOAP  
Technologia trwałości danych: ORM  
Kod projektu: k73012
```

W technologii widoku dodatkowo dodano jsp, który nie zastępuje servletu, a ułatwia przekazywanie do niego i wczytywanie od niego parametrów. Dodatkowo dodano CSS I JavaScript - w celach estetycznych oraz do zaimplementowania ozdobnego zegarka na stronie.

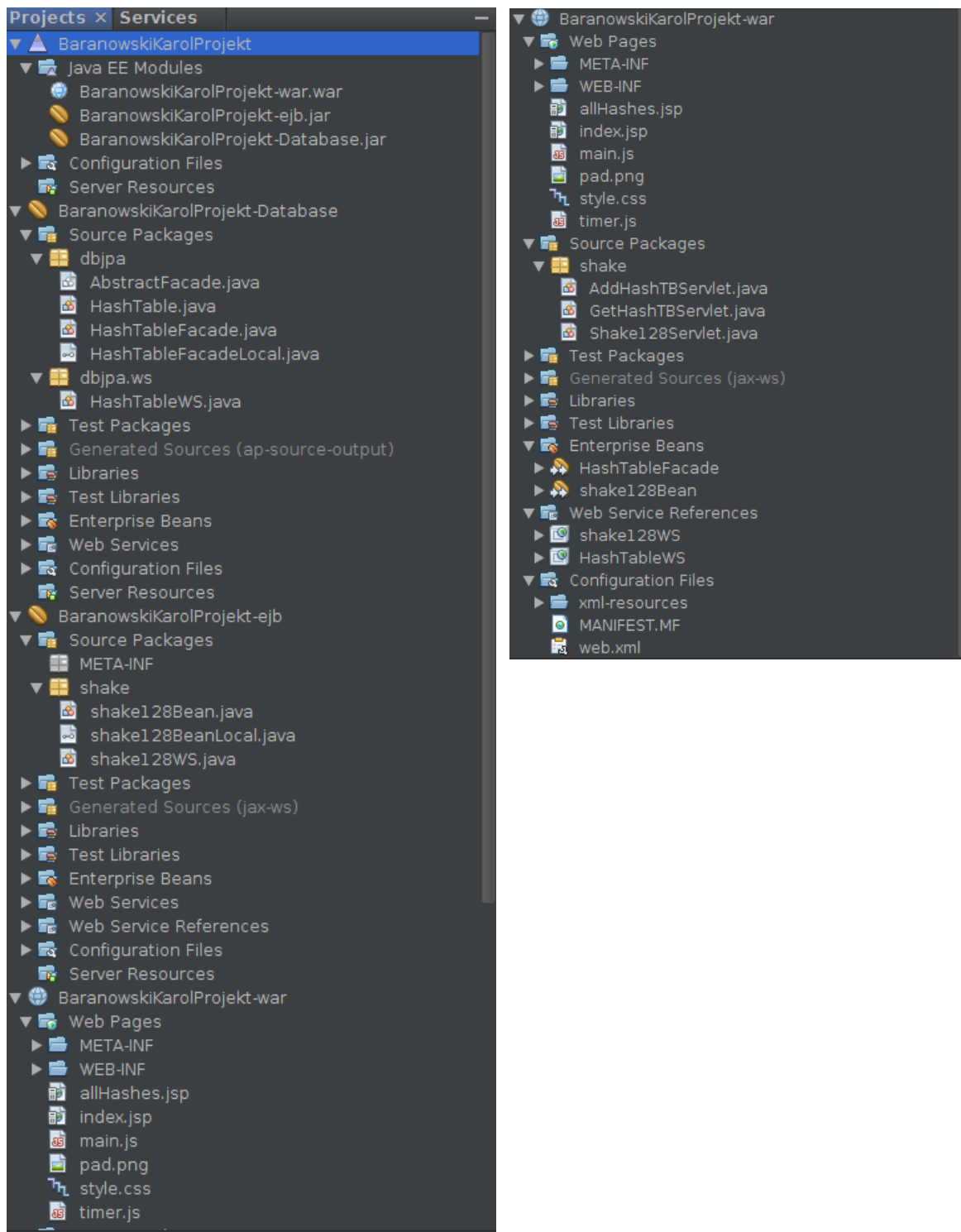
Technologię kontrolera EJB Stateful zamieniono na EJB Stateless z powodu SOAP, aby było można przysyłać składniki z Beana do usługi webowej.

W technologii ORM, która jest ogólnym mechanizmem dodatkowo wykorzystano JPA

Dodatkowo dodano CSS - w celach estetycznych oraz JavaScript - do wyliczenia shake128 po stronie klienta oraz do ozdobnego zegarka na stronie.

3 Struktura programu

Poniżej zamieszczono strukturę programu:

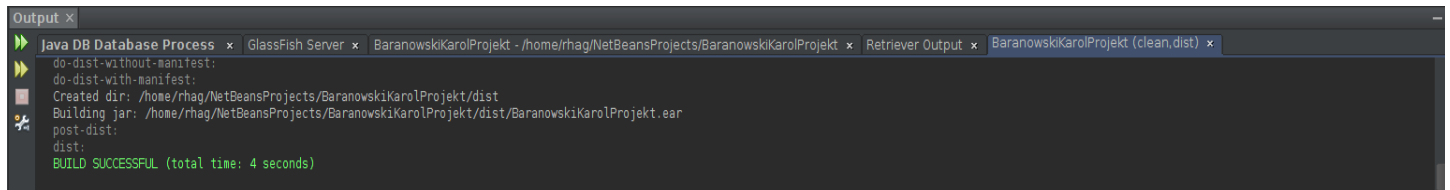


Jest to projekt Java Enterprise Edition, składający się z trzech głównych modułów EE:

- *BaranowskiKarolProjekt-war* – Zawierający pliki frontendowe, serwlety obsługujące dane i generujące wygląd przy użyciu własnych metod oraz usług webowych, ziarna Enterprise, referencje do web serwisów oraz pliki konfiguracyjne.
- *BaranowskiKarolProjekt-ejb* – Zawierający logikę przesyłu danych pobieranych ze strony i obsługę funkcji skrótów shake128 poprzez użycie Beana I usługi webowej.

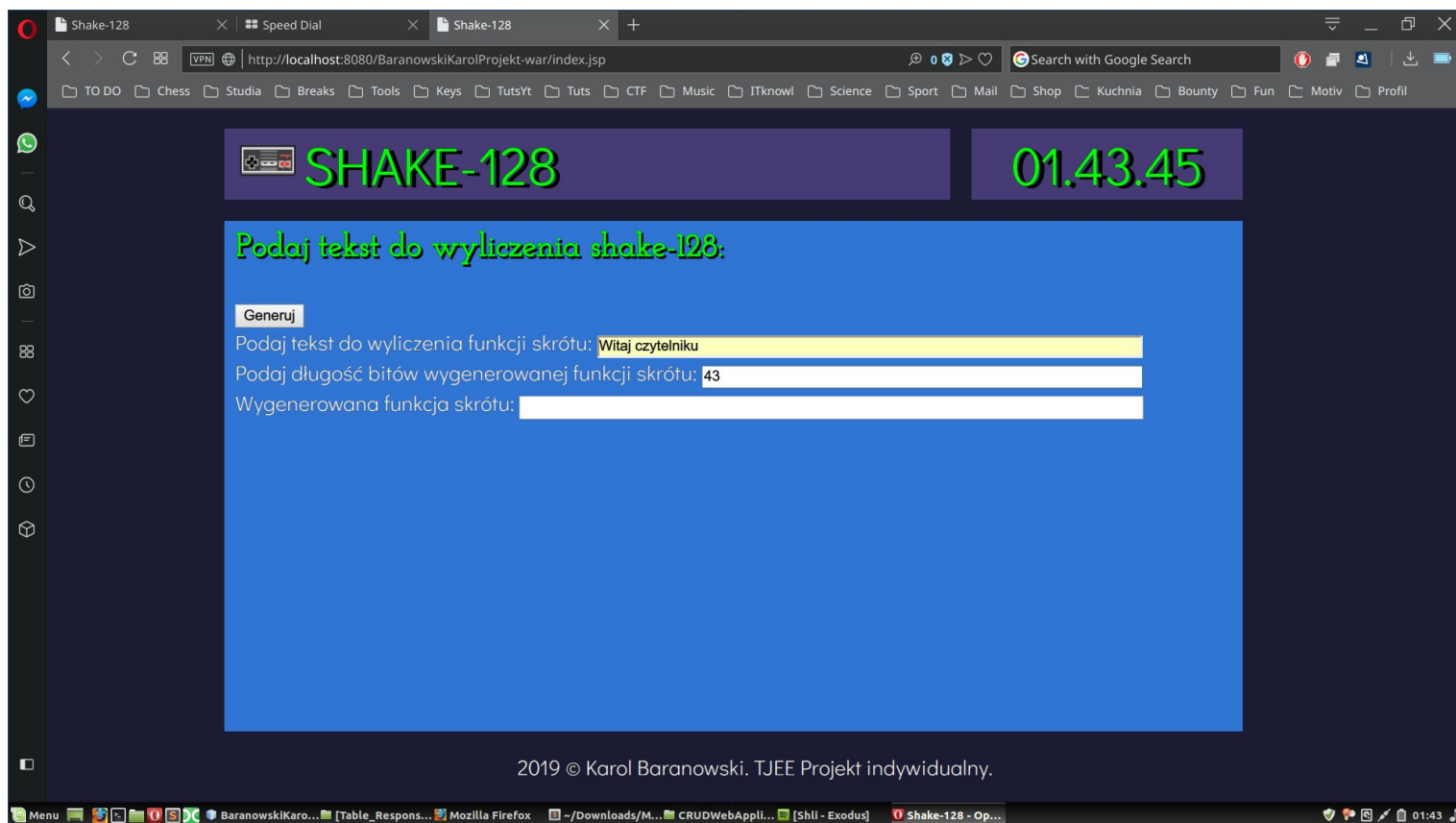
- *BaranowskiKarolProjekt-Database* – Zawierający obsługę bazy danych Java Persistence (JPA), który jest oficjalnym standardem mapowanie obiektowo – relacyjnego (ORM)

4 Działanie programu



```
Output X
Java DB Database Process x GlassFish Server x BaranowskiKarolProjekt - /home/rhag/NetBeansProjects/BaranowskiKarolProjekt x Retriever Output x BaranowskiKarolProjekt (clean, dist) x
do-dist-without-manifest:
do-dist-with-manifest:
Created dir: /home/rhag/NetBeansProjects/BaranowskiKarolProjekt/dist
Building jar: /home/rhag/NetBeansProjects/BaranowskiKarolProjekt/dist/BaranowskiKarolProjekt.ear
post-dist:
dist:
BUILD SUCCESSFUL (total time: 4 seconds)
```

Po pomyślnym zbudowaniu aplikacji i deploy'u na Glassfish 4.1 włącza się strona startowa index.jsp



Po wpisaniu wymaganych informacji można wygenerować skrót shake128 i pojawia się przycisk przejścia do strony zarządzania (w przesyłaniu informacji na stronę zarządzania allHashes.jsp bierze udział usługa webowa 2 z polecenia) .



SHAKE-128

01.44.12

Podaj tekst do wyliczenia shake-128:

Generuj

Podaj tekst do wyliczenia funkcji skrótu: Witaj czytelniku

Podaj długość bitów wygenerowanej funkcji skrótu: 43

Wygenerowana funkcja skrótu: 5ead4924f8

Przejdź do zarządzania

2019 © Karol Baranowski. TJEE Projekt indywidualny.



SHAKE-128

01.44.53

Podaj tekst do wyliczenia shake-128:

Tekst do wyliczenia funkcji skrótu: Witaj czytelniku

Długość bitów wygenerowanej funkcji skrótu: 43

Wygenerowana funkcja skrótu: 5ead4924f8

Dodaj

Podaj id w bazie:

Wyświetl Usun

Generuj nowy skrot

Baza funkcji skrótów:

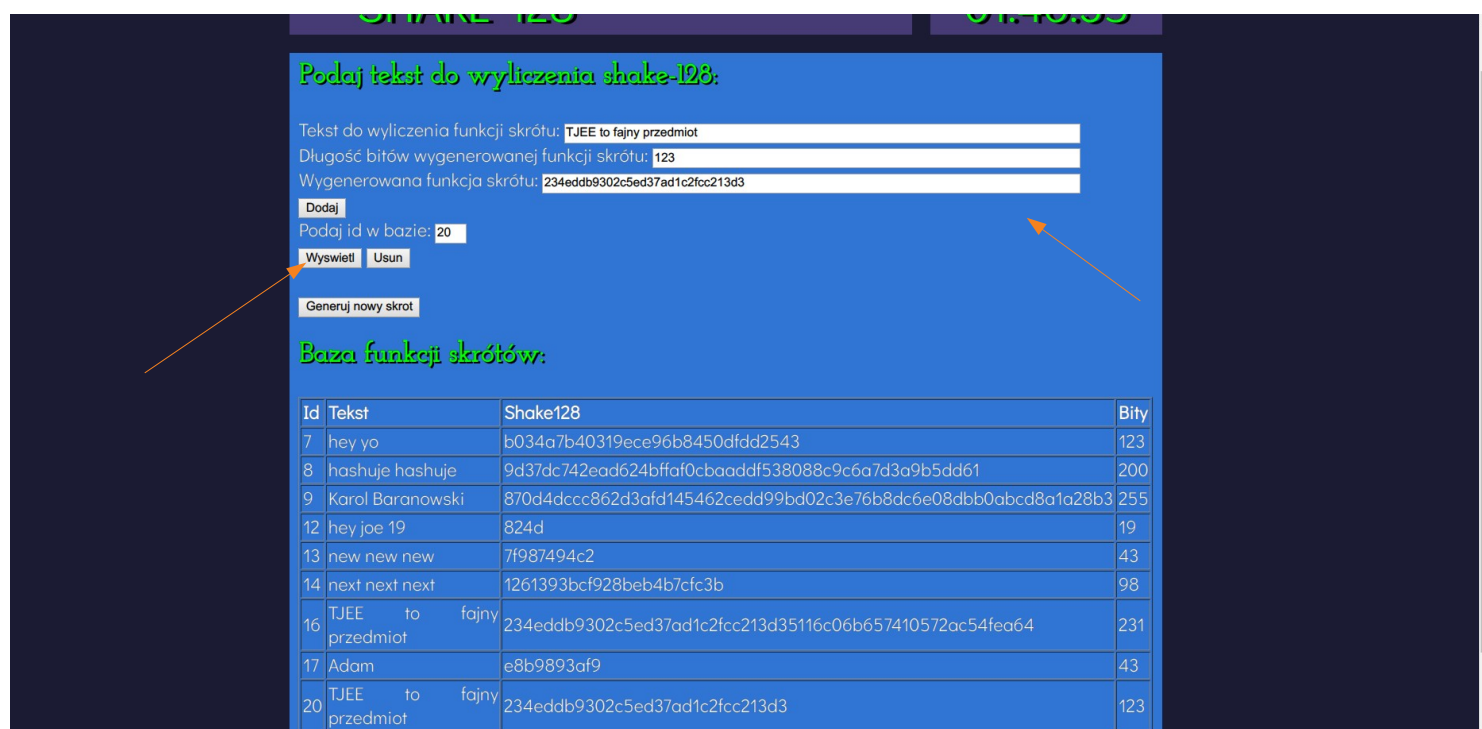
Id	Tekst	Shake128	Bity
7	hey yo	b034a7b40319ece96b8450dfdd2543	123
8	hashuje hashuje	9d37dc742ead624bffa0cbaaddf538088c9c6a7d3a9b5dd61	200
9	Karol Baranowski	870d4dccc862d3afd145462cedd99bd02c3e76b8dc6e08dbb0abcd8a1a28b3	255
12	hey joe 19	824d	19
13	new new new	7f987494c2	43
14	next next next	1261393bcf928beb4b7cfc3b	98
16	TJEE to fajny przedmiot	234eddb9302c5ed37ad1c2fcc213d35116c06b657410572ac54fea64	231
17	Adam	e8b9893af9	43
20	TJEE to fajny przedmiot	234eddb9302c5ed37ad1c2fcc213d3	123

2019 © Karol Baranowski. TJEE Projekt indywidualny.

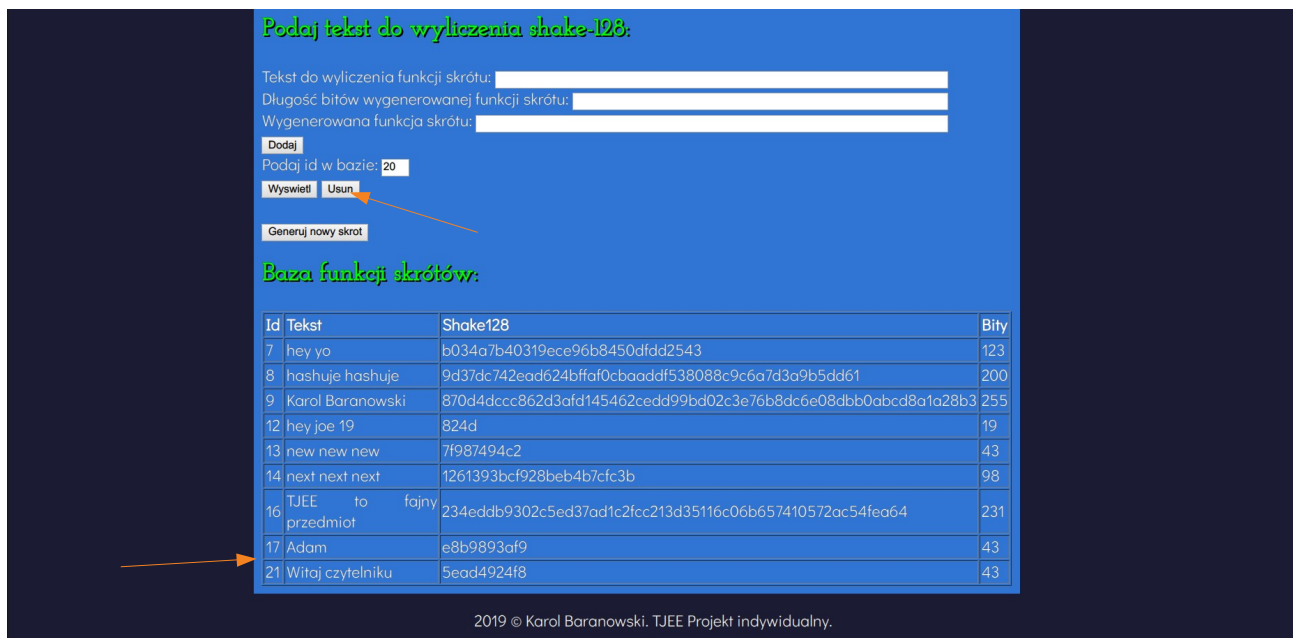
Jak widać jest to strona administracyjna, której komponenty generowane są przez różne serwlety używając metod usługi webowej 1 z polecenia odpowiedzialnej za obsługę bazy danych. W poniższych zrzutach ekranu zaprezentowane jest prawidłowe działanie aplikacji w kolejności: Dodanie wpisu, pobranie rekordu oraz usunięcie.



Dodanie



Pobranie



Usunięcie

Aplikacja dodatkowo ma zapewnioną walidację błędów

5 Działanie programu

Najważniejsze fragmenty programu:

```

13 | onload="odliczanie();"
14 | <div id="container">
15 |   <div class="rectangle1">
16 |     <div id="logo">SHAKE-128</div>
17 |
18 |     <div id="timer"></div>
19 |     <div style="clear: both;"></div>
20 |   </div>
21 |   <div class="square">
22 |     <div class="tile5">
23 |       <h3>Podaj tekst do wyliczenia shake-128:</h3>
24 |       <button onclick="displayResult();">Generuj</button>
25 |       <form action="Shake128Servlet" method="POST" name="f">
26 |
27 |         Podaj tekst do wyliczenia funkcji skrótu: <input type="text" name="tekst" style="width:500px;" ></br>
28 |         Podaj długość bitów wygenerowanej funkcji skrótu: <input type="text" name="l_bitow" style="width:403px;" ></br>
29 |         Wygenerowana funkcja skrótu: <input type="text" name="hash" style="width:572px;" readonly ></br>
30 |
31 |         <input name="sub" id="sub" type="submit" value="Przejdź do zarządzania" style="visibility: hidden;">
32 |       </form>
33 |     </div>
34 |     <script>
35 |       function displayResult() {
36 |         if(parseInt(document.f.l_bitow.value,10)>256 || parseInt(document.f.l_bitow.value,10)<8 || document.f.tekst
37 |           return
38 |         else{
39 |           document.f.hash.value = shake128(document.f.tekst.value, parseInt(document.f.l_bitow.value,10));
40 |           document.f.sub.style.visibility="visible";
41 |         }
42 |       }
43 |     </script>
44 |   </div>
45 |   <div class="rectangle">

```

przykładowy frontend z wywołaniem funkcji shake128(), która oblicza funkcję skrótu z zadanego tekstu po stronie klienta przeglądarki bez zbędnego obciążania serwera. Funkcja nie jest autorska, została sklonowana i zmodyfikowana do potrzeb zadania z :

<https://github.com/emn178/js-sha3> na licencji MIT od autorów : Chen, Yi-Cyuan .

SHAKE128 – jest odmianą kryptograficznej funkcji skrótu SHA-3 z 2012 roku.

Funkcja wykorzystuje algorytm Keccak, który ma architekturę "gąbki" (sponge construction) — bloki wejściowe są stopniowo "wchłaniane" w kolejnych etapach i mieszane z dużym rejestrem stanu. Blok wyjściowy jest konstruowany w podobny sposób, przez "wyciskanie" kolejnych fragmentów danych wyjściowych z rejestru stanu, wielokrotnie go mieszając pomiędzy wyciskanymi blokami. Wchłanianie i wyciskanie odbywa się na małej części rejestru stanu, poprzez wykonanie funkcji binarnej alternatywy wykluczającej (xor) z danymi wejściowymi, lub odczyt tej samej małej części przy odczycie danych wyjściowych. Pozostała część stanu nigdy nie jest bezpośrednio używana do konstrukcji danych wyjściowych, ani nie oddziałuje bezpośrednio z danymi wejściowymi.

Funkcja mieszająca stanu, składa się z wielokrotnej aplikacji funkcji rundy (do 24 razy w przypadku największej wersji algorytmu). Każda runda z kolei, składa się z kompozycji 5 prostych i wydajnych w implementacji funkcji, które dokonują odwracalnych permutacji, rotacji, mieszań albo dyfuzji. Ostatnia z tych funkcji w rundzie, dodatkowo jest parametryzowana stałą wartością zależną od numeru rundy (i wersji algorytmu), w celu usunięcia symetrii z funkcji rundy.

Instance	Output size d	Rate r = block size	Capacity c	Definition	Security strengths in bits		
					Collision	Preimage	2nd preimage
SHA3-224(M)	224	1152	448	Keccak[448](M 01, 224)	112	224	224
SHA3-256(M)	256	1088	512	Keccak[512](M 01, 256)	128	256	256
SHA3-384(M)	384	832	768	Keccak[768](M 01, 384)	192	384	384
SHA3-512(M)	512	576	1024	Keccak[1024](M 01, 512)	256	512	512
SHAKE128(M, d)	d	1344	256	Keccak[256](M 1111, d)	$\min(d/2, 128)$	$\geq \min(d, 128)$	$\min(d, 128)$
SHAKE256(M, d)	d	1088	512	Keccak[512](M 1111, d)	$\min(d/2, 256)$	$\geq \min(d, 256)$	$\min(d, 256)$

With the following definitions

- $\text{Keccak}[c](N, d) = \text{sponge}[\text{Keccak-f}[1600], \text{pad}10^*1, r](N, d)^{[24]:20}$
- $\text{Keccak-f}[1600] = \text{Keccak-p}[1600, 24]^{[24]:17}$
- c is the capacity
- r is the rate = $1600 - c$
- N is the input bit string

SHAKE 128 jest tzw, XOF'em (Extendable Output Function), czyli taką, w której można określić długość bitów na skrótu na wyjściu o bezpieczeństwie mierzonym w bitach na 128.

Wykorzystanie metod usługi webowej 2 do generowania parametrów:

```
Start Page x index.jsp x main.js x shake128Bean.java x Shake128Servlet.java x
Source History
19 @WebServiceRef(wsdlLocation = "WEB-INF/wsdl/localhost_8080/HashTableWS/HashTableWS.wsdl")
20 private HashTableWS_Service service_1;
21
22 @WebServiceRef(wsdlLocation = "http://localhost:8080/Shake128WS/Shake128WS?wsdl")
23
24 private Shake128WS_Service service;
25
26 String tekst;
27 String hash;
28 int bits;
29 int id;
30
31 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
32     throws ServletException, IOException {
33
34     tekst=request.getParameter("tekst");
35     hash=request.getParameter("hash");
36     bits=Integer.parseInt(request.getParameter("l_bitow"));
37     setText(tekst);
38     setHash(hash);
39     setBits(bits);
40
41     request.setAttribute("tekst", getText());
42     request.setAttribute("l_bitow", getBits());
43     request.setAttribute("hash", getHash());
44     request.setAttribute("allHashes", getAllHashTB());
45     request.getRequestDispatcher("allHashes.jsp").forward(request, response);
46 }
47
48 HttpServlet methods. Click on the + sign on the left to edit the code.
86
87 private void setText(java.lang.String text) {
```

Wykorzystanie metod usługi webowej 1 do obsługi bazy danych:

```
Start Page x index.jsp x main.js x shake128Bean.java x Shake128Servlet.java x GetHashTBServlet.java x
Source History
13 public class GetHashTBServlet extends HttpServlet {
14
15     @WebServiceRef(wsdlLocation = "WEB-INF/wsdl/localhost_8080/HashTableWS/HashTableWS.wsdl")
16     private HashTableWS_Service service;
17
18
19     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
20         throws ServletException, IOException {
21         String action = request.getParameter("action");
22         String IDstring = request.getParameter("id");
23         int ID=0;
24         if(IDstring!=null && !IDstring.equals("")){
25             if(isNumeric(IDstring))
26                 ID=Integer.parseInt(IDstring);
27         }
28         for(HashTable hashTable: getAllHashTB()){
29             if(hashTable.getId()==ID){
30                 hashTable = getHashTB(ID);
31                 if("Wyświetl".equalsIgnoreCase(action)){
32                     request.setAttribute("id", hashTable.getId());
33                     request.setAttribute("tekst", hashTable.getTekst());
34                     request.setAttribute("l_bitow", hashTable.getBity());
35                     request.setAttribute("hash", hashTable.getShake128());
36                 }else if(action.equalsIgnoreCase("Usun"))
37                     removeHashTB(hashTable);
38             }
39         }
40         request.setAttribute("allHashes", getAllHashTB());
41         request.getRequestDispatcher("allHashes.jsp").forward(request, response);
42     }
43     public static boolean isNumeric(String str){
44         for (char c : str.toCharArray())
```

```
AddHashTBServlet.java
Source History
10 import shake.DBclient.HashTable;
11 import shake.DBclient.HashTableWS_Service;
12
13
14 public class AddHashTBServlet extends HttpServlet {
15
16     @WebServiceRef(wsdlLocation = "WEB-INF/wsdl/localhost_8080/HashTableWS/HashTableWS.wsdl")
17     private HashTableWS_Service service;
18
19     @Override
20     protected void doGet(HttpServletRequest request, HttpServletResponse response)
21         throws ServletException, IOException {
22         String tekst = request.getParameter("tekst");
23         String hash = request.getParameter("Shake128");
24         String bityStr = request.getParameter("l_bitow");
25
26         if(tekst!=null && !tekst.equals("") && hash!=null && !hash.equals("") && bityStr!=null && !bityStr.equals("")){
27             HashTable hashTable = new HashTable();
28             hashTable.setShake128(hash);
29             hashTable.setTekst(tekst);
30             hashTable.setBity(Integer.parseInt(bityStr));
31             addHashTB(hashTable);
32         }
33         request.setAttribute("allHashes", getAllHashTB());
34         request.getRequestDispatcher("allHashes.jsp").forward(request, response);
35     }
36     @Override
37     protected void doPost(HttpServletRequest request, HttpServletResponse response)
38         throws ServletException, IOException {
39
40     }
41     @Override
42     public String getServletInfo() {
```

Klasa zawierająca zapytania do bazy danych:

```
16 |
17 | @Entity
18 | @Table(name = "HashTable")
19 | @XmlRootElement
20 | @NamedQueries({
21 |     @NamedQuery(name = "HashTable.findAll", query = "SELECT h FROM HashTable h")
22 |     , @NamedQuery(name = "HashTable.findById", query = "SELECT h FROM HashTable h WHERE h.id = :id")
23 |     , @NamedQuery(name = "HashTable.findByTekst", query = "SELECT h FROM HashTable h WHERE h.tekst = :tekst")
24 |     , @NamedQuery(name = "HashTable.findByShake128", query = "SELECT h FROM HashTable h WHERE h.shake128 = :shake128")
25 |     , @NamedQuery(name = "HashTable.findByBity", query = "SELECT h FROM HashTable h WHERE h.bity = :bity"))
26 | public class HashTable implements Serializable {
27 |
28 |     private static final long serialVersionUID = 1L;
29 |     @Id
30 |     @GeneratedValue(strategy = GenerationType.IDENTITY)
31 |     @Basic(optional = false)
32 |     @Column(name = "id")
33 |     private Integer id;
34 |     @Size(max = 300)
35 |     @Column(name = "Tekst")
36 |     private String tekst;
37 |     @Size(max = 300)
38 |     @Column(name = "Shake128")
39 |     private String shake128;
40 |     @Column(name = "Bity")
41 |     private Integer bity;
42 |
43 |     public HashTable() {
44 |     }
45 |
46 |     public HashTable(Integer id) {
47 |         this.id = id;
48 |     }
49 | }
```

Przykładowe metody usługi webowej do zarządzania bazą:

```
10 | import javax.ejb.Stateless;
11 |
12 |
13 | @WebService(serviceName = "HashTableWS")
14 | @Stateless()
15 | public class HashTableWS {
16 |
17 |     @EJB
18 |     private HashTableFacadeLocal hashTableFacade;
19 |
20 |     @WebMethod(operationName = "addHashTB")
21 |     public void addHashTB(@WebParam(name = "hashTB") HashTable hashTB){
22 |         hashTableFacade.create(hashTB);
23 |     }
24 |
25 |     @WebMethod(operationName = "removeHashTB")
26 |     public void removeHashTB(@WebParam(name = "hashTB") HashTable hashTB){
27 |         hashTableFacade.remove(hashTB);
28 |     }
29 |
30 |     @WebMethod(operationName = "getHashTB")
31 |     public HashTable getHashTB(@WebParam(name = "id") int id){
32 |         return hashTableFacade.find(id);
33 |     }
34 |
35 |     @WebMethod(operationName = "getAllHashTB")
36 |     public List<HashTable> getAllHashTB(){
37 |         return hashTableFacade.findAll();
38 |     }
39 | }
40 |
```

5 Podsumowanie i wnioski

Zadanie zostało zrealizowane pomyślnie, chociaż było pracochłonne i wymagało przekrojowej wiedzy z całego przedmiotu. Poza trudnościami w programowaniu czasochłonne było tutaj zsynchronizowanie wszystkich komponentów w Netbeansie i skonfigurowanie go, wszystkich plików konfiguracyjnych, serwera Glassfish oraz naprawienie wielu nieprzewidzianych problemów z tym IDE.

- JPA jest bardzo dobrą technologią, dzięki której można wygenerować wiele kodu i całkowicie pominąć pisanie zapytań sql a do rekordów z bazy szybko odnosić się jako do obiektów na liście.
- Użycie web metod / web serwisów bardzo ułatwia pracę, ponieważ najczęściej są to w miarę proste metody prostą biznesową logikę komponentów EJB, których można bardzo szybko użyć w każdej klasie czy serwlecie w projekcie poza tym zdalnie gdyż w uproszczeniu web service to mechanizm umożliwiający wywołanie jakiejś funkcjonalności za pośrednictwem internetu.