



**AGH**

**AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE**

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,  
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA INFORMATYKI STOSOWANEJ

Praca magisterska

*System rozstrzygania konkursów internetowych  
Web contest winner determination system*

Autor:

*Tomasz Szepczyński*

Kierunek studiów:

*Informatyka*

Opiekun pracy:

*dr Paweł Skrzyński*

Kraków, 2015

*Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.*

*Serdecznie dziękuję wszystkim za wsparcie*



## Spis treści

<b>1. Wprowadzenie</b>	9
1.1. Cele pracy	9
1.2. Dotychczasowe dostępne rozwiązania	9
1.3. Zawartość pracy	10
<b>2. Źródła danych</b>	11
2.1. Facebook	11
2.1.1. Połączenie	11
2.1.1.1 Stworzenie aplikacji Facebook'owej	11
2.1.1.2 Tokeny	12
2.1.1.3 Proces generowania tokenów	12
2.1.2. Dostępne SDK	12
2.1.3. Możliwości API	14
2.2. Twitter	14
2.2.1. Połączenie	15
2.2.1.1 Stworzenie aplikacji na Twitterze	15
2.2.1.2 Tokeny	15
2.2.1.3 OAuth	15
2.2.2. Dostępne SDK	15
2.2.3. Możliwości API	16
<b>3. Algorytmy losowania</b>	17
3.1. Problem losowości w komputerach	17
3.1.1. Generatory LCG	18
3.2. Algorytmy	19
3.2.1. Algorytm prostego losowania	19
3.2.1.1 Działanie	19
3.2.2. Algorytm losowania z uwzględnieniem wag - Ruletka	21
3.2.2.1 Funkcja przystosowania	21

3.2.2.2	Działanie . . . . .	22
3.2.3.	Algorytm losowania z uwzględnieniem wag - Turniej .....	23
3.2.3.1	Działanie . . . . .	23
3.3.	Testy wydajnościowe.....	24
3.3.1.	Algorytmy progresywne .....	24
3.3.2.	Algorytm Ruletki .....	25
3.3.3.	Algorytm Turniejowy.....	26
<b>4.</b>	<b>Projekt - Sposoby prezentacji losowania .....</b>	<b>29</b>
4.1.	Wizualizacje .....	29
4.1.1.	Wyścig.....	29
4.1.2.	Słupki .....	29
4.1.3.	Liczby.....	29
4.1.4.	Ruletka .....	30
4.1.5.	Turniej .....	30
4.2.	Komunikacja z GUI.....	31
4.2.1.	Long polling .....	31
4.2.2.	Dalszy rozwój w WebGL .....	31
4.2.2.1	Możliwości WebGL . . . . .	32
4.2.2.2	Problemy z WebGL . . . . .	32
<b>5.</b>	<b>Projekt - możliwości GUI .....</b>	<b>35</b>
5.1.	Użytkownik.....	35
5.1.1.	Przypadki użycia .....	35
5.1.2.	Single Sign-On.....	35
5.1.3.	Logowanie.....	36
5.1.4.	Tworzenie wydarzenia .....	37
5.1.5.	Obejrzenie wydarzenia.....	38
5.2.	Administrator.....	39
5.2.1.	Przypadki użycia .....	39
5.2.2.	Panel administracyjny .....	39
5.2.3.	Panel wiadomości .....	39
<b>6.</b>	<b>Projekt - architektura aplikacji .....</b>	<b>45</b>
6.1.	MVC .....	45
6.2.	Diagram warstw.....	46
6.3.	Diagram komunikacji .....	46

6.4.	Aplikacja w chmurze .....	47
6.5.	Bezpieczeństwo aplikacji .....	49
6.5.1.	Popularne ataki.....	49
6.5.2.	Obrona przed atakami .....	49
6.5.2.1	SQL Injection . . . . .	49
6.5.2.2	XSS . . . . .	50
6.5.2.3	DDoS . . . . .	50
6.6.	Wykorzystane technologie.....	51
6.6.1.	Angular.....	52
6.6.2.	Yeoman.....	52
6.6.3.	Grunt .....	52
6.6.4.	Bower .....	52
6.6.5.	HTML/CSS/LESS/jQuery .....	52
6.6.6.	Nodejs .....	53
6.6.7.	Npm.....	53
6.6.8.	Protractor.....	53
6.6.9.	Karma.....	53
6.6.10.	Spring .....	53
6.6.11.	Hibernate.....	53
6.6.12.	QueryDSL .....	54
6.6.13.	Jackson .....	54
6.6.14.	MySQL.....	54
<b>7.</b>	<b>Podsumowanie.....</b>	<b>55</b>





# 1. Wprowadzenie

Pomysł na System Rozstrzygania Konkursów Internetowych zrodził się z narastającego popytu użytkowników internetu domagających się uczciwości w przeprowadzanych konkursach. Kiedy ktoś tworzy internetowe wydarzenie, w którym jest nagroda do wygrania użytkownicy nie mają możliwości zweryfikowania, czy konkurs odbył się zgodnie z zasadami fair-play.

Często takie konkursy mogą być fałszowane, a nagrody przekazywane znajomym. Przykładowo takim wydarzeniem może być konkurs na Facebook'u polegający na polubieniu wydarzenia i wybraniu z nich osób, które wezmą udział w losowaniu. Twórcy takich wydarzeń udostępniają jedynie na koniec wydarzenia listę zwycięzców.

## 1.1. Cele pracy

Celem pracy jest stworzenie Systemu Rozstrzygania Konkursów Internetowych, czyli aplikacji internetowej, która zapewni możliwość śledzenia danego losowania. Gwarancją zasad fair-play jest obarczenie algorytmem losowania niezależnego serwisu internetowego.

Aplikacja składa się z serwisu www utrzymywanego na odseparowanym od miejsca utworzenia wydarzenia serwerze. System również zawiera warstwę kliencką GUI, zajmującą się utworzeniem konkursu na podstawie dostępnych źródeł danych: Facebook oraz Twitter.

## 1.2. Dotychczasowe dostępne rozwiązania

Przeprowadzony został rekonesans w zakresie dotychczasowych dostępnych rozwiązań online. Znalaziono kilka dostępnych systemów:

- <http://www.strutta.com/>[1]
- <http://wpcontestcreator.com/>[2]
- <http://heyo.com/>[3]

Rozwiązania te są płatne oraz nie uwzględniają możliwości skorzystania z już utworzonego wydarzenia, np. zbierając liczbę polubień.

### 1.3. Zawartość pracy

W rozdziale 2 przedstawiono rekonesans w zakresie dostępnych źródeł danych do tworzenia konkursów: Facebook oraz Twitter. Przedstawiono również możliwości SDK wyodrębnionego dla programistów oraz wyjaśnione zostały także sposoby działania token'ów.

W rozdziale 3 przedstawione zostały możliwe algorytmy przeprowadzania symulacji. Poruszony zostaje problem pseudolosowości w komputerach oraz wachlarz dostępnych algorytmów wraz z zagadnieniami optymalizacji.

W rozdziale 4 przedstawione zostają możliwości aplikacji klienckiej w zakresie wyświetlania animacji wydarzenia. Wyjaśniony zostaje system komunikacji pomiędzy warstwą GUI a serwisem http.

W rozdziale 5 przedstawiono możliwości warstwy prezentacji - GUI. Zostaje wyjaśniona interakcja z zewnętrznymi api: Facebook API oraz Twitter API poprzez Simple Sign On[4]. Zostały wyodrębnione najciekawsze przypadki użycia serwisu z warstwy prezentacji.

W rozdziale 6 przedstawione zostają diagramy architektury serwisu, czyli podział na warstwy. Omówione zostają zalety i wady rozwiązań dotyczących komunikacji wraz z wykorzystanymi wzorcami oraz technologiami. Zostają również przedstawione problemy współczesnych aplikacji internetowych ze względu na bezpieczeństwo wraz z możliwymi ataki oraz sposobami ochrony przed nimi.

## 2. Źródła danych

Poniższy rozdział opisuje dostępne źródła danych, z których można czerpać informacje dotyczące wydarzeń.

Przeprowadzono rekonesans w sprawie dostępnych zewnętrznych platform, z których można wyłuskiwać informacje dotyczące utworzonych wydarzeń konkursowych. Wybrano dwie najpopularniejsze platformy, na których użytkownicy mają możliwość tworzenia konkursów :

- Facebook - `www.facebook.com`[5]
- Twitter - `www.twitter.com`[6]

Każda z nich dostarcza SDK programistom, udostępniając swoje API. W aplikacji wykorzystano głównie komunikację klient-serwer na bazie stylu architektury oprogramowania Representational State Transfer (REST) [7].

### 2.1. Facebook

Facebook dostarcza dla programistów potężne narzędzie zwane Graph API [5], które pozwala na transfer danych między Facebookiem a klientem tegoż API. Jest niskopoziomowym komponentem http bazującym na systemie zapytań. Działanie przebiega na podstawie komunikacji żądanie-odpowiedź, a więc preparowania odpowiednich zapytań. W odpowiedzi dostajemy dane w postaci obiektów - węzłów. W Systemie Rozstrzygania Konkursów Internetowych wykorzystano wersję Graph API 2.2, która posiada interfejs zorientowany obiektowo, w przeciwieństwie do wersji 1.x bazującej na strukturalnym stylu programowania[5].

#### 2.1.1. Połączenie

W celu integracji aplikacji z Facebook’iem należy poczynić kilka kroków, aby udało się wykonać jakiegokolwiek zapytania do Graph API.

##### 2.1.1.1. Stworzenie aplikacji Facebook’owej

Pierwszym krokiem jest stworzenie aplikacji deweloperskiej na Facebooku. Jest to niezbędne, gdyż dla niej będzie generowany token.

### 2.1.1.2. Tokeny

Tokeny dostępu są łańcuchami znaków, które identyfikują aplikację i mogą zostać wykorzystane do wywołań Graph API. Są tymczasowym niezbędnym narzędziem gwarantującym bezpieczeństwo. Facebook SDK dostarcza kilka typów tokenów:

- User Access Token - najczęściej wykorzystywany token - otrzymywany jest podczas okna dialogowego logowania.
- App Access Token - modyfikuje i odczytuje ustawienia aplikacji. Może służyć publikacji akcji Open Graph. Jest generowany podczas konfiguracji aplikacji.
- Page Access Token - podobny token do User Access Token, lecz gwarantuje dostęp do zapisu, odczytu danych należących do strony. Otrzymać go można przez Graph API.
- Client Token - jest to token, który można osadzić w aplikacjach desktopowych lub kodzie binarnym mobilnej aplikacji natywnej. Nie jest utajniony oraz posiada sporo ograniczeń.

### 2.1.1.3. Proces generowania tokenów

Aby uzyskać token należy wykonać zapytanie metodą GET do Facebook'a (Rys. 2.1):

```
GET /oauth/access_token?  
    client_id={app-id}  
    &client_secret={app-secret}  
    &grant_type=client_credentials
```

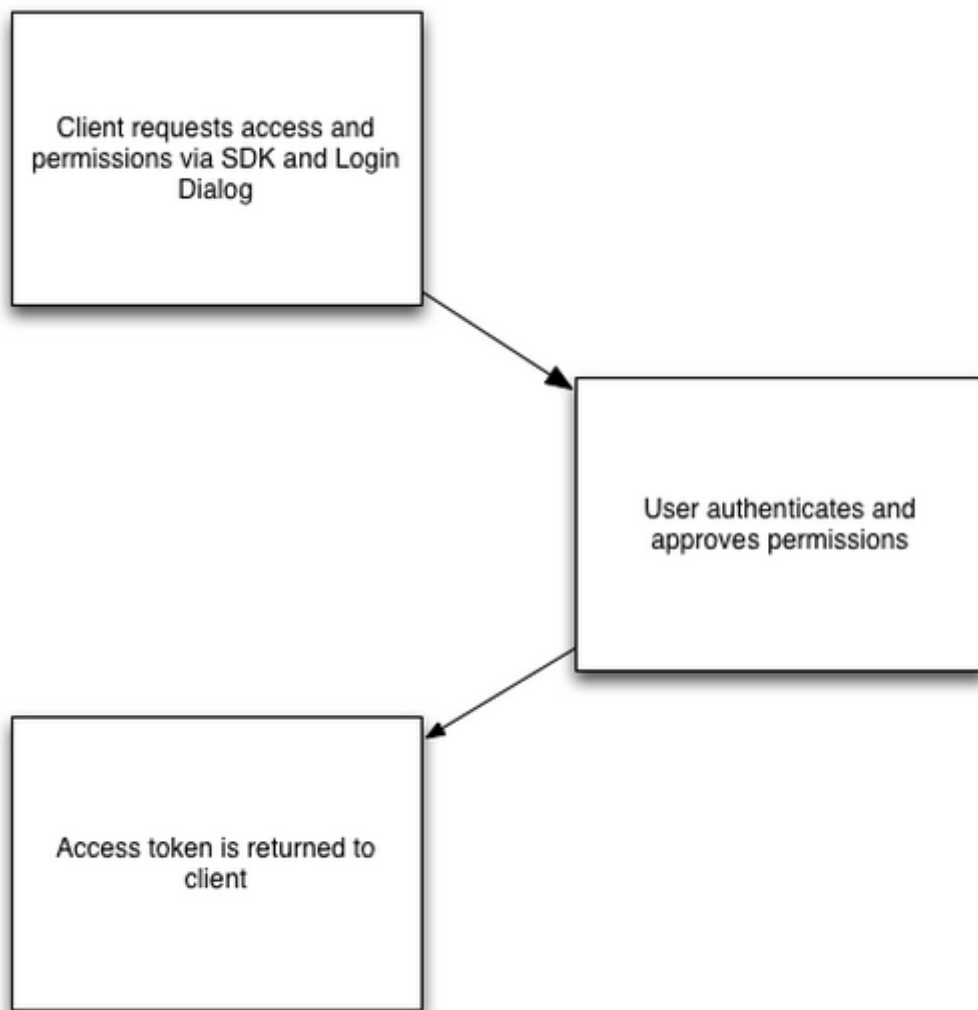
Rysunek 2.1: Zapytanie o token

Teraz aplikacja może dowolnie korzystać z Graph API. Otrzymanego sekretnego tokenu aplikacji nie należy udostępniać nikomu, ponieważ jest to prywatny łańcuch znaków przeznaczony tylko dla niej[5]. Każda platforma posiada różne API do generowania tokenów, jednak wszystkie działają na podstawie jednej strategii zaprezentowanej na Rys. 2.2.

### 2.1.2. Dostępne SDK

Facebook dostarcza następujące oficjalne SDK:

- Facebook SDK for iOS
- Facebook SDK for Android
- Facebook SDK for Unity
- Facebook SDK for JavaScript



Rysunek 2.2: Strategia tworzenia tokenu

- Facebook SDK for PHP
- Facebook Ads SDK for PHP

oraz wiele nieoficjalnych, w tym dla języka Java:

- Java (Spring) - Spring Social
- Java (Blackberry) - RIM
- Kinvey - Kinvey
- RestFB - Mark Allen

W celach obarczenia odpowiedzialnością losowania serwer, a nie klienta wykorzystano w aplikacji nieoficjalne rozwiązanie RestFB.

### 2.1.3. Możliwości API

Graph API jest reprezentacją informacji składającą się z:

- Węzłów - obiekty takie jak użytkownik, zdjęcie itd
- Krawędzi - połączenia między obiektami
- Pól - informacji na temat obiektów takie jak np. data urodzenia użytkownika

Graph API dostarcza wszelkie informacje za pomocą metod HTTP GET oraz POST, które są wymierzone w serwer `graph.facebook.com`[5]. Posiadając identyfikator danego węzła, czy połączenia można budować najróżniejsze zapytania. Przykładowe żądanie zostało przedstawione na Rys.2.3

```
POST graph.facebook.com
/me/feed?
  message="Hello, World."&
  access_token={your-access-token}
```

Rysunek 2.3: Przykładowe zapytanie do Graph API

## 2.2. Twitter

Twitter posiada kilka pakietów programistycznych, które pozwalają na wykorzystywanie jego zasobów[8].

- Fabric - modularny zestaw narzędzi programistycznych "Kits"
- Twitter for Websites - zestaw osadzalnych widget'ów oraz narzędzi przeznaczonych dla skryptów
- Cards - służy wyświetlaniu dodatkowych danych razem z Tweet'em
- OAuth - narzędzie do autoryzacji oraz uwierzytelniania
- REST APIs - programistyczny dostęp do zapisu oraz odczytu danych z Twitter'a
- Streaming APIs - służy do nasłuchiwanie na zdarzenia Twitterowe i reagowanie na nie
- Ads API - służy integrowaniem aplikacji z reklamami
- MoPub - zaawansowane narzędzie do integracji aplikacji z reklamami

W aplikacji ze względu na ograniczenia dotyczące przesyłania danych nakładane przez Twitter zamiast streaming API wykorzystano REST API.

### 2.2.1. Połączenie

Przed możliwością wykorzystania Twitter API należy poczynić kilka kroków, aby udało się wykonać jakiegokolwiek zapytania.

#### 2.2.1.1. Stworzenie aplikacji na Twitterze

Pierwszym krokiem do wykonania jest stworzenie aplikacji na Twitterze w celu uzyskania niezbędnych kluczy autoryzacyjnych.

#### 2.2.1.2. Tokeny

Aplikacja dostarcza cztery wygenerowane klucze

- Consumer Key (API Key)
- Consumer Secret (API Secret)
- Access Token
- Access Token Secret

Wszystkie są wykorzystywane w aplikacji klienckiej używającej Twitter API.

#### 2.2.1.3. OAuth

Twitter API v1.1 dostarcza dwie formy uwierzytelniania:

- Application-user - jest to powszechna forma uwierzytelniania. Sygnowane Żądanie identyfikuje aplikację mając dodatkowo prawa dostępu do wykonywania zapytań w kontekście użytkownika do którego należy token dostępu.
- Application-only - jest to forma uwierzytelniania, gdzie aplikacja wykonuje żądania w swoim imieniu bez kontekstu użytkownika. Ten sposób zezwala na mniejsze ograniczenia dotyczące wykorzystywania zasobów niż forma Application-user, jednak niektóre API nie posiadają dla niego wsparcia.

### 2.2.2. Dostępne SDK

Twitter dostarcza programistom dwa SDK :

- Hbc - klient dla języka Java przeznaczony dla Stream API
- Twitter-kit-android - wielomodułowy projekt zawierający TweetComposer, TwitterCore i TweetUI

Powstało wiele nieocifjalnych bibliotek dla różnych języków programowania. Jedną z nich jest Twitter4J stworzona przez człowieka sygnującego się pseudonimem "yusuke".

Cechy biblioteki Twitter4J:

- Stworzona w 100% w języku Java dla JRE > 1.5
- Nie posiada dodatkowych zależności do innych projektów
- Wspiera OAuth
- Kompatybilna z Twitter API 1.1

W projekcie wykorzystano wersję Twitter4J 4.0.4, działającą na licencji Apache License 2.0.

### 2.2.3. Możliwości API

Wykorzystane w aplikacji REST API posiada wiele ograniczeń dotyczących wykonywanych żądań GET dla danego token'u. Interwałem czasowym jest 15 minut, po którym pula startuje od nowa [8]. Różne URL posiadają inne ograniczenia.

Przykładowo zapytanie GET o pozostały limit: `application/rate_limit_status` zezwala na 180 zapytań, lecz już dla wyłuskania retweet'ów: `statuses/retweets/:id` liczba ta wynosi tylko 60. Dla URL wyszukiwających, tzn. zawierających w adresie `/search/` i formy uwierzytelniania Application-user liczba zapytań wynosi 180, zaś dla Application-only jest to 450. W przypadku przekroczenia tego limitu zapytanie zwraca kod 429 "Too many Requests".

Zapytania są wykonywane pod adres `https://api.twitter.com/1.1/`. Przykładowe zapytanie wyłuskujące najnowsze tweety: `https://api.twitter.com/1.1/search/tweets.json`.



## 3. Algorytmy losowania

### 3.1. Problem losowości w komputerach

Liczba losowa (ang. random number) jest liczbą  $r$  należącą do pewnego zbioru wartości (wzór 3.1)

$$\{r_1, \dots, r_n\} \quad (3.1)$$

wybieraną z pewnym prawdopodobieństwem. Jeśli jako  $r$  może pojawić się każda z liczb zbioru z tym samym prawdopodobieństwem  $p(r) = 1/n$ , to mówimy o równomiernym rozkładzie prawdopodobieństwa liczb losowych z tego zbioru. Na przykład rozważając rzut kostką. Każdy rzut daje liczbę losową  $r$  ze zbioru  $\{1, 2, 3, 4, 5, 6\}$ . Jeśli kostka nie jest wadliwa, to każda z możliwych wartości  $r$  pojawia się w rzucie kostką z prawdopodobieństwem  $p(r) = 1/6$ . Liczby losowe  $r$  posiadają zatem równomierny rozkład prawdopodobieństwa. Problem z otrzymaniem liczb losowych wynika z deterministycznego charakteru komputera i wykonywanych przez niego operacji. Gdy człowiek dokonuje rzutu kością, nie wie co wypadnie. Taka sama operacja na komputerze wymaga działania, którego wynik jest nieprzewidywalny, wszak żadna z operacji wykonywanych przez procesor nie posiada takiej cechy (o ile procesor jest sprawny).

Problem starano się rozwiązać wykorzystując zewnętrzne źródła sygnałów losowych (np. generatory białego szumu), jednakże w sprzęt tego typu nie są standardowo wyposażone komputery osobiste – należałoby wspomnieć o próbach wykorzystania szumów kart graficznych, jednakże system ten nie rozpowszechnił się z prostej przyczyny – różne karty dźwiękowe szumią w innych sposób, a te z górnej półki nie szumią prawie wcale. Dlatego osiągnięte wyniki nie były jednakowe i pomysł zarzucono.

Z drugiej strony liczby losowe używane są powszechnie przy programowaniu komputerów – gry losowe, symulacje różnych procesów losowych, statystycznych, testowanie algorytmów dla losowych zestawów danych itp. Ponieważ nie można w prosty sposób mieć prawdziwych liczb losowych, trzeba się zadowolić ich sztucznym odpowiednikiem – liczbami pseudolosowymi (ang. pseudorandom numbers). Liczby pseudolosowe wyglądają jak losowe, lecz tworzy się je algorytmicznie. Oznacza to, iż znając wzór generacyjny oraz kilka kolejnych liczb pseudolosowych możemy bez problemu wygenerować wszystkie dalsze – tej cechy nie posiadają liczby losowe[9].

### 3.1.1. Generatory LCG

Do rozwiązania problemu generacji liczb pseudolosowych opracowano specjalne funkcje modularne zwane liniowymi generatorami kongruencyjnymi liczb pseudolosowych (ang. pseudorandom number linear congruential generator – w skrócie LCG) o postaci przedstawionej wzorem 3.2.

$$X_n = (a * X_{n-1} + c) \bmod m \quad (3.2)$$

,gdzie:

- $X_n$  - n-ta liczba pseudolosowa
- $X_{n-1}$  - poprzednia liczba pseudolosowa
- $a$  - mnożnik
- $c$  - przyrost
- $m$  - moduł

Ze wzoru wynika, iż kolejna liczba pseudolosowa  $X_n$  powstaje z poprzedniej  $X_{n-1}$ . Liczby te tworzą zatem ściśle określony ciąg kolejno następujących po sobie wartości. Drugą cechą charakterystyczną jest to, iż liczba pseudolosowa  $X_n$  jest resztą z dzielenia przez moduł  $m$ . Skoro tak, to może przyjmować wartości od 0 do  $m - 1$ . Z pierwszej i drugiej własności wynika, iż po  $m$  cyklach obliczeniowych, liczby pseudolosowe zaczynają się powtarzać (wzór 3.3):

$$X_0 \rightarrow X_1 \rightarrow X_2 \rightarrow \dots \rightarrow X_{m-2} \rightarrow X_{m-1} \rightarrow X_0 \quad (3.3)$$

Jeśli współczynniki  $a$ ,  $c$  i  $m$  są źle dobrane, to cykl powtarzania może być krótszy niż  $m$ . Rozróżniane są dwa podstawowe rodzaje generatorów LCG (Tabela 3.1):

Tablica 3.1: Rodzaje generatorów LCG

Metoda	Wzór
<b>Addytywny LCG</b>	$X_n = (aX_{n-1} + c) \bmod m$
<b>Multiplikatywny LCG</b>	$X_n = aX_{n-1} \bmod m$

Podstawowa różnica pomiędzy nimi jest taka, iż generator addytywny LCG może generować liczby pseudolosowe z zakresu od 0 do  $m - 1$ , a generator multiplikatywny generuje je z zakresu od 1 do  $m - 1$  [9].

## 3.2. Algorytmy

System Rozstrzygania Konkursów Internetowych musi posiadać algorytmy bazujące na pseudolosowości wyłaniające zwycięzcę losowania  $Z$ , gdzie  $Z$  jest liczbą  $1 \dots m$ , czyli istnieje możliwość wygenerowania kilku wygranych. W niektórych przypadkach istotna jest kolejność, by ustalić komu przyporządkować jaką nagrodę. Ważne mogą być również atrybuty użytkowników np. wiek, by wyłaniać zwycięzców tylko z danego zakresu wiekowego.

### 3.2.1. Algorytm prostego losowania

Najprostszym i najbardziej intuicyjnym algorytmem jest wybranie losowej liczby z zakresu  $1 \dots n$ , gdzie  $n$  to liczba wszystkich ludzi biorących udział w wydarzeniu. Algorytm sprawuje się dobrze, dla prostego przypadku wyboru wygranych, jednak nie nadaje się dla przypadków z wykorzystaniem atrybutów użytkowników oraz wtedy, gdy jest potrzeba przypisać danym osobom odpowiednie wagi (na przykład konkursy, gdzie faworyzowane są kobiety).

#### 3.2.1.1. Działanie

**Dane:**

- *people* - tablica  $1 \dots n$  ludzi biorących udział w wydarzeniu
- *m* - liczba zwycięzców do wygenerowania

**Rezultat:**

- *winners* - lista ludzi wylosowanych w ustalonej kolejności

Kod Java został przedstawiony na listingu 3.1.

Listing 3.1: Algorytm prostego losowania

```
1 Random random = new Random();
2
3 public List<Person> getWinners(List<Person> people, int m) {
4
5     if (m==1) {
6         Integer randomIndex = random.nextInt(people.size());
7         return getPeopleAtIndexes(Arrays.asList(randomIndex));
8     } else {
9         List<Integer> randomUniqueNumbers = generateRandomUniqueNumbers(m);
10        return getPeopleAtIndexes(randomUniqueNumbers);
11    }
12 }
```

Ciekawą metodą może być wygenerowanie losowych liczb bez zwracania. W przypadku błędnego algorytmu przedstawionego na listingu 3.2 rozwiązanie może być bardzo wolne, a rezultatu można się

teoretycznie nigdy nie doczekać. W skrajnych przypadkach może dojść do zawieszenia systemu: wystarczy dla bardzo dużej liczby osób losować bardzo dużą liczbę zwycięzców  $m$ .

Listing 3.2: Zły przykład generowania tablicy losowych liczb

```
1 Random random = new Random();
2 public List<Integer> generateRandomUniqueNumbers(int m) {
3     List<Integer> numbers = new ArrayList<Integer>();
4     int i=0;
5     while(i<m) {
6         int result = random.nextInt(m);
7         if(!numbers.contains(result)) {
8             numbers.add(result);
9             i++;
10        }
11    }
12    return numbers;
13 }
```

Rozwiązaniem tego problemu jest stworzenie tablicy i wypełnienie jej liczbami  $1 \dots m$ , a następnie pomieszczenie jej i zwrócenie początkowych  $m$  liczb(listing 3.3):

Listing 3.3: Prawidłowy przykład generowania tablicy losowych liczb

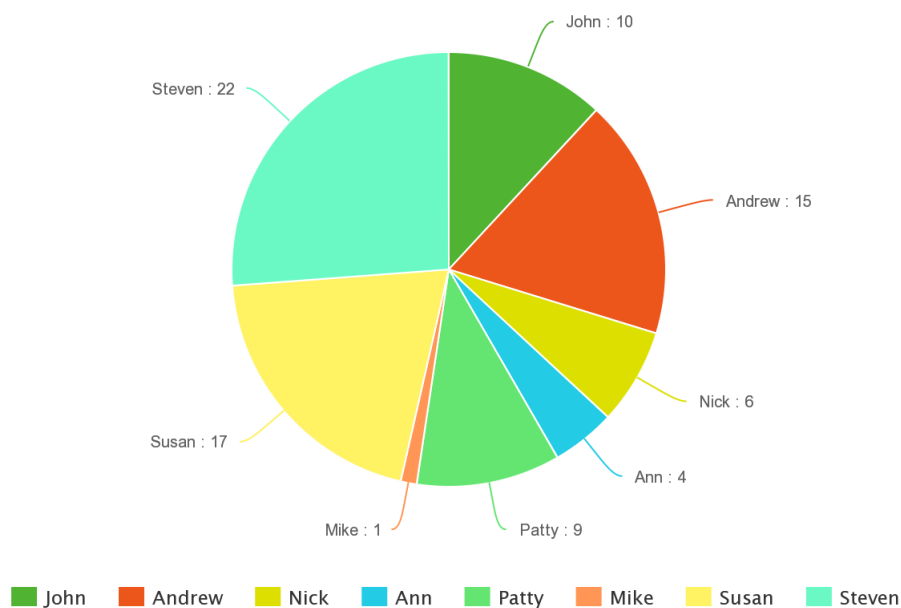
```
1 Random random = new Random();
2
3 public List<Integer> generateRandomUniqueNumbers(int m) {
4     ArrayList<Integer> numbers = new ArrayList<Integer>();
5     int i=0;
6     for(i=0; i<m; ++i) {
7         numbers.add(i);
8     }
9     //pomieszaj tablicę
10    for(i=0; i<m; ++i) {
11        int randomIndex = random.nextInt(m);
12        //zamień elementy
13        int tmp = numbers.get(randomIndex);
14        numbers.set(randomIndex, numbers.get(i));
15        numbers.set(i, tmp);
16    }
17    return numbers.subList(0, m);
18 }
```

Czasem twórca konkursu chce dopuścić do losowania tylko osoby, które spełniają określone kryteria np. tylko osoby powyżej 18 roku życia. W tym wypadku należy przefiltrować osoby spełniające postawione założenia przed uruchomieniem losowania.

### 3.2.2. Algorytm losowania z uwzględnieniem wag - Ruletka

W konkursach, w których istnieje potrzeba faworyzowania osób o danym atrybucie użyta została metoda podobna do tej używanej w algorytmach genetycznych - Ruletce[10]. Algorytm polega na wypełnieniu koła rezultatem funkcji przystosowania osób, tak by te mające wyższą wartość wypełniały większy obszar.

Następnie z takiego koła jest losowany wycinek i osoba, do której on należy jest osobą wygraną.



Rysunek 3.1: Metoda ruletki

#### 3.2.2.1. Funkcja przystosowania

Funkcją przystosowania[11] nazywana jest funkcja wyznaczona empirycznie na podstawie zestawu atrybutów. Służy ona ocenie osobnika w kontekście potencjalnego zwycięzcy. Dla przykładu, kiedy twórca konkursu postanowi faworyzować nastolatki w danym losowaniu, funkcja przystosowania będzie zależna od dwóch czynników: wieku oraz płci. Funkcja przystosowania może wyglądać jak ta przedstawiona we wzorze 3.4.

$$f(w, p) = \frac{16}{w} + \frac{1}{2}p \quad (3.4)$$

,gdzie:

- $w$  - wiek danej osoby
- $p$  - płeć danej osoby (mężczyzna - 0, kobieta - 1).

Tabela 3.2 ukazuje wyniki funkcji przystosowania 3.4 dla danego wieku oraz płci.

Tablica 3.2: Wartości dla funkcji przystosowania

Wiek	Funkcja przystosowania(Kobieta)	Funkcja przystosowania(Mężczyzna)
12	1,833333333	1,333333333
13	1,730769231	1,230769231
14	1,642857143	1,142857143
15	1,566666667	1,066666667
16	1,5	1
17	1,441176471	0,941176471
18	1,388888889	0,888888889
19	1,342105263	0,842105263
20	1,3	0,8
21	1,261904762	0,761904762
22	1,227272727	0,727272727
23	1,195652174	0,695652174

### 3.2.2.2. Działanie

#### Dane:

- *people* - tablica  $1 \dots n$  ludzi biorących udział w wydarzeniu

#### Rezultat:

- *winner* - zwycięzca

Kod Java został przedstawiony na listingu 3.4.

Listing 3.4: Metoda Ruletki

```

1      Random random = new Random();
2
3      public Person getWinner(List<Person> people){
4          List<PersonMark> circleChart = createCircleChart(people);
5          float sumHeadingTo = random.nextFloat() * getMarksSum(circleChart);
6          final Float MAX_STEP= sumHeadingTo/20f;
7          final Float MIN_STEP= 1f;
8
9          Float currentAlgorithmSum=0;
10         Person currentPerson=null;
11         while(currentAlgorithmSum<sumHeadingTo){
12             PersonMark personMark=getPersonMarkWithinSum(currentAlgorithmSum);
13             currentPerson =personMark.getPerson();

```

```

14         float step = random.nextFloat() * (MAX_STEP-MIN_STEP) + MIN_STEP;
15         currentAlgorithmSum += step;
16     }
17     return currentPerson;
18 }

```

Algorytm można wzbogacić o losowanie  $m$  wygranych osób. Wówczas takie zachowanie będzie polegało na  $m$ -krotnym powtórzeniu powyższego algorytmu z wykluczaniem zwycięzcy.

### 3.2.3. Algorytm losowania z uwzględnieniem wag - Turniej

Dla algorytmów z uwzględnieniem wag istnieje opcja wyłaniania zwycięzców metodą turniejową. Algorytm polega na podziale osób w równe grupy, z których wyłaniany jest jeden kandydat. Ten z wyższą wartością funkcji przystosowania zostanie wybrany z większym prawdopodobieństwem[12]. W każdym obiegu grupy turniejowe się zacieśniają łącząc liderów aż do wyłonienia  $m$  zwycięzców.

#### 3.2.3.1. Działanie

**Dane:**

- *people* - tablica  $1 \dots n$  ludzi biorących udział w wydarzeniu
- $m$  - liczba zwycięzców do wygenerowania

**Rezultat:**

- *winners* - lista ludzi wylosowanych w ustalonej kolejności

Kod Java został przedstawiony na listingu 3.5.

Listing 3.5: Metoda Turniejowa

```

1     Random random = new Random();
2
3     public List<Integer> getWinners(List<Person> people, int m) {
4         final int GROUP_SIZE=10;
5         List<Group> groups = null;
6         List<Person> currentWinners = people;
7         while(true) {
8
9             if(canGetWinners(currentWinners, GROUP_SIZE, m) {
10                 return currentWinners.subList(0, m);
11             }
12             groups = dividePeopleIntoGroups(currentWinners, GROUP_SIZE);
13
14             currentWinners= new ArrayList<Person>();
15             for(Group group: groups){
16                 Person groupWinner = group.getWinner();
17                 currentWinners.add(groupWinner);

```

```


18         }
19     }
20 }
```

Populacja zwycięzców zmniejsza się 10-krotnie w każdym obiegu, co w efekcie daje  $m$  zwycięzców.

### 3.3. Testy wydajnościowe

#### 3.3.1. Algorytmy progresywne

Algorytmy progresywne polegają na okresowej inkrementacji określonego wskaźnika postępu - korzysta z nich metoda wyścigu, metoda liczb oraz metoda słupków. Na listingu 3.6 widać przeprowadzoną przykładową symulację, a jej wynik czasowy został ukazany na Rys.3.2.

 algorithmNumbersShouldGetWinner [Runner: JUnit 4] (2,649 s)

Rysunek 3.2: Symulacja algorytmu progresywnego

Listing 3.6: Algorytmy progresywne

```

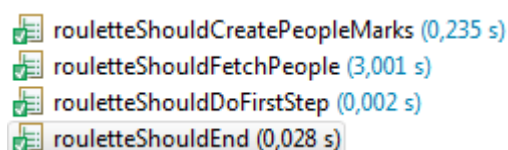
1 Fetching people data..Done
2 Starting algorithm.. people number: 50
3 Performing next step...
4 14,14,14,14,14,14,14,14,13,13,13,13,13,13,12,12,12,11,11,11,11,11,10,10,10,10,
5 10,10,9,9,9,9,9,9,9,9,8,8,8,8,8,8,8,7,7,7,6,6,6
6 Performing next step...
7 28,26,25,25,24,24,24,24,24,23,23,23,22,22,22,22,22,22,21,21,21,21,21,21,
8 21,20,20,20,20,19,19,19,19,18,18,18,18,17,17,17,16,16,16,15,15,15,14,13
9 Performing next step...
10 37,36,35,35,35,35,34,34,34,34,34,33,33,32,32,32,32,32,32,31,31,31,30,30,30,
11 30,30,30,30,29,29,29,29,29,28,28,28,28,28,27,27,27,27,26,24,21,20,20
12 Performing next step...
13 51,49,47,46,45,45,45,44,43,43,43,42,42,42,42,42,42,41,41,41,41,41,40,40,40,40,
14 40,39,38,37,37,37,37,37,37,36,36,35,35,35,35,34,34,34,33,33,33,29,29,28
15 Performing next step...
16 61,58,57,55,55,54,54,53,53,53,53,53,52,51,51,51,50,50,50,50,49,49,49,49,48,48,
17 47,47,47,47,46,46,46,46,45,45,45,45,45,44,44,43,43,42,42,42,41,40,35,34
18 Performing next step...
19 74,71,67,67,66,65,64,64,63,63,62,62,62,61,61,61,61,61,60,60,59,59,58,58,58,57,
20 57,57,57,57,57,56,56,55,55,53,53,53,53,52,52,52,51,50,50,49,48,47,42,40,
21 Performing next step...
22 85,80,78,78,77,77,75,74,73,73,72,72,72,72,71,71,71,71,71,70,70,70,69,68,67,67,
23 67,67,66,66,65,65,65,65,64,63,63,62,62,61,61,60,60,57,55,55,55,54,54,47
24 Performing next step...
25 98,90,88,87,86,85,84,84,84,84,84,84,83,83,80,80,80,80,80,79,79,79,78,77,77,76,
26 75,75,75,75,74,74,73,73,72,72,72,71,71,70,69,69,69,69,68,68,66,62,61,59
```



```
27 Performing next step...
28 100,90,88,87,86,85,84,84,84,84,84,84,83,83,80,80,80,80,80,79,79,79,78,77,77,76
29 ,75,75,75,75,74,74,73,73,72,72,72,71,71,70,69,69,69,69,68,68,66,62,61,59
30 Algorithm has ended
31 Winners: [User68]
32 Ended successfully after 9 steps and the winner is User68
```

### 3.3.2. Algorytm Ruletki

Algorytm ruletki tworzy wybrane koło na podstawie obliczonych ocen, a następnie wybiera wycinek koła, do którego będzie zmierzał. Na listingu 3.7 widać przeprowadzoną przykładową symulację, a jej wynik czasowy został ukazany na Rys.3.3.



Rysunek 3.3: Symulacja algorytmu ruletki


#### Listing 3.7: Algorytm Ruletki

```
1 Heading to sum 3868.8936
2 Stepping by: 36.15982
3 Stepping by: 95.29172
4 Stepping by: 145.34573
5 Stepping by: 47.60253
6 Stepping by: 161.38754
7 Stepping by: 75.011444
8 Stepping by: 56.1572
9 Stepping by: 73.51316
10 Stepping by: 130.69806
11 Stepping by: 115.75128
12 Stepping by: 46.849808
13 Stepping by: 77.82817
14 Stepping by: 107.22652
15 Stepping by: 170.12004
16 Stepping by: 29.62485
17 Stepping by: 37.811543
18 Stepping by: 8.264225
19 Stepping by: 172.97832
20 Stepping by: 76.81204
21 Stepping by: 60.28087
22 Stepping by: 148.21983
23 Stepping by: 6.5851135
24 Stepping by: 73.43045
```

```
25 Stepping by: 161.81088
26 Stepping by: 8.321165
27 Stepping by: 109.40624
28 Stepping by: 32.377197
29 Stepping by: 94.54863
30 Stepping by: 182.09035
31 Stepping by: 180.81508
32 Stepping by: 147.1804
33 Stepping by: 110.292274
34 Stepping by: 9.883128
35 Stepping by: 35.64439
36 Stepping by: 106.62233
37 Stepping by: 21.50307
38 Stepping by: 144.04553
39 Stepping by: 106.02528
40 Stepping by: 133.26276
41 Stepping by: 148.4961
42 Stepping by: 42.962112
43 Stepping by: 24.329105
44 Stepping by: 134.08755
45 Ended successfully after 43 steps and the winner is User129
```

### 3.3.3. Algorytm Turniejowy

Algorytm turniejowy ma za zadanie dzielić ludzi na grupy, do czasu gdy nie będzie to już dalej możliwe. Na listingu 3.8 widać przeprowadzoną przykładową symulację, a jej wynik czasowy został ukazany na Rys.3.4.

 tournamentShouldGetWinners (3,590 s)

Rysunek 3.4: Symulacja algorytmu turniejowego

#### Listing 3.8: Algorytm Turniejowy

```
1 Fetching people data..Done
2 Group size is 5
3 Starting people number: 150
4 Dividing into groups ..
5 Getting winners..
6 Winners:
7 User112,User39,User44,User23,User79,User57,User47,User106,User23,User41,User83,User50,
8 User93,User4,User45,User65,User120,User38,User108,User81,User141,User83,User51,User19,
9 User123,User37,User139,User35,User78,User89
10 Dividing into groups ..
11 Getting winners..
12 Winners:
```

```
13 User41,User120,User139,User93,User89,User23,  
14 Dividing into groups ..  
15 Getting winners..  
16 Winners:  
17 User89  
18 Ended successfully after 3 divisions and the winner is User89
```



## **4. Projekt - Sposoby prezentacji losowania**

Użytkownik podczas tworzenia wydarzenia może wybrać preferowany sposób prezentacji przebiegu losowania. Gdy nastąpi godzina uruchomienia wydarzenia system przeprowadza konkurs komunikując się na bieżąco z warstwą prezentacji stopniowo wizualizując postęp użytkownikom.

### **4.1. Wizualizacje**

Warstwa prezentacji zawiera następujące metody wizualizacji przebiegu konkursu:

- Wyścig
- Słupki
- Liczby
- Ruletka
- Turniej

#### **4.1.1. Wyścig**

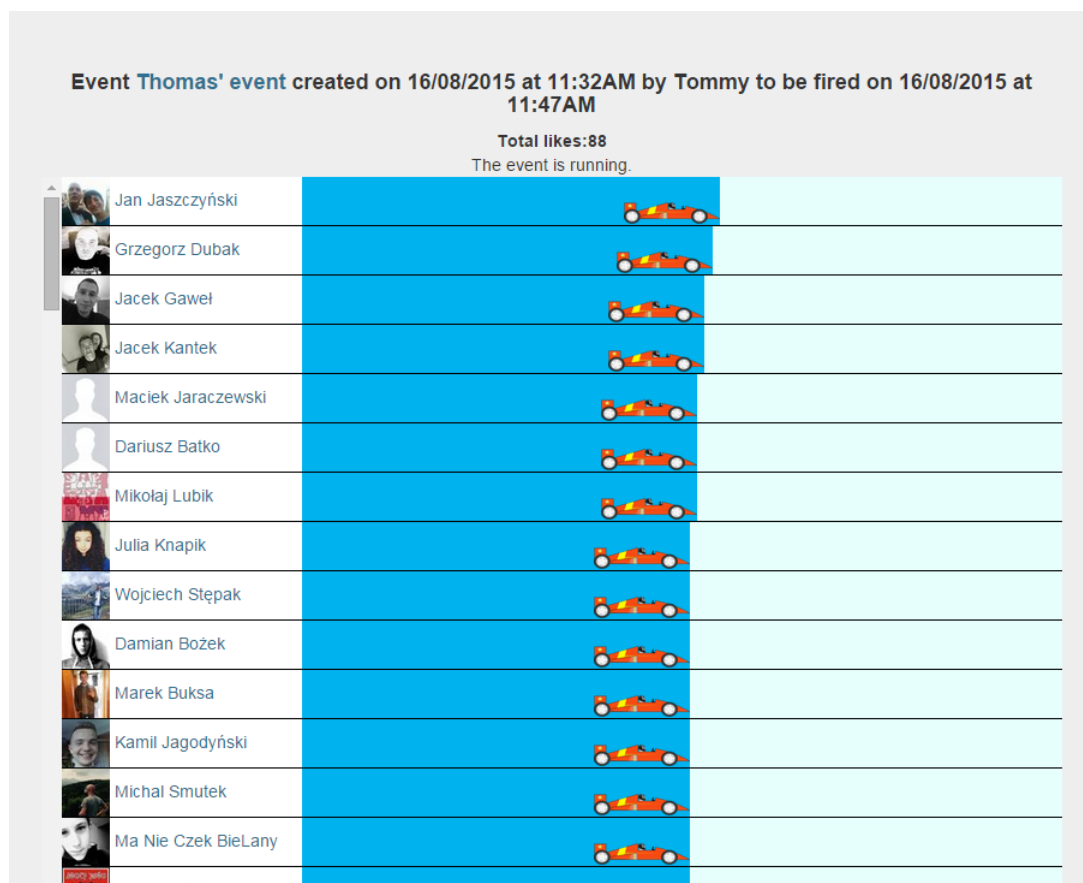
Wyścig przedstawiony na Rys. 4.1 polega na przedstawieniu osób biorących w konkursie w postaci uczestników wyścigu samochodowego. W przypadku dużej liczby osób wyświetlani są tylko Ci, którzy są na czele stawki. GUI okresowo dostaje informacje o aktualnym procencie ukończonej trasy każdego uczestnika.

#### **4.1.2. Słupki**

Słupki przedstawione na Rys. 4.2 są wizualizacją podobną do wyścigu, lecz jest to przedstawienie uczestników na zasadzie diagramu słupkowego.

#### **4.1.3. Liczby**

Liczby przedstawione na Rys. 4.3 polegają na wybraniu z góry ustalonej losowej liczby. Następnie każdy użytkownik okresowo otrzymuje wartość akumulowaną do swojego wyniku. Wygrywa jeśli jego



Rysunek 4.1: Wyścig

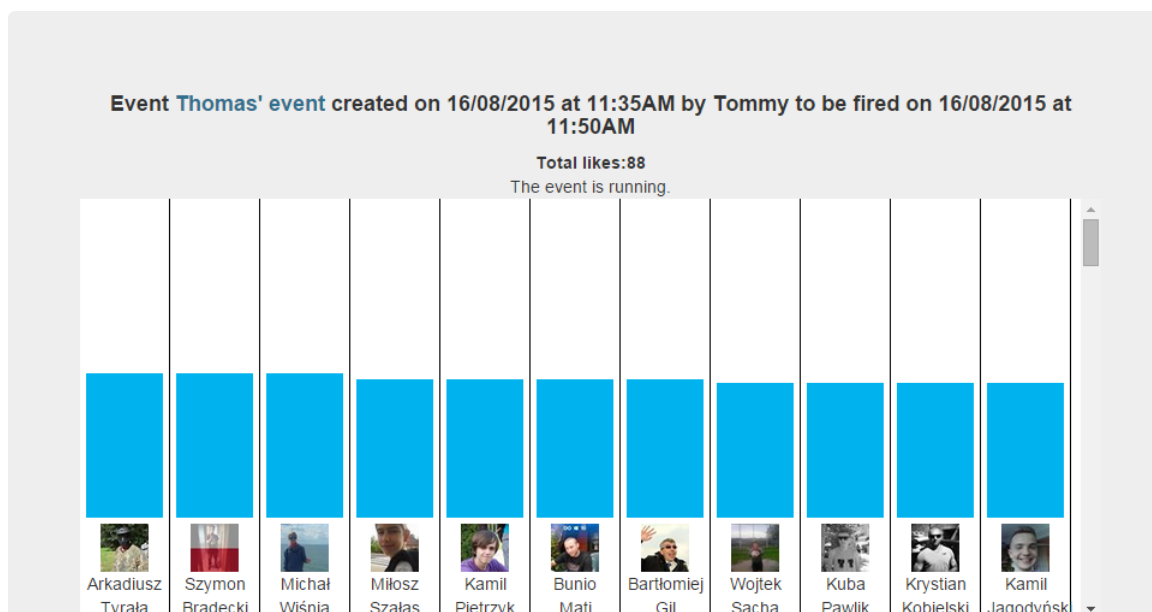
suma równa się ustalonej liczbie, jeżeli zaś maksimum zostanie przekroczone suma jest akumulowana od nowa.

#### 4.1.4. Ruletka

Ruletka przedstawiony na Rys. 4.4 jest wizualizacją polegającą na prezentacji uczestników w postaci diagramu kołowego i kursora zmierzającego wokół niego. W pewnym momencie kursor się zatrzymuje i wyłaniany jest wygrany.

#### 4.1.5. Turniej

Turniej przedstawiony na Rys. 4.5 polega na podziale osób w odpowiednie grupy, a następnie prezentacji zwycięzców każdej fazy i łączeniu ich w kolejne grupy.



Rysunek 4.2: Słupki

## 4.2. Komunikacja z GUI

System w postaci serwisu www komunikuje się z GUI przekazując co pewien okres czasu odpowiednie stany postępu losowania oglądającym. Do serwisu przesyłane jest żądanie zawierające znacznik czasowy ostatniej zmiany algorytmu. Następnie serwis przy zmianie stanu algorytmu przekazuje odpowiedź z nowym znacznikiem czasowym.

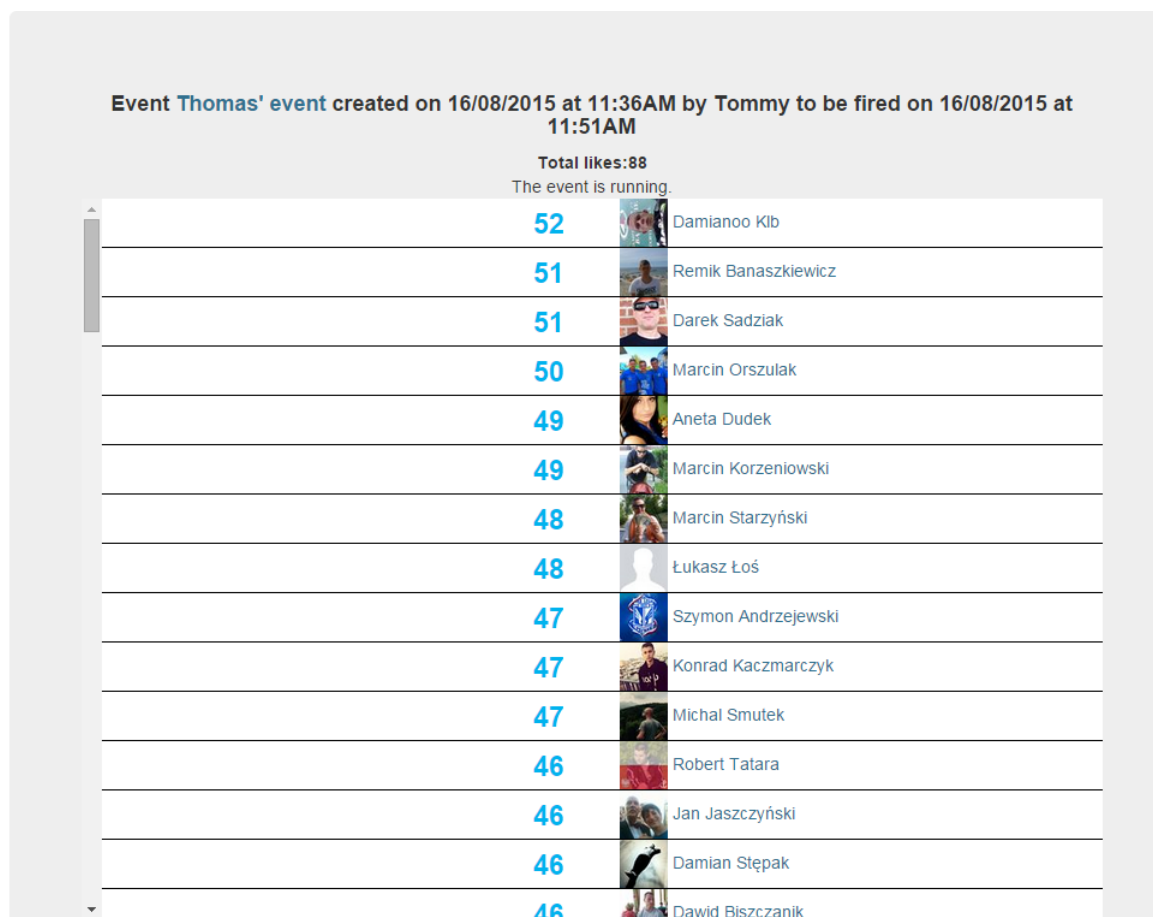
### 4.2.1. Long polling

Long polling pozwala zoptymalizować wysyłanie informacji do klienta. Klient żąda informacji z serwera w sposób podobny do normalnego żądania. Jednakże, jeżeli serwer nie posiada żadnych informacji dostępnych dla klienta, zamiast wysłać pustą odpowiedź, czeka aż informacje staną się dostępne. Kiedy informacja jest możliwa do udostępnienia kompletna odpowiedź jest wysyłana. Klient natychmiast ponownie wysyła żądanie do serwera o nowe informacje. W ten sposób serwer może odpowiedzieć praktycznie od razu po ich otrzymaniu[13]. Ideę long polling zaprezentowano na rysunku Rys. 4.6.

System wykorzystuje technikę long polling'u w celu optymalizacji zapytań po stronie serwera jak i klienta unikając redundantnych żądań i odpowiedzi.

### 4.2.2. Dalszy rozwój w WebGL

Istnieje możliwość rozwinięcia wizualizacji do postaci 3d. W tym celu może zostać użyta technologia WebGL, która potrafi wykorzystać moc karty graficznej i stworzyć treści trójwymiarowe bez konieczności instalowania dodatkowego oprogramowania.



Rysunek 4.3: Liczby

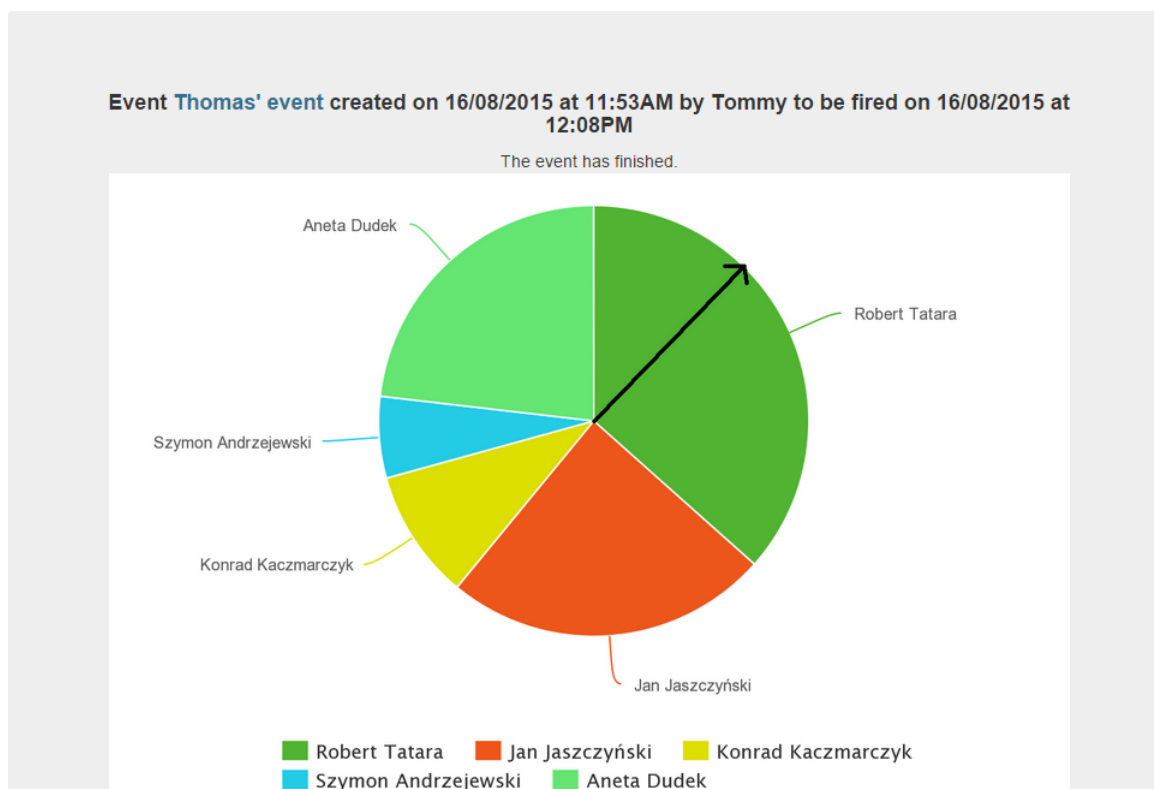
#### 4.2.2.1. Możliwości WebGL

WebGL bazuje na OpenGL ES w wersji 2.0 i dostarcza programistom interfejs grafiki 3D. Korzysta z występującego w HTML 5 elementu Canvas i daje dostęp do modelu DOM. Aktualnie zaimplementowane jest w przeglądarkach Mozilla Firefox, Google Chrome i Apple Safari oraz Opera i Internet Explorer od wersji 11. Do rysowania grafiki za pomocą GPU wymagają urządzeń z kartami graficznymi wspierającymi minimalnie OpenGL 2.0 lub OpenGL ES 2.0. W przypadku braku zgodnej karty graficznej w przeglądarkach Google Chrome i Internet Explorer 11 możliwe jest rysowanie grafiki programowalnej[14].

#### 4.2.2.2. Problemy z WebGL

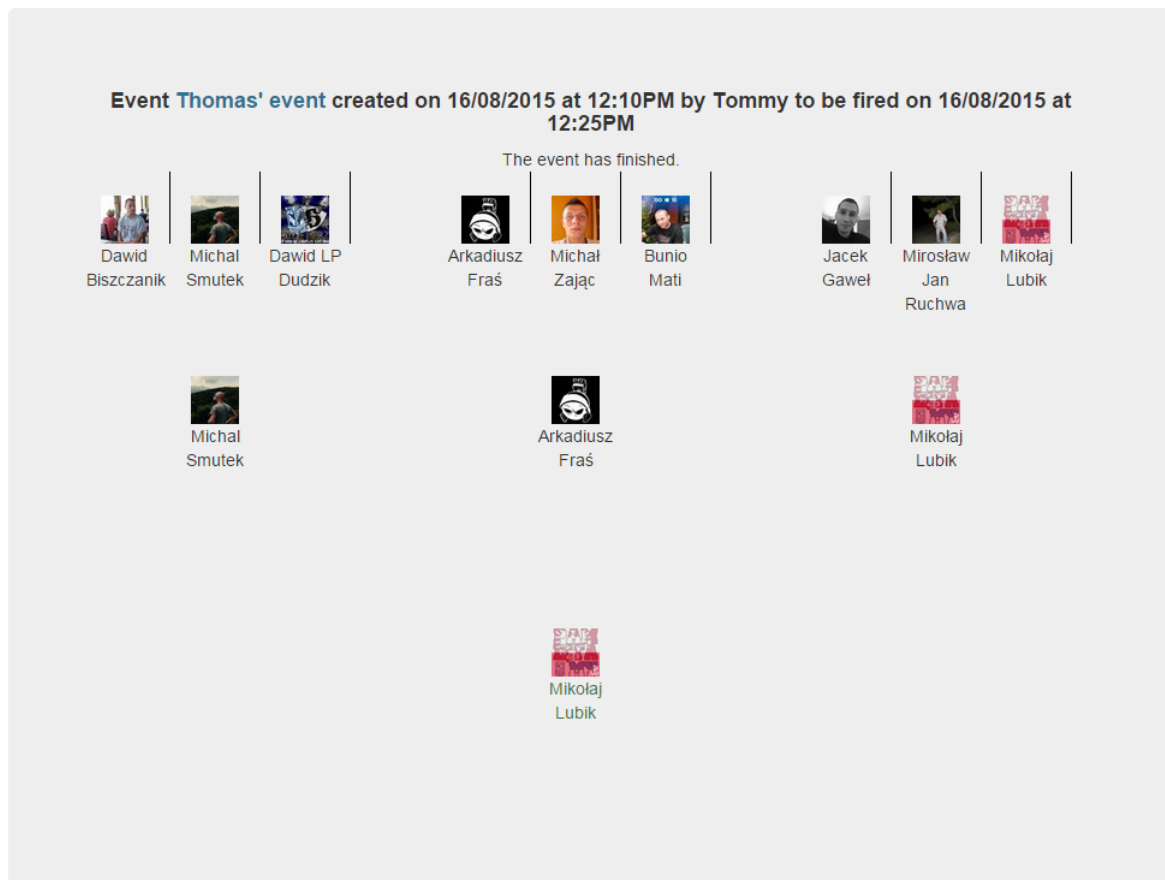
W maju 2011 niemiecka grupa Context Information Security opublikowała artykuł opisujący wiele podatności związanych z bezpieczeństwem WebGL oraz jego implementacji w przeglądarkach Mozilla Firefox oraz Google Chrome. Przykładowe ataki obejmują zawieszenie komputera oraz wykonanie i przesłanie przez sieć Internet zrzutu ekranu użytkownika. W kolejnych tygodniach, zagrożenia wynikające z użycia WebGL potwierdził US-CERT oraz Microsoft[14]. Dodatkowym problemem jest domyśl-



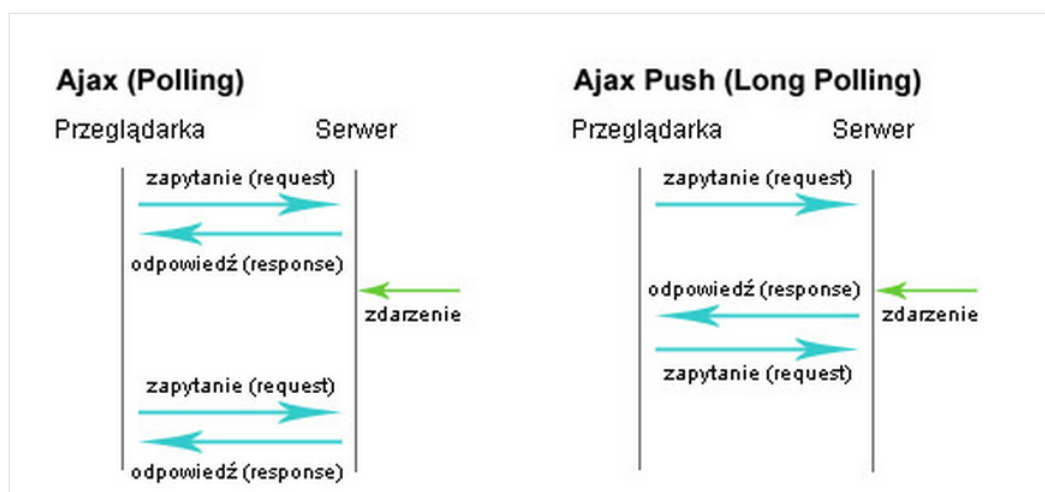


Rysunek 4.4: Ruletka

nie wyłączona obsługa WebGL w takich przeglądarkach jak np. Mozilla Firefox. Należałoby poinstruować użytkownika w jaki sposób odblokować tę możliwość, co może okazać się dla niego problematyczne.



Rysunek 4.5: Turniej



Rysunek 4.6: Long polling

## 5. Projekt - możliwości GUI

W aplikacji Systemu Rozstrzygania Konkursów Internetowych warstwę pośrednią między użytkownikiem a serwisem stanowi strona internetowa. Ta warstwa prezentacji zawiera wszelkie możliwości interakcji klienta z serwisem. GUI przystosowane jest do wykorzystywania przez dwa rodzaje użytkowników

- Użytkownika - każdy klient serwisu z utworzonym kontem Facebook lub Twitter
- Administratora - osoba z uprawnieniami przewyższającymi zwykłego klienta serwisu

### 5.1. Użytkownik

#### 5.1.1. Przypadki użycia

Użytkownik posiada ograniczoną rolę do podstawowych czynności dostarczanych przez system. Przypadki użycia użytkownika zaprezentowano na Rys. 5.1.

Do systemu załogować się można poprzez Facebook, Twitter za pośrednictwem mechanizmu pojedynczego uwierzytelnienia(Simple Sign-On) oraz jako administrator. .

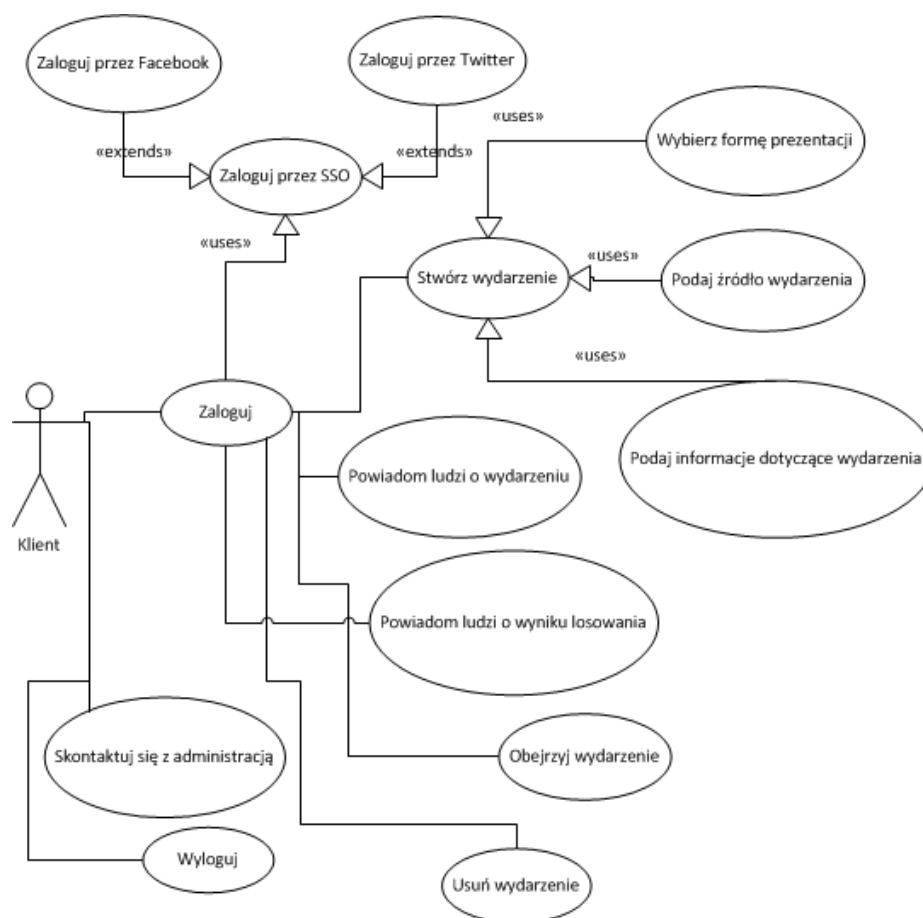
#### 5.1.2. Single Sign-On

Uwierzytelnienie będące podstawowym mechanizmem bezpieczeństwa informacji należy rozumieć jako proces potwierdzenia tożsamości użytkownika. Jeśli ma miejsce poprawne uwierzytelnienie, wówczas zachodzi pewność, iż użytkownik jest tym za kogo się podaje.

Potrzeba uwierzytelnienia wzrosła w dobie Internetu z uwagi na rozpowszechnienie pracy zdalnej przy jednoczesnym ograniczeniu możliwości identyfikowania użytkowników fizycznie, czy osobiście. Wymagane zatem jest aby omawiany mechanizm był starannie zaimplementowany w każdym systemie, a przede wszystkim w aplikacji WWW.

Pojedyncze logowanie (Single Sign-On) to proces uwierzytelnienia podmiotu, który pozwala na jednokrotne wprowadzenie danych uwierzytelniających (np. login i hasło ), a następnie na dostęp usług bez ponownego wprowadzania tych danych.

Im więcej przeróżnych aplikacji i systemów dostępnych dla użytkowników, tym więcej może być odmiennych sposobów uzyskania uprawnionego dostępu do ich zasobów. Z reguły duże firmy i korporacje



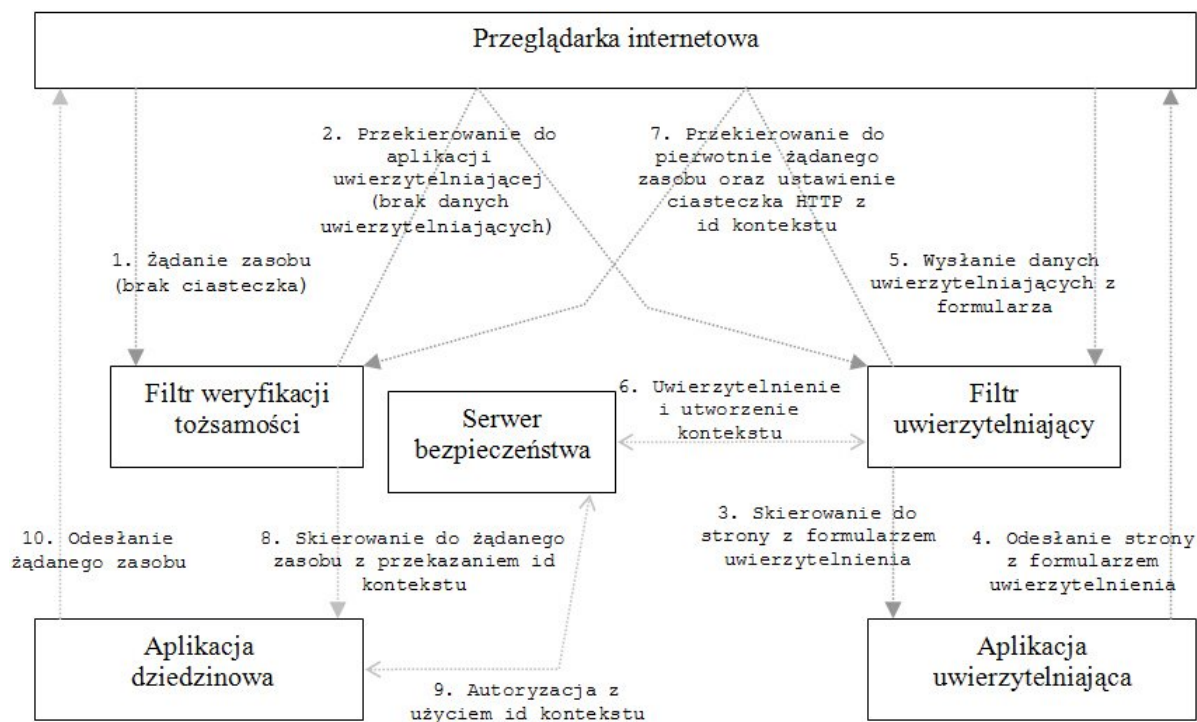
Rysunek 5.1: Przypadki użycia - użytkownik

nie wykorzystują jednorodnych rozwiązań, a są one tworzone i dostarczane przez wielu dostawców na bazie różnych technologii, a także mogą spełniać zupełnie rozbieżne oczekiwania poszczególnych działów organizacji. To powoduje, że nie jest łatwe (lub jest niemożliwe) opracowanie jednakowego sposobu logowania do wszystkich dostępnych systemów. Oczywiście, im więcej różnych systemów z odmiennymi metodami logowania, dopuszczalną składnią haseł, czy też częstotliwością wymuszenia zmiany hasła na nowe (co 30 dni, co 90 dni, a nawet wcale), tym więcej danych dostępowych do zapamiętania i tym większe ryzyko nieumyślnego ich ujawnienia, utraty, czy zapomnienia[4].

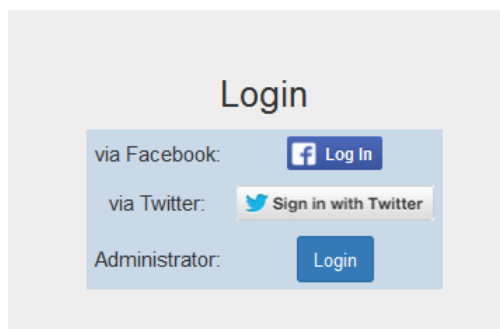
Działanie systemu pojedynczego uwierzytelnienia przedstawia Rys. 5.2.

### 5.1.3. Logowanie

Po uruchomieniu aplikacji klienckiej klientowi przedstawiona jest strona logowania prezentowana na Rys. 5.3.



Rysunek 5.2: Działanie SSO(źródło: <http://www.mariuszlipinski.pl/2008/04/mechanizm-pojedynczego-uwierzytelnienia.html>)



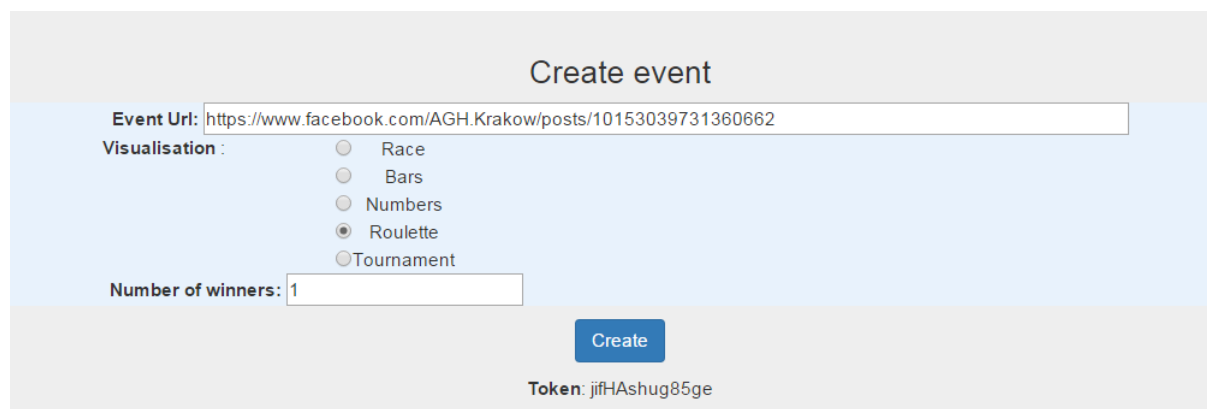
Rysunek 5.3: Panel logowania

#### 5.1.4. Tworzenie wydarzenia

Po poprawnym logowaniu użytkownik ma do dyspozycji dwie opcje:

- Stworzenie wydarzenia
- Obejrzenie wydarzenia

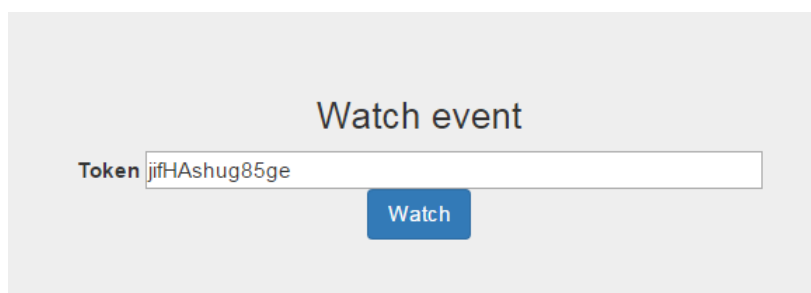
Stronę umożliwiającą stworzenie wydarzenia zaprezentowano na Rys. 5.4.



Rysunek 5.4: Tworzenie wydarzenia

### 5.1.5. Obejrzenie wydarzenia

Stronę umożliwiającą obejrzenie wydarzenia zaprezentowano na Rys. 5.5.

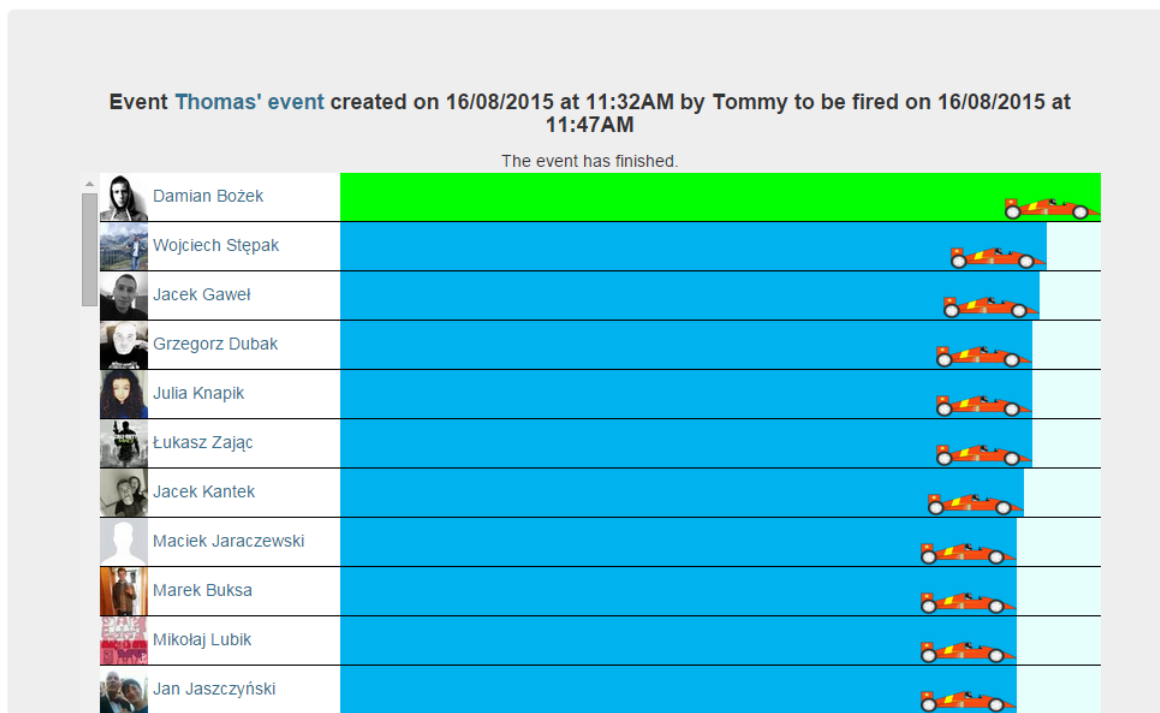


Rysunek 5.5: Obejrzenie wydarzenia

Konkurs może mieć trzy stany:

- Nierozpoczęty
- Trwający
- Zakończony

Strona dotycząca prezentacji konkursu odpowiednio informuje o jego stanie z adekwatnymi możliwościami np. poinformowaniem innych o zakończonym losowaniu. Jest to realizowane przy użyciu API dostarczanych przez zewnętrzne serwisy Facebook oraz Twitter. Przykładową stronę z zakończonym losowaniem i wyłonionym zwycięzcą zaprezentowano na Rys. 5.6.



Rysunek 5.6: Wyłoniony zwycięzca

## 5.2. Administrator

### 5.2.1. Przypadki użycia

Administrator posiada rozszerzone możliwości służące zarządzaniu wydarzeniami oraz użytkownikami. Przypadki użycia administratora przedstawiono na Rys. 5.7.

### 5.2.2. Panel administracyjny

Administrator po zalogowaniu ma do dyspozycji panel administracyjny widoczny na Rys. 5.8, z którego może wykonywać akcje służące zarządzaniu serwisem.

- Zarządzanie użytkownikami
- Zarządzanie konkursami

Panel zarządzania użytkownikami przedstawiono na Rys. 5.9.

Panel zarządzania wydarzeniami przedstawiono na Rys. 5.10.

### 5.2.3. Panel wiadomości

Administrator ponadto zajmuje się obsługą zgłaszanych problemów przez użytkowników. Takowe żądania realizowane są poprzez panel wiadomości widoczny na Rys. 5.11.



Rysunek 5.7: Przypadki użycia - administrator

Web contest determination system

Admin panel

Admin panel

Manage users

Manage events

©2015 Tomasz Szepczyński












Like
 Share
 Be the first of your friends to like this.

Rysunek 5.8: Panel administracyjny



**Admin panel**  
**Manage users**

Browse people from an event:

	Jędrrek Możejko
	Remik Banaszkiewicz
	Asia Chlebda
	Jerzy Drela
	Rajmund Kapustka
	Wojciech Stępak
	Marcin Antosz
	Ola Tynor
	Bartek Bąbel
	Maciek Jaraczewski
	Iza Batko

Ban person:

Send notification:

Rysunek 5.9: Panel zarządzania użytkownikami

## Web contest determination system

Admin panel

**Admin panel**  
**Manage events**

Browse an event:

People involved: 11

Change Event Url:

Visualisation :

- ☐ Race
- ☐ Bars
- ☐ Numbers
- ☐ Roulette
- ☒ Tournament

Number of winners:

©2015 Tomasz Szepczyński

  Be the first of your friends to like this.


Rysunek 5.10: Panel zarządzania wydarzeniami

## Web contest determination system

Admin panel

Admin panel

Send notification

Asia Chlebda

Hello Asia.  
You have won the event created by Thomas. Please contact him for further info.

Send

©2015 Tomasz Szepczyński

 Like

 Share

 Be the first of your friends to like this.

Rysunek 5.11: Panel wiadomości



## 6. Projekt - architektura aplikacji

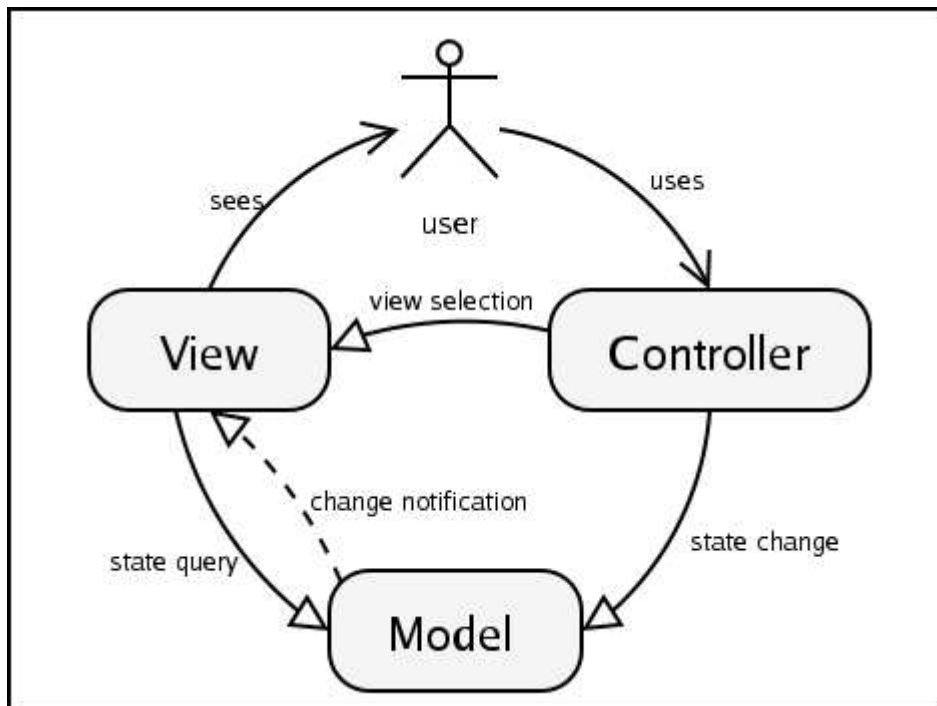
Wybór odpowiedniej architektury dla tworzonego systemu stanowi poważne wyzwanie szczególnie w przypadku dużych aplikacji internetowych wykorzystujących najnowsze możliwości technologiczne. Ma bowiem istotny wpływ m.in. na łatwość pielęgnacji, czyli utrzymywalność (ang. maintainability) systemu czy jego skalowalność (ang. scalability)[15].

### 6.1. MVC

Źródła MVC (Model-View-Controller) sięgają końca lat siedemdziesiątych i języka Smalltalk-80. Paradygmat MVC stał się centralną koncepcją realizacji interfejsów użytkownika w tym języku, jak również w innych językach i systemach zorientowanych obiektowo. Jest również znakomitą ilustracją zasady podziału odpowiedzialności (ang. separation of concerns). Zasada ta znajduje odzwierciedlenie w podziale klas aplikacji na trzy grupy: model (ang. model), widok (ang. view) i kontroler (ang. controller), co obrazuje Rys. 6.1.

Semantyka triady MVC i reguły komunikacji pomiędzy jej poszczególnymi elementami są następujące. Model związany jest z domeną aplikacji, zawiera jej elementy statyczne i behawioralne. Najważniejszą składową modelu jest logika aplikacji oraz stojąca za nią logika biznesowa. Innymi słowy, model określa działanie aplikacji oraz przetwarzane dane. Powinien być tak zaprojektowany, żeby był niezależny od wybranego rodzaju prezentacji oraz systemu obsługi akcji użytkownika (model nic nie wie o widoku i kontrolerze). Jedyne powiązanie wychodzące z modelu to powiadomienie widoku o aktualnych zmianach (change notification). W większości implementacji jest to rozwiązane za pomocą systemu komunikatów lub z zastosowaniem wzorca Observer.

Widok zarządza graficzną lub tekstową prezentacją modelu (jest jego wizualnym odwzorowaniem). Realizacja widoku jest już powiązana z konkretnym modelem. Prezentacja nie może zostać wygenerowana bez znajomości specyfiki danych czy operacji, które obrazuje użytkownikowi. Rys. 6.1 ilustruje to powiązanie: widok pobiera informacje z modelu (state query) ilekroć zostaje powiadomiony o jego zmianie. Z drugiej strony model nie jest związany ze sposobem jego prezentacji. W związku z tym zmiany widoku nie pociągają za sobą zmian w modelu. Kontroler jest natomiast odpowiedzialny za reagowanie na akcje użytkownika (np. kliknięcia myszką), odwzorowując je na operacje zawarte w modelu (state change) oraz na zmiany widoku (view selection). W połączeniu z widokiem odpo-



Rysunek 6.1: Model MVC

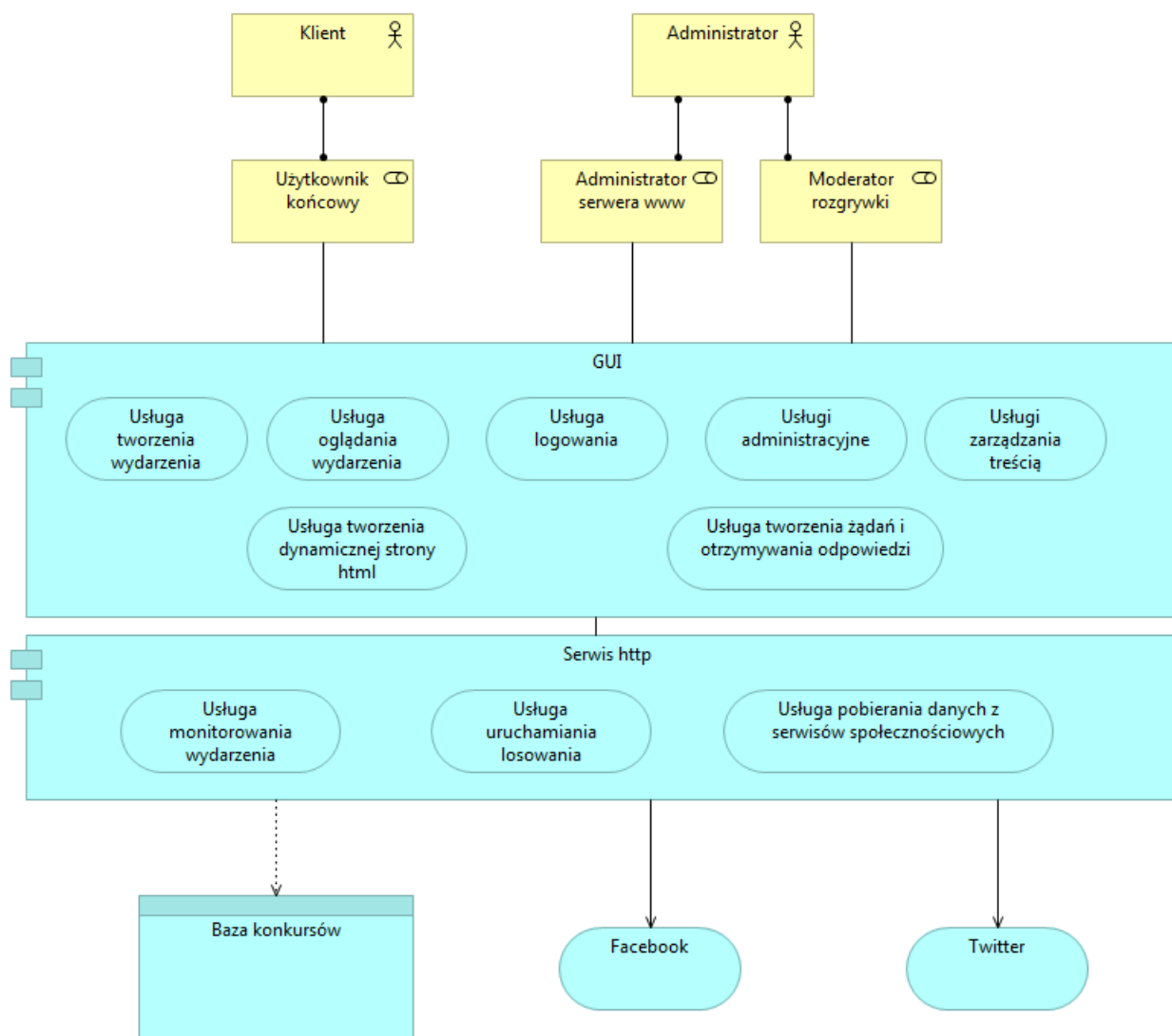
wiada za dwa aspekty interfejsu użytkownika (look and feel). odwołania do niej. Wartość szkieletu architektonicznego MVC leży w dwóch podstawowych zasadach. Pierwsza to separacja prezentacji i modelu, umożliwiającą zmianę interfejsu użytkownika (np. udostępnienie usług aplikacji poprzez interfejs graficzny i tekstowy). Druga zasada to separacja widoku i kontrolera. Klasycznym przykładem jest użycie dwóch kontrolerów związanych z jednym widokiem, które umożliwiają edycję widoku lub jej nie umożliwiają [15].

## 6.2. Diagram warstw

Rozdzielenie aplikacji na warstwy pozwala na luźne łączenie(ang. loose-coupling) komponentów oraz na przejrzystość w działaniu systemu. Diagram przedstawiony na Rys. 6.2 ukazuje zastosowaną warstwową architekturę Systemu Rozstrzygania Konkursów Internetowych.

## 6.3. Diagram komunikacji

Aplikacja komunikuje się głównie za pośrednictwem protokołu http. Wyjątkiem jest połączenie do bazy danych przez konektor jdbc. Zarówno żądania jak i odpowiedzi http są przekazywane w formacie JSON. Diagram przedstawiony na Rys. 6.3 ukazuje system komunikacji zastosowany w aplikacji.

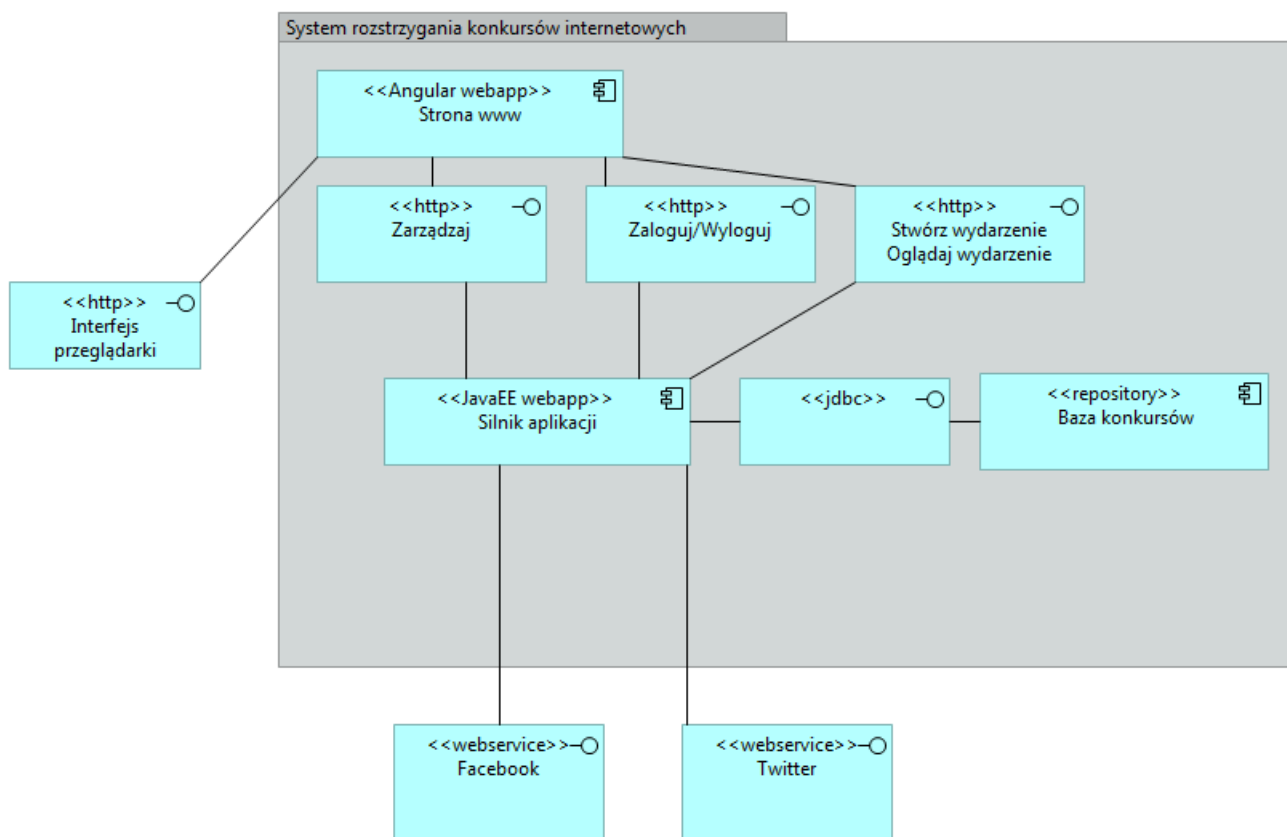


Rysunek 6.2: Warstwy aplikacji

## 6.4. Aplikacja w chmurze

Bardzo interesującym rozwinięciem aplikacji byłoby umieszczenie jej w chmurze. Pozwoli to m.in. na spory wzrost wydajności aplikacji, co może być kluczowe przy konkursach z dużą liczbą uczestników.

Usługi w modelu cloud computing nie wprowadzają nowych technologii, a raczej zmieniają sposób wykorzystania już istniejących. Najbardziej oczywistą korzyścią, jaką wnoszą, jest możliwość skorzystania z elementów infrastruktury klasy enterprise po kosztach bezpośrednio odzwierciedlających wartość ich możliwości operacyjnych. Dla małego biznesu, który potrzebuje niewielu serwerów i stosunkowo niewielkiej pamięci masowej dla swoich aplikacji, usługa w chmurze oferuje zazwyczaj coś znacznie tańszego niż jakakolwiek opcja "on-premise" nawet w dłuższym okresie. Niewielkie obciążenie



Rysunek 6.3: Diagram komunikacji

nia nie zapewniają pełnego wykorzystania sprzętu i oprogramowania, w które trzeba zainwestować przy wdrożeniach własnych.

Sukces chmur publicznych zachęca wiele organizacji do skorzystania z tej techniki dostarczania usług i coraz więcej firm decyduje się na wdrażanie projektów cloud services. Uruchomienie usług w chmurze (prywatnej lub publicznej) może zapewnić zarówno nowy strumień zysków, jak i wygodną samoobsługę dla klientów firmy, przy minimalnych nakładach na rozszerzenie istniejącej infrastruktury. Zdefiniowanie problemów, które mają rozwiązywać potencjalne cloud services to pierwszy krok w określaniu korzyści, jakie przedsięwzięcie to może wnieść do biznesu. Trzeba pamiętać, że w niektórych przypadkach konieczne może być przeniesienie całych aplikacji, w innych potrzebne będzie tylko kilka prostych usług (eksponowanych np. przez protokół web services).

Po ustaleniu celu i rodzaju zastosowania cloud services, trzeba określić, jak zostaną wykorzystane i ile to będzie kosztować. Koszty migracji zależą od stanu istniejących aplikacji, ich architektury oraz zakresu prac niezbędnych do wykonania, aby udostępnić ich funkcjonalności w chmurze.

Ocena kosztów w dużej mierze zależy od aplikacji i usług, które zamierza się udostępnić w



chmurze. Do podstawowych składowych takiej oceny można zaliczyć: ocenę stanu istniejących aplikacji, ocenę istniejących projektów interfejsu użytkownika, ocenę istniejącego projektu bezpieczeństwa, określenie, jak ma być zaimplementowane rozliczanie użytkownika, oszacowanie zakresu zmian w podstawowych aplikacjach oraz dodatkowych zadań związanych z ich przeprojektowaniem[16].

## 6.5. Bezpieczeństwo aplikacji

Każda aplikacja internetowa narażona jest na szereg ataków ze strony szkodliwych użytkowników. Wraz z rozwojem internetu powstają kolejne techniki służące atakowaniu serwisów www. System Rozstrzygania Konkursów Internetowych posiada implementację technik sprzyjających ochronie aplikacji.

### 6.5.1. Popularne ataki

- SQL Injection
- XSS
- DDoS

### 6.5.2. Obrona przed atakami

#### 6.5.2.1. SQL Injection

Atak SQL Injection polega na wstrzyknięciu zapytania SQL przez pole formularza i wykonanie go po stronie serwera na bazie danych. Taki atak może pobrać poufne dane, podszyć się pod inną osobę lub zmodyfikować bazę[17]. Przykładowe wejście może wyglądać w sposób przedstawiony na listingu 6.1.

Listing 6.1: Atak SQL Injection

```
select * from user where username='admin' OR '1'='1' and password='_'
```

Takie zapytanie zezwala atakującemu na zalogowanie się do strony bez dostarczania hasła, ponieważ wyrażenie OR jest zawsze prawdziwe[18].

Najpopularniejsze sposoby obrony przed SQL Injection to:

- Parametryzowane zapytania
- Procedury składowane
- Filtracja danych wejściowych

W aplikacji zastosowano podejście pierwsze - parametryzowane zapytania(listing 6.2).

Listing 6.2: Parametryzowane zapytania

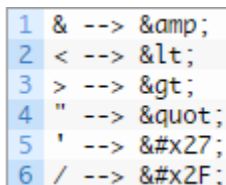
```
1 String selectStatement = "SELECT_*_FROM_User_WHERE_userId=_?_";
2 PreparedStatement prepStmt = con.prepareStatement(selectStatement);
3 prepStmt.setString(1, userId);
4 ResultSet rs = prepStmt.executeQuery();
```

### 6.5.2.2. XSS

Cross-site-Scripting jest to przede wszystkim atak na klienta korzystającego z podatnej aplikacji internetowej (w przeciwieństwie np. do SQL injection, którego głównym celem jest część serwerowa). Po drugie, atak polega na wstrzyknięciu do przeglądarki ofiary fragmentu javascript bądź innego języka skryptowego (np. VBScript), który może być uruchomiony w przeglądarce. W efekcie, atakujący ma możliwość wykonania dowolnego kodu skryptowego w przeglądarce, co posiada poniższe konsekwencje:

- Wykradanie cookies - przejęcie sesji ofiary
- Dynamiczna podmiana zawartości strony www
- Uruchomienie keylogger'a w przeglądarce

Ochrona przed XSS polega przede wszystkim na tym, by w odpowiedni sposób filtrować dane przesyłane przez użytkownika – przed ich wyświetleniem w aplikacji. Najczęściej przybiera to formę zamiany pewnych istotnych znaków kontrolnych HTML (głównie znaki otwierające / zamykające tagi oraz atrybuty tagu) na encje HTML widoczne na Rys.6.4.



1	&	-->	&amp;
2	<	-->	&lt;
3	>	-->	&gt;
4	"	-->	&quot;
5	'	-->	&#x27;
6	/	-->	&#x2F;

Rysunek 6.4: Zamiana encji

Taka filtracja najprawdopodobniej zlikwiduje większość podatności XSS w aplikacji.

Inną i zazwyczaj niewiele kosztującą metodą ochrony przed jednym z efektów XSS – tj. wykradaniem cookies – jest zastosowanie parametru HttpOnly dla ciasteczek sesyjnych[19].

### 6.5.2.3. DDoS

Ataki DDoS wykorzystują dostępną przepustowość serwera, co prowadzi do niedostępności usługi lub infrastruktury.

W trakcie ataku DDoS, na serwer jest przesyłanych wiele zapytań jednocześnie, a ich ilość może doprowadzić do przerw w dostępie, a nawet do całkowitego braku połączenia z serwerem. Istnieją trzy

strategie(Tabela 6.1), które są wykorzystywane do zablokowania działania strony internetowej, serwera lub infrastruktury[20].

Tablica 6.1: Strategie ochrony przed DDoS

Metoda	Opis
Przepustowość	Kategoria ataku polegającego na wykorzystaniu zasobów sieciowych serwera. Serwer jest niedostępny.
Zasoby	Kategoria ataku polegającego na wykorzystaniu zasobów systemu maszyny i zablokowaniu przekazywania odpowiedzi na poprawne zapytania.
Wykorzystanie błędu w oprogramowaniu	Atak o nazwie "exploit", który wykorzystuje błąd w oprogramowaniu do zablokowania maszyny lub do przejęcia kontroli nad maszyną.

Ochrona przed DDoS jest bardzo trudna i często polega na :

- Analizowania wszystkich pakietów w czasie rzeczywistym i z dużą prędkością
- Pobierania ruchu przesyłanego na serwer
- Filtrowania czyli identyfikowania niepoprawnych pakietów IP - pozostałe pakiety są akceptowane

Najlepszym rozwiązaniem dla aplikacji internetowej jest obarczenie odpowiedzialnością istniejące rozwiązania, np. umieszczenie aplikacji w chmurze Amazon z usługą Amazon CloudFront oray Amazon Route 53[21].

## 6.6. Wykorzystane technologie

W Systemie Rozstrzygania Konkursów Internetowych wykorzystanych zostało wiele technologii wykorzystanych w warstwie prezentacji oraz w warstwie serwisu http.

### Dla Warstwy GUI:

- Angular
- Yeoman
- Grunt
- Bower
- HTML/CSS/LESS/jQuery

- Nodejs
- Npm
- Protractor
- Karma

**Dla serwisu http:**

- Spring
- Hibernate
- QueryDSL
- Jackson

**Dla Bazy danych:**

- MySQL

### **6.6.1. Angular**

AngularJS to framework JavaScript stworzony przez inżynierów z Google. Służy on do szybkiego i łatwego budowania aplikacji internetowych, tak zwanych – single app. Model oparty o MVW (Model – View – Whatever) pozwala pogodzić idee JavaScript i modelu MVC.

### **6.6.2. Yeoman**

Yeoman jest ekosystemem generatorów służących tworzeniu szkieletu aplikacji.

### **6.6.3. Grunt**

Grunt jest to system automatyzacji pracy, umożliwia wykonanie określonych zestawów zadań na danym folderze, czy plikach.

### **6.6.4. Bower**

Bower jest menadżerem paczek klienckich i umożliwia pobieranie zależności.

### **6.6.5. HTML/CSS/LESS/jQuery**

HTML, CSS oraz jQuery tworzą stronę przeznaczoną do przeglądarki. LESS służy łatwiejszej generacji arkuszy stylów.

### 6.6.6. Nodejs

Node.js służy wykonywaniu kodu javascript po stronie serwera. Node.js to platforma, która zapewnia najwyższą wydajność dzięki wykorzystywaniu nieblokujących operacji I/O oraz asynchronicznego mechanizmu zdarzeń. Działa na bazie najwydajniejszego silnika obsługującego język JavaScript – V8 (autorstwa firmy Google) – i pozwala programistom osiągać niezwykle efekty[22].

### 6.6.7. Npm

Node Packaged Module jest system paczek dla środowiska Node. Dostarcza narzędzia serwerowe, dotyczące framework'ów.

### 6.6.8. Protractor

Protractor to narzędzie, które udostępnia pewne mechanizmy służące testowaniu aplikacji Angular.js, jednocześnie uruchamiając w tle serwer Selenium.

### 6.6.9. Karma

Karma jest narzędziem, które pozwala uruchamiać testy kodu JavaScript pod różnymi przeglądarkami, a w zasadzie ich emulowanym środowiskiem. Karma nie jest pełnoprawnym frameworkiem do testowania, a jedynie rusztowaniem, na którego bazie zbudować można pełny stos testujący JavaScript. Karma zbudowana jest w oparciu o serwer Node.js oraz technologię Socket.io. Pozwala na wykonywanie testów w różnych środowiskach pracy (deweloperskim, testowym, produkcyjnym itd.), a ponadto wykonuje wszystkie testy w separacji od kodu właściwego. Pozwala także na wykonywanie testów pod różnymi przeglądarkami jednocześnie[23].

### 6.6.10. Spring

Spring to szkielet wytwarzania aplikacji (framework), dzięki któremu proces budowania oprogramowania w języku Java dla platformy J2EE staje się znacznie prostszy i efektywniejszy. Spring oferuje usługi, które można z powodzeniem używać w wielu środowiskach – od apletów po autonomiczne aplikacje klienckie, od aplikacji internetowych pracujących w prostych kontenerach serwletów po złożone systemy korporacyjne pracujące pod kontrolą rozbudowanych serwerów aplikacji J2EE. Spring pozwala na korzystanie z możliwości programowania aspektowego, znacznie sprawniejszą obsługę relacyjnych baz danych, błyskawiczne budowanie graficznych interfejsów użytkownika oraz integrację z innymi szkieletami takimi, jak Struts czy JSF[24].

### 6.6.11. Hibernate

Hibernate - framework do realizacji warstwy dostępu do danych. Hibernate zwiększa wydajność operacji na bazie danych dzięki buforowaniu i minimalizacji liczby przesyłanych zapytań. Jest to projekt

rozwijany jako open source[25].

#### **6.6.12. QueryDSL**

QueryDSL to narzędzie do obsługi zapytań SQL, które dostarcza generatory kod i współpracę z wieloma bazami, również NoSQL. Zapytania w QueryDSL są bardziej przyjazne dla użytkownika i bezpieczne typologicznie.

#### **6.6.13. Jackson**

Jackson API zawiera wiele funkcjonalności służące serializacji oraz deserializacji do formatu JSON.

#### **6.6.14. MySQL**

MySQL to wolnodostępny system zarządzania relacyjnymi bazami danych rozwijany przez firmę Oracle.

## 7. Podsumowanie

Celem pracy było stworzenie Systemu Rozstrzygania Konkursów Internetowych, by losowania przeprowadzane za pośrednictwem sieci były przeprowadzane zgodnie z zasadami fair-play. Cel udało się zrealizować – powstał silnik systemu w postaci serwisu www wraz z aplikacją kliencką tożsamą z warstwą prezentacji.

Przedstawione zostały zewnętrzne usługi serwisów społecznościowych zezwalające na pobieranie danych użytkowników. Ujawniono problemy oraz ograniczenia wynikające z korzystania z powyższych API. Bardziej wnikliwa inwestycja wykazała, że w przypadku bardzo dużej porcji danych mogą istnieć problemy z limitami zapytań.

Omówiono dokładnie problem losowości w komputerach oraz wyjaśniono jak ją symulować na przykładzie kilku dostępnych algorytmów. Problemem wydajnościowym nie okazał się silnik wykonujący logikę, a dostęp do zewnętrznych źródeł danych oraz jednoczesna komunikacja z dużą liczbą użytkowników.

System został stworzony z użyciem najnowszych technologii przedstawiając współczesne możliwości aplikacji internetowych. Wykorzystano najpopularniejsze wzorce projektowe, by stworzyć zestaw komponentów łączących się w spójną całość. Użycie istniejących rozwiązań w dużym stopniu ułatwia programiście tworzenie aplikacji, pomaga w utrzymaniu systemu oraz może zaoszczędzić czas przeznaczony na jej rozwój.

Przedstawiono jak aplikacja może być dalej rozwijana w przypadku olbrzymich porcji danych, gdy przepustowość pojedynczego serwera może nie wystarczać. Ukazano również w jaki sposób współczesna aplikacja internetowa jest narażona na ataki działające na niekorzyść systemu i jak powinna tworzyć ochronę przed podstawowymi zagrożeniami.





## Bibliografia

- [1] “Strutta,” <http://www.strutta.com/>.
- [2] “Wp contest creator,” <http://wpcontestcreator.com/>.
- [3] “Heyo,” <http://heyo.com/>.
- [4] “What is single sign on?,” <http://www.authenticationworld.com/Single-Sign-On-Authentication/>.
- [5] “Facebook developers api,” <https://developers.facebook.com/docs/graph-api>.
- [6] “Twitter documentation,” <https://dev.twitter.com/overview/documentation>.
- [7] I. R. Jim Webber, Savas Parastatidis, *REST in Practice*. O’Reilly Media, 2010.
- [8] K. Makice, *Twitter API: Up and Running*. O’Reilly Media, 2009.
- [9] “Liczby pseudolosowe,” [http://edu.i-lo.tarnow.pl/inf/utills/010\\_2010/0213.php](http://edu.i-lo.tarnow.pl/inf/utills/010_2010/0213.php).
- [10] Z. Michalewicz, *Algorytmy genetyczne + struktury danych = programy ewolucyjne*. Wydawnictwa Naukowo Techniczne, 2003.
- [11] “Klasyczny algorytm genetyczny,” <http://kolos.math.uni.lodz.pl/~archive/Sztuczna%20inteligencja/6%20Klasyczny%20algorytm%20genetyczny%20cz1.pdf>.
- [12] M. Melanie, *An Introduction to Genetic Algorithms*. Cambridge, Massachusetts - London, England: A Bradford Book The MIT Press, 1999.
- [13] “Long polling,” <http://designconcept.webdev20.pl/articles/long-polling/>.
- [14] “Webgl,” <https://pl.wikipedia.org/wiki/WebGL>.
- [15] “Architektura nowoczesnych aplikacji internetowych,” <http://madeyski.e-informatyka.pl/download/19.pdf>.

- [16] “Aplikacja w chmurze - wybór i przygotowywanie,” <http://www.computerworld.pl/news/388840/Aplikacja.w.chmurze.wybor.i.przygotowywanie.html>.
- [17] “Sql injection,” [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection).
- [18] “Preventing sql injection in java,” [https://www.owasp.org/index.php/Preventing\\_SQL\\_Injection\\_in\\_Java](https://www.owasp.org/index.php/Preventing_SQL_Injection_in_Java).
- [19] “Czym jest xss?,” <http://sekurak.pl/czym-jest-xss/>.
- [20] “Czym jest ochrona anty-ddos ?,” <https://www.ovh.pl/anti-ddos/zasada-anty-ddos.xml>.
- [21] “Aws best practices for ddos resiliency,” [https://d0.awsstatic.com/whitepapers/DDoS\\_White\\_Paper\\_June2015.pdf](https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf).
- [22] T. H. N. R. Mike Cantelon, Marc Harter, *Node.js w akcji*. Helion, 2014.
- [23] “Angularjs 6 – karma – pierwszy test,” <http://mrzepinski.pl/angularjs-6-karma-pierwszy-test.html>.
- [24] A. A. T. R. C. S. Rod Johnson, Juergen Hoeller, *Spring Framework. Profesjonalne tworzenie oprogramowania w Javie*. Helion, 2006.
- [25] “Mapowanie obiektowo-relacyjne z wykorzystaniem hibernate,” <http://sens.e-informatyka.pl/wp-content/uploads/WIRP2/Hibernate.pdf>.