abooble

Gra przeglądarkowa

Dokumentacja projektu

Autor: Antoni Cepak

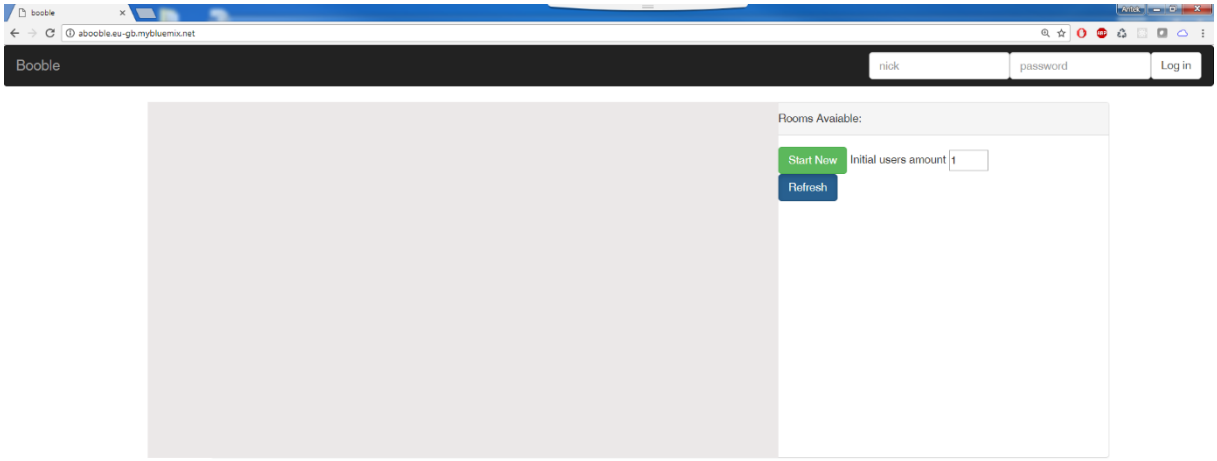
Kraków, 20.09.2017r.

1. Wstęp

Abooble to aplikacja umożliwiająca rozgrywanie pojedynków w czasie rzeczywistym między rozproszonymi użytkownikami, posiadającymi łącze dostępu do Internetu. Osiągnięto do dzięki zastosowaniu architektury typu klient – server. Za moduł serwera użyto serwera aplikacyjnego WAS Liberty a cała aplikacja jest wdrożona w chmurze Bluemix. Część serwerowa została wykonana w technologii JavaEE 7, cliencka oparta jest na technologii html5 oraz języku JavaScript. Komunikacja między modułami została zaimplementowana przy użyciu technologii WebSocket.

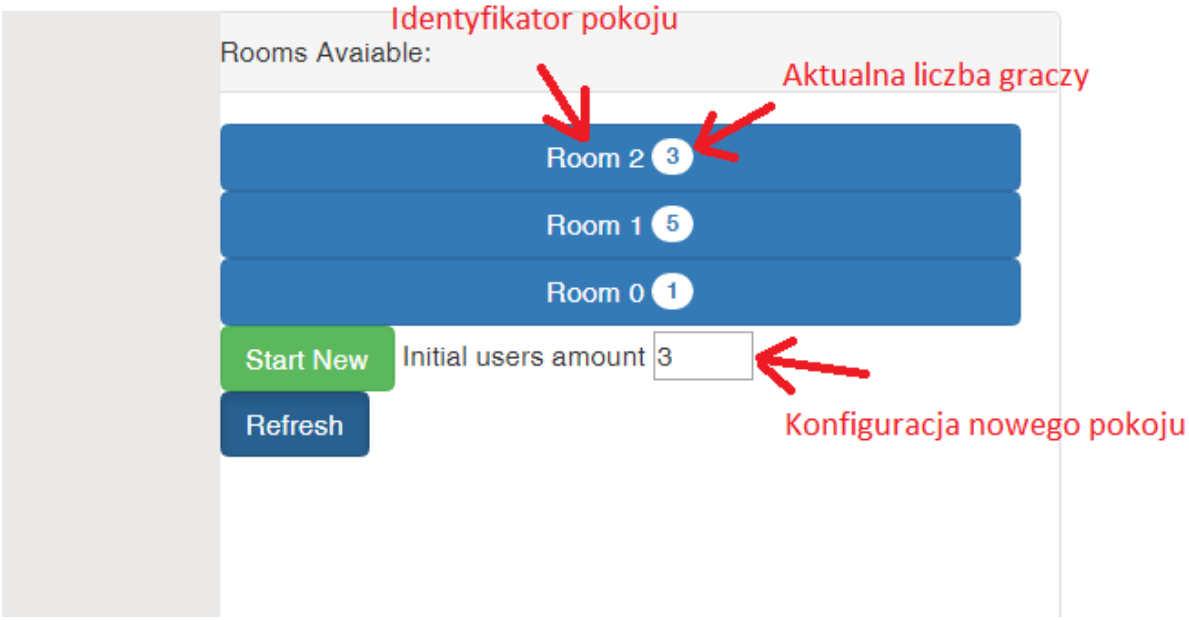
2. Opis aplikacji

Aby rozpocząć korzystanie z aplikacji należy w przeglądarce wpisać adres: http://abooble.eu-gb.mybluemix.net/. Przeglądarka pobierze zasoby modułu klienckiego pozwalające na komunikację użytkownika z serwerem.



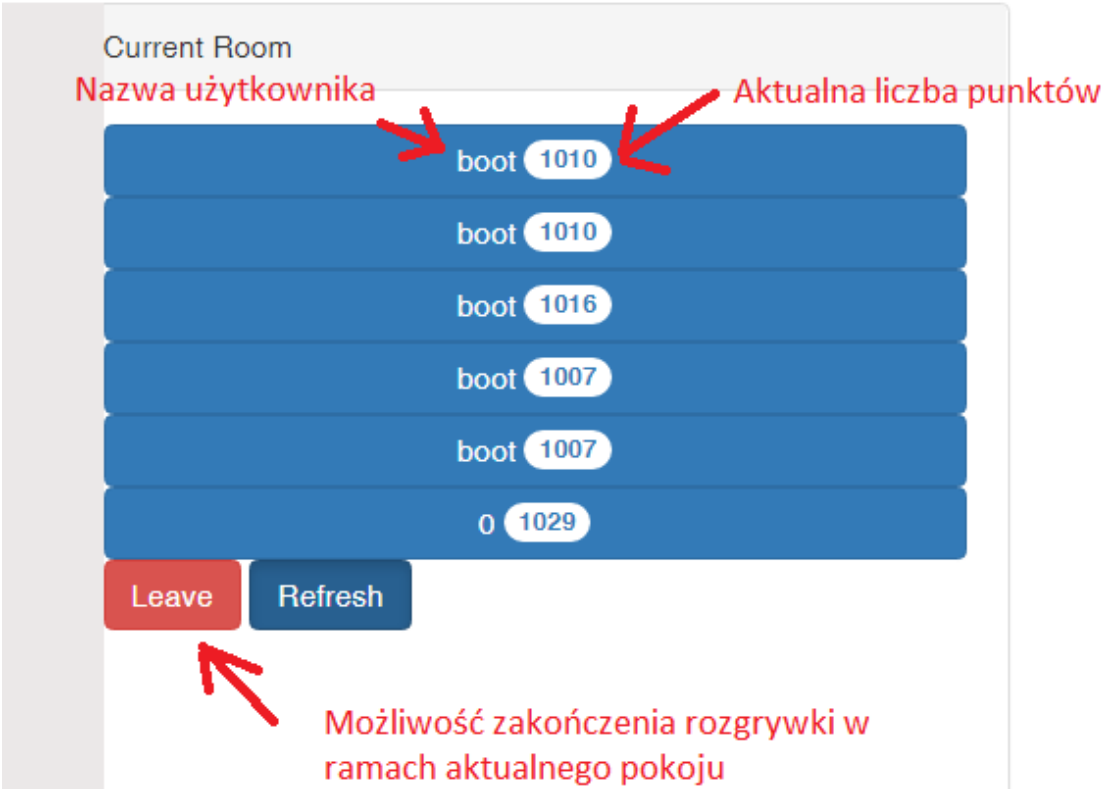
Rysunek 1 Ekran startowy aplikacji

Abooble jest grą typu multiplayer. Oznacza to, że do jej rozegrania niezbędna jest obecność przynajmniej dwóch graczy. Aby rozpocząć zabawę należy więć wybrać pokój w ramach którego rozgrywane będą kolejne rundy. Domyślnie na serwerze dostępny jest jeden pokój z jednym graczem typu boot. Pozwala to na korzystanie z aplikacji nawet jeśli aktualnie do serwera nie sa podłączeni inni użytkownicy. Użytkownik ma także możliwość utworzenia nowego pokoju z dowolnie wybrana początkową liczbą grzczy typu boot.



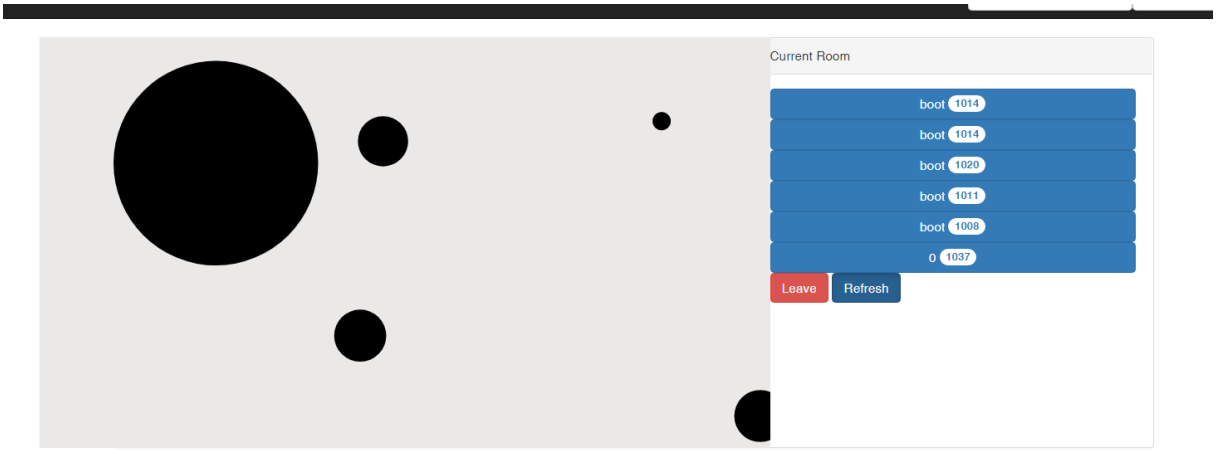
Rysunek 2 Wybór pokoju w ramach którego będzie toczona rozgrywka

Po wybraniu pokoju zostaną wyświetlone informacje na temat graczy w nim sie znajdujących. Gra abooble składa się z kolejnych rund. Runda kończy się z momentem kiedy wszystcy użytkownicy wykonanją ruch. Polegaja on na narysowaniu koła. Po każdej rundzie następuje podsumowanie. Użytkowikowi zostja przedstawione ruchy przciwników a ranking punktowy jest aktualizowany.



Rysunek 3 Panel informacyjny po dołączeniu do pokoju.

Użytkownikowi otrzymuje premie punktową za każde koło przeciwnika z którym jego koło ma część wspólną. W sytualcji zaś kiedy koło przeciwnika jest całkowicie zawarte w kole urzytkownika zostaje on ukarany przez zabranie 2 punktów z jego konta.



Rysunek 4 Ekran aplikacji po zakończonej rundzie - podsumowanie.

Do udziału w grze nie ma potrzeby rejestracji. Użytkownik może natomiast dokonać logowania. Dzięki temu podczas rozgwywki będzie widoczy pod wybranym unikalnym w ramach serwera nickem. W przypadku użytkoników niezalogowanych nazwa użytkownika wynika bezpośrednio z identyfikatora sesji jaki temu użytkownikowi został przyznany.

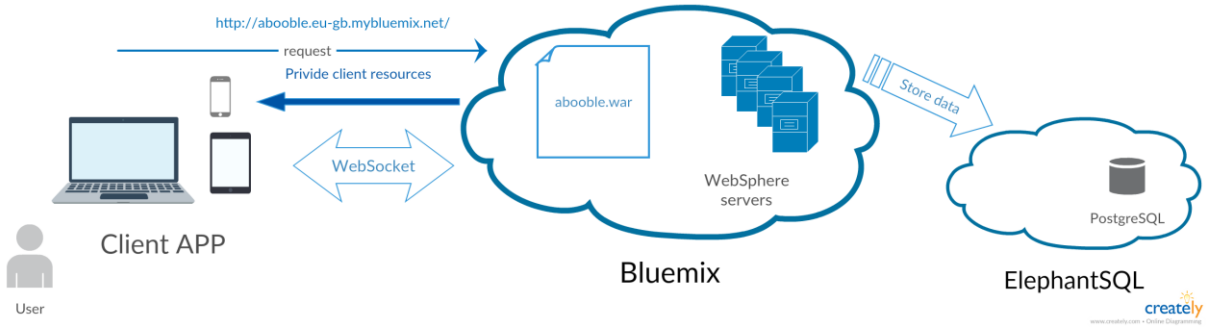
3. Funkcjonalność systemu niewidoczna dla użytkownika

Aplikacja zawuera w sobie moduł bazodanowy. Nie jest on bezpośrednio wykorzystywany przez użytkownika. Aplikacja nie przechowuje informacji o swoich użytkownikach w sposób trwały, nie prowadzi rankingów. Natomiast pozwala na śledzienie rozgrywanych partii. Modół ten ma na celu zgromadzenie informacji, które następnie będą wykorzystane jako dane wejściowe do uczenia sieci nauronowych.

W celu utwralenia tych danych wykorzystano bazę danych postgreSQL dostępną w ramach chmury ElephantSQL.

4. Architektura systemu

Aplikacja została stworzona w języku Java, w technologii Java EE 7. Program został wdrożony w chmurze Bluemix na serwerze WAS Liberty. Komunikacja między modułem serwera i klienta odbywa się przy pomocy technologii WebSocket. Po stronie serwera użyta została biblioteka javax.websocket.api. Do mapowania obiektowo – relacyjnego użyto JPA. Jako bazę danych wykorzystano bazę postgreSQL. Wdrożenie aplikacji następuje przy pomocy narzędzia Cloud Foundry CLI. Dla użytkownika końcowego przygotowana została aplikacja kliencka umożliwiająca korzystanie z serwisu przy pomocy przeglądarki internetowej. Ogólna architekture przedstawia Rysunek 5.



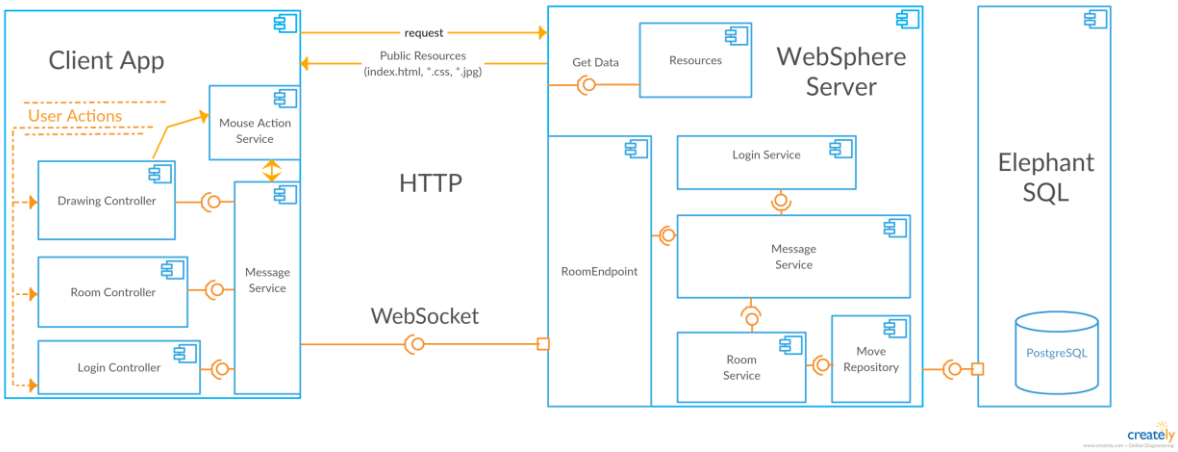
Rysunek 5 Ogólna architektura systemu.

Komunikacja między klientem i serwerem odbywała się przy pomocy wiadomości. Każda z nich o ściśle określonej zawartości opatrzona została typem – wcześniej umówionym identyfilatorem *messageType*. Dzięki temu po odebraniu wiadomości część serwerowa i kliencka mogła właściwie obsłużyć dane. Wiadomości przekazywane było pomiędzy komponentami w formacie JSON. Do kodowania i dekodowania wiadomości zapisanych w formacie JSON po stronie serwera użyto biblioteki *gson*. Wiadomości z serwera i klienta miały inną strukturę. W systemie obowiązują dwa modele. Drugi model został wprowadzony, aby w łatwy sposób przesyłac dane w sieci. Umożliwił ograniczenie widoczności niektórych pól przed dostarczeniem ich do klienta. Poniżej znajduje się obiekt wiadomości która wysyłana jest przez serwer.

```
public class CServerMessage {
    private String messageType;
    private String status;
    private List<RoomProxy> rooms = new LinkedList<>();
    private CRoom room;

     // ....... getters and setters

}
```



Rysunek 6 Szczegółowa architektura systemu.

Rysunek 6 przedstawia bardziej szczegółową architekturę systemu.

5. Testowanie

W celu przetestowania aplikacji użyto biblioteki jUnit. Jej zastosowanie pozwoliło nie tylko poprawić jakość pisanego kodu ale również usprawnić proces tworzenia aplikacji, poniważ błędy były wychwytywane już na początkowym etapie tworzenia aplikacji. Poniżej zamieszczono przykładowy kod źródłowy przykładowych testów. Pierwszy z nich testuje poprawność jenego z narzędź używanych w ramach aplikacji. Drugi sprawdza poprawność zachowania systemu, po symulowanym dostarczaniu mockowej wiadomości od klienta.

```
@Test
public void testOverlappingOverlaps() {
    //given
    Gamester user = new Gamester();
    user.setMove(new Move(0, 0, 10));
    Gamester user2 = new Gamester();
    user2.setMove(new Move(0, 0, 9));

    //when
    boolean result = PointsEngine.doesFirstOverlap(user, user2);

    //then
    assertTrue(result);
}
```

```java
@Test
public void testOverlappingNoCommonPart() {
    //given
    Gamester user = new Gamester();
    user.setMove(new Move(0, 0, 10));
    Gamester user2 = new Gamester();
    user2.setMove(new Move(0, 0, 9));

    //when
    boolean result = PointsEngine.doesFirstOverlap(user2, user);

    //then
    assertFalse(result);
}

@Test
public void testOverlappingNoCommonPart2() {
    //given
    Gamester user = new Gamester();
    user.setMove(new Move(0, 0, 10));
    Gamester user2 = new Gamester();
    user2.setMove(new Move(12, 0, 1));

    //when
    boolean result = PointsEngine.doesFirstOverlap(user2, user);

    //then
    assertFalse(result);
   }


@Test
public void testRoomEndPoint() {
    Session session = new MockSession();
    ClientMessage message = new ClientMessage();
    message.setMessageType(MessageTypes.JOIN.enumValue());
    message.setRoomId("Room0");

    roomEndpoint.onMessage(session, message);
}
```

6. Wdrożenie

Wdrożenie aplikacji przy pomocy narzędzia Cloud Foundry CLI na serwerze aplikacyjnym WasLiberty w chmurze Bluemix.

6.1. Tworzenie archiwum

Deployment aplikacji na serwerze aplikacyjnym jest możliwy przez umieszczenie na nim archiwum war projektu. Projekt abooble to projekt typu Java Maven Project. Pozwala to utworzyc wrchimum przez wykonanie komendy:

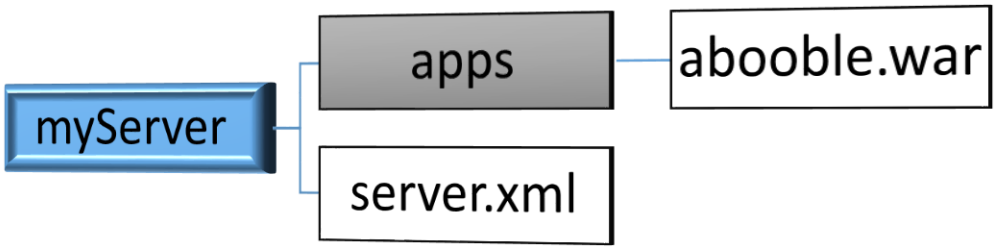
~ *mvn compile war:war*

Operacja taka wymaga wcześniejszej instalacji programu mavaen. Oraz skonfigorowania zmiennej systemowej mvn, tak aby wskazywała na lokalizację zainstalowanego programu.

Alternatywnie można tez utworzyc archimum bezpośrednio z poziomu IDE. W przypasku eclipse należy prawym przyciskiem wspacać projekt, wybrac obcję *Export* następnie *WAR file,* wybrać lokalizację oraz nazwę tworzonego archiwum i zakończyc akcje kreatora.

6.2. Umieszczenie aplikacji w chmurze.

Wdrożenie aplikacji w chmurze przy pomocy narzędzie *CF* wymaga zastosowania pewnej konwencji. Należy utworzyc strukturę katalogów jak na rysunku 5.



Rysunek 7 Lokalna struktura katalogów

Aby umieścić uruchomić serwer aplikacyjny w chmurze należy z poziomu katalogu myServer wykonać polecenie:

```
cf push webticket -m 512M -p myServer -b liberty-for-java
```

Jego wynonanie wymaga wcześniejszego zalogowania się do systemu.

```
cf login
```

Konfiguracji adresu serwera: `cf api https://api.eu-gb.bluemix.net`

Podanie adresu email z którym konto jest powiązane: Antoni.Cepak@fis.agh.edu.pl

Podanie hasła: ******

Po pomyślnym wdrożeniu aplikacja jest dostępna pod adresem:

http://abooble.eu-gb.mybluemix.net/

Spis treści

1. Wstęp 2
2. Opis aplikacji 2
3. Funkcjonalność systemu niewidoczna dla użytkownika 4
4. Architektura systemu 5
5. Testowanie 6
6. Wdrożenie 8
   6.1. Tworzenie archiwum 8
   6.2. Umieszczenie aplikacji w chmurze. 8