Buzzword-Bingo Dokumentation für die Blockveranstaltung Verteilte Systeme (VSY) im SEP-Semester SS 2011

Kristian Kraljic Pascal Krause Manuel Gedack Timo Kodowski Marcel Sinn Markus Opitz

April 8, 2011

Contents

1 Einführung

Im Rahmen dieses Projektes soll das beliebte Spiel Buzzword-Bingo (BWB) auf einem verteilten System entworfen und implementiert werden.

Das Spiel Buzzword-Bingo ist die gelungene Verknüpfung einerseits eines klassischen Bingospiels, bei welchem Zahlen durch einen Spielleiter aufgerufen werden, und die Teilnehmer auf ihren (unterschiedlichen) Spielkarten versuchen, alle abgebildeten Zahlen schnellstmöglich anzukreuzen, um als erster "Bingo" zu rufen, und dadurch das Spiel zu gewinnen.

Im Rahmen des BWB-Spiels werden anstatt der Zahlen allerdings Begriffe ("Buzzwords") verwendet, die üblicherweise einem tödlich langweiligem Vortrag entstammen, und insofern das Auditorium vor dem Wachkoma behüten soll.

Buzzword-Bingo enthält jedoch noch ein weiteres Element, welches dem bekannten Spiel Tic-Tac-Toe entnommen wurde:

Die Buzzwords sind in einer quadratischen nxn Matrix angeordnet (häufig 5x5), und derjenige Spieler welcher als erster eine horizontale, vertikale oder diagonale Linie durch seine angkreuzten Begriffe bilden kann, durch lautes Ausrufen von "Buzzword-Bingo" kundtut, daß er das Spiel gewonnen hat.

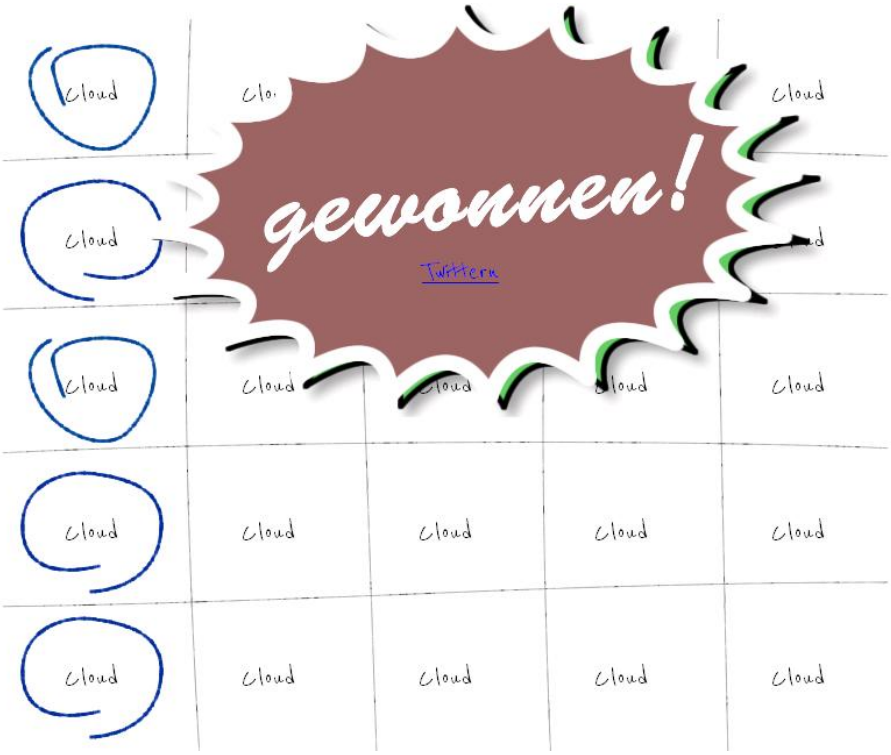
In manchen Kulturkreisen soll dieser Ausruf durch ein kräftiges "Bullshit-Bingo" ersetzt worden sein.

Böse Zungen behaupten, daß eine klare proportionale Abhängigkeit zwischen dem Langeweilefaktor eines Vortrags/Präsentation und der Spieldichte von BWB bestehen soll.

Komplexe Spielkarte:

Cloud	Cloud	Cloud	Cloud	Cloud
Cloud	Cloud	Cloud	Cloud	Cloud
Cloud	Cloud	Cloud	Cloud	Cloud
Cloud	Cloud	Cloud	Cloud	Cloud
Cloud	Cloud	Cloud	Cloud	Cloud

And the winner is:



Oder eben auch “BUUULLSHIT-BINGOOOO”!

2 Anforderungen an das Projekt

Im Rahmen dieser SEP Blockveranstaltung soll ein Spiel Buzzword-Bingo als grafisch verteilte Anwendung erstellt werden. Dabei dürfen folgende Plattformsysteme verwendet werden:

- Mobilanwendung unter Android oder iPhone/iPod touch.
- Webanwendung in einer Cloud-Plattform (PaaS) wie Google App Engine oder AWS Elastic Beanstalk.
- Webanwendung mit HTML und PHP.
- Java-Anwendung mit AWT oder Swing.
- Kommandozeilenanwendung mit grafischer Bibliothek ncurses.
- Anwendung mit GTK+ oder QT.

Die Wahl der entsprechenden Plattform wird jedem Entwicklerteam freigestellt. Ebenso kann jedes Team frei entscheiden, ob es eine Client-Server-Architektur oder ein Ad-hoc-Netzwerk erstellt.
Folgende Voraussetzungen sind jedoch zu beachten:

- Über den Client der Anwendung sollen Spieler ein neues Spiel eröffnen oder einem bestehenden Spiel beitreten können.
- Will ein Spieler ein neues Spiel eröffnen, kann er im Client festlegen, wie groß das Spielfeld ist und welche Buzzwords zufällig auf dem Spielfeld verteilt werden sollen.
- Der Spieler/Client, der ein neues Spiel angelegt hat, steuert den Server. Ob es sich dabei um einen zentralen Server handelt oder ob der erste Client gleichzeitig der Server ist, liegt bei den Entwicklern.
- Die Spieler sollen über ein Netzwerk miteinander spielen können.
- Im Client kann man eine Liste der Mitspieler einsehen.
- Hat ein Spieler gewonnen, werden die Mitspieler benachrichtigt.

Zusätzlich zu der funktionierenden Software soll eine 10 seitige Dokumentation erstellt werden, und zwar ausschließlich auf Basis des Latex Textsatzprogrammes. Darin sollen eine Spielanleitung sowie die Darstellung der Softwarearchitektur enthalten sein. Weiterhin sollen Vor- und Nachteile der gewählten Softwarestruktur erläutert werden.

3 Darstellung der Softwarearchitektur

Für die Realisation unserer Projektarbeit haben wir uns nach umfassenden Diskussionen und mehrfacher Abwägung für eine klare Client-Server Architektur entschieden.

Der Vorteil gegenüber einer Ad-hoc Struktur besteht insbesondere darin, daß ein einmalig aufgesetzter Server problemlos mit mehreren Clients betrieben werden kann.

Die Clients können ihrerseits auf die generierten Spielkarten bzw. die Liste der gezogenen Buzzwords zugreifen, ohne daß sie diesbezüglich weitere Informationen untereinander austauschen müssten.

Der Nachteil gegenüber einer Ad-hoc Struktur besteht im erhöhten Installations- und Maintenance-aufwand für den Server.

Da sich dieser erhöhte Aufwand lediglich einmalig ergibt, hielten wir die Client-Server Struktur für die rationalere und ökonomisch sinnvollere Lösung.

Um einen strukturierten Austausch von Informationen zwischen den Systemteilnehmern zu erreichen, entschieden wir uns für den Einsatz von Datencontainern.

Aus naheliegenden Gründen insbesondere dem Einsatz eines JAVA-Clients, verwendeten wir eine JSON Bibliothek (JavaScript Object Notation).

Da diese in verschiedenen Punkten nicht unseren Anforderungen entsprach, wurde sie speziell für die Realisation unseres BWB Spiels modifiziert.

4 Die Struktur der Server-Komponente

Die Server Komponente besteht aus einem PHP-Server mit einer angeschlossenen MySQL Datenbank.

Der PHP Server stellt dabei eine bestimmte Anzahl von im Spiel benötigten Web-Services zur Verfügung, wohingegen die MySQL Datenbank die Wortlisten der unterschiedlichen Kategorien verwaltet.

Durch den Aufruf des Webservices GET werden mittels Übergabeparameter an den jeweiligen Spieler(-Client) übertragen, die zu einer Rückgabe verschiedener Rückgabeparameter führt, die von dem jeweiligen aktuellen Spielstand abhängen.

Allgemein sind folgende Webservices implementiert worden:

- Words.php

Parameter: gameid , category
Funktion: Liefert alle Wörter eines Spiels oder einer Kategorie
Ergebnis: {"status":"success","words":[...]}

- New.php

Parameter: player, title, category, rows, columns
Funktion: Erstellt ein neues Spiel
Ergebnis: {"status":"error","message":"XXX"} falls ein Fehler auftritt {"status":"success","gameid": XXX } falls kein Fehler auftritt

- List.php

Parameter: keine
Funktion: Listet alle offenen Spiele auf
Ergebnis: {"status":"success",games:[...]}

- Start.php

Parameter: player, gameid
Funktion: Startet das Spiel (nur vom Spielersteller aufrufbar)
Ergebnis: {"status":"success"}

- Get.php

Parameter: player, gameid

Funktion: Startet das Spiel für einen Benutzer, falls er noch nicht im Spiel ist. Liefert auch die aktuellen Wörter des Spiels und das eigene Brett zurück.

Ergebnis: {"status":"won"/"lost","players":[...]} lost/won bezieht sich dabei auf den aktuellen Player {"status":"wait","players":[...]} falls das Spiel noch nicht gestartet wurde {"status":"success","players":[...],"board":[...],"words":[...]}

- Bingo.php

Parameter: player, gameid

Funktion: Wird vom Spieler aufgerufen wenn er denkt er hätte gewonnen

Ergebnis: {"status":"won"} {"status":"lost"}

- Word.php

Parameter: words, category

Funktion: Fügt ein neues Wort in die Datenbank hinzu Ergebnis: {"status":"success"}

5 Die Struktur der Client-Komponente

Die Client-Komponente wurde vollständig in JAVA realisiert, unter Verwendung der JSON Bibliothek.

Ankommende JSON Objekte werden schrittweise in JAVA Datenstrukturen (beispielsweise Arrays) umgewandelt, und in dieser From als originäre JAVA Datenstruktur weiterverwendet.

Dabei wurden folgende Methoden verwendet:

- startGame()

Funktion: Startet das erstellte Spiel. Sollte man versuchen ein Spiel zu starten, dessen Host man nicht ist, wird eine NoGameHostException geworfen. Ansonsten wird true zurückgegeben. Nach dem Starten wird sofort das Board geholt und die ersten Wörter erstellt.

- bingo()

Erklärung: Schickt an den Server die Nachricht, dass man Bingo hat. Es wird true zurückgegeben, falls es so ist, andernfalls wird false zurückgegeben. Ein Aufruf von mehr als einmal ist nicht möglich, da ein bingo Aufruf ohne tatsächliches Bingo zum verlieren des Spiels führt.

- getBoardFields()

Erklärung: Gibt das aktuelle Spielbrett zurück. Das Spielbrett besteht aus einer bestimmten Anzahl von Feldern (den Fields). Jedes Field besitzt eine column, eine row und das darin enthaltene Wort. Entsprechende Getter sind vorhanden. Reihenfolge der Liste kann von der eigentlichen Reihenfolge der Wörter abweichen.

- getDrawnWords()

Erklärung: Gibt die Liste der gezogenen Wörter zurück. Sie enthält alle bereits gezogenen Worte. Die neuen Worte werden an die Liste hinten angehängt.

- joinGame(int gameID)

Erklärung: Es wird ein Spiel über eine gameID gejoint. Die gameID kann aus der Liste von Games, die getGame zurückliefert eines jeden offenen Games entnommen werden.

- createGame(int columns, int rows, String category, String gameTitle)

Erklärung: Erlaubt es, ein neues Spiel zu erstellen. Es muss die Anzahl der columns und rows übergeben werden, deren Wert mindestens 1 sein muss. Die category ist in unserem Fall immer buzzwords.

- getWordsInDB(String category)

Erklärung: Liefert sämtliche Wörter zurück die einer Kategorie angehören. Damit kann man sich vor dem Erstellen eines Spieles die möglichen Worte betrachten.

- getGames()

Erklärung: Liefert eine Liste sämtlicher offenen Games. In Game wird die gameID, der Name des Hosts, der Titel des Spiels und der Name der Kategorie die gespielt wird verwaltet.

- setPlayerName(String newName)

Erklärung: Erlaubt das nachträgliche Ändern des Spielernamens.

- getPlayerName()

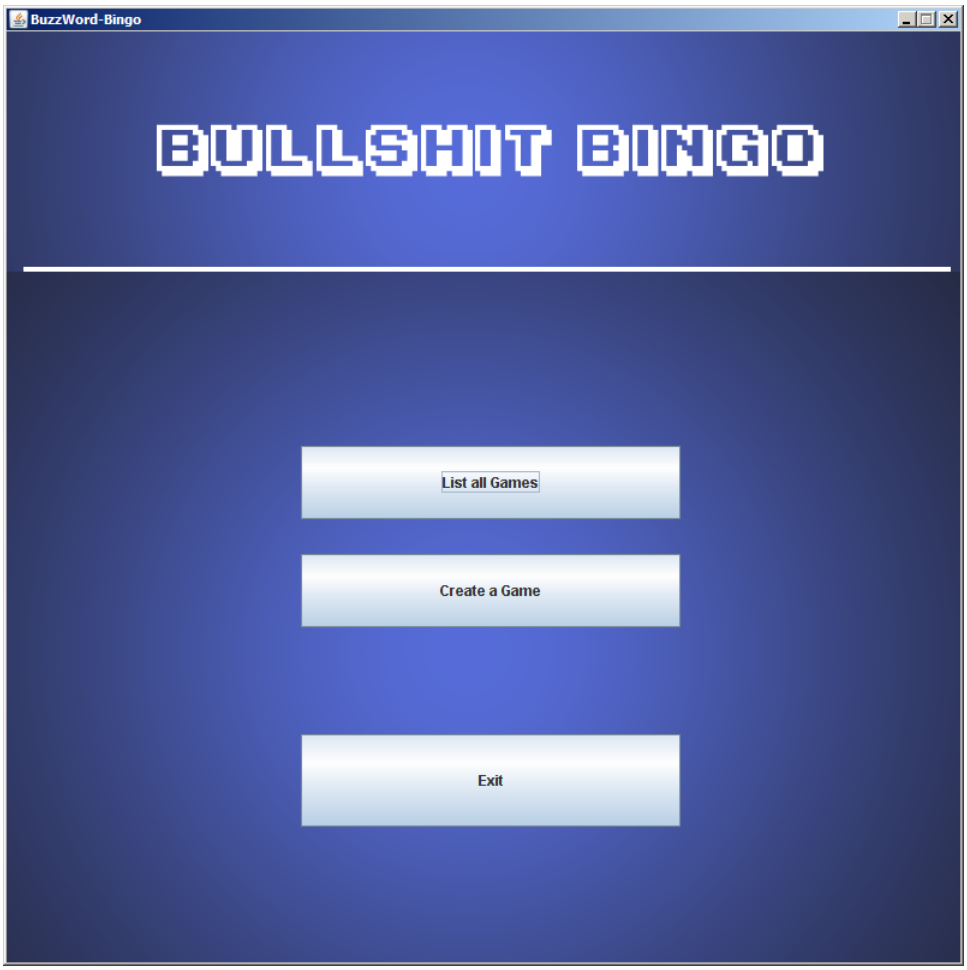
Erklärung: Liefert den aktuellen Spielernamen zurück.

- getBoardSizes()

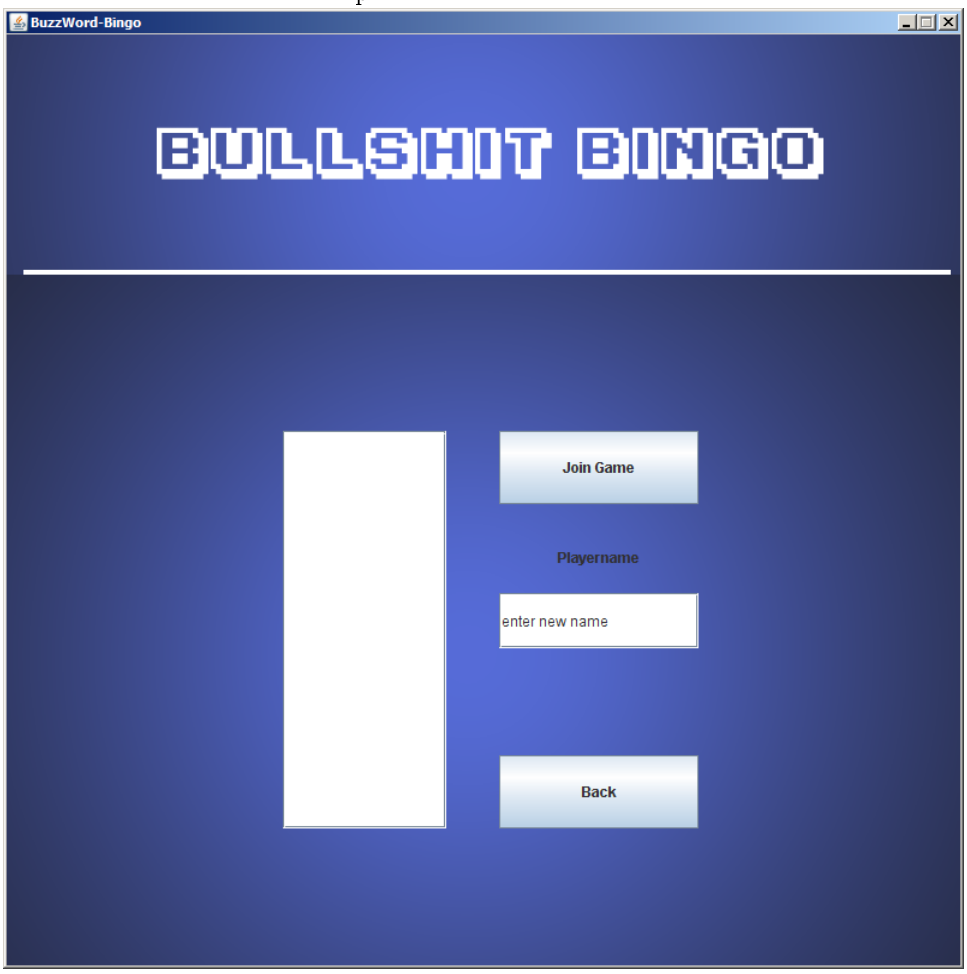
Erklärung: Liefert ein 2 Dimensionales Array zurück, falls das Spiel bereits gestartet wurde. In der Position[0] befindet sich die Anzahl der Rows in [1] die Anzahl der Columns.

6 Das Design des Graphical User Interface

Der Startbilschirm der JAVA-App präsentiert sich folgendermaßen:

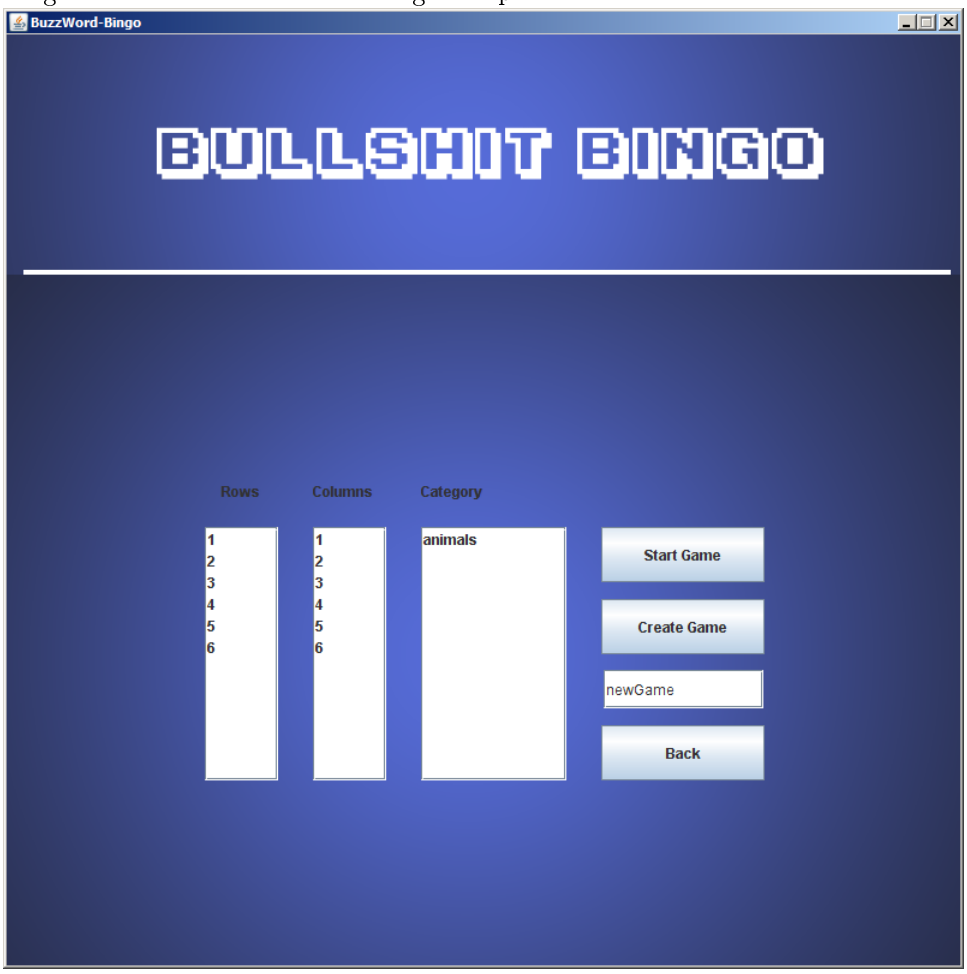


Eine Liste aller bestehenden Spiele lässt sich dann aufrufen:



Ein laufendes Spiel wird folgendermaßen dargestellt:

Zu guter letzt lässt sich noch ein eigenes Spielfeld aufbauen:



Vielen Dank für die Aufmerksamkeit!