1 Vorüberlegungen

1.1 Vision

Die Vision des Projekts besteht in einer ersten prototypischen Umsetzung des Szenarios. Letztlich soll die Anwendung Bullshit Bingo einsetzbar sein. Die Nutzung des Spiels und der Einsatz von der Programmiersprache Ruby wurden ebenfalls vorgesehen. Die Anwendung sollte zudem im Browser lauffähig sein, um so auf mehreren Geräten und Plattformen einsatzfähig werden. Zur Bedienung sind Maus und Tastatur vorgesehen. Anderseits der Multitouch-Technologie und dem Eye- und Motiontracker weiterführende Möglichkeiten, die im Einsatz kommen könnten.Die modernisierten Oberflächen werden nicht nur optisch aufgewertet, sondern bieten dem Anwender einen erhöhten Nutzen und Bedienkomfort.

1.2 Szenario

Diese Web-Anwendung soll allen Teilnehmern das Spielen von Bullshit Bingo ermöglichen. Mit Spielen sind in diesem Szenario in erster Linie einen Spieler gemeint, der allein spielt, oder andere Teilnehmer zu dem schon bestehenden Spiel einladen kann. Auf Seite der Teilnehmer soll, beim Einsatz unterschiedlicher Webbrowsers, nicht mehr unterschieden werden müssen.

1.3 Ziele

Mit Blick auf die Vision ist das Ziel in erster Linie das Spiel lauffähig zu machen. An Zweiter Stelle muss das Spiel der Einsatz mehreren Teilnehmern gewährleiten. Unter anderen soll es auch nicht der Einsatz verschiedenen Webbrowser eine Stolperfalle für das Projekt sein. Nach Möglichkeit soll bis zum 08.03.2011 ein lauffähiges Programm entstanden sein. In den Anwendungen können mehr fachlich getriebene Anforderungen eingearbeitet werden, weil weniger technische Einschränkungen der Oberflächen bestehen. Kurz zusammengefasst kann man ein lauffähiges Programm erwarten, die leicht zu bedienen ist, auf

verschiedenen Webbrowsern läuft und dem Spieler Spaß beim Spielen bereitet.

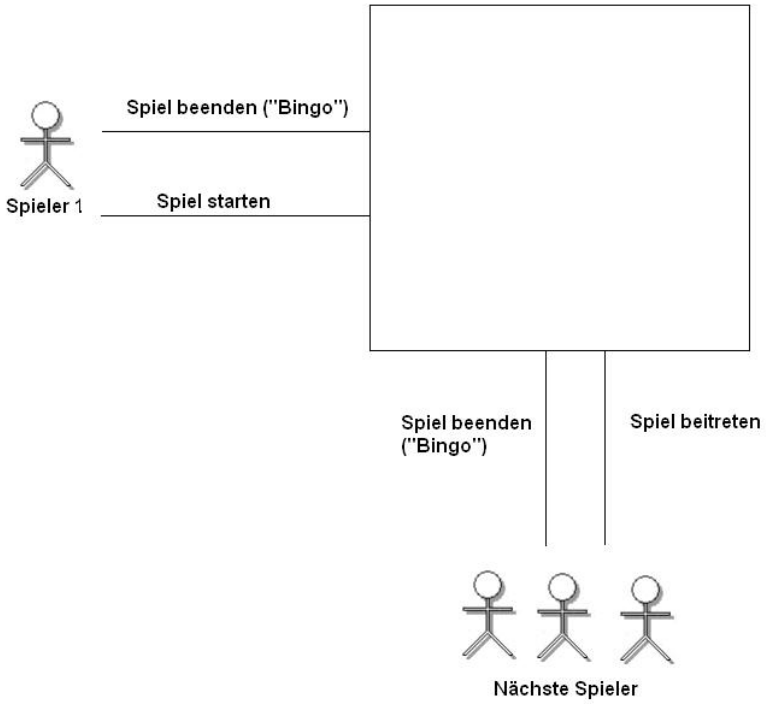
1.4 Vorgehensweise

Zu aller erst soll die nötige Infrastruktur geschaffen werden, was sich über eine Entwicklungsumgebung und mehrere verschiedenen Internetbrowser erstreckt. Als nächstes soll der Einsatz von Ruby untersucht werden, um die Möglichkeiten und Grenzen seiner Fähigkeiten in Bezug auf das Szenario zu bewerten.

1.5 Web-Browser

Im Rahmen des Projekts wurde ein kurzer Test der Anwendung in verschiedenen Browsern vollzogen. Die Anwendung soll in Firefox 3.x, 4.x, Chrome 10.x, Internet Explorer 9 sowohl 32 als auch 64 Bit und Safari ohne Probleme laufen. Die Anwendung funktionierte erstaunlicherweise in vollem Umfang beim allem oben erwähnten Browser. Je nach weiterer Ausführung des Projektes sind hier aber durchaus Probleme zu erwarten.

1.6 Kontextdiagramm



Der erste Spieler, der das Spiel startet, kann den Namen für das Spiel wählen und weitere Spieler einladen. Die nächsten, die die Seite aufrufen, treten dem Spiel mit einem selbst gewählten Namen bei, wobei jedem ein neues Spielfeld zur Verfügung gestellt wird. Jeder der Spieler soll dann in der Lage sein, das Spiel zu beenden, wenn er eine ganze Zeile oder Reihe von Buzzwords angekreuzt bzw. markiert hat.

1.7 Visionsszenario

Seit ein Paar Monaten hat die PR-Abteilung einer großen weltweit aktiven Firma einen neuen Manager, der an einer renommierten amerikanischen Universität studiert hat und für seine sinnleere, dafür aber langwierige Reden berüchtigt ist.

Die Abteilung von Peter hat wieder mal ein langweiliges All-Hands-Meeting, wo der Manager seine Vision für die globale PR-Strategie der Firma präsentieren wird. Peter hat neulich im Internet über Bullshit-Bingo gelesen und schafft es seine Kollegen für die Idee zu begeistern während der Präsentation zu spielen. Da es aber möglicherweise schlechte Konsequenzen für ihre berufliche Zukunft haben könnte, mitten im Meeting „Bullshit!“ zu rufen, sucht Peter nach einer elektronischen Variante des Spiels und findet die vorliegende Applikation im Web. Das Meeting beginnt und Peter startet das Spiel, wobei er es benennt und seinen Spielernamen auswählt. Den generierten Link schickt er dann über ein IM-Programm seinen drei Kollegen im Raum. Maria, Werner und Otto klicken auf den Link und geben jeweils ihren Namen ein. Wenn einer von ihnen ein der Buzz-Wörter hört, markiert er sie. Peter ist mit den Wörtern in seinem Bogen unzufrieden und möchte ein neues Spiel starten, dafür muss aber das aktuelle beendet werden, weshalb er versucht seine Kollegen zu betrügen und klickt auf „bingo!“. Das wird aber vom Server nicht akzeptiert, und er bekommt die entsprechende Fehlermeldung. Nach 60 Minuten hört Otto „Synergie-Effekte“, was das letzte unmarkierte Wort in einer Zeile seines Bogens ist und klickt dann auf „bingo!“, um es den anderen mitzuteilen. Auf den Bildschirmen von allen Spielern wird sein Bogen mit den markierten Feldern angezeigt und keiner

kann etwas an seinem Spielfeld ändern. Glücklicherweise ist das Meeting gleich danach zu Ende, sodass keine Zeit für ein weiteres Spiel bleibt.

2 Architektur

2.1 Zentraler Server- Client

Der Server berechnet den ganzen Simulationszustand und übernimmt die Spiellogik. Der Client sorgt nur dafür, dass der Simulationszustand bei jedem Benutzer angezeigt wird. Die Kommunikation läuft folgendermaßen ab: Der Client teilt dem Server Kommandos mit und der Server berechnet den neuen Zustand. Anschließend werden die Updates an den Client geschickt (das gleiche geschieht parallel zwischen jedem einzelnen Client und dem Server). Auf dem Bildschirm des Client-Rechners wird dann der aktuelle Spielzustand angezeigt (bzw. die durch die Spiellogik bestimmten Segmenten des Zustandes). Damit wird eine Konsistenz der Daten, die auf den Client-PCs angezeigt werden sichergestellt und es wird kein groäer Administrationsaufwand benötigt. Eine Schwäche des gewählten Modells ist die potenzielle Single-Point-Failure: falls der Server versagt, funktioniert die Anwendung nicht mehr. Außerdem wird der Server zum Flaschenhals: falls viele Clients auf ihn zugreifen wird er zu einer knappen Ressource. Es kann aber mit relativer Sicherheit davon ausgegangen, dass diese Gefahr in diesem konkreten Fall nicht vorliegt, da die Anzahl der Spieler (und somit der Clients) relativ gering ist – auf jeden Fall weniger als 100.

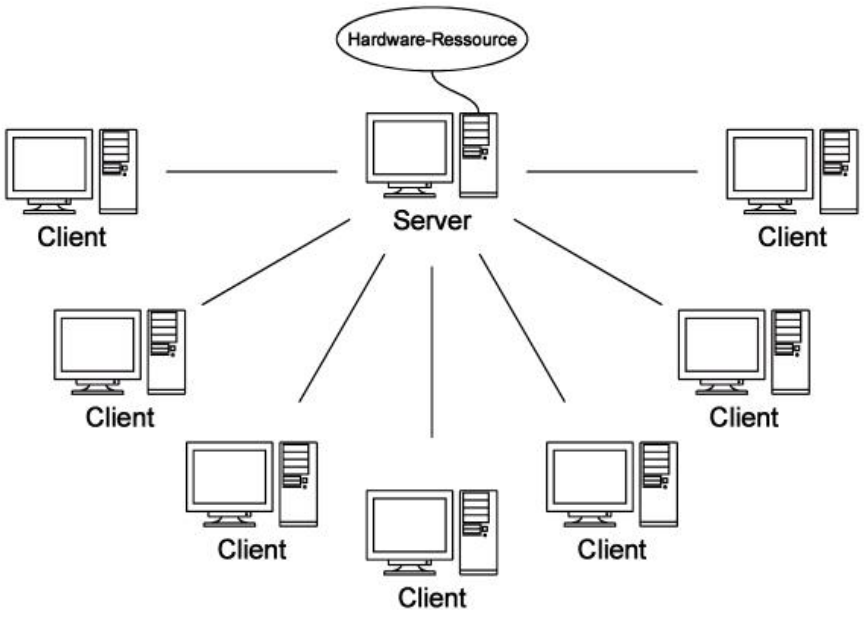


Figure 1: Client-Server-Architektur

2.2 Model-View-Controller

Eingesetzt wurde das Paradigma Model-View-Controller. Das bedeutet, dass es zum einen eine Benutzeroberfläche gibt (realisiert als HTML-Seite im Browser), womit der Benutzer interagiert und seine Handlungen ausführt. Zum anderen werden alle Datenabfragen und –ströme vom Controller bearbeitet und gelenkt. Dabei greift der Controller auf in einer Datenbank gespeicherten Modellen, die die erlaubten Handlungen und die Reaktionen auf bestimmte Ereignisse vorschreiben.

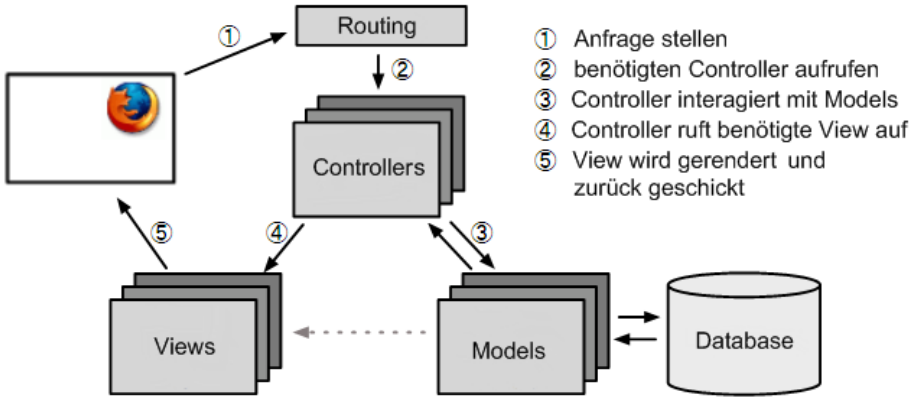


Figure 2: Model-View-Controller

2.3 Klassendiagramm

Die Klassen wurden in Ruby geschrieben (im Ruby on Rails Framework). Dabei besteht jedes Sheet aus Feldern, die mit Buzzwords belegt sind; jedes Feld hat eine Position als eindeutiger Bezeichner und ein Buzzword als Inhalt; außerdem kann es entweder markiert oder nicht markiert sein. Die Klassen Spieler (Player) und Bogen (Sheet) werden in der Klasse Participaction verknüpft. Ein komplettes Spiel seinerseits enthält entsprechend eine oder mehr Spieler und die dazugehörenden Participations:

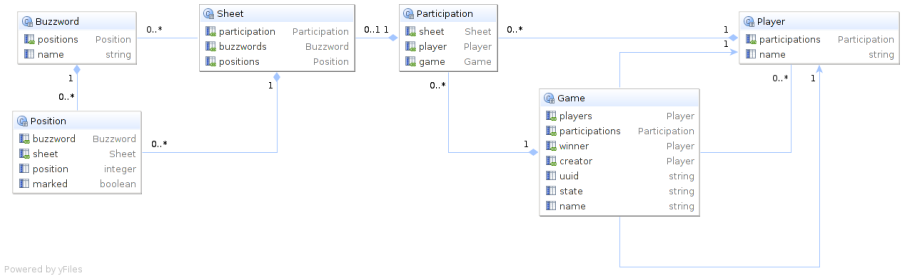


Figure 3: Klassendiagramm

3 Eingesetzte Technologien

3.1 Ruby

Ruby ist eine objekt-orientierte dynamische Programmsprache, die Mitte der 90er Jahre in Japan entwickelt worden ist. Der Name soll eine Anspielung an Perl sein, da Ruby ähnlichen Zwecken (als objektorientierte Skriptsprache) dienen sollte.

3.2 Javascript

Javascript ist eine Skriptsprache um dynamische Websites zu erschaffen. Mit Hilfe von Javascript kann wahlweise ein Teil des Inhalts einer Internetseite neu generiert werden und nachgeladen werden, sodass nicht die komplette Seite wegen kleinen Veränderungen neu geladen werden muss. Javascript ist eine Clientseitige-Script-Sprache und läuft auf dem Client und nicht auf dem Server auf dem die Seite gespeichert ist. Dabei ist zu beachten, dass die Ressourcen des Clients für die Veränderung des HTML-Dokuments verantwortlich beansprucht werden und somit der Server nicht zusätzlich belastet wird.

3.3 AJAX

AJAX ist keine selbständige Technologie sondern lediglich die Zusammenfassung schon bestehender Techniken zu einem Begriff (AJAX steht für Asynchronous Javascript and XML). Wie schon aus dem Namen ersichtlich, ermöglicht AJAX die asynchrone Datenübertragung zwischen Server und Client. Dabei werden Browser Events von Javascript Eventhandlern abgefangen und ein XMLHttpRequest wird angelegt, das Aufrufparameter weiterleitet und eine Callback-Funktion spezifiziert (die ihrerseits die Behandlung der Antwort bestimmt). Auf dem Server wird die eingetroffene Nachricht bearbeitet und eine Antwort wird an den Client zurückgesendet. Dort werden die Parameter die eventuell geändert werden müssen aktualisiert. Die Kommunikation zwischen Server und Client geschieht im XML-Format:

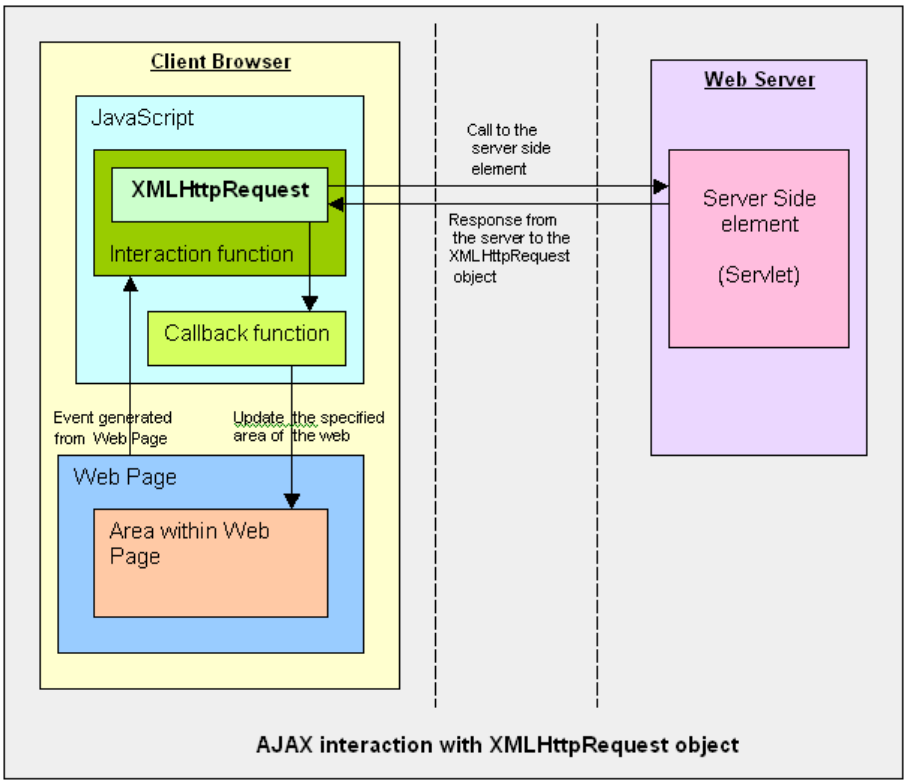


Figure 4: AJAX Interaktion

Bekannte Beispiele für den Einsatz von AJAX sind viele der Projekte von Google wie z. B. Google Maps (Reaktion auf Benutzerhandlungen – zoomen, verschieben) oder Google Suggest (nach Eingabe eines Zeichens im Suchfeld, werden vom Server passende Suchbegriffe gefordert und dem Benutzer vorgeschlagen).

3.4 HTML

HTML steht für „Hypertext Markup Language“ und ist eine textbasierte Auszeichnungssprache zur Strukturierung von Inhalt und Text. Außerdem kann man damit per Verweise Bilder, Multimedia-Dateien etc. einbinden. HTML sollte nur zum Darstellen des Inhaltes benutzt werden, die Gestaltung ist nicht die Aufgabe von HTML sondern wird von CSS übernommen.

3.5 CSS

CSS ist die Abkürzung von Cascading Style Sheets und lässt sich mit „stufenförmigen Stilvorlagen“ übersetzen. Vereinfacht gesagt geht es bei CSS um die Formatierung von HTML-Elementen. Mann kann mehrere CSS-Sheets in eine HTML-Datei einbinden, sodass das Aussehen von der Seite zum Beispiel für verschiedene Medien (Rechner, Handy, Drucker etc.) anders gestaltet wird. Auch lässt sich mit CSS das Layout verändern.

4 Vor- und Nachteile der Lösung

Die ausgewählte Implementierung hat den Vorteil, dass keine Installation von Client-Software beim Benutzer notwendig ist. Eine andere Vorgehensweise wäre zum Beispiel eine Java-Server-Anwendung auf dem Server-Rechner und Client-Anwendungen auf den Client-Rechnern zu installieren. Diese Lösung hätte aber im Vergleich zu unserer zwei weitere Nachteile:

1. der Server ist festgelegt
2. JRE muss auf allen Rechnern vorhanden sein (was man zwar heutzutage im Normalfall voraussetzen kann, aber an sich eine Beschränkung ist)

Alles, was man für die vorliegende Applikation braucht, ist einen Browser und eine Internet-Verbindung. Das könnte man auch teilweise Nachteil einschätzen, weil man nur online spielen kann. Andererseits gibt es eigentlich keine gleichwertige Alternative, da das Spiel im Netz gespielt werden soll und dafür extra ein Netz aufzubauen wäre überflüssiger Aufwand, wenn man die Protokolle, Ressourcen und Kanäle des WWW benutzen kann. Da der Datenaustausch im XML-Format geschieht und lediglich Standardtechnologien wie HTML und CSS benutzt werden, liegt auch eine hohe Kompatibilität vor: das Spiel wird von den aktuellen Versionen aller verbreiteten Browser unterstützt (Internet Explorer, Firefox, Chrome, Safari). Da http zustandslos ist (es besteht also keine dauernde Verbindung zwischen Server und Client und dementsprechend Server Push nicht möglich), bietet der Einsatz von AJAX einen Umweg, indem durch AJAX-Polling alle 2 Sekunden der Client den aktuellen Spielzustand beim Server abfragt. Somit entsteht bei dem Spieler die Illusion einer direkten und unverzögerten Verbindung.

5 Link zu Spiel

http://kanetontli.no-ip.org/bingo/