Wollen Wir Wetten?

Konzeption und Realisierung einer interaktiven Smartphone-Applikation

Bachelorarbeit

Vorgelegt von: Patrice Kaufmann
Matrikelnummer: 942923

Bereich Medieninformatik
Erstkorrektur: Prof. Dr. Oliver Vornberger
Zweitkorrektur: Prof. Dr.-Ing. Elke Pulvermüller

Universität Osnabrück
2-Fächer Bachelor, Mathematik / Informatik

Erklärung zur selbstständigen Abfassung der Bachelorarbeit

Ich versichere, dass ich die eingereichte Bachelorarbeit selbstständig und ohne unerlaubte Hilfe verfasst habe. Anderer als der von mir angegebenen Hilfsmittel und Schriften habe ich mich nicht bedient. Alle wörtlich oder sinngemäß den Schriften anderer Autoren entnommenen Stellen habe ich kenntlich gemacht.

Osnabrück, den 19.05.2015

_____

(Patrice Kaufmann)

Inhaltsverzeichnis

Kapitel 1 - Einführung

1.1 Motivation

Heutzutage ist vielen Menschen in Deutschland der Begriff „Smartphone“ kein Fremdwort mehr. Nach der Markteinführung des iPhone 3 im Jahr 2008 stieg die Anzahl der verkauften Smartphones die nächsten Jahre rapide an. So beläuft sich die Anzahl der Smartphone-Nutzer in Deutschland im Jahr 2014 auf *41,1 Millionen*. Dies entspricht in Etwa bereits der Hälfte der Bevölkerung.

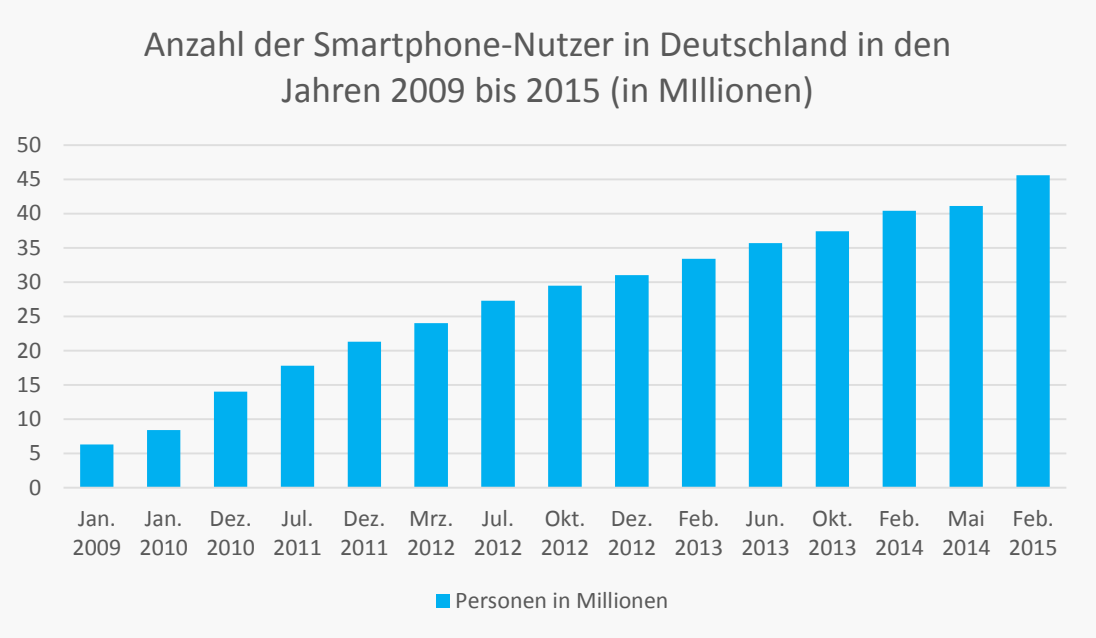


Abbildung 1 - Smartphone-Nutzer in Deutschland 2009-2015 [1]

Allseits beliebt unter den Smartphone-Nutzern sind dabei sind die verfügbaren Applikationen, die sich in Portalen wie dem Apple App Store, dem Google Play Store oder dem Microsoft Store finden und erwerben lassen. Der Markt für Applikationen ist groß und bietet viele Möglichkeiten. Mit einer guten Applikationen kann ein Entwickler viele Leute erreichen und eventuell große Gewinne einfahren. Es steht also zur Frage, ob man nicht selbst Beteiligung an diesem Markt finden kann.

Alles, was es dazu braucht, ist eine gute Idee, eine saubere technische Umsetzung und eine gute Präsentation in den Portalen. Diese Arbeit beschäftigt sich mit dem konkreten Versuch, diese Anforderungen zu erfüllen.

1.2 Aufbau der Arbeit

Diese Arbeit gliedert sich in vier Kapitel. Das erste Kapitel liefert einen thematischen Einblick in Die Smartphone-Applikation „Wollen Wir Wetten?“, indem hier primär das Konzept vorgestellt wird.

In dem zweiten Kapitel wird unter Berücksichtigung des vorgestellten Konzepts ermittelt, welche Komponenten zu realisieren sind und welche Technologien sich jeweils gut dafür eignen.

Das dritte Kapitel beschäftigt sich mit der Durchführung des Projektes. Dabei wird detailliert auf Schlüsselstellen in den verschiedenen Komponenten eingegangen.

Einen Abschluss liefert das vierte Kapitel. Es wird kritisch über die Realisierung reflektiert und ein Überblick darüber gegeben, was noch nach Markteinführung der Applikation zu tun ist.

1.3 Idee & Konzept

Das Konzept für die zu entwickelnde Applikation, genannt „Wollen Wir Wetten?“, basiert auf der grundlegenden Idee, Smartphone-Nutzern eine Plattform zu bieten, auf der sie auf viele verschiedene Sachen wetten können. Für die Einsätze sollte die Applikation den Nutzern eine eigene Währung in Form von *Wettmünzen* zur Verfügung stellen. Das primäre Ziel für einen Nutzer wird es dabei sein, sein Guthaben an Wettmünzen zu vergrößern und somit in den Bestenlisten aufzutauchen.

1.3.1 Bestenlisten

Bestenlisten stellen den primären Anreiz für die Nutzer dar, ihr Guthaben an Wettmünzen fortlaufend zu vergrößern. Es sollte dabei drei verschiedene Bestenlisten geben, welche jeweils einen unterschiedlichen Zeitraum umfassen. Es sollte das *Tagesranking*, welches sich auf die letzten 24 Stunden bezieht, das *Wochenranking*, das sich auf die aktuelle Woche bezieht und das *Monatsranking*, welches sich wiederrum auf den aktuellen Monat bezieht, geben. Damit auch neue

Nutzer eine Chance auf eine Position in der Bestenliste haben, sollte nicht der absolute Betrag an Wettmünzen als Bewertungsgrundlage herangezogen werden, sondern der relative Anstieg, gemessen vom aktuellen Zeitpunkt bis hin zum Start des jeweiligen Bewertungszeitraumes.

1.3.2 Erhalt von Münzen

Es wird verschiedene Wege geben, in „Wollen Wir Wetten?“ an Wettmünzen zu gelangen. Jedem Nutzer sollte zunächst bei Registrierung in der Applikation ein *Startguthaben* an Wettmünzen zur Verfügung gestellt werden. Ebenso sollten sich Wettmünzen über ein *tägliches Anmelden* in der Applikation generieren lassen. Das vorhandene Guthaben sollte dazu verwendet werden können, *Einsätze* auf Wetten von anderen Nutzern zu platzieren. Des Weiteren sollte ein Nutzer auch selbst bis zu drei Wetten pro Tag verfassen können. Bei Abschluss einer eigens erstellten Wette wird dem Verfasser ein Teil der gesetzten Wettmünzen zustehen, um die Nutzer zum Erstellen von Wetten zu motivieren.

1.3.3 Die Wette im Detail

Eine Wette in „Wollen Wir Wetten?“ zeichnen sich dadurch aus, dass sie in jedem Fall mit *Ja* oder *Nein* zu beantworten sind. Dies wird eine eindeutige Einteilung in Gewinner und Verlierer ermöglichen, was wiederrum eine eindeutige Auszahlung möglich machen wird. Wetten gehören dabei verschiedenen Typen und Kategorien an. Es existieren drei verschiedene Typen von Wetten: Standardwetten, Zufallswetten und Schnittstellenwetten.

Standardwetten sollten allerlei Kategorien umfassen, wie beispielsweise Sport, Finanzen oder auch Filme. Sie sollten dabei lediglich über eine vordefinierte Internetadresse verfügen, auf der im Zweifelsfall der Ausgang einer Wette nachgeprüft werden kann.

Zufallswetten zeichnen sich hingegen dadurch aus, dass bei Wettabschluss eine Zahl gewürfelt wird, auf die in der Wette Bezug genommen werden muss.

Wetten des Typs Schnittstellenwette zeichnen sich wiederrum dadurch aus, dass diese an eine externe Datenschnittstelle angebunden sein sollten. Beim Erstellen der Wette sollten dabei die benötigten Parameter abgefragt werden und bei Wettabschluss eine automatische Abfrage an die Schnittstelle erfolgen. Die Anzahl und Art der abgefragten Parameter sollte variabel sein, um ein einfaches Einpflegen neuer Schnittstellen zu ermöglichen.

Darüber hinaus sollte zu einer Wette eine zeitliche Gewichtung gehören, die während des Zeitraums jeden Einsatz eines Nutzers unabhängig von der Höhe des Einsatzes bewertet. Dies dient dazu, dass, beispielsweise bei Wetten über das Wetter, die Nutzer, die kurz vor Ende des Wettzeitraums wetten, keinen Vorteil gegenüber den Nutzern haben, die bereits zu Beginn des Wettzeitraums gewettet haben. Diese Gewichtung sollte auf die Kategorie angepasst sein und sich beim Erstellen von Standardwetten frei wählen lassen.

1.3.4 Auszahlung einer Wette

Da eine Wette nur mit Ja oder Nein beantwortet werden kann, werden bei Abschluss einer Wette lediglich die gesetzten Münzen der Verlierer an die Gewinner verteilt. Dem Verfasser der Wette werden dabei erstrangig 15% der aufzuteilenden Wettmünzen zugewiesen. Die weitere Auszahlung sollte dann mittels der zeitlichen Gewichtung und dem Anteil, den der Einsatz eines Nutzers an der Menge der aufzuteilenden Münzen einnimmt, erfolgen.

1.3.5 Rahmenbedingungen

Die Administration der Applikation sollte über ein passwortgeschütztes Administrationssystem erfolgen. In diesem sollte es die Möglichkeit geben, von Benutzern erstellte Wetten freizuschalten, abgeschlossene Wetten zur Auszahlung freizugeben und Informationen über die Applikation an sich erfahren. Es sollte ebenfalls die Möglichkeit gegeben sein, das System bei Bedarf mit weiteren Funktionen zu befüllen.

Der für die Nutzer zugängliche Teil sollte in Form einer nativen Smartphone-Applikation realisiert werden und für die Betriebssysteme Android und iOS lauffähig sein und später in den jeweiligen Portalen angeboten werden.

1.4 Interaktion in der Smartphone-Applikation

Dem Nutzer der Smartphone-Applikation sollte ein bestimmter Funktionsumfang zur Verfügung stehen. Dazu gehört zunächst die Registrierung innerhalb der Applikation. Diese sollte vor aller anderen Aktionen erfolgen und ist Pflicht für die Nutzung der Applikation. Befindet sich der Nutzer dann in der App, sollten ihm mehrere Möglichkeiten zur Nutzung der Applikation zur Verfügung stehen.

Die Möglichkeit, eigene Wetten zu erstellen, sollte ebenfalls leicht zu erreichbar sein. Nach der Auswahl des Typs steht dem Nutzer dann die entsprechende Eingabemaske bereit.

Eine Übersicht über aktuell laufende Wetten bietet dem Nutzer einen schnellen Einstieg in das Wettgeschehen. Dabei sollte dem Nutzer die Möglichkeit gegeben werden, die Anzeige nach verschiedenen Kriterien wie beispielsweise Beliebtheit oder Anzahl an gesetzten Wettmünzen zu sortieren. Eine Wette des Tages, welches eine besonders beliebte Wette der letzten 24 Stunden darstellt, sollte gesondert hervorgehoben werden. Zu einer Wette sollte ein Benutzer ohne große Umwege einen Einsatz platzieren können und eine Wette beobachten können.

Eine Auflistung alle laufenden Einsätze, beobachteten Wetten und selbst erstellten Wetten sollte jederzeit aufrufbar sein. Ein Einblick in die Bestenlisten sollte dem Benutzer auch auf schnelle Art und Weise zugänglich sein. Des Weiteren sollte der Nutzer in der Lage sein, einen Verlauf über seine gesetzten und erhaltenen Wettmünzen anzusehen.

Kapitel 2 - Projektstrukturanalyse

Im Folgenden wird das Projekt in der Struktur analysiert. Dabei wird das Projekt in verschiedene Komponenten unterteilt. Dies wird einen Überblick darüber verschaffen, welche Aufgaben zu absolvieren sind.

2.1 Komponentenstruktur

Anhand des Konzepts ergibt sich bereits, dass sich das Projekt generell in die sogenannten Bereiche Backend und Frontend aufteilt.

Das Frontend umfasst dabei primär die *Smartphone-Applikation*, die von den Nutzern bedient wird. Den Nutzern wird dort die Möglichkeit gegeben sein, Wetten zu erstellen, Einsätze zu platzieren, die Bestenlisten zu betrachten und das Verzeichnis laufender Wetten einzusehen. Das *Administrationssystem*, das später von den Administratoren genutzt werden wird, gehört ebenfalls zum Frontend.

Das Backend beinhaltet hingegen eine *Datenbank* zur Speicherung von beispielsweise Wetten oder Benutzerdaten, eine *Server-Applikation* zur Verwaltung der Datenbank und eine *Schnittstelle*, die eine Kommunikation zwischen Frontend und Backend ermöglichen soll. Die Schnittstelle selbst wird dabei keinen direkten Zugriff auf die Datenbank besitzen, sondern den Zugriff über die Server-Applikation erhalten.

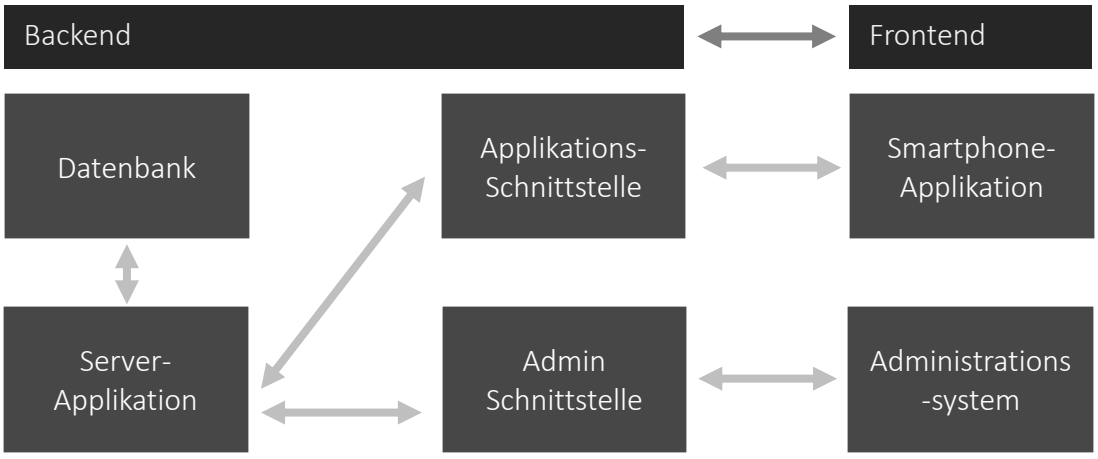


Abbildung 2 - Komponentenstruktur

2.2 Verwendete Technologien

In den nächsten Abschnitten wird es nun darum gehen, die für jede Komponente geeignetste Technologie zu ermitteln. In dem nächsten Kapitel wird dann auf die konkreten Realisierungen und Anforderungen eingegangen.

2.2.1 Datenbank & Server-Applikation

Bei der Datenbank fiel die Wahl auf das relationale Datenbankmanagementsystem *MySQL*. Zur Wahl stand auch ein weiteres relationales DBMS, PostgreSQL. PostgreSQL hat zwar in der letzten Zeit einiges an Beliebtheit gewinnen können, allerdings ist es durchaus nützlich, sich einmal explizit mit einem DBMS auseinander-zusetzen, dass bereits viele Jahre eingesetzt worden ist und daher weit verbreitet ist.

Bei der Server-Applikation findet die Skriptsprache *PHP* Verwendung. Auch hier standen weitere Sprachen wie Perl, Python oder Ruby zur Auswahl. Aus den gleichen Gründen wie bei der Wahl der Datenbank fiel hier die Wahl auf PHP. Zudem ist die Kombination von PHP und MySQL ebenfalls weit verbreitet.

2.2.2 Smartphone-Applikation

Mittlerweile gibt es recht viele Technologien, welche die Entwicklung von Applikationen für Smartphones ermöglichen. Neben den plattformspezifischen Varianten wie Java für Android oder Objective-C [1] für iOS [2] gibt es auch viele *plattformübergreifende* Methoden. Eine dieser plattformübergreifenden Methoden stellt die Entwicklung mit Adobe Air dar. Adobe Air bietet die Möglichkeit, gleichzeitig für die Betriebssysteme Android und iOS zu entwickeln. Da dies bereits einen sehr großen Teil aller in Deutschland verkauften Smartphones abdeckt, ist *Adobe Air* eine gute Wahl für die Umsetzung der Smartphone-Applikation.

[1] Objective-C bezeichnet die Programmiersprache, mit der u.a. Applikationen für iOS entwickelt werden.
[2] iOS nennt sich das Betriebssystem der Apple iPhones.

2.2.3 Schnittstellen

Die Schnittstellen stellen die Verbindung zwischen Backend und Frontend her. Jede Aktion, die ein Nutzer im Frontend durchführen kann und die sich auf die Daten in der Datenbank bezieht, erfordert eine Verbindung zur Schnittstelle. Mit Adobe Air und Javascript lässt sich sehr gut mit dem Format JSON[3] arbeiten, sodass dies eine gute Wahl für das Ausgabeformat der Schnittstelle ist.

Für die Aktionen, die im Administrationssystem durchführbar sein sollen, ist eine separate Schnittstelle von Nöten. Dies betrifft beispielsweise Aktionen wie das Freischalten beziehungsweise Abschließen von Wetten. Die Schnittstelle würde dem gleichen Verzeichnisschutz wie das Administrationssystem unterliegen und somit vor unbefugtem Zugriff schützen.

2.2.4 Administrationssystem

Bei der Administrationsoberfläche bietet es sich an, eine simple Oberfläche mit *HTML*, *CSS* und *Javascript* zu realisieren. Der HTML Code würde mit PHP ausgegeben werden, womit ein einfacher Zugriff auf die Server-Applikation stattfinden kann. Mit Javascript lässt sich zudem effizient auf die Admin-Schnittstelle zugreifen. Ein Passwortschutz würde sich einfach durch ein Verzeichnisschutz realisieren lassen, sodass die Anforderungen erfüllt sind.

2.3 Allgemeine Anforderungen

Eine allgemeine Anforderung an die Entwicklung von allen Komponenten ist, dass diese einfach erweiterbar sein müssen. Eine Smartphone-Applikation dieser Kategorie lebt dadurch, dass diese ständig mit neuen Inhalten versorgt werden kann. Bei der Entwicklung muss also Wert darauf gelegt werden, dass beispielsweise einfach neue Schnittstellenwetten eingerichtet werden können.

[3] JSON beschreibt ein uniformiertes Textformat, mit dem sich Strukturen wie Arrays und Objekten darstellen lassen. [13]

Kapitel 3 - Durchführung des Projektes

Dieses Kapitel beinhaltet konkrete Details der Realisierung von „Wollen Wir Wetten?“. Dabei wird zunächst eine Beschreibung der Arbeitsweise gegeben, woraufhin die Realisierung der einzelnen Komponenten beschrieben wird.

Dabei folgt zunächst eine Übersicht der Anforderungen, woraufhin verstärkt auf die Architektur der Komponenten und besonders wichtige Stellen in der Umsetzung eingegangen wird.

3.1 Arbeitsweise

Im Folgenden wird darauf eingegangen, welche Software für die Entwicklung benutzt wurde. Auch wird beschrieben, in welcher Art und Weise die einzelnen Komponenten zusammengefügt wurden.

3.1.1 Verwendete Software

Bei der Umsetzung der Komponenten wurden verschiedene Entwicklungs-umgebungen eingesetzt.

Für die Entwicklung von PHP, HTML, CSS und Javascript wurde *PhpStorm*[4] verwendet. Diese Entwicklungsumgebung ist für Studenten frei zugänglich und im Rahmen von Bildungszwecken zu verwenden. Diese zeichnet sich dadurch aus, dass es für alle genannten Sprachen über Code-Überprüfung, Autovervollständigung und Syntax-hervorhebung verfügt.

Zur Entwicklung der Datenbank und dem Testen von Datenbankabfragen wurde die offizielle *MySQL Workbench* verwendet. Damit ist es möglich, Datenbankstrukturen

[4] PhpStorm ist eine von JetBrains entwickelte Software, optimiert auf die Programmierung mittels PHP.

zu erzeugen und Abfragen zu analysieren. Diese ist für alle Nutzer frei erhältlich, was es zu einer guten Wahl macht.

Da das Projekt später an einen Webserver angebunden werden sollte, bedarf es einer serverähnlichen Umgebung, um die serverseitige Skriptsprache PHP ausführen zu können. Diese Umgebung wurde mit *XAMPP*[5] simuliert. Mit XAMPP fungiert der lokale Rechner als Server, was eine Ausführung von PHP ermöglicht. Dies wiederrum gestattet der Smartphone-Applikation den Zugriff auf die Applikations-Schnittstelle.

Bei der Entwicklung der eigentlichen Smartphone-Applikation kamen verschiedene Werkzeuge von Adobe[6] zum Einsatz: Flash Builder, Flash Professional und Photoshop. Frei zugänglich ist zwar nur der Flash Builder, mit dem sich die Smartphone-Applikation mittels Adobe Air realisieren lässt, doch wurden Flash Professional und Photoshop hierbei nur für den gestalterischen Teil eingesetzt. Eine rein technische Umsetzung ist mit dem Flash Builder durchaus möglich.

3.2 Realisierung

Die verschiedenen Komponenten wurden parallel entwickelt und schon sehr früh miteinander verbunden. Dadurch war es möglich, das Projekt sukzessive umzusetzen und so nach und nach die geforderten Funktionalitäten zu implementieren. Jeder Umsetzungsschritt umfasste dabei primär Erweiterungen an der Smartphone-Applikation, der Applikations-Schnittstelle und der Server-Applikation. Im Folgenden wird nun auf die konkreten Herausforderungen und Implementierungsdetails der jeweiligen Komponenten eingegangen.

3.2.1 Datenbank

Die Datenbank stellt, obwohl diese nur im Hintergrund arbeitet, einen wichtigen Teil des Projekts dar. Für die Datenbank gelten daher spezielle Anforderungen.

[5] XAMPP umfasst u.a. die Installation eines Apache-Webservers und bietet Unterstützung für PHP.
[6] Adobe Systems Incorporated ist ein US-amerikanisches Softwareunternehmen

3.2.1.1 Anforderungen

Bei einer großen Menge an Nutzern ist es wichtig, dass die Datenbank auf effiziente Weise die Abfragen abarbeiten kann. Des Weiteren ist es unabdingbar, dass stets die Integrität der Daten bewahrt wird. Das bedeutet, dass stets folgende Integritäts-bedingungen [2] erfüllt sein müssen:

- Bereichsintegrität
- Entitätsintegrität
- Referentielle Integrität
- Logische Konsistenz

Die Bereichsintegrität erwartet, dass Werte stets in vorhergesehenen Bereichen liegen. Dies sorgt dafür, dass Operationen mit diesen Werten stets durchführbar sind. Bei der Entitätsintegrität wird hingegen verlangt, dass jede Tabelle über einen eindeutigen Primärschlüssel verfügt. Dies ermöglicht einem Eintrag in einer Tabelle eine eindeutige Zuordnung. Die referentielle Integrität bezieht sich auf die Struktur von zwei voneinander abhängigen Tabellen. Bezieht sich eine Spalte einer Tabelle auf den Primärschlüssel einer anderen Tabelle, so ist der Wert in dieser Spalte entweder nicht festgelegt (*null*), oder aber entspricht einem eindeutigen Primärschlüssel der anderen Tabelle. Darüber hinaus erwartet die logische Konsistenz, dass die Daten in den Tabellen stets logischen Sinn ergeben. Da die logische Konsistenz nicht durch die korrekte Verwendung eines Datenbankmanagementsystems gewährleistet wird, ist es zunächst eine wichtige Aufgabe, eine gute Datenbankstruktur zu entwerfen. Ebenfalls wichtig für eine gut gestaltete Datenbank ist es, Richtlinien für die Benennung von Tabellen und Spalten einzuhalten.

3.2.1.2 Anforderungsanalyse

Bei dem Entwurf der Datenbankstruktur wurden zunächst die formalen Fragen geklärt. Bezüglich der Benennung der Tabellen galt es, stets den Plural von Bezeichnungen zu verwenden. Bei einer Tabelle, die sich auf eine andere Tabelle bezieht, würde der Name der referenzierten Tabelle mit einem Unterstrich konkateniert werden und dem Tabellennamen vorangestellt werden.

Um die Unterstützung von Fremdschlüsseln zu ermöglichen, würde stets die MySQL Datenbank-Engine *InnoDB* genutzt werden. Dies würde die Laufzeit bei Abfragen über mehrere Tabellen hinweg beschleunigen und ebenfalls die referentielle Integrität wahren. [3] Bezüglich der Entitätsintegrität würde jede Tabelle über eine gesonderte Spalte verfügen, die nur als Primärschlüssel dient und somit eine eindeutige Zuordnung ermöglicht. Die logische Konsistenz hingegen ist Teil der Datenbankstruktur selbst und der Art und Weise, wie Datensätze in die Datenbank gelangen.

3.2.1.3 Datenbankstruktur

Die Struktur der Datenbank wurde anhand des Konzepts vorab entworfen. Anhand folgender Abbildung wird nun die Struktur der Datenbank erläutert. Dabei wird der Fokus auf die Beziehung der Tabellen untereinander eingegangen. Die einzelnen Spalten der jeweiligen Tabellen stehen hierbei nicht im Vordergrund, sind allerdings in einem EER-Diagramm in Anhang 1 enthalten.

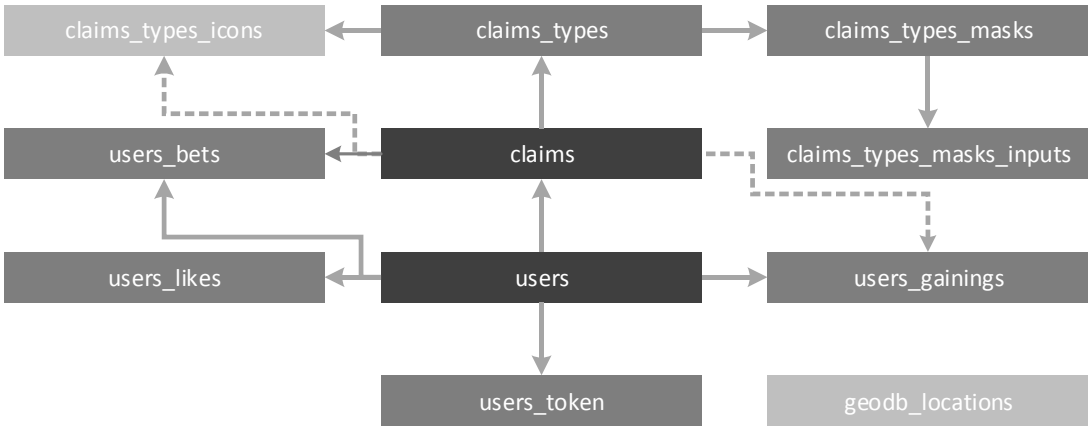


Abbildung 3 - Datenbankstruktur

Ein Nutzer kann sich in der Applikation registrieren. Eine Tabelle „**users**“, die für die Speicherung von Benutzerdaten vorgesehen ist, ist daher unabdingbar.

Ebenfalls essentiell ist die Tabelle *claims* für die Speicherung der Wetten. Wie dem Konzept entnommen werden kann, gehören die Wetten verschiedenen Typen an. Sie besitzen also eine Referenz in Form eines Fremdschlüssels zu einem konkreten Eintrag in der Tabelle *claims_types*. Pro Typ existieren verschiedene Eingabemasken, sodass ein Typ eine Referenz zu einer konkreten Eingabemaske in der Tabelle

claims_types_masks besitzt. Eine Eingabemaske ist zu konfigurieren, indem verschiedene Eingabefelder mit Referenz zur Eingabemaske in der Tabelle *claims_types_masks_inputs* gespeichert werden. Falls nun also neue Typen hinzugefügt werden, kann einfach auf bereits vorhandene Eingabemasken zurückgegriffen werden, oder eine neue konfiguriert werden.

Platziert ein Nutzer einen Einsatz auf eine Wette, wird dieser Einsatz in Form eines Eintrages in der Tabelle *users_bets* gespeichert. Dieser Eintrag besitzt Referenzen zu dem Benutzer, der den Eintrag platziert hat und der Wette, auf die sich der Einsatz bezieht.

Jeglicher Erhalt von Münzen wird in der Tabelle *users_gainings* festgehalten. Dazu gehören der erstmalige Erhalt von Münzen bei der Registrierung, der tägliche Bonus für das Anmelden, Auszahlungen für gewonnene Wetten und der Erhalt durch den Anteil, dem der Verfasser einer Wette zusteht.

3.2.1.4 Die Wette des Tages

Um die Funktionsweise der beschrieben Datenbankstruktur zu veranschaulichen, wird im Folgenden die Datenbankabfrage nach der Wette des Tages beschrieben. Diese zeichnet sich dadurch aus, dass sie die in den letzten 24 Stunden erfolgreichste Wette bestimmt. Dabei werden die Wetten nach drei gleichgewichteten Kriterien sortiert und abschließend die Wette mit der höchsten Bewertung als Wette des Tages ermittelt. Die Kriterien umfassen hierbei jeweils die Eigenschaften „Anzahl gesetzte Wettmünzen“, „Anzahl Teilnehmer“ und „Anzahl Beobachtungen“ in den letzten 24 Stunden im Verhältnis zu der Wette mit den größten Anzahlen. Diese Art von Abfrage ist in erster Linie dadurch möglich, dass beim Entwurf der Datenbank Wert darauf gelegt wurde, keine berechneten Werte in der Datenbank zu speichern, sondern jeder Aktion ein Datensatz zugewiesen wird.

Um diese Abfrage nun in eine MySQL Abfrage umzuformen, bedarf es einiger Unterabfragen, die letztlich zu einer Gesamtabfrage kombiniert werden. Die kombinierte MySQL Abfrage ist Anhang 2 zu entnehmen.

3.2.2 Server-Applikation

Die Server-Applikation ist verantwortlich für den Zugriff auf die Datenbank und die Bereitstellung von Methoden zum Abruf dieser Daten, welche letztendlich von den Schnittstellen aufgerufen werden.

3.2.2.1 Slim Framework

Eine Herausforderung, die bei der Entwicklung einer Server-Applikation entsteht, ist die Abbildung von aufrufbaren Pfaden / Adressen zu Aktionen, die in der Applikation aufgerufen werden sollen. Da die Server-Applikation primär für die Bereitstellung von Schnittstellenfunktionen entwickelt wird, ist eine einfache Abbildung von Pfaden zu definierbarem Verhalten von großem Nutzen. Wird also beispielsweise „/api /login/register“ aufgerufen, sollte es einfach möglich sein, bestimmte Ausgaben für diesen Pfad festzulegen und bestimmte Prozeduren aufzurufen. In PHP lässt sich zwar der aufgerufene Pfad abfragen, doch bietet in diesem Fall das Slim Framework die Möglichkeit, auf einfache Art ein Mapping zu definieren. Folgendes Beispiel veranschaulicht die Abbildung des oben genannten Pfades auf ausführbaren Code:

```
$app = new \Slim\Slim();

$app->group('/api',  function () use ($app) {
      $app->group('/user', 'initUser', function () use ($app) {
            $app->map('/checkdaily', function () use ($app) {
                              // do something…
            });
      });
});
```

Abbildung 4 - Pfadzuweisung

Mit Hilfe dieses Frameworks ist es also möglich, eine strukturierte Schnittstelle zu entwickeln, welche ebenfalls einfach zu erweitern ist. Darüber hinaus ist es möglich, für verschiedene Gruppen sogenannte *Middlewares* festzulegen. Diese ermöglichen den Aufruf von Funktionen, bevor der eigentliche Code innerhalb des Pfades aufgerufen wird. Die Funktion „initUser“ aus Abbildung 4 ist eine solche Middleware.

3.2.2.2 Architektur

Der Kern der Server-Applikation setzt sich zusammen aus der Verbindung zu der Datenbank und dem Zugriff darauf. Gefordert ist also eine Umsetzung, die ein einfaches Arbeiten mit der Datenbank ermöglicht.

Es wurde daher die Idee der objektrelationalen Abbildung verfolgt und im Ansatz übernommen. Jeder Tabelle wurde eine eigene Klasse in PHP zugewiesen. Diese Klassen erben von einer Klasse *MainModel***,** welche die Basisfunktionalitäten zum Speichern, Aktualisieren und Löschen von Einträgen in der Datenbank ermöglicht. Darüber hinaus verfügt die Klasse MainModel über die Funktion, SQL Abfragen auszuführen und die abgefragten Daten direkt den Instanzvariablen zuzuweisen.

Jede Instanz einer dieser Klassen entspricht einem Datensatz in der Tabelle. Die eigentliche Instanziierung von diesen Klassen findet nur innerhalb der jeweiligen Klasse oder über klassenspezifische statische Methoden statt. Einen öffentlich zugänglichen Konstruktor bieten die Klassen nicht. Dies soll sicherstellen, dass die Klassen korrekt verwendet werden. Um die Arbeit zu vereinheitlichen, besitzt jede Tabelle die Spalte „**id**“ als Primärschlüssel. Relationale Abhängigkeiten werden mit Hilfe dieses Primär-schlüssels über den Zugriff auf die klassenspezifischen statischen Methoden möglich gemacht. Zwar werden relationale Abbildungen so nicht, wie eigentlich beim Konzept der objektrelationalen Abbildung vorgesehen, realisiert, allerdings besteht nun voller Zugriff auf die MySQL Abfragen, was ein Optimieren dieser Abfragen ermöglicht und somit für jeden Anwendungsfall eine optimierte Abfrage bereitgestellt werden kann.

Die konkreten MySQL Abfragen an die Datenbank sind dabei an die Model Klassen gebunden und nur über die Klassen-Methoden zu erreichen. Dieses Vorgehen würde einen eventuellen Umstieg auf eine andere Datenbank erleichtern, da alle Abfragen zentral in einem Paket zu finden sind.

Ein Beispiel für die Abfrage eines bestimmten Nutzers stellt folgender Aufruf dar:

```
$user = User::FindUserByAliasAndPassword($password, $alias);

class User extends MainModel {

 public static function FindUserByAliasAndPassword($password, $alias) {

  $user = new User();

  $SQL = "
  SELECT
    *
  FROM
    `" . static::TABLE . "`
  WHERE
    `alias` = '" . $alias . "'
  AND
    `password` = SHA2((CONCAT(`salt`,'" . $password . "')), 512)
  LIMIT 1";

  return $user->mapSQLToVariables($SQL);

 }
}
```

Abbildung 5 - Beispielaufruf PHP

In dem Beispiel wird mittels Passwort und Benutzername ein Objekt des Typs „User“ zurückgegeben, welches genau einem Datensatz in der Tabelle „users“ entspricht. Darüber hinaus wird in dem Beispiel bei der Passwortabfrage eine SHA2 [7] Verschlüsselung durchgeführt. Dies ist unter Anderem Gegenstand einer der nächsten Abschnitte, in denen detailliert einige Kernfunktionen der Server-Applikation beschrieben werden.

3.2.2.3 Registrierung & Passwortschutz

Um die sensiblen Daten der Nutzer so gut wie möglich abzusichern, wird das Passwort eines Nutzers bei Registrierung nicht im Klartext in der Datenbank gespeichert. Der Ansatz, der in diesem Projekt verfolgt wurde, sieht die Verschlüsselung des Passwortes mit Hilfe des SHA2 Algorithmus vor. Der generierte Hash Wert des SHA2 Algorithmus lässt sich nicht umkehren, sodass sich dieser sehr gut für die Speicherung in der Datenbank eignet und dann bei einer Authentifizierung zu Vergleichszwecken

[7] Der SHA2 Algorithmus gehört der Klasse der Secure Hash Algorithmen an und ermöglicht die Erzeugung von größeren Hash Werten.

herangezogen werden kann. Ein Passwort wird also in erster Linie dann akzeptiert, wenn die generierten Hash Werte übereinstimmen.

Um dieses Konzept noch weiter zu verfeinern, wurde eine zusätzliche Maßnahme getroffen, die den SHA2 Lookup-Tables[8] entgegen wirken soll. Bei Registrierung eines Nutzers wird zusätzlich ein sogenannter Salt gespeichert. Dies ist eine zufällig generierte Zeichenkette, welche zur Generierung des SHA2 Hash Wertes dient. Dazu wird dieser wie in Abbildung 4 dem eigentlichen Passwort vorangestellt und sorgt dafür, dass ein gewähltes Passwort nicht eindeutig auf einen Hash Wert abgebildet werden kann.

3.2.2.4 Auszahlung einer Wette

Die Auszahlung einer Wette findet statt, nachdem der Ausgang der Wette ermittelt wurde, was je nach Typ der Wette automatisch oder manuell geschieht. Der eigentliche Vorgang des Auszahlens ist dabei in drei Schritte unterteilt:

Im *ersten* Schritt werden die einer Wette zugehörigen Einsätze ermittelt. Für jeden Einsatz wird nun anhand der festgelegten zeitlichen Gewichtsfunktion die zeitliche Gewichtung berechnet. Die Einsätze, die zu den Gewinnern gehören, werden für die weitere Berechnung herangezogen. Die Höhe des Einsatzes wird dabei mit der zeitlichen Gewichtung multipliziert, um den zeitlich gewichteten Wert des Einsatzes zu bestimmen. Im *zweiten* Schritt werden nun die gewichteten Einsätze der Gewinner miteinander verglichen, um so für jeden Gewinner den Anteil zu bestimmen, den sie von gesetzten Wettmünzen der Verlierer erhalten. Dabei wird zunächst die Höhe aller gewichteten Einsätze aufsummiert und dann der Anteil eines gewichteten Einsatzes an dieser Summe bestimmt. Im *letzten* Schritt erfolgt die eigentliche Auszahlung an die Gewinner und an den Verfasser der Wette (vgl. Anhang 3).

[8] Eine Lookup-Table ist eine Abbildung von Zeichenketten zu den entsprechenden Hash Werten.

3.2.2.5 Wetterwetten

Ein besonderer Typ von Wetten ist durch die Wetterwetten gegeben. Diese stellen die erste Implementation der Schnittstellenwetten dar und verfügen über eine benutzerdefinierte Eingabemaske, um alle notwendigen Parameter für einen finalen Schnittstellenaufruf abzufragen. Bei den Wetterwetten handelt es sich dabei um das Wetterkriterium, den Tag des Wettvorgangs und eine frei zu wählende Postleitzahl. Die Postleitzahl ist eine eindeutige Identifikation des Ortes und bietet sich daher gut an, um eine Wette zu definieren.

Nach Ende des Wettzeitraums einer Wette findet ein Schnittstellenaufruf an die Wunderground API statt. Die API von Wunderground bietet die Möglichkeit, Wetterdaten von bereits vergangenen Tagen abzufragen. Die Wetterdaten umfassen dabei eine tägliche Zusammenfassung und eine Übersicht aller Messdaten, die an dem Tag verzeichnet wurden. In der Regel sind dabei drei Messwerte pro Stunde verfügbar. Dies ermöglicht Entscheidungswetten über Regenfall, Gewitter, Hagel, Nebel und Operatorwetten über die Temperatur, wie beispielsweise die Durchschnittstemperatur. Je nach Wetterkriterium werden die Wetterdaten des Wetttages bei Wettabschluss verarbeitet und die Wette automatisch evaluiert.

Bei dem Erstellen einer Wetterwette werden mit Hilfe von Satzschablonen ein Titel und eine Beschreibung generiert, die von dem gewählten Kriterium, dem gewählten Ort und dem gewählten Tag abhängt. Der Name des Ortes wird dabei mittels der Postleitzahl der Datenbank entnommen. Dabei wurde auf die öffentlich zugängliche Datenbank von OpenGeoDB zurückgegriffen [4], welche eine Identifikation eines Ortes mittels der Postleitzahl ermöglicht. Ein Beispiel für eine solche Schablone ist durch folgenden Code-Abschnitt gegeben:

```
$descriptions = array(
      ...
      "RAIN" => array(
        "no" => "Es wird in @city@ am @date@ kein Tropfen Regen vom
        Himmel fallen.",
        "yes" => "Es wird in @city@ am @date@ mindestens einmal ein
        Tropfen Regen vom Himmel fallen."
      )
)
```

Abbildung 6 - Satzbausteine Wetter

3.2.3 Applikations-Schnittstelle

Da mit dem Slim Framework bereits eine Abbildung von Pfaden zu ausführbarem Code existiert, ist der nächste Schritt bei der Entwicklung einer Schnittstelle die Authentifizierung eines Nutzers, um so Aktionen bereitzustellen, welche sich unmittelbar auf einen Nutzer beziehen. Dazu gehören beispielsweise das Platzieren von Einsätzen oder das Erstellen von eigenen Wetten.

3.2.3.1 Authentifizierung

Die Authentifizierung eines Nutzers basiert auf der Idee, auf eine frei aufrufbare Schnittstellenanfrage hin einen *Zugangsschlüssel* zu generieren, welcher eine eindeutige Identifikation des Nutzers ermöglicht und für weitere Schnittstellenaufrufe verwendet werden kann. Dies hat den Vorteil, dass das Nutzerpasswort nur einmalig bei dem Einholen eines solchen Zugangsschlüssels verwendet werden muss. Der Nutzer wird damit vor solchen Angriffen geschützt, die darauf abzielen, unbefugt Anfragen an die Schnittstelle aufzuzeichnen und auszuwerten.

Soll nun also ein solcher Zugangsschlüssel generiert werden, werden das übermittelte Passwort und der übermittelte Benutzername auf Übereinstimmung mit einem Datensatz in der Datenbank hin überprüft. Falls ein Benutzer mit den übermittelten Daten existiert, wird ein neuer Zugangsschlüssel generiert und in der Datenbank hinterlegt. Der Zugangsschlüssel ist nur für einen definierten Zeitraum gültig und verfällt bei erneutem Login, da maximal ein Zugangsschlüssel gleichzeitig aktiv sein kann. Darüber hinaus ist ohne Zugangsschlüssel nur ein beschränkter Zugriff auf die Schnittstelle möglich, da keine Identifikation des Nutzers stattfinden kann.

3.2.3.2 Ausgabe

Die Server-Applikation bietet Zugriff auf die Daten, die sich in der Datenbank befinden. Dabei werden alle verfügbaren Daten übergeben. Um die Menge an Informationen und Ladezeiten auf ein Minimum zu reduzieren, müssen diese Daten gefiltert werden. So ist es beispielsweise in der Übersicht über die laufenden Wetten nicht nötig, Informationen über die dazugehörige Eingabemaske zu erhalten.

Die Filterung dieser Daten geschieht dabei durch sogenannte Views. Eine View ist in der Lage, eine Menge an Daten entgegenzunehmen und eine für den jeweiligen Anwendungsfall optimierte Ausgabe zu erzeugen. Da es sich bei dem Ausgabeformat nach Kapitel 2 um JSON handelt, bietet es sich an, die Ausgabe in Form eines Arrays zu erzeugen:

```
$claims = ClaimController::getInstance()->getClaimsDefault();

$app->json(array(
        'success' => true,
        'data' =>
        \core\view\ClaimOverviewItem::RenderMultiple($claims)
));
```

Abbildung 7 – Zugriff auf eine View

3.2.4 Smartphone-Applikation

Die Entwicklung der Smartphone-Applikation gliedert sich in mehrere Schritte.

Zunächst würde ein Navigationskonzept derart konzipiert werden müssen, dass sich Nutzer einfach innerhalb der Applikation zurechtfinden können, ohne dabei auf wichtige Funktionen verzichten zu müssen.

Daraufhin würden die technischen Voraussetzungen analysiert werden, um ein Gerüst für eine Smartphone-Applikation, die sich auf Android und iOS exportieren lässt, zu schaffen. Die nächsten Schritte umfassen schließlich die konkrete technische Umsetzung des geforderten Konzeptes, einschließlich Anbindung an die Applikations-Schnittstelle.

3.2.4.1 Navigationskonzept

Der Hauptfokus des Navigationskonzeptes liegt auf einer einfachen Bedienung. Der Nutzer soll sich direkt zu Beginn in der Applikation zurechtfinden, ohne dabei auf wichtige Funktionen verzichten zu müssen. Er soll weiterhin in der Lage sein, sich schnell und ohne große Umwege zwischen diesen Bereichen zu bewegen und schnell erkennen können, in welchem Bereich er sich gerade befindet.

In Anbetracht der geforderten Funktionen erschein eine grobe Aufteilung in vier Bereiche als sinnvoll, welche weiterhin durch unterschiedliche Farben gekennzeichnet und durch eine primäre Navigation am unteren Ende der Applikation zu erreichen sind.

Abbildung 8 - Farbgebung Bereiche

Jedem Bereich wird dabei eine Seite zugewiesen, die durch Verwendung der Navigation im unteren Bereich aufgerufen wird. Die Bereiche umfassen des Weiteren jeweils verschiedene Unterseiten zur Anzeige der benötigten Inhalte. Dabei soll es stets möglich sein, zu zuvor besuchten Seiten zurückzukehren.

3.2.4.2 Gestaltung & Präsentation

In diesem Abschnitt wird die finale Fassung von „Wollen Wir Wetten?“ vorgestellt und anhand Ausschnitten aus der Smartphone-Applikationen das Navigations-konzept erläutert. Bei der Gestaltung wurde hauptsächlich Wert auf Übersicht und eine ansprechende Oberfläche gelegt, was den spielerischen Charakter der Applikation unterstreichen soll.

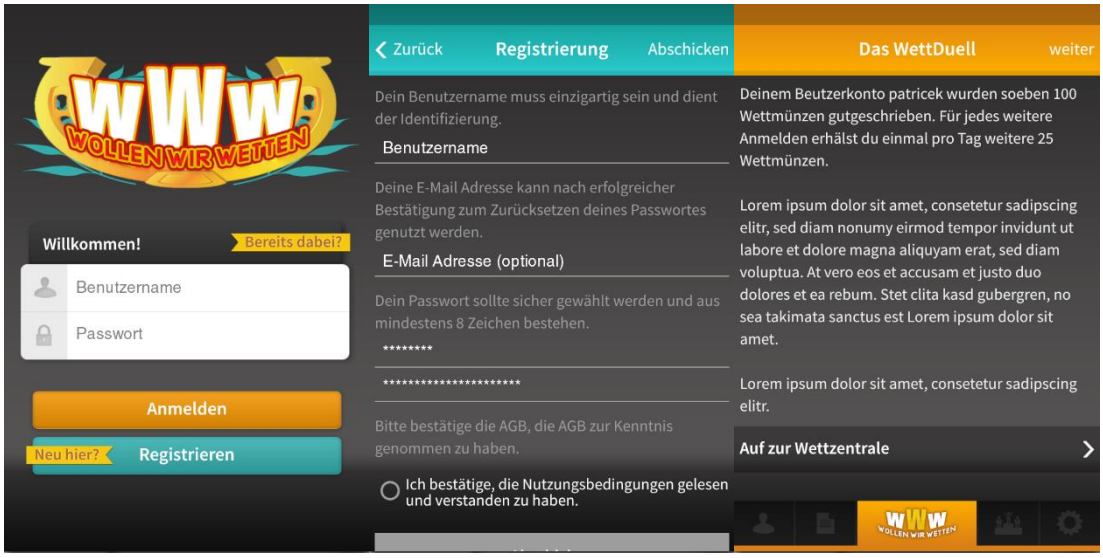


Abbildung 9 - Registrierung

Bei erstmaligen Aufruf wird dem Nutzer der Registrierungsbildschirm präsentiert. Nach erfolgreicher Registrierung wird dem Nutzer nach Anmeldung zunächst ein

Intro angezeigt, in dem er alles Wichtige über „Wollen Wir Wetten?“ erfährt. Von dort aus gelangt er auch direkt zur Wettzentrale, dem zentralen Anlaufpunkt in „Wollen Wir Wetten?“.

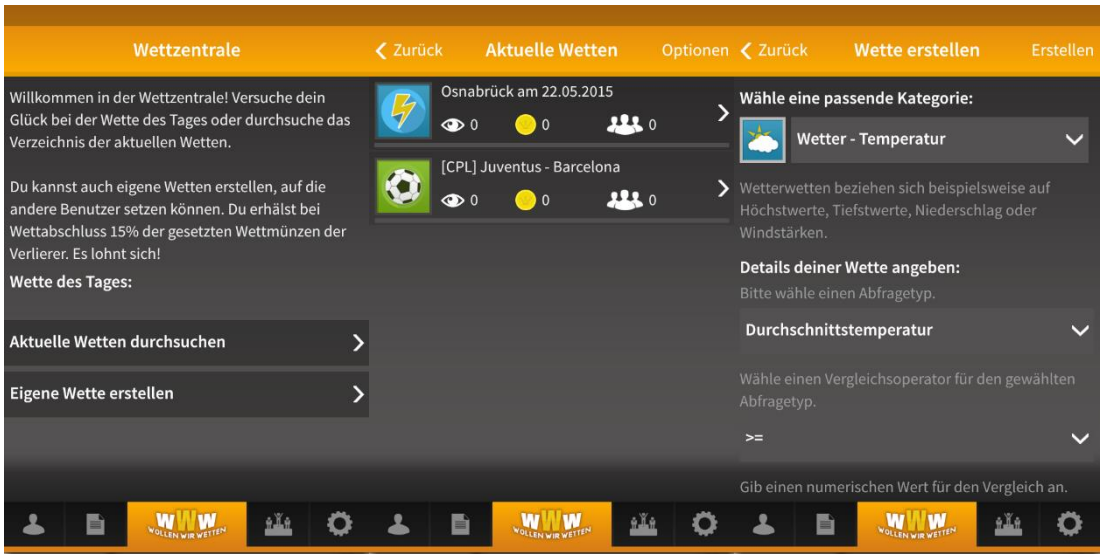


Abbildung 10 - Wettzentrale

In der Wettzentrale steht dem Benutzer eine Übersicht über alle laufenden Wetten zur Verfügung. Des Weiteren gelangt er von der Wettzentrale ohne große Umwege direkt zu der Seite, auf der Wetten erstellt werden können. Eine Übersicht über alle erstellten Wetten ist unter anderem in der Benutzerzentrale zu finden:

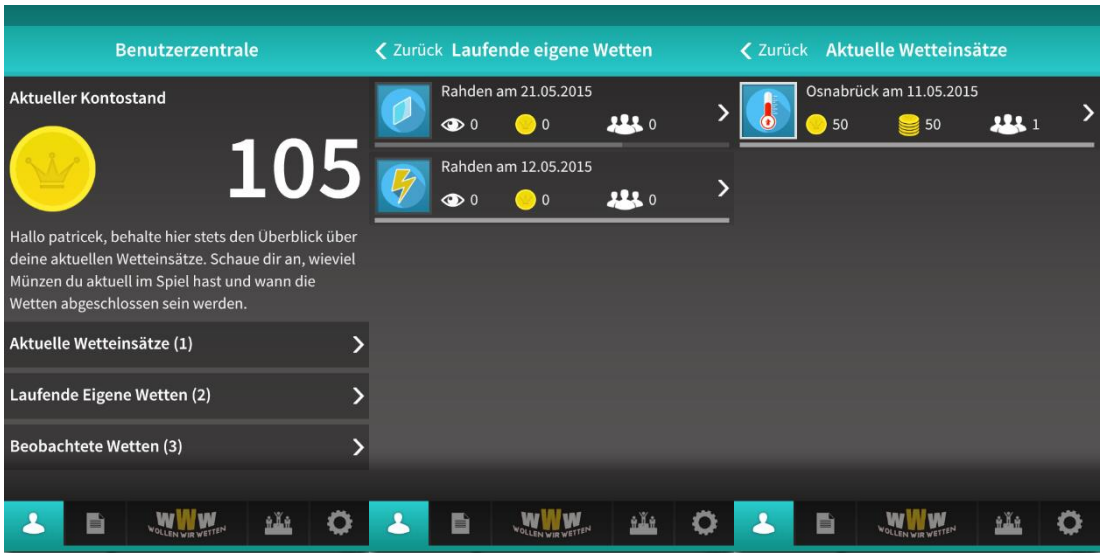


Abbildung 11 - Benutzerzentrale

Die Benutzerzentrale liefert dem Nutzer des Weiteren eine Übersicht über seinen Kontostand und bietet zusätzlich eine Übersicht über aktuelle Wetteinsätze, laufende

eigene Wetten und beobachtete Wetten. Dies soll eine einfache Organisation der eigenen Wetten und Einsätze ermöglichen.

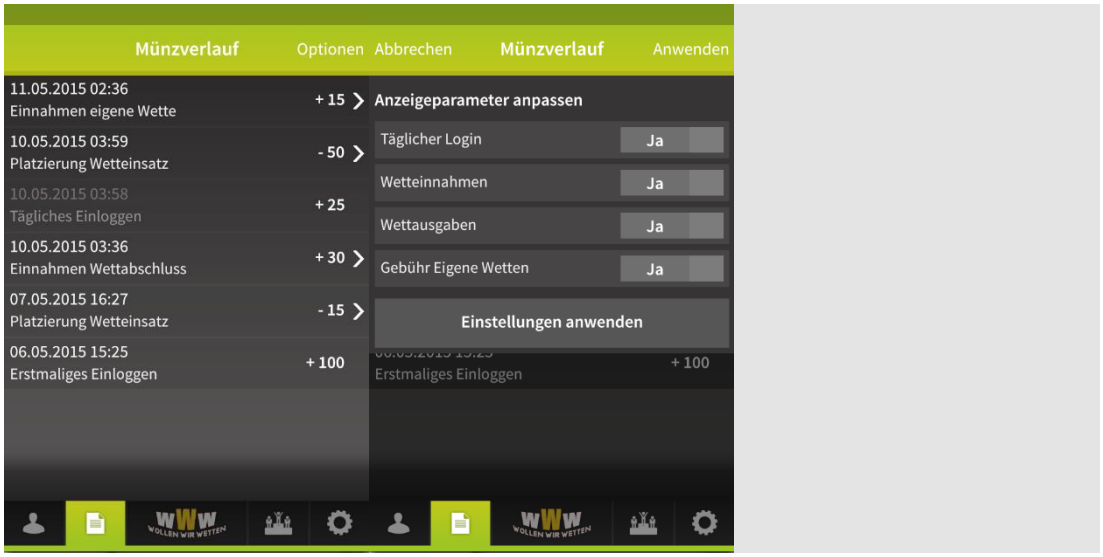


Abbildung 12 - Münzverlauf

Der Münzverlauf stellt dem Benutzer eine Auflistung aller vergangen Einkünfte und Ausgaben zur Verfügung. Die Anzeige kann dabei nach verschiedenen Kriterien gefiltert werden, um stets einen optimalen Überblick zu erhalten. Der Münzverlauf stellt darüber hinaus eine Erklärung zu der eigenen Position in der Bestenliste dar.

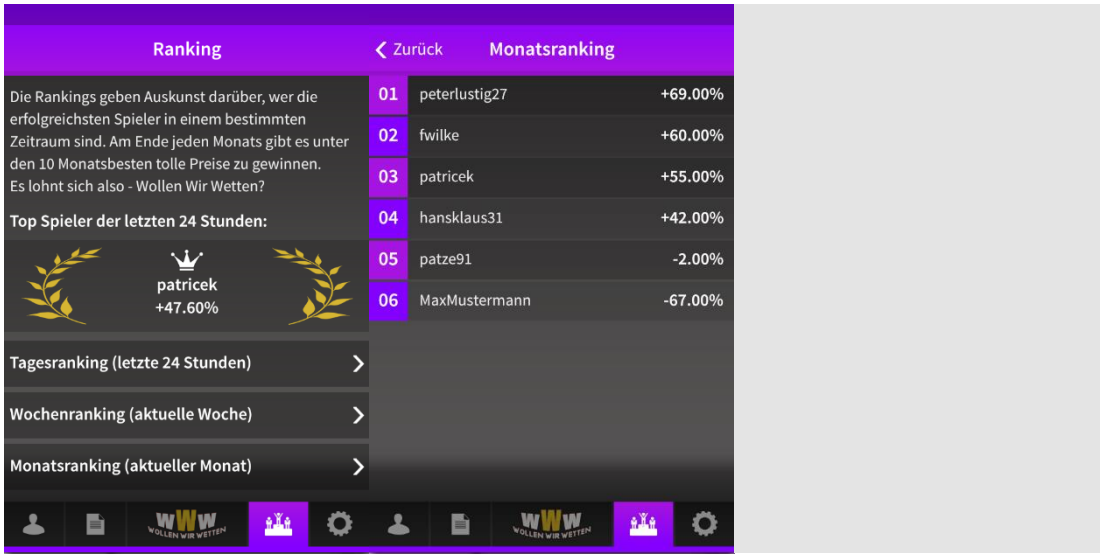


Abbildung 13 - Bestenlisten

Die Bestenlisten bieten den primären Anreiz, sein Guthaben zu verößern. Daher sollten auch diese schnell zu erreichen sein und in einer ansprechenden Weise präsentiert werden. Dem Top Spieler der letzten 24 Stunden wird zudem ein

besonders gestalteter Platz in der Applikation zugewiesen, sodass der Nutzer Motivation fasst, diesen Platz selbst einzunehmen.

3.2.4.3 Adobe Air, Starling & Feathers

Adobe Air stellt eine plattformunabhängige Laufzeitumgebung dar, welche von Adobe zur Verfügung gestellt wird. Seit der Version AIR3 wird auch die GPU beschleunigte Anzeige ermöglicht, was diese Laufzeitumgebung zu einer guten Wahl bei mobilen Applikationen macht. Applikationen in Adobe Air werden mittels der objektorientierten Programmiersprache Actionscript 3 entwickelt.

Möchte man in Adobe Air auf das volle Potential GPU[9] beschleunigter Anzeige zugreifen, bedarf es einiger Hilfsmittel. Dies liegt daran, dass die Inhalte in vergangenen Versionen von Adobe Air stets über die CPU gerendert worden sind und dieses Verhalten in aktuellen Version von Adobe Air standardmäßig noch präsent ist. Um also auf die GPU Beschleunigung zurückzugreifen, bedarf es eines Frameworks, das es ermöglicht, Texturen in die GPU zu laden und diese anzuzeigen. Ein Framework, welches sich explizit damit beschäftigt, ist Starling [5]. Starling ermöglicht die Anzeige von 2D Inhalten und verfolgt dabei den Ansatz, eine Arbeitsweise zu ermöglichen, welche ähnlich zu der aus vergangenen Versionen von Adobe Air ist. So gibt es für alle Anzeigeklassen, die Adobe Air zur Verfügung stellt, ein Äquivalent in Starling. Intern organisiert Starling die Darstellung von 2D Inhalten in einer projektiven 3D Umgebung.

Ein weiteres Framework, das bei der Realisierung der Smartphone-Applikation zum Einsatz kommt, ist Feathers [6]. Feathers basiert auf Starling und bietet eine Sammlung an Komponenten für ein mobiles Benutzerinterface. Dazu gehören beispielsweise Textfelder zur Eingabe, Schaltflächen und Listen zur Anzeige von Datenmengen. Ein Aspekt bei der Entwicklung der Smartphone-Applikation besteht darin, die Komponenten anzupassen und benutzerdefinierte Komponenten hinzuzufügen. Um für spezielle Skins den Zugriff auf benutzerdefinierte Texturen zu erhalten, ist die Verwendung eines sogenannten TextureAtlas notwendig.

[9] Die GPU ist ein auf die Berechnung von Grafiken spezialisierter Prozessor.

3.2.4.4 TextureAtlas

Ein TextureAtlas [7] dient in erster Linie dazu, Texturen in Form von Grafiken in der Applikation darstellen zu können. Er fasst dabei mehrere Texturen in einer Grafik zusammen. Jede Textur verfügt über einen Schlüssel, über den mittels Koordinaten und Größendaten auf die Textur zugegriffen werden kann. Diese Zuordnungen werden in einer XML Datei gespeichert. Ein TextureAtlas besteht also aus einer XML Datei und einer transparenten PNG Datei, welche in Abbildung 14 zu sehen ist:

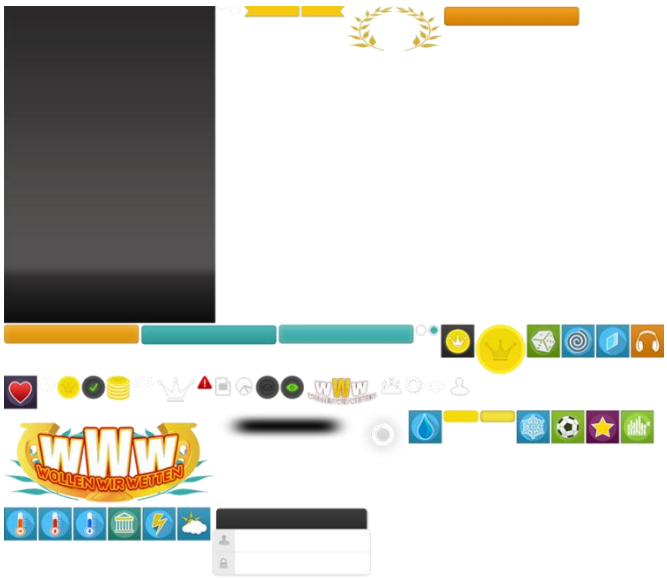


Abbildung 14 – TextureAtlas

Eine Zuordnung der XML Datei schaut dabei folgendermaßen aus:

```
<TextureAtlas imagePath="wollenwirwetten.png">
      …
      <SubTexture name="icon_coin.png" x="161" y="1120" width="70"
      height="70" pivotX="0" pivotY="0"/>
      …
</TextureAtlas>
```

Abbildung 15 - TextureAtlas XML

Der technische Nutzen eines TextureAtlas ist groß. Um nämlich Texturen mittels der GPU anzeigen zu können, müssen diese vorher in die GPU geladen werden. Starling verfügt über eine sogenannte Stapel-Renderfunktion. Diese rendert Objekte in einem Durchlauf, bis sich die Textur ändert. Ein TextureAtlas sorgt nun dafür, dass es nur eine Textur ist, die in die GPU geladen und gerendert wird. Einer Textur eines Anzeigeobjektes wird letztlich ein Ausschnitt in der TextureAtlas Grafik zugewiesen.

3.2.4.5 Anzeigekomponenten & Listenansicht

Um die Umsetzung des Layouts zu vereinfachen, wurden vorab benutzerdefinierte Komponenten erstellt. Dazu gehören hauptsächlich Schaltflächen, Eingabefelder, Listendarstellungen oder auch Optionsmenüs, wie in Abbildung 12 zu erkennen ist.

Die benutzerdefinierten Komponenten würden dabei mit dem eingesetzten Feathers Framework verträglich sein müssen, um eine korrekte Ausführung zu gewährleisten. Eine Feathers Komponente zeichnet sich durch einen besonderen Lebenszyklus aus, welcher hauptsächlich die Schritte der Initialisierung, der Validierung, des Renderns und des Löschens umfasst [8].

Abbildung 16 - Listenansicht

Eine wichtige Komponente in „Wollen Wir Wetten?“ stellt die Listenansicht einer Wette dar. Dieser sollten direkt so viele Informationen wie möglich entnommen werden können. Daher verfügt die Listenansicht über ein typspezifisches Icon, den Titel der Wette, die Anzahl an Beobachtungen, die gesetzten Wettmünzen und die registrierten Teilnehmer. Darüber hinaus ist mittels eines eingebauten Fortschrittsbalkens über verschiedene Farbstufen direkt zu erkennen, in welchem Stadium sich die Wette befindet. Wurde eine Wette vom Nutzer erstellt, so muss diese erst noch freigeschaltet werden und der Fortschrittsbalken wird ausgeblendet. Dies lässt sich ebenfalls über einen angezeigten Text erkennen. Ein aktiver Wettzeitraum wird durch einen leicht ausgegrauten Fortschrittsbalken symbolisiert, welcher bei Annahmeschluss der jeweiligen Wette gefüllt sein wird. Ist ein Wettvorgang aktiv, beispielsweise der laufende Tag bei einer Wetterwette, so ist der Fortschrittsbalken stärker sichtbar und ist komplett gefüllt, wenn der Vorgang abgeschlossen ist.

Diese Komponente kommt an vielen Stellen zum Einsatz und ist durch die Implementierung des oben beschriebenen Lebenszyklus einfach in eine Feathers Liste zu integrieren.

3.2.4.6 Der Lokale Speicher & SQLite

Wie bereits aus dem Abschnitt 3.2.3.1 bekannt ist, erwartet die Applikations-Schnittstelle, dass ein Zugriffsschlüssel an diese übergeben wird. Damit dieser Zugriffsschlüssel nicht mit jedem Start der Applikation erstellt werden muss, bedarf es einer Möglichkeit, diesen lokal zu speichern. Adobe Air bietet dabei Unterstützung für eine SQLite Datenbank [9]. Die Speicherung von den Daten geschieht dabei in einer einzigen Tabelle, welche über eine Schlüsselspalte und eine Wertspalte verfügt. Bei Start der Applikation werden initial alle gespeicherten Werte und Schlüssel abgefragt und in einem Array gespeichert. Der Zugriff auf die gespeicherten Daten ist damit jederzeit über den jeweiligen Schlüssel möglich.

3.2.4.7 Schnittstellenaufrufe & Typisierung

Schnittstellenaufrufe werden mittels der Klasse ApiRequest realisiert. Diese verfügt über die Funktionalität, Anfragen an die Applikations-Schnittstelle durchzuführen. Darüber hinaus verfügt sie über Funktionen, die Ausgabe der Schnittstelle zu verarbeiten. Es ist vorgesehen, dass jeder Schnittstellenaufruf von der Klasse ApiRequest erbt und die Funktion zum Verarbeiten der Ausgabe überschreibt.

Um die Arbeit mit der Applikations-Schnittstelle zu erleichtern, wurde für jede Schnittstelle die Ausgabe in typisierte Datenobjekte umgewandelt. Über diese Objekte ist dann der Zugriff auf Variablen eines bestimmten Typs möglich. Des Weiteren lässt sich als zusätzliche Hilfe für die Entwicklung pro Schnittstellenaufruf eine Menge an erforderlichen Parametern festlegen, sodass Fehler in der Benutzung leicht nachverfolgt werden können.

Ein Beispiel für einen konkreten Schnittstellenaufruf ist der Aufruf zum Einholen eines Zugangsschlüssels.

```
ApiRequest.createRequest(AuthorizeRequest.AUTHORIZE).performRequest(
{
        alias: (e.data['inputName']).text,
        password: (e.data['inputPassword']).text
},
onWelcomeLoginResponse
);
```

Abbildung 17 – Schnittstellenaufruf

Die Typisierung erfolgt daraufhin durch Speicherung der Ausgabe in Instanzvariablen der Klasse AuthorizeRequest:

```
override public function processRequestData(data:Object):void {
        _alias = data["alias"];
        _accessToken = data["accessToken"];
}
```

Abbildung 18 - Verarbeitung Ausgabe

3.2.4.8 Seitenverwaltung

Aus dem Navigationskonzept geht hervor, dass es verschiedene Seiten und weiterführende Unterseiten geben soll. Es musste also eine Verwaltung solchermaßen entwickelt werden, dass eine Koordination dieser ermöglicht wird.

Zunächst wurden daher Navigation und Header umgesetzt. Der Header besteht aus einem Titel und zwei verfügbaren Schaltflächen, welche sich jederzeit ändern lassen. So ist es möglich, das Verhalten in Abbildung 12 zu realisieren. Die Schaltflächen des Headers ändern sich auf dieser Seite bei Anzeige der Filterkriterien. Die Navigation ist eine simple Komponente, die über mehrere Schaltflächen verfügt, welche jedoch je nach Anwendungsfall zu deaktivieren sind.

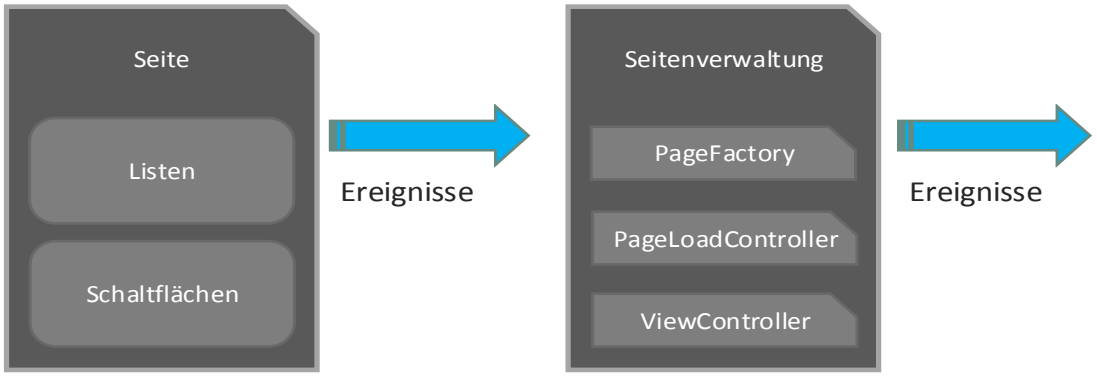


Abbildung 19 - Seite zu Seitenverwaltung

Daraufhin wurde die eigentliche Klasse für die Seite entwickelt. Eine Seite verfügt dabei hauptsächlich über Funktionen zum Ein- und Ausblenden, zur Positionierung der enthaltenen Elemente wie Schaltflächen, Listen und Grafiken und zum Weiterleiten von Ereignissen, beispielsweise Klicks auf Schaltflächen, an die zugehörige Seitenverwaltung. Der eigentliche Kontrollfluss der Applikation wird jedoch nicht von der Seitenverwaltung kontrolliert, sodass die Ereignisse weitergeleitet werden und von einer zentralen Klasse verarbeitet werden, die primär für den Kontrollfluss verantwortlich ist.

Die folgende Abbildung zeigt den Aufbau der Seitenverwaltung, deren Funktionsweise im nächsten Abschnitt erläutert wird:

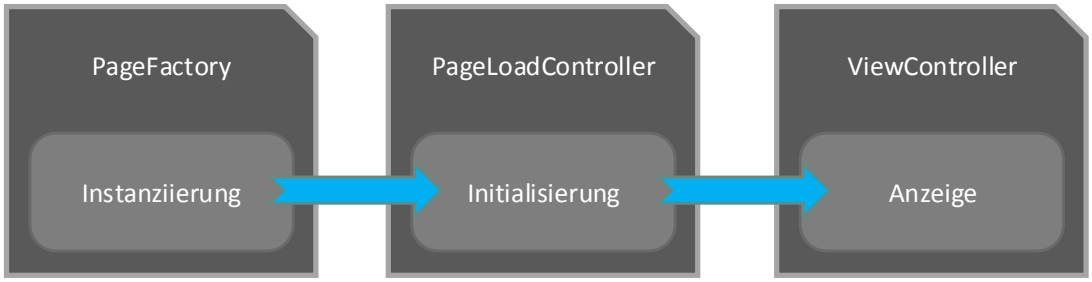


Abbildung 20 - Seitenverwaltung

Die Seitenverwaltung setzt sich aus mehreren Klassen zusammen und ist verantwortlich für das Instanziieren, Initialisieren und Anzeigen von Seiten. Seiten werden dabei mittels eines Schlüssels identifiziert. Die Instanziierung einer Seite geschieht anhand des Schlüssels durch eine Fabrikklasse [10], welche einzig zum Erstellen von Seiten-Klassen dient. Die Initialisierung einer Seite stellt einen Prozess dar, in dem, falls erforderlich, ebenfalls Schnittstellenaufrufe stattfinden. Erforderliche Schnittstellenaufrufe werden bei der Instanziierung einer Seite festgelegt, sodass eine Seite erst nach abgeschlossenem Schnittstellenaufruf angezeigt wird. So wird sichergestellt, dass eine Seite stets über alle benötigten Informationen verfügt.

[10] Eine Klasse, welche nach dem Fabrikmuster umgesetzt wurde. [14, p. 109]

3.2.4.9 Handhabung von Ereignissen

Wie Abbildung 19 zu entnehmen ist, werden die ausgelösten Ereignisse von der Seitenverwaltung weitergeleitet. Diese werden in der zentralen Kontrollfluss-Klasse MainController verarbeitet, womit diese Klasse für die Validierung von Eingabedaten, Anzeige von Fehler- und Erfolgsmeldungen und die Initialisierung von Schnittstellen-aufrufen verantwortlich ist. Dies ermöglicht dem Entwickler, stets den Überblick darüber zu behalten, wo eine Aktion in der Applikation auftritt und wie darauf reagiert wird.

3.2.4.10 Verschiedene Endgeräte

Verschiedene Versionen der Smartphone-Applikation lassen sich einfach durch angepasste Startklassen bereitstellen. So wird beispielsweise bei iOS Geräten der Header erweitert, um so Platz für die transparente Statusleiste einzuräumen.

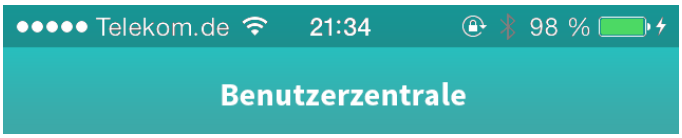


Abbildung 21 - Statusleiste im Header

Die Bereitstellung der Applikation für Apple iOS Geräte erfordert die Erstellung eines sogenannten Bereitstellungsprofils. Dieses lässt sich nur mit einer aktiven Entwicklerlizenz von Apple erstellen, welche für 99$ / Jahr zu erwerben ist [10]. In dem Profil lassen sich bestimmte MAC-Adressen[11] hinterlegen, wodurch ein Testen der Applikation auf diese Geräte beschränkt ist.

Für Android Geräte ist es hingegen nur eine Option in den Einstellungen, die aktiviert werden muss, um ein Testen und Debuggen auf diesen Geräten zu ermöglichen. Dabei wird vorab Adobe Air auf dem Gerät installiert, damit die Applikation zum Testen ausgeführt werden kann. In der finalen Version, die später auch im Google Play Store angeboten wird, ist die verwendete Laufzeitumgebung von Adobe Air mit enthalten [11].

[11] Die Mac-Adresse ist eine Adresse, welche die Identifikation eines Netzwerkadapters ermöglicht.

3.2.5 Administrationssystem

Die Freischaltung von neuen Wetten sowie der Abschluss von Wetten ist die Aufgabe eines Administrators. Das Administrationssystem stellt dem Administrator dabei die Funktionalitäten zur Verfügung, die erforderlich sind, um diese Aktion durchzuführen. Wichtig bei der Realisierung des Administrationssystems ist, dass alle Aktionen schnell durchführbar sind. Dies würde eine schnelle Bearbeitung der Aufgaben ermöglichen. Ein schlechter Ansatz ist dabei die Idee, nach jeder Aktion die Seite im Browser neuzuladen. Dies würde viel Zeit kosten. Eine bessere Idee ist es an dieser Stelle, alle Schnittstellenaufrufe mit Javascript auszuführen, was allerdings in der Entwicklung deutlich aufwendiger ist, da auf die Resultate von Aktionen reagiert werden muss.

Um also die Möglichkeit von Schnittstellenaufrufen innerhalb der Administration zu ermöglichen, wurde eine eigene Administrations-Schnittstelle entwickelt, welche gesonderte Funktionalitäten zum Pflegen der Applikation bereitstellt. Schnittstelle und Administrationssystem unterliegen dabei einem Verzeichnisschutz, um vor unbefugtem Zugriff zu schützen.

3.2.5.1 Freischalten einer neuen Wette

Für das Freischalten einer neuen Wette steht dem Administrator eine nach der Aktualität sortierten Übersicht aller neuen Wetten zur Verfügung. Eine Spalte in der Tabelle der Übersicht enthält Schaltflächen für den Aufruf von festgelegten Aktionen. In diesem Fall kann ein Administrator die Wetten freischalten oder löschen.

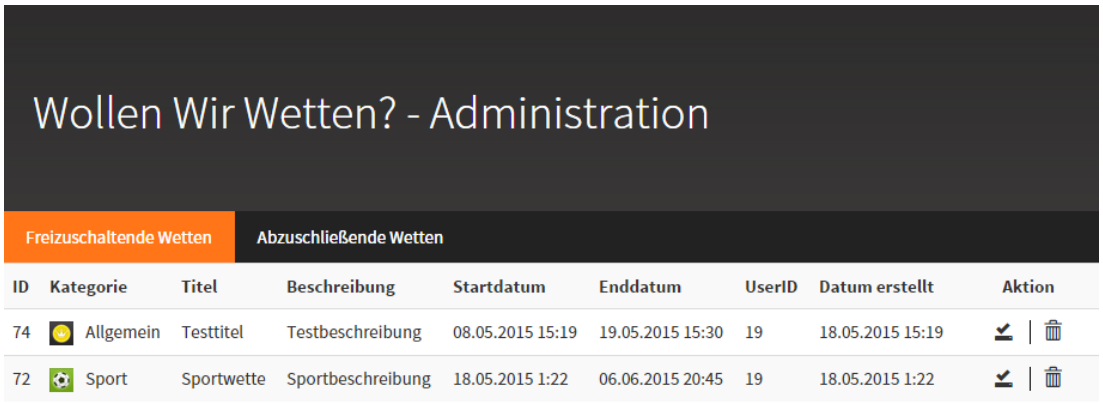


Abbildung 22 - Administrationssystem I

Hat der Administrator sich dafür entschieden, eine Wette freizugeben, wird nun ein Schnittstellenaufruf ausgeführt und auf die Antwort im Administrationssystem reagiert:

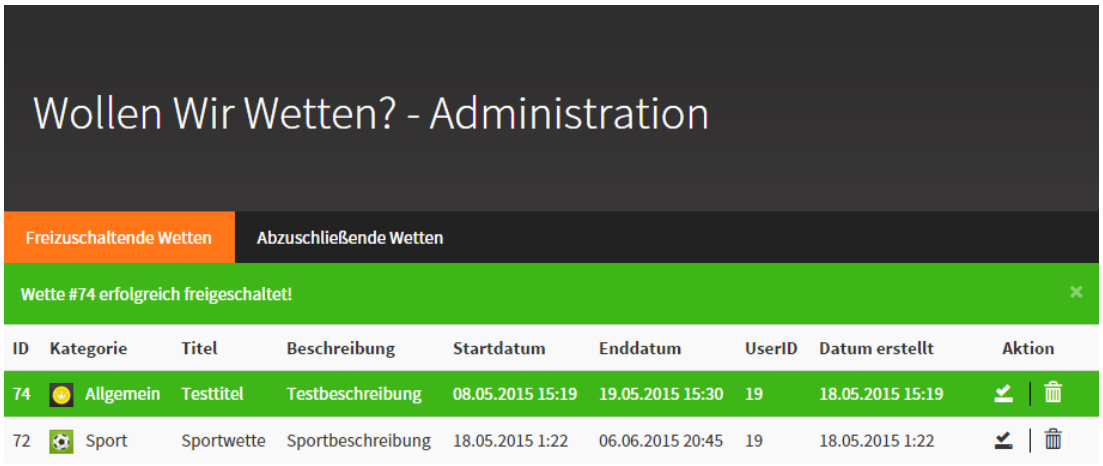


Abbildung 23 - Administrationssystem II

Der Administrator kann nun direkt mit dem Bearbeiten weiterer Wetten fortfahren, ohne auf einen erneuten Seitenaufbau warten zu müssen.

3.2.5.2 Abschluss einer Wette

Der Abschluss einer Wette funktioniert ähnlich wie das Freischalten einer neuen Wette. Der Prozess des Abschlusses ist allerdings in mehrere Schritte unterteilt:

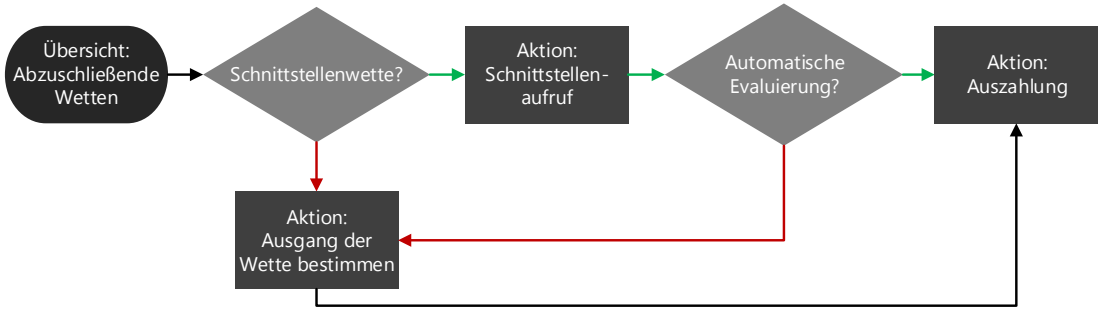


Abbildung 24 - Flussdiagram Abschluss Wette

Zunächst wird der Ausgang der Wette bestimmt. Das Verhalten im Administrationssystem hängt dabei, wie in Abbildung 24 zu erkennen, von dem Typ der Wette ab. Es gibt Wetten, bei denen manuell über den Ausgang entschieden werden muss und Wetten, die den Ausgang mittels eines Schnittstellenaufrufes bestimmen, zum Beispiel die Wetten über das Wetter. Allerdings gibt es bei den Schnittstellenwetten auch den Fall, dass der Ausgang noch manuell bestimmt werden

muss. Dies trifft zum Beispiel bei einer Zufallswette auf. Für die Schnittstellenwetten muss der Administrator zunächst den Aufruf an die eigentliche Schnittstelle in Auftrag geben. Die Antwort wird daraufhin in der Übersicht sichtbar. Wurde der Ausgang der Wette schließlich bestimmt, ist eine Auszahlung möglich. Anhang 4 zeigt die Umsetzung dieses Verhaltens in Javascript.

Kapitel 4 - Schlussfolgerungen

Zusammenfassend lässt sich bezüglich der Realisierung des Projekts zunächst sagen, dass alle geforderten Funktionalitäten umgesetzt werden konnten. In diesem Kapitel wird daher nach einer Zusammenfassung der Arbeit kritisch über die Umsetzung der einzelnen Komponenten reflektiert. Wichtige Aspekte stellen dabei die Erweiterbarkeit und die Einhaltung von Programmierrichtlinien dar.

4.1 Zusammenfassung

Zu Anfang dieser Arbeit wurde das Konzept von „Wollen Wir Wetten?“ vorgestellt. In diesem Konzept wird speziell auf die Rahmenbedingungen, den Erhalt von Münzen, die Wette im Detail und die Auszahlung einer Wette eingegangen.

Das Konzept wurde daraufhin analysiert, um eine Übersicht über die zu realisierenden Komponenten zu erhalten. Diese Analyse ergab, dass eine Datenbank, eine Server-Applikation, zwei Schnittstellen und eine Smartphone-Applikation zu realisieren sind. Des Weiteren wurde der Analyse entnommen, welche Technologien für diese Komponenten zu verwenden sind und welche Alternativen existieren.

Die Durchführung des Projekts war Gegenstand des nächsten Kapitels. Hier wurde die Realisierung der verschiedenen Komponenten erläutert und anhand einiger Schlüsselstellen wurden die Kernfunktionen der jeweiligen Komponente verdeutlicht. Dabei wurde stets auf die Anforderungen eingegangen und der Nutzen der jeweiligen Schlüsselstelle in der dazugehörigen Komponente verdeutlicht.

4.2 Reflexion

4.2.1.1 Datenbank

Bei der Realisierung der Datenbank wurde Wert auf die Einhaltung der Integritätsbedingungen gelegt. Ein abschließender Blick auf den Entwurf der Datenbank zeigt, dass diese größtenteils eingehalten worden sind. Eine kritische Stelle ist jedoch die Speicherung von Zeitangaben in der Datenbank. Da jede Zeitangabe in Form eines *UNIX-Zeitstempels*[12] gespeichert wird, ist die logische Konsistenz der Datenbank nicht sichergestellt. So lassen sich bei Fehlern im Speichern der Daten Zeitangaben eintragen, welche keinen logischen Sinn ergeben. Dazu gehören beispielsweise die verschiedenen Zeitangaben, die einer Wette zugehörig sind. Eine Wette, deren Wettstart zeitlich nach dem Wettende liegt, ist in der Datenbank durchaus zu speichern, ergibt aber keinen logischen Sinn.

Bezüglich logischer Richtlinien könnte die Datenbank noch über Spaltenkommentare zur Erläuterung dieser Richtlinien verfügen. Dies würde die Übersicht für andere Entwickler verbessern.

4.2.1.2 Server-Applikation

Der in der Server-Applikation verfolgte Ansatz der objektrelationalen Abbildung ermöglichte eine strukturierte Entwicklung. Zu optimieren ist die Lösung noch durch eine Möglichkeit, Objekte, die in einem relationalen Zusammenhang zueinander stehen, direkt in Listen einer Datensatzinstanz zu speichern. Ein Beispiel für solch ein Verhalten wäre folgender Aufruf, um die Übersicht über alle laufenden Einsätze zu erhalten:

```
$betsRunning = $user->fetchBetsRunning();
```

Abbildung 25 - Fiktiver Aufruf, laufende Einsätze

Die aktuelle Lösung sieht vor, mittels des Primärschlüssels eines Nutzers eine der verfügbaren statischen Methoden aufzurufen, welche dann die laufenden Wetten

[12] Ein UNIX Zeitstempel beschreibt die Anzahl vergangener Sekunden seit dem 1. Januar 1970 00:00 UTC.

ermittelt. Die aktuelle Lösung hat zwar den Vorteil, dass alle MySQL Abfragen thematisch gruppiert sind, doch sind relationale Abhängigkeiten nur schwer zu erkennen.

Es lässt sich daher generell sagen, dass die Realisierung der Server-Applikation durch die Filterung der Ausgaben, Gruppierung der MySQL Abfragen und die Verwendung des Slim Frameworks sehr gut strukturiert ist. Funktionalitäten sind einfach hinzuzufügen, sodass beispielsweise mittels variabler Eingabemasken neue Schnittstellenwetten einfach zu implementieren sind.

4.2.1.3 Schnittstelle & Administrationssystem

Bei der Entwicklung der Schnittstelle wurde Wert auf eine korrekte Fehlerbehandlung, eine auf das Minimum reduzierte Ausgabe und die Erweiterbarkeit gelegt. Dies wurde konsequent eingehalten, sodass Fehlverhalten stets abgefangen wird. Auch werden Eingaben in der Applikation stets serverseitig validiert, sodass eine Korrektheit der Daten gewährleistet ist.

Das Administrationssystem, welches ebenfalls auf die Schnittstelle zugreift, ist durch die Verwendung des Slim Frameworks einfach erweiterbar. Neue Funktionalitäten sind einfach hinzuzufügen. Bezüglich der Benutzung eines Administrators lässt sich hauptsächlich sagen, dass erforderliche Aktionen schnell und einfach durchführbar sind.

4.2.1.4 Smartphone-Applikation

Die Realisierung Smartphone-Applikation nahm den größten Anteil an Zeit ein. Es ist sehr umständlich, gestaltete Seiten für verschiedene Seiten umzusetzen. Jegliche Positionierung von Elementen musste variabel sein, um verschiedene Displaygrößen und Punktdichten [13] zu unterstützen. Was die Entwicklung einzelner Seiten möglicherweise vereinfacht hätte, wäre die Entwicklung einer benutzerdefinierten Auszeichnungssprache, in der Idee ähnlich zu HTML, gewesen, mit der sich verschiedene Komponenten beschreiben lassen würden. Für den Fall, dass in Zukunft

[13] Die Punktdichte (oft: DPI) eines Displays gibt einen numerischen Wert für dessen Pixeldichte an.

noch viele weitere Seiten entwickelt werden müssen, wäre dies eine Aufgabe mit hoher Priorität. Die Idee wurde bei der Erzeugung variabler Eingabemasken bereits verfolgt.

Eine weitere, möglicherweise zu überdenkende Lösung stellt das Konzept der Handhabung von Ereignissen dar. Durch eine zentrale Verwaltung, in der alle Events verarbeitet werden, entsteht bei wachsendem Funktionsumfang eine stets größere Klasse. Es besteht an dieser Stelle die Möglichkeit, eigene Unterverwaltungen bereitzustellen, die für verschiedene Aufgabenbereiche verantwortlich sind. Ein anderer Lösungsansatz sähe vor, die Verarbeitung von Ereignissen direkt in die Seiten selbst zu verpacken. Dies hätte allerdings den Nachteil, bei großem Funktionsumfang schnell den Überblick über diesen zu verlieren.

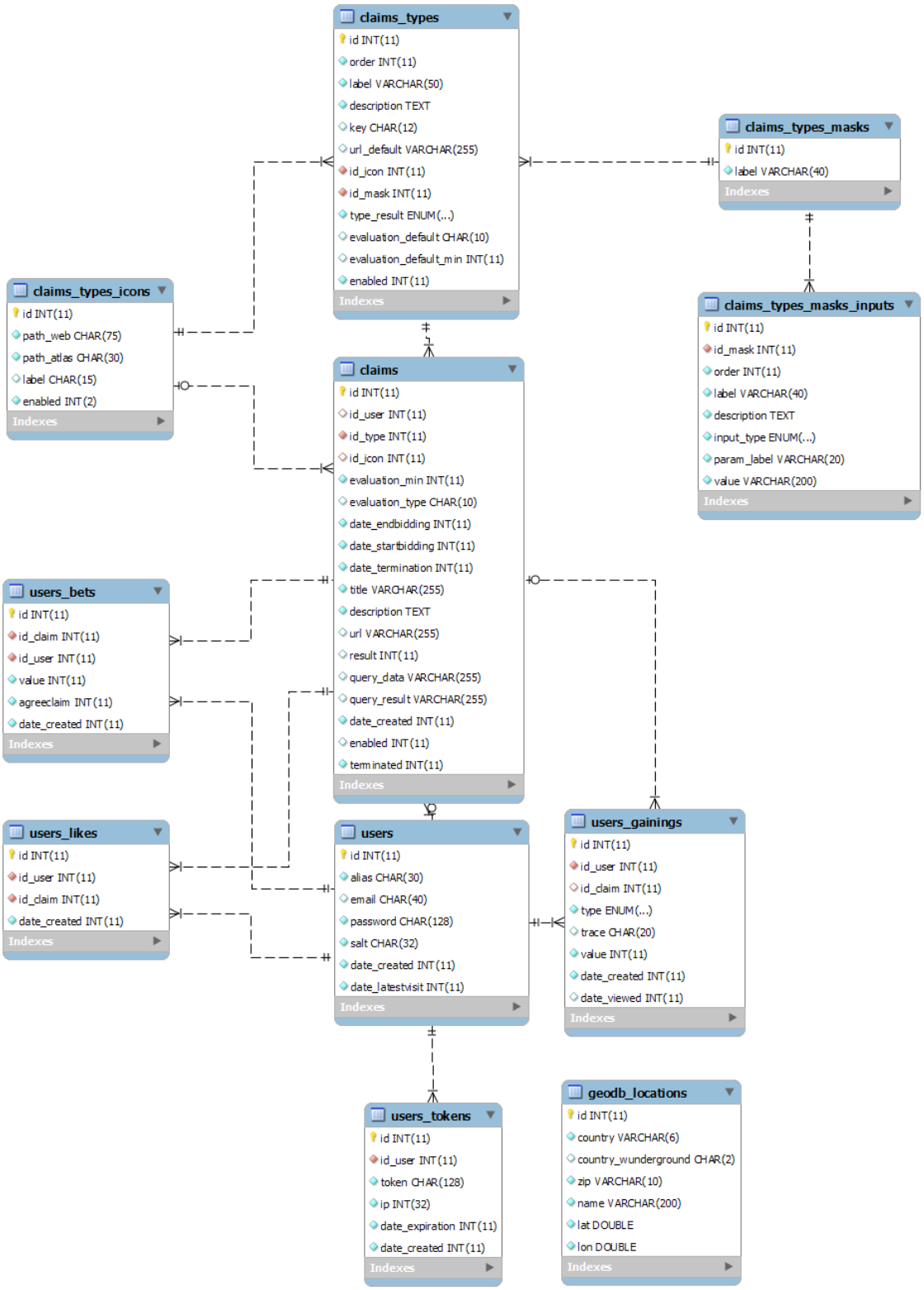
Die Performance der Smartphone-Applikation ist zufriedenstellend. So läuft „Wollen Wir Wetten?“ ohne Probleme auf einem Samsung Galaxy der ersten Generation. Des Weiteren wurde durch die Verwendung des Feathers Frameworks sichergestellt, dass Eingabefelder korrekt auf allen Geräten funktionieren.

4.3 Ausblick

Die Realisierung des Konzepts ist beendet. In der Zukunft wird es nun darum gehen, dieses Konzept zu erweitern und mit weiteren Funktionen zu befüllen. Eine Basis dafür wurde geschaffen. Der unmittelbar nächste Schritt ist die Markteinführung von „Wollen Wir Wetten?“. Dazu muss die Smartphone-Applikation für den Google Play Store und den Apple App Store bereitgestellt werden und die Datenbank auf einem Webserver installiert werden. Die Datenbank würde dann einer stetigen Überwachung unterliegen, um gegebenenfalls die Performance einiger MySQL Abfragen zu optimieren. Um dies zu erreichen, wäre es möglich, Ergebnisse aufwändiger Abfragen zu speichern und beispielsweise automatisch alle zehn Minuten zu aktualisieren. Dies umfasst dabei speziell die Abfrage der Wette des Tages und die der Bestenlisten.

Anhänge

Anhang 1 - EER Diagramm des Datenbankkonzepts



Anhang 2 - MySQL Abfrage für die Wette des Tages

```
SELECT DISTINCT
 `data`.*
FROM
 (SELECT

  `claims`.`id`,
  `claims`.`title`,
  `claims`.`description`,
  `claims`.`date_created`,
  `claims`.`id_user`,
  `claims`.`date_endbidding`,
  `claims`.`date_startbidding`,
  `claims`.`date_termination`,
  `claims`.`evaluation_min`,
  `claims`.`evaluation_type`,
  `claims`.`id_type`,
  `claims`.`url`,
  `icons`.`path_web` as `icon_path_web`,
  `icons`.`path_atlas` as `icon_path_atlas`,

  IFNULL(`bets_user`.`value`, 0) as `callinguser_betvalue`,
  IFNULL(`likes_total`.`likes`,0) as `likes_total`,
  IFNULL(`bets_total`.`people`,0) as `people_total`,
  IFNULL(`bets_total`.`coins`,0) as `coins_total`,

  IFNULL(`likes_pastday`.`likes`,0) as `likes_pastday`,
  IFNULL(`bets_pastday`.`people`,0) as `people_pastday`,
  IFNULL(`bets_pastday`.`coins`,0) as `coins_pastday`,

  IFNULL((SELECT
    SUM(`bets`.`value`) as `coins_max`
   FROM
    `users_bets` as `bets`
   WHERE
    `bets`.`date_created` > 1431905113
   GROUP BY
    `bets`.`id_claim`
   ORDER BY
    `coins_max` DESC
   LIMIT 1),0) as `coins_pastday_max`,

  IFNULL((SELECT
    COUNT(`likes`.`id`) as `likes_max`
   FROM
    `users_likes` as `likes`
   WHERE
    `likes`.`date_created` > 1431905113
   GROUP BY
    `likes`.`id_claim`
   ORDER BY
    `likes_max` DESC
   LIMIT 1),0) as `likes_pastday_max`,

  IFNULL((SELECT
    COUNT(`bets`.`id_user`) as `people_max`
   FROM
    `users_bets` as `bets`
   WHERE
    `bets`.`date_created` > 1431905113
   GROUP BY
    `bets`.`id_claim`
   ORDER BY
    `people_max` DESC
```

```
  LIMIT 1),0) as `people_pastday_max`
FROM
 `claims` as `claims`

 LEFT JOIN
  (SELECT
   id_claim,
   SUM(value) as `coins`,
   COUNT(id_user) as `people`
  FROM
   `users_bets`
  GROUP BY
   `id_claim`
  ) as `bets_total`
  ON `claims`.`id` = `bets_total`.`id_claim`

 LEFT JOIN
  (SELECT
   id_claim,
   COUNT(id) as `likes`
  FROM
   `users_likes`
  GROUP BY
   `id_claim`
  ) as `likes_total`
  ON `claims`.`id` = `likes_total`.`id_claim`

 LEFT JOIN
  (SELECT
   id_claim,
   SUM(value) as `coins`,
   COUNT(id_user) as `people`
  FROM
   `users_bets`
  WHERE
   `date_created` > 1431905113
  GROUP BY
   `id_claim`
  ) as `bets_pastday`
  ON `claims`.`id` = `bets_total`.`id_claim`

 LEFT JOIN
  (SELECT
   id_claim,
   COUNT(id) as `likes`
  FROM
   `users_likes`
  WHERE
   `date_created` > 1431905113
  GROUP BY
   `id_claim`
  ) as `likes_pastday`
  ON `claims`.`id` = `likes_total`.`id_claim`

 LEFT JOIN `claims_types` as `types`
  ON `claims`.`id_type` = `types`.`id`

 LEFT JOIN `claims_types_icons` as `icons`
  ON `icons`.`id` = IFNULL(`claims`.`id_icon`, `types`.`id_icon`)

 LEFT JOIN `users_bets` as `bets_user`
  on (
   `claims`.`id` = `bets_user`.`id_claim`
   AND
   `bets_user`.`id_user` = 9
  )
WHERE
 `claims`.`date_endbidding` > 1431991513
 AND
 `claims`.`enabled` = 1
```

```
  AND
   `claims`.`date_startbidding` < 1431991513
 GROUP BY
   `claims`.`id`
 ) as `data`
ORDER BY
 (
   (`data`.`coins_pastday`/ `data`.`coins_pastday_max`) +
   (`data`.`likes_pastday` / `data`.`likes_pastday_max`) +
   (`data`.`people_pastday` / `data`.`people_pastday_max`)
 ) DESC
LIMIT 1
```

Anhang 3 - Auszahlung einer Wette in PHP

```
public function terminateClaim($claimid) {

  // claimdetails abfragen
  $claim = Claim::FindClaimByID($claimid);

  if ($claim->tbl_result === NULL) {
   throw new Exception(ErrorCodes::CLAIM_TERMINATE_RESULTISNULL);
  }

  if ($claim->tbl_terminated == 1) {
   throw new Exception(ErrorCodes::CLAIM_TERMINATE_ALREADYTERMINATED);
  }

  // alle gewinner bets abfragen, die zu claimid passen und zwischenspeichern
von betTimeFactor, ausgehend von bets.date_created
  $bets = Bet::FindBetsByClaim($claimid);

  $winnerBetsMapping = array();
  $winnerCoinsSum = 0;
  $loserCoinsSum = 0;

  foreach ($bets as $bet) {

   if ($bet->tbl_agreeclaim === $claim->tbl_result) {

    $winnerCoinsSum += $bet->tbl_value;

    $winnerBetsMapping[$bet->tbl_id_user] = array(
     "betTimeFactor" => $claim->getBetTimeFactor($bet->tbl_date_created),
     "betValue" => $bet->tbl_value,
     "betId" => $bet->getId()
    );

   } else {

    $loserCoinsSum += $bet->tbl_value;

   }

  }
  // alle gewichteten summen aufaddieren und nun die anteile pro bet berechnen
  $weightedWinnerCoinsSum = 0;

  foreach ($winnerBetsMapping as &$winnerBetsEntry) {
   $winnerBetsEntry["weightedBetValue"] = $winnerBetsEntry["betTimeFactor"]
* $winnerBetsEntry["betValue"];
   $weightedWinnerCoinsSum += $winnerBetsEntry["weightedBetValue"];
  }

  WettDuell::getInstance()->getPDO()->beginTransaction();

  try {

   // verliermünzen zwischen gewinnern aufteilen
   foreach ($winnerBetsMapping as $key => &$winnerBetsEntry) {

    $winnerBetsEntry["payoutFinal"]                                    =
floor(($winnerBetsEntry["weightedBetValue"]  /  $weightedWinnerCoinsSum)  *
($loserCoinsSum * \Config::get('app/payout/percentageSplit')));

    Gaining::NewGainingWithParams(
     array(
      "userId" => $key,
      "type" => Gaining::TYPE_GAINING_PAYOUT,
      "value"          =>          $winnerBetsEntry["betValue"]          +
$winnerBetsEntry["payoutFinal"],
```

```
      "date_created" => time(),
      "claimId" => $claim->getId()
     )
    )->save();

   }

   // verfasser der wette erhätlt eigenen anteil
   Gaining::NewGainingWithParams(
    array(
     "userId" => $claim->tbl_id_user,
     "type" => Gaining::TYPE_GAINING_PAYOUT_FEE,
     "value" => $loserCoinsSum * \Config::get('app/payout/percentageOwner'),
     "date_created" => time(),
     "claimId" => $claim->getId()
    )
   )->save();

   $claim->tbl_terminated = 1;
   $claim->save();

   WettDuell::getInstance()->getPDO()->commit();

  } catch (Exception $e) {

   WettDuell::getInstance()->getPDO()->rollBack();

   return false;

  }

  return true;

 }
```

Anhang 4 - Administrativer Schnittstellenaufruf

```
function performClaimQuery(claimid) {

 $.ajax({ url: '/admin/api/claims/querydata',
  data: {
   claimid:claimid
  },
  type: 'post',
  success: function (response) {

   if (response["success"]) {

    var tr = $('tr[data-claimid="'+response["data"]["claimid"]+'"]');

    tr.find('.td-query-result').html(response["data"]["queryResult"]);
    tr.find('.td-actions .claim-no').removeClass('active');
    tr.find('.td-actions .claim-yes').removeClass('active');
    tr.find('.td-actions .claim-query').addClass('disabled');

    switch(response["data"]["result"]) {

     // automatisches Ergebnis = Nein
     case 0:
     case '0':
      tr.addClass('success');
      tr.find('.td-actions .claim-no').addClass('active');
      tr.find('.td-actions .claim-terminate').removeClass('disabled');
      showSuccessMessage("Wette #" + response["data"]["claimid"] + "
erfolgreich beantwortet! Eine Auszahlung ist jetzt möglich!");
      break;

     // automatisches Ergebnis = Ja
     case 1:
     case '1':
      tr.addClass('success');
      tr.find('.td-actions .claim-yes').addClass('active');
      tr.find('.td-actions .claim-terminate').removeClass('disabled');
      showSuccessMessage("Wette #" + response["data"]["claimid"] + "
erfolgreich beantwortet! Eine Auszahlung ist jetzt möglich!");
      break;

     // Manuelle Ergebnisbestimmung
     default:
     case null:
     case undefined:
      tr.find('.td-actions').removeClass('disabled');
      tr.find('.td-actions .claim-no').removeClass('disabled');
      tr.find('.td-actions .claim-yes').removeClass('disabled');
      tr.addClass('info');
      showInfoMessage("Wette #" + response["data"]["claimid"] + " erfolgreich
beantwortet! Eine Evaluierung ist jetzt möglich!");
      break;

    }
   } else {
    showErrorMessage("Fehler beim Abfragen der Daten für die Wette!");
   }
  },
  dataType: 'json'
 });
}
```

Anhang 5 - Quelldateien

Auf der beiliegenden CD befinden sich Quelldateien zu Frontend, Backend und der Datenbank.

Literatur- und Quellenverzeichnis

[1] c. MobiLens und statista, „comScore MobiLens,“ 2009-2015. [Online].

[2] Microsoft Technet, „Data Integrity,“ 17 April 2015. [Online]. Available: https://technet.microsoft.com/en-us/library/aa933058%28v=sql.80%29.aspx.

[3] MySQL, „The InnoDB Storage Engine,“ 20 04 2015. [Online]. Available: http://dev.mysql.com/doc/refman/5.0/en/innodb-storage-engine.html.

[4] OpenGeoDB, „OpenGeoDB,“ 2 Mai 2015. [Online]. Available: http://opengeodb.org/wiki/OpenGeoDB.

[5] Starling, „Starling - The Cross Platform Game Engine,“ 5 Mai 2015. [Online]. Available: http://gamua.com/starling/.

[6] Feathers, „Feathers - Open Source User Interface Components,“ 6 Mai 2015. [Online]. Available: http://feathersui.com/.

[7] Starling, „Textures and Images,“ 7 Mai 2015. [Online]. Available: http://wiki.starling-framework.org/manual/textures_and_images.

[8] Feathers, „The Feathers user interface component lifecycle,“ 8 Mai 2015. [Online]. Available: http://feathersui.com/help/component-lifecycle.html.

[9] SQLite, „SQLite,“ 13 Mai 2015. [Online]. Available: https://www.sqlite.org/.

[10] Apple, „iOS Developer Program,“ 9 Mai 2015. [Online]. Available: https://developer.apple.com/programs/ios/.

[11] Adobe, „Package and export a mobile application to an online store,“ 10 Mai 2015. [Online]. Available:

http://help.adobe.com/en_US/flex/mobileapps/WS4bebcd66a74275c3-416562ab12ee52291fa-8000.html.

[12] W. API, „Wunderground API,“ 20 04 2015. [Online]. Available: http://www.wunderground.com/weather/api/.

[13] JSON, „JSON,“ 15 April 2015. [Online]. Available: http://www.json.org.

[14] Head First, Design Patterns.

Abbildungsverzeichnis