Tippspiel

Im Modul *Workshop Distributed Software Systems* werden die Themen der Vertiefungsrichtung *Verteilte Software Systeme* in einem durchgehenden Beispiel angewendet und umgesetzt. Dieses Dokument beschreibt die Aufgabenstellung für die Durchführung 2018.

1. Problembeschreibung

In diesem Jahr findet in Russland die Endrunde der Fussball-Weltmeisterschaft statt. Aus diesem Anlass soll im Rahmen des Moduls DSS Workshop eine Plattform entwickelt werden, auf der Tipps auf die Resultate der 64 WM-Spiele abgegeben werden können (ein Tipp-Spiel). Ihre Aufgabe ist es, eine Web-Applikation zu entwickeln, über die die Teilnehmer Tipps zu den Spielen abgeben können und mit welcher der beste Tipper und das beste Tipp-Team gekürt werden können.

Nutzer sollen sich auf der Plattform registrieren können und können danach für die Spiele der Gruppenphase und der KO-Phase ihre Tipps abgeben. Die Tipps dürfen natürlich nur abgegeben werden solange das Spiel noch nicht angepfiffen wurde. Für jeden Tipp werden nach Ende des Spiels automatisch Punkte vergeben die sich auf den Punktestand der Tipper auswirken. Sie sind frei in der Definition, wie sie die Punkte verteilen wollen. Punkte gibt es natürlich wenn das Resultat exakt vorhergesagt wurde, aber es könnte auch Punkte geben falls Sieger und Tordifferenz bzw. eine Torzahl richtig getippt wurden oder falls zumindest die Tendenz (Sieger bzw. Unentschieden) richtig ist. Dabei gelte jeweils das Endergebnis des Spiels nach 90 Minuten resp. nach 120 Minuten im Falle einer Verlängerung, also nach der regulären Spielzeit (exkl. Elfmeterschiessen!).

Um ein Gesamtranking über alle am Tippspiel teilnehmenden Personen zu berechnen muss überlegt werden, wie bei Punktegleichheit vorzugehen ist. Man könnte die Arten der Resultate gewichten und z.B. bei Punktegleichheit jene zu bevorzugen, welche mehr richtige exakte Vorhersagen gemacht haben. Bei Punktegleichheit kann entweder das Los entscheiden oder die Tipper teilen sich den entsprechenden Platz.

Nach jedem Spiel muss ein Administrator das Spielresultat erfassen. Allenfalls können Sie das Resultat auch automatisch aus einem News-Ticker oder von einem FIFA-Dienst (wenn es denn so einen gibt) extrahieren. Für die KO-Phase sind die beteiligten Mannschaften auf Grund der Resultate in der Gruppenphase bekannt. Sie können die an der KO-Phase teilnehmenden Mannschaften also automatisch (auf Grund der Resultate der Gruppenphase) oder aber von Hand über einen Administrator erfassen. Ob Tipps auf die Spiele der KO-Phase bereits abgegeben werden können bevor die Teilnehmer feststehen ist Ihnen überlassen.

2. Zielsetzung

Das Ziel dieser Arbeit liegt darin, ein relativ einfaches Geschäftsmodell mit Hilfe von Internet-Technologien komponentenbasiert abzubilden. Wir wollen uns dabei auf die im Unterricht vorgestellten Plattformen konzentrieren. Das GUI-Frontend sollen Sie deshalb als eine Single Page Applikation (SPA) entwickeln und für das Backend wird Spring Boot mit einem REST-API und Spring-Data eingesetzt.

Da die Aufgabe ziemlich umfangreich ist, soll sie in Gruppen zu 3 Studierenden bearbeitet werden. Die Arbeit wird benotet und ergibt die Modulnote.

3. Anforderungsbeschreibung

Funktionale Anforderungen

Die Benutzer, welche auf Ihrer Plattform Tipps abgeben wollen, können sich selbständig auf der Plattform registrieren. Falls sie das Passwort vergessen haben so sollen sie dieses zurücksetzen können (Notifikation per Email).

Ein Nutzer kann mehreren Tipp-Teams beitreten (und auch wieder austreten) und kann auch eigene Tippteams erzeugen (und wieder löschen).

Auf der Plattform soll in einer Tabelle die aktuelle Rangliste dargestellt werden. Damit man den Platz einer bestimmten Person leicht findet, soll eine Suche angeboten werden. Die Rangliste der Tipp-Teams soll ebenfalls abrufbar sein. Die Applikation und das entsprechende User Interface muss so gestaltet werden, dass eine Teilnehmerzahl im 3-stelligen und eine Teamzahl im 2-stelligen Bereich effizient behandelt werden kann.

Bei jedem Spiel soll abgefragt werden können, was die Mehrheit getippt hat (entweder nur prozentuale Angaben zu Sieg/Unentschieden/Niederlage oder prozentuale Angaben zu den Endresultaten).

Diese und weitere Interaktionen sind in Abbildung 1 dargestellt.

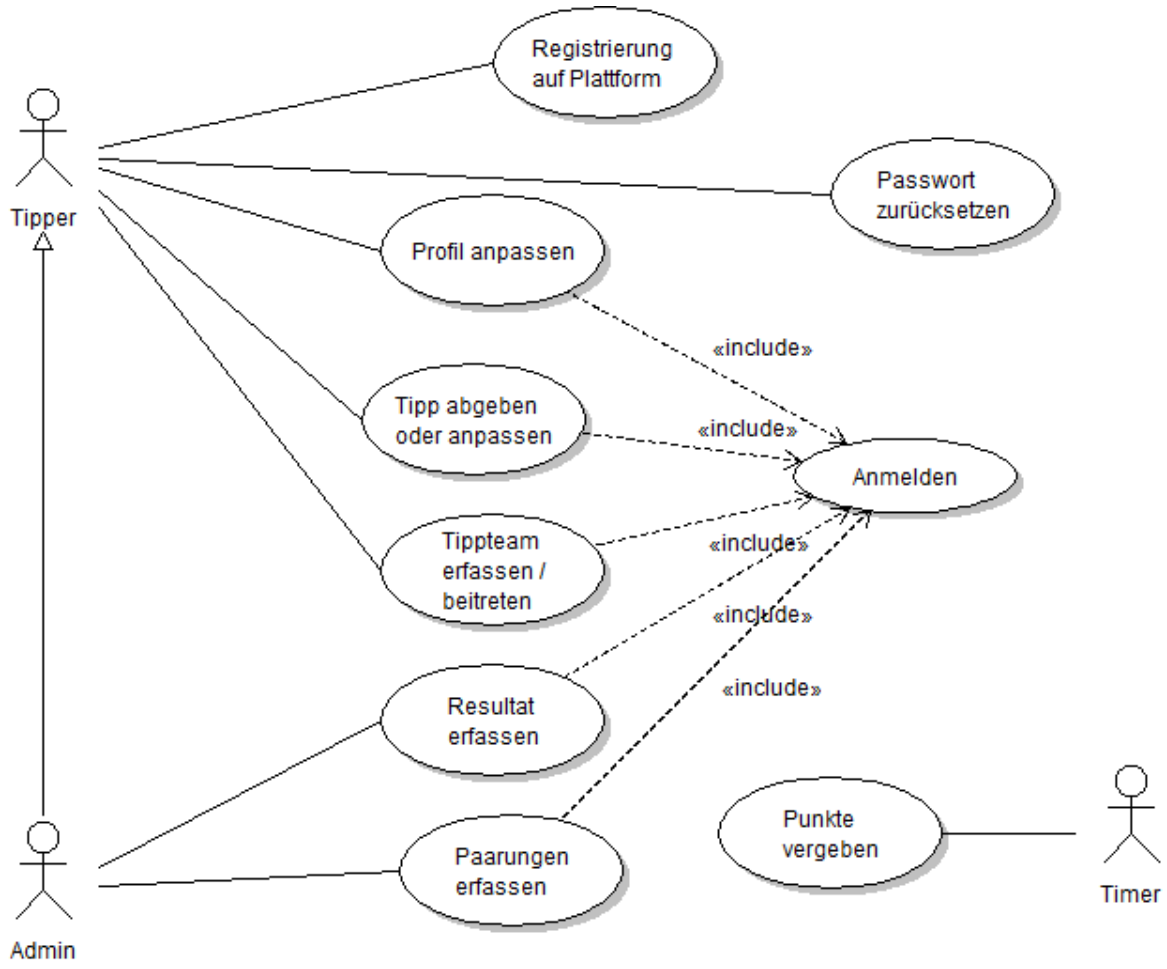

Abbildung 1: Use Case Diagramm

Folgende Rollen sind vorgesehen:

Tipper
Kann sich selbständig auf der Plattform registrieren und nach der Registrierung Tipps für die Spiele abgeben. Pro Spiel ist pro Tipper nur ein Tipp möglich. Dieser Tipp kann jederzeit (vor Spielanpfiff) geändert (oder gelöscht) werden.

Admin
Der Administrator kann die Resultate erfassen (wenn diese nicht automatisch von einem Dienst übernommen werden können) und er kann auch die Teilnehmer der KO-Phase erfassen (falls diese nicht automatisch aus den Resultaten der Gruppenphase berechnet werden können). Die Liste der an der WM teilnehmenden Mannschaften muss nicht editierbar sein, diese können einmalig (z.B. von einem DB-Skript) in die Datenbank geladen werden.

Timer
Ein Timer stellt sicher, dass die Resultate der Spiele von einem entsprechenden Dienst übernommen werden (falls diese nicht durch einen Administrator erfasst werden) und steuert die Zuweisung der Punkte auf die Tipper (falls die Punkte nicht *on demand* berechnet werden). Zudem könnten auch regelmässig Emails an die Teilnehmer verschickt werden (z.B. mit der Angabe, welcher Tipper bzw. welches Team aktuell gerade führt).

Nichtfunktionale Anforderungen

Technologie:
- Die Applikation muss auf einem Unix-basierten Server eingesetzt werden können und soll als `init.d`-Service registriert werden. Infos dazu finden Sie unter http://docs.spring.io/spring-boot/docs/current/reference/html/deployment-install.html
- Es müssen die modernen Browser-Versionen unterstützt werden (Chrome >= 64, Firefox >= 58, Edge >= 16).
- Es soll ein *Java Application Server* verwendet werden (Spring Boot, Tomcat oder JavaEE)

Benutzerinterface:
- Die Applikation soll ohne Benutzerhandbuch bedienbar sein.
- Die Sprache der Applikation ist Deutsch. Eine Unterstützung für die Internationalisierung der Webpage soll aber vorhanden sein.

Architektur:
- Eine saubere Schichtenarchitektur ist notwendig. Es soll möglich sein, die Serviceschicht und die Serverapplikation später auch über weitere Clients nutzen zu können (z.B. mobile Applikation), ohne dass Änderungen im Business-Layer erforderlich werden.
- Alle relevanten Konfigurationseinstellungen müssen ohne Kompilation der Applikation möglich sein und bei einem Neustart des Servers aktiv werden.

Sicherheit:
- Allfällige Passwörter werden mit ARGON2 mit geeigneten Parametern sicher abgelegt. Es ist eine Funktion vorgesehen mit der Benutzer selbstständig ein vergessenes Passwort neu setzen können (vorzugsweise basierend auf der Email Adresse des Benutzers).
    - https://owasp.org/index.php/Password_Storage_Cheat_Sheet
    - https://owasp.org/index.php/Forgot_Password_Cheat_Sheet
- Wenn mit Session-Cookies gearbeitet wird, dann werden die üblichen Sicherheitsmassnahmen implementiert, z.B. Setzen der "HttpOnly" und "secure" Flags. Weiterhin werden keine Daten die der Benutzer nicht sehen darf (wenn es solche gibt) unverschlüsselt in Cookies abgelegt.
    - https://www.owasp.org/index.php/HttpOnly
    - https://www.owasp.org/index.php/Session_Management_Cheat_Sheet
- Das Zugriffskontrollmodell ist adäquat
    - https://owasp.org/index.php/Access_Control_Cheat_Sheet
- Die App weist keine SQL-Injection- oder ähnliche Injection-Schwachstellen auf
    - https://owasp.org/index.php/Query_Parameterization_Cheat_Sheet
    - https://owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
- Die Kommunikation erfolgt immer über HTTPS.
    - entsprechende HTTP-Response-Headers werden genutzt
    - Falls die Web-App gegenüber einem Web-Service auch als Client fungiert, siehe http://www.cs.utexas.edu/~shmat/shmat_ccs12.pdf
- Die App weist keine XSS-Schwachstelle auf
    - https://owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet
- Die App weist keine CSRF-Schwachstelle auf
    - Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet, https://owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet
- Die App hat insgesamt ein dem Anwendungsszenario angemessenes Sicherheitsniveau.
- Sämtliche Aktivitäten (abgeschlossene Use Cases, und jedes Versenden von Emails) werden in einer Log-Datei protokolliert.
- Es gibt keine Information Leackage, insbesondere beim Log-In und bezüglich Daten anderer Benutzer.

- Die Applikation verhält sich bei Überlast sinnvoll und erleichtert insbesondere DoS (Denial-of-Service) Angriffe nicht.

siehe auch:
- Iron-Clad Java: Building Secure Web Applications
  http://www.mhprofessional.com/product.php?isbn=0071835881&cat=112
- Slides zur Vorlesung APSI Herbst 2017

4. Architektur und Aufgaben

Grundsätzlich soll diese Applikation in mehreren logischen Schichten aufgebaut werden:

- Webschicht
- Geschäftslogikschicht
- Datenbankschicht

Die Datenbankschicht bildet eine standardisierte Schnittstelle zur Datenbank. Das zugrunde liegende Datenbankmodell wird durch diese Schicht verborgen. Diese Schicht ist in Java mit JPA zu realisieren und der Zugriff soll mit Spring Data erfolgen. Alternativ darf auch JOOQ verwendet werden. Falls Sie eine andere Technologie für die Persistenzschicht verwenden möchten (z.B. eine NoSQL Datenbank) dann fragen Sie kurz ihre Betreuer.

Basierend auf der Datenbankschicht baut die Geschäftslogikschicht auf. In der Geschäftslogik werden die Geschäftstransaktionen abgewickelt. Eine Geschäftstransaktion bildet die Geschäftsregeln ab (z.B. das Abgeben eines Tipps oder das Vergeben der Punkte). Die Geschäftslogik soll auf Basis des Spring Frameworks realisiert werden. Zur Laufzeit soll ein aussagekräftiges Logging die Nutzung der Logik aufzeigen.

Die Webschicht muss die Daten über ein REST-API der Webapplikation zur Verfügung stellen.

Die Webapplikation ist eine Single Page Applikation und wird mit React (oder Angular) erstellt.

Für den Versand von Emails (z.B. Password-Reset) verwenden Sie das JavaMail API. Um regelmässige Aktionen auszuführen (z.B. das Vergeben der Punkte) muss ein Batch-Job realisiert werden, der im Hintergrund eine Aufgabe in einem regelmässigen Intervall ausführt.

Für die Authentifizierung / Autorisierung auf Ihrer Plattform ist JAAS oder Spring Security zu verwenden.

Für Ihre Geschäftslogik sollen Test-Fälle bereitgestellt werden (JUnit-Tests). Ihr Projekt soll am Ende so bereitgestellt sein, dass wir Ihre Projekte lokal installieren und die Testfälle ausführen können.

5. Organisation

Da die Aufgabe ziemlich umfangreich ist, soll sie in Gruppen zu 2-3 Studierenden bearbeitet werden (fünf Gruppen mit drei, zwei Gruppen mit zwei Studierenden). Die Applikation und die Dokumentation werden benotet und ergeben die Modulnote.

6. Termine

Mo, 19.02.18 Formierung der Teams (Angabe per Email)
19 Studierende => typischerweise 3er Teams (zwei 2er Team)

Mo, 05.03.18 Abgabe eines Projektplans (per Email an die Betreuer)
Angaben zu Grobarchitektur und Rollenzuteilung sowie ein ungefährer Zeitplan

Mo, 26.03.18 Abgabe folgender Dokumente:
- GUI Vorschlag (auch als Papierprototyp möglich, der Ablauf der Webapplikation soll daraus ersichtlich sein).
- Statisches Klassendiagramm (Datenbankschema)
- Beschreibung der Schnittstellen der Business-Logik. Hier neben Syntax auch Semantik angeben!

Mo, 09.04.18 Zwischenreview, für alle Studierenden
Kurze Präsentation der zu realisierenden Lösung und anschliessende Diskussion mit den Dozierenden.

Mo, 28.05.18 Abgabe der Lösung: Diese enthält folgende Teile:
- Dokumentation (Beschreibung der Lösung)
- Code auf CD oder Memory-Stick (inkl. Build Umgebung und allen eingesetzten Bibliotheken)
- URL auf lauffähige Version

Mo, 28.05.18 Präsentation der Lösungen (für alle Studierenden)

Mo, 04.06.18 Verteidigungen / Projektklärungen, gruppenweise (15-18 Uhr)
Mo, 11.06.18 Verteidigungen / Projektklärungen, gruppenweise (15-18 Uhr)
Di, 12.06.18 Verteidigungen / Projektklärungen, gruppenweise (17-18 Uhr)

Dokumentation
Als Dokumentation erwarten wir eine Beschreibung Ihrer Lösung. Es sollten alle wichtigen Designentscheidungen, die Sie getroffen haben, beschrieben sein. In der Dokumentation sollen auch interessante Aspekte und Lösungen dargestellt werden und die Dokumentation soll helfen, dass wir uns im Code zurechtzufinden. Ein UML-Diagramm Ihrer Klassen ist dazu geeignet (bzw. zwingend nötig).

Den Code geben sie bitte auf Memory-Stick oder als tag auf ihrem Git Repository. Das Projekt soll von uns einfach installiert und getestet werden können (durch Ausführung der Testfälle).

Ihre Lösung soll auch auf einem Rechner zugreifbar sein, damit wir Ihre Lösung (mind. im FHNW Intranet) anschauen können. Geben Sie im Bericht die URL der Startseite an.

Geben Sie im Bericht auch an, wer in Ihrem Team welche Teile realisiert hat.

Präsentation
In der Präsentation soll auf jene Aspekte hingewiesen werden, welche die anderen Gruppen interessieren könnten, z.B. spezielle Probleme, elegante Lösungen (wie Authentisierung/Autorisierung, GUI Framework, ...), verwendete Technologien, Erfahrungen, etc.
Die Präsentation soll auch eine Live-Demo des Systems enthalten.

Mögliche Themen der Präsentation sind also:
- Architektur Ihrer Lösung
- Erfahrungen mit Web Framework
- Erfahrungen mit Applikationsserver
- Spezielle Sicherheitsmerkmale, die Sie in Ihrer Applikation berücksichtigt haben

Sie haben für diese Präsentation 15-20 Min Zeit (inkl. Diskussion)

7. Bewertung

Ihre Arbeit (Dokumentation + Code) wird mit einer Projektnote bewertet. In diese Note fliessen folgende Aspekte ein:

- Fachliche Umsetzung: 75%
- Dokumentation: 15%
- Präsentation: 10%

Pro Gruppe gibt es im Normalfall *eine* Projektnote.

Nach der Abgabe der Dokumentation findet eine Klärung statt. In dieser Klärung werden die Dozierenden dem Team spezifische Fragen zur Lösung stellen. Fragen zu allen Themengebieten können an alle gerichtet werden. Diese Klärung dauert 60 Min. Anwesend sind alle am Modul beteiligten Dozierenden und die Mitglieder jedes Teams.

Die Projektnote kann auf Grund der Klärung individuell korrigiert werden.

Der Studierende hat das Modul bestanden, falls

- die individuelle Projektnote genügend ist (>= 3.8)
- die Testat-Bedingung erfüllt ist, d.h. Anwesenheit bei Zwischenreview, Schlusspräsentation und Verteidigung.

8. Betreuung

Die Dozierenden stehen bei Fragen zur Verfügung:

- Sicherheit	Arno Wagner
- Web Framework	Dierk König
- Web-Framework / Architektur	Jürg Luthiger
- Services / ORM	Dominik Gruntz

Wir empfehlen bei Problemen einen Termin zu vereinbaren und vorab das Problem per Email zu schildern.