Universität Leipzig

Softwaretechnik-Praktikum

Entwurfsbeschreibung Gruppe cz17a - Gamification

Lisa Vogelsberg, Felix Fink, Michael Fritz, Thomas Gerbert, Steven Lehmann, Fabian Ziegner, Willy Steinbach, Christian Schlecht

supervised by
Dr. Christian Zinke, Julia Friedrich, Christian Frommert

19. März 2018

Inhaltsverzeichnis

1 Allgemeines **2**

2 Produktübersicht **2**

3 Grundsätzliche Struktur - und Entwurfsprinzipien **3**

4 Struktur - und Entwurfsprinzipien einzelner Pakete **5**
4.1 Arbeitspaket 1 - Vorprojekt 6
4.1.1 Server 6
4.1.2 Client - Quizapp 7
4.2 Arbeitspaket 2 - Spieler und Quiz 7
4.2.1 Server 7
4.2.2 Client 7
4.3 Arbeitspaket 3 - Spielmechanismus 7

5 Datenmodell **7**

6 Glossar **8**

1 Allgemeines

Dieses Projekt setzt sich mit der Entwicklung eines gamifizierten Wissenquiz auseinander, welches in Form von betrieblicher Weiterbildung zum Einsatz kommen soll. Es steht also das Aneignen neuen Wissens im Vordergrund, jedoch soll der als "Last" empfundene Lernprozess durch Gamification so unterdrückt werden, dass freiwillig auf diese Methode zurückgegriffen wird. Somit soll durch das Spielen des Quiz der Lernprozess Spaß bereiten. Das Quiz wird als App für Android-Endgeräte verfügbar gemacht.

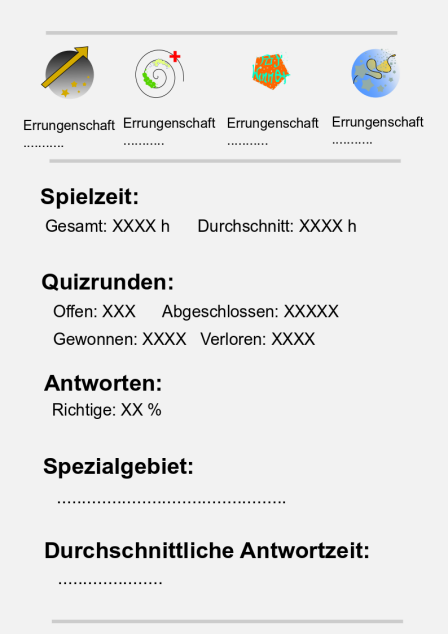

Abbildung 2.1: Dashboardansicht für den User

2 Produktübersicht

Die Quiz-App wird für mobile Android - Endgeräte entwickelt. Ein Spieler loggt sich unter Angabe von Nutzernamen und Passwort ein und kann dann entweder sein Dashboard oder sein Spielerprofil ansehen oder ein Spiel starten. Wenn er ein Spiel startet, wird ihm eine Übersicht über verfügbare Themen gegeben, von denen er eine auswählt, um in die Spiel-Lobby zu gelangen. Sobald mindestens fünf Spieler in der Lobby sind, startet das Spiel. Dabei erfolgt das Spielen anonym, das heißt die Spieler wissen untereinander nicht, gegen wen sie antreten. Es werden darauffolgend die Fragen und Auswahlmöglichkeiten dargestellt, von denen der Spieler eine Antwort auswählt und eine Rückmeldung darüber erhält, ob seine Antwort richtig oder Falsch war. Ggf. wird die richtige Antwort angezeigt. In den Abbildungen 2.1, 2.2 und 2.3 sind beispielhafte Ansichten dargestellt.
Zusätzlich wird eine Administrationsoberfläche in Form einer Webanwendung geben, in welcher der Administrator Fragen bearbeiten und alle relevanten Einstellungen durchführen kann. Die Grafiken werden bei Implementierung durch Screenshots der echten User Interfaces ersetzt.

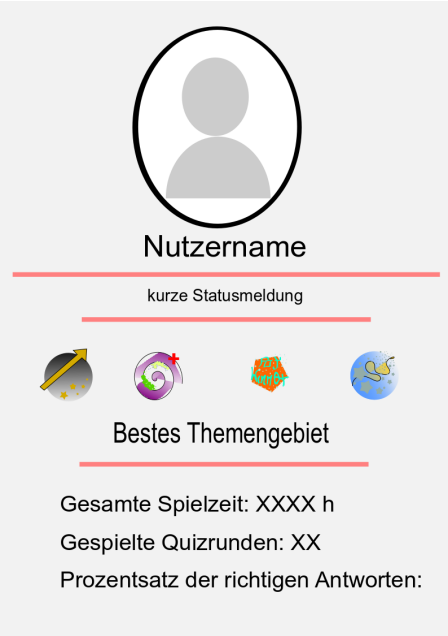

Abbildung 2.2: Profilansicht für den User

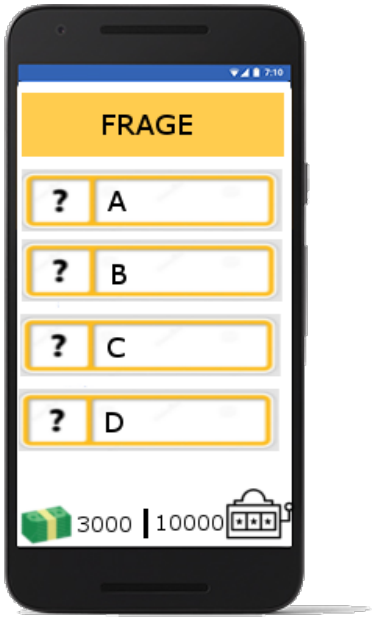

Abbildung 2.3: Fragenansicht für den User

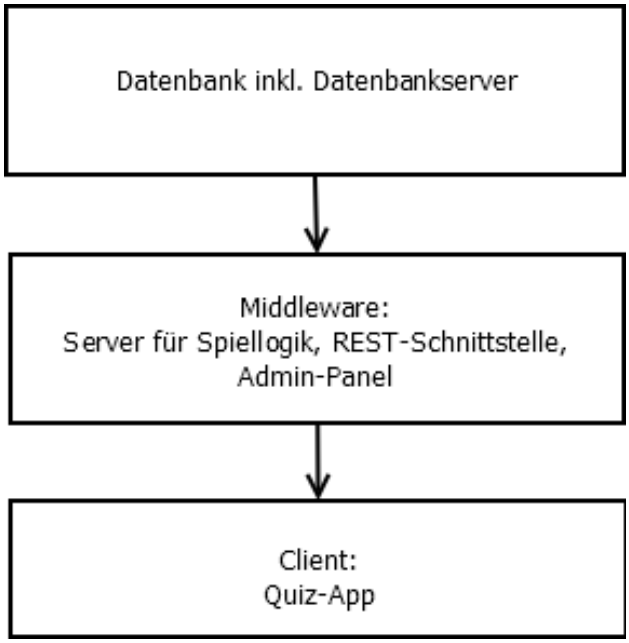

Abbildung 3.1: Architektur der Anwendung

3 Grundsätzliche Struktur - und Entwurfsprinzipien

Die gewählte Architektur ist eine Server-Client-Architektur. Sie ist unter Abbildung 3.1 schematisch dargestellt. Die erste Schicht bildet die PostgreSQL Datenbank inkl. Datenbankserver. Die Middleware besteht aus einem Server, welcher die Spiellogik händelt, einer REST-Schnittstelle und dem Admin-Panel. Der Client, also die eigentliche App, kommuniziert ausschließlich über REST mit Server und Datenbank. Der Server hingegen kann über ein objektrelationales Datenbankmapping mittels Hibernate direkt auf die Datenbank zugreifen. Dazu werden spezielle Klassen (DataAccessObjects, DAO) verwendet. Als Build-Tool wird auf der Serverseite Maven und auf Clientseite Gradle verwendet.

4 Struktur - und Entwurfsprinzipien einzelner Pakete

4.1 Arbeitspaket 1 - Vorprojekt

4.1.1 Server

Das UML Klassendiagramm ist zu groß, um es innerhalb dieses Dokuments leserlich einzubinden und ist daher separat verfügbar.
Der Server übernimmt die zentrale Verarbeitung aller Vorgänge im Quiz.

REST-Schnittstelle

Der Server verfügt über eine REST-Schnittstelle zur Kommunikation mit dem Client. Es steht diverse GET- und POST-Requests zur Verfügung. Es können User, Fragen und Antworten, Runden, verfügbare Quiz' und eine Anfrage des Clients, ein Spiel zu starten, angefragt werden. Die REST-Schnittstelle ist unter der Webseite des Praktikums [1] mit einer Übersicht über verfügbare Anfragen einsehbar. Die Clients kommunizieren ausschließlich über diesen Webserver.

Game-Server

Den zweiten Teil des Servers bildet die Einheit zum Handling der Spiellogik. Da im Vorprojekt noch keine solche Logik existiert, werden bereits implementierte Funktionen noch nicht verwendet, jedoch für die nächsten Arbeitspakete benötigt. Da der Game-Server ohne REST auf die Daten zugreifen kann, wurden um Übersicht zu wahren, Data-Access-Object-Klassen eingerichtet. Diese agieren mittels Hibernate auf den Daten. Andere Klassen sollen nicht selbst auf die Datenbank zugreifen können, da sonst schwer behebbare Fehler und unübersichtliche Zugriffsstrukturen entstehen.

Admin-Panel

Das Admin-Panel ist ebenfalls eine Web Applikation. In diese kann sich der Administrator über Nutzername und Passwort einloggen. Dort sind alle verfügbaren Einstellungen vorzunehmen (spätere Arbeitspakete) und es können die Fragen in die Datenbank importiert werden. Das Admin-Panel wird an den Server angebunden und ebenfalls über den Praktikumsserver erreichbar sein. Da es nicht getrennt wird, kann es ebenfalls die DataAccessObject-Klassen zur Datenbankmodifikation verwendet. Somit spart man sich an dieser Stelle REST-Abfragen.

[1] `http://pcai042.informatik.uni-leipzig.de:1810/restserver`

4.1.2 Client - Quizapp

Der Client stellt per REST eine Anfrage an den Server, um eine Quizfrage zu bekommen. Diese enthält auch Informationen über die Antwortmöglichkeiten und welche richtig ist. Um Datenredundanz zu vermeiden, ist in der Liste der Antworten stets die erste Antwort die Richtige. Bei Anzeige der Möglichkeiten werden die Antworten durchgemischt, um nicht auch hier stets die erste Antwortmöglichkeit als die Richtige identifizieren zu könnnen. Nach Auswahl durch den User erscheint eine Rückmeldung, ob die gegebene Antwort richtig oder falsch ist und ggf. wird die korrekte Antwort angezeigt. Es werden noch keine Speichervorgänge für die Statistiken durchgeführt.

4.2 Arbeitspaket 2 - Spieler und Quiz

4.2.1 Server

REST-Schnittstelle

Das Registrieren, Anmelden, Abmelden und Löschen eines Spieleraccounts erfolgt über die REST-Schnittstelle. Bei Registrierung erstellt sie anhand der angegeben Daten einen neuen User in der Datenbank. Für den Login wird anhand des angegebenen Usernamen die Übereinstimmung des eingegebenen Passworts mit dem in der Datenbank verglichen. Als Antwort erhält der Client stets einen Statuscode: 200 für Erfolg, 400 bzw. 404 für Misserfolg.

Game-Server

Der Server verfügt über Methoden, um die grundlegende Spiellogik zu verarbeiten. Dazu zählen das erstellen und verwalten einer Lobby sowie die grundlegende Spiellogik. Sobald fünf Spieler in einer Lobby sind, startet das Quiz. Am Ende einer Quizrunde erfolgt das Senden der Ergebnisse an die teilnehmenden Clients.

4.2.2 Client

Der Client wird um Screens für Registrierung, Login und Logout erweitert. Die eingegebenen Daten werden als POST-Request an die REST-Schnittstelle zur Verarbeitung gesendet.

4.3 Arbeitspaket 3 - Spielmechanismus

Es wird die Kommunikation zwischen Server und Client implementiert. Über die Sockets wird der Spielablauf gesteuert, das heißt Signale vom Server zum Client zum Starten eines Spiels und zum Starten einer neuen Frage werden über die Sockets gesendet. Die Antwort vom Client darauf wird jedoch über REST erfolgen.

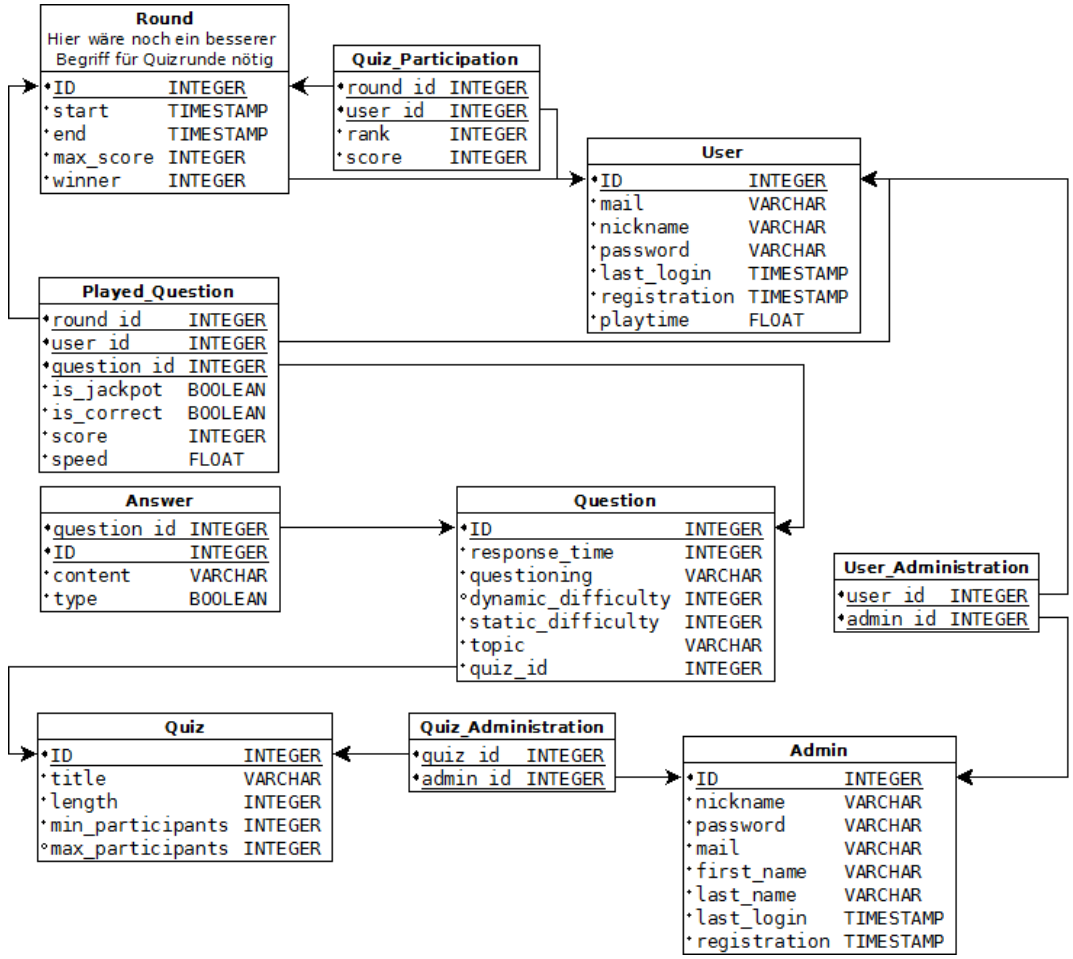

Abbildung 5.1: verwendetes Datenbankschema

5 Datenmodell

Die gewählte Datenbank ist eine PostgreSQL-Datenbank, da diese objektrelationales Mapping unterstützt. Sie ist ebenfalls auf dem Praktikumsserver installiert. Die Abbildung 5.1 zeigt das verwendete Schema.

6 Glossar

Gamification
: ist der Einsatz von spieltypischen Elementen und Vorgängen, die in einen spielfremden Zusammenhang gebracht werden. Ziel ist dabei die Motivationssteigerung des Anwenders für die Kernleistung. Häufig verwendete Spielmechaniken sind z.B. Punkte, Ziele und Errungenschaften.

Server-Client-Prinzip
: beschreibt eine grundlegendes Architekturprinzip, in welchem ein zentrales Objekt (Server) verschiedene Dienste bereit stellt. Ein Client kann eine Anfrage an diesen Server schicken und erhält eine gewünschte Rückmeldung (z.B. Client fragt nach Wetter in Berlin, Server antwortet mit 20 Grad, Sonne).

REST-Schnittstelle
: Die REST-Schnittstelle ist ein Teil des Servers, welche für die Kommunikation mit den Clients zuständig ist. Sie ist zumeist Web-basiert. Ein Client kann verschiedene Anfragen (z.B. GET um Daten abzufragen, POST um Daten zu senden) über eine URL stellen und erhält ggf. eine Antwort in einem standardisierten Format (zumeist XML oder JSON). REST basiert auf dem World Wide Web und arbeitet mit HTTP-Standards.

Hibernate
: ist ein Framework zur Einbindung eines objektrelationalen Datenbankmappings. Es können über spezielle Klassen die Objekte anstatt SQL-Queries für den Datenbankzugriff verwendet werden.