23.01.2020

User

BBBaden

IPERKA M151

Projektdokumentation zum Modul 151 «Datenbanken in Webauftritt einbinden»

Inhaltsverzeichnis

[1 Informieren 2](#_Toc30703069)

[1.1 Einleitung 2](#_Toc30703070)

[1.2 Anforderungsanalyse 3](#_Toc30703071)

[1.3 Ideen 4](#_Toc30703072)

[2 Planen 5](#_Toc30703073)

[2.1 Systemgrenze 5](#_Toc30703074)

[2.2 Use-Case 5](#_Toc30703075)

[2.3 Klassendiagramm 5](#_Toc30703076)

[2.4 Mock-Up (GUI) 6](#_Toc30703077)

[2.4.1 Quiz-Seite 6](#_Toc30703078)

[2.4.2 Admininterface 6](#_Toc30703079)

[2.5 Storyboard 7](#_Toc30703080)

[2.6 Konzeptionelles Datenmodell (Datenbank) 7](#_Toc30703081)

[2.7 Logisches Datenmodell (Datenbank) 7](#_Toc30703082)

[2.8 Testfallspezifikationen 8](#_Toc30703083)

[3 Entscheiden 9](#_Toc30703084)

[3.1 Entscheidungsmatrix 9](#_Toc30703085)

[3.2 Entscheidungen 9](#_Toc30703086)

[3.2.1 Dynamische Elemente der Anwendung 9](#_Toc30703087)

[4 Realisieren 10](#_Toc30703088)

[4.1 Programm 10](#_Toc30703089)

[4.1.1 Ideenumsetzung 10](#_Toc30703090)

[4.2 GUI 10](#_Toc30703091)

[5 Kontrollieren 11](#_Toc30703092)

[5.1 Testprotokoll 11](#_Toc30703093)

[5.2 Testfazit 12](#_Toc30703094)

[6 Auswerten 13](#_Toc30703095)

[6.1 Reflexion 13](#_Toc30703096)

[7 Anhang 14](#_Toc30703097)

[7.1 Quellen 14](#_Toc30703098)

[7.2 Code 14](#_Toc30703099)

# Informieren

## Einleitung

Ziel dieses Projekts ist es, eine Web-Applikation zu implementieren und diese mit einer Datenbank zu verknüpfen. Die Applikation muss in die 4 N-Tiers getrennt werden: Presentation, Webserver, Application Server, Dataserver.

Die Datenbank muss mit Hilfe einer Schnittstelle an die Applikation gebunden werden, welche Daten einlesen kann und alle CRUD-Operationen abdeckt. Sowie ein ORM Framework welches die eingelesenen Daten in Objekte mappt.

Wie gewohnt muss die Applikation client-seitig sowie im back-end abgesichert werden und alle Interaktionen mit der Datenbank sollen mit Prepared Statements durchgeführt werden damit Angriffsmöglichkeiten auf diese durch den Benutzer vermindert werden.

Spezifisch muss hier das Spiel «Wer Wird Millionär?» implementiert werden. Benutzer können Fragen mit einer Auswahl von 4 Antworten beantworten. Dazu müssen Systemadministratoren mit einem Admininterface Fragen hinzufügen bzw. entfernen können, sowie Kategorien verwalten und Einträge aus der Highscoreliste löschen.

## Anforderungsanalyse

**Anforderungsanalyse:** WerWird Millionär?

|  |  |  |
| --- | --- | --- |
| Anforderungs-nummer | Anforderung | Beschreibung |
| 1 | Grundanforderungen |  |
| 1.1 | Tiers | Die verschiedenen Komponenten des Systems werden voneinander und der Datenbank getrennt. Presentation-Layer wird für jeden Client geladen und Daten davon nicht getraut, Business-Logik wird einmal im dritten Tier programmiert und die Datenbank ist nur für den Server sichtbar. |
| 1.2 | Persistenz | Quizfragen und Antworten, sowie Benutzerstatistiken werden in einer Datenbank hinterlegt und vom Back-End mit Hilfe eines Database-Connectors und einem ORM Framework eingelesen und an das Front-End weitergegeben. |
| 1.3 | Quiz | Der Benutzer kann seinen Namen eingeben und das Quiz starten. Fragen werden angezeigt und dazu 4 Antworten wovon nur eine richtig ist. Der Benutzer kann am Anfang die Kategorie, von welcher die Fragen gestellt werden, auswählen. |
| 1.3.1 | Zeitmessung | Während dem Quiz wird die vergangene Zeit seit dem Start gemessen und dem Benutzer angezeigt, um die Recherchemöglichkeiten einzuschränken. |
| 1.3.2 | 50:50 Joker | Der Benutzer kann einmal den 50:50 Joker einsetzen welches zwei von den drei Falschen Antworten ausblendet und das Beantworten der Frage vereinfacht. |
| 1.3.3 | Fragen beantworten | Für richtig beantwortete Fragen erhält der Benutzer 30 Stück der Geldwährung und kann die nächste Frage spielen, bzw. das Spiel dann beenden. Bei falsch beantworteten Fragen geht das ganze Geld verloren und das Spiel wird beendet. |
| 1.4 | Highscoreliste | Benutzer können die Statistiken anderer Spieler anzeigen lassen und vergleichen; Der Rang in der Highscoreliste, die gewichteten Punkte, der Name des Spielers, der Zeitpunkt des Spiels, die Anzahl errungenen Punkte, die Dauer des Quiz und die gewählten Kategorien sollen alle ersichtlich sein. |
| 1.5 | Admininterface |  |
| 1.5.1 | Quiz verwalten | Über das Admininterface können Quizfragen hinzugefügt werden, ihre Kategorie eingerichtet werden und Antworten zu jeder Quizfrage zuweisen. |
| 1.5.2 | Highscoreliste verwalten | Administratoren können einzelne Einträge von der Highscoreliste löschen. |
| 2 | Erweiterte Anforderungen |  |
| 2.1 | Fragen/Antworten kommentieren | Admins können Fragen und Antworten mit Kommentaren versehen als interne Notiz. |
| 2.2 | Statistiken | Fragen speichern wie oft sie hervorgekommen sind, Kategorien wie oft sie gewählt wurden und Antworten speichern wie oft sie von Benutzern als die richtige Antwort gewählt wurden. |

## Ideen

# Planen

## Systemgrenze

## Use-Case

## Klassendiagramm

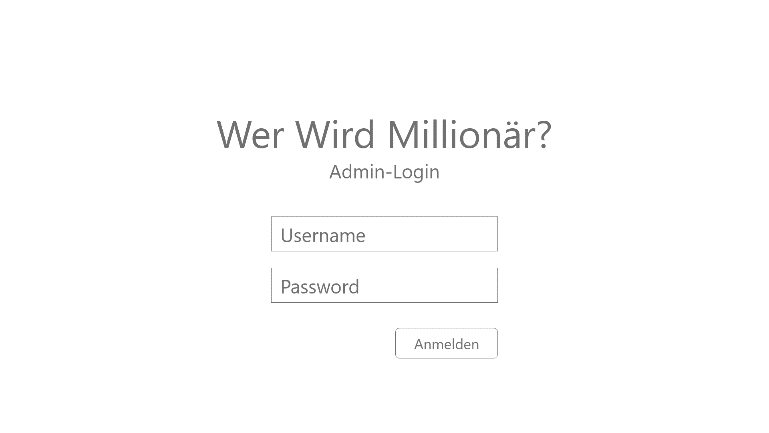
## Mock-Up (GUI)

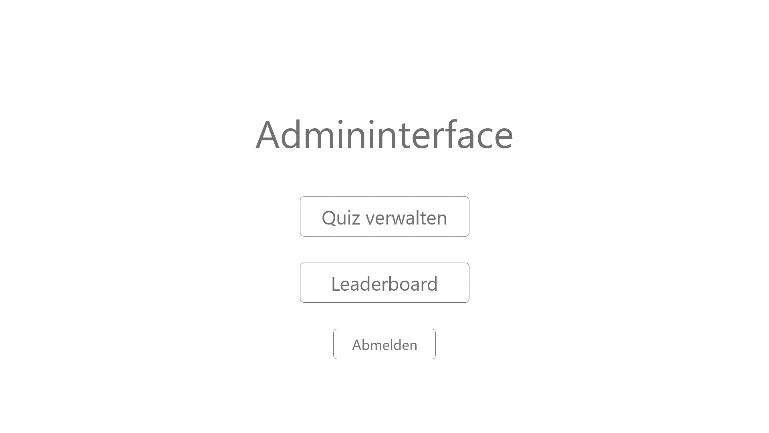
### Quiz-Seite

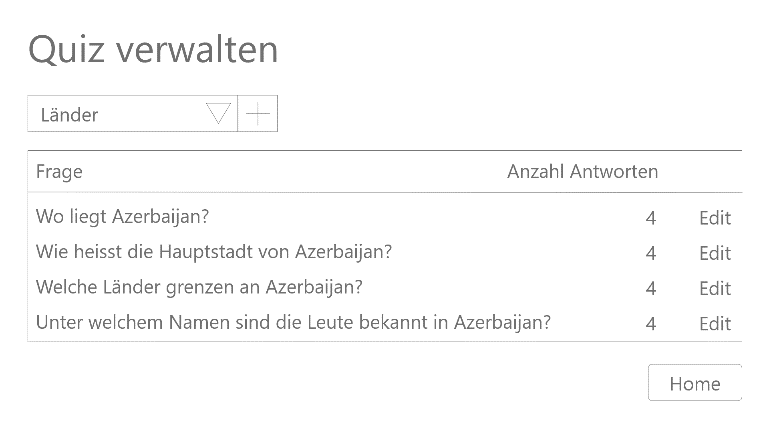
### Admininterface

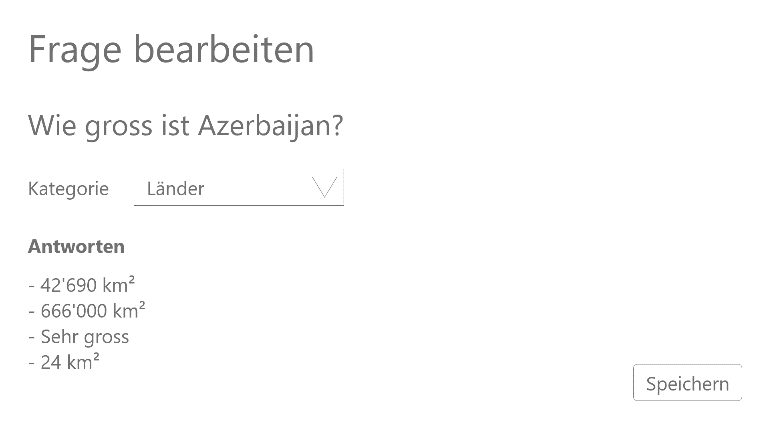
Nach dem Anmelden sehen Administratoren eine Auswahl welches ihnen erlaubt das Quiz zu bearbeiten oder Einträge aus der Highscoreliste zu löschen. Damit nicht zu viele Fragen auf einmal bei der Verwaltung des Quiz angezeigt werden, werden die Fragen unter ihrer Kategorie angezeigt und mit einem Dropdown kann zwischen Kategorien gewechselt werden. Sind mehr als eine vordefinierte Anzahl Fragen definiert kann man die Seite wechseln («Pagination») damit einzelne Seiten nicht zu viele Elemente auf einmal beinhalten.

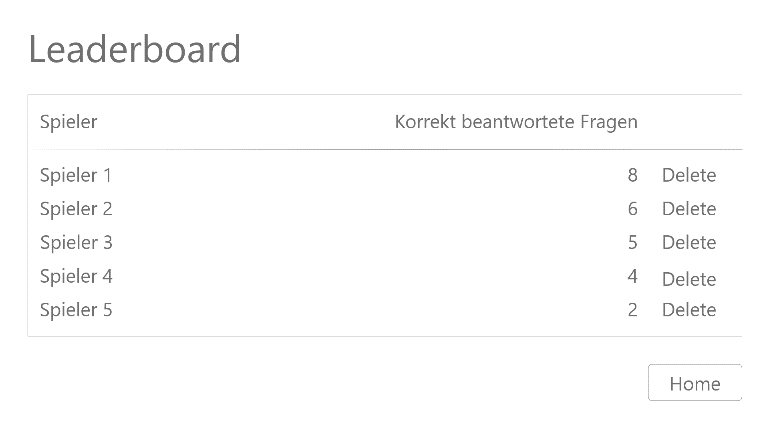
Bei der Bearbeitung einer Frage kann die Kategorie verändert werden und die Antworten der Frage können bearbeitet werden. Da jedoch jede Frage 4 Antworten erfordert können keine Fragen hinzugefügt/gelöscht werden. Ganze Fragen können von Systemadministratoren gelöscht werden.











## Storyboard

## Konzeptionelles Datenmodell (Datenbank)

## Logisches Datenmodell (Datenbank)

## Testfallspezifikationen

**Testfallspezifikation:** [Projektname]

**Betriebssystemversion:** Windows 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testfallnummer | Anforderung | Voraussetzungen | Eingaben | Erwartetes Resultat |
| 1 |  |  |  |  |
|  |  |  |  |  |

# Entscheiden

## Entscheidungsmatrix

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kriterien | Gewichtung | Idee 1 | | Idee 2 | | Idee 3 | |
| Punkte | Total | Punkte | Total | Punkte | Total |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Total |  |  | |  | |  | |

## Entscheidungen

### Dynamische Elemente der Anwendung

Die dynamischen Elemente der Anwendung werden grundsätzlich auf dem **ersten Tier (Presentation)** untergebracht da Benutzer (inklusive Administratoren) über das GUI, welches auf diesem Tier untergebracht ist, ihre Aufgaben erledigen werden.

Input-Validation passiert auf dem front- sowie back-end weshalb dieser Teil dann vom **zweiten Tier (Webserver)** übernommen wird, alle anderen dynamischen Elemente der Anwendung befinden sich, wie vorher erwähnt, auf dem **ersten Tier (Presentation)**.

# Realisieren

## Programm

### Ideenumsetzung

## GUI

# Kontrollieren

## Testprotokoll

**Testprotokoll:** [Projektname]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Datum | Tester | Bemerkungen | Resultat | Unterschrift |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Funktionstest

## Integrationstest (Selenium)

## Testfazit

# Auswerten

## Reflexion

# Anhang

## Quellen

## Code