[Dokumenttitel]

[Untertitel des Dokuments]

[Schule]

[Kurstitel]

[Entwickler]

Inhaltsverzeichnis

[1. Informieren 2](#_Toc30102692)

[1.1. Anforderung 2](#_Toc30102693)

[1.2. Anforderungsanalyse 2](#_Toc30102694)

[1.3. Ideen 2](#_Toc30102695)

[2 Planen 2](#_Toc30102696)

[1.4. Systemgrenze 2](#_Toc30102697)

[1.5. UseCase 2](#_Toc30102698)

[1.6. Klassendiagramm 2](#_Toc30102699)

[1.7. MockUp (GUI) 2](#_Toc30102700)

[1.8. Konzeptionelles Datenmodell (Datenbank) 2](#_Toc30102701)

[1.9. Logisches Datenmodell (Datenbank) 2](#_Toc30102702)

[1.10. Testfallspezifikationen 2](#_Toc30102703)

[3. Entscheiden 2](#_Toc30102704)

[3.1. Entscheidungsmatrix (optional) 2](#_Toc30102705)

[3.2. Entscheidungen 2](#_Toc30102706)

[4. Realisieren 2](#_Toc30102707)

[4.1. Programm 2](#_Toc30102708)

[4.1.1. Ideenumsetzung 2](#_Toc30102709)

[4.2. GUI 2](#_Toc30102710)

[5. Kontrollieren 2](#_Toc30102711)

[5.1. Testprotokoll 2](#_Toc30102712)

[6.1. Testfazit 2](#_Toc30102713)

[7. Auswerten 2](#_Toc30102714)

[7.1. Reflexion 2](#_Toc30102715)

[8. Anhang 2](#_Toc30102716)

[8.1. Code 2](#_Toc30102717)

# Informieren

## Anforderung

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.

## Anforderungsanalyse

**Anforderungsanalyse:** Wer Wird Millionär?

|  |  |  |
| --- | --- | --- |
| Anforderungsnummer | 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 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 ein Mal 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. |
| 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 | Fragestatistik | Fragen speichern wie oft sie hervorgekommen sind, Kategorien wie oft sie gewählt wurden und Antworten speichern wie oft sie als richtig gewählt wurden. |

## Technologien

## Ideen

//todo Alle Ideen auflisten

# Planen

## Systemgrenze

//todo Darstellung was zum Projekt und System gehört, was ausserhalb ist, wie das interagiert.

## Use-Case

//todo

## Klassendiagramm

//todo

## Mock-Up (GUI)

//todo

## Konzeptionelles Datenmodell (Datenbank)

//todo

## Logisches Datenmodell (Datenbank)

//todo

## Testfallspezifikationen

//todo

**Testfallspezifikation: [Projekttitel] Betriebssystemversion: Windows 10**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Testfallnummer** | **Anforderung** | **Voraussetzungen** | **Eingaben** | **Erwartetes Resultat** |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |
| **4** |  |  |  |  |
| **5** |  |  |  |  |
| **6** |  |  |  |  |

# Entscheiden

## Entscheidungsmatrix (optional)

//todo (Höchstbewertete Idee aus einem Themenbereich wird gewählt.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Idee | Bewertungskriterium 1 | Bewertungskriterium 2 | Gewichtung | Punkte |
|  |  |  |  |  |
|  |  |  |  |  |
| Total |  |  |  |  |

## Entscheidungen

//todo (Alle im Informieren erarbeiteten Aspekte begutachten, Für eines entscheiden) (nur wenn keine Entscheidungsmatrix vorhanden essenziell)

# Realisieren

## Programm

//todo (Beschreibung des Programms)

### Ideenumsetzung

//todo (Title umbenennen, Idee wurde im Entscheiden festgelegt…), Codebeispiele

## GUI

//todo falls GUI vorhanden darstellen und mit MockUp vergleichen, Codebeispiel falls sehr interessant

# Kontrollieren

## Testprotokoll

**Testprotokoll: [Projekttitel]**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr.** | **Datum** | **Tester** | **Bemerkungen** | **Resultat** | **Unterschrift** |
| **1** | 11.04.2017 | Dominik Berger | - | Funktioniert | D.Berger |
|  |  |  |  |  |  |

## Testfazit

//todo Fliesstext über die Funktionstüchtigkeit des Projekts und ob die Abgabe des Projekts tragbar ist.

# Auswerten

## Reflexion

//todo

# Anhang

## Code

//todo