**Flaggen-Quiz**

**Software Architecture Document**

**Version <1.0>**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 120309 |  | Initial Document for Elaboration Iteration | Peter Edelmann |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Inhaltsverzeichnis**

[1.](#_3o7alnk) Einführung

[1.1](#_23ckvvd) Zweck

[1.2](#_3znysh7) Umfang

[1.3](#_ihv636) Definitionen, Akronyme, und Abkürzungen

[1.4](#_32hioqz) Referenzen

[2.](#_1t3h5sf) Architektonische Representation

[3.](#_4d34og8) Architekturziele und Einschränkungen

[4.](#_1hmsyys) Class View

[4.1](#_41mghml) Class Diagram

[4.2](#_3rdcrjn) Architektonische Konzepte

[4.3](#_2grqrue) Short Description of Architecturally Significant Classes 7

[5.](#_vx1227) Use-Case Realizations

[5.1](#_3fwokq0) Auf Start Button drücken

[5.1.1](#_1ksv4uv) Erfolgreich auf den Start Button klicken

[5.2](#_2jxsxqh) Auf Antwort Button klicken

[5.2.1](#_z337ya) Erfolgreich auf einen Antwort Button klicken

[5.3](#_3j2qqm3) Auf Highscore Button klicken

5.3.1 Erfolgreich auf Highscore Button klicken

[6.](#_1y810tw) Komponenten Sicht

[6.1](#_4i7ojhp) Komponentendiagramm

[6.2](#_2xcytpi) Kurzbeschreibung der Komponenten

[7.](#_1v1yuxt) Paket Sicht

[8.](#_19c6y18) Deployment View

[8.1](#_3as4poj) Deployment Diagram

[8.2](#_1pxezwc) Notes

[9.](#_3tbugp1) Design of significant classes

[9.1](#_2p2csry) Class specifications

[9.2](#_147n2zr) Overview of classes

9.3 Most significant attributes and operations

9.4 State diagram UI()

9.5 Activity diagram

**Software Architecture Document**

1. **Einführung**
   1. **Zweck**

Dieses Dokument liefert eine umfassende architektonische Übersicht vom Flaggen-Quiz, dabei verwenden wir eine Reihe von architektonischen Ansichten um verschiedene Aspekte des Systems abzubilden. Es soll die wesentlichen architektonischen Entscheidungen, die an dem System getroffen wurden, erfassen und vermitteln.

* 1. **Umfang**

Dieses Software Architektur Dokument gilt für das Flaggen-Quiz-System.

* 1. **Definitionen, Akronyme und Abkürzungen**

Siehe Glossar.

* 1. **Referenzen**

Anwendbare Referenzen sind:

[1] Vision

[2] Iteration Plan Inception

[3] Glossar

[4] Software Requirements Specification

1. **Architektonische Repräsentation**Die Architektur wird mit Hilfe von Modellen, Diagrammen und UML 2.0 dargestellt.
2. **Architekturziele und Einschränkungen**

Das Quiz sollte leicht verständlich und Betriebssystems unabhängig sein.

Es soll mit Java implementiert werden.

Die JavaFX-Bibliothek soll für die Implementierung der Client-GUI verwendet werden.

Das System muss einfach zu bedienen sein und für erfahrene Personen ohne zusätzliche Schulung am System geeignet sein.

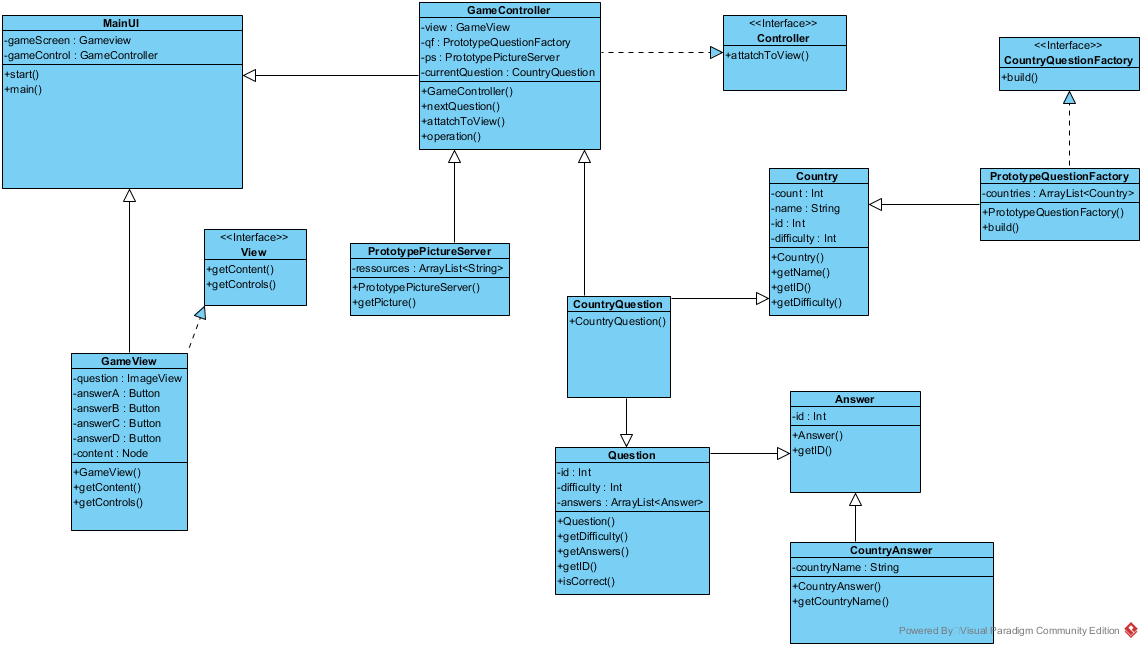
Die GUI sollte von Spiellogik und Daten getrennt sein, um eine Implementierung zu einer anderen GUI-Technologie, zum Beispiel Android, zu ermöglichen.

Der Softwareentwicklungsprozess muss dem Rational Unified Process (RUP) entsprechen.

Die resultierende Software sollte möglichst Erweiterbar und Modifizierbar sein um auf Wünsche von Benutzern nach der Veröffentlichung der Software eingehen zu können.

1. **Class View**
   1. **Class Diagram**

Nur Architektonisch bedeutsame Klassen



* 1. **Übergreifende Architektur Entscheidungen  
       
     MVC**Die Software soll dem MVC Pattern folgen um Modelle von der eigentlichen Logik und Logik von der UI zu trennen. Zudem soll es ermöglicht werden, in einer darauffolgenden Entwicklung, Abstraktionsebenen zwischen der UI und dem Rest des Programms zu erzeugen, welche es erlaubt das verwendete UI Framework auszuwechseln, ohne das der Rest des Programms beeinflusst wird.  
       
     Dies soll es ermöglichen zu einem späteren Zeitpunkt zu einer anderen Umgebung zu wechseln - wie z.B. Android.
  2. **Kurze Beschreibung der architektonisch signifikanten Klassen**

**Main UI**

Diese Klasse ist für die gesamte Benutzeroberfläche des Quizz zuständig.

**GameController**

Diese Klasse ist für die Kontrolle des Spielablaufs sowie die Modifikation der Ansicht währenddessen zuständig.

**Game View**

Diese Klasse ist für Ansicht während des Spiels zuständig.

**Question**

Diese Klasse stellt eine Frage und die dazugehörigen möglichen Antworten zuständig.

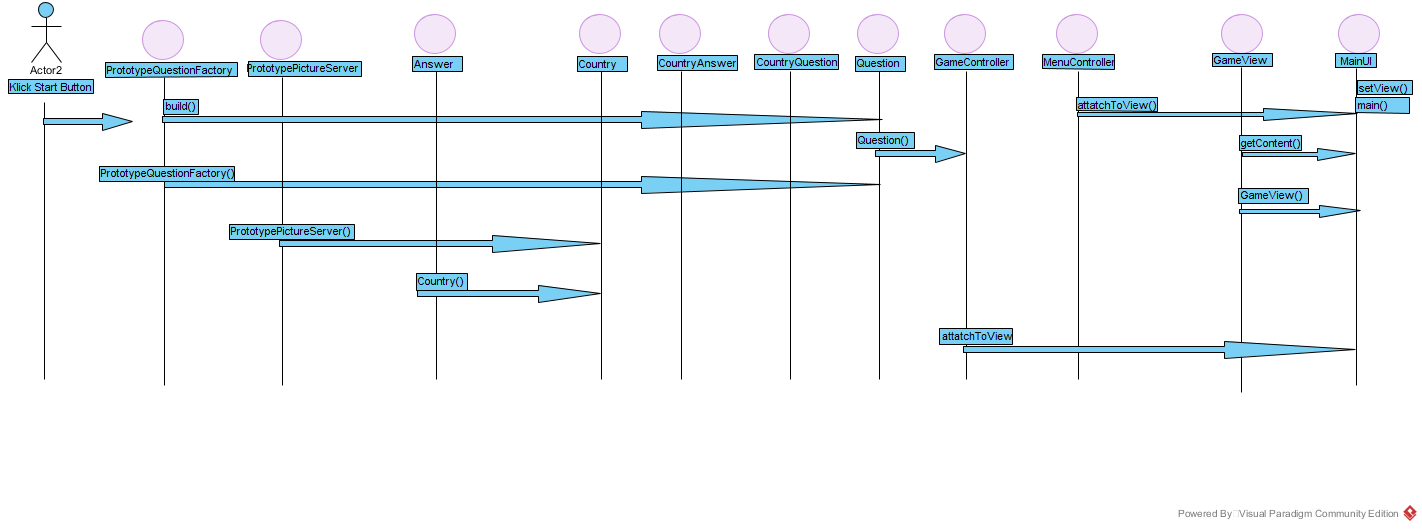
**Country**

Diese Klasse stellt ein Land und die dazugehörigen Informationen zuständig.

**CountryQuestionFactory**

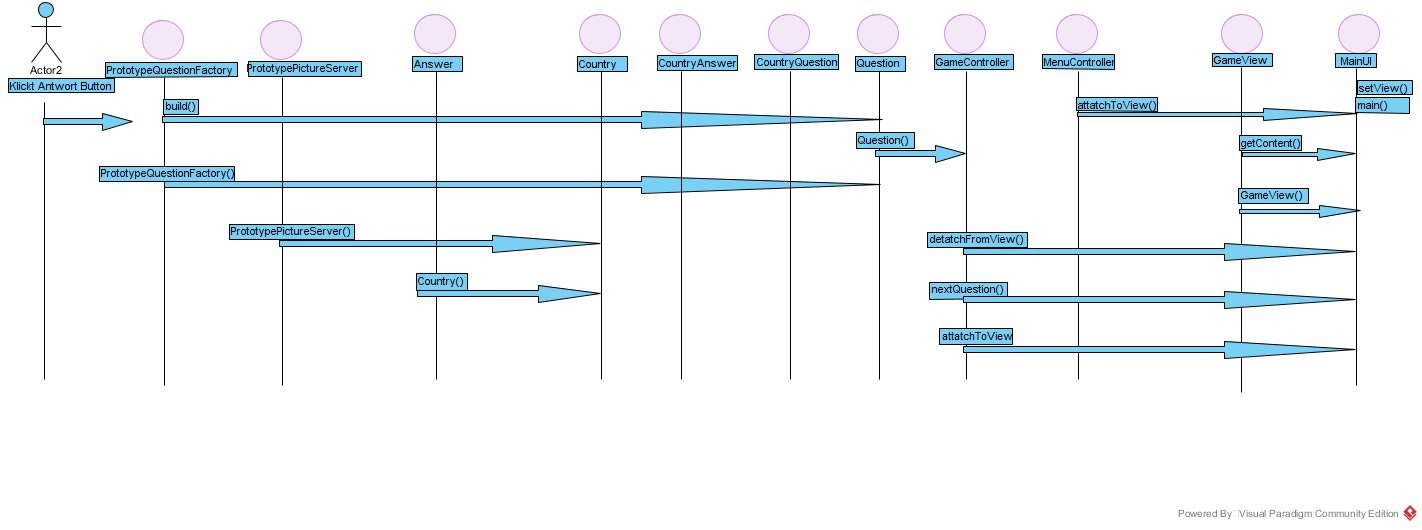
Klassen, die dieses Interface implementieren sind für das anfertigen von Klassen zuständig.

1. **Use-Case Realizations**
   1. **Auf Start Button drücken**
      1. *Erfolgreich auf den Start Button klicken*



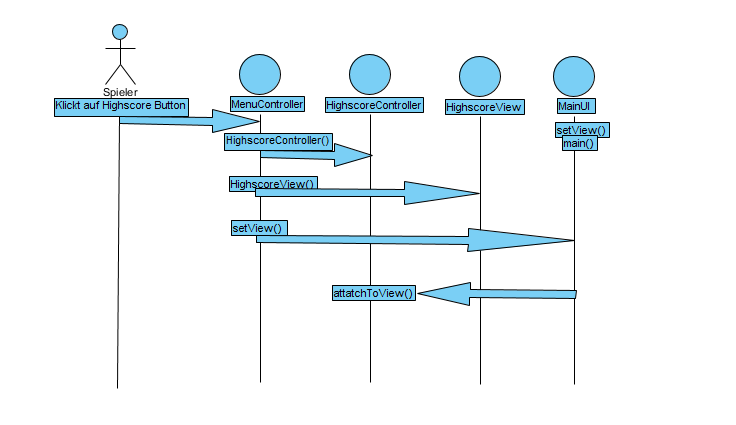
**5.2****Auf Antwort Button klicken**

*5.1.2* Erfolgreich auf einen Antwort Button klicken

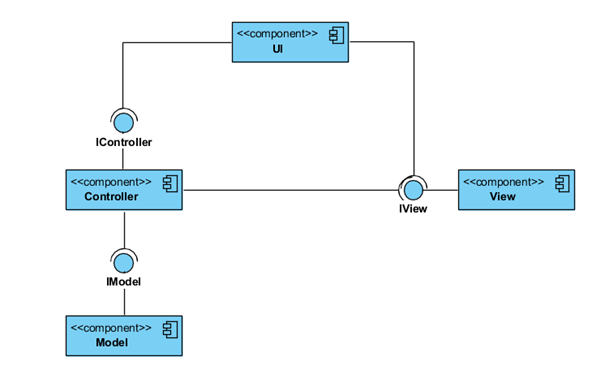


**5.3 Auf Highscore Button klicken**

*5.1.3 Erfolgreich auf HIghscore Button klicken*



1. **Komponenten Sicht**
   1. **Komponentendiagramm**



* 1. **Kurzbeschreibung der Komponenten**

UI

Diese Komponente ist für den Start des Spiels verantwortlich. Des Weiteren ist die Aufgabe dieser Komponente, sämtliche Spielinformationen in Form einer grafischen Benutzeroberfläche darzustellen.

View

Die View-Komponente ist für die Erzeugung der UI-Elemente verantwortlich, aus denen die Quiz Ansicht besteht. Hierbei wird unter anderem das Bild für die Flagge erzeugt, sowie die vier Antwortmöglichkeiten. Die Komponente stellt das Interface IView zur Steuerung sowie der grafischen Darstellung der einzelnen Elemente zur Verfügung.

Controller

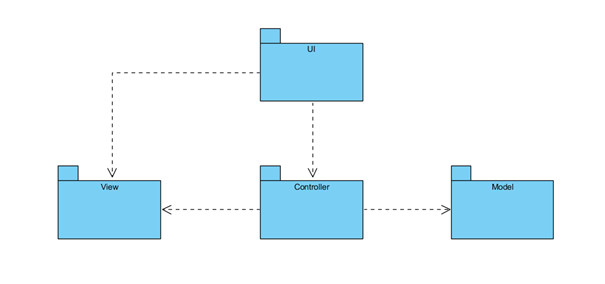
Diese Komponente erzeugt anhand des angegebenen Schwierigkeitsgrades zufällige Länder Fragen und lädt das dazugehörige Bild. Der Controller ist verantwortlich für das Laden einer neuen Frage. Sobald die Frage beantwortet wurde, wird die Spielansicht mit neuen Daten (Frage und Antworten) gefüllt.

Die Controller-Komponente stellt über das Interface IController die Spieledaten dem UI zur Verfügung.

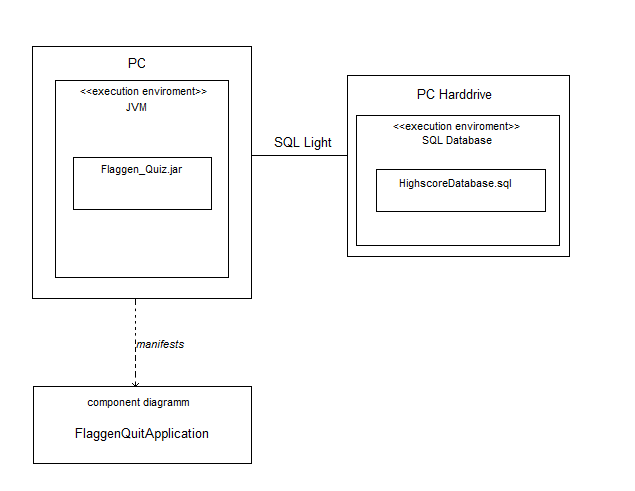
Model

Die Model-Komponente ist unter anderem für die Erzeugung der Antworten verantwortlich. In dieser Komponente wird die ID der gewählten Antwort mit der ID der Frage verglichen, stimmen diese überein ist die Antwort korrekt. Die richtige Antwort wird unter die weiteren (falschen) Antworten gemischt und so dem Nutzer zur Beantwortung der Frage bereitgestellt.

1. **Paket Sicht**

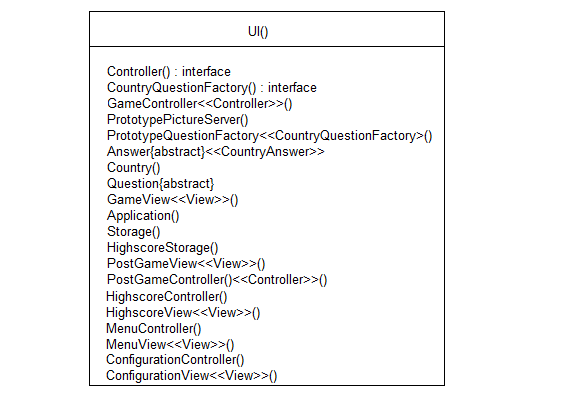
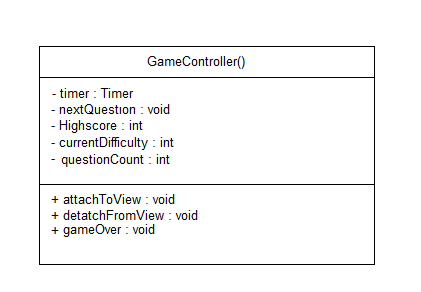


1. **Deployment View** 
   1. **Deployment Diagram**



* 1. **Notiz**

Highscores werden auf der SQLite Datenbank Lokal gespeichert.

1. **Design signifikanter Klassen**
   1. **Klassen Spezifikation**Dieses Diagramm einer Klasse UI enthält alle Klassen die wichtig für die Funktionsweise der Software sind.
   2. **Überblick über alle Klassen**
   3. **Signifikanteste Klasse**

-timer ist eine variable welche eine Referenz zu einem Timer Objekt enthält

-highscore enthält den Score den der Spieler momentan hat.

-currentDifficulty definiert den jetzigen Schwierigkeitsgrad des Spiels.

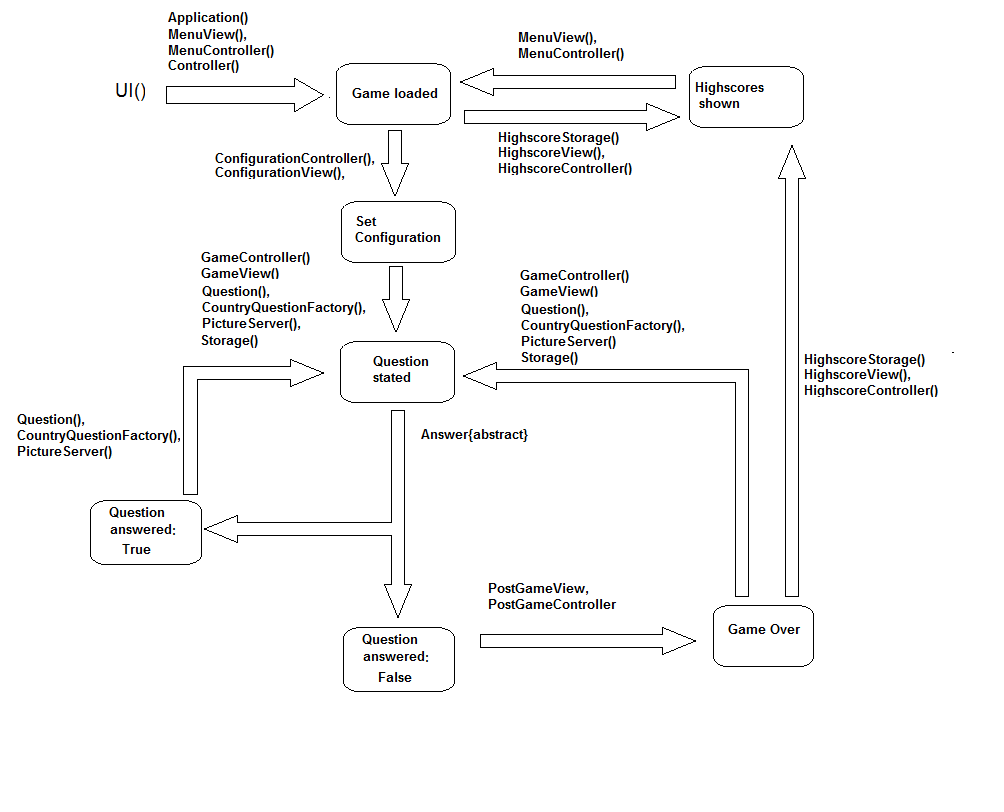
-questionCount speichert die Anzahl der bereits beantworteten Fragen ab.

-nextQuestion methode zum wechseln zur nächsten Frage.

-attachToView übernimmt die Kontrolle einer view.

-detatchFromView entfernt den Einfluss dieses Controllers von einer View.

-gameOver Methode wird aufgerufen sobald der Spieler verliert.

* 1. **Zustandsdiagramm UI()**
  2. **Aktivitäten Diagramm**

