Pflichtenheft

Projekt: GeschichteApp

Auftraggeber: Meinhard Mair

Meinhard.Mair@schule.suedtirol.it

<Telefon / Fax>

Meinhard Mair

Auftragnehmer: Daniel Nagler, Silas Keim, Sebastian Hofer

danielnagler1507@gmail.com

silaskeim02@gmail.com

sebi.hofer2001@gmail.com

342 366 7854, 333 423 638, 340 983 3307

Daniel Nagler, Silas Keim, Sebastian Hofer

1. Zielbestimmung

Das Projekt “GeschichteApp” soll eine neue Lernmethode für Schüler sein. Der Lehrer kann Fragen für die Schüler hochladen, die Schüler können als Vorbereitung für den Test mit dem Programm die Fragen beantworten. Die Fragen der einzelnen Themenbereiche werden im laufe des Jahres gesammelt, so erhält man eine große Datenbank über welche man dann jederzeit Fragen zu allen Themenbereichen aufrufen kann.

- Lehrer: Fragen und Antworten zur Datenbank hinzufügen
- Schüler: Quiz zu einem Themenbereich starten und “spielen”

Mit dem Projekt wollen wir Lehrer und Schüler ansprechen die Umwelt zu schonen und weniger Papier zu verschwenden.

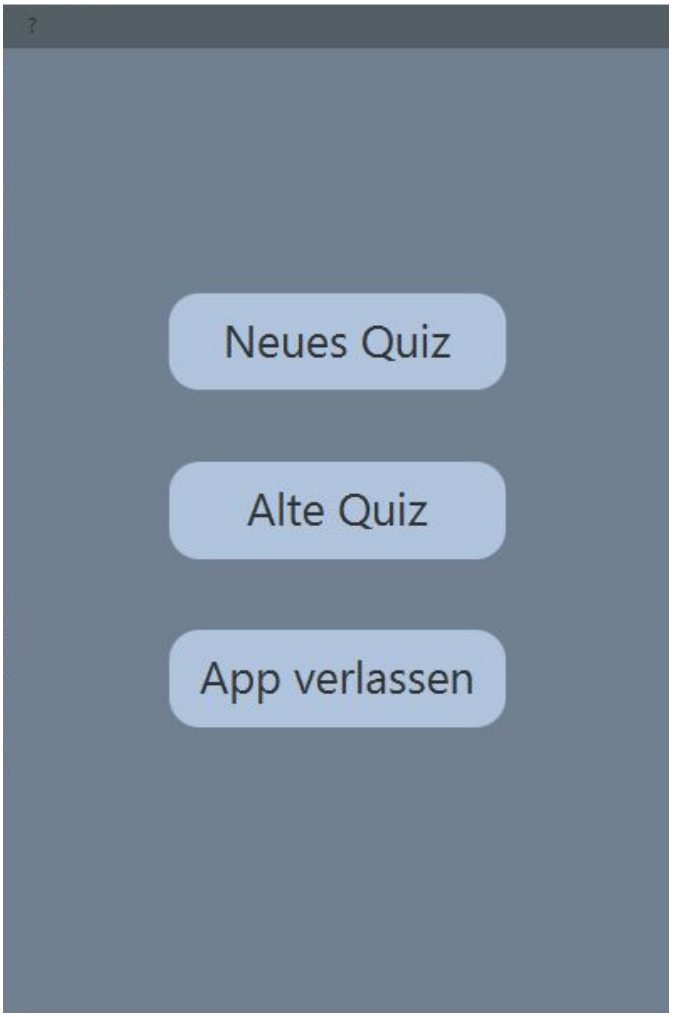
2. Produkteinsatz

Es gibt verschiedene Lernmethoden, mit unserem Projekt wollen wir eine neue Methode erstellen mit der man auch etwas für unsere Umwelt machen kann. Weil alles mit dem Computer oder Handy (Handyfunktion wird erst später verfügbar sein) verschwendet man kein Papier, somit werden weniger Bäume gefällt um neues Papier herzustellen.

Das Programm wird nur für Windows 10 verfügbar sein (Android kommt später). Das Windowsprogramm wird ein JavaFX programm sein. Als Produkt wird eine Executable Datei geliefert, die Installation von Java auf dem Computer wird nicht notwendig sein.

2.1. Beschreibung des Problembereiches

Abbildung 2.1:



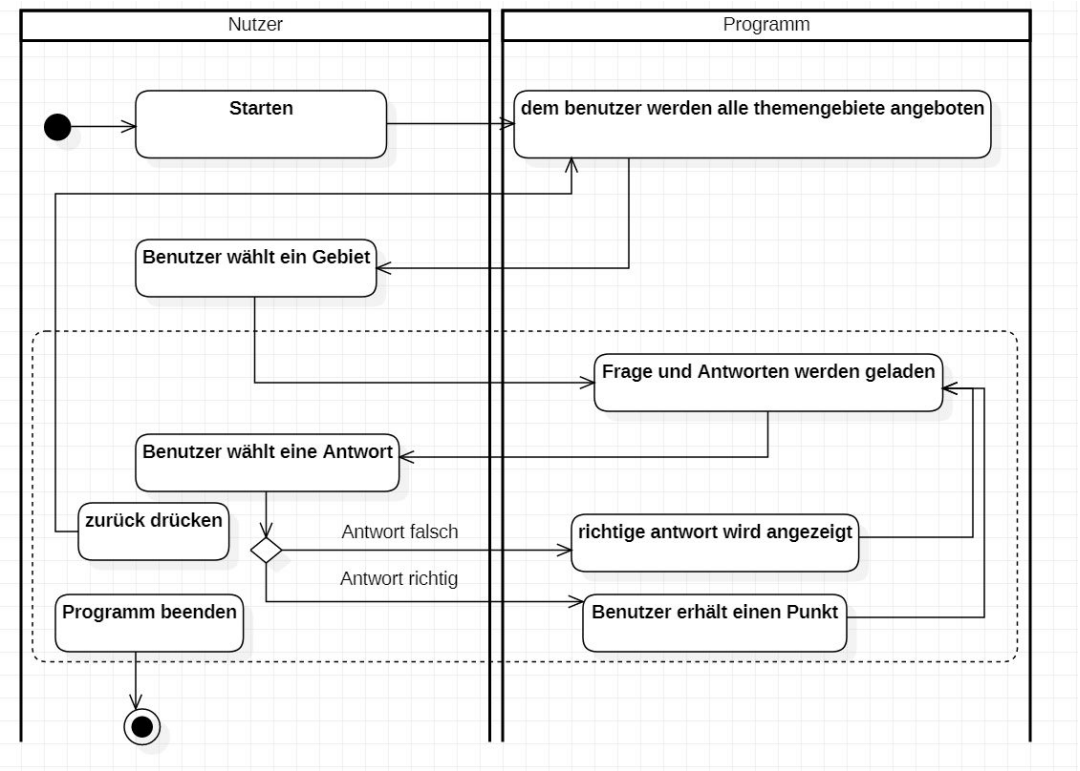
<Wichtig ist es auch noch, gemachte Annahmen sauber von den oben beschriebenen Fakten getrennt aufzulisten. Dies erleichtert eine spätere Fehlersuche, wenn das System nicht die Erwartungen erfüllt.>

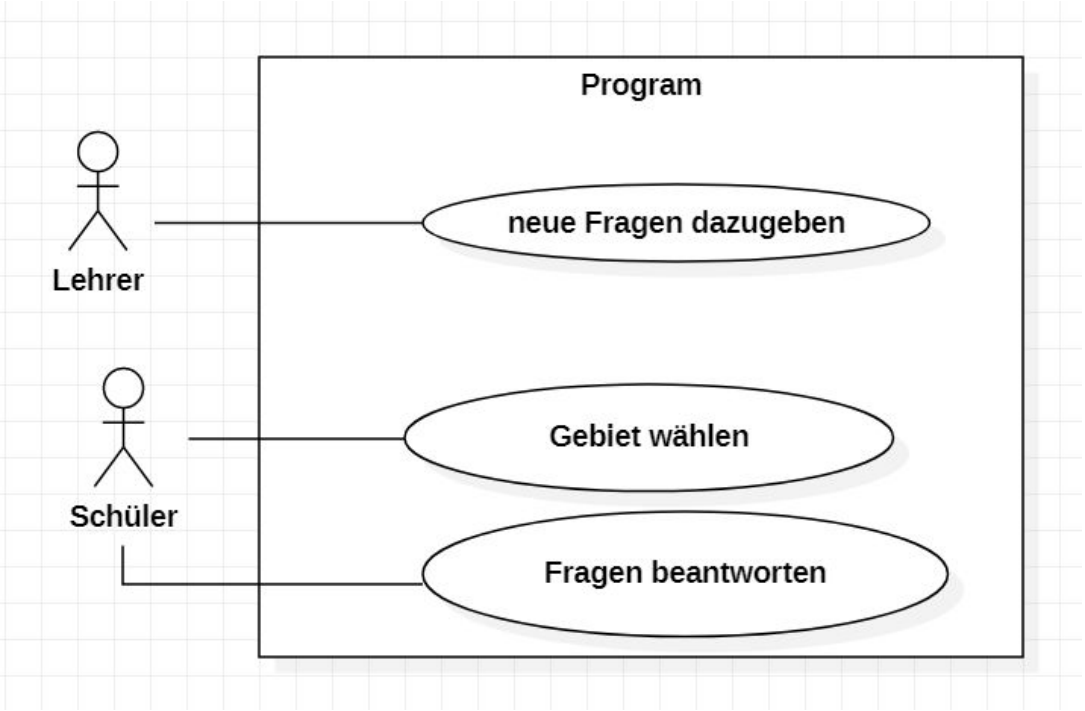
3. Produktfunktionen

Die Primäre funktion des Programms ist das ausgeben von Fragen und der dazugehörigen Antworten. Der benutzer soll die richtige Antwort auf die Fragen finden. Dies dient als

Vorbereitung für Tests. Der Benutzer kann zwischen verschiedenen Themenbereichen wählen und erhält ein Feedback zu seinen Antworten.

3.1. Use Case Diagramme





Beschreibung zu <Use Case-ID>: <Use Case-Name>

Dieser Abschnitt muß als Template für jeden Use Case aus dem vorigen Abschnitt wiederholt werden. Es ist sein Aufgabe die Beschreibung der Funktionalität mit Details anzureichern.

Aufgabe dieses Abschnittes ist die Erfassung der Hintergründe der Existenz des Use Cases.

Übergeordneter elementarer Geschäftsprozess:	*Prozess-ID: <elementarer Geschäftsprozess (verweist auf Abschnitt 2.5)>*
Ziel des Use Cases:	*Der Lehrer kann eine Frage mit der dazugehörigen antwort zu einem Themenbereich hinzufügen*
Umgebende Systemgrenze:	*<System, das betrachtet wird (Systemgrenze im Diagramm des vorigen Abschnittes)>*
Vorbedingung:	*Programm installiert und gestartet*
Nachbedingung bei erfolgreicher Ausführung:	*<Was muss sichergestellt werden für eine erfolgreiche Ausführung des Use Case>*
Beteiligte Nutzer:	*Lehrer*
Auslösendes Ereignis:	*Der Lehrer sieht die Notwendigkeit eine neue Frage hinzuzufügen*

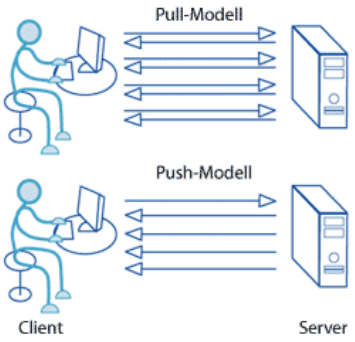
GUI für den Aufruf des Use Cases:

4. Reengineering (Ist-Zustand)

Zum aktuellen Zeitpunkt gibt es unsere App noch nicht. Somit können keine Klassendiagramme oder Sequenzdiagramme zum aktuellen Zustand gezeigt werden.

5. Grobentwurf (Soll-Zustand)

Die genaue Einteilung der Klassen ist noch nicht definiert. Bei den ersten Versionen des Programms werden keine Netzwerkknoten benötigt, weil man die Fragen mit Antworten als Client selber importieren muss. Später wenn die möglichkeit besteht wird höchstwahrscheinlich einer der beiden Server/Client Architekturen verwende und in das bestehende Programm eingebaut.



Pull-Modell:
Bei dieser Server/Client Architektur wird zum Beispiel eine Frage von Server zum Client geschickt, danach wird die eingegebene

Antwort des Clients dem Server geschickt. Dieses modell hat den Vorteil dass der Server alles weiß (z.B: ob der Client eine falsche Antwort eintippt). Aber es wird von der Netzwerkgeschwindigkeit eingeschränkt weil jede Information muss vom Server heruntergeladen werden.

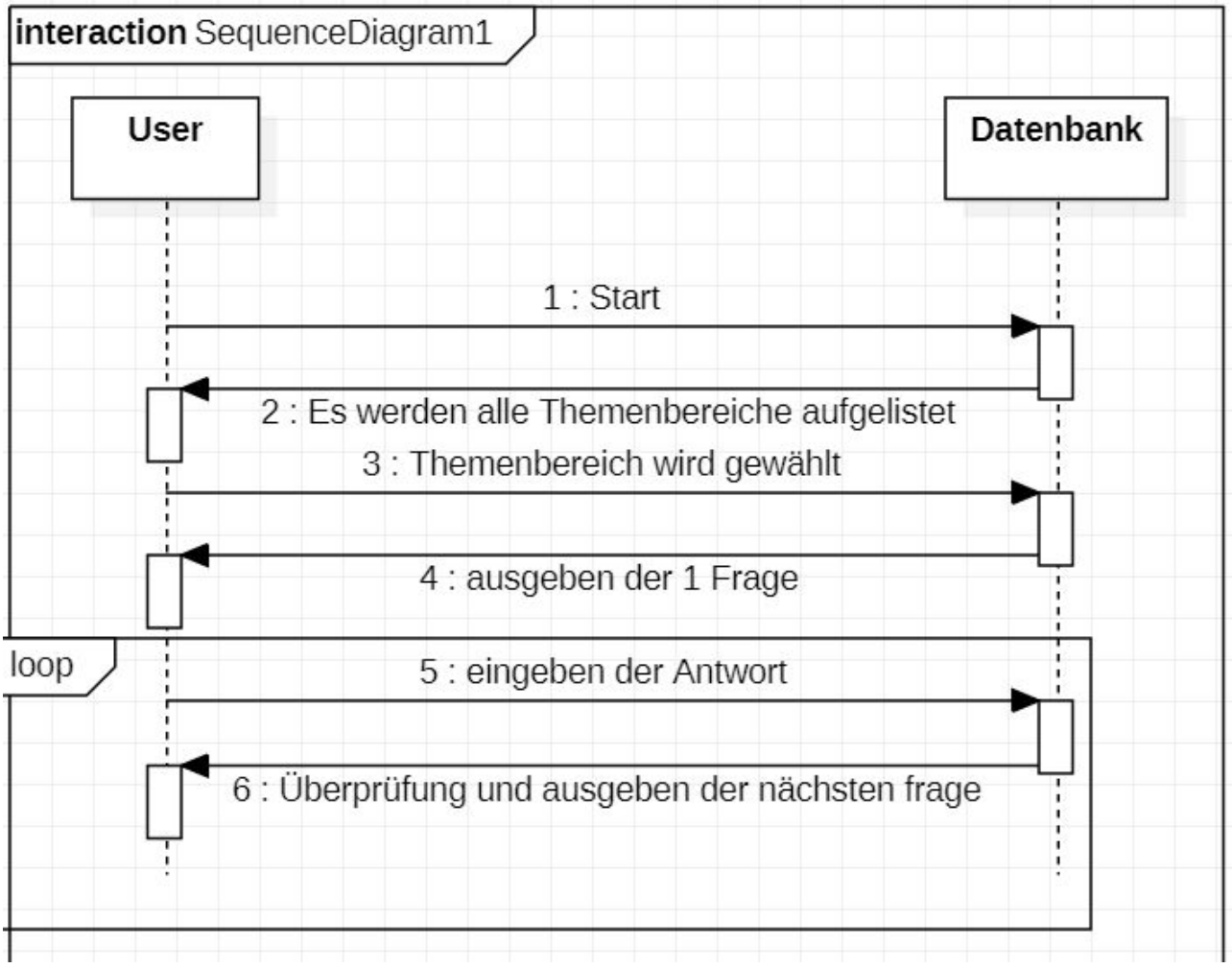
Pusch-Modell:
Bei dieser Server/Client Architektur wird zum Beispiel dem Server eine Nummer einer Frage geschickt und er schickt die ganzen Informationen(Frange, Antwort, ...) zurück. Bei dieser Methode müsste man am ende des Quizzes dem Server sagen wie viele Fragen man richtig bzw. falsch beantwortet hat. Aber sie hat den Vorteil sobald eine Frage geladen ist ist sie lokal gespeichert und kann sehr schnell abgerufen werden.

5.1. Klasseninteraktion

Die Interaktion zwischen den Klassen während der Ausführung eines Use Case wird durch Sequenzdiagramme beschreiben.

Klasseninteraktion bei Ausführung von

<Use Case-ID>*: *<Use Case-Name>



Zustandsdiagramm für Klassen mit signifikanten Zustandswechsel

.

<<Durch eigenes Zustandsdiagramm ersetzen>>

Durch die Verwendung eines graphischen Modells (UML Klassendiagramm) sollen die Zusammenhänge zwischen den Fachbegriffen präzisiert und übersichtlich dargestellt werden.

<Einleitende Worte können ergänzt werden durch Begründungen, warum an bestimmten Stellen eine bestimmte Art der Modellierung gewählt wurde.>

Abbildung 2.2: Klassendiagramm für den Problembereich <Problembereich>

6. Qualitätsmerkmale

Ø Typ USE: ***Benutzbarkeit Anforderung***

Das Programm soll sehr ansprechend auf den Benutzer wirken. Es sollte ein einfaches Design verwenden in welchem sich der Benutzer leicht zurechtfindet.

Ø Typ EFFIZIENZ: ***Effizienzanforderung***

Das Programm muss weder besonders schnell noch besonders effizient arbeiten. Es werden nur Fragen ausgegeben wo es nicht auf millisekunden ankommt.

Ø Typ PFLEGE: Wartbarkeits- und Portierbarkeitsanforderung

Das Produkt soll so programmiert werden dass jederzeit neue Themenbereiche mit ihren Fragen und Antworten hinzugefügt werden können. Eventuell ist eine Handy App geplant.

Ø Typ SICHER: Sicherheitsanforderung

Das Programm enthält keine wirklich wichtigen Daten welche vor dritten geschützt werden müssten. Auch ein Ausfall des Systems führt weder zu wirtschaftlichen Schäden noch werden Personen zu schaden kommen.

Ø Typ LEGAL: Gesetzliche Anforderung

Es gibt keine besonderen gesetzlichen Anforderungen an das Programm.

7. Umsetzung Details

Die App wird mithilfe von JavaFX programmiert. Verwendet werden dabei die Standardbibliotheken von JavaFX. Für die App benötigt man Windows 10 und mindestens einen freien Speicherplatz von ca 10-20MB. Da die App nicht grafisch aufwendig ist benötigt man nicht eine gute Grafikkarte oder einen guten Prozessor

8. Projektplan

Die Fertigstellung der App wird ca. 2-3 Wochen dauern. Es werden keine Kosten für die App benötigt.