Teachify - Lernspiele für Kinder

Dokumentation, Spezifikation, Konstruktion

Angelina Scheler, Bastian Kusserow, Christian Pfeiffer,
Christian Pöhlmann, Johannes Franz, Marcel Hagmann, Maximilian Sonntag,
Normen Krug, Patrick Niepel, Phillipp Dümleim, Carl Phillipp Knoblauch

09.07.2018
Vorgelegt bei Prof. Dr. Sven Rill

Inhaltsverzeichnis

1 Schüler UI, Anbindung an die Schnittstelle und entwicklung Mathe Piano Spiel

Christian Pfeiffer, Normen Krug & Johannes Franz

1.1 Einleitung

In diesem Abschnitt werden die Ziele und die Motivation des Projektes definiert. Dabei werden unter anderem die Erwartungen an das Projekt genannt.

1.1.1 Motivation

Die Hauptmotitvation des Projektes war das Lernen und Einarbeiten in neue Apple Frameworks und Erfahrungen sammeln in der Zusammenarbeit mit anderen Teams. Durch den Aufbau der Studienarbeit war es zwingend notwendig, sich mit anderen Teams zu verständigen und auszutauschen.

1.1.2 Ziele

Als Ziele der Studienarbeit wurden folgende Punkte definiert:

- Kinderfreundliches Design und Layout
- Erstellen eines Mathelernspieles
- Aufgaben die von Lehrern erstellt werden anzeigen und in ein spielbare Form überführen
- Die von Schüler beantworteten an den Lehrer weiterleiten
- Den Schülern die Möglichkeit bieten die Spiele im Endlos Modus, unabhängig der von Lehrer zugewiesenen Aufgaben, zu spielen
- Lernen eines neuen Apple Frameworks (**SpriteKit**)
- Erfahrung sammeln in Zusammenarbeit mit anderen Teams

1.2 Spezifikation

Als Strategie für die Umsetzung des Projektes wurde das Prinzip "Funktionalität vor UI-Design"gewählt.

1.2.1 Mathe Piano Spiel

Beim Spiel war es wichtig, möglichst schnell einen funktionsfähigen Prototypen zu entwickeln. Dieser wurde im Verlauf des Projektes nach und nach immer weiter verbessert.

1.2.1.1 Game Engine

Als Game Engine wurde die von Apple entwickelte Engine namens SpriteKit verwenden. Diese Engine hat einige Vorteile gegenüber anderen Spielengines:

- Gute Dokumentation
- Wiederverwendungen bereits gelernter Paradigmen
- Einfache Integration Möglichkeit in die App
- Einfache Anbindung an andere iOS API‘s
- Swift als Programmiersprache
- Schnelles Entwickeln und Testen von Funktionen durch Swift Playgrounds

1.2.1.2 Herausforderungen

Nach der ausgiebigen Einarbeitungen in das *SpriteKit Framework* haben sich einige Hürden ergeben. Das verwenden von dynamischen Buttons ist nicht trivial, weil es keine Buttons per Default gibt. Deshalb muss eine eigene Button Klasse implementiert und mit der gewünschten Funktionalität erweitert werden. Des weiteren war es schwierig den Code sinnvoll zu strukturieren, aufgrund der durch Spiel vorgegebenen Skript artigen Programmierung.

1.2.1.3 Testen des Spieles

Um das Spiel sinnvoll und schon währendes Entwicklungsprozesses testen zu können, musste ein Generator entwickelt werden der zufällige Aufgaben generiert.

1.2.1.4 Anbindungen an interne Schnittstellen

Von Anfang an musste darauf geachtet werden das, dass Spiel an die interne Schnittstelle angebunden werden muss. Da die Schnittstelle nicht von Beginn an verfügbar ist, muss eine temporäre Datenstruktur implementiert werden. Diese muss einfach austauschbar und erweiterbar sein.

1.2.1.5 User Interface

Bei der Gestaltung des User Interface muss explizit darauf geachtete werden, dass die Software primär von Kinder bedient werden wird. Das bedeute das die Größe der Bedienelemente deutlich größer ausfallen muss als bei herkömmlichen Applikationen.

1.2.2 User Interface

1.3 Implementierung Phase

An dieser Stelle wird die Implementierung der Aufgaben beschrieben. Dabei wird auf die Schnittstellenanbindung, das Mathe Piano Spiel sowie das User Interface eingegangen.

1.3.1 Mathe Piano Spiel

1.3.2 User Interface

1.4 Fazit

Dieses Projekt war das erste große neu begonnene Projekt der Studiengruppe Mobile Computing 6. Semester mit entsprechend vielen Contributern. Dies stellte das ganze Team immer wieder vor Herausforderungen. Der Umgang mit Git (Teachify Projekt Link) brachte zugleich viele Vorteile aber auch Herausforderungen.
Durch die unterschiedlichen Herangehensweisen (Design vs. Funktion) und den damit weitgehend einhergehenden Verzicht auf einen Prototypen lenkte das Projekt gegen Ende des Projektzeitraums auf einen “Big-Bang“ Ansatz. Positiv zu erwähnen war das Zusammenwachsen des Teams und die Zusammenarbeit untereinander. So hatten die meisten Teams eine Domäne in die sie sich eingearbeitet hatten und mussten für die Schnittpunkte ihrer Gebiete zusammenarbeiten.

2 Schnittstelle iCloud

Patrick Niepel, Marcel Hagmann, Carl Philipp Knoblauch

2.1 Einleitung

In diesem Abschnitt wird die Schnittstelle mit iCloud beschrieben. Dabei wird erklärt wie die Architektur aufgebaut ist, wie mit der Schnittstelle kommuniziert wird und welche Probleme aufgetreten sind.

2.2 Warum iCloud/CloudKit?

Wenn man bei der Entwicklung einer iOS App auf Cloud Services zurückgreifen will, bietet sich natürlich das Apple eigene CloudKit für iCloud an. Es ergab sich dadurch auch die Möglichkeit eine neues Framework kennenzulernen, da wir zuvor noch nicht mit CloudKit gearbeitet hatten. Über die Cloud-Schnittstelle sollen zwischen Lehrer und Schüler alle Aufgaben geteilt werden. Der Lehrer kann seine Aufgaben/Spiele in iCloud laden und diese mit seinen Schülern teilen. Die Schüler sollen dann diese Aufgaben/Spiele erledigen und ihre Lösungen wieder in iCloud laden. Dadurch wird dem Lehrer wiederum ermöglicht sich einen Überblick über die Lösungen seiner Klasse zu machen. Mit CloudKit konnten wir diese Ziele alle umsetzten.

2.3 Architektur

Auch TeachKit ist strikt nach der Model-View-Controller - Architektur aufgebaut. Hierbei erben alle Models, die in der Cloud gespeichert werden, von ihrer Superklasse TKCloudObject. Die Controller die diese Models verwalten, sind mit Hilfe eines generischen Controllers TKGenericCloudController implementiert worden. Alle Cloud-Models und Controller bedienen sich von verschiedenen Enumerations, die die Funktionalität übersichtlicher gestalten.

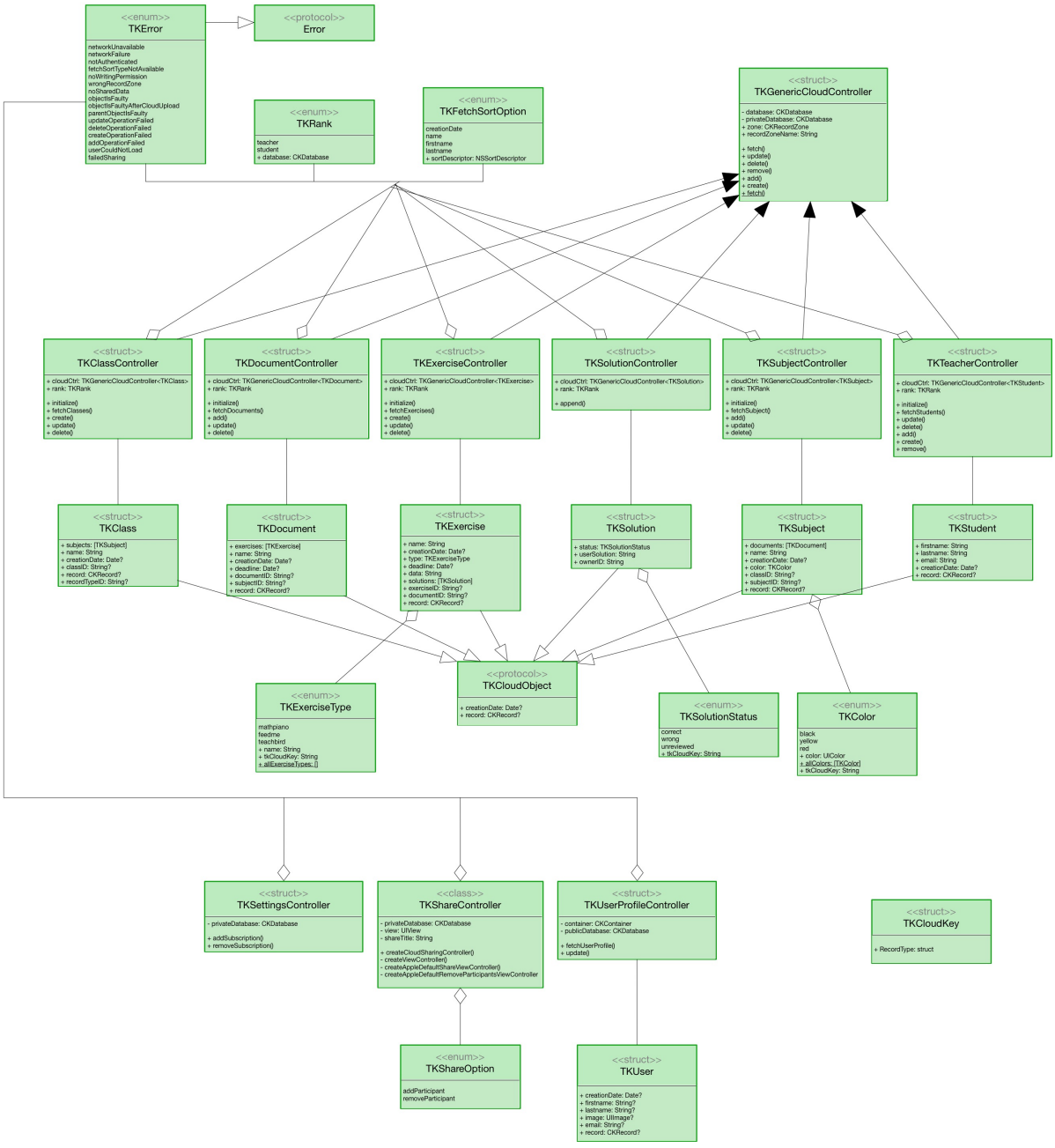

Abbildung 2.1: TeachKit Schnittstellen Klassendiagramm

2.4 Features

2.4.1 Upload/Download/Delete/Fetch/Update

Für jeden Datentyp in TeachKit, gibt es einen Controller der für die Operationen auf den Datentyp verantwortlich ist. Somit gibt es folgende Controller für die Datentypen:

- TKClassController
- TKSubjectController

2.4.2 User Profile

2.4.3 Sharing

2.4.4 Push/Subscriptions

2.4.5 TKError

2.5 Aufgetretene Probleme

2.5.1 Subscription

2.5.2 Sharing

2.5.3 TKSolution

Abbildungsverzeichnis