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 des Mathe Piano Spiels

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 Einarbeitung 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 skriptartigen 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. Dieser befindet sich in der *RandomQuestionGenerator.swift* Klasse.

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, die von einem anderen Team entwickelt wurde. Da die Schnittstelle nicht von Beginn an verfügbar ist, muss eine temporäre Datenstruktur implementiert werden. Diese soll einfach austauschbar und erweiterbar sein.

1.2.1.5 User Interface

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

1.2.2 User Interface

1.3 Implementierungsphase

An dieser Stelle wird die Implementierung der Aufgaben beschrieben. Dabei wird auf die Schnittstellenanbindung, das Mathe Piano Spiels 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 mit sich. 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 bei der Überschneidung 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.

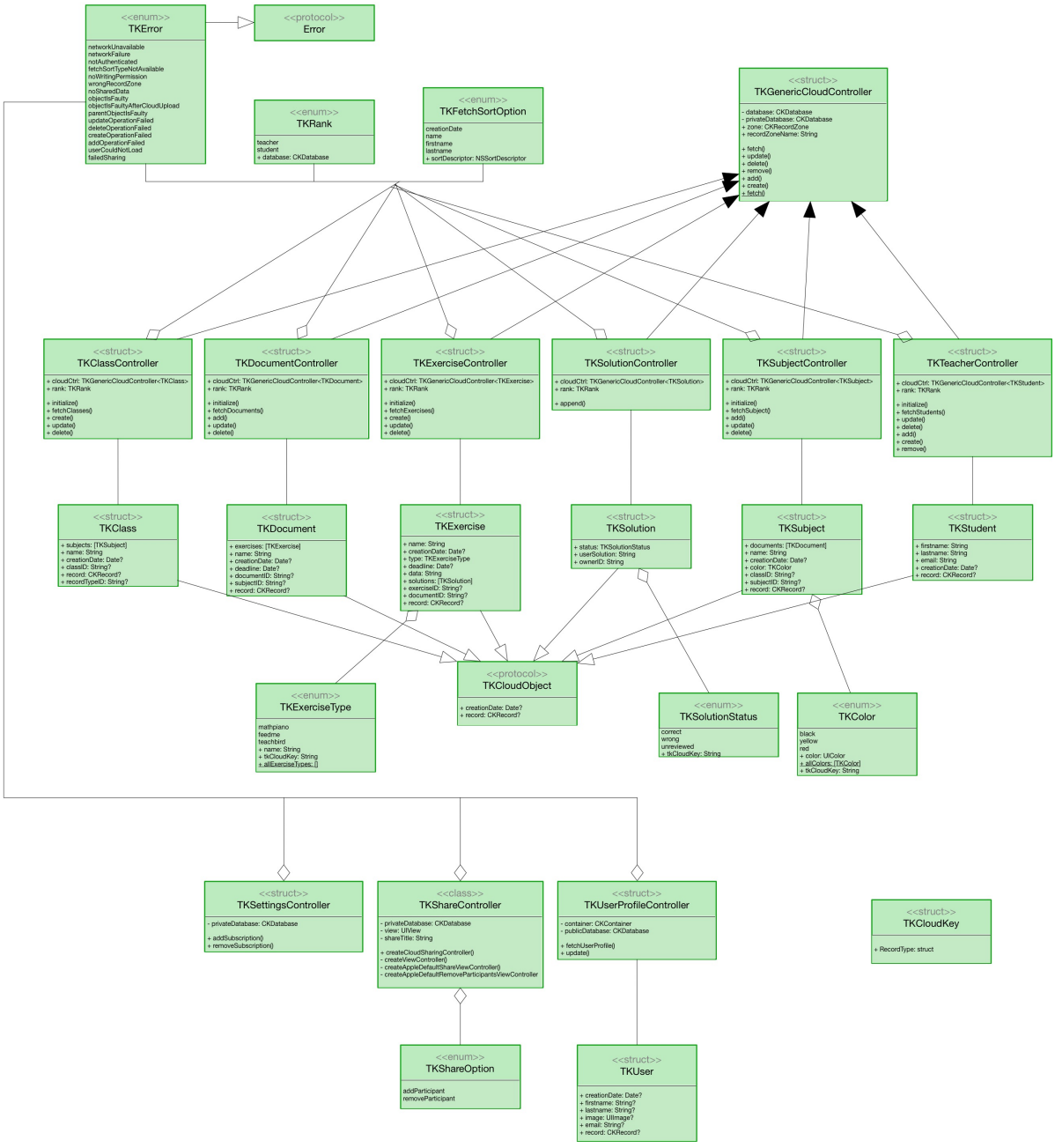

Abbildung 2.1: TeachKit Schnittstellen Klassendiagramm

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
- TKDocumentController
- TKExerciseController
- TKSolutionController

Alle Controller sind ähnlich aufgebaut. Der Grund weshalb der Controller über die Methode initialize(...) funktionsfähig gemacht werden muss ist der, dass der Student auf die Shared-Database zugreift und diese zuerst gefetched werden muss.

Um doppelten Code zu vermeiden, arbeiten alle der oben genannten Controller mit dem TKGenericCloudController, der die Grundfunktionen übernimmt und in den jeweiligen Controllern dann spezialisiert werden.

2.4.2 User Profile

Zu jedem Nutzer kann der Vorname, Nachname und ein Profilbild gespeichert werden. Der TKUserProfileController der für die Nutzerverwaltung verantwortlich ist, arbeitet auf der Public-Database. Das bedeutet, dass alle Nutzer diese Informationen sehen können. Die aktuelle implementation erlaubt nur den download der Daten für den derzeit eingeloggten Nutzer. Diese kann erweitert werden, hätte aber momentan keine Verwendung gefunden.

2.4.3 Sharing

Eines der wichtigsten Features unserer App ist das Teilen von Daten zwischen Teacher und Student. Nach dem Teilen gemeinsamer Daten haben beide Zugriff auf das Subject und alles was darunter angelegt ist.

CloudKit arbeitet mit drei verschiedenen Datenbanken.

Private-Database
Der aktuell angemeldete Nutzer ist der Inhaber der Daten, nur dieser hat Zugriff auf diese Datenbank und hat das Recht zu lesen und zu schreiben.
Shared-Database
Der aktuelle Nutzer ist nicht der Besitzer der geteilten Daten, und hat die beim Teilen zugewiesenen lese und/oder schreib Rechte.
Public-Database
Der aktuelle Nutzer ist nicht der Besitzer der geteilten Daten, und hat die beim Teilen Jeder App Nutzer hat lese Recht auf diese Daten, auch ohne aktiven iCloud Account.

Legt der Teacher seine Daten an, befinden diese sich in seiner private-Database. Nachdem dieser das Subject geteilt hat, befindet sich dies immer noch in der private-Database. Bei jedem Student der nun berechtigt ist, das geteilte Subject einzusehen, wird eine Referenz in der shared-Database gespeichert. Diese Referenz zeigt auf die Daten des Teachers in der private-Database.

Der Teacher teilt mit dem Student ein Subject, damit der Student die neue Arbeitsblätter einsehen kann. Mit dem Student wird der Zugriff auf Class nicht geteilt, weil er diese Information nicht benötigt.

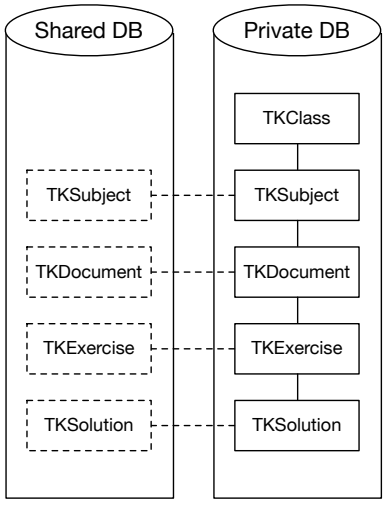

Abbildung 2.2: hared DB / Private DB

Das Teilen findet über den TKShareController statt. Mit der Methode createCloudSharingController(...), wird der ViewController der für das Teilen verantwortlich ist erstellt und kann angezeigt werden. Bei der Verwendung des Controllers müssen keine weiteren Bedingungen beachtet werden, der Controller kümmert sich um alles was zum Teilen benötigt wird.

2.4.4 Push/Subscriptions

Eine Besonderheit des CloudKits ist die einfache Implementierung von Push Benachrichtigungen bei Datenbankänderungen. PushNotifications in einer App zu implementieren, die über einen eigenen Server laufen, setzen PHP Kenntnisse voraus und können einiges an Zeit in Anspruch nehmen. CloudKit Subscriptions sind jedoch einfacher zu implementieren und können auf Insert, Update, Delete und Create in der Datenbank reagieren.

Der Verwendungszweck der Subscriptions war dafür gedacht, dass der Student über Änderungen zu seinem Fach informiert wird. Dies könnte beispielsweise der Fall sein, wenn der Professor ein neues Arbeitsblatt seinem Fach hinzufügt.

Wir implementierten die CloudKit Subscriptions, die darauf reagiert, wenn der Teacher ein neues Document einem Subject hinzufügt. Die ersten Tests auf zwei unterschiedlichen Geräten mit der gleichen Apple ID waren erfolgreich. Beim Testen über zwei unterschiedliche Apple ID's, bei dem der Teacher ein Subject mit dem Student teilt, kamen wir allerdings an unsere Grenzen. Der Student wurde beim Hinzufügen eines neuen Objekts nicht benachrichtigt. In der CloudKit API wurden wir dann auf die folgende Anmerkung aufmerksam, die besagt, dass Subscriptions auf der Shared-Database nicht unterstützt werden.

2.4.5 TKError

Bei den meisten Aktionen die innerhalb des TeachKits Fehler auslösen können, werden Fehler vom Datentyp TKError erstellt. Jeder Fehler beinhaltet eine genauere Beschreibung des aufgetretene Problems. Folgende Fehler existieren:

networkUnavailable Es besteht keine Internetverbindung.

networkFailure Es besteht eine Internetverbindung, es konnte allerdings keine Verbindung zur Cloud hergestellt werden.

wrongRecordZone Die Operation wird auf der falschen RecordZone ausgeführt oder existiert nicht. An Schnittstelle wenden.

failedSharing Das Objekt konnte nicht geteilt werden.

parentObjectIsFaulty Operation konnte nicht ausgeführt werden, da das Parent-Objekt Fehlerhaft ist.

objectIsFaulty Operation konnte nicht ausgeführt werden, da das Objekt Fehlerhaft ist.

objectIsFaultyAfterCloudUpload Auf dem Objekt wurde eine Operation in der Cloud ausgeführt. Das von der Cloud erhaltene Objekt ist inkonsistent.

userCouldNotLoad Beim Zugriff auf die Nutzer Informationen ist ein Fehler unterlaufen.

updateOperationFailed Der Datensatz konnte nicht geupdated werden.

deleteOperationFailed Der Datensatz konnte nicht gelöscht werden.

createOperationFailed Der Datensatz konnte nicht erstellt werden.

addOperationFailed Der Datensatz konnte nicht hinzugefügt werden.

fetchSortTypeNotAvailable Das Attribut nach dem sortiert werden soll existiert nicht.

noWritePermission Der Nutzer hat keine Berechtigung die Daten zu ändern.

noSharedData Die Operation konnte nicht ausgeführt werden, da noch keine Daten geteilt werden.

2.5 Aufgetretene Probleme

2.5.1 Subscription

Da auf das Problem mit den Subscriptions bereits im Abschnitt Push/Subscriptions eingegangen wurde, erwähnen wir an dieser Stelle nur noch einmal, das Subscriptions nicht auf der Shared-Database unterstützt werden.

2.5.2 Sharing

Während der Implementation der Sharing Funktion, hatten wir über einen längeren Zeitraum das Problem, dass das Record zwischen den Nutzern geteilt wurde. Der darunter hängende Baum war allerdings nicht beim Student einzusehen.

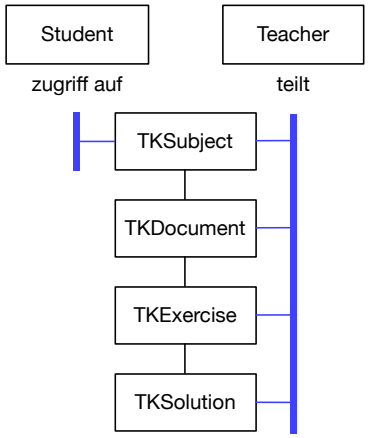

Abbildung 2.3: Zugriff Teacher/Student

Letztendlich stellte sich heraus, dass die Verbindung zwischen Parent- und dem Child-Record nicht hergestellt wurde. Dieses Problem konnte ganz einfach mit der Methode setParent(CKRecord?) gelöst werden, allerdings musste man dafür erst wissen, dass so eine Funktion überhaupt existiert.

2.5.3 TKSolution

Die Idee hinter TKSolution war die, dass ein Student seine Lösung darin erstellt und anschließend zu der TKExercise hinzufügt. Allerdings ist es dem Student nicht möglich, weitere Records in der geteilten Datenbank des Teachers zu erstellen und hinzuzufügen.

```
TKGenericCloudController-create-Error: Optional(<CKError 0x103781b20: "Permission Failure" (10/2007); server message = "CREATE operation not permitted"; uuid =
6A7512C1-4170-4D82-A2BF-23564DADBCA7; container ID = "iCloud.iosapps.hof-university.teachify">)
solution upload error: Optional(Teachify.TKError.objectIsFaultyAfterCloudUpload)
```

Abbildung 2.4: Error

Da diese Fehlermeldung auftritt obwohl wir die publicPermission des CKShare Objektes richtig gesetzt haben und immer noch kein Record erzeugen konnten, mussten wir eine andere Lösung dafür suchen. Wir entschlossen uns die Lösungen in ein Data-Objekt zu serialisieren und diese in TKExercise hinzuzufügen. Unseren neuen Lösungsansatz testen wir zuerst mit einem Attribut vom Datentyp String und nicht den eigentlich benötigten Datentyp Data. Beim ersten Upload in die iCloud werden die Records in der Cloud automatisch angelegt. Der erste Test hat geklappt und anschließend wollten wir unseren benötigten Daten vom Typ Data sichern. Da allerdings die angelegten Constraints nicht mehr für den neuen Datentyp gestimmt haben, konnten keine Records mehr hochgeladen werden. Die Constraints müssen zuerst im iCloud-Dashboard gelöscht und erneut hochgeladen werden.

Abbildungsverzeichnis