

Stundenplan–App für iOS

Dokumentation, Spezifikation, Konstruktion

Daniel Glaser, Stefan Scharrer

06.07.2017

Inhaltsverzeichnis

1	Calendar Interface	1
1.1	Einleitung	1
1.2	CalendarController	2
1.2.1	createCalendar() -> EKAuthorizationStatus	2
1.2.2	removeCalendar()	2
1.2.3	createAllEvents(lectures : [Lecture])	2
1.2.4	updateAllEvents(changes : [ChangedLecture])	3
1.2.5	removeAllEvents(lectures : [Lecture])	3
1.2.6	CalendarRoutine() -> Bool	3
1.3	CalendarInterface	4
1.3.1	createCalenderIfNeeded() -> Bool	4
1.3.2	removeCalendar() -> Bool	4
1.3.3	createEvent(p_event : EKEEvent, key : String, isChanges : Bool)	4
1.3.4	updateEvent(eventID : String, updatedEvent : EKEEvent, key : String, lectureToChange : Bool)	4
1.3.5	removeEvent(p_eventId: String, p_withNotes: Bool?=false) -> Bool	4
1.3.6	saveIDs	4
1.4	CalendarData	5

1 Calendar Interface

1.1 Einleitung

Im folgenden Diagramm ist die Architektur und Integration des Calendar Interface in die Stundenplan App dargestellt.

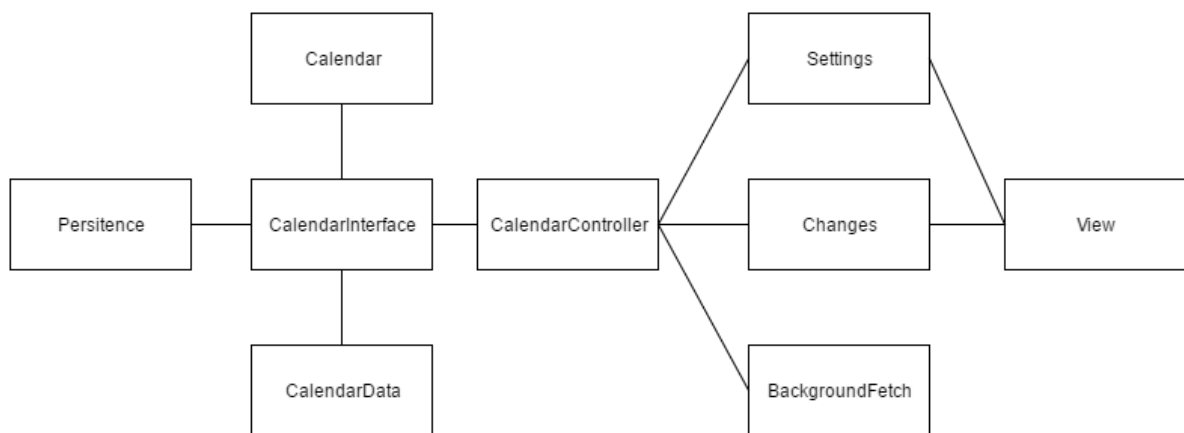


Abbildung 1.1: CalendarInterface Diagramm

Das Calendar Interface besteht aus folgenden drei Klassen:

- CalendarController
- CalendarInterface
- CalendarData

Im Folgenden wird auf die Klassen genauer eingegangen.

1.2 CalendarController

Der CalendarController ist der Controller für das Calendar Interface. Über ihm wird auf das Interface zugegriffen.

Im Folgenden wird auf die wichtigsten Methoden eingegangen:

- createCalendar() -> EKAuthorizationStatus
- removeCalendar()
- createAllEvents(lectures : [Lecture])
- updateAllEvents(changes : [ChangedLecture])
- removeAllEvents(lectures : [Lecture])
- CalendarRoutine() -> Bool

Vor dem Ausführen der Aufgabe der jeweiligen Methode wird überprüft, ob der Benutzer die Berechtigung auf den Kalender zuzugreifen, gewährt hat.

1.2.1 createCalendar() -> EKAuthorizationStatus

Erzeugt den Kalender falls die Berechtigung vorhanden ist. Falls er erzeugt wurde werden die Events mit createAllEvents in den Kalender geschrieben. Als Rückgabewert gibt er den Berechtigungsstatus zurück. Es gibt drei Rückgabewerte:

- authorized
Bedeutet das der Nutzer die Berechtigung erteilt hat und der Kalender angelegt wurde oder bereits angelegt war.
- notDetermined
Bedeutet das die Berechtigung gerade abgefragt wird.
- denied
Bedeutet das die Berechtigung verweigert wurden.

1.2.2 removeCalendar()

Löscht den Kalender und damit alle Einträge die darin vorhanden sind mit Hilfe des KalenderInterface.

1.2.3 createAllEvents(lectures : [Lecture])

Erzeugt aus den Vorlesungen die Events, die anschließend mit Hilfe des CalendarInterface in den Kalender geschrieben werden. Falls der Kalender noch nicht erzeugt wurde wird er angelegt. Die EventID's werden mittels der saveIDs-Methode des CalendarInterface persistent gespeichert.

1.2.4 **updateAllEvents(changes : [ChangedLecture])**

Ermittelt welche Art von Änderung vorliegt und passt mit Hilfe des CalendarInterface die entsprechend Events im Kalender an. Die EventID's werden mittels der saveIDs-Methode des CalendarInterface persistent gespeichert.

1.2.5 **removeAllEvents(lectures : [Lecture])**

Holt sich für die übergebenden Vorlesungen die EventID's und löscht die entsprechenden Events mit Hilfe des CalendarInterface. Die Änderungen an den EventID's werden mittels der saveIDs-Methode des CalendarInterface persistent gespeichert.

1.2.6 **CalendarRoutine() -> Bool**

Aktualisiert die Vorlesungen im Kalender. Dazu holt sie sich die abgewählten und neu ausgewählten Vorlesungen und übergibt sie der removeAllEvents() oder createAllEvents()-Methode. Der Rückgabewert gibt an, ob der Benutzer die Berechtigung erteilt hat, in den Kalender zu schreiben.

1.3 CalendarInterface

Dies ist die Klasse die direkt auf den Kalender zugreift.

Im Folgenden wird auf die wichtigsten Methoden eingegangen:

- `createCalendarIfNeeded()` -> Bool
- `removeCalendar()` -> Bool
- `createEvent(p_event : EKEvent, key : String, isChanges : Bool)`
- `updateEvent(eventID : String, updatedEvent : EKEvent, key : String, lectureToChange : Bool)`
- `removeEvent(p_eventId: String, p_withNotes: Bool?=false)` -> Bool
- `saveIDs()`

1.3.1 createCalendarIfNeeded() -> Bool

Erzeugt den Kalender falls er nicht bereits vorhanden ist. Gibt einen Boolean-Wert zurück ob der Kalender angelegt wurde.

1.3.2 removeCalendar() -> Bool

Löscht den Kalender und damit alle Einträge, die er beinhaltet falls dieser vorhanden ist. Gibt einen Boolean-Wert zurück ob das Löschen erfolgreich war.

1.3.3 createEvent(p_event : EKEvent, key : String, isChanges : Bool)

Schreibt das übergebende Event in den Kalender. Unterscheidet dabei, ob es sich um eine Änderung oder eine normale Vorlesung handelt und fügt die EventID dem entsprechenden Dictionary hinzu.

1.3.4 updateEvent(eventID : String, updatedEvent : EKEvent, key : String, lectureToChange : Bool)

Das zu der EventID dazugehörige Event, wird mit den entsprechend Werten des übergebenen Event angepasst. Durch den lectureToChange Boolean-Wert wird unterschieden, ob aus einer Vorlesung einer Änderung wird oder ob aus einer Änderung wieder eine Vorlesung wird. Dabei werden die EventsIDs in das entsprechende Dictionary verschoben.

1.3.5 removeEvent(p_eventId: String, p_withNotes: Bool?=false) -> Bool

Löscht das übergebene Event. Gibt einen Boolean-Wert zurück ob das Löschen erfolgreich war.

1.3.6 saveIDs()

Speichert die EventIDs der Vorlesungen und Änderungen.

1.4 CalendarData

Die CalendarData-Klasse dient dazu, die EventIDs der Vorlesungen und Änderungen getrennt in Dictionarys zu speichern. Die Dictionarys bestehen aus einer Zuordnung von einer ID einer Vorlesung zu mehreren EventIDs.