
# Stundenplan–App für iOS

# Dokumentation, Spezifikation, Konstruktion

Daniel Glaser, Stefan Scharrer

06.07.2017

# Inhaltsverzeichnis

# 1 Calendar Interface

## 1.1 Einleitung

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

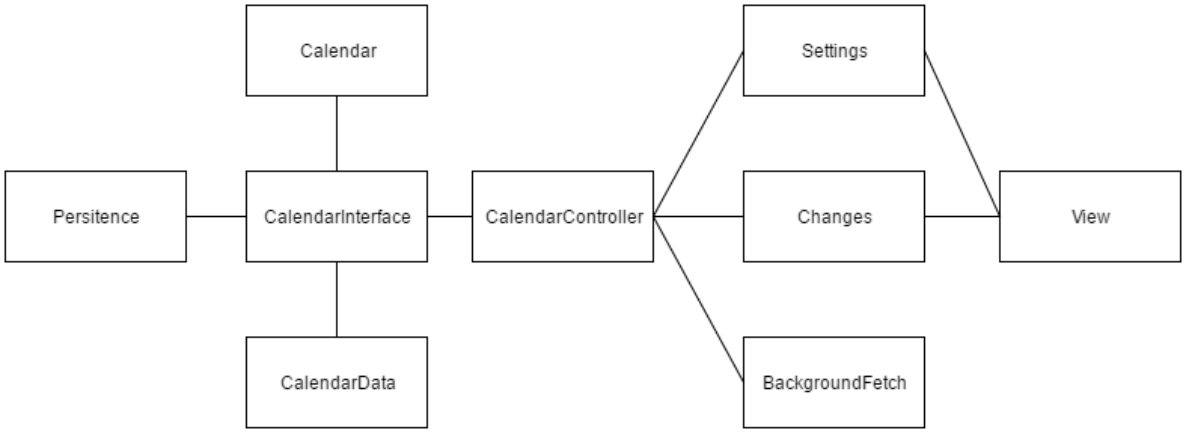


Abbildung 1.1: CalendarInterface Diagramm

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 ihn 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, dass der Nutzer die Berechtigung erteilt hat und der Kalender angelegt wurde oder bereits angelegt war.

- notDetermined

  Bedeutet, dass die Berechtigung gerade abgefragt wird.

- denied

  Bedeutet, dass die Berechtigung verweigert wurden.

### 1.2.2 removeCalendar()

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

### 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 Kalendar zu schreiben.

## 1.3 CalendarInterface

Dies ist die Klasse die direkt auf den Kalender zugreift.

Im Folgenden wird auf die wichtigsten Methoden eingegangen:

- createCalenderIfNeeded() -> 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 createCalenderIfNeeded() -> 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 Dictonary 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 eine Änderung wird oder ob aus einer Änderung wieder eine Vorlesung wird. Dabei werden die EventsIDs in das entsprechende Dictonary 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 in CalendarData persistent.

## 1.4 CalendarData

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

# 2 Daten und Zugrifsschichten

## 2.1 Einleitung

Benutzer und App-Daten werden in zwei verschiedenen Datentöpfen gespeichert. Auf diese Datentöpfe soll nur durch Zugriffsschichten zugegriffen werden. Im Folgenden werden die Daten und die zugehörigen Zugrifsschichten genauer beschrieben.

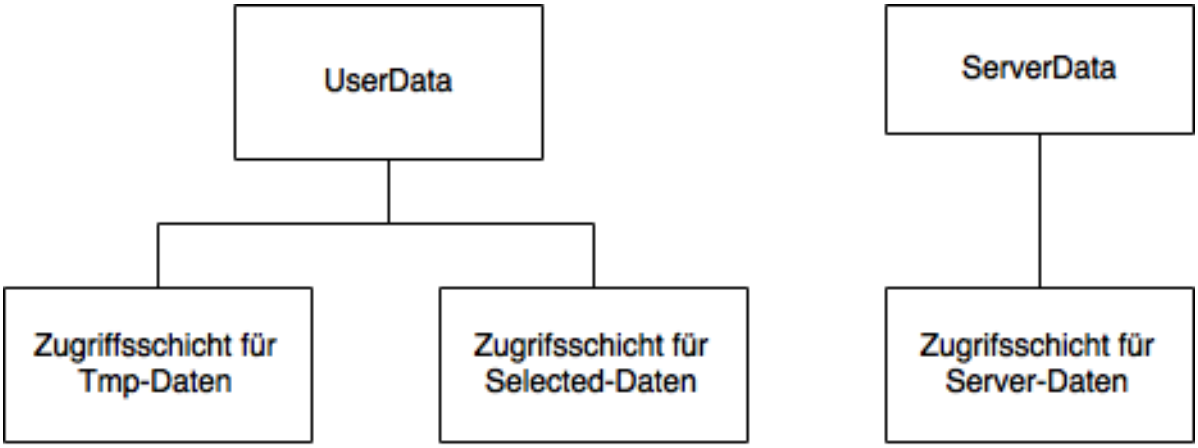


Abbildung 2.1: Daten und zugehörige Zugrifsschichten

Folgende Datentöpfe werden genutzt:

- UserData
- ServerData

Folgende Zugrifssschichten sind vorhanden:

- SelectedCourse
- SelectedSemester
- SelectedLectures
- AllChanges
- AllCourses
- AllLectures

# 3 Hintergrundaktualisierung

## 3.1 Einleitung

Die App soll im Hintergrund regelmäßig prüfen ob neue Stundenplanänderungen für den Benutzer entstanden sind. Daher wurde der Background Fetch implementiert.

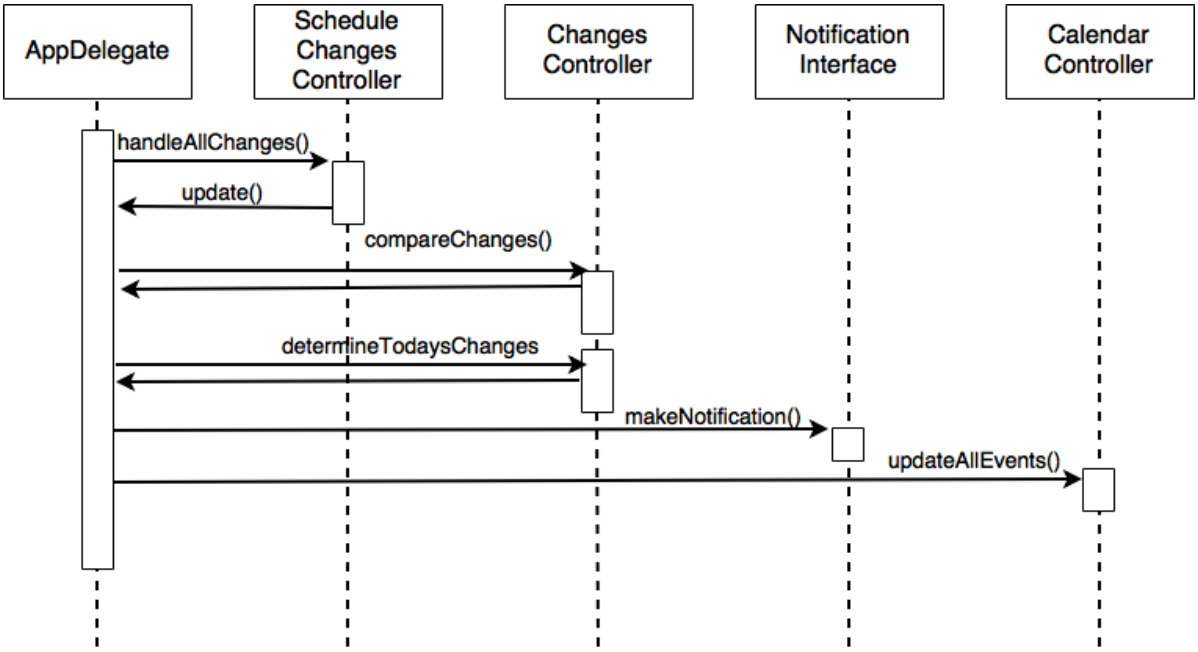


Abbildung 3.1: Ablauf des Background Fetch

In diesem Sequenzdiagramm wird des Ablauf der application(performFetchWithCompletionHandler) Methode dargestellt. Zu Beginn werden durch den ScheduleChangesController die aktuellen Stundenplanänderungen vom Server geladen. Anschließend werden mit dem ChangesController die neu dazugekommenen Änderungen ermittelt und welche davon am aktuellen Tag stattfinden. Als Nächstes wird über das NotificationInterface eine lokale Notification für den User erstellt, die Ihn über die aktuellen Stundenplanänderungen informiert. Zuletzt wird über den CalendarController der iOS Kalender aktualisiert.

## 3.2 CalendarController

Der CalendarController ist der Controller für das Calendar Interface. Über ihn 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.

### 3.2.1 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, dass der Nutzer die Berechtigung erteilt hat und der Kalender angelegt wurde oder bereits angelegt war.

- notDetermined

  Bedeutet, dass die Berechtigung gerade abgefragt wird.

- denied

  Bedeutet, dass die Berechtigung verweigert wurden.

### 3.2.2 removeCalendar()

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

### 3.2.3 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.

### 3.2.4 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.

### 3.2.5 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.

### 3.2.6 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 Kalendar zu schreiben.

## 3.3 CalendarInterface

Dies ist die Klasse die direkt auf den Kalender zugreift.

Im Folgenden wird auf die wichtigsten Methoden eingegangen:

- createCalenderIfNeeded() -> 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()

### 3.3.1 createCalenderIfNeeded() -> Bool

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

### 3.3.2 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.

### 3.3.3 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 Dictonary hinzu.

### 3.3.4 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 eine Änderung wird oder ob aus einer Änderung wieder eine Vorlesung wird. Dabei werden die EventsIDs in das entsprechende Dictonary verschoben.

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

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

### 3.3.6 saveIDs()

Speichert die EventIDs der Vorlesungen und Änderungen in CalendarData persistent.

## 3.4 CalendarData

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