

Beta 2.0

Full Stack iOS Entwicklung mit Swift

WPF im MIM - WS 17/18
Alexander Dobrynin, M.Sc.

Heute

Alerts and Action-Sheets
Notification-Center
User-Notifications (Local & Remote)
Background-Updates

Demo
Assignment

Alerts and Action-Sheets

- ViewController welcher den Benutzer entweder **benachrichtigt** oder eine **Entscheidung** einholt
- UIAlertController unterscheidet zwischen zwei Stilen, **Alert** und **Action-Sheet**, wobei die API exakt die selbe ist
- Alert
 - Wird zentriert und modal präsentiert, während die restliche App “ausgegraut” wird
 - **Beschlagnahmt den aktuellen Kontext der App, unterbricht die aktuelle Arbeit und entzieht die Aufmerksamkeit des Benutzers**
 - Sollte mit absoluter Vorsicht und nicht als Hauptbestand der Benutzer-Interaktion verwendet werden (Anti-Pattern)
 - Wird häufig zum Präsentieren von Fehlermeldungen (Kenntnisnahme) verwendet
- Action-Sheet
 - Lässt den Benutzer eine **Aktion auswählen**, mit der fortgefahren werden soll
 - Kommt von unten hochgefahren (iPhone)
 - Oder wird als Popover (iPad) präsentiert. Den Popover muss man **explizit einfordern, ansonsten crashed die App beim iPad**. Zudem **muss die Source** des PopoverPresentationController angegeben werden, entweder ein UIBarButtonItem oder eine andere SourceView + SourceRect (dort wo der Pfeil hinzeigt)

Alerts and Action-Sheets

Type .alert

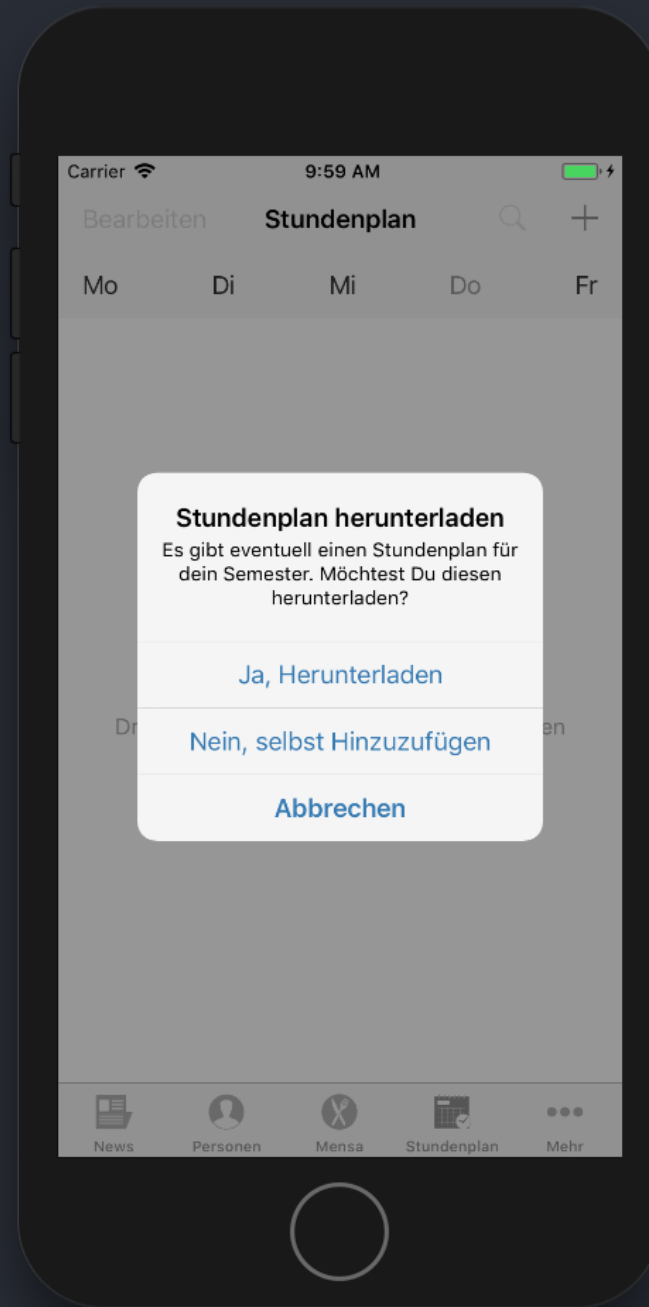
Title

Message

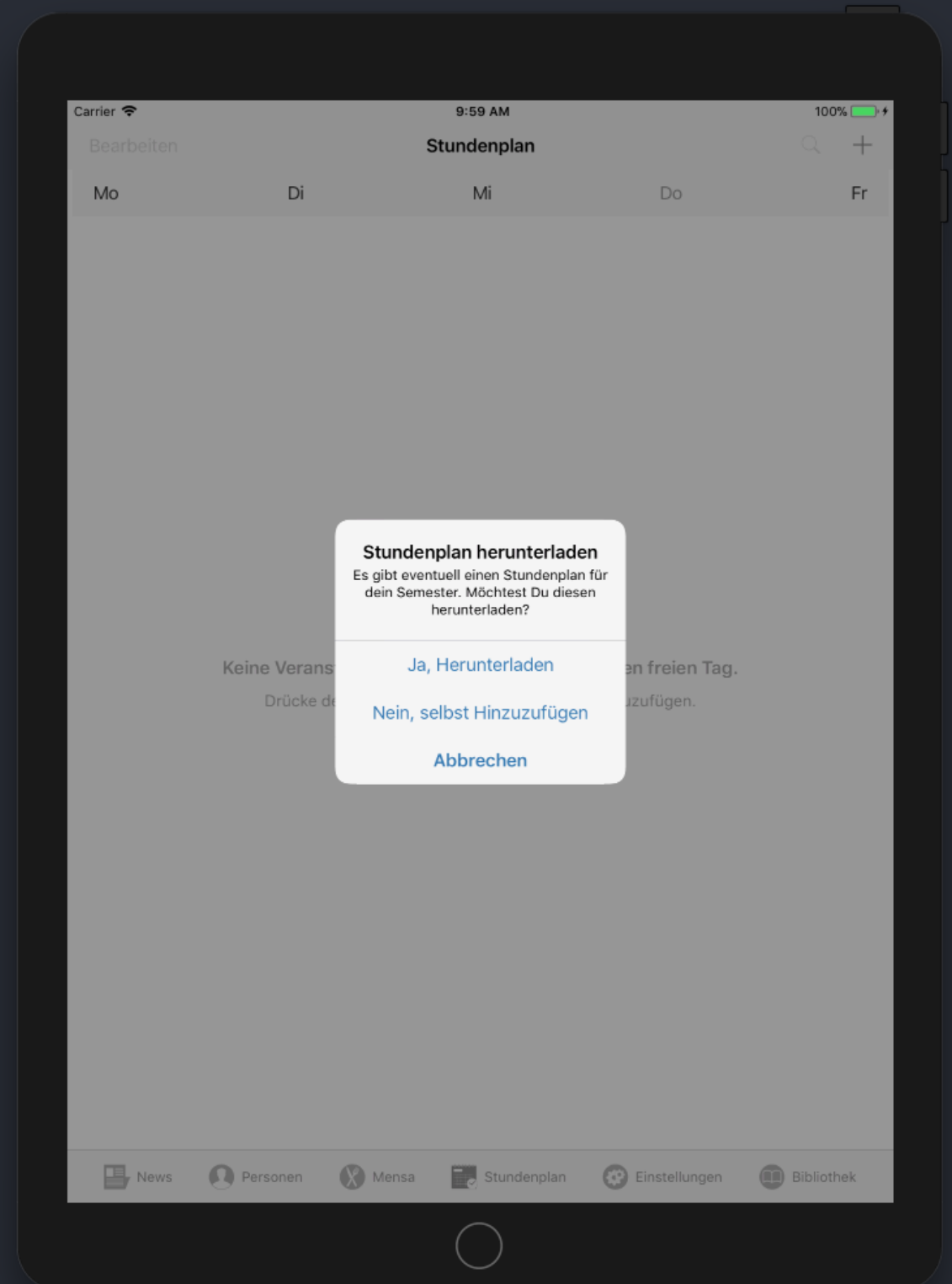
1 Default Action

2 Default Action

Cancel Action



iPhone 8 - 11.2



iPad Air 2 - 11.2

Alerts and Action-Sheets

Type `.actionSheet`

Title

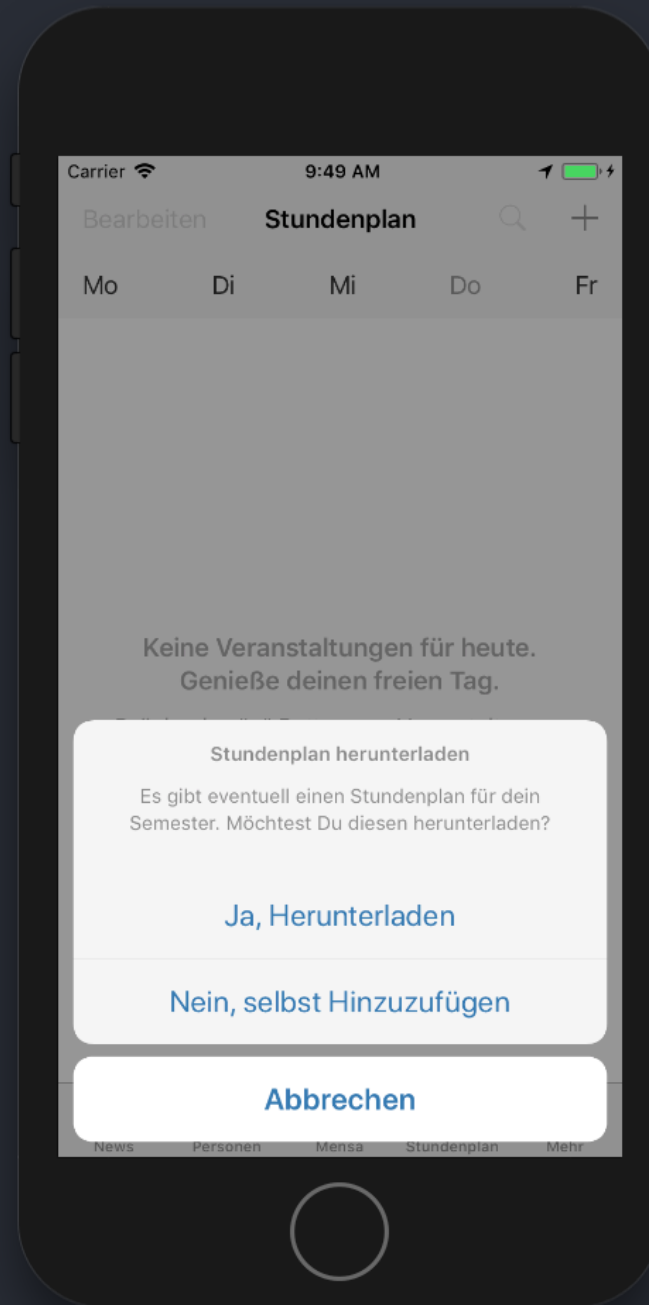
Message

1 Default Action

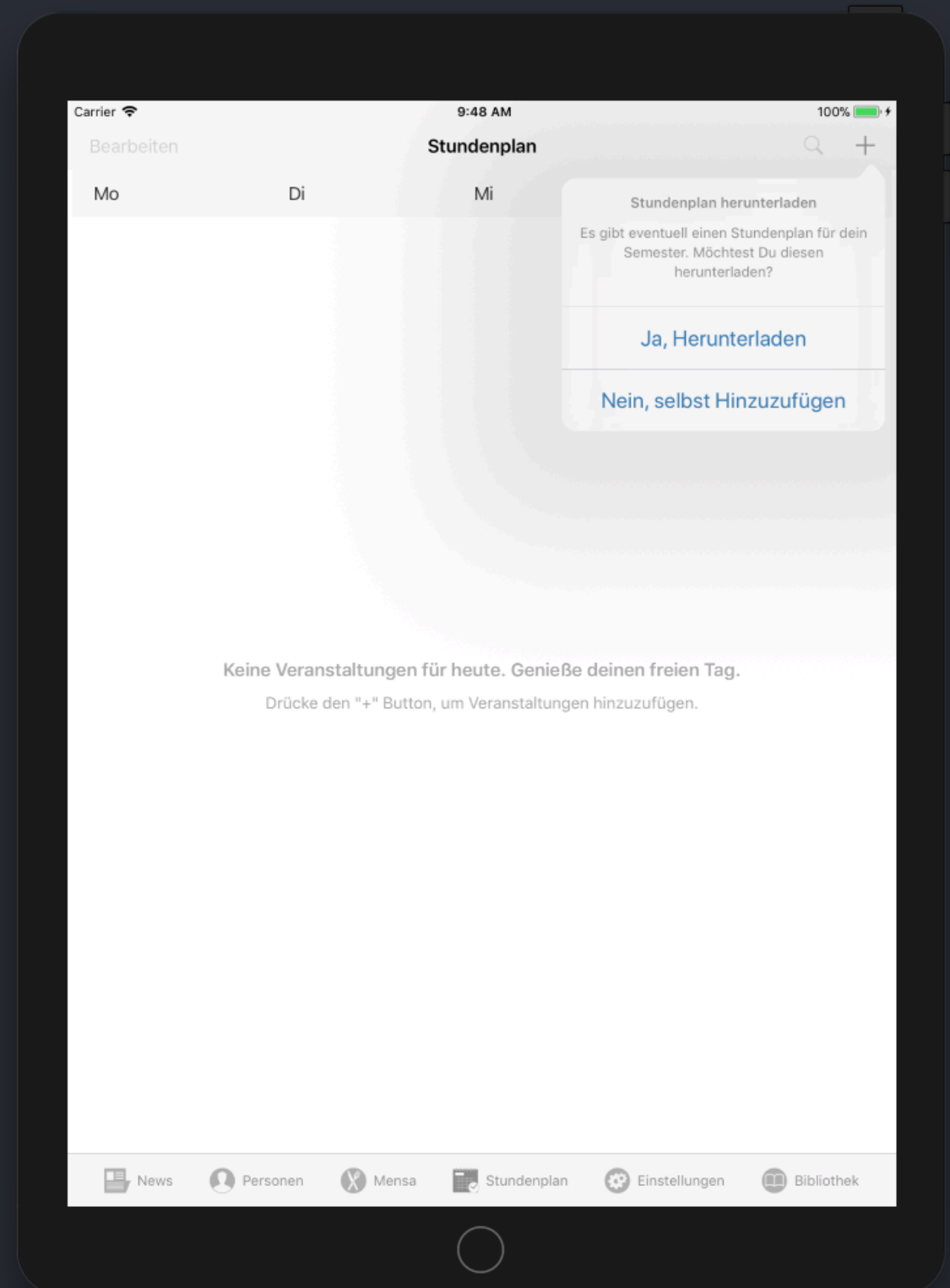
2 Default Action

Cancel Action (iPhone)

Popover source (iPad)



iPhone 8 - 11.2



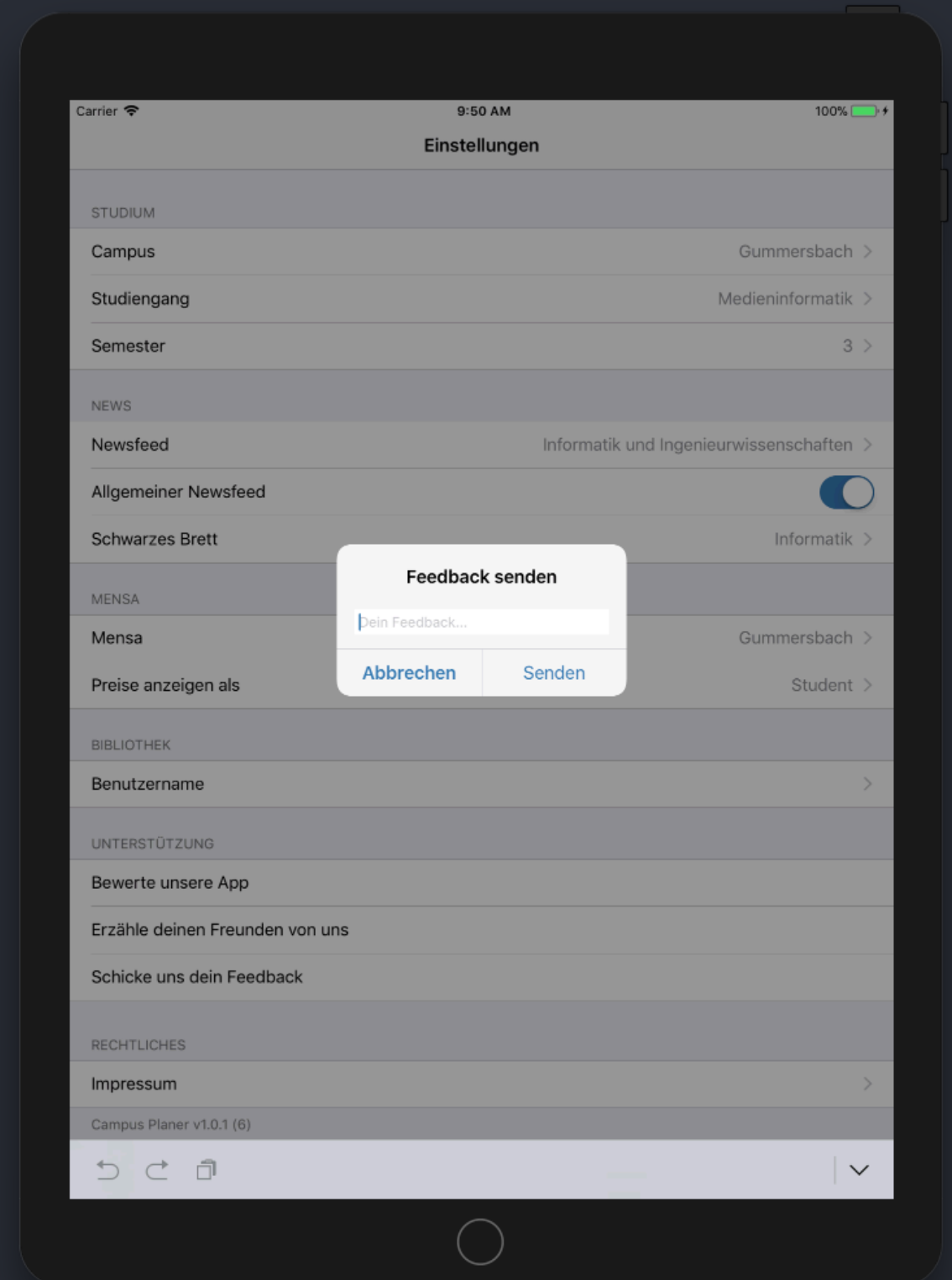
iPad Air 2 - 11.2

Alerts and Action-Sheets

Type .alert
Title
Message TextField
1 Default Action
Cancel Action



iPhone 8 - 11.2



iPad Air 2 - 11.2

Alerts and Action-Sheets

Type .actionSheet

Title

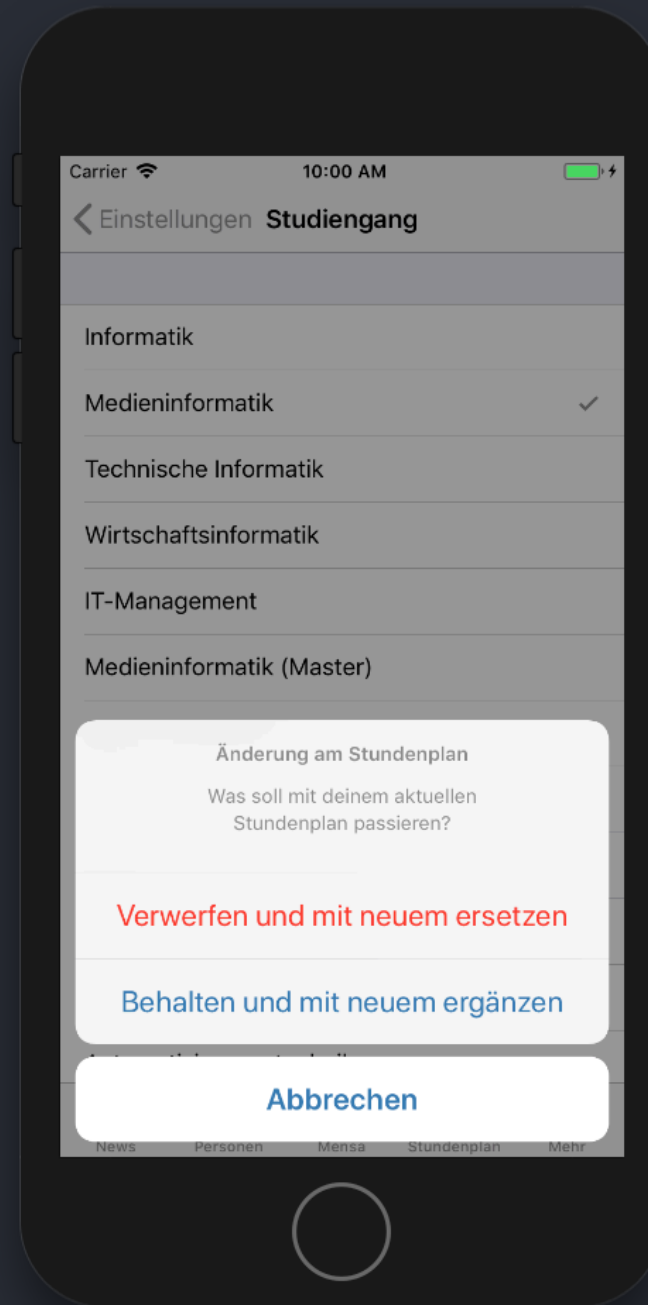
Message

Destructive Action

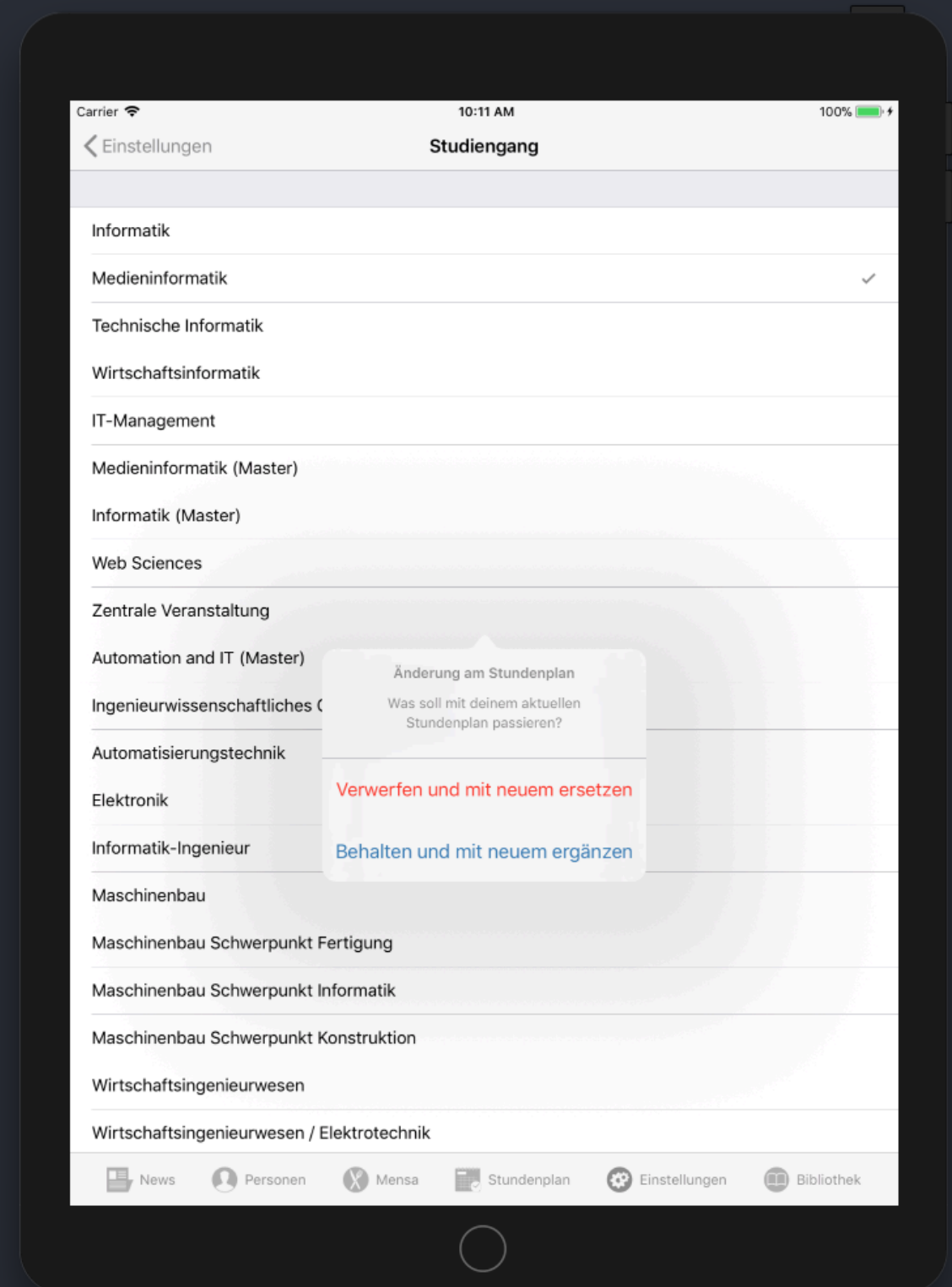
Default Action

Cancel Action (iPhone)

Popover source (iPad)



iPhone 8 - 11.2



iPad Air 2 - 11.2

Alerts and Action-Sheets

```
let alert = UIAlertController(
    title: ScheduleAlertStrings.Title,
    message: ScheduleAlertStrings.Message,
    preferredStyle: .alert // alert style
)

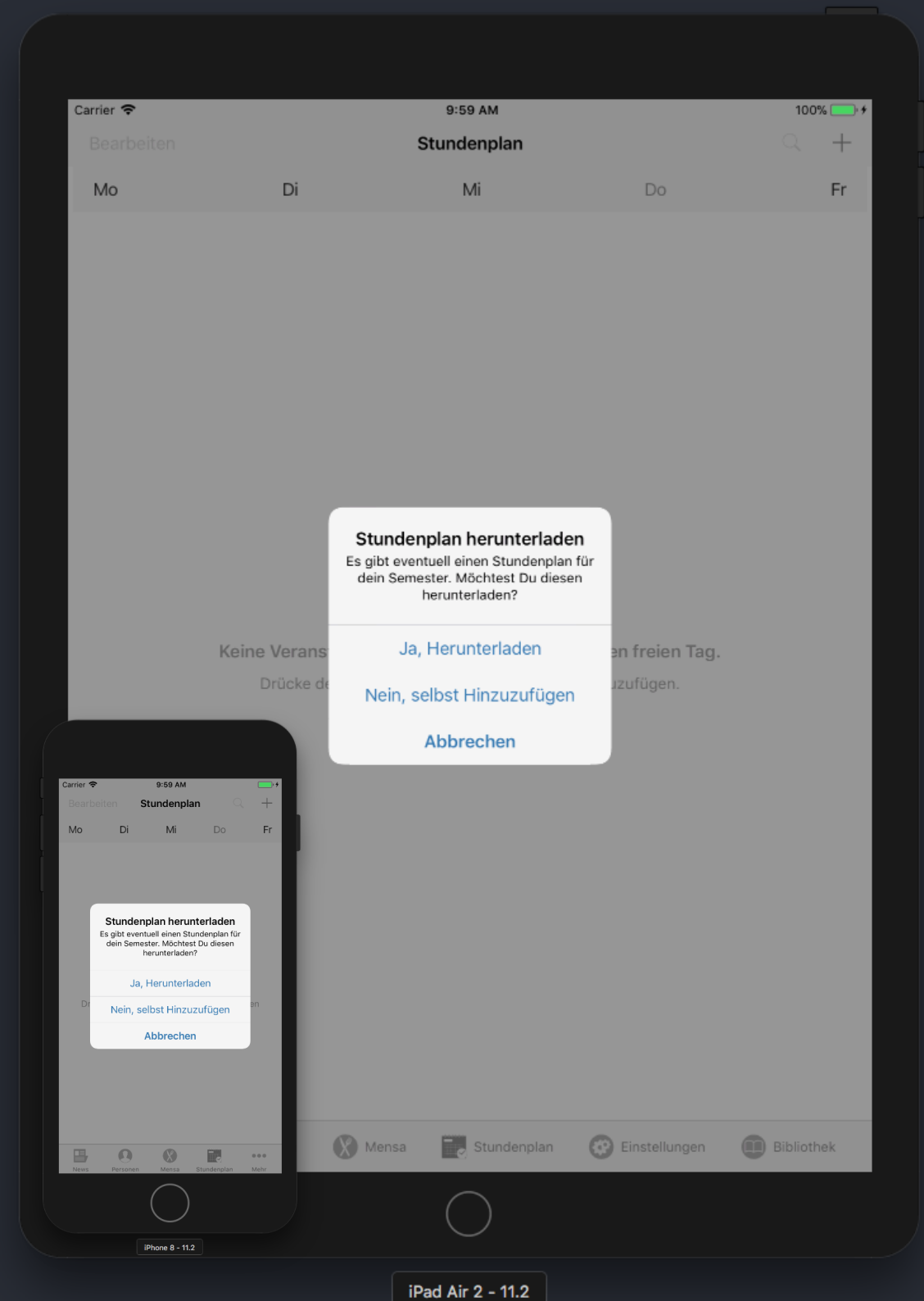
alert.addAction(UIAlertAction(
    title: ScheduleAlertStrings.DownloadButton,
    style: .default,
    handler: { action in self.downloadAll() })
)

alert.addAction(UIAlertAction(
    title: ScheduleAlertStrings.ContinueButton,
    style: .default,
    handler: { action in self.addEntry() })
)

alert.addAction(UIAlertAction(
    title: ScheduleAlertStrings.CancelButton,
    style: .cancel,
    handler: nil)
)

// enable iPad support
alert.modalPresentationStyle = .popover
alert.popoverPresentationController?.barButtonItem =
    navigationItem.rightBarButtonItem

self.present(alert, animated: true, completion: nil)
```



Alerts and Action-Sheets

```
let alert = UIAlertController(
    title: ScheduleAlertStrings.Title,
    message: ScheduleAlertStrings.Message,
    preferredStyle: .actionSheet // action sheet style
)

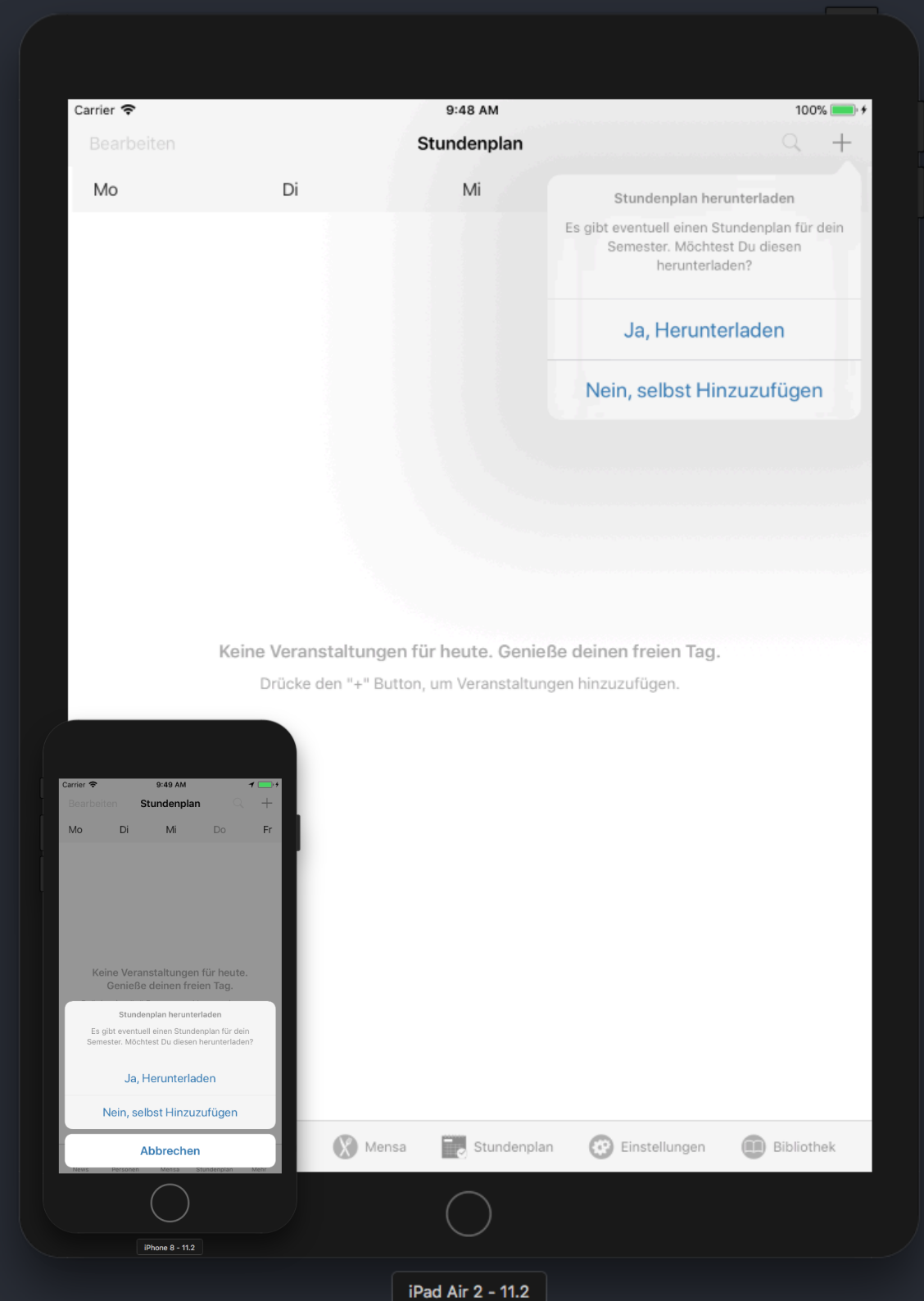
alert.addAction(UIAlertAction(
    title: ScheduleAlertStrings.DownloadButton,
    style: .default,
    handler: { action in self.downloadAll() })
)

alert.addAction(UIAlertAction(
    title: ScheduleAlertStrings.ContinueButton,
    style: .default,
    handler: { action in self.addEntry() })
)

alert.addAction(UIAlertAction(
    title: ScheduleAlertStrings.CancelButton,
    style: .cancel,
    handler: nil)
)

// enable iPad support
alert.modalPresentationStyle = .popover
alert.popoverPresentationController?.barButtonItem =
    navigationItem.rightBarButtonItem

self.present(alert, animated: true, completion: nil)
```



Heute

Alerts and Action-Sheets
Notification-Center
User-Notifications (Local & Remote)
Background-Updates

Demo
Assignment

Notification-Center

- Lose gekoppelte (“blinde”) Kommunikation, bswp. von View/Model zu ViewController (siehe Slides zu MVC)
- Senden (broadcasten) von Notifications

```
extension Notification.Name { // create own topic of type 'Notification.Name'
    static let MyModelDidChange = Notification.Name("MyModelDidChange")
}

NotificationCenter.default.post( // post (broadcast) notifications
    name: Notification.Name.MyModelDidChange, // under which topic to broadcast
    object: self, // sender
    userInfo: ["SomeKey": newData] // payload
)
```

- Registrieren für den Empfang (observe) von Notifications

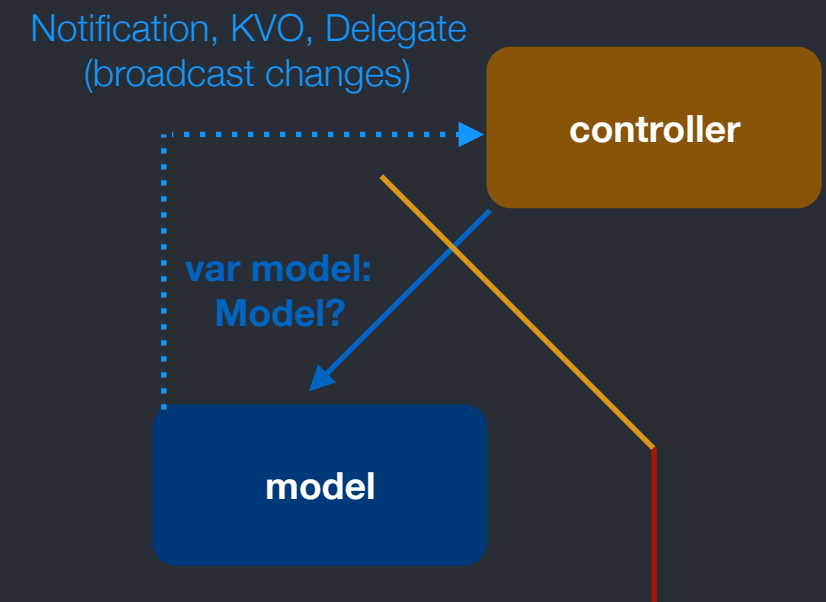
```
observer = NotificationCenter.default.addObserver(
    forName: Notification.Name.MyModelDidChange, // topic to observe for
    object: nil, // who should be the sender. nil for 'anyone'
    queue: OperationQueue.main, // queue where the following closure should be executed on. nil for 'same queue as sender'
    using: { notification in // called every time a new notification was posted 'forName:' from 'object:'
        notification.userInfo?.forEach { print("key:", $0, "value:", $1) }
    }
)
```

- Deregistrieren von Notifications

```
var observer: NSObjectProtocol? // 'cookie' of observation. used for unobserving when finished

NotificationCenter.default.removeObserver(observer) // remove observation from above
```

- Wann und Wo registrieren/deregistrieren? Bei UI Code immer im ViewController an einer Stelle wo es semantisch Sinn ergibt, z.B. in viewWillAppear (add) und viewWillDisappear (remove)



Heute

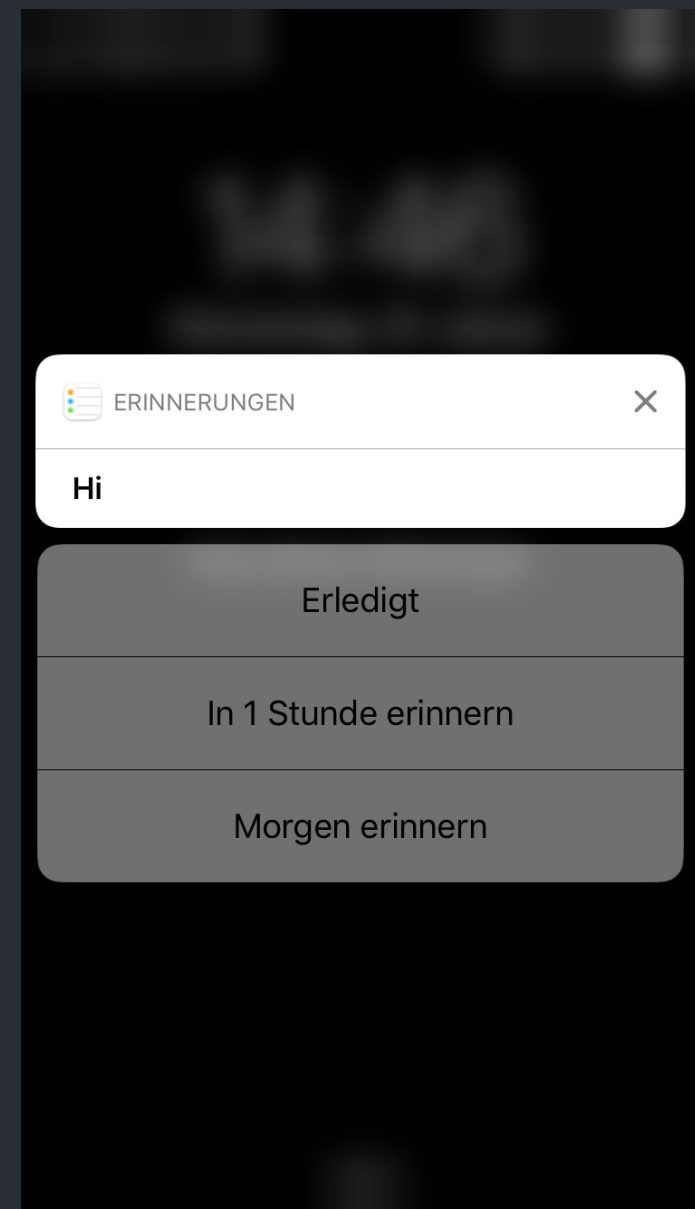
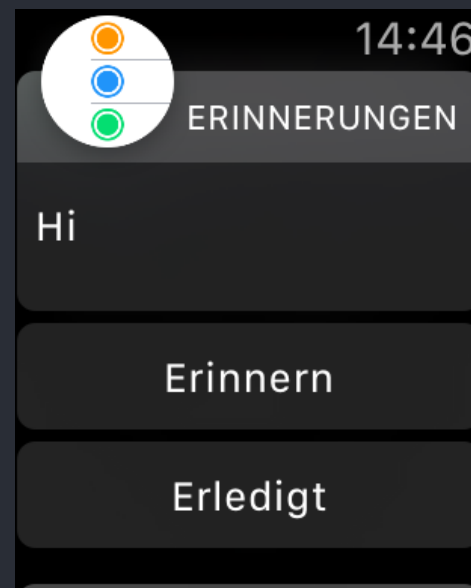
Alerts and Action-Sheets
Notification-Center
User-Notifications (Local & Remote)
Background-Updates

Demo
Assignment

User-Notifications

- Benachrichtigung über die Grenzen (Lebenszyklus) der App
- **Eventgetriebene Ereignisse** (Push) anstelle von benutzermotivierten Handlungen (Poll)
- Erfordert die gleichnamige **Berechtigung** (Opt-In)
- Visuell (Banner und App Icon Badge), auditiv und haptisch (Vibration)
- Notifications vom iPhone werden im vollen Umfang automatisch an die Apple-Watch weitergeleitet
- Unterstützung für Media Attachments und unterschiedliche Actions, inkl. Text-Input
- Aktualisieren von Ausstehenden (Pending) und Zugestellten (Delivered) Notifications, wobei die selbe Notification “erneut zugestellt” wird
- ... und weitere interessante Features, siehe [Dokumentation](#)

User-Notifications



User-Notifications

- iOS unterscheidet zwei Arten von User-Notifications
 - Local: die App selbst erzeugt eine Notification und setzt einen Auslöser (Trigger) fest. iOS nimmt die Notification entgegen und präsentiert diese nach bestem Gewissen, selbst wenn die App nicht im Speicher ist
 - Remote: das Backend* der App sendet eine Notification an alle Devices**, die die App installiert haben. Die serverseitige Notification kann entweder dem Benutzer präsentiert werden (siehe Local) oder eine Hintergrundaktualisierung (Silent Update) anstoßen

Scope von FSiOS

*erfordert ein signiertes Zertifikat (Developer Account), um über Apple-Push-Notification-Service (APNS) zu senden

** nicht direkt an alle Devices, sondern über den APNS-Server von Apple (Proxy)

- In jedem Falle muss man die Berechtigung für Notifications einholen

```
UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .badge, .sound]) { (granted, error) in }
```

- Zudem lassen sich die Einstellungen (z.B. AuthStatus) zur Laufzeit abfragen

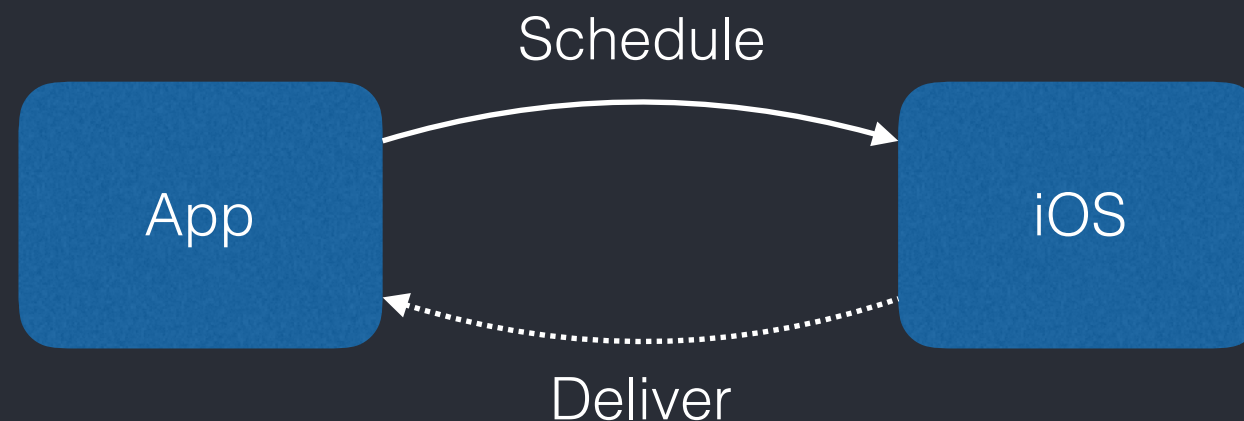
```
UNUserNotificationCenter.current().getNotificationSettings { settings in  
    switch settings.authorizationStatus { } // can be .authorized, .notDetermined, .denied  
}
```

- Für APNS muss zusätzlich ein Token registriert werden (siehe [Dokumentation](#))

```
UIApplication.shared.registerForRemoteNotifications()
```

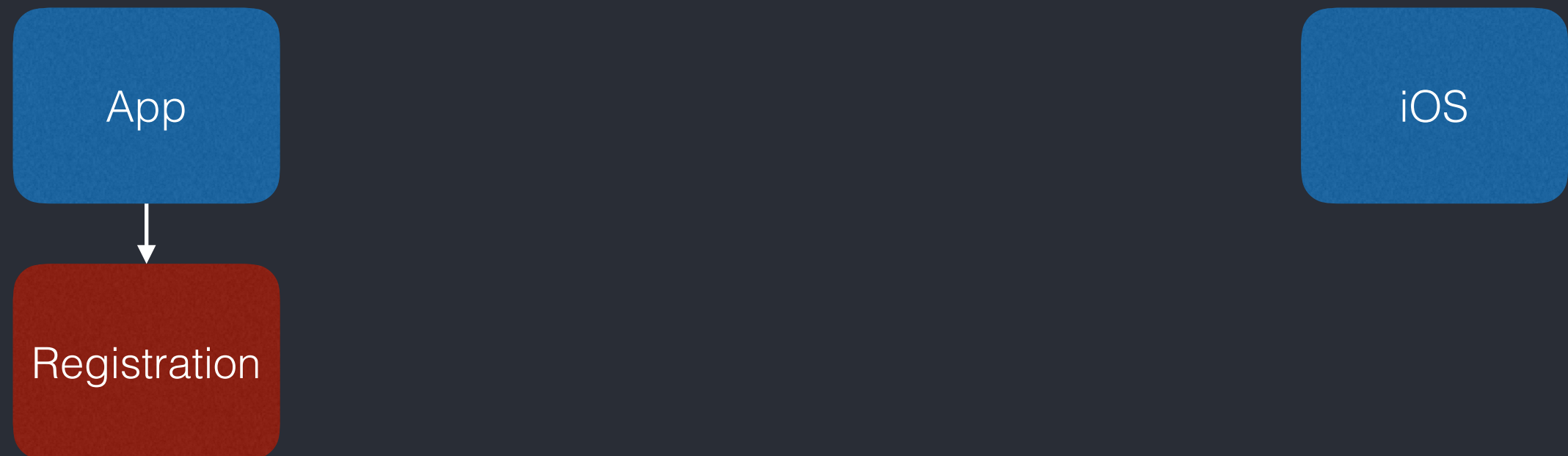
User-Notifications (Local)

- Local: die App selbst erzeugt eine Notification und setzt einen Auslöser (Trigger) fest. iOS nimmt die Notification entgegen und präsentiert diese nach bestem Gewissen, selbst wenn die App nicht im Speicher ist
- Beispiele
 - Zeitbasiert (Interval & Calendar): Erinnerung für heute Abend, Timer für 7 Minuten, Wecker um 8:00 Uhr, Geburtstage von Personen
 - Ortsbasiert (Location): Erinnerung wenn Ankunft zu Hause, Angebote wenn in der Nähe eines Beacons



User-Notifications (Local)

- Local: die App selbst erzeugt eine Notification und setzt einen Auslöser (Trigger) fest. iOS nimmt die Notification entgegen und präsentiert diese nach bestem Gewissen, selbst wenn die App nicht im Speicher ist
- Beispiele
 - Zeitbasiert (Interval & Calendar): Erinnerung für heute Abend, Timer für 7 Minuten, Wecker um 8:00 Uhr, Geburtstage von Personen
 - Ortsbasiert (Location): Erinnerung wenn Ankunft zu Hause, Angebote wenn in der Nähe eines Beacons



User-Notifications (Local)

```
// register user notification (both local and remote)
static func requestAuthorization(completion: @escaping (Bool) -> Void) {
    let center = UNUserNotificationCenter.current()

    center.requestAuthorization(options: [.alert, .badge, .sound]) { (granted, error) in
        // this closure will be executed off the main queue

        if let error = error {
            debugPrint(#function, error.localizedDescription)
        }

        DispatchQueue.main.async { // thus we have to dispatch back on main queue
            completion(granted)
        }
    }
}
```

User-Notifications (Local)

- Local: die App selbst erzeugt eine Notification und setzt einen Auslöser (Trigger) fest. iOS nimmt die Notification entgegen und präsentiert diese nach bestem Gewissen, selbst wenn die App nicht im Speicher ist
- Beispiele
 - Zeitbasiert (Interval & Calendar): Erinnerung für heute Abend, Timer für 7 Minuten, Wecker um 8:00 Uhr, Geburtstage von Personen
 - Ortsbasiert (Location): Erinnerung wenn Ankunft zu Hause, Angebote wenn in der Nähe eines Beacons



User-Notifications (Local)

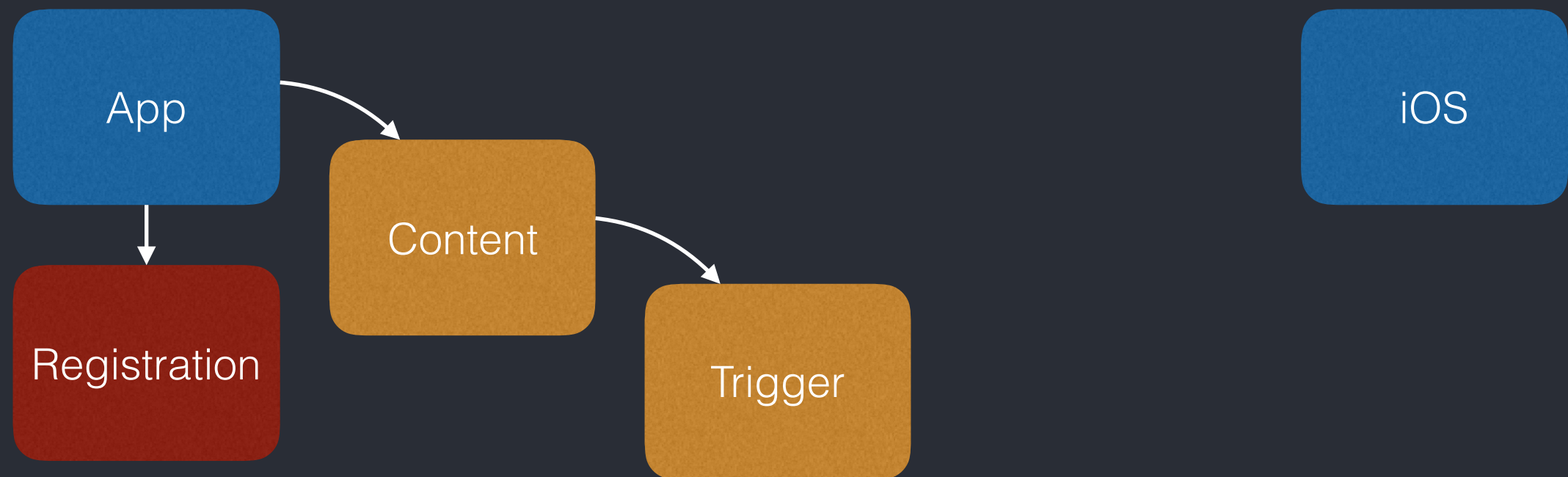
```
struct Model: Codable { }

static func notificationContent(with model: Model) -> UNMutableNotificationContent {
    let content = UNMutableNotificationContent()
    content.title = "New Data Available!"
    content.subtitle = "we are happy to announce new stuff"
    content.badge = 1 // number to appear on app icon
    content.sound = .default()
    content.userInfo = ["Payload": try! PropertyListEncoder().encode(model)] // value must be a property list

    return content
}
```

User-Notifications (Local)

- Local: die App selbst erzeugt eine Notification und setzt einen Auslöser (Trigger) fest. iOS nimmt die Notification entgegen und präsentiert diese nach bestem Gewissen, selbst wenn die App nicht im Speicher ist
- Beispiele
 - Zeitbasiert (Interval & Calendar): Erinnerung für heute Abend, Timer für 7 Minuten, Wecker um 8:00 Uhr, Geburtstage von Personen
 - Ortsbasiert (Location): Erinnerung wenn Ankunft zu Hause, Angebote wenn in der Nähe eines Beacons

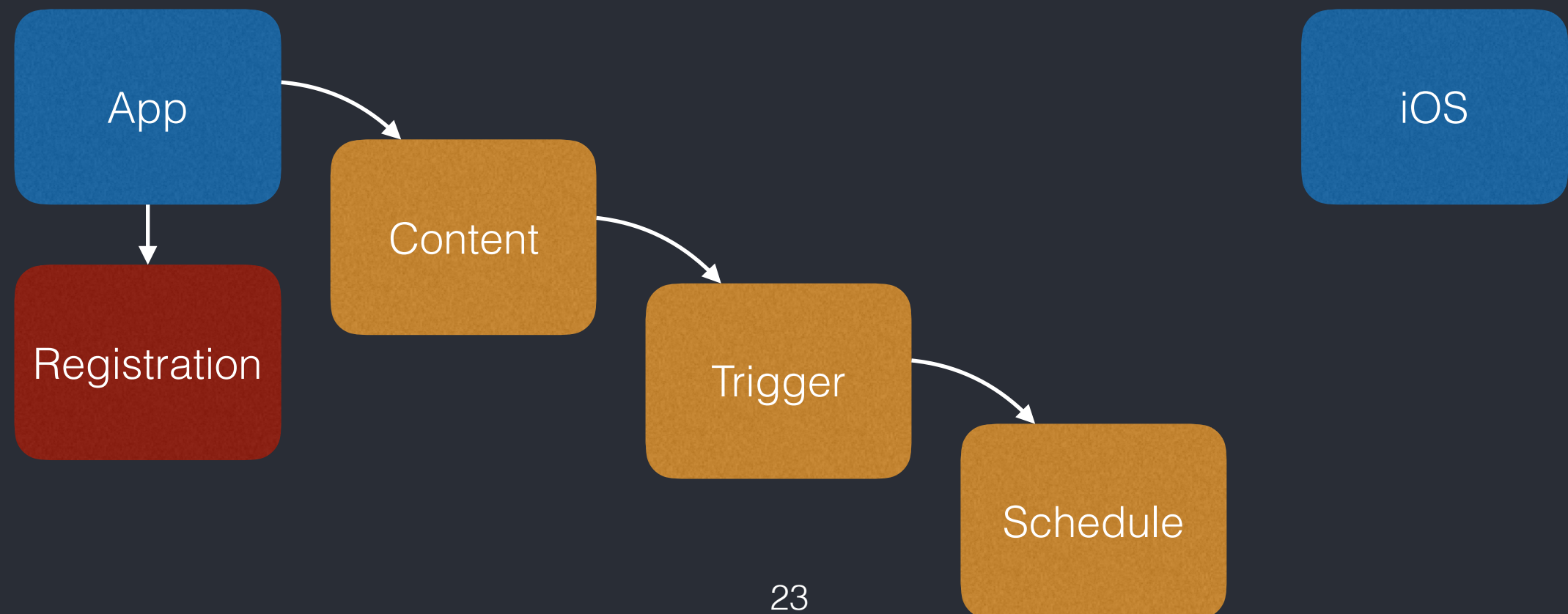


User-Notifications (Local)

```
static func notificationTrigger() {  
    // interval  
    let inFiveMinutes = UNTimeIntervalNotificationTrigger(timeInterval: 60 * 5, repeats: false)  
    let everyMinute = UNTimeIntervalNotificationTrigger(timeInterval: 60, repeats: true)  
  
    // calender  
    var comps = DateComponents()  
    comps.year = 2019  
    comps.month = 3  
    comps.day = 15  
    comps.hour = 17  
    comps.minute = 0  
    // or comps.date = ....  
  
    let atBirthday = UNCalendarNotificationTrigger(dateMatching: comps, repeats: false)  
  
    // location  
    // import CoreLocation  
    let center = CLLocationCoordinate2D(latitude: 51.0232, longitude: 7.5621)  
    let region = CLCircularRegion(center: center, radius: 2000.0, identifier: "THK – Campus Gummersbach")  
    region.notifyOnEntry = true  
    region.notifyOnExit = false  
  
    let arrivingAtCampusGummersbach = UNLocationNotificationTrigger(region: region, repeats: false)  
}
```

User-Notifications (Local)

- Local: die App selbst erzeugt eine Notification und setzt einen Auslöser (Trigger) fest. iOS nimmt die Notification entgegen und präsentiert diese nach bestem Gewissen, selbst wenn die App nicht im Speicher ist
- Beispiele
 - Zeitbasiert (Interval & Calendar): Erinnerung für heute Abend, Timer für 7 Minuten, Wecker um 8:00 Uhr, Geburtstage von Personen
 - Ortsbasiert (Location): Erinnerung wenn Ankunft zu Hause, Angebote wenn in der Nähe eines Beacons

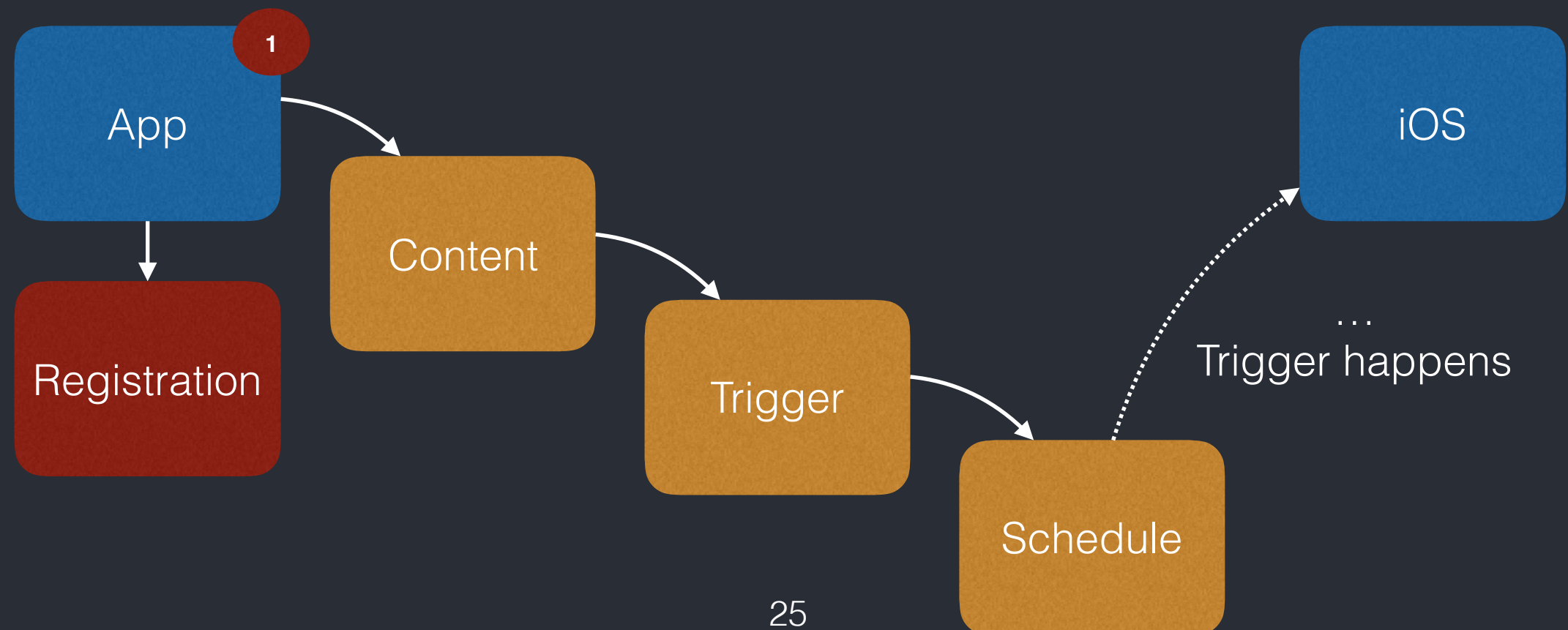


User-Notifications (Local)

```
static func schedule(notification: UNNotificationContent, at trigger: UNNotificationTrigger?) {  
    let request = UNNotificationRequest(  
        identifier: String(Date().timeIntervalSince1970), // id of notification... create a new request each time  
        content: notification,  
        trigger: trigger // nil means 'fire immediately'  
    )  
  
    UNUserNotificationCenter.current().add(request, withCompletionHandler: nil) // schedule notification request  
}
```


User-Notifications (Local)

- Local: die App selbst erzeugt eine Notification und setzt einen Auslöser (Trigger) fest. iOS nimmt die Notification entgegen und präsentiert diese nach bestem Gewissen, selbst wenn die App nicht im Speicher ist
- Beispiele
 - Zeitbasiert (Intervall& Calendar): Erinnerung für heute Abend, Timer für 7 Minuten, Wecker um 8:00 Uhr, Geburtstage von Personen
 - Ortsbasiert (Location): Erinnerung wenn Ankunft zu Hause, Angebote wenn in der Nähe eines Beacons



User-Notifications (Local)

```
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

        UNUserNotificationCenter.current().delegate = self // assign delegate to self

        return true
    }
}

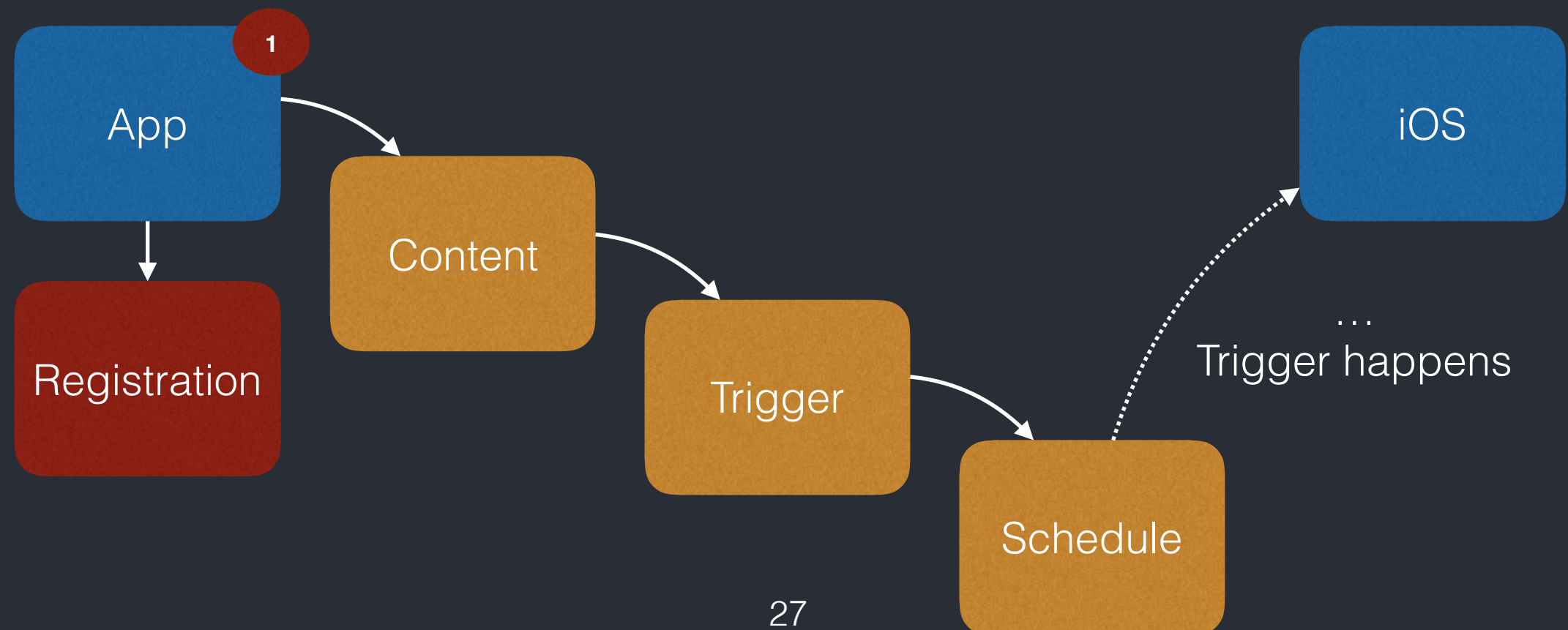
extension AppDelegate: UNUserNotificationCenterDelegate {
    // called to let your app know which action was selected by the user for a given notification.
    func userNotificationCenter(
        _ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: @escaping () -> Void) {
        if let data = response.notification.request.content.userInfo.first as? Data {
            // do something with data...
        }

        completionHandler() // need to be called
    }

    // called when a notification is delivered to a foreground app
    func userNotificationCenter(
        _ center: UNUserNotificationCenter,
        willPresent notification: UNNotification,
        withCompletionHandler completionHandler: @escaping (UNNotificationPresentationOptions) -> Void) {
        completionHandler([]) // empty options means 'dont show while app is in foreground'
    }
}
```

User-Notifications (Local)

- Local: die App selbst erzeugt eine Notification und setzt einen Auslöser (Trigger) fest. iOS nimmt die Notification entgegen und präsentiert diese nach bestem Gewissen, selbst wenn die App nicht im Speicher ist
- Beispiele
 - Zeitbasiert (Interval & Calendar): Erinnerung für heute Abend, Timer für 7 Minuten, Wecker um 8:00 Uhr, Geburtstage von Personen
 - Ortsbasiert (Location): Erinnerung wenn Ankunft zu Hause, Angebote wenn in der Nähe eines Beacons



User-Notifications

- User-Notifications lassen sich im ausstehenden (Pending) und zugestellten (Delivered) Zustand modifizieren
- Entfernen von Notifications

```
center.removeAllPendingNotificationRequests()
center.removeAllDeliveredNotifications()

center.removeDeliveredNotifications(withIdentifiers: ["first"])
center.removePendingNotificationRequests(withIdentifiers: ["second"])
```

- Aktualisieren von Notifications

```
let identifier = "same identifier"

let req = UNNotificationRequest(identifier: identifier, content: content, trigger: trigger)
center.add(req, withCompletionHandler: nil)


// later on ... something changed
let updatedReq = UNNotificationRequest(identifier: identifier, content: content, trigger: trigger)
center.add(updatedReq, withCompletionHandler: nil) // update existing notification, whether delivered or pending
```

Heute

Alerts and Action-Sheets
Notification-Center
User-Notifications (Local & Remote)
Background-Updates

Demo
Assignment

Background-Updates

- Background-Execution: iOS gibt einer App **unter bestimmten Umständen Zeit**, um eine Aufgabe zu erledigen, obwohl die App **nicht im Vordergrund** ist
 - Im Vordergrund gestartete Downloads im Hintergrund fortsetzen
 - Abspielen von Audio, Streaming von AirPlay oder Halten von VoIP Anrufe
 - Jegliche Art von Standortbezogenen Diensten (Location Tracking)
 - Verbindung und Management von Bluetooth (-LE) Zubehör
 - Hintergrundaktualisierungen (neuste Daten herunterladen) 
 - Sonstige “Finite-Length Tasks”, wie Datenbank oder Caches säubern
- Erfordert die entsprechende **Berechtigung** (Opt-In)
- Background-Execution kann die User-Experience einer App deutlich verbessern, sollte aber trotzdem eine Ausnahme sein
- Ausführliche Dokumentation und Code-Beispiele sind dem Programming Guide zu entnehmen

Background-Updates

- Hintergrundaktualisierungen (neuste Daten herunterladen)

Capabilities

ON

- Modes:
- ☐ Audio, AirPlay, and Picture in Picture
 - ☐ Location updates
 - ☐ Newsstand downloads
 - ☐ External accessory communication
 - ☐ Uses Bluetooth LE accessories
 - ☐ Acts as a Bluetooth LE accessory
 - ☒ Background fetch
 - ☐ Remote notifications

Steps: ✓ Add the Required Background Modes key to your info plist file

▼ Required background modes	Array	(1 item)
Item 0	String	App downloads content from the network
Launch screen interface file base...	String	App plays audio or streams audio/video using AirPlay
Main storyboard file base name	String	App registers for location updates
► Required device capabilities	Array	App provides Voice over IP services
► Supported interface orientations	Array	App processes Newsstand Kit downloads
► Supported interface orientations (i...	Array	App communicates with an accessory
Supports Document Browser	Boolean	App communicates using CoreBluetooth
		App shares data using CoreBluetooth
		App downloads content from the network
		App downloads content in response to push notifications



Background-Updates

```
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication,
                    didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        // specifies the minimum amount of time that must elapse between background fetch operations (default is never)
        application.setMinimumBackgroundFetchInterval(UIApplicationBackgroundFetchIntervalMinimum)

        return true
    }

    // tells the app that it can begin a fetch operation if it has data to download.
    func application(_ application: UIApplication,
                    performFetchWithCompletionHandler completionHandler: @escaping (UIBackgroundFetchResult) -> Void) {
        // your app has up to 30 seconds of wall-clock time to perform the download operation
        print(application.backgroundTimeRemaining)

        Downloader.download { res in
            if let res = res {
                // do something with 'res' ...

                // must be called with a value that best describes the results of your download operation
                completionHandler(.newData)
            } else {
                // even with .noData or .failed
                completionHandler(.noData)
            }
        }
    }
}
```

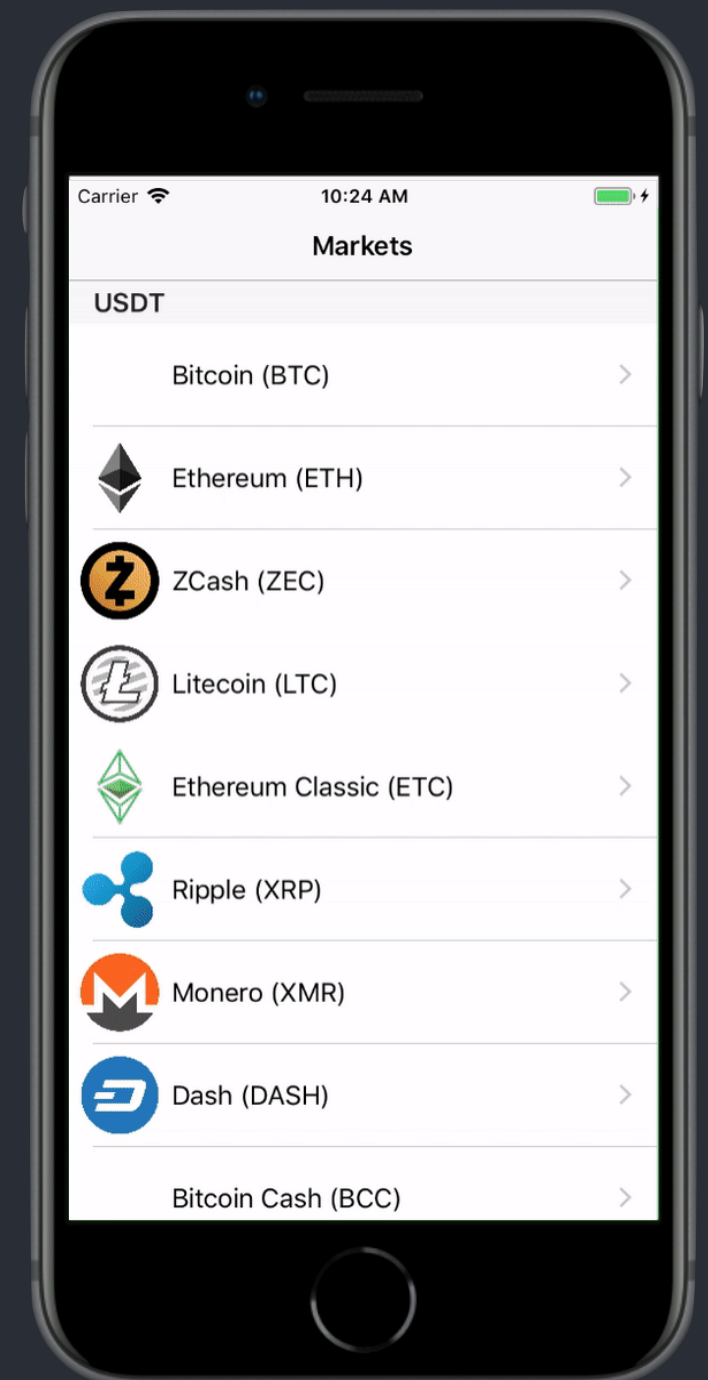

Heute

Alerts and Action-Sheets
Notification-Center
User-Notifications (Local & Remote)
Background-Updates

Demo
Assignment

Cryptomarket - Demo

- NotificationCenter (Broadcasting) bei Änderungen in der DiskDatabase
 - MarketViewController beobachtet Änderungen von MarketSummaries und zeigt die Anzahl dieser beim jeweiligen Markt an
- Sämtliche Errors (propagiert durch Result) werden dem Benutzer als Alert präsentiert
- Background-Updates und User-Notifications
 - Updates laden den letzten Kurs eines hartkodierten Markets
 - Das Delta wird berechnet und anschließend dem Benutzer als User-Notification präsentiert
 - Bei Tap auf die Notification öffnet die App den DetailViewController des jeweiligen Markets



Heute

Alerts and Action-Sheets
Notification-Center
User-Notifications (Local & Remote)
Background-Updates

Demo
Assignment

Cryptomarket - Assignment

- Option A: Benutzer wählt aus der Liste aller Märkte aus, für welchen Market Background-Updates und User-Notifications durchgeführt werden sollen
 - UIBarButtonItem mit Notification-Icon (siehe Assets Catalog) im MarketTableViewController
 - Nur eine Auswahl möglich, die in `UserDefaults` gespeichert wird. Tipp: benutzt `PropertyListEncoder`- und `Decoder` (`UserDefaults` speichert nur `PropertyLists`)
 - Optional: für eine bessere UX wird der Permission-Dialog in den Notification-TVC verschoben (semantische Nähe). Mithilfe von `center.getNotificationSettings` wird der `.authorizationStatus` abgefragt und entsprechend behandelt
 - `.authorized`; zeige alle Märkte an
 - `.notDetermined`: hole erst hier die Berechtigung ein (`requestAuthorization`)
 - `.denied` (optional): zeige keine Märkte an und verweise den Benutzer in die Einstellungen, damit er die Berechtigung manuell vergibt. Tipp: den TVC von `EmptyDataTableViewController` erben lassen und `EmptyDataTableViewControllerDataSource` implementieren. Dann bekommt man den "Empty Data Screen" geschenkt
- Option B: "Buy" und "Sell" Actions bei der User-Notification. Bei einem Tap wird ein entsprechender VC präsentiert welcher anzeigt, dass für den letzten Kurs verkauft oder gekauft wurde
 - Schaut euch hierfür Introduction to Notifications (insb. ab Minute 21:34, aber auch sonst empfehlenswert) an
- Sonstige Änderungen und Verbesserungen sind Willkommen
- Bis zum 30.01.18, 13:59 Uhr per Pull-Request einreichen

