Beta 2.0

Full Stack iOS Entwicklung mit Swift

WPF im MIM - WS 17/18

Alexander Dobrynin, M.Sc.

Heute

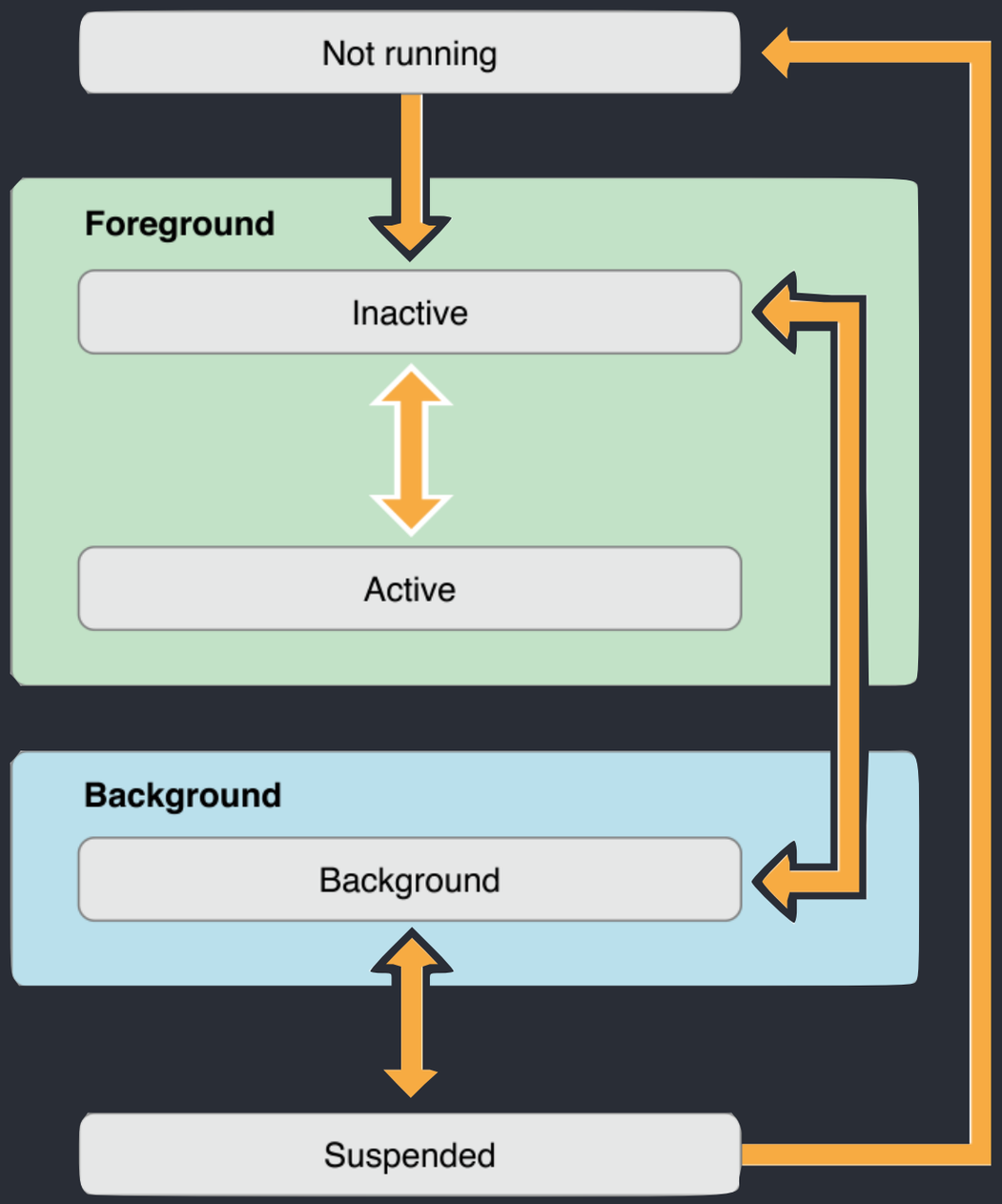
Application Lifecycle
ViewController Lifecycle
Segues (Detail, Modal, Popover, Unwind, Embedded)
Segues im ViewController Lifecycle

Demo
Assignment

Application Lifecycle

- Die Klasse AppDelegate ist der Delegate (Reagierende) für alle Application Lifecycle Events
- Lifecycle deshalb, weil es eine definierte State-Machine gibt, welche Reihenfolge und Regeln definiert
- Typische Events sind
    - App kommt in den Vordergrund
    - App geht in den Hintergrund
    - App bekommt Zeit für Hintergrundaktualisierungen
- Die Events können auch über das Notification-Pattern unter dem jeweiligen Notification.Name abonniert werden
- Achtung: Es gilt als Anti-Pattern, wenn man State (z.B. Einstellungen) und Anwendungslogik in der AppDelegate hat. Es soll lediglich ein Delegate sein

Application Lifecycle



- application:willFinishLaunchingWithOptions: This method is your app's first chance to execute code at launch time.
- application:didFinishLaunchingWithOptions: This method allows you to perform any final initialization before your app is displayed to the user.
- applicationDidBecomeActive: Lets your app know that it is about to become the foreground app. Use this method for any last minute preparation.
- applicationWillResignActive: Lets you know that your app is transitioning away from being the foreground app. Use this method to put your app into a quiescent state.
- applicationDidEnterBackground: Lets you know that your app is now running in the background and may be suspended at any time.
- applicationWillEnterForeground: Lets you know that your app is moving out of the background and back into the foreground, but that it is not yet active.
- applicationWillTerminate: Lets you know that your app is being terminated. This method is not called if your app is suspended.

https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html

Heute

Application Lifecycle
ViewController Lifecycle
Segues (Detail, Modal, Popover, Unwind, Embedded)
Segues im ViewController Lifecycle

Demo
Assignment

ViewController Lifecycle

- Ähnlich zum Application Lifecycle, durchlebt jeder ViewController ebenfalls einen definierten Lifecycle
- Typische Events sind
    - ViewController ist vollständig instanziiert
    - ViewController ist sichtbar / unsichtbar
    - ViewController wird rotiert
- Zudem sind das Setzten von Outlets und das Vorbereiten eines ViewControllers (Transitionen zwischen VC's) ebenfalls ein Teil des Lifecycles
- Achtung: Das Vorbereiten eines ViewController ist das Erste was passiert, weshalb man nicht auf Outlets, andere Properties oder Funktionen zugreifen kann

ViewController Lifecycle

- viewDidLoad: Erst hier sind Preparation und Outlets durchlaufen. Ein guter Platz für Setup-Code
- viewWill/DidAppear: Der VC ist für den Benutzer sichtbar. Hier werden oftmals Animationen oder sonstiger UI bezogener Code gestartet
- viewWill/DidDisappear: Der VC verschwindet. Hier werden häufig die entgegengesetzten Aktionen zu viewWill/DidAppear vorgenommen. Des Weiteren werden laufende Routinen o.ä. aufgeräumt
- Ist der VC einmal geladen, kann er öfters zwischen appear und disappear wechseln. Deshalb sollte man sichergehen, dass hier kein Code ausgeführt wird, der zu viewDidLoad gehört. Geeigneter ist es die Inhalte neu zu laden, Animationen zu starten oder die View mit dem Model zu synchronisieren
- viewWill/DidLayoutSubviews: Die Geometrie des VC hat sich geändert
- viewWillTransition: Der Benutzer rotiert das Gerät
- didReceiveMemoryWarning: Der Speicher wird knapp… hier bekommt man Zeit zum Reagieren
- deinit: Der ViewController wurde erfolgreich dealloziert und liegt nicht mehr im Speicher. Hier kann man mit print's sicherstellen, dass der VC keinen Memory-Leak hat

Heute

Application Lifecycle
ViewController Lifecycle
Segues (Detail, Modal, Popover, Unwind, Embedded)
Segues im ViewController Lifecycle

Demo
Assignment

Segues

- Segues sind Transitionen zwischen zwei ViewController src und dest, wobei src den kompletten MVC-Stack von dest präsentieren möchte
- Der Stil der Präsentation kann über var modalPresentationStyle: UIModalPresentationStyle u.a. auf .fullScreen, .overFullScreen, .popover oder .formSheet gesetzt werden
- Die Animation der Transition kann über var modalTransitionStyle: UIModalTransitionStyle auf .coverVertical, .flipHorizontal, .crossDissolve oder .partialCurl gesetzt werden
- Segues können entweder im Storyboard mit einem Ctrl-Drag zwischen src- und dest-VC modelliert oder im Code ausgeführt werden

```
self.performSegue(withIdentifier: Storyboard.SegueIdentifier, sender: self.model)
```

- Der src-VC kann die Segue vorbereiten, um bswp. das Model des dest-VC zu setzten oder sonstigen Payload zu übertragen

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    switch segue.identifier {
    case Storyboard.SegueIdentifier:
        guard let model = sender as? Model, let dest = segue.destination as? DestViewController else { return }
        dest.model = model
    }
}
```

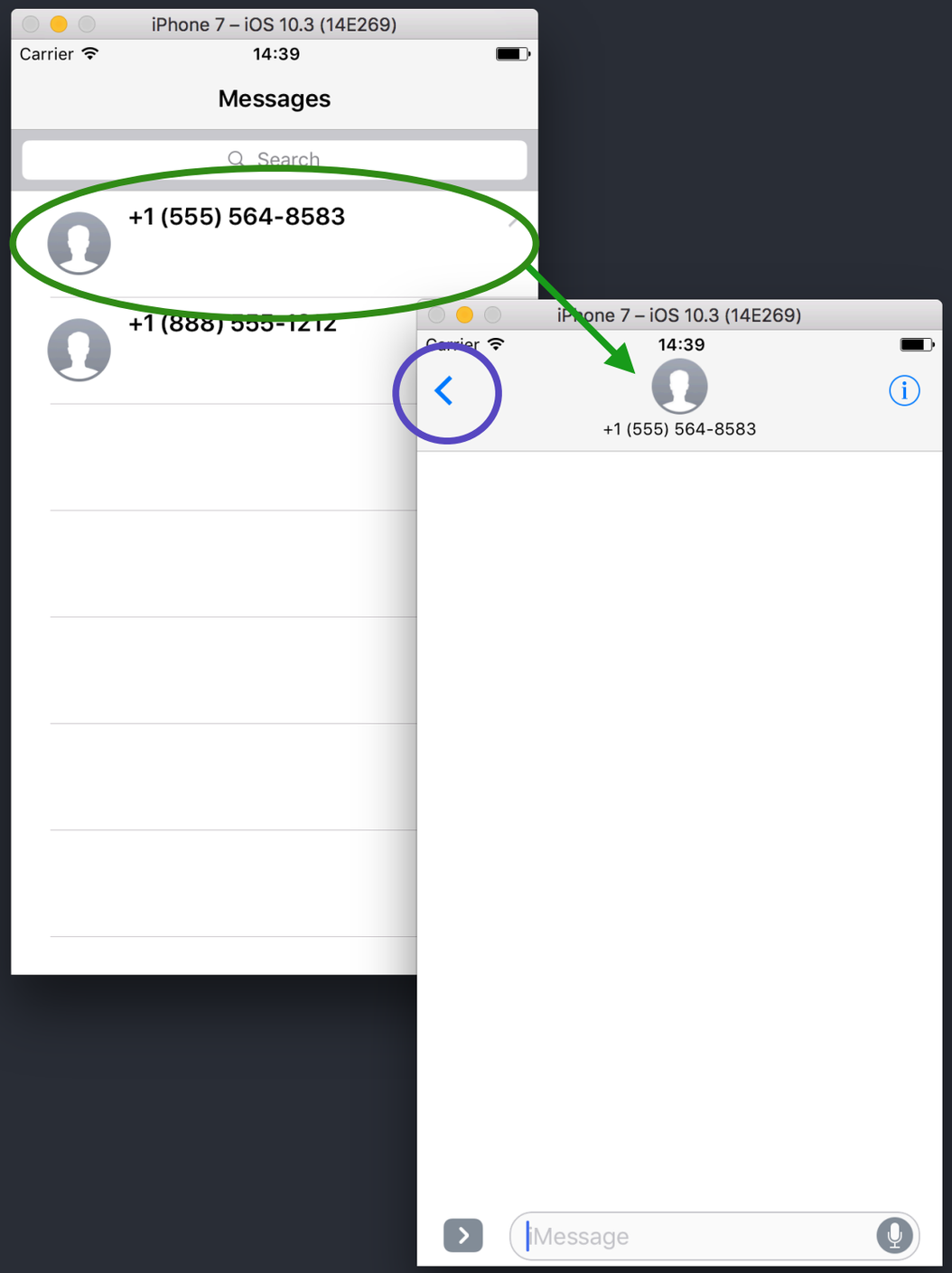
- Achtung: Innerhalb von prepare(for segue) ist auf den VC-Lifecycle zu achten. Die Outlets des dest-VC werden erst nach der Preparation gesetzt

Segues

- Es gibt (min.) 3 Möglichkeiten, um Daten vom DestVC zum SrcVC zu senden
    - DestVC hat ein eigenes Delegate und ruft diesen auf in
        - viewWillDisappear oder
        - navigationController(_ willShow viewController:, animated:) // “<“ in der NavigationBar
    - Notification Pattern, wo der DestVC das Ergebnis in die Datenbank speichert und der SrcVC durch PubSub benachrichtigt wird
    - Uwind-Segue, welche eine @IBAction Funktion des SrcVC aufruft und dort die Möglichkeit gibt, auf den DestVC zuzugreifen, ähnlich zu prepareForSegue
- Die Auswahl der Möglichkeit sollte sich nach dem Zweck und Fähigkeiten des DestVC richten*
    - DestVC wurde in einer Modal-Segue präsentiert: Modal-Segues haben keine hierarchische Struktur zum SrcVC. Zudem werden sie FullScreen präsentiert, weshalb es keinen Zurück Button gibt -> Unwind-Segue
    - DestVC wurde in einer Detail-Segue präsentiert: Hierarchische Struktur kann genutzt werden -> Delegate, aber auch die anderen
    - Sollen Daten zurückgesendet und der DestVC frühzeitig verlassen werden? -> Unwind-Segue
    - Gibt es mehr als einen Grund zum Zurücksenden? -> Delegate oder enum UnwindReason mit Unwind-Segue
- Unwind-Segue löst ein implizites Verlassen des DestVC aus. Sonstige Möglichkeiten sind dismiss(animated: true, completion: nil) für Modal- und navigationController?.popViewController(animated: true) für Detail-Segues

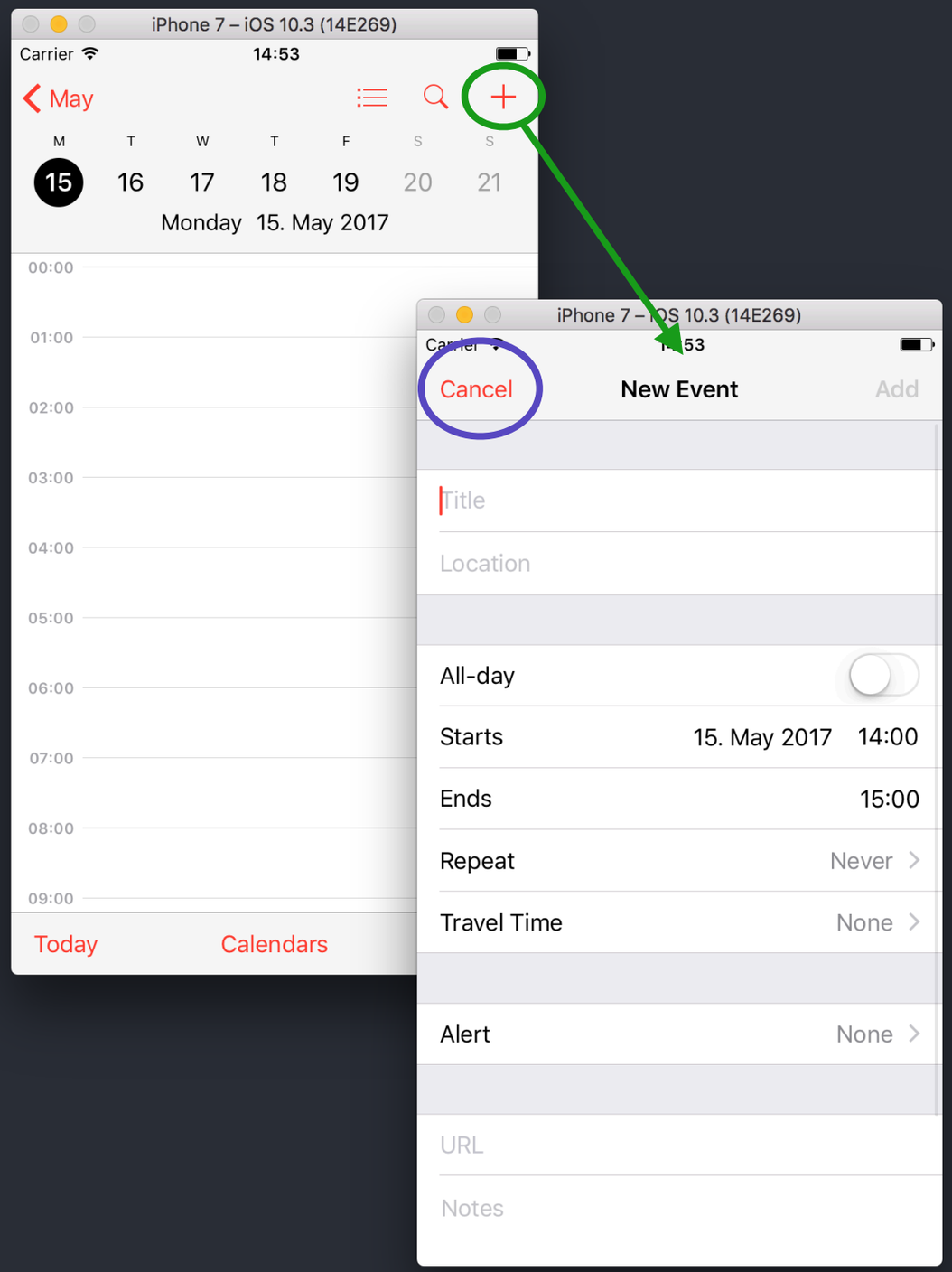
*Das sind lediglich Empfehlungen. Theoretisch kann man alles verwenden. Einiges ist geeigneter als anders.

Segues



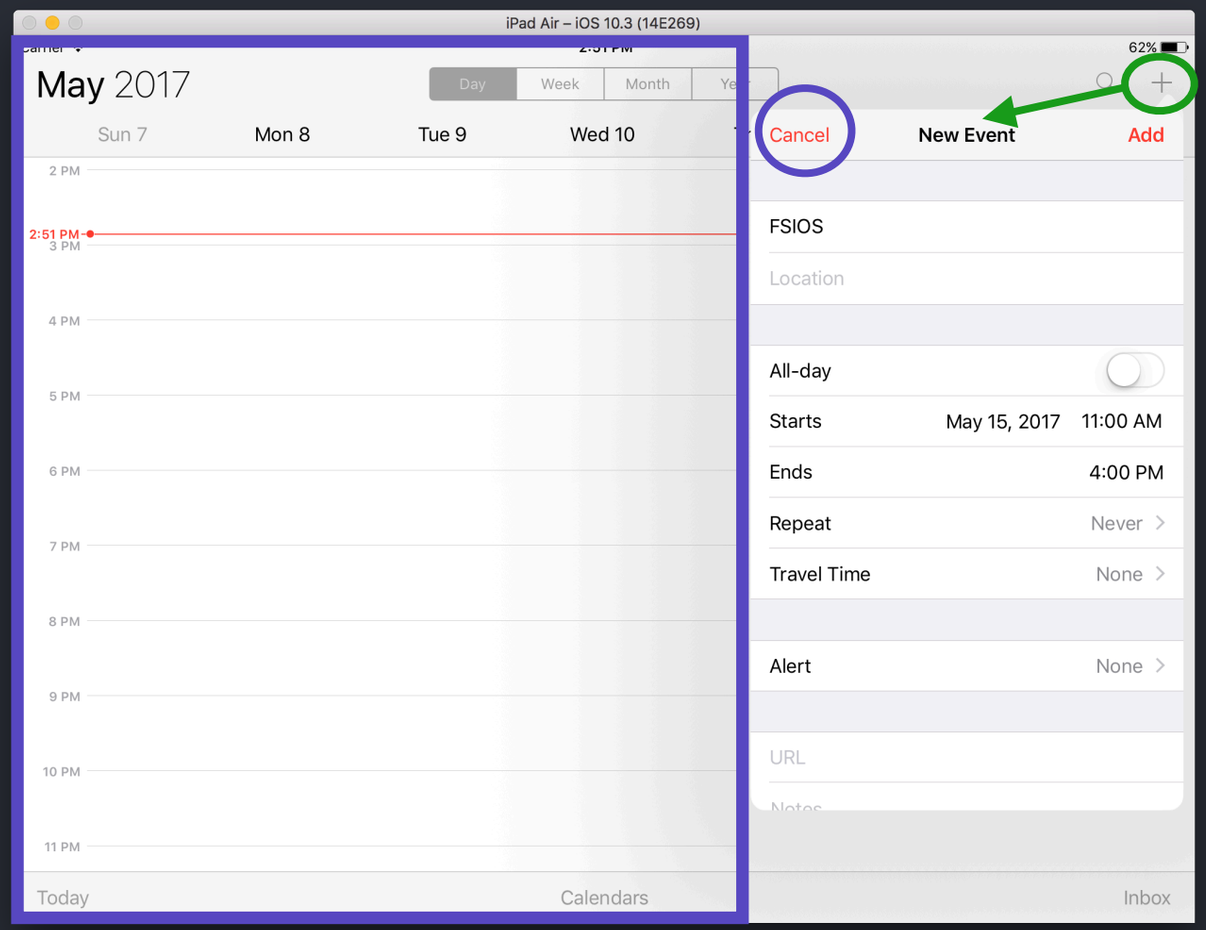
- Detail: Erstellt und präsentiert einen MVC, indem der VC auf den aktuellen Navigations-Stack gepushed wird
- Modal: Erstellt und präsentiert ein MVC über den gesamten Screen
- Popover: Fast das gleiche wie Modal, nur dass es beim iPad “über” den Screen gelegt wird
- Unwind: Erstellt kein neues MVC, sondern kehrt zu einem bestehenden MVC zurück
- Embed: Einen gesamten MVC-Stack innerhalb eines VC “einbetten”

Segues



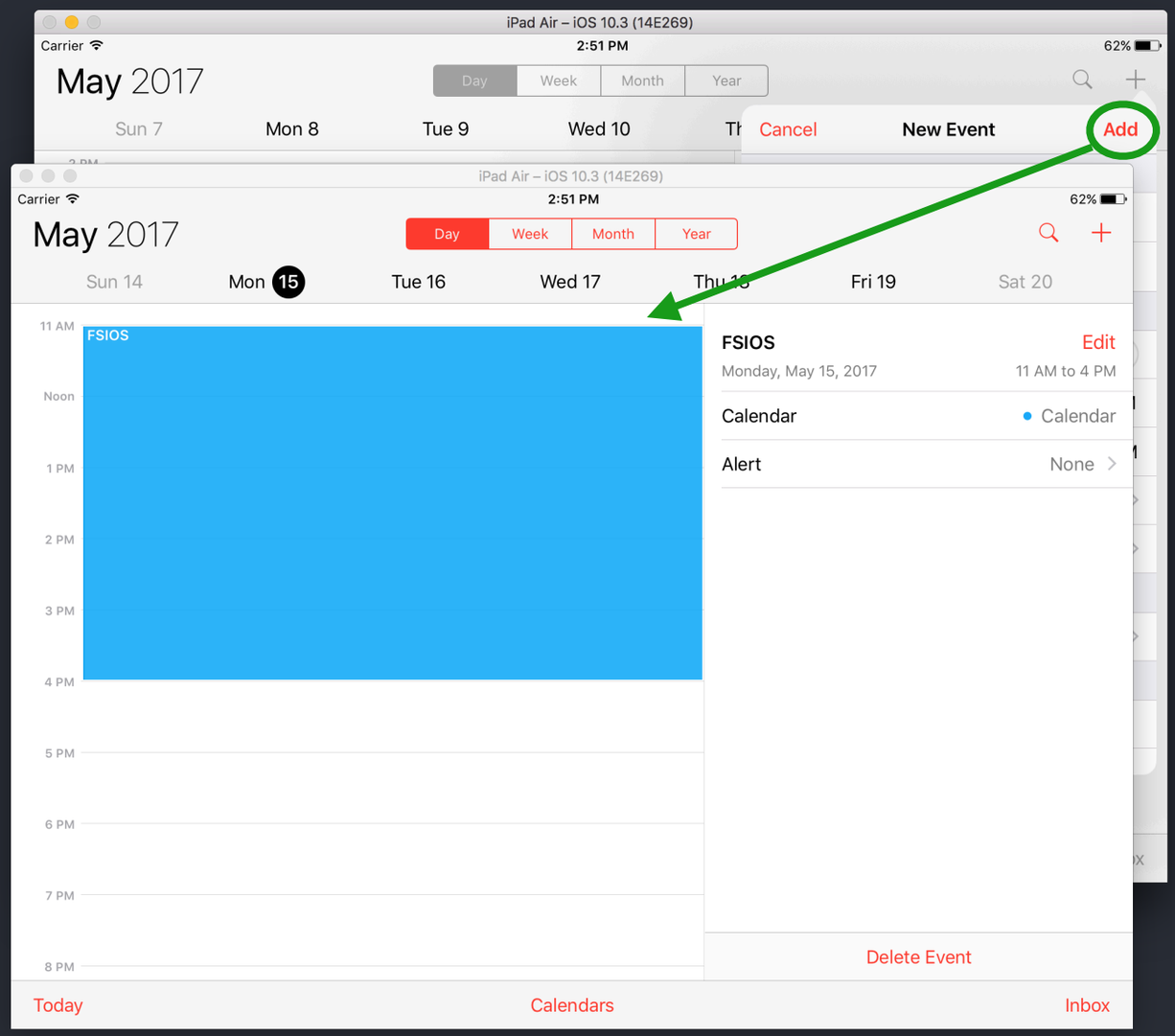
- Detail: Erstellt und präsentiert einen MVC, indem der VC auf den aktuellen Navigations-Stack gepushed wird
- Modal: Erstellt und präsentiert ein MVC über den gesamten Screen
- Popover: Fast das gleiche wie Modal, nur dass es beim iPad “über” den Screen gelegt wird
- Unwind: Erstellt kein neues MVC, sondern kehrt zu einem bestehenden MVC zurück
- Embed: Einen gesamten MVC-Stack innerhalb eines VC “einbetten”

Segues



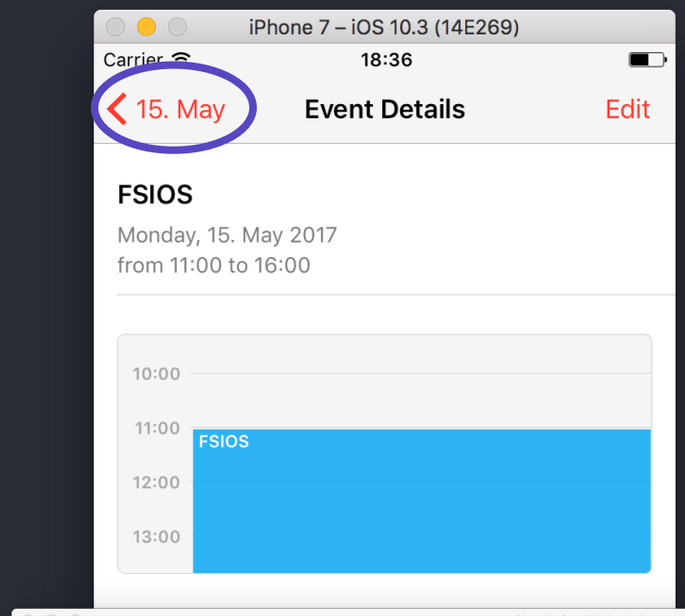
- Detail: Erstellt und präsentiert einen MVC, indem der VC auf den aktuellen Navigations-Stack gepushed wird
- Modal: Erstellt und präsentiert ein MVC über den gesamten Screen
- Popover: Fast das gleiche wie Modal, nur dass es beim iPad “über” den Screen gelegt wird
- Unwind: Erstellt kein neues MVC, sondern kehrt zu einem bestehenden MVC zurück
- Embed: Einen gesamten MVC-Stack innerhalb eines VC “einbetten”

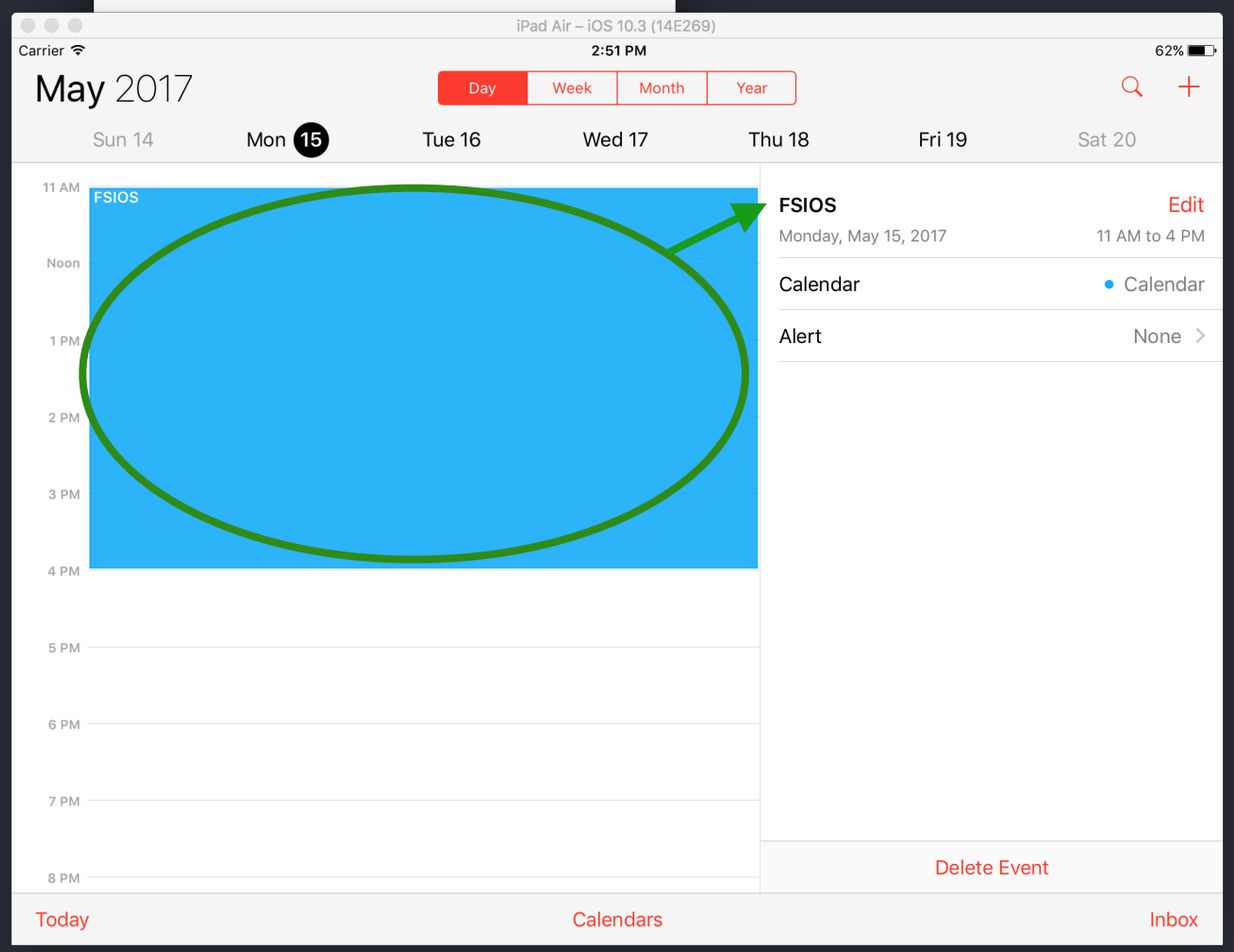
Segues



- Detail: Erstellt und präsentiert einen MVC, indem der VC auf den aktuellen Navigations-Stack gepushed wird
- Modal: Erstellt und präsentiert ein MVC über den gesamten Screen
- Popover: Fast das gleiche wie Modal, nur dass es beim iPad “über” den Screen gelegt wird
- Unwind: Erstellt kein neues MVC, sondern kehrt zu einem bestehenden MVC zurück
- Embed: Einen gesamten MVC-Stack innerhalb eines VC “einbetten”

Segues





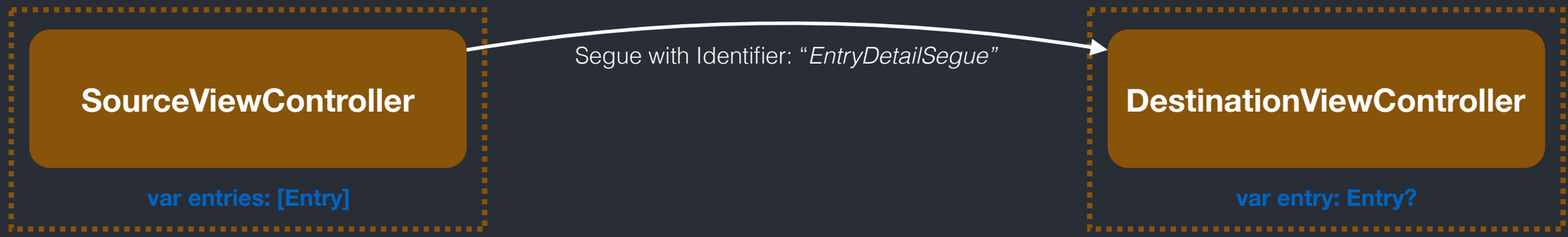
- Detail: Erstellt und präsentiert einen MVC, indem der VC auf den aktuellen Navigations-Stack gepushed wird
- Modal: Erstellt und präsentiert ein MVC über den gesamten Screen
- Popover: Fast das gleiche wie Modal, nur dass es beim iPad “über” den Screen gelegt wird
- Unwind: Erstellt kein neues MVC, sondern kehrt zu einem bestehenden MVC zurück
- Embed: Einen gesamten MVC-Stack innerhalb eines VC “einbetten”

Heute

Application Lifecycle
ViewController Lifecycle
Segues (Detail, Modal, Popover, Unwind, Embedded)
Segues im ViewController Lifecycle

Demo
Assignment

Segues im ViewController Lifecycle



Segues im ViewController Lifecycle

Perform Segue

```
override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    let entry = entries[indexPath.section].value[indexPath.row]
    performSegue(withIdentifier: "EntryDetailSegue", sender: entry)
}
```

Prepare Segue

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    guard let identifier = segue.identifier else { return }

    switch identifier {
        case "EntryDetailSegue":
            guard let entry = sender as? Entry,
                  let dvc = segue.destination as? DestinationViewController
                else { return }

                dvc.entry = entry

        default: break
    }
}
```

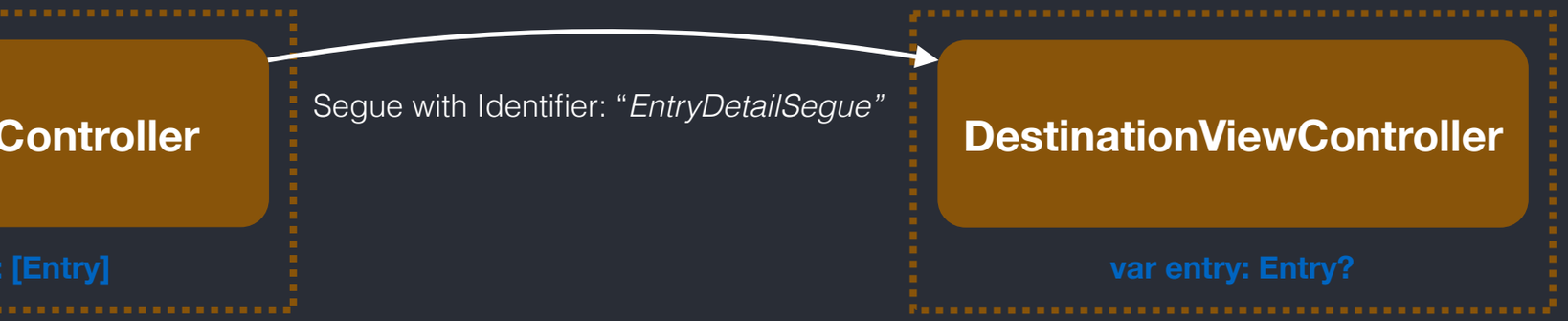
later on…

Yield Results

```
@IBAction func unwindFormDestinationViewController(segue: UIStoryboardSegue) {
    guard let svc = segue.source as? DestinationViewController else { return }

    let updatedEntry = svc.entry!
    let index = entries(of: updatedEntry)
    entries[index!] = updatedEntry
}
```

Segues im ViewController Lifecycle



later on…

Build UI with Model

```
override func viewDidLoad() {
    super.viewDidLoad()

    // TODO: remove in production
    assert(entry != nil, "entry must be set")

    title = entry?.name
    // set outlets based on models data
}
```

Perform Unwind

```
func done() {
    // do something …

    performSegue(withIdentifier: "EntryUnwindSegue", sender: self)
}
```

Prepare Unwind

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    guard let identifier = segue.identifier else { return }

    switch identifier {
    case "EntryUnwindSegue":
        self.entry = updateEntryFromUI()

    default: break
    }
}
```

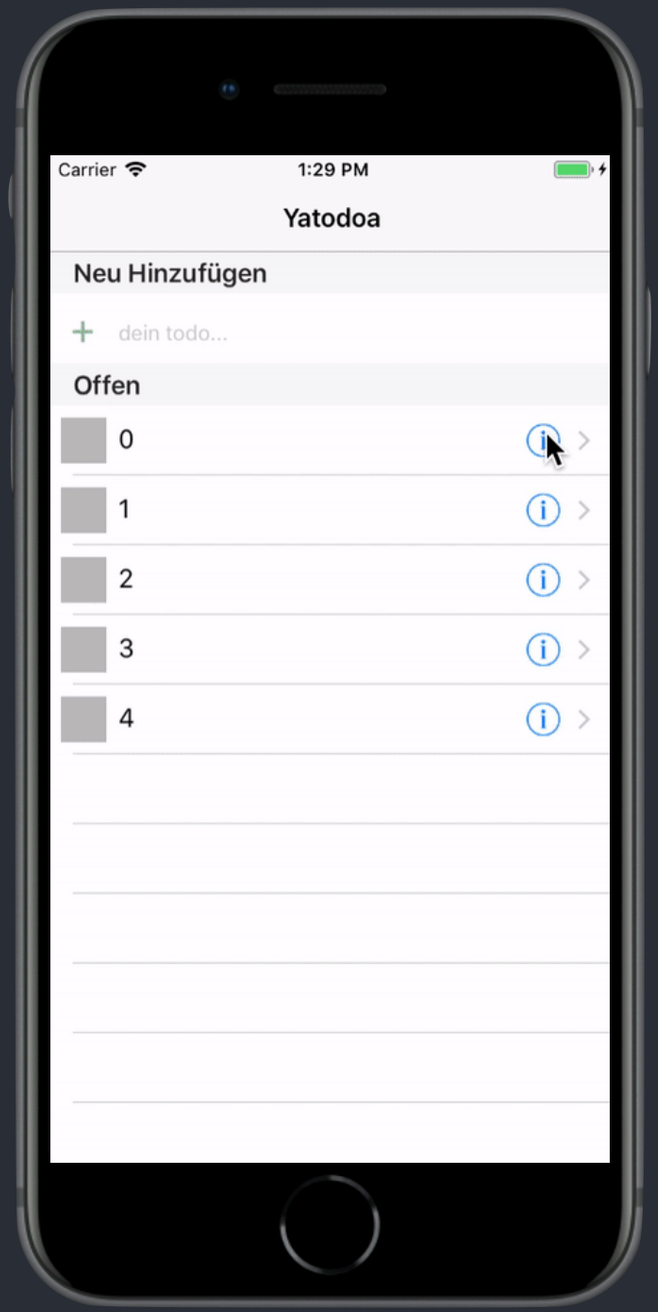
Heute

Application Lifecycle
ViewController Lifecycle
Segues (Detail, Modal, Popover, Unwind, Embedded)
Segues im ViewController Lifecycle

Demo
Assignment

Yatodoa - Demo

- Detail-, Modal- und Unwind-Segue
  - Perform- und Prepare Segue
  - Übertragen und Extrahieren von Payload
- ViewController Lifecycles
- Application Lifecycles
- Statische TableViewController
- UITextView, UISwitch, UIBarButtonItems und UIAccessoryTypes
- Assets und UIImage-Literale
- Enums mit assoziierten Values
- High-Order-Functions



Heute

Application Lifecycle
ViewController Lifecycle
Segues (Detail, Modal, Popover, Unwind, Embedded)
Segues im ViewController Lifecycle

Demo
Assignment

Yatodoa - Assignment

1. TagTableViewController MVC-Stack
    - TVC zeigt eine Liste von vordefinierten Tags an
    - Jeder Tag besteht aus einem Namen und einer Farbe (UIColor)
    - Hat das Todo einen Tag, so wird dieser mit einem Checkmark (AccessoryType) markiert. Wenn ein andere Tag ausgewählt wird, so wird dieser markiert und der vorherige demarkiert
    - Die TextColor der Zelle hat die jeweilige Farbe des Tags
- Optional
    - Es besteht die Möglichkeit, ähnlich zum Konzept der Todos, Tags zu erstellen, welche der Liste angefügt werden
    - Die Farbe der Checkbox hat ebenfalls die Farbe des Tags, damit man die semantische Ähnlichkeit direkt im TodoTableViewController sieht (siehe gif)
- Tipps
    - Arbeitet mit einem Enum TagType und verwendet die AddTodoTableViewCell wieder. Hierfür müsst ihr diese etwas generisch machen
    - Verwendet Basic und keine Custom Cells
    - Setzt die Farbe eines Tags immer auf Random. Hier der Code dazu
    - Achtung: Im TagTVC könnte ihr nicht den Delegaten des NavigationController verwenden, weil dieser bereits vom DetailTodoTVC verwendet wird. Stattdessen könnt ihr Delegate/Unwind in viewWillDisappear verwenden
    - Achtung: Wenn ihr eine DetailSegue macht, müsst ihr sofort den Titel des DestVC setzen, da ansonsten die NavigationBar nicht angezeigt wird
- Sonstige Änderungen und Verbesserungen sind Willkommen
- Bis zum 05:12, 13:59 Uhr per Pull-Request einreichen