



Praktikum – iOS-Entwicklung

Wintersemester 2019/20

Prof. Dr. Linnhoff-Popien

Markus Friedrich, Christoph Roch

Crashkurs

XCode, Projektgrundgerüst, App-Zustände, Views und ViewController

XCode

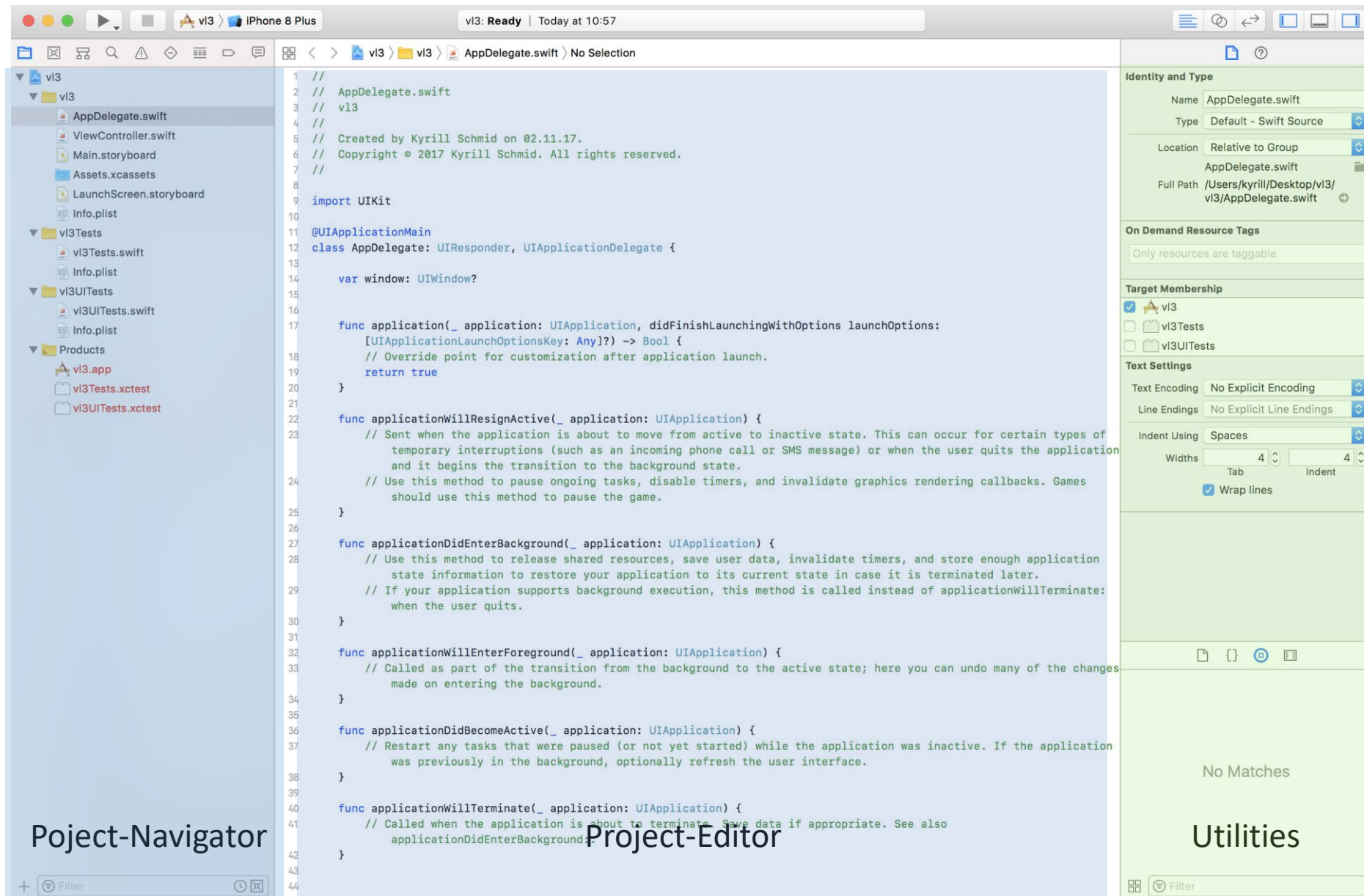
XCode beinhaltet im Grunde schon alles was man zum Entwickeln von iOS-Apps benötigt:

- Software Development Kit (SDK)
- Code Editor
- UI Editor
- Debugging Tools
- Simulator

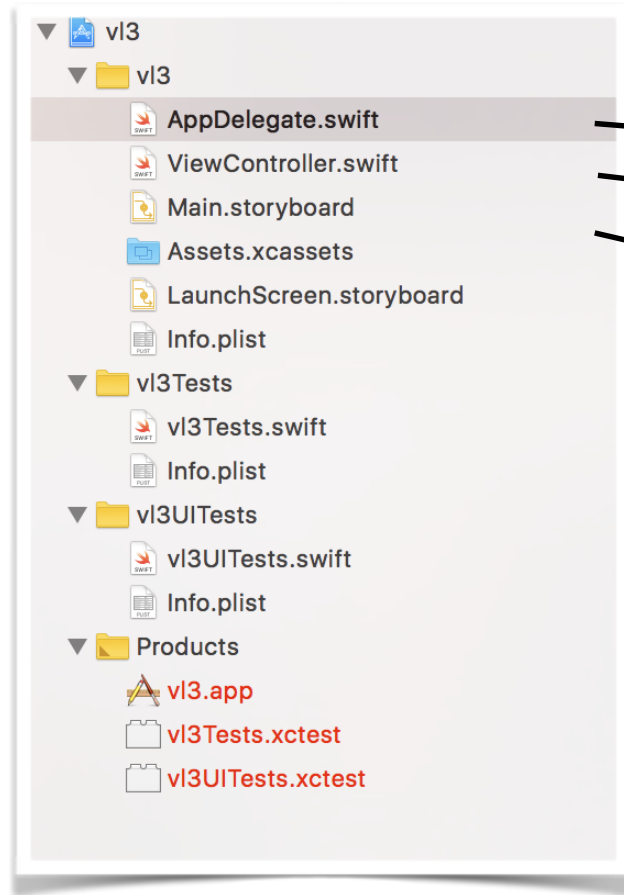


Ein neues Projekt anlegen





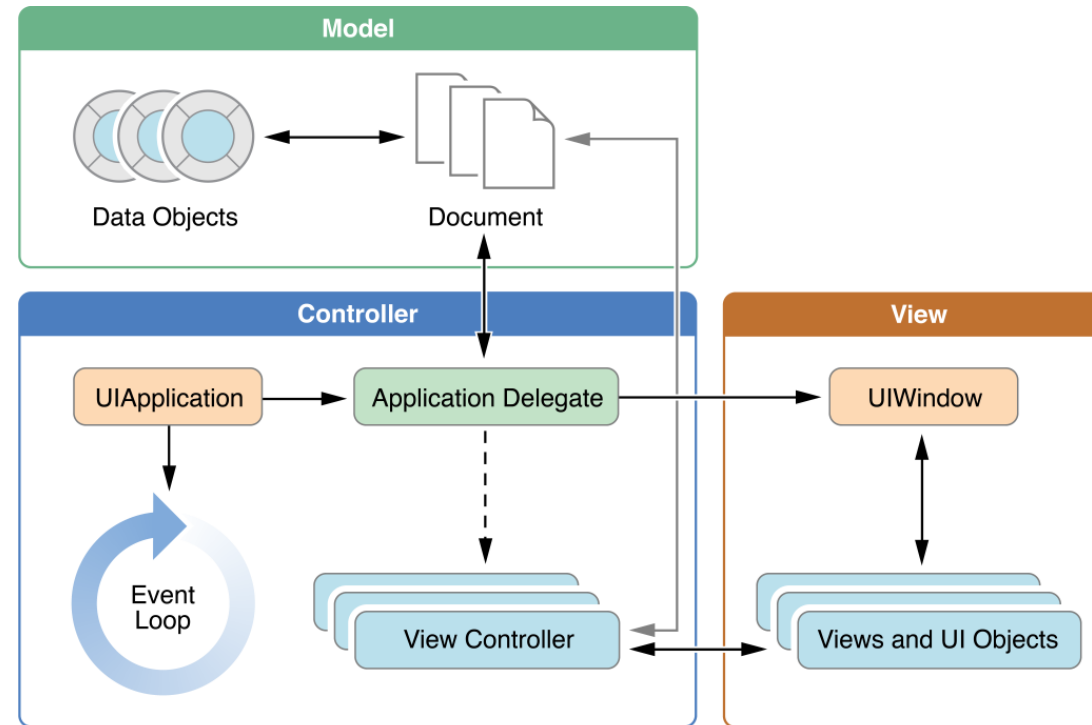
Neues Projekt: Single View Application



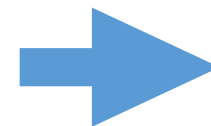
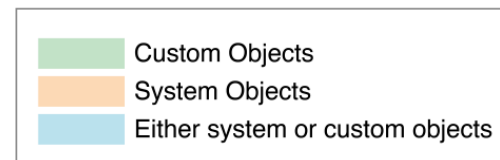
- XCode generiert beim anlegen eines neuen Projekts initiale Files:

- AppDelegate.swift
- ViewController.swift
- Main.storyboard
- ...

Bestandteile einer iOS-App



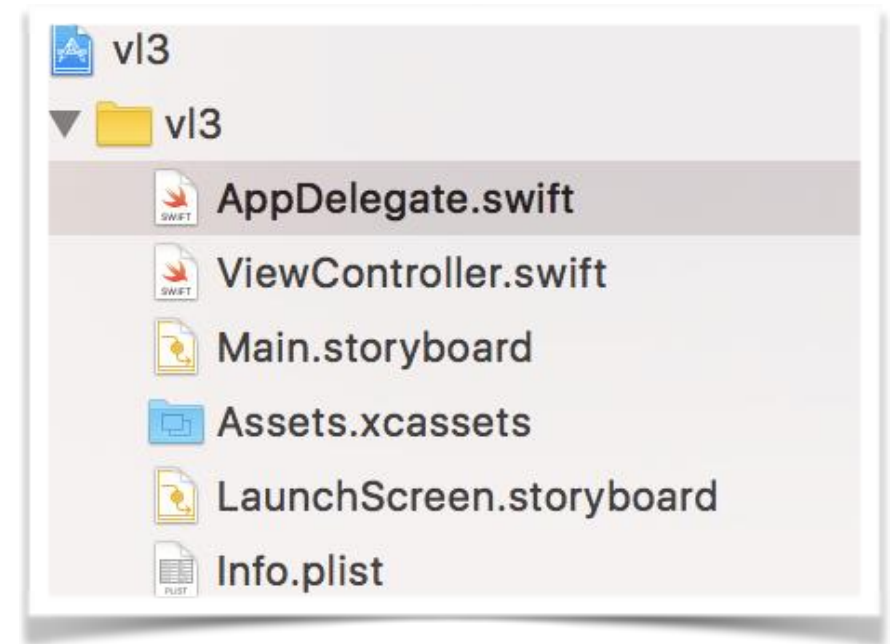
<https://developer.apple.com>



Mehr zum
Model-View-Controller Pattern in Kürze.

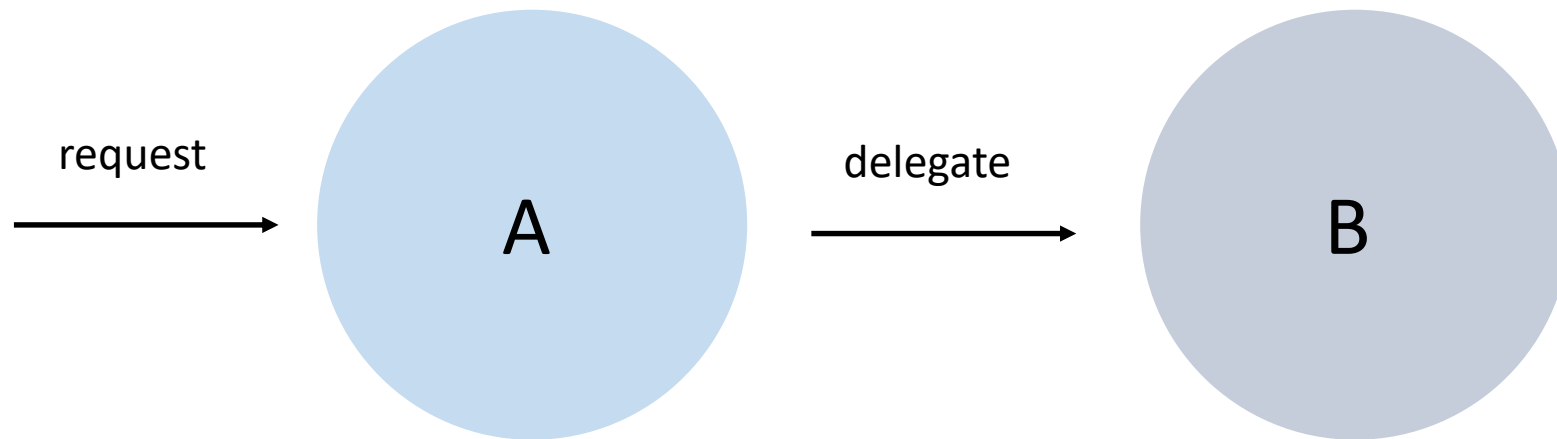
Der AppDelegate

- Der AppDelegate ist ein **Singleton** und das erste customisierbare Objekt einer iOS App.
- AppDelegate ist „Ansprechpartner“ des Betriebssystems für einige wichtige Events.
- AppDelegate ist der **Delegate** der Application.
- Das **Delegation Pattern** ist zentral in iOS und findet an vielen Stellen Verwendung.



Einschub: Das Delegation Pattern

- Delegation ist ein Pattern der Objekt-Orientierten Programmierung um Objekt-Komposition zu unterstützen.
- Bei Delegation behandelt ein Objekt A eine Anfrage indem es an ein weiteres Objekt B delegiert.



- In Swift wird Delegation durch die Verwendung von Protokollen unterstützt.

Protokolle in Swift

- Protokolle sind ähnlich zu Interfaces in z.B. Java.
- Eine Klasse (oder Struktur) kann Protokoll konform sein d.h. die Anforderungen des Protokolls werden umgesetzt.

```
protocol SomeProtocol {  
    // protocol definition goes here  
    func requiredFunction()  
}
```

```
class SomeClass: SomeProtocol{  
    func requiredFunction() {  
        print("Have to do that")  
    }  
}
```

Protokolle und Delegation

```
protocol Assistant {  
    func writeEmail()  
}
```

Protokolle und Delegation

```
protocol Assistant {  
    func writeEmail()  
}  
  
class WorkerA : Assistant {  
    func writeEmail() {  
        print("WorkerA: I wrote an Email!")  
    }  
}
```

Protokolle und Delegation

```
protocol Assistant {  
    func writeEmail()  
}
```

```
class WorkerA : Assistant {  
    func writeEmail() {  
        print("WorkerA: I wrote an Email!")  
    }  
}
```

```
class Boss {  
    var delegate : Assistant?  
  
    func work(){  
        print("Boss: I have no time to write Emails!")  
        delegate?.writeEmail()  
    }  
}
```

Protokolle und Delegation

```
let boss = Boss()  
let worker = WorkerA()  
boss.delegate = worker  
boss.work()
```

// Boss: I have no time to write Emails!

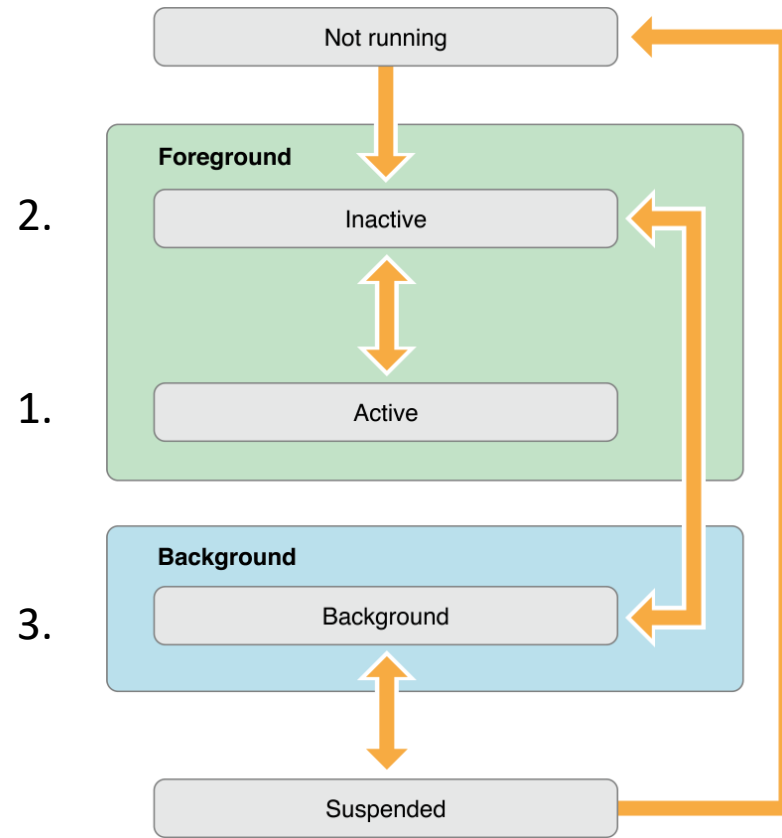
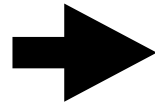
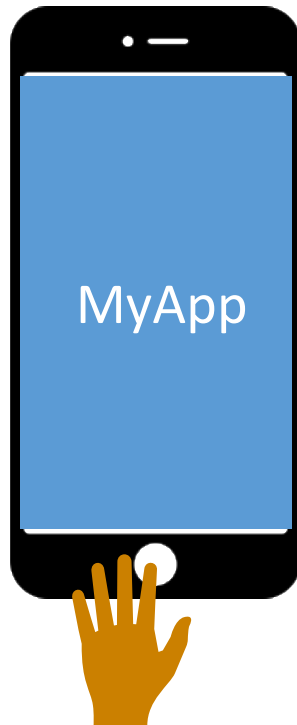
// WorkerA: I wrote an Email!

AppDelegate als Delegate der Application

- AppDelegate ist der **Delegate** der Application.
- Das bedeutet konkret, dass er **UIApplicationDelegate** konform ist.
- Eine wichtige Verantwortlichkeit, die der AppDelegate übernimmt ist das Reagieren auf Zustands-Transitionen der App.

App Zustands-Transition

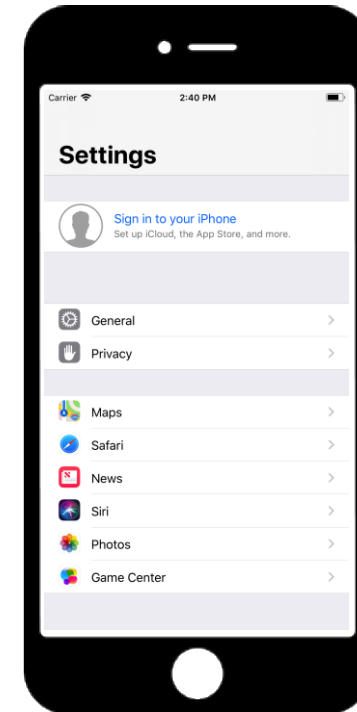
User drückt Home Button



App Zustands-Transition

- Wenn die App in den Zustand ‚Background‘ übergeht müssen u.U. noch...
 - ...Daten gesichert werden.
 - ...Speicher freigegeben werden (reduce memory footprint).
- Die letzte Chance dazu ergibt sich in der **applicationDidEnterBackground(_:)** Methode des AppDelegate

Home Button Press



App Zustands-Transition

Weitere Methoden des AppDelegate:

[applicationWillTerminate\(_ :\)](#)

[application\(_ :willFinishLaunchingWithOptions:\)](#)

[applicationDidBecomeActive\(_ :\)](#)

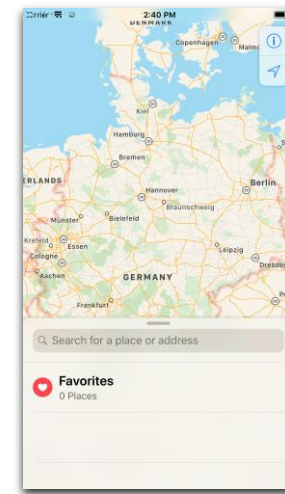
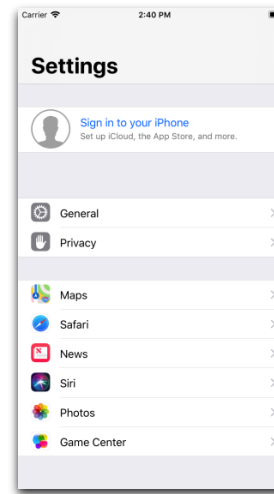
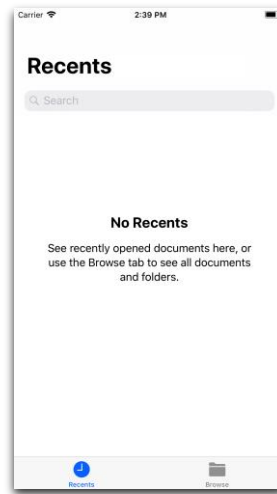
[applicationWillResignActive\(_ :\)](#)

[applicationWillEnterForeground\(_ :\)](#)

[application\(_ :didFinishLaunchingWithOptions:\)](#)

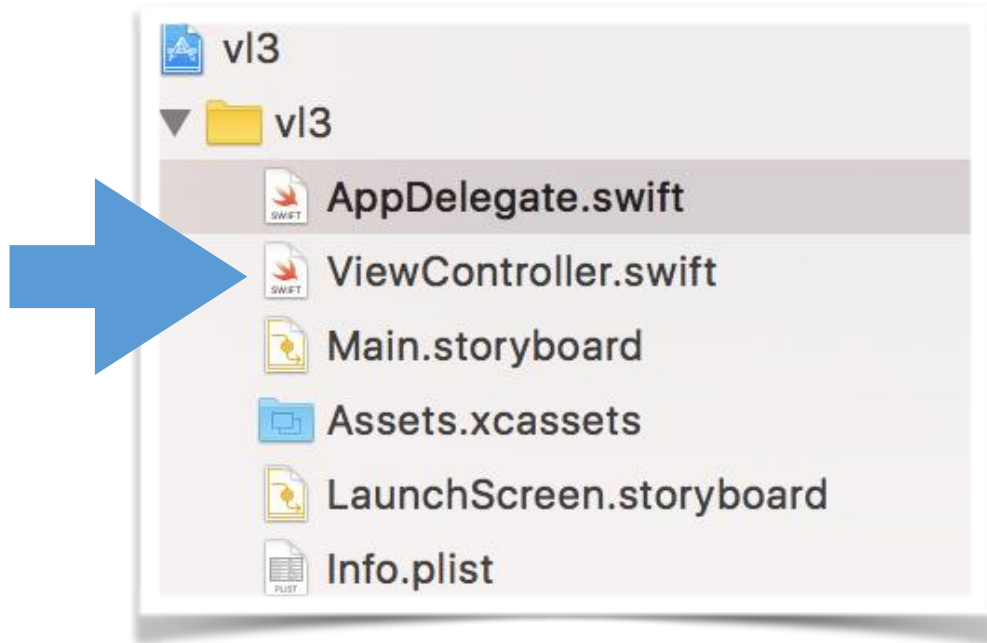
Views

- Eine App besteht aus einer Menge von **Views** durch die der Nutzer navigieren kann.



- **Views** zeigen aber lediglich Daten an.
 - Daten kommen aus Modell und werden vom **ViewController** übermittelt.
 - Benutzerinteraktion wird vom **ViewController** interpretiert und an das **Model** übermittelt.

ViewController



- Jede **View** hat einen zugehörigen **Controller -> ViewController**
- ViewController sind zentrale Objekte für die iOS-Programmierung und übernehmen viele wichtige Aufgaben.
- Geben eine bewährte Struktur vor für die Gestaltung eines Interface.
- Viele verschiedene ViewController Typen, die sofort verwendet werden können (NavigationView, TableView, CollectionView, etc.).

Beispiel: SimpleApp – Live Coding

Einfache App mit grundlegenden UI-Elementen:

- Label
- Button
- Textfield
- ...

