

Full Stack iOS Entwicklung mit Swift

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

Heute

Frameworks
App-Extensions
Today-Widgets

Demo



Frameworks

- Ein Framework ist ein eigenes Verzeichnis oder Xcode-Projekt, welches in andere Projekte eingebunden wird
- Ein Framework ist ein **Bundle mit eigenem Namespace** und **eigener Sichtbarkeit**. Deshalb gilt es als Best Practise seine Typen innerhalb des Frameworks mit einem zweistelligen Prefix (**UILabel**, **NSString**, **CGSize**) zu benennen
- Ein Framework ist ein **eigenes Target** und kann bswp. unabhängig gebaut (Deployment Target), getestet, versioniert und veröffentlicht werden
- Deshalb sind gute Frameworks in sich geschlossen, vollständig und **unabhängig** von einer konkreten Anwendung
- Neben dem Code können Frameworks auch Ressourcen wie Nib- und Sound-Files, Images und sonstige Assets enthalten
- 3rd-Party Libraries sind Frameworks, ebenso wie UIKit, Foundation, UserNotifications, CoreData
- Auch wenn man keine 3rd-Party Library entwickeln (und veröffentlichen) möchte, kann es durchaus sinnvoll sein ein eigenes Framework zu schreiben, um **gemeinsamen Code**, wie Anwendungslogik und bildschirm-unabhängigen Code, **auszulagern** und diese für **unterschiedliche UI-Targets** (Today-Widget, iPad-, Watch-, TV-, Mac-App) zu verwenden
- Nebenbei setzt letzteres gute Anreize, um MVC einzuhalten
- Weitere Informationen sind dem [Framework Programming Guide](#) zu entnehmen



Frameworks

Cryptomarket | Build Cryptomarket: **Succeeded** | Today at 10:48

Choose a template for your new target:

PROJECT Cryptomarket

TARGETS Cryptomarket

Application

- Single View App
- Game
- Augmented Reality App
- Document Based App
- Master-Detail App

Framework & Library

- Cocoa Touch Framework
- Cocoa Touch Static Library
- Metal Library

Rules

Identity and Type

- Name: Cryptomarket
- Location: Relative to Group
- Full Path: /Users/Alex/Developer/Lehre/FSIOS-Instructor/WS17_18/12_framework_toda ywidget/Prep/Cryptomarket/Cryptomarket.xcodeproj

Project Document

- Project Format: Xcode 8.0-compatible
- Organization: Alexander Dobrynin
- Class Prefix:

Text Settings

- Indent Using: Spaces
- Widths: Tab 4, Indent 4
- Wrap lines: checked

Cancel Previous Next

Launch Screen File: LaunchScreen

Embedded Binaries: CryptomarketKit.framework ...in build/Debug-iphoneos

Linked Frameworks and Libraries:

Name	Status
CryptomarketKit.framework	Required
Pods_Cryptomarket.framework	Required



Frameworks

The screenshot shows the Xcode interface with the project 'Cryptomarket' selected. The left sidebar displays the project structure:

- Cryptomarket**
 - ↳ **Cryptomarket**
 - ↳ Utils
 - ↳ Model
 - ↳ View
 - ↳ ViewController
 - ↳ Market Summary
 - ↳ Chart
 - MarketSummaryTableViewController.swift
 - MarketSummaryDashboardViewController.swift
 - MarketSummaryChartViewController.swift
 - Notification
 - Market
 - EmptyDataTableViewController.swift
 - AppDelegate.swift
 - Main.storyboard
 - Supporting Files
 - Info.plist
 - ↳ **CryptomarketKit**
 - ↳ CryptomarketKit.h
 - Info.plist
 - ↳ **CryptomarketKitTests**
 - ↳ Products
 - ↳ Pods
 - ↳ Frameworks
- ↳ **Pods**

The main area shows the **General** tab of the project settings. The **Identity** section includes:

- Display Name: CryptomarketKit
- Bundle Identifier: com.alex-dobrynin.CryptomarketKit
- Version: 1.0
- Build: \$(CURRENT_PROJECT_VERSION)

The **Signing** section has the **Automatically manage signing** checkbox checked. The **Deployment Info** section shows Deployment Target set to 11.2 and Devices set to Universal.

The right side of the screen shows the **Identity and Type**, **Project Document**, and **Text Settings** panes.



Frameworks

The screenshot shows the Xcode interface with the project 'Cryptomarket' selected. The left sidebar displays the project structure:

- Cryptomarket (group)
 - Utils
 - Model
 - View
 - ViewController
 - Market Summary
 - Chart
 - MarketSummaryTableViewController.swift
 - MarketSummaryDashboardViewController.swift
 - MarketSummaryChartViewController.swift
 - Notification
 - Market
 - EmptyDataTableViewController.swift
 - AppDelegate.swift
 - Main.storyboard
 - Supporting Files
 - Info.plist
 - CryptomarketKit (group)
 - CryptomarketKit.h
 - Info.plist
 - Products
 - Pods
 - Frameworks
 - Pods (group)

The main area shows the 'General' tab of the project settings. Key configuration details include:

 - Deployment Info:** Deployment Target is set to 11.2, Devices to Universal, Main Interface to Main. Device Orientation includes Portrait, Landscape Left, and Landscape Right. Status Bar Style is Default.
 - App Icons and Launch Images:** App Icons Source is AppIcon, Launch Images Source is Use Asset Catalog..., Launch Screen File is LaunchScreen.
 - Embedded Binaries:** CryptomarketKit.framework is listed.
 - Linked Frameworks and Libraries:** CryptomarketKit.framework and Pods_Cryptomarket.framework are listed with Required status.

The right sidebar contains detailed project information:

 - Identity and Type:** Name is Cryptomarket, Location is Relative to Group, Full Path is /Users/Alex/Developer/Lehre/FSIOS-Instructor/WS17_18/12_framework_tutorial/Prep/Cryptomarket/Cryptomarket.xcodeproj.
 - Project Document:** Project Format is Xcode 8.0-compatible, Organization is Alexander Dobrynin, Class Prefix is.
 - Text Settings:** Indent Using is Spaces, Widths are 4 and 4, Tab and Indent are checked, Wrap lines is checked.



Frameworks

- Ein Framework ist ein eigenes Verzeichnis oder Xcode-Projekt, welches in andere Projekte eingebunden wird
- Ein Framework ist ein **Bundle mit eigenem Namespace** und **eigener Sichtbarkeit**. Deshalb gilt es als Best Practise seine Typen innerhalb des Frameworks mit einem zweistelligen Prefix (**UILabel**, **NSString**, **CGSize**) zu benennen
- Ein Framework ist ein **eigenes Target** und kann bswp. unabhängig gebaut (Deployment Target), getestet, versioniert und veröffentlicht werden
- Deshalb sind gute Frameworks in sich geschlossen, vollständig und **unabhängig** von einer konkreten Anwendung
- Neben dem Code können Frameworks auch Ressourcen wie Nib- und Sound-Files, Images und sonstige Assets enthalten
- 3rd-Party Libraries sind Frameworks, ebenso wie UIKit, Foundation, UserNotifications, CoreData
- Auch wenn man keine 3rd-Party Library entwickeln (und veröffentlichen) möchte, kann es durchaus sinnvoll sein ein eigenes Framework zu schreiben, um **gemeinsamen Code**, wie Anwendungslogik und bildschirm-unabhängigen Code, **auszulagern** und diese für **unterschiedliche UI-Targets** (Today-Widget, iPad-, Watch-, TV-, Mac-App) zu verwenden
- Nebenbei setzt letzteres gute Anreize, um MVC einzuhalten
- Weitere Informationen sind dem [Framework Programming Guide](#) zu entnehmen



Frameworks

The screenshot shows the Xcode interface with the following details:

- Top Bar:** Shows the project name "CryptomarketKit" and target "iPhone 8".
- Status Bar:** Displays "Cryptomarket | Build CryptomarketKit: Failed | Today at 10:53".
- Left Sidebar (File Browser):** Shows the project structure under "Cryptomarket". A large red oval surrounds this entire sidebar area.
- Code Editor:** Displays the file "ModelType.swift" with the following content:

```
1 //  
2 // ModelType.swift  
3 // Cryptomarket  
4 //  
5 // Created by Alex on 17.01.18.  
6 // Copyright © 2018 Alexander Dobrynin. All rights reserved.  
7 //  
8  
9 import Foundation  
10  
11 enum ModelType {  
12     case marketSummary(market: Market)  
13     case markets  
14 }  
15
```
- Right Sidebar (Identity and Type):** Shows the file's properties:
 - Name: ModelType.swift
 - Type: Default - Swift Source
 - Location: Relative to Group
 - Full Path: /Users/Alex/Developer/Lehra/FSIOS-Instructor/WS17_18/12_framework_todawidget/Prep/Cryptomarket/CryptomarketKit/Model/ModelType.swift
- Target Membership:** Shows the file is part of the "Cryptomarket" target, indicated by a checked checkbox.
- Text Settings:** Includes options for Text Encoding, Line Endings, Indent Using (Spaces), and Wrap lines.



Frameworks

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure under the `Cryptomarket` project. A red arrow points to the `CryptomarketKit` folder.
- Editor:** Displays the `ModelType.swift` file content:

```
1 //  
2 // ModelType.swift  
3 // Cryptomarket  
4 //  
5 // Created by Alex on 17.01.18.  
6 // Copyright © 2018 Alexander Dobrynin. All rights reserved.  
7 //  
8  
9 import Foundation  
10  
11 enum ModelType {  
12     case marketSummary(market: Market)  
13     case markets  
14 }  
15
```
- Build Bar:** Shows "Cryptomarket | Build CryptomarketKit: **Succeeded** | Today at 10:53".
- Identity and Type:** Shows the file is a Default - Swift Source file located in the `Model` folder of the `CryptomarketKit` target.
- Target Membership:** Shows the file is part of the `CryptomarketKit` target, indicated by a checked checkbox.
- Text Settings:** Shows standard text encoding settings.



Frameworks

- Ein Framework ist ein eigenes Verzeichnis oder Xcode-Projekt, welches in andere Projekte eingebunden wird
- Ein Framework ist ein **Bundle mit eigenem Namespace** und **eigener Sichtbarkeit**. Deshalb gilt es als Best Practise seine Typen innerhalb des Frameworks mit einem zweistelligen Prefix (**UILabel**, **NSString**, **CGSize**) zu benennen
- Ein Framework ist ein **eigenes Target** und kann bswp. unabhängig gebaut (Deployment Target), getestet, versioniert und veröffentlicht werden
- Deshalb sind gute Frameworks in sich geschlossen, vollständig und **unabhängig** von einer konkreten Anwendung
- Neben dem Code können Frameworks auch Ressourcen wie Nib- und Sound-Files, Images und sonstige Assets enthalten
- 3rd-Party Libraries sind Frameworks, ebenso wie UIKit, Foundation, UserNotifications, CoreData
- Auch wenn man keine 3rd-Party Library entwickeln (und veröffentlichen) möchte, kann es durchaus sinnvoll sein ein eigenes Framework zu schreiben, um **gemeinsamen Code**, wie Anwendungslogik und bildschirm-unabhängigen Code, **auszulagern** und diese für **unterschiedliche UI-Targets** (Today-Widget, iPad-, Watch-, TV-, Mac-App) zu verwenden
- Nebenbei setzt letzteres gute Anreize, um MVC einzuhalten
- Weitere Informationen sind dem [Framework Programming Guide](#) zu entnehmen



Frameworks

The screenshot shows the Xcode interface with a red circle highlighting the status bar at the top. The status bar displays "Cryptomarket | Build Cryptomarket: Failed | Today at 10:54" with 7 warnings and 29 errors. The main area shows a Swift file, MarketsTableViewController.swift, with several code snippets and error annotations. A red circle highlights the "Target Membership" section in the Utilities panel on the right, which lists "Cryptomarket" as the selected target. The Utilities panel also contains sections for "Identity and Type", "On Demand Resource Tags", and "Text Settings".

```
1 // MarketsTableViewController.swift
2 // Cryptomarket
3 //
4 // Created by Alex on 05.01.18.
5 // Copyright © 2018 Alexander Dobrynin. All rights reserved.
6 //
7 //
8
9 import UIKit
10
11 class MarketsTableViewController: UITableViewController {
12
13     private struct Storyboard {
14         static let MarketSummarySegueIdentifier = "ShowMarketSummarySegue"
15         static let NotificationsSegueIdentifier = "Show Notifications"
16     }
17
18     // MARK: - Model
19
20     // var markets: [(key: String, value: [Market])] = []
21     var markets: [(key: String, value: [(market: Market, summaries: Int)])] = [] ! Use of undeclared type 'Market'
22     didSet { tableView.reloadData() }
23
24
25     // MARK: - Private Properties
26
27     private let service = MarketService.instance() ! Use of unresolved identifier 'MarketService'
28
29     // MARK: - ViewController Lifecycle
30
31     override func viewDidLoad() {
32         super.viewDidLoad()
33
34         tableView.register(
35             UINib(nibName: MarketTableViewCell.Identifier, bundle: nil),
36             forCellReuseIdentifier: MarketTableViewCell.Identifier
37         )
38
39         service.model(from: .both, completion: handle)
40     }
41
42     var marketSummaryObserver: NSObjectProtocol?
43
44     override func viewDidAppear(_ animated: Bool) {
45         super.viewDidAppear(animated)
```



Frameworks

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure with "Cryptomarket" as the main target.
- Editor:** Displays the content of `Market.swift`. The code defines a `Market` struct with properties: `baseCurrency`, `logoUrl`, `currency`, `currencyLong`, `name`, and `active`.
- Build Issues:** The left sidebar shows two sections:
 - Charts:** 7 Swift Compiler Warnings related to deprecated 'characters' usage.
 - Cryptomarket:** 28 Swift Compiler Errors, mostly due to undeclared types and internal protection level issues.
- Identity and Type:** Inspector panel on the right shows the file is named `Market.swift`, has type `Default - Swift Source`, and is located at `/Users/Alex/Developer/Lehra/FSIOS-Instructor/WS17_18/12_framework_todawidget/Prep/Cryptomarket/CryptomarketKit/Model/Market/Market.swift`.
- Target Membership:** The `Cryptomarket` target is selected (indicated by a red circle).
- Text Settings:** Inspector panel shows text encoding is set to `No Explicit Encoding`.



Frameworks

Running Cryptomarket on iPhone 8

```
final class Diskdatabase: Database {
    static let MarketSummaryDataNotificationKey = "MarketSummaryDataNotificationKey"
    private let url: URL
    private let encoder: JSONEncoder
    private let decoder: JSONDecoder
    private let shouldPostNotification: Bool

    static func with(model: ModelType) -> Diskdatabase? {
        guard let base = try? FileManager.default.url(for: .documentDirectory, in: .userDomainMask, appropriateFor: nil,
            create: true) else { return nil }

        return Diskdatabase(
            url: base.appendingPathComponent(model.identifier).appendingPathExtension(model.fileType),
            encoder: defaultJsonEncoder,
            decoder: defaultJsonDecoder,
            shouldPostNotification: model.shouldPostNotification
        )
    }

    private init(url: URL, encoder: JSONEncoder, decoder: JSONDecoder, shouldPostNotification: Bool) {
        self.url = url
        self.encoder = encoder
        self.decoder = decoder
        self.shouldPostNotification = shouldPostNotification
    }

    func getAll<Model>() -> [Model] where Model : UniqueEntityiy {
        let data = try? Data(contentsOf: url)
        let decoded = data.flatMap { try? decoder.decode([Model].self, from: $0) }

        return decoded ?? []
    }

    func insert<Model>(_ model: Model) -> [Model] where Model : UniqueEntitiy {
        var existing: [Model] = getAll()
        existing.append(model)

        override(with: existing)
        return existing
    }

    func overrideModels<(with models: [Model]) where Model : UniqueEntitiy>
}
```

Identity and Type

- Name: DiskDatabase.swift
- Type: Default - Swift Source
- Location: Relative to Group
- Full Path: /Users/Alex/Developer/Lehra/FSIOS-Instructor/WS17_18/12_framework_todawidget/Prep/Cryptomarket/CryptomarketKit/Model/Storage/DiskDatabase.swift

On Demand Resource Tags

- Only resources are taggable

Target Membership

- Cryptomarket
- CryptomarketKit
- CryptomarketKitTests

Text Settings

- Text Encoding: No Explicit Encoding
- Line Endings: No Explicit Line Endings
- Indent Using: Spaces
- Widths: 4 Tab Indent
- Wrap lines



Frameworks

Running Cryptomarket on iPhone 8

```
1 // Networking.swift
2 // Cryptomarket
3 // Created by Alex on 05.01.18.
4 //
5 // Copyright © 2018 Alexander Dobrynin. All rights reserved.
6
7 import Foundation
8
9 fileprivate let base = "https://bittrex.com/api/v1.1/public/"
10
11 fileprivate extension ModelType {
12     var url: URL {
13         switch self {
14             case .markets: return URL(string: "\(base)getmarkets")!
15             case .marketSummary(let market): return URL(string: "\(base)getmarketsummary?market=\(market.name)")!
16         }
17     }
18
19     var method: String {
20         switch self {
21             case .markets, .marketSummary: return "GET"
22         }
23     }
24 }
25
26
27 func decode<T>(data: Data, withJsonDecoder decoder: JSONDecoder) throws -> T {
28     switch self {
29         case .markets: return try decoder.decode(MarketResult.self, from: data) as! T
30         case .marketSummary: return try decoder.decode(MarketSummaryResult.self, from: data) as! T
31     }
32 }
33
34
35 final class Webservice {
36     private let model: ModelType
37     private let decoder: JSONDecoder
38
39     init(model: ModelType, decoder: JSONDecoder = JSONDecoder()) {
40         self.model = model
41         self.decoder = decoder
42     }
43
44     func get<T>(completion: @escaping (Result<T>) -> Void) {
45 }
```

Identity and Type

- Name: Networking.swift
- Type: Default - Swift Source
- Location: Relative to Group
- Full Path: /Users/Alex/Developer/Lehra/FSIOS-Instructor/WS17_18/12_framework_todayswidget/Prep/Cryptomarket/CryptomarketKit/Model/Networking/Networking.swift

On Demand Resource Tags

Target Membership

- Cryptomarket
- CryptomarketKit
- CryptomarketKitTests

Text Settings

- Text Encoding: No Explicit Encoding
- Line Endings: No Explicit Line Endings
- Indent Using: Spaces
- Widths: Tab 4 Indent 4
- Wrap lines



Frameworks

Running Cryptomarket on iPhone 8

```
10 public final class MarketSummaryService: DataProvider {
11
12     private let market: Market
13     private let webservice: Webservice
14     private let database: Database?
15
16
17     public static func with(market: Market) -> MarketSummaryService {
18         let model = ModelType.marketSummary(market: market)
19
20         return MarketSummaryService(
21             market: market,
22             webservice: Webservice(model: model, decoder: defaultJsonDecoder),
23             database: Diskdatabase.with(model: model)
24         )
25     }
26
27     private init(market: Market, webservice: Webservice, database: Database?) {
28         self.market = market
29         self.webservice = webservice
30         self.database = database
31     }
32
33     public func model(from source: ModelSource, completion: @escaping (Result<[MarketSummary]>) -> Void) {
34         switch source {
35             case .network: fromNetwork(completion)
36             case .database: fromDatabase(completion)
37             case .both: fromDatabase(completion); fromNetwork(completion)
38         }
39     }
40
41     private func fromDatabase(_ completion: (Result<[MarketSummary]>) -> Void) {
42         if let local: [MarketSummary] = database?.getAll() { // return from db first
43             debugPrint(#function, "local", local.count)
44             completion(.success(local))
45         }
46     }
47
48     private func fromNetwork(_ completion: @escaping (Result<[MarketSummary]>) -> Void) {
49         webservice.get { (result: Result<MarketSummaryResult>) in
50             switch result {
51                 case let .failure(e):
52                     completion(.failure(e))
53
54                 case let .success(s) where !s.success:
```

Identity and Type

- Name: MarketSummaryService.swift
- Type: Default - Swift Source
- Location: Relative to Group
- Full Path: /Users/Alex/Developer/Lehre/FSIOS-Instructor/WS17_18/12_framework_todawidget/Prep/Cryptomarket/CryptomarketKit/Model/Market Summary/MarketSummaryService.swift

On Demand Resource Tags

Only resources are taggable

Target Membership

- Cryptomarket (unchecked)
- CryptomarketKit (checked)
- CryptomarketKitTests (unchecked)

Text Settings

- Text Encoding: No Explicit Encoding
- Line Endings: No Explicit Line Endings
- Indent Using: Spaces
- Widths: Tab 4 Indent 4
- Wrap lines (checked)

Heute

Frameworks
App-Extensions
Today-Widgets

Demo



App-Extensions

- App-Extensions erweitern die Fähigkeiten einer App indem **ausgewählte Funktionen innerhalb anderer Interaktionskontexten zugänglich werden**. Die eigene App kann **eingebettet** werden, wodurch der Benutzer bspw.
 - Daten aus einer App in die eigene App importieren kann und vice versa
 - bestimmte Teile der eigenen App in iMessage, Maps oder anderen Apps verwenden kann
 - über die Suche in Spotlight innerhalb der eignen App suchen kann
 - zugelassene Anfragen über Siri formulieren und an die eigene App senden kann
- Eine **Host-App** bezeichnet jene App, die den Kontext für eine App-Extensions bereitstellt und diese einbettet
- App-Extensions sind ein **zusätzlicher optionaler Dienst**, die mit der eigentlichen App ausgeliefert werden. Während die App unabhängig von allen Extensions gebaut, veröffentlicht und verwendet werden kann, gilt das nicht für die App-Extensions (im Gegensatz zu Frameworks)
- Da App-Extensions überschaubare Einheiten sind, sollte die UX auf eine schnelle, direkt zugängliche und fokussierte Durchführung einer Aufgabe abzielen
- (Embedded) Frameworks eignen sich ideal dafür, um **gemeinsamen Code zwischen App und App-Extension zu teilen**
- App und App-Extensions können untereinander kommunizieren und Daten (UserDefaults, Disk) austauschen*
*Hierfür muss die Capability "App Groups" aktiviert werden. Benötigt ein Apple Developer Account und App-Signing
 - Anschließend sind die APIs zu verwenden, die einen **App-Group-Identifier** entgegennehmen

```
let appGroupIdentifier = "group.de.fhkoeln.inf.adv.fsios.YourAppName"
let url = FileManager.default.containerURL(forSecurityApplicationGroupIdentifier: appGroupIdentifier)
let defaults = UserDefaults(suiteName: appGroupIdentifier)
```

Essentials

App Extensions Increase Your Impact

Understand How an App Extension Works

Creating an App Extension

Handling Common Scenarios

App Extension Types

Revision History

There Are Several Types of App Extensions

iOS and macOS define several types of app extensions, each of which is tied to a single, well-scoped area of the system, such as sharing, Notification Center, and the iOS keyboard. A system area that enables extensions is called an *extension point*. Each extension point defines usage policies and provides APIs that you use when you create an app extension for that area. You choose an extension point based on the functionality you want to provide.

Table 1-1 lists the extension points in iOS and macOS and gives an example of tasks you might enable in an app extension for each extension point.

Table 1-1 Extension points on Apple platforms

Extension point	Typical app extension functionality
Action (iOS and macOS; UI and non-UI variants)	Manipulate or view content originating in a host app.
Audio Unit (iOS and macOS; UI and non-UI variants)	Generates an audio stream to send to a host app, or modifies an audio stream from a host app and sends it back to the host app.
Broadcast UI (iOS and tvOS)	
Broadcast Upload (iOS and tvOS)	
Call Directory (iOS)	Identify and block incoming callers by their phone number. To learn more, see CallKit Framework Reference .
Content Blocker (iOS and macOS)	Indicate to WebKit that your content-blocking app has updated its rules. (This app extension has no user interface.)
Custom Keyboard (iOS)	Replace the iOS system keyboard with a custom keyboard for use in all apps.
Document Provider (iOS; UI and non-UI variants)	Provide access to and manage a repository of files.
Finder Sync (macOS)	Present information about file sync state directly in Finder. (This app extension has no user interface.)
Game App (watchOS)	Provide a game app for Apple Watch, as described in App Programming Guide for watchOS . (The Game App template is a version of the WatchKit App template, configured for game content.)
iMessage (iOS)	Interact with the Messages app. To learn more, see Messages .

Revision History	Game App (watchOS)	watchOS . (The Game App template is a version of the WatchKit App template, configured for game content.)
	iMessage (iOS)	Interact with the Messages app. To learn more, see Messages .
	Intents (iOS)	Handle tasks related to supporting Siri integration with your app. To learn more, see SiriKit Programming Guide .
	Intents UI (iOS)	Customize the Siri or Maps interface after handling a task related to supporting Siri integration with your app. To learn more, see SiriKit Programming Guide .
	Notification Content (iOS)	
	Notification Service (iOS)	
	Photo Editing (iOS and macOS)	Edit a photo or video within the Photos app.
	Share (iOS and macOS)	Post to a sharing website or share content with others.
	Smart Card Token (macOS)	
	Spotlight Index (iOS)	Index content within your app while it isn't running. To learn more, see Index App Content .
	Sticker Pack (iOS)	Provide a set of stickers that users can use within the Messages app. To learn more, see Messages .
	Today (iOS and macOS)	Get a quick update or perform a quick task in the Today view of Notification Center. (A Today extension is called a <i>widget</i> .)
	TV Services (tvOS)	
	VPN (iOS and macOS)	Create clients for your business's custom, remote-access VPN servers using the Packet Tunnel Provider or App Proxy Provider extension points. Create content filtering for managed devices, such as for school environments, using the Filter Control Provider and Filter Data Provider extension points.
	WatchKit App (watchOS)	Provide an app or a notification UI for Apple Watch, as described in App Programming Guide for watchOS .
	Xcode Source Editor (macOS)	

Feedback



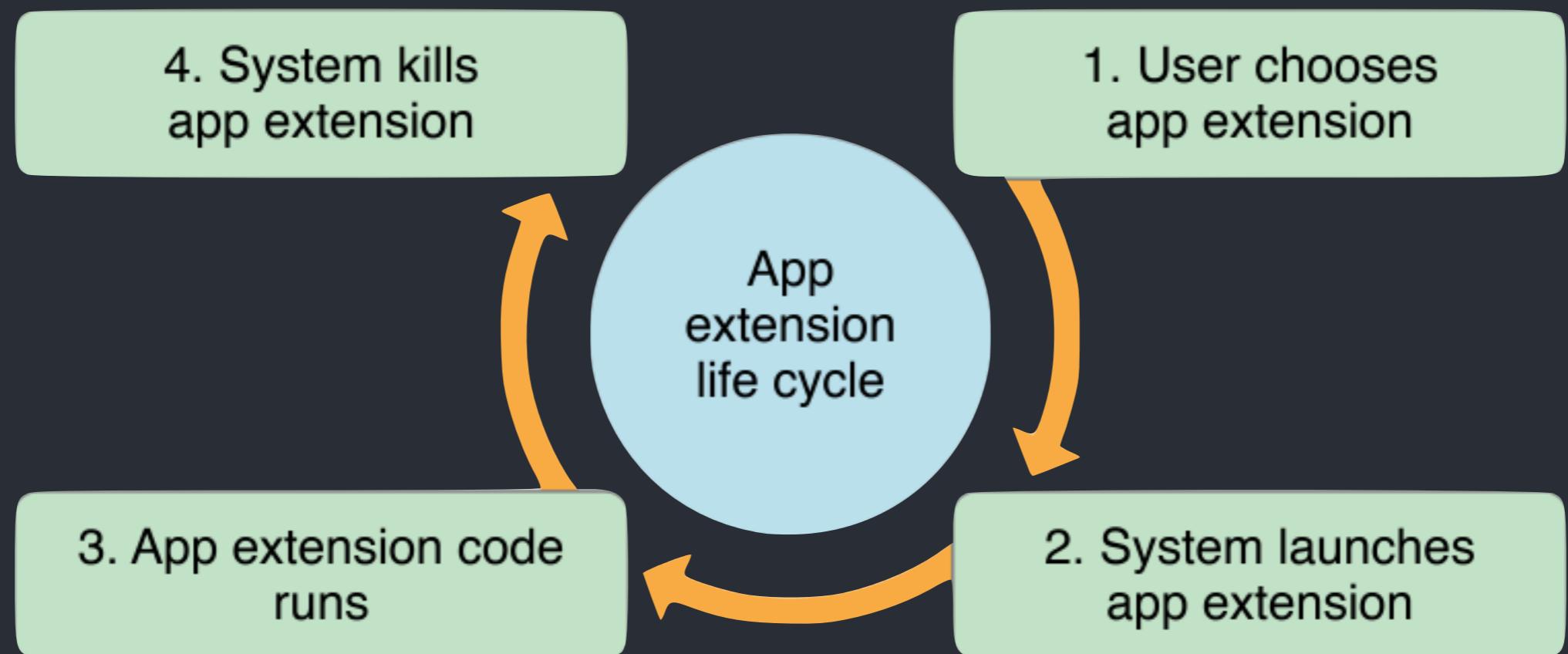
App-Extensions

- App-Extensions erweitern die Fähigkeiten einer App indem **ausgewählte Funktionen innerhalb anderer Interaktionskontexten zugänglich werden**. Die eigene App kann **eingebettet** werden, wodurch der Benutzer bspw.
 - Daten aus einer App in die eigene App importieren kann und vice versa
 - bestimmte Teile der eigenen App in iMessage, Maps oder anderen Apps verwenden kann
 - über die Suche in Spotlight innerhalb der eignen App suchen kann
 - zugelassene Anfragen über Siri formulieren und an die eigene App senden kann
- Eine **Host-App** bezeichnet jene App, die den Kontext für eine App-Extensions bereitstellt und diese einbettet
- App-Extensions sind ein **zusätzlicher optionaler Dienst**, die mit der eigentlichen App ausgeliefert werden. Während die App unabhängig von allen Extensions gebaut, veröffentlicht und verwendet werden kann, gilt das nicht für die App-Extensions (im Gegensatz zu Frameworks)
- Da App-Extensions überschaubare Einheiten sind, sollte die UX auf eine schnelle, direkt zugängliche und fokussierte Durchführung einer Aufgabe abzielen
- (Embedded) Frameworks eignen sich ideal dafür, um **gemeinsamen Code zwischen App und App-Extension zu teilen**
- App und App-Extensions können untereinander kommunizieren und Daten (UserDefaults, Disk) austauschen*
*Hierfür muss die Capability "App Groups" aktiviert werden. Benötigt ein Apple Developer Account und App-Signing
 - Anschließend sind die APIs zu verwenden, die einen **App-Group-Identifier** entgegennehmen

```
let appGroupIdentifier = "group.de.fhkoeln.inf.adv.fsios.YourAppName"
let url = FileManager.default.containerURL(forSecurityApplicationGroupIdentifier: appGroupIdentifier)
let defaults = UserDefaults(suiteName: appGroupIdentifier)
```



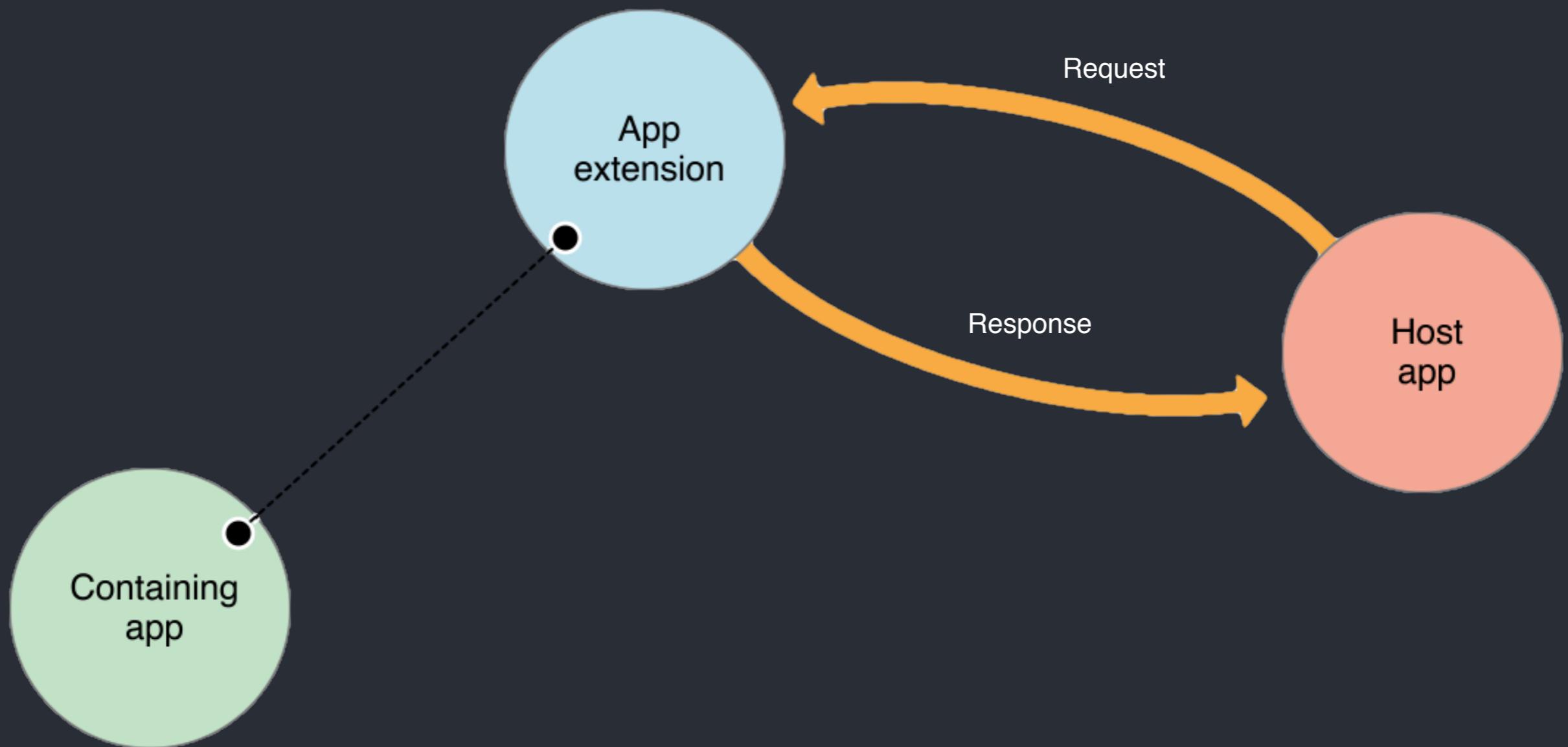
App-Extensions



Modifiziert übernommen aus https://developer.apple.com/library/content/documentation/General/Conceptual/ExtensibilityPG/ExtensionOverview.html#/apple_ref/doc/uid/TP40014214-CH2-SW2



App-Extensions



Modifiziert übernommen aus https://developer.apple.com/library/content/documentation/General/Conceptual/ExtensibilityPG/ExtensionOverview.html#/apple_ref/doc/uid/TP40014214-CH2-SW2



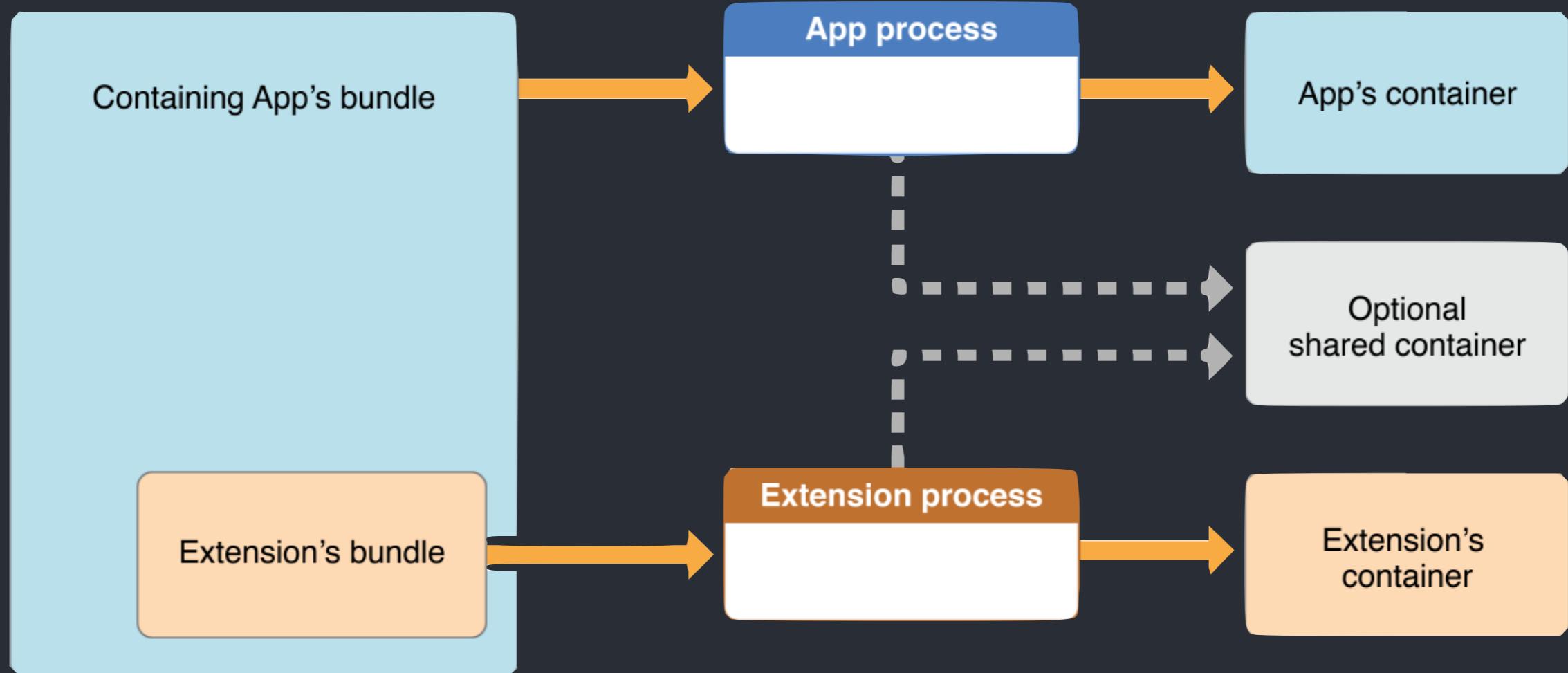
App-Extensions

- App-Extensions erweitern die Fähigkeiten einer App indem **ausgewählte Funktionen innerhalb anderer Interaktionskontexten zugänglich werden**. Die eigene App kann **eingebettet** werden, wodurch der Benutzer bspw.
 - Daten aus einer App in die eigene App importieren kann und vice versa
 - bestimmte Teile der eigenen App in iMessage, Maps oder anderen Apps verwenden kann
 - über die Suche in Spotlight innerhalb der eignen App suchen kann
 - zugelassene Anfragen über Siri formulieren und an die eigene App senden kann
- Eine **Host-App** bezeichnet jene App, die den Kontext für eine App-Extensions bereitstellt und diese einbettet
- App-Extensions sind ein **zusätzlicher optionaler Dienst**, die mit der eigentlichen App ausgeliefert werden. Während die App unabhängig von allen Extensions gebaut, veröffentlicht und verwendet werden kann, gilt das nicht für die App-Extensions (im Gegensatz zu Frameworks)
- Da App-Extensions überschaubare Einheiten sind, sollte die UX auf eine schnelle, direkt zugängliche und fokussierte Durchführung einer Aufgabe abzielen
- (Embedded) Frameworks eignen sich ideal dafür, um **gemeinsamen Code zwischen App und App-Extension zu teilen**
- App und App-Extensions können untereinander kommunizieren und Daten (UserDefaults, Disk) austauschen*
*Hierfür muss die Capability "App Groups" aktiviert werden. Benötigt ein Apple Developer Account und App-Signing
 - Anschließend sind die APIs zu verwenden, die einen **App-Group-Identifier** entgegennehmen

```
let appGroupIdentifier = "group.de.fhkoeln.inf.adv.fsios.YourAppName"
let url = FileManager.default.containerURL(forSecurityApplicationGroupIdentifier: appGroupIdentifier)
let defaults = UserDefaults(suiteName: appGroupIdentifier)
```



App-Extensions



Modifiziert übernommen aus https://developer.apple.com/library/content/documentation/General/Conceptual/ExtensibilityPG/ExtensionScenarios.html#/apple_ref/doc/uid/TP40014214-CH21-SW1

Heute

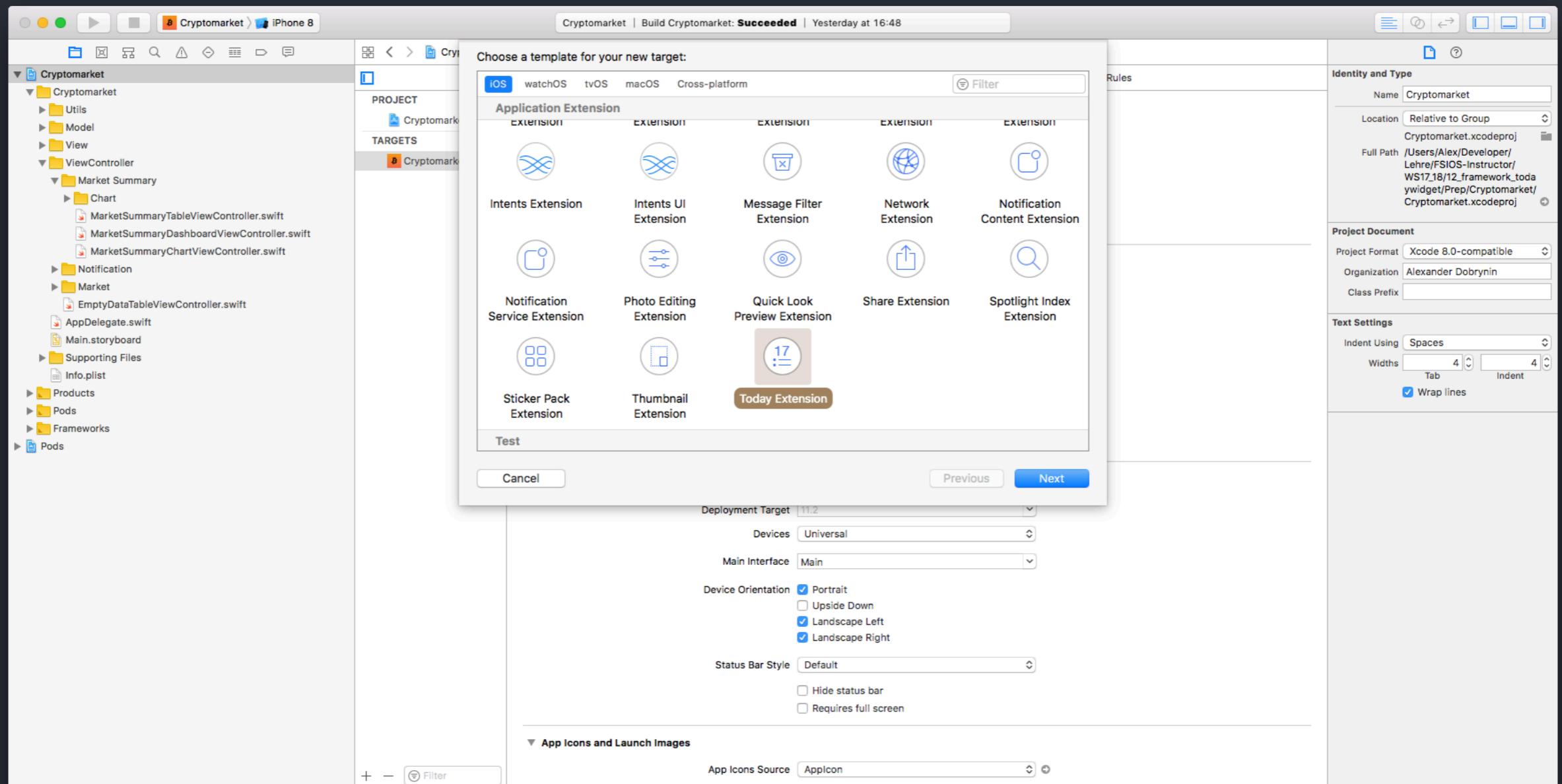
Frameworks
App-Extensions
Today-Widgets

Demo

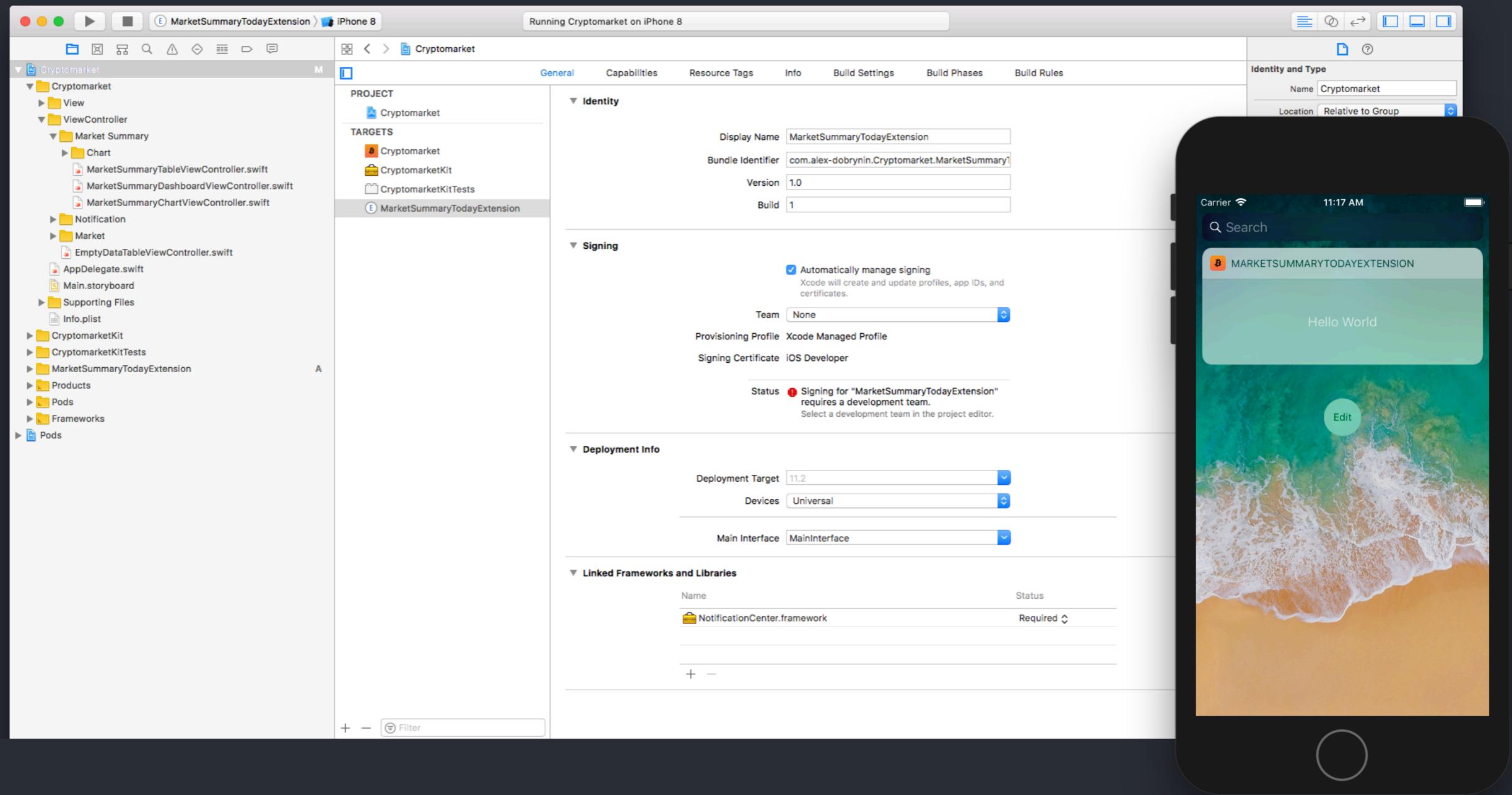
Today-Widgets

- Today-Widgets zeigen den **neusten/wichtigsten Inhalt für die aktuelle (heutige) Zeit** an oder reagieren auf eine **schnelle (typische) Aktion** der App
 - Es können mehrere Today-Widgets (oder Extensions im Allgemeinen) mit der App ausgeliefert werden
 - Der Benutzer kann diese entweder als Widget hinzufügen (linker Homescreen) oder mit Force-Touch auf dem App-Icon einsehen
 - Today-Widgets haben zwei **Display-Modes**
 - **Compact** (default) zeigt die wichtigsten Informationen in einem Widget mit einer Höhe von **110pt** an
 - **Expanded** gibt die Möglichkeit auf “Show More”, wodurch das Widget bis zu einer Höhe von **528pt*** erweitert wird
 - Gerade Today-Widgets sind auf die Kommunikation mit der eigentlichen App angewiesen, weil der zu präsentierende Inhalt häufig von einer Benutzereinstellung abhängt
 - Zudem wird der **gemeinsame Code benötigt**, um bspw. eine entsprechende Netzwerkkabfrage zu starten und das Ergebnis in die gemeinsame Datenbasis zu schreiben
 - Des Weiteren kann ein Tap auf das Today-Widget einen **bestimmten ViewController der App öffnen***
- * Das Öffnen erfolgt über URLs. Die App muss ein zuerst URL-Schema registrieren, um darauf zu reagieren

Today-Widgets



Today-Widgets



Today-Widgets

- Today-Widgets zeigen den **neusten/wichtigsten Inhalt** für die aktuelle (heutige) **Zeit** an oder reagieren auf eine **schnelle (typische) Aktion** der App
 - Es können mehrere Today-Widgets (oder Extensions im Allgemeinen) mit der App ausgeliefert werden
 - Der Benutzer kann diese entweder als Widget hinzufügen (linker Homescreen) oder mit Force-Touch auf dem App-Icon einsehen
 - Today-Widgets haben zwei **Display-Modes**
 - **Compact** (default) zeigt die wichtigsten Informationen in einem Widget mit einer Höhe von **110pt** an
 - **Expanded** gibt die Möglichkeit auf “Show More”, wodurch das Widget bis zu einer Höhe von **528pt*** erweitert wird
 - Gerade Today-Widgets sind auf die Kommunikation mit der eigentlichen App angewiesen, weil der zu präsentierende Inhalt häufig von einer Benutzereinstellung abhängt
 - Zudem wird der **gemeinsame Code benötigt**, um bspw. eine entsprechende Netzwerkkabfrage zu starten und das Ergebnis in die gemeinsame Datenbasis zu schreiben
 - Des Weiteren kann ein Tap auf das Today-Widget einen **bestimmten ViewController der App öffnen***
- * Das Öffnen erfolgt über URLs. Die App muss ein zuerst URL-Schema registrieren, um darauf zu reagieren

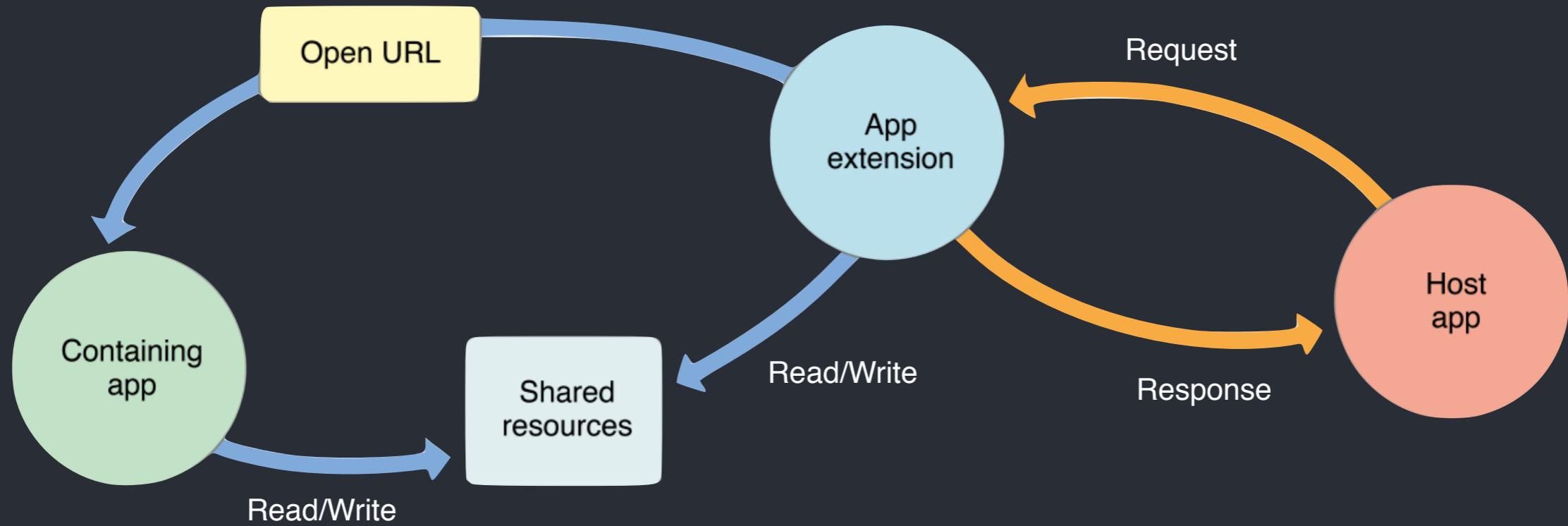
Today-Widgets



Today-Widgets

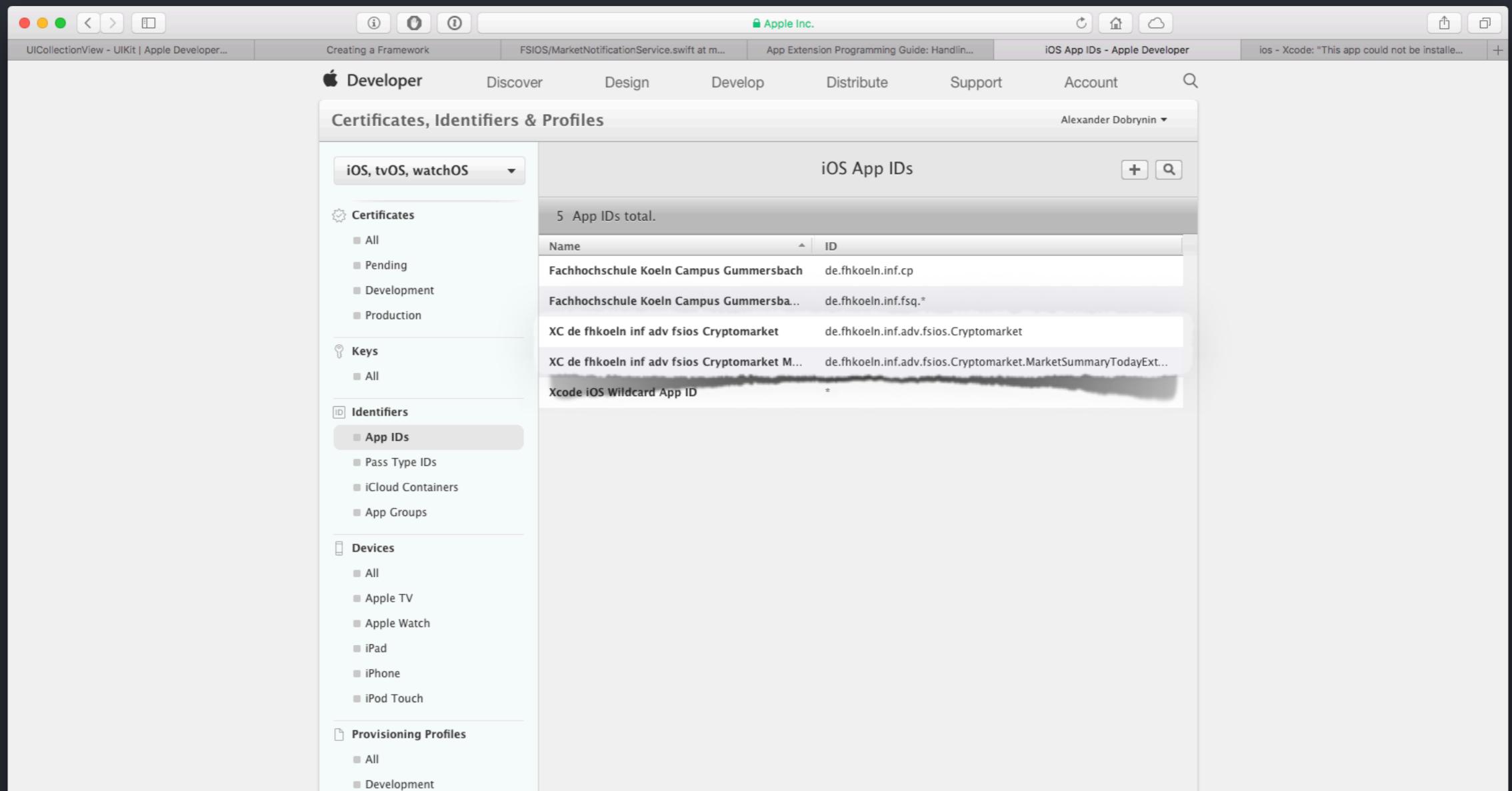
- Today-Widgets zeigen den **neusten/wichtigsten Inhalt** für die aktuelle (heutige) **Zeit** an oder reagieren auf eine **schnelle (typische) Aktion** der App
- Es können mehrere Today-Widgets (oder Extensions im Allgemeinen) mit der App ausgeliefert werden
- Der Benutzer kann diese entweder als Widget hinzufügen (linker Homescreen) oder mit Force-Touch auf dem App-Icon einsehen
- Today-Widgets haben zwei **Display-Modes**
 - Compact (default) zeigt die wichtigsten Informationen in einem Widget mit einer Höhe von **110pt** an
 - Expanded gibt die Möglichkeit auf “Show More”, wodurch das Widget bis zu einer Höhe von **528pt*** erweitert wird
- * Betrifft das iPhone 8. Beim iPhone X sind es 660pt usw.
- Gerade Today-Widgets sind auf die Kommunikation mit der eigentlichen App angewiesen, weil der zu präsentierende Inhalt häufig von einer Benutzereinstellung abhängt
 - Zudem wird der **gemeinsame Code benötigt**, um bspw. eine entsprechende Netzwerkkabfrage zu starten und das Ergebnis in die gemeinsame Datenbasis zu schreiben
 - Des Weiteren kann ein Tap auf das Today-Widget einen **bestimmten ViewController der App öffnen***
- * Das Öffnen erfolgt über URLs. Die App muss ein zuerst URL-Schema registrieren, um darauf zu reagieren

Today-Widgets

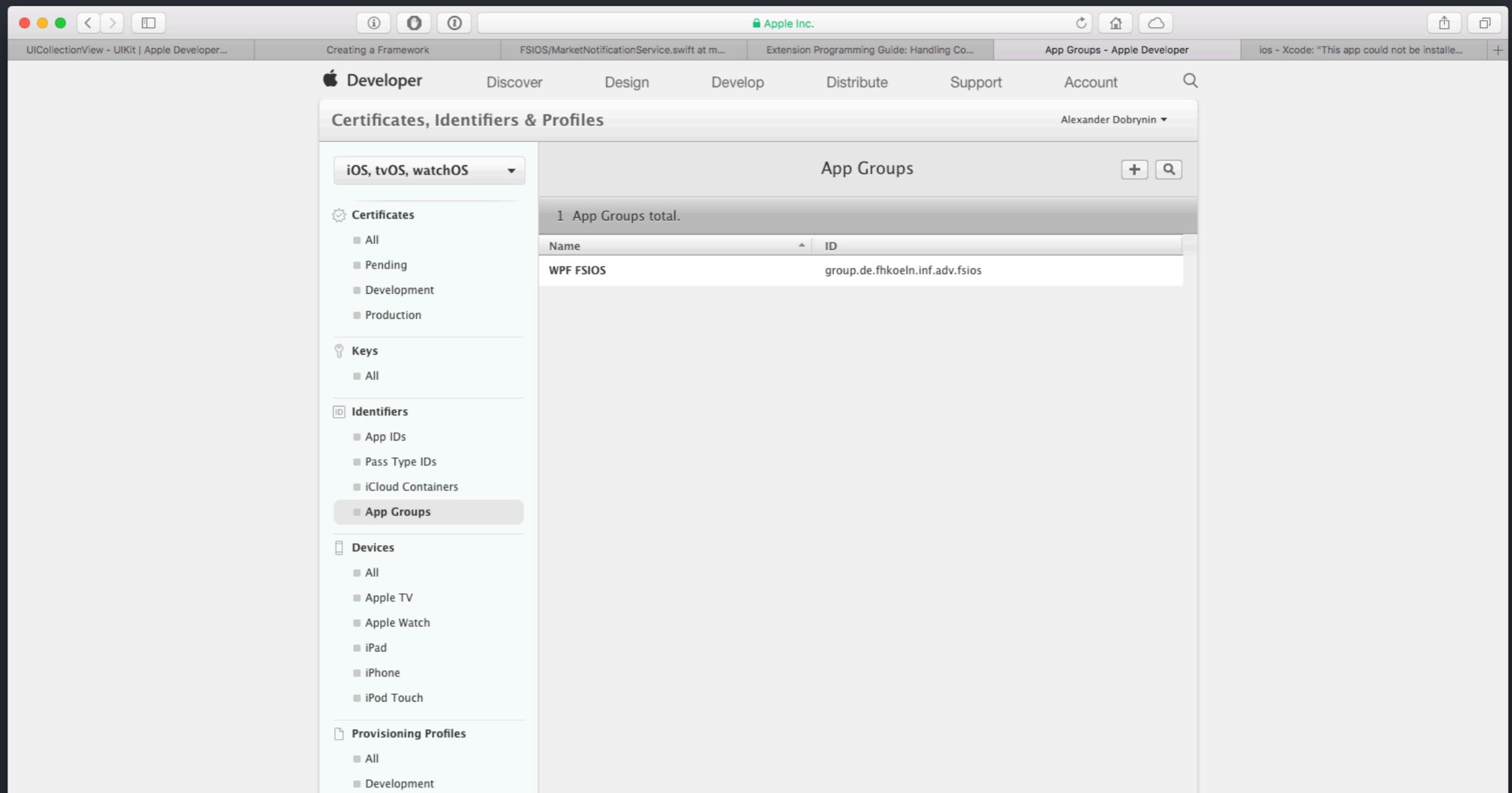


Modifiziert übernommen aus https://developer.apple.com/library/content/documentation/General/Conceptual/ExtensibilityPG/ExtensionOverview.html#/apple_ref/doc/uid/TP40014214-CH2-SW2

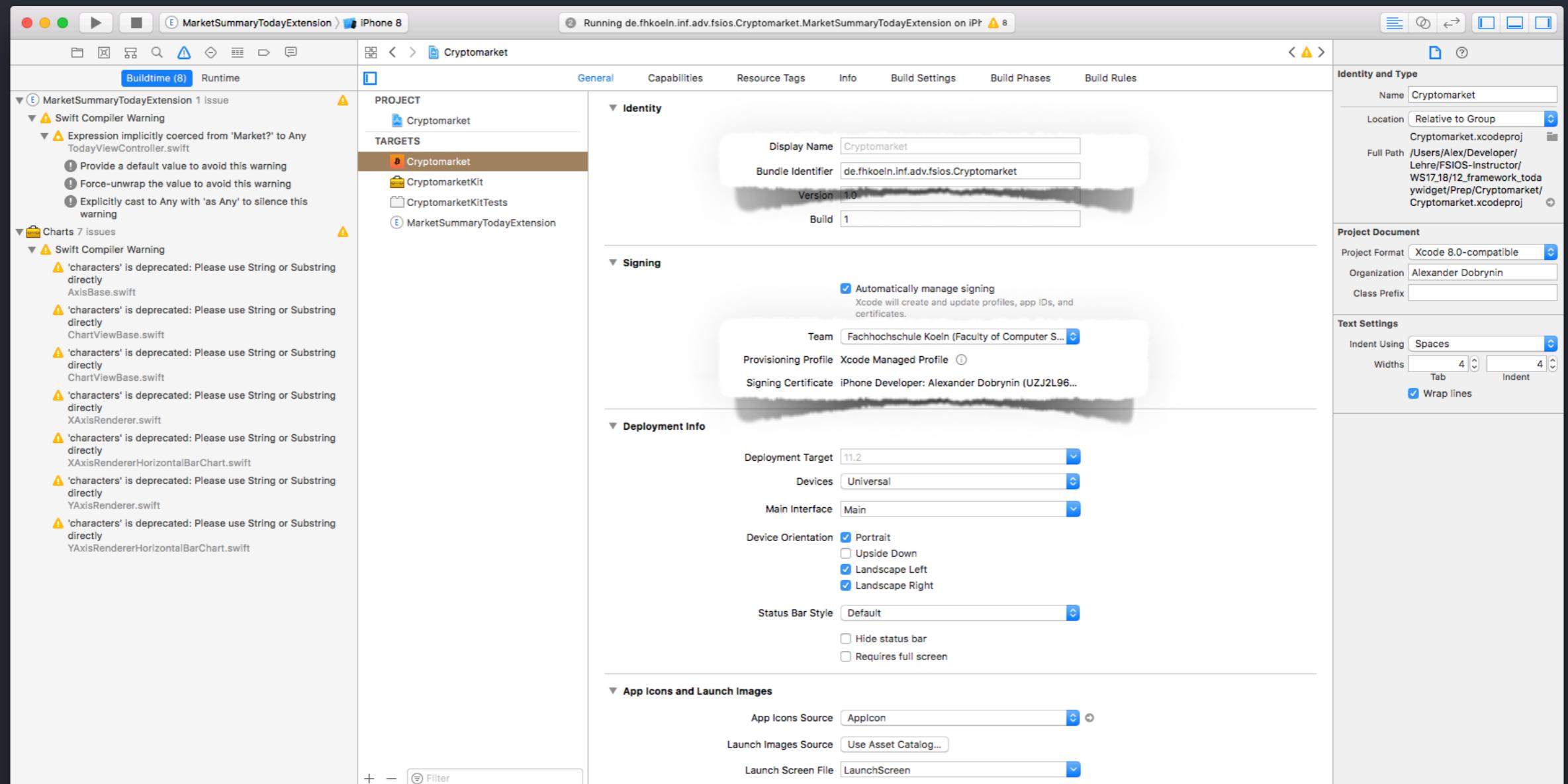
Today-Widgets



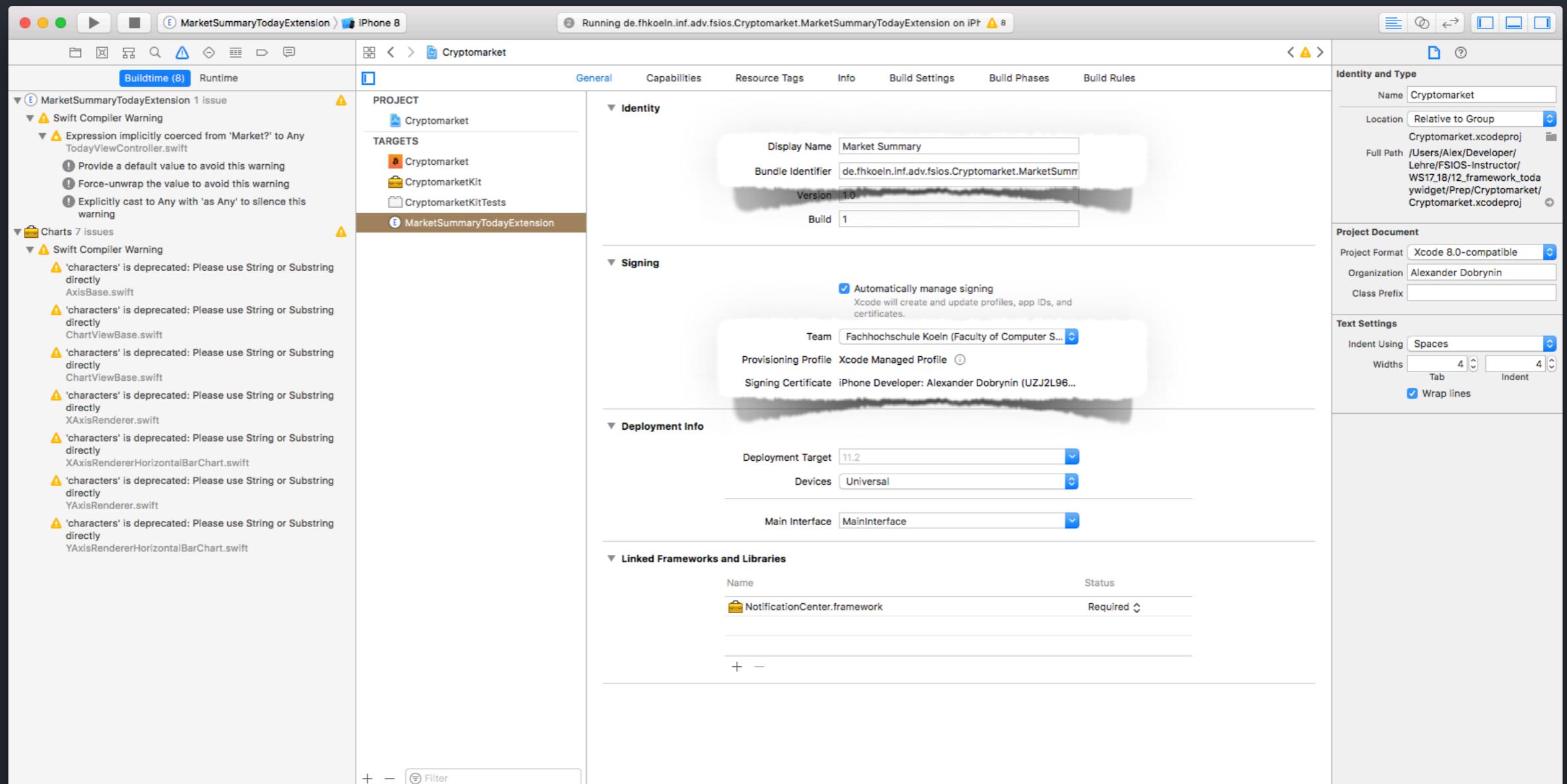
Today-Widgets



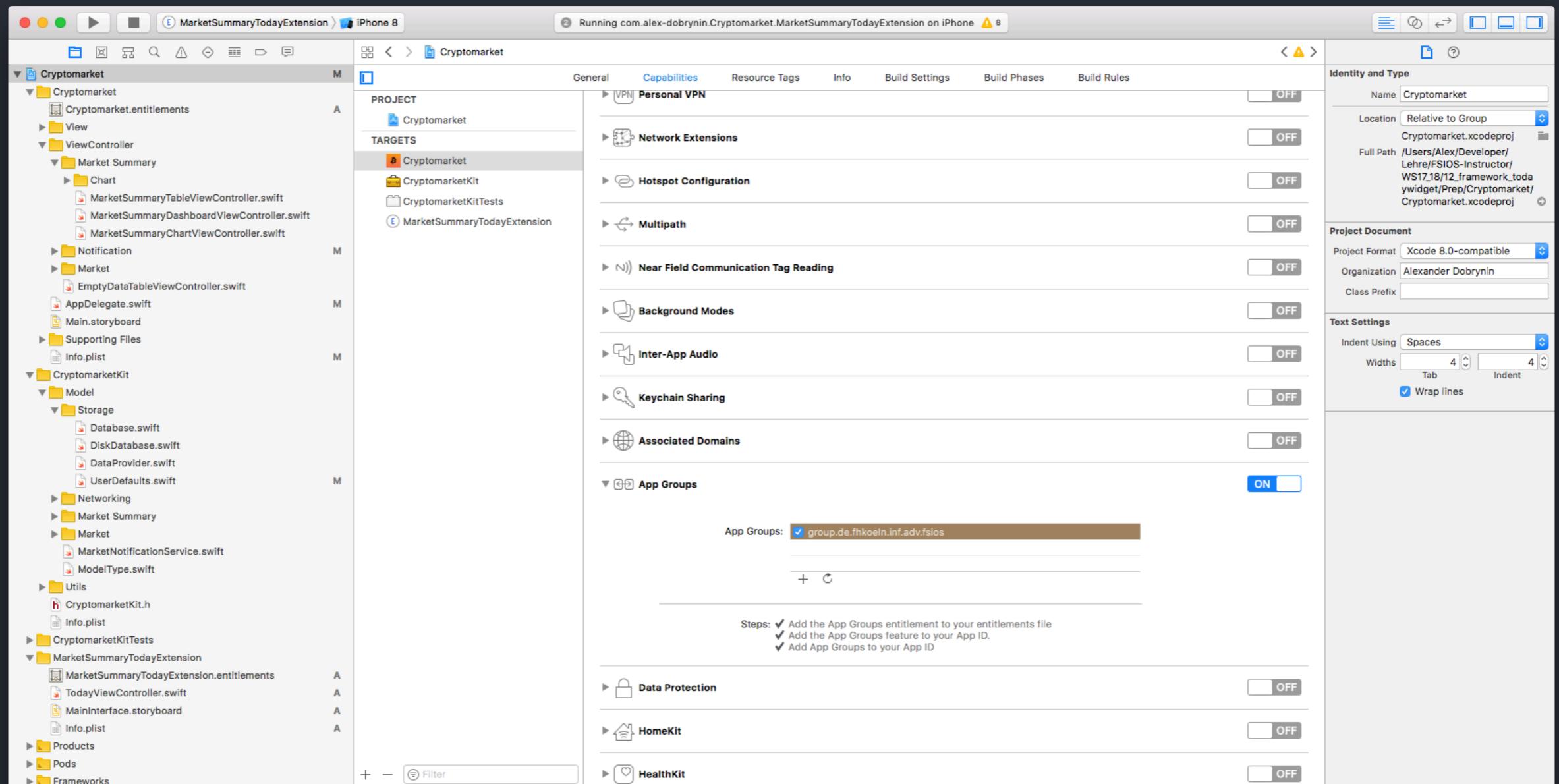
Today-Widgets



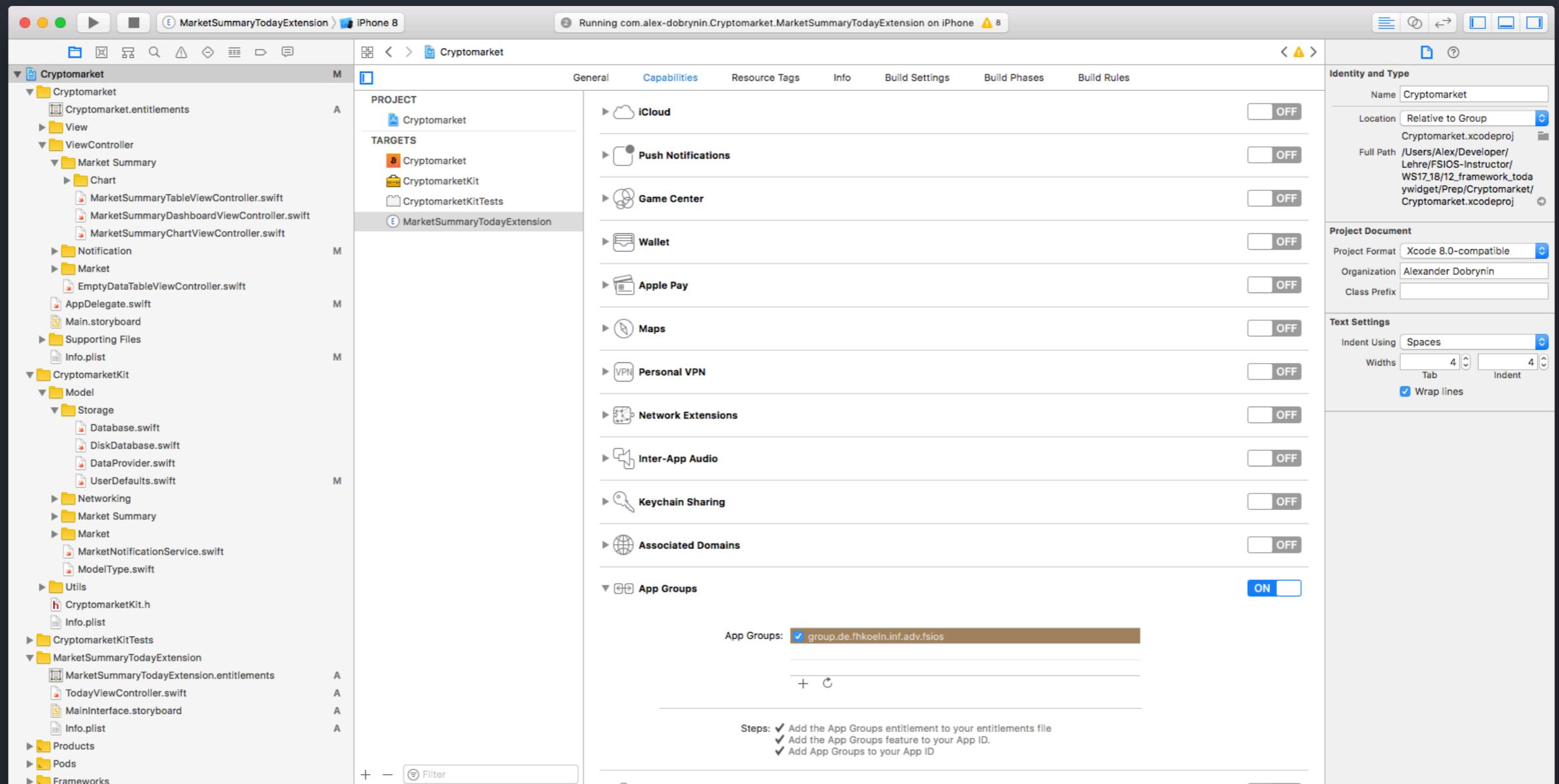
Today-Widgets



Today-Widgets



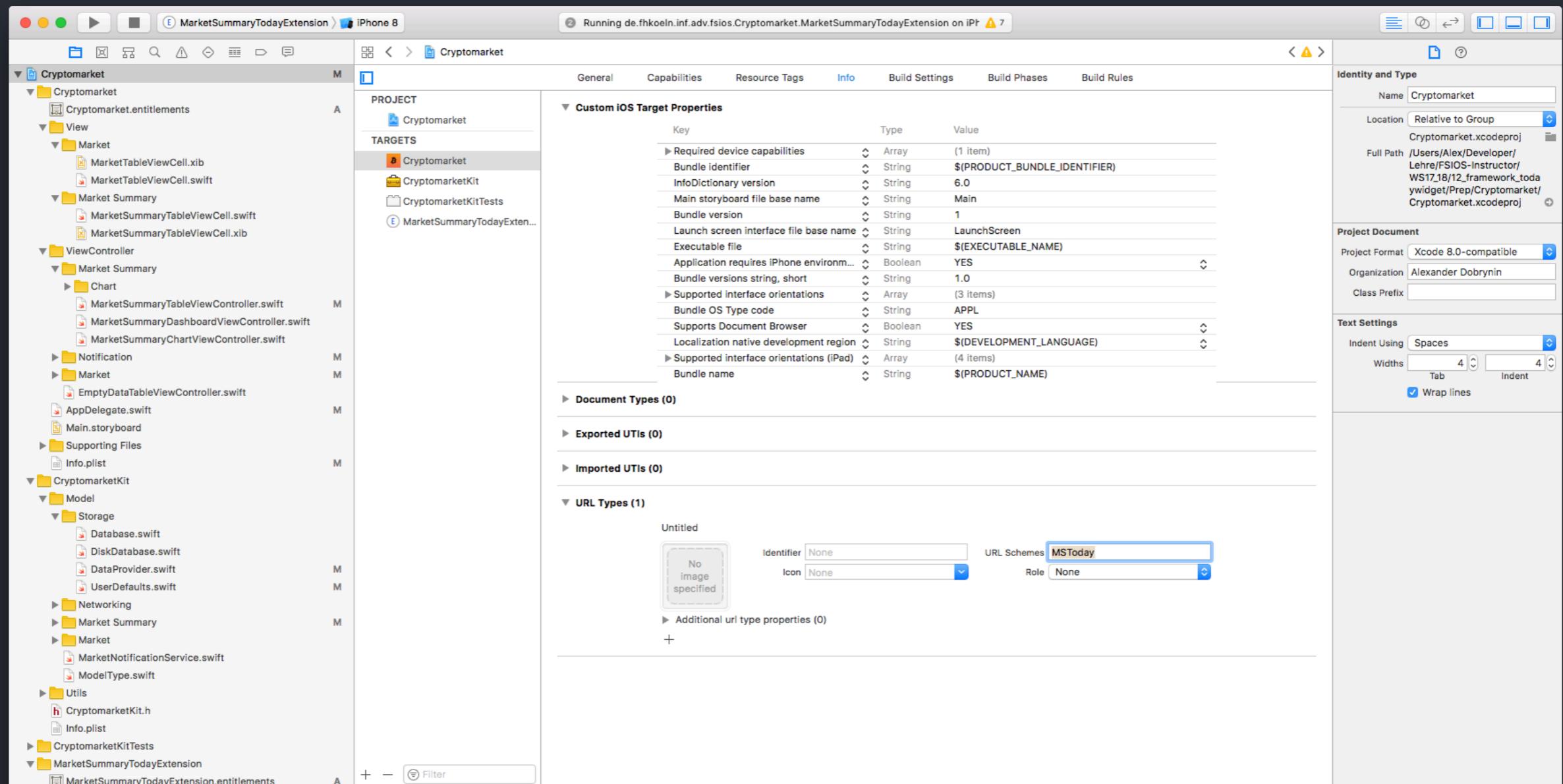
Today-Widgets



Today-Widgets

- Today-Widgets zeigen den **neusten/wichtigsten Inhalt** für die aktuelle (heutige) **Zeit** an oder reagieren auf eine **schnelle (typische) Aktion** der App
 - Es können mehrere Today-Widgets (oder Extensions im Allgemeinen) mit der App ausgeliefert werden
 - Der Benutzer kann diese entweder als Widget hinzufügen (linker Homescreen) oder mit Force-Touch auf dem App-Icon einsehen
 - Today-Widgets haben zwei **Display-Modes**
 - **Compact** (default) zeigt die wichtigsten Informationen in einem Widget mit einer Höhe von **110pt** an
 - **Expanded** gibt die Möglichkeit auf “Show More”, wodurch das Widget bis zu einer Höhe von **528pt*** erweitert wird
 - Gerade Today-Widgets sind auf die Kommunikation mit der eigentlichen App angewiesen, weil der zu präsentierende Inhalt häufig von einer Benutzereinstellung abhängt
 - Zudem wird der **gemeinsame Code benötigt**, um bspw. eine entsprechende Netzwerkkabfrage zu starten und das Ergebnis in die gemeinsame Datenbasis zu schreiben
 - Des Weiteren kann ein Tap auf das Today-Widget einen **bestimmten ViewController der App öffnen***
- * Das Öffnen erfolgt über URLs. Die App muss ein zuerst URL-Schema registrieren, um darauf zu reagieren

Today-Widgets



Today-Widgets

```
class TodayViewController: UIViewController, NCWidgetProviding {

    override func viewDidLoad() {
        super.viewDidLoad()
        view.addGestureRecognizer(UITapGestureRecognizer(target: self, action: #selector(onTap))) // `view` should be clickable
    }

    @objc func onTap() {
        extensionContext?.open(URL(string: "MyAppScheme://viewController")!, completionHandler: nil) // remember to enable `MyAppScheme`
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        extensionContext?.widgetLargestAvailableDisplayMode = .expanded // enable "more" button
    }

    func widgetPerformUpdate(completionHandler: (@escaping (NCUpdateResult) -> Void)) { // update your content here
        Network().go { result in
            if let result = result {
                updateUI(with: result)
                completionHandler(.newData) // remember to call completionHandler with .newData
            } else {
                completionHandler(.noData) // ... or with .noData
            }
        }
    }

    func widgetActiveDisplayModeDidChange(_ activeDisplayMode: NCWidgetDisplayMode, withMaximumSize maxSize: CGSize) {
        switch activeDisplayMode { // which display mode are we currently on?
        case .expanded:
            let heightAfterExpanded = expandUI()
            self.preferredContentSize = CGSize(width: maxSize.width, height: heightAfterExpanded)
        case .compact:
            shrinkUI()
            self.preferredContentSize = maxSize
        }
    }
}
```

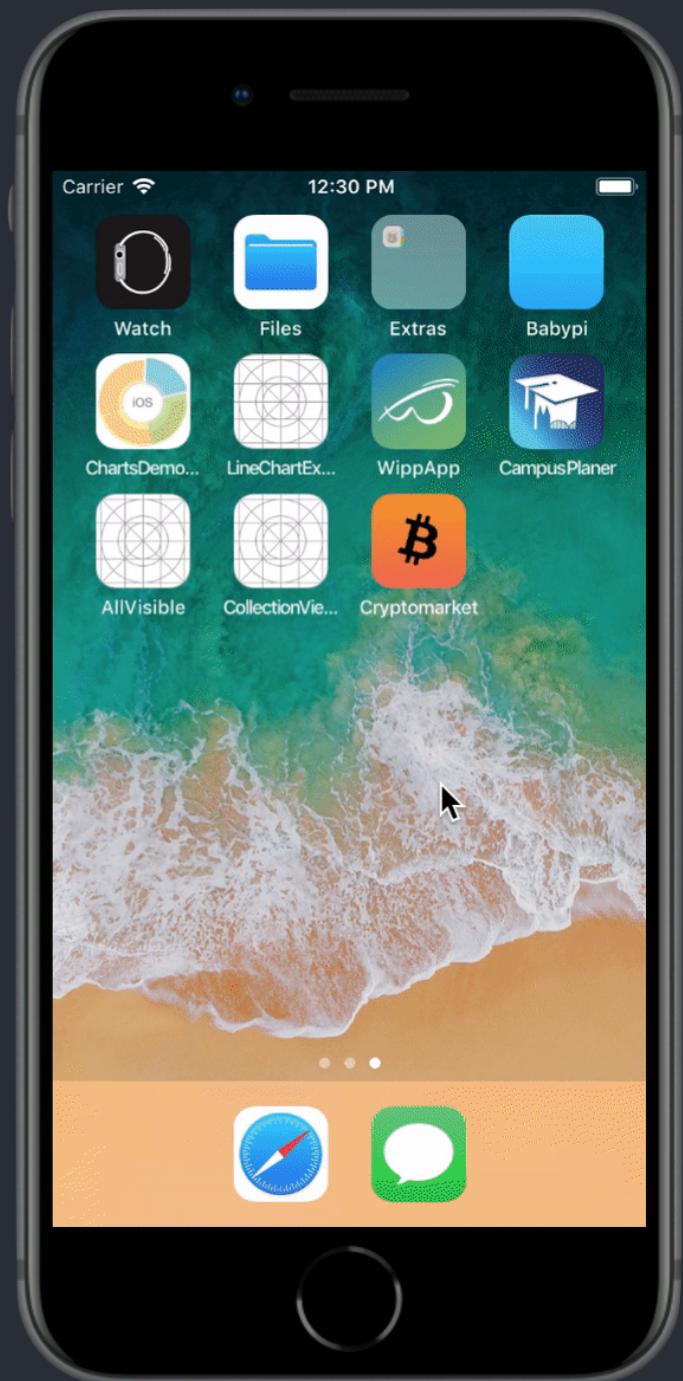
Heute

Frameworks
App-Extensions
Today-Widgets

Demo

Cryptomarket - Demo

- Refactoring von Code
- Trennung von Anwendungslogik und display-spezifischem Code
- Erstellen eines eigenen Cocoa-Touch-Framework
- Einbetten eines Frameworks, Umgang mit unterschiedlichen Targets und Sichtbarkeiten (public)
- App-Groups und geteilte UserDefaults
- Today-Widget (Compact und Expanded)
- URL-Schemes und App-URLs
- Unit Tests



The End



Viel Spaß und Erfolg bei euren Projekten 😊💻