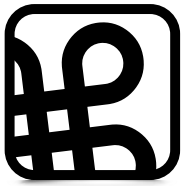


Macoun

Voice Commands mit Siri in iOS 12

Alexander Heinrich

Ablauf

- Möglichkeiten
- Anwendungen
- Hintergrund
- Implementierung
- Tips und Tricks
- Fragen

Möglichkeiten

Was bietet Siri für Entwickler?

- Aktionen (m)einer App im Hintergrund ausführen
- Automatisierungen ausführen
- Abkürzungen (Shortcuts)

Das ist neu!

- Shortcuts direkt in die eigene App
  - “Zeige meinen aktuell Score an”
- Eigene Siri Intents erstellen
- Offen für alle Arten von Apps

Was bietet Siri nicht?

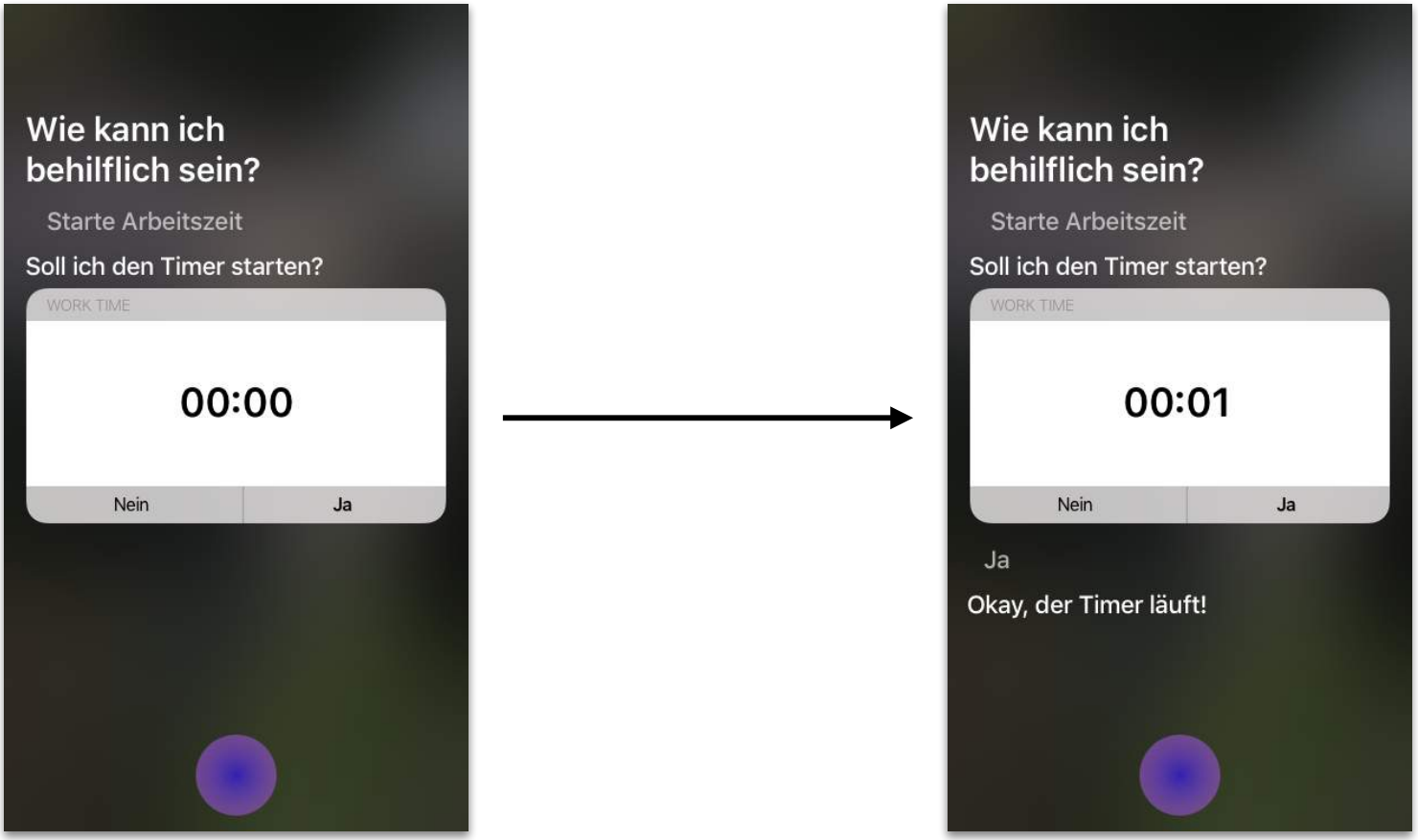
- Variablen in allen Anfragen einbauen
- Intents ohne Nutzerinteraktion hinzufügen
- Rückfragen stellen

Anwendungen

Time Tracking App

- Arbeitszeiten Timer starten
- Timer beenden
- Wochenübersicht anzeigen
- Report erstellen und per E-Mail versenden

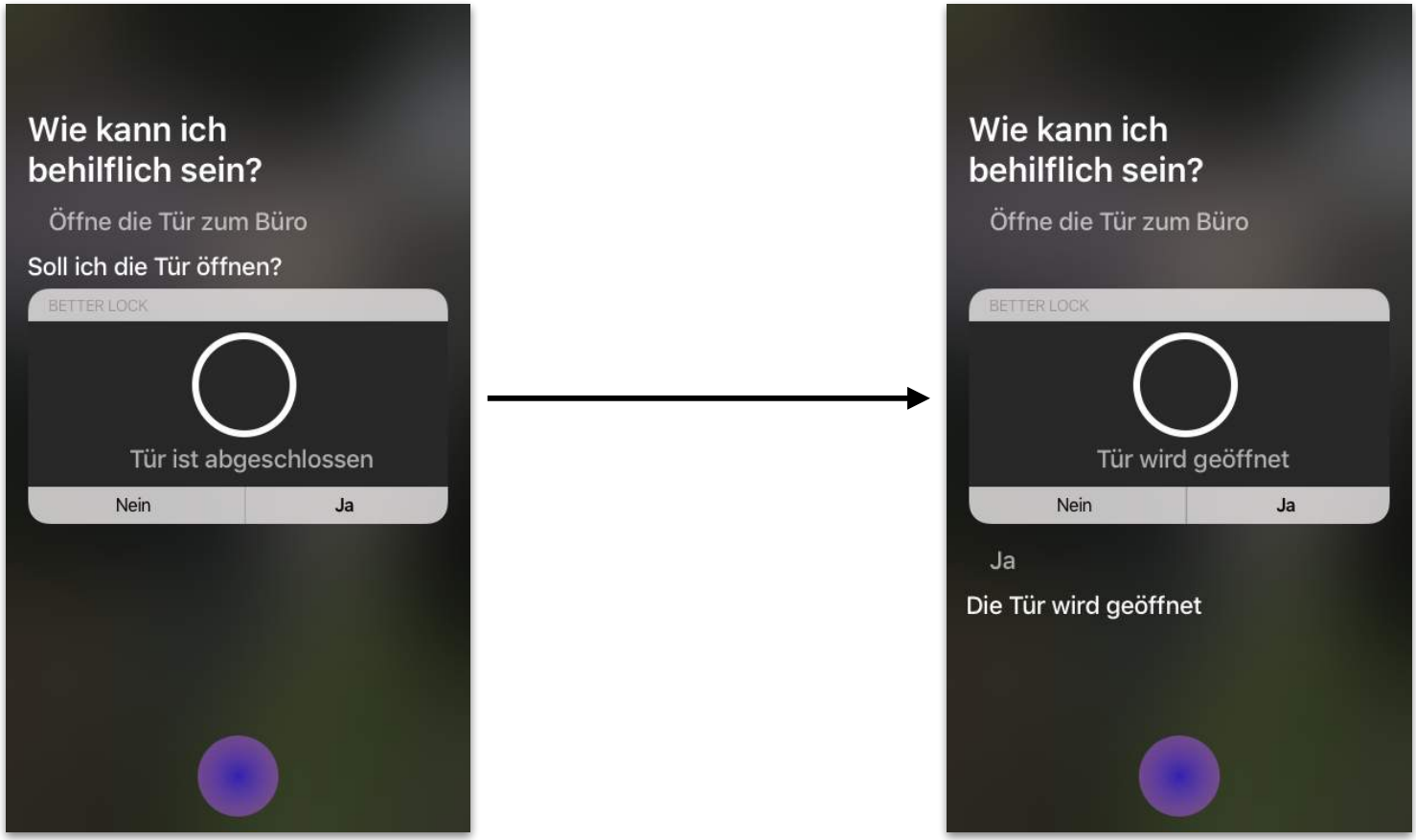
Time Tracking App



Heimautomatisierung

- Ohne HomeKit Lizenz
- Über Shortcuts App mit HomeKit Szenen kombinierbar
- Alle möglichen Aktionen für IoT Geräte möglich

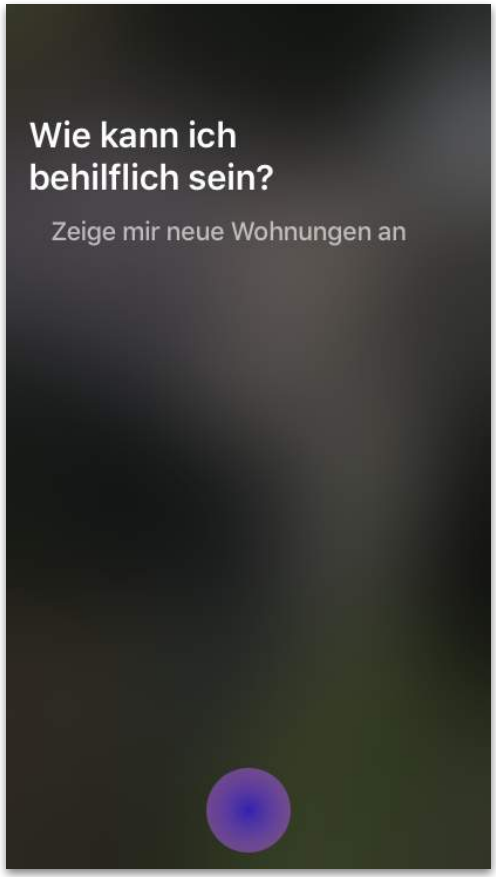
Heimautomatisierung

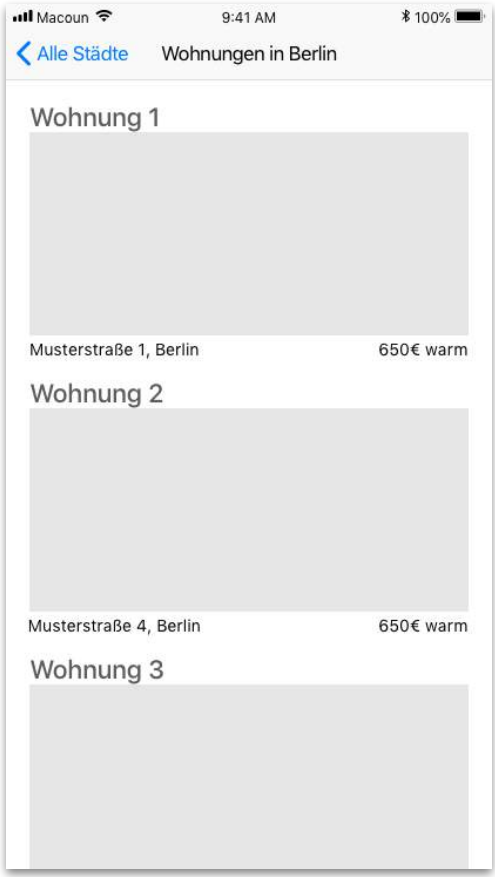


Wohnungssuche

- Häufig wiederholende Aufgabe
- Suche nach einer Wohnung in einer bestimmten Stadt
- Ständig muss nach neuen Angeboten gesucht werden

Wohnungssuche





Viele mehr

- Mitgliedskarten anzeigen
- Konzertkarten / Monatstickets anzeigen
- Soundsysteme steuern
- Wettervorhersage
- …

Hintergrund

NSUserActivity & Siri Intents

- Bereits seit iOS 8
- Hauptsächlich für Spotlight und Siri Vorschläge
- Öffnen die App im Vordergrund

- Früher nur für limitierte Aktionen
- Mittlerweile können eigene Intents definiert werden
- Aktionen sind immer mit Sprachbefehl verbunden
- App läuft im Hintergrund

Welche Methode?

NSUserActivity

- Aktionen, die aufwendig in der App ausgeführt werden
- Schnell und einfach zu implementieren
- Gut für “Shortcuts” in die eigene App
- Anzeigen von etwas in der eigenen App
- Siri Vorhersagen und Spotlight integriert

Welche Methode?
Siri Intents

- Einfache Aktionen, die wiederholt verwendet werden
- App wird nicht gestartet
- Können direkt aus dem Siri Interface verwendet werden
- Kombination mit anderen Intents möglich in der Shortcuts App

Voice Commands

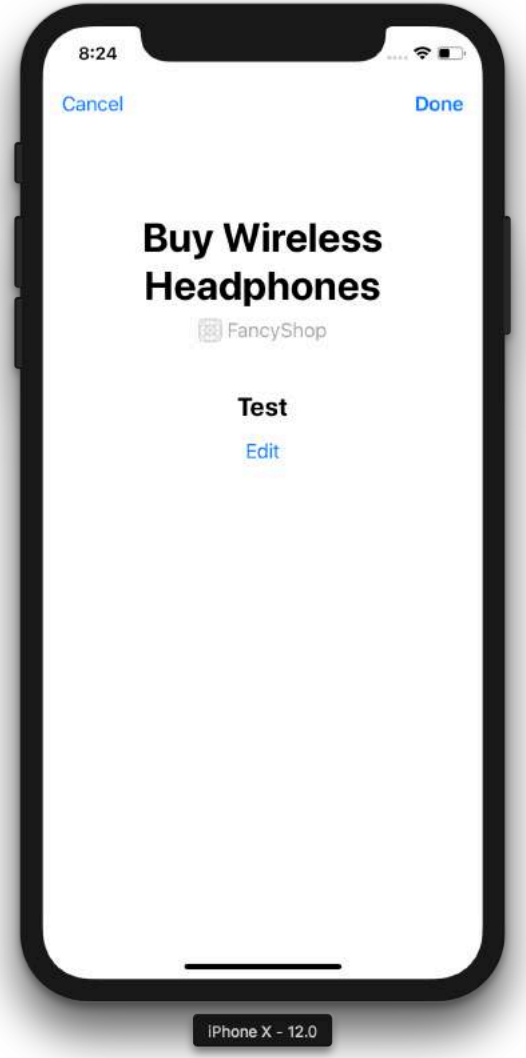
1. Nutzer wählt eine vordefinierte Aktion aus
2. Spricht eigenes Kommando in Siri ein
3. Durch das Kommando wird die Aktion in Siri ausgeführt

Voice Commands - Beispielsätze

- “Zeige meinen Highscore an”
  - Highscore Tabelle in einem Spiel anzeigen
- “Zeig mir den Weg zu meinem Lieblingsrestaurant”
  - Startet Navigation in Navigations App
- “Bestelle Klopapier”
  - Klopapier wird bei einem Lieferdienst bestellt


8:23
Cancel
Add to Siri
Buy Wireless Headphones
FancyShop
After you record your personalized phrase, Siri can use it to tell FancyShop to run this shortcut.
iPhone X - 12.0


8:24
Cancel
iPhone X - 12.0


8:24
Cancel
Done
Buy Wireless Headphones
FancyShop
Test
Edit
iPhone X - 12.0

Implementierung

NSUserActivity

Erstellen einer NSUserAcitvity

```
// UserActivityViewController

 let userActivity = NSUserActivity(activityType: bundleId +
 “.viewItemActivity")

  userActivity.title = "View " + item.name
  userActivity.isEligibleForSearch = true
  userActivity.isEligibleForPrediction = true
  userActivity.userInfo = ["itemId" : item.id]
  self.userActivity = userActivity
```

UserInfo Dictionary hinzufügen

```
// UserActivityViewController

 override func updateUserActivityState(_ activity:
 NSUserActivity) {
   //Update the userActivities userInfo
   activity.userInfo = ["itemId" : item.id]
 }
```

Auf Ausführung reagieren

```
// AppDelegate

 func application(_ application: UIApplication, continue userActivity:
 NSUserActivity, restorationHandler: @escaping ([Any]?) → Void) -> Bool {

  if userActivity.activityType.hasSuffix(".viewItemActivity") {
    //Show the item
    self.showItem(withUserActivity: userActivity)
    return true
   }
   return false
}
```

Shortcut zu Siri hinzufügen

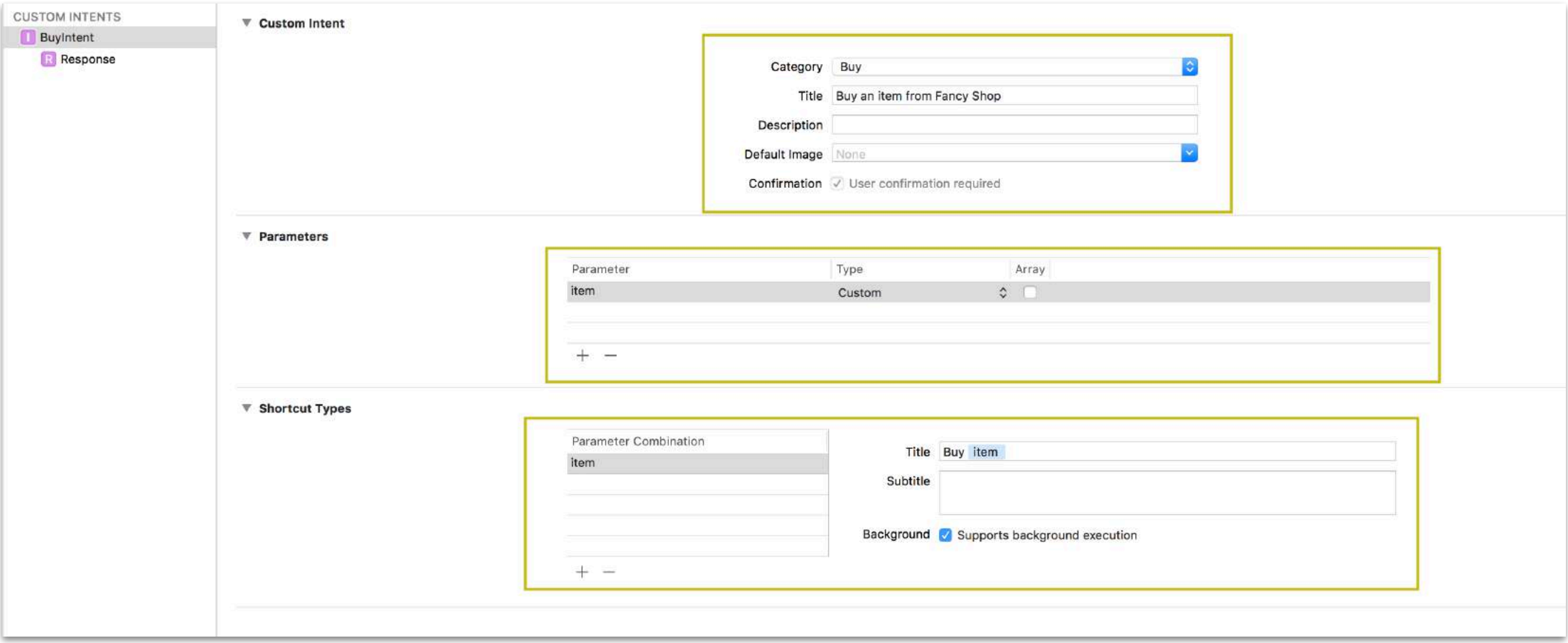
```
// UserActivityViewController
    func addActivityToSiri() {
        let buyActivity = DonatationManager.buyingUserAcitivity(forItem: item)
        let shortcut = INShortcut(userActivity: buyActivity)
        let vc = INUIAddVoiceShortcutViewController(shortcut: shortcut)
        present(vc, animated: true)
    }
```

Live Demo - NSUserActivity

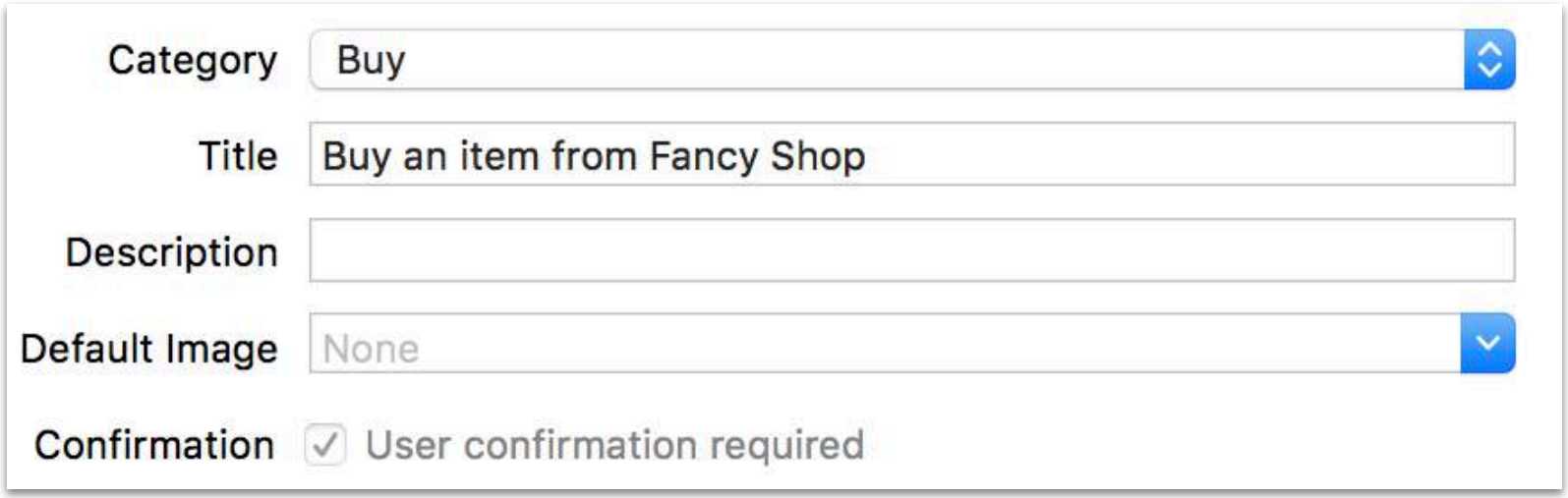
Siri Intents

Intents.intentdefinition

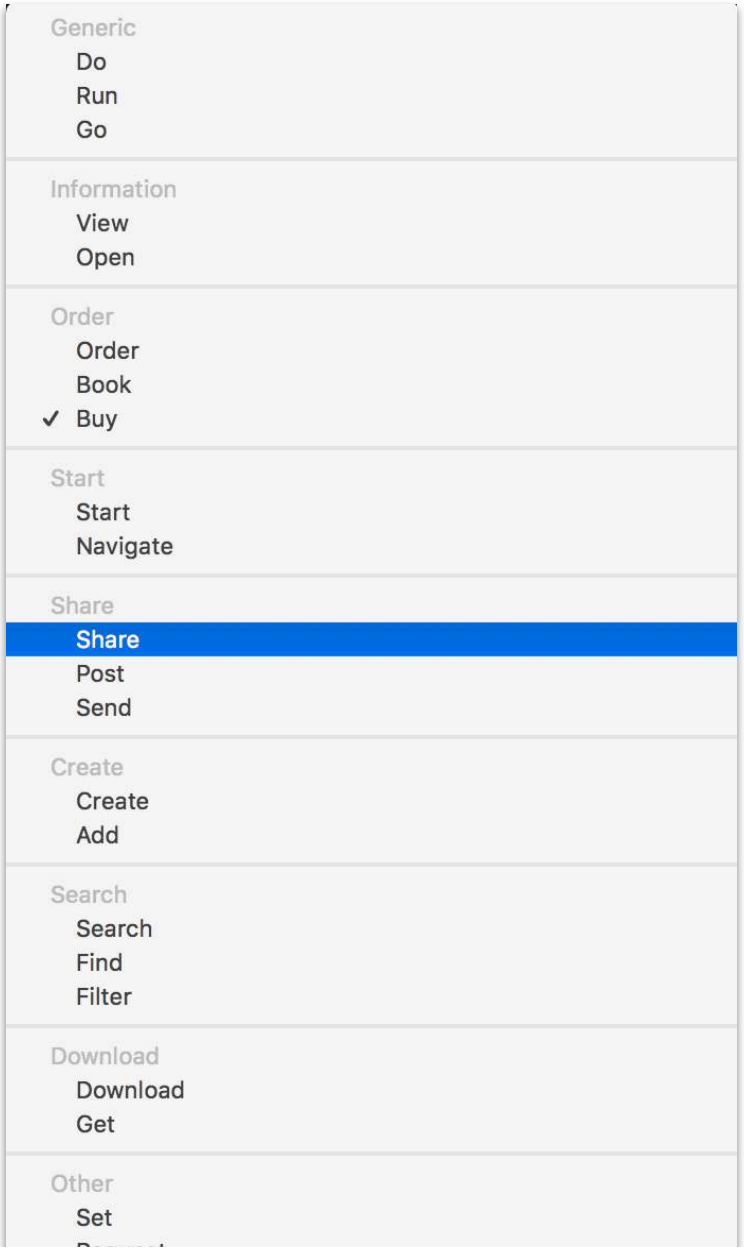
- Intents File erstellen
- Eigene Intents definieren im File
- Xcode generiert automatisch entsprechende Klassen


CUSTOM INTENTS
BuyIntent
Response
Custom Intent
Category
Buy
Title
Buy an item from Fancy Shop
Description
Default Image
None
Confirmation
User confirmation required
Parameters
Parameter
Type
Array
item
Custom
Shortcut Types
Parameter Combination
item
Title
Buy item
Subtitle
Background
Supports background execution

Intent definieren

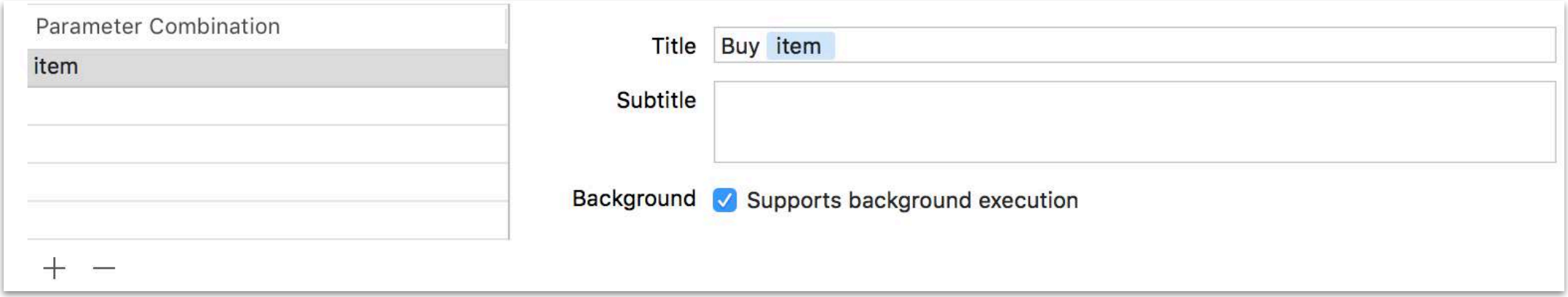


Intent Kategorien

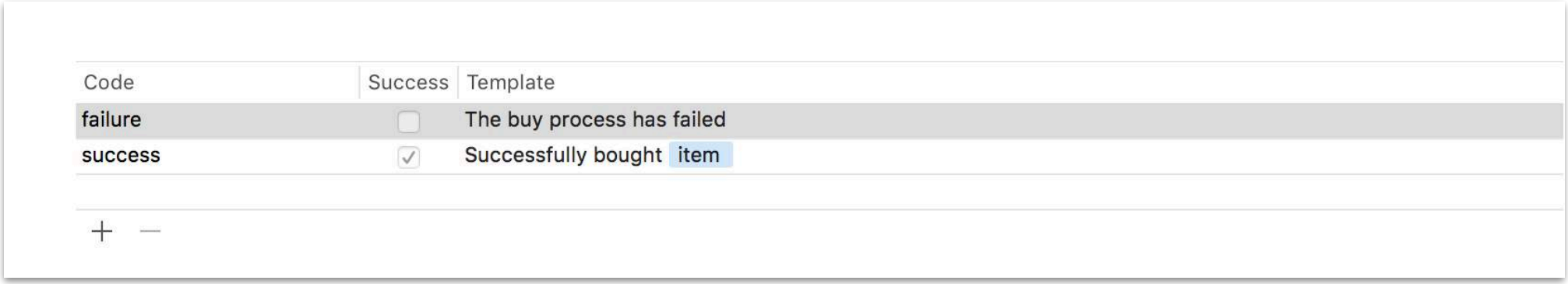


Parameter definieren

Shortcut types definieren



Responses definieren



Code	Success	Template
failure	☐	The buy process has failed
success	☑	Successfully bought item

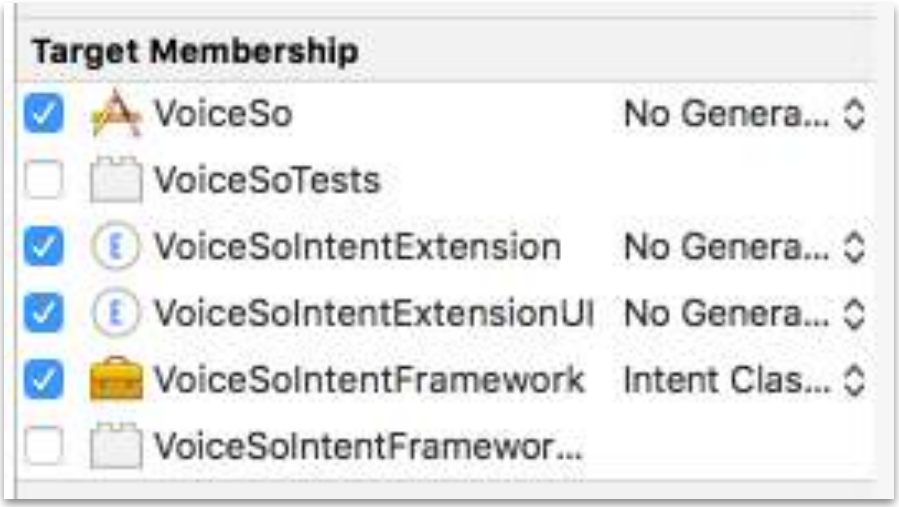
+ −

Benötigte Targets

- App Framework
- Intents Extension
- Intents Extension UI
- App

Target Membership konfigurieren

Intents.intentdefinition



- IntentFramework auf "Intent Classes" stellen
- Alle anderen auf "No generated classes"
- Andere Targets importieren das IntentFramework

Intent Framework

- Eigenes Framework in der App
- Hilft dabei die Intents zu bearbeiten
- Kann zwischen Extensions geteilt werden

Eigene IntentHandler

- Für jeden Intent eine eigenen IntentHandler
- Implementiert das entsprechende IntentHandling Protokoll
- Im IntentFramework enthalten

IntentHandling Protokoll

```
// BuyIntentHandler

 func confirm(intent: BuyIntentIntent, completion: @escaping
 (BuyIntentIntentResponse) -> Void) {
   guard let iItem = intent.item,
       let item = ShopItemManager().getItem(withId: item.identifier) else {
           //Respond with error if item is not available
           completion(BuyIntentResponse(code: .failure, userActivity: nil))
           return
   }
       // Bestätigen, dass die Aktion ausgeführt werden kann
       // Prüfen, ob das Item noch auf Lager ist
  completion(BuyIntentIntentResponse(code: .ready, userActivity: nil))
  }
```

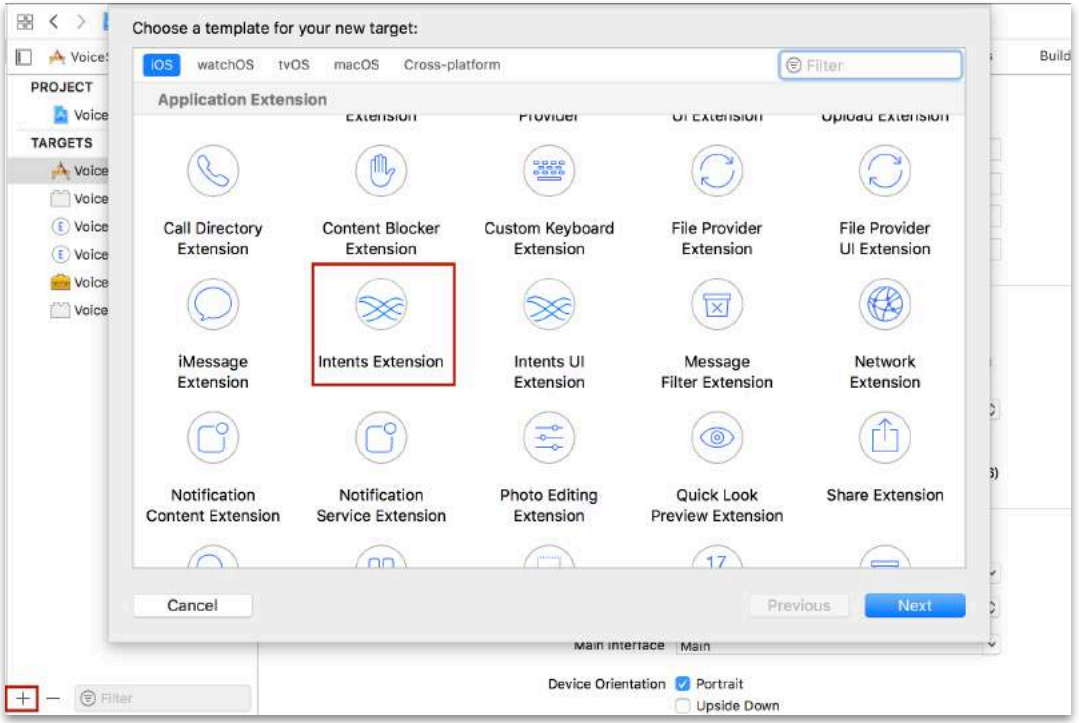
IntentHandling Protokoll

```
// BuyIntentHandler

public func handle(intent: BuyIntent, completion: @escaping
(BuyIntentResponse) -> Void) {
    guard let iItem = intent.item,
        let item = ShopItemManager().getItem(withId: item.identifier) else {
            //…
    }

    let response = BuyIntentResponse(code: .success, userActivity: nil)
    response.item = INObject(identifier: String(item.id), display: item.name)
    response.price = item.priceString
    completion(response)
}
```

IntentExtension anlegen



Intent Extension - IntentHandler

```
// IntentHandler

class IntentHandler: INExtension {
    override func handler(for intent: INIntent) -> Any? {
        if intent is BuyIntent {
            return BuyItemIntentHandler()
        }
        else {
            fatalError("Unhandled intent type: \(intent)")
        }
    }
}
```

Intents in Info.plist aufzählen

```
<key>NSExtension</key>
<dict>
    <key>NSExtensionAttributes</key>
    <dict>
        <key>IntentsSupported</key>
        <array>
            <string>BuyIntent</string>
        </array>
    </dict>
 </dict>
```

Intents Extension UI

- Wird mit der Intent Extension erzeugt
- Normaler ViewController
- Kann in Siri angezeigt werden und mit beliebigen Inhalt gefüllt
- Keine Touch Interaktionen

Intents Extension UI

```
 // IntentViewController
func configureView(for parameters: Set<INParameter>, of interaction:
INInteraction, …, completion: @escaping (Bool, Set<INParameter>, CGSize) ->
Void) {
    switch (interaction.intent) {
    case is BuyIntent:
        self.configureView(withBuyIntent: interaction)
    default:
        break
    }

    completion(true, parameters, self.desiredSize)
}
```

Intents Extension UI

```
func configureView(withBuyIntent interaction: INInteraction) {
    guard let buyIntent = interaction.intent as? BuyIntent,
    //…
    let item = ShopItemManager().getItem(withId: id) else {return}
    //…
    if interaction.intentHandlingStatus == .ready {
      messageLabel.text =
       String(format: "Möchten Sie %@ für %@ kaufen", item.name, priceText)
    }else if interaction.intentHandlingStatus == .success {
      messageLabel.text =
      String(format: "Danke für den Kauf von %@ für %@", item.name, priceText)
    }}
```

Live Demo - Siri Intents

Tips und Tricks

Debugging

- Intent Extension Scheme in Xcode auswählen
- Siri starten
- Breakpoints setzen

App Framework

- Möglichst früh einführen
- Abstraktion vom User-Interaktivem Code

UpdateUserActivityState

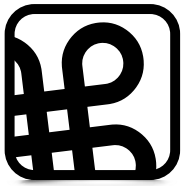
- Unbedingt im ViewController überschreiben
- In dieser Methode die userInfo für die NSUserActivity angeben
- Sonst nichts gespeichert im System

Fazit

- NSUserActivity ist einfach zu verwenden und einzubauen
- Intents sind komplexer und können auch komplexe Abläufe verarbeiten
- Bei Intents müssen alle Schritte beachtet werden
- Keine direkten Nutzereingaben möglich bei Custom Intents

Fragen?

GitHub Repository



Macoun

Vielen Dank