

Programmierung in iOS mit Swift Studiengang MI

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- <http://mwilhelm.hs-harz.de>
- Raum 2.202
- Tel. 03943 / 659 338

Gliederung

Überblick:

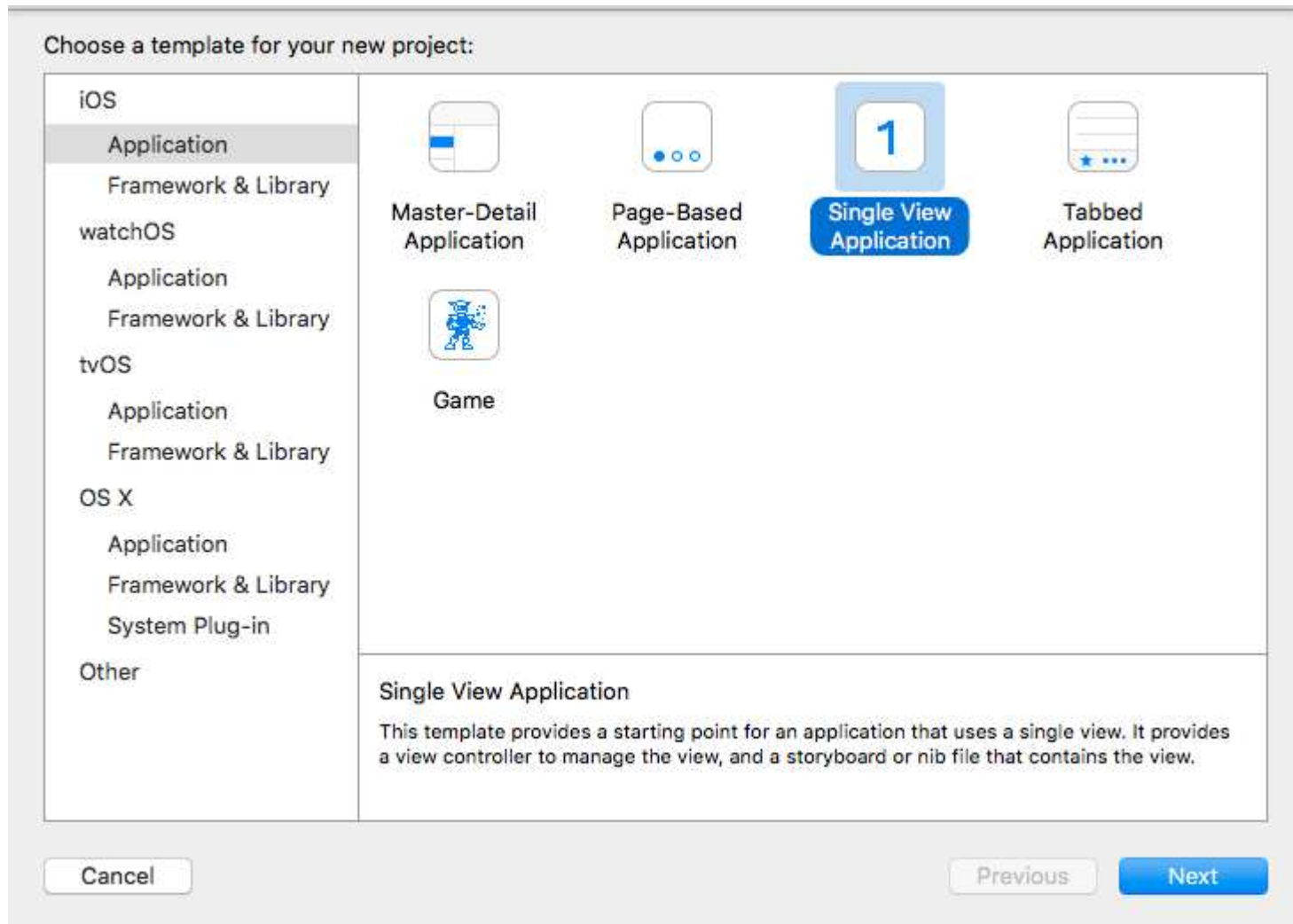
- Einleitung, Geschichte, xcode
- **Sprache**
- **Playground**
- **Grafische Oberfläche**
 - Projekt erstellen
 - UI erstellen
 - UI-Elemente
 - UI-Variablen
 - Action-Methoden (onClick-Events)
 - Layout-Typen
 - Core-Daten
 - Sensoren

Projekt erstellen

- Starten von xcode
- Neues Projekt
 - Master-Detail Application (MVC)
 - Paged Bases Application: Editor
 - **Single View Application**
 - Tabbed Application (Register)
 - Game



Projekt erstellen: Single View Application



Projekt erstellen: Projektdaten

Choose options for your new project:

Product Name:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

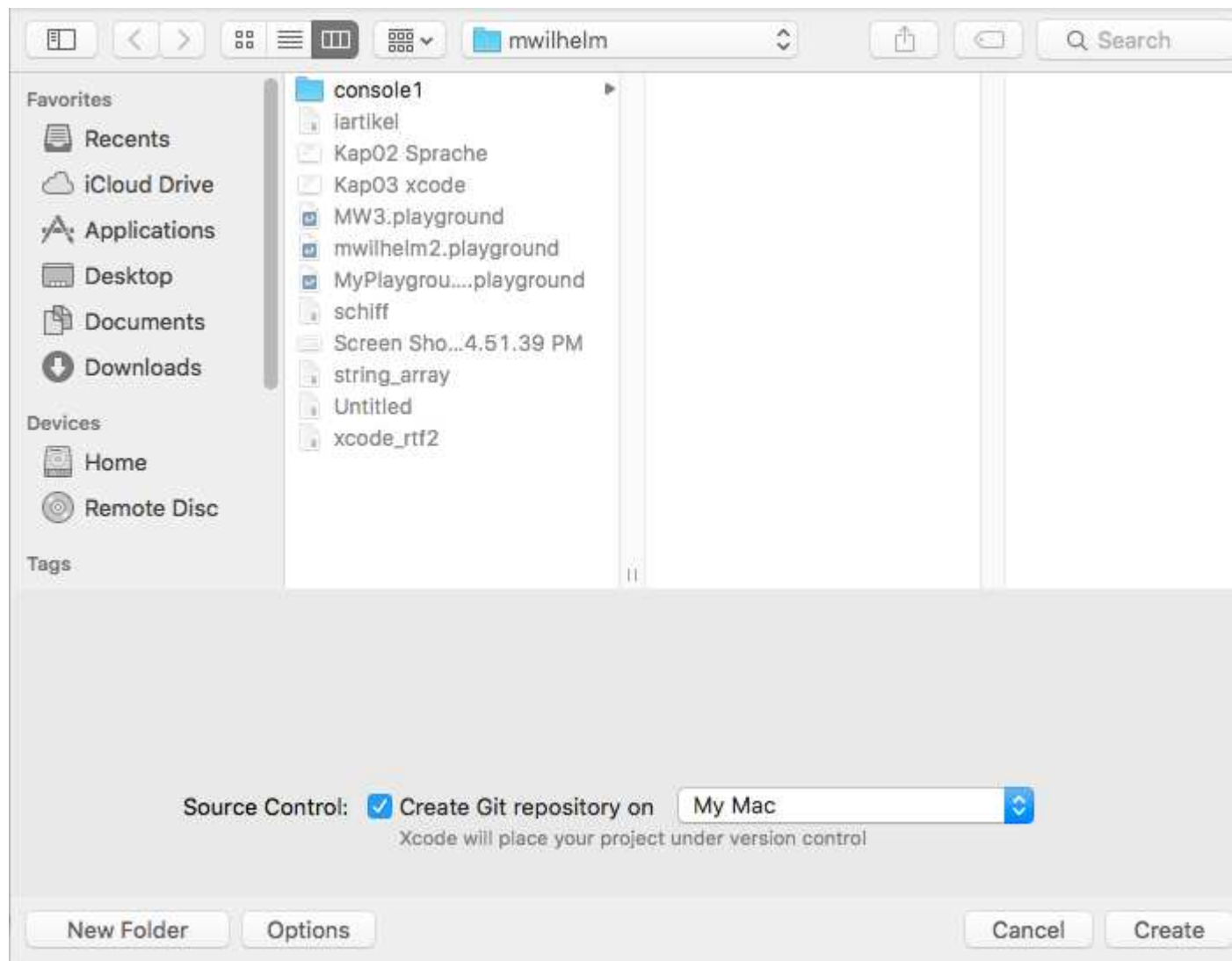
Devices:

☐ Use Core Data

☐ Include Unit Tests

☐ Include UI Tests

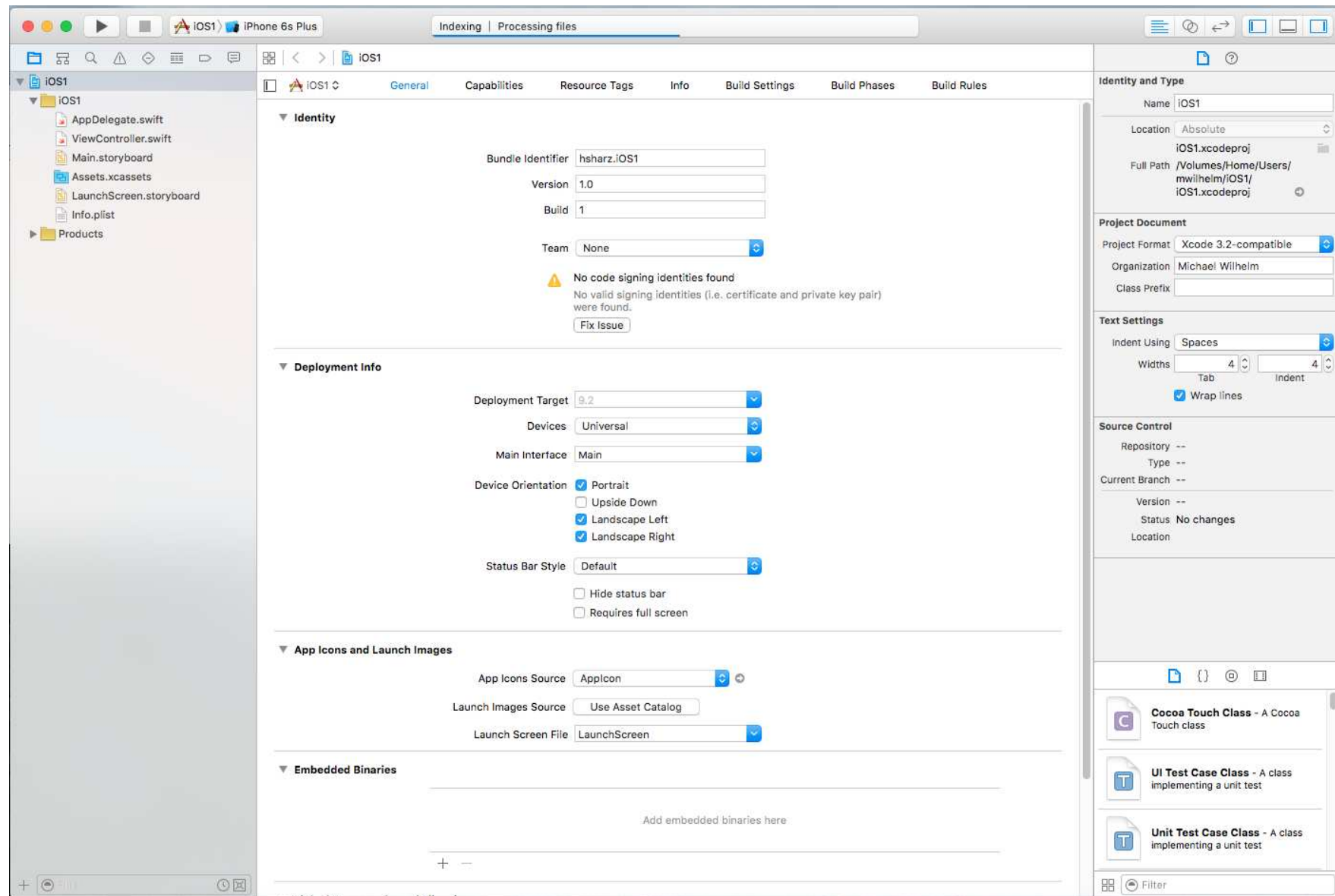
Projekt erstellen: Speicherort



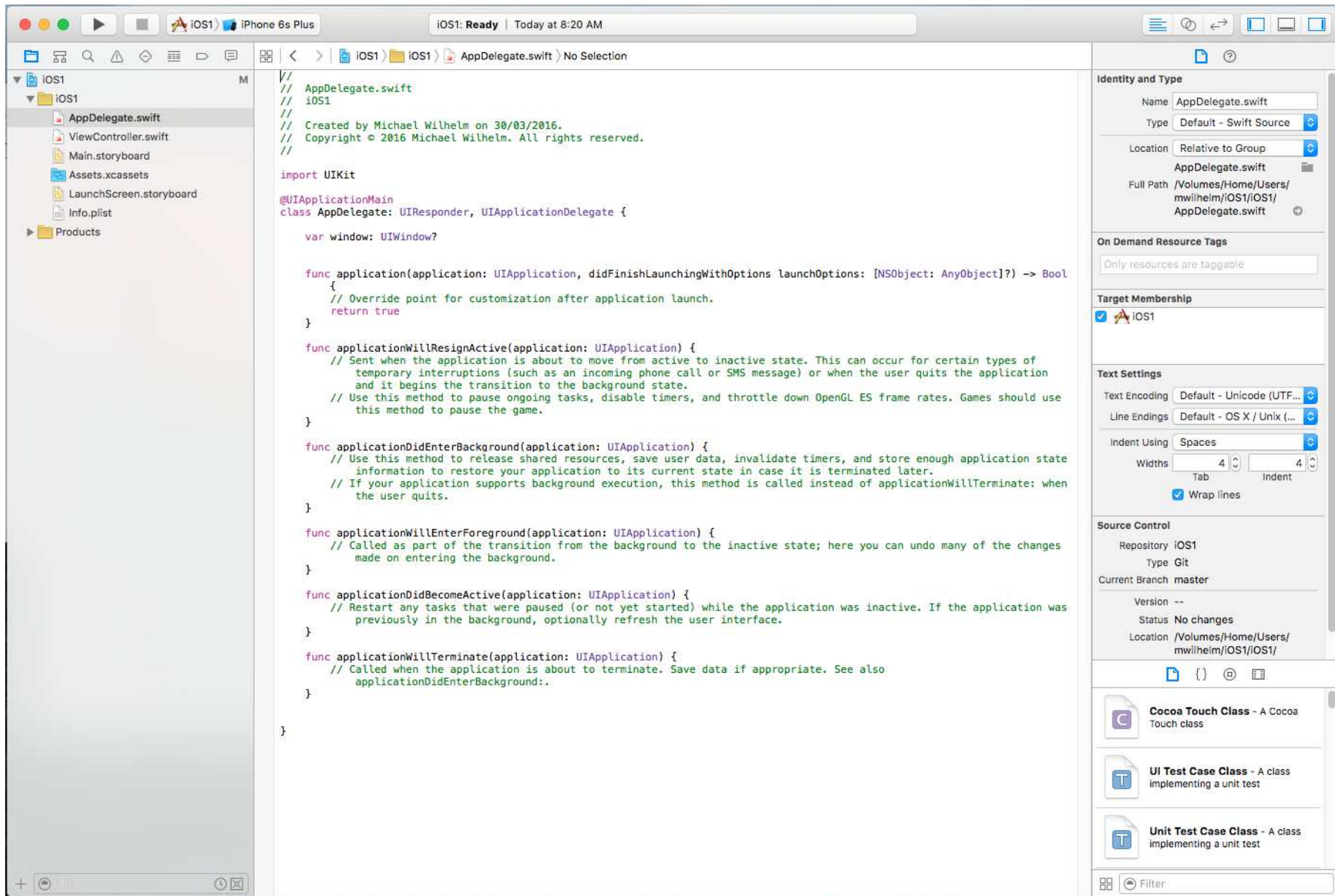
Projektdateien

- AppDelegate.swift
 - Startpunkt der App
- ViewController.swift
 - Beinhaltet den Swift-Code eines UI-Views
- Main.storyboard
 - Verwaltet den View oder die Views, „Controller“
 - Hier kann man die UI-Elemente Einfügen und Bearbeiten
- Assets.xcassets
 - Verwaltet die Logos, Symbole
- LaunchScreen.storyboard
 - Startfenster, Startlogos
 - Localisation, mehrere Sprachen
- Info.plist
 - Property List Files, auch als Speicherung eigener Daten

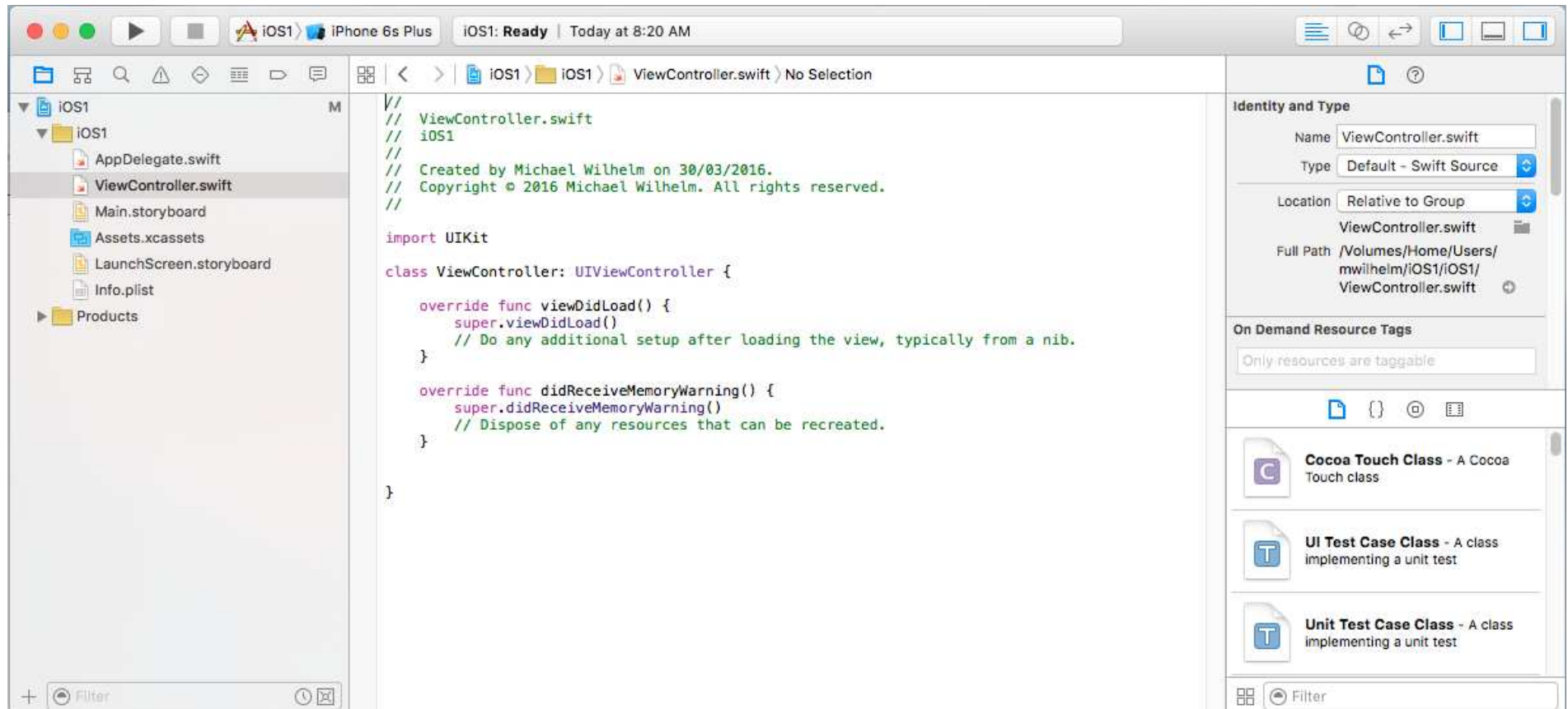
Projekt in xcode: IDE



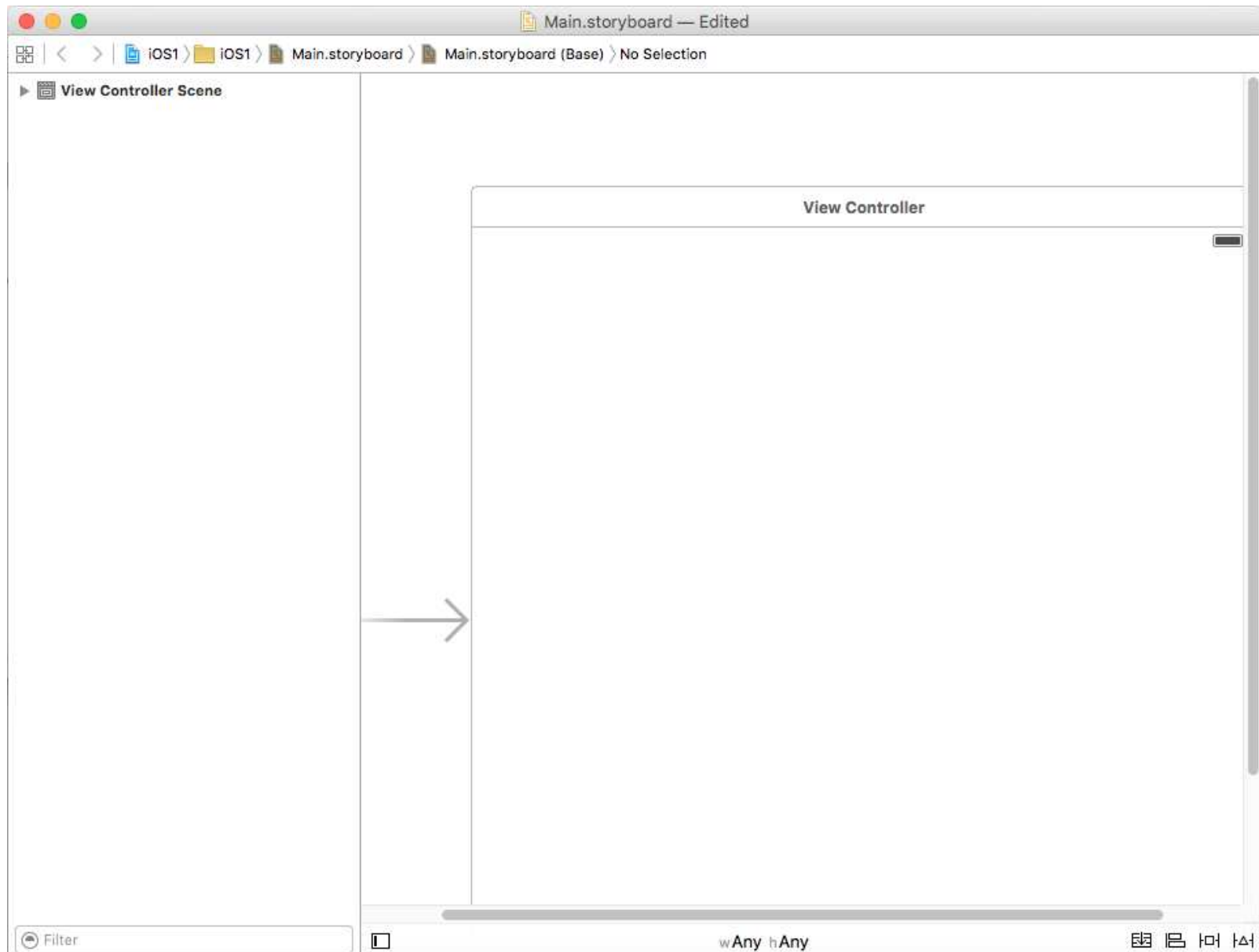
Projekt in xcode: IDE



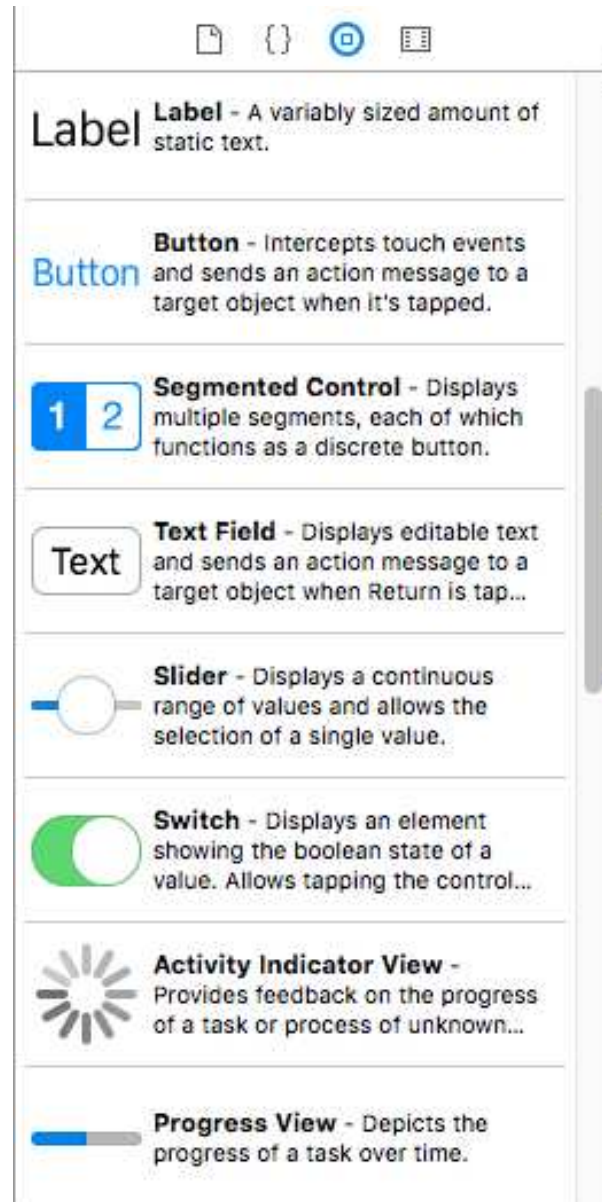
Projekt in xcode: IDE



UI-Oberfläche in xcode: UI-View (noch leer)

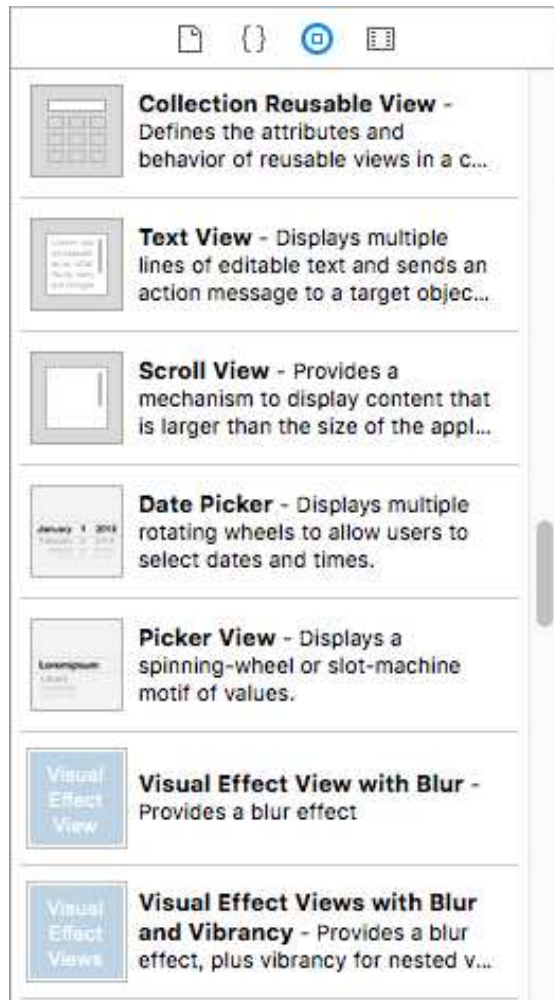


UI-Elemente in xcode



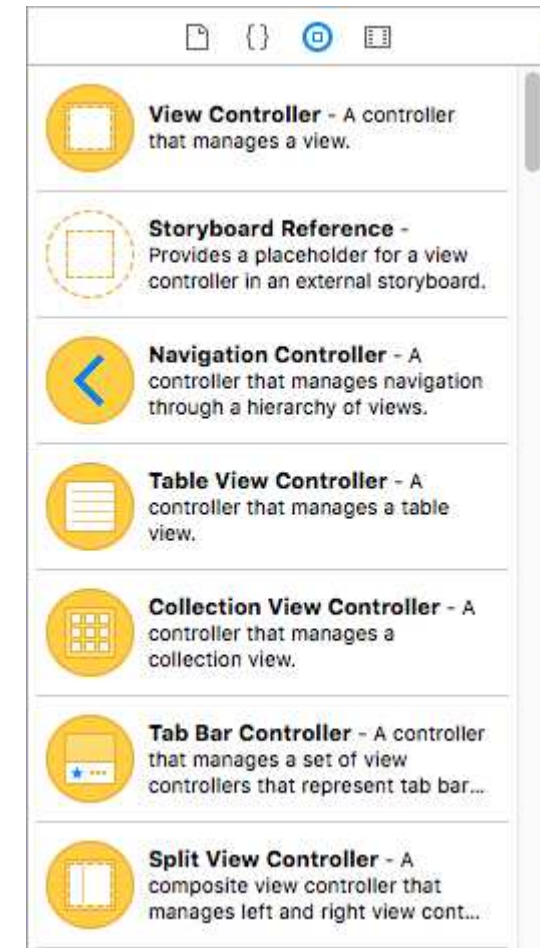
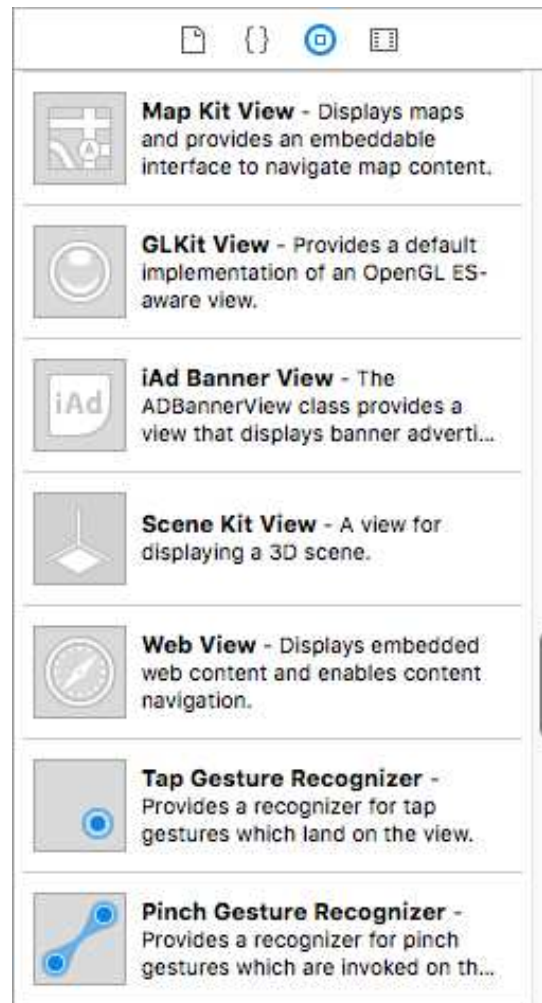
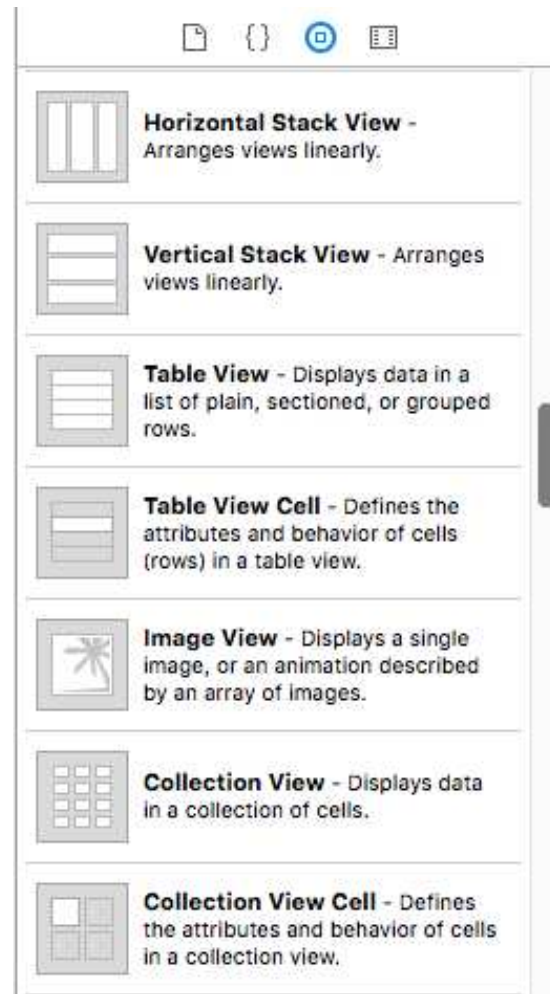
- Label
 - Anzeige von Texten, readonly
- Button
 - onClick-Event
- Segemented Control
 - Mehrere Schalter in einem „JPanel“ (RadioButton)
- Text
 - Texteditor, zeilenweise
- Slider
 - Auswahl aus einem Bereich
- Switch
 - Schalter, Anzeige eines Zustandes (An,Aus)
- Activity Indicator
 - „Ich habe zu tun“
- Progress View
 - Progressbar, Fortschrittsbalken, 0,0 bis 1,0

UI-Elemente in xcode

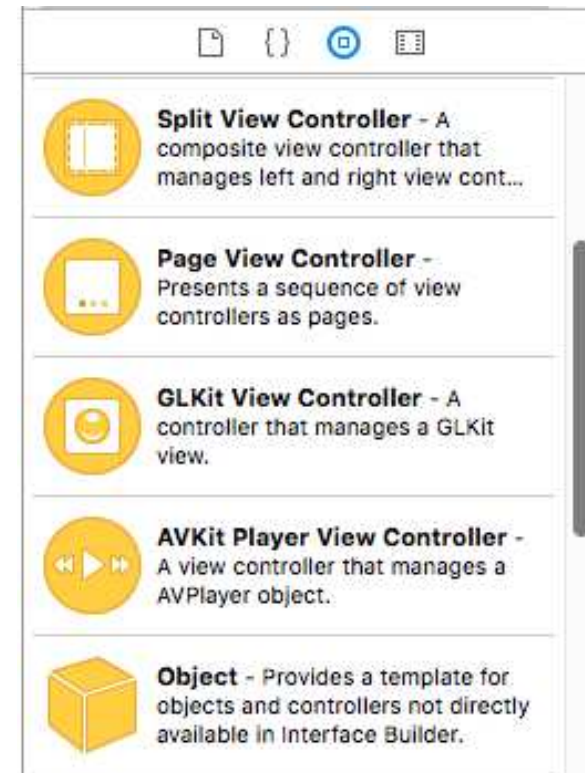
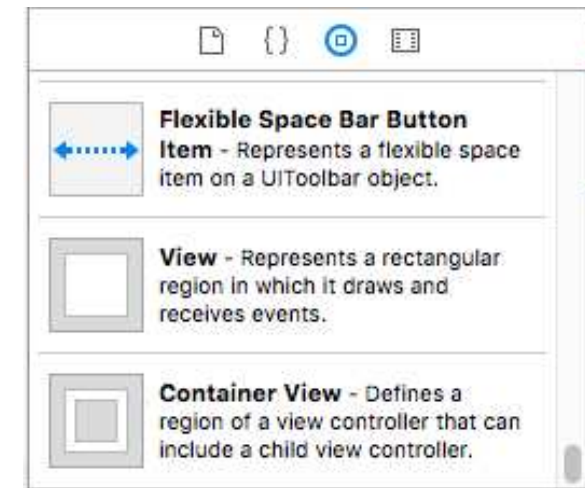
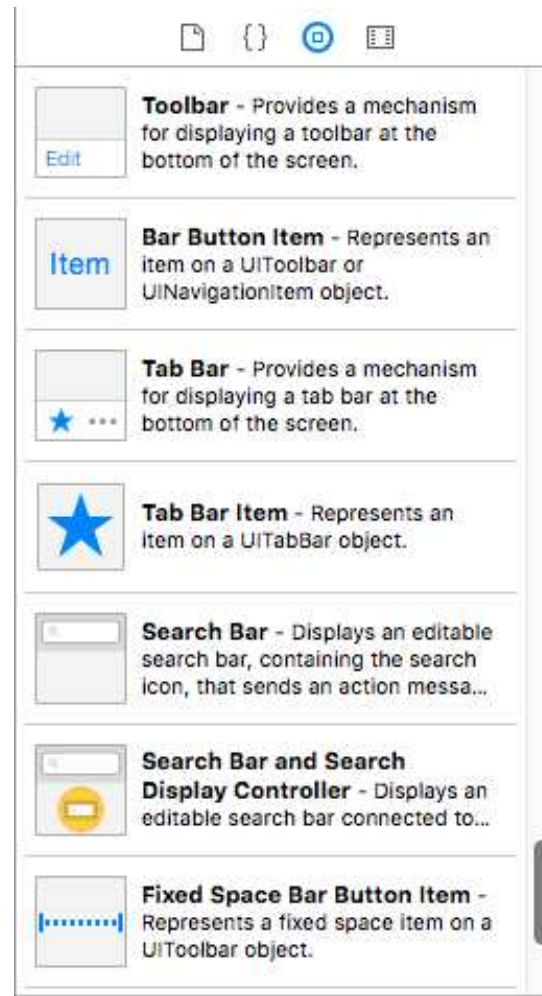
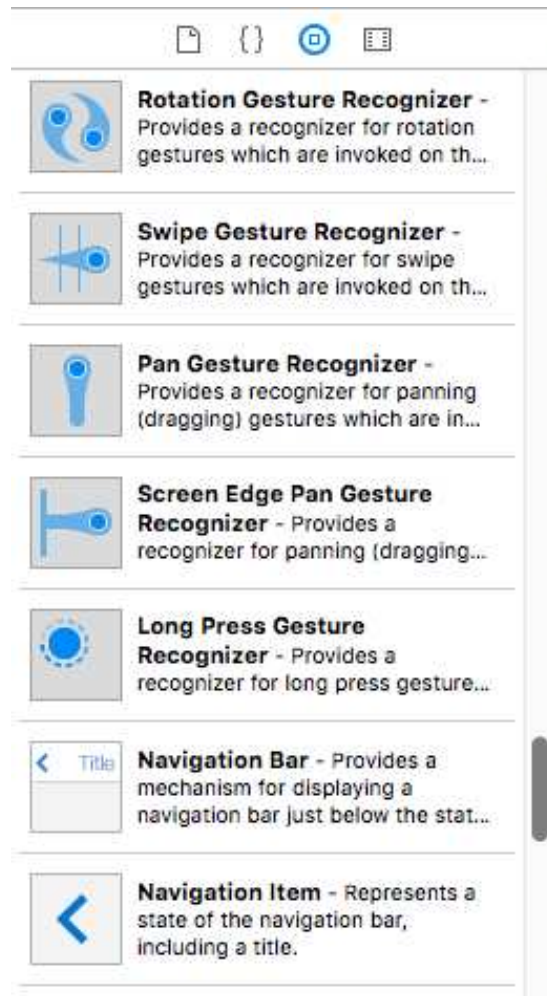


- Text View
 - Texteditor, mehrere Zeilen
- Scroll View
 - Erlaubt das UI-Element zu scrollen
- Date Picker
 - Datum auswählen
- Picker View
 - Jlist aus Java, eine Auswahl aus einer Liste
 - Benutzt einen Delegate, Konstruktor
- Stepper
 - Schalter + und –
- Table View Controller, JTable
- Tab Bar Controller
- SplitView, JPlitPane
- ImageView
- WebView

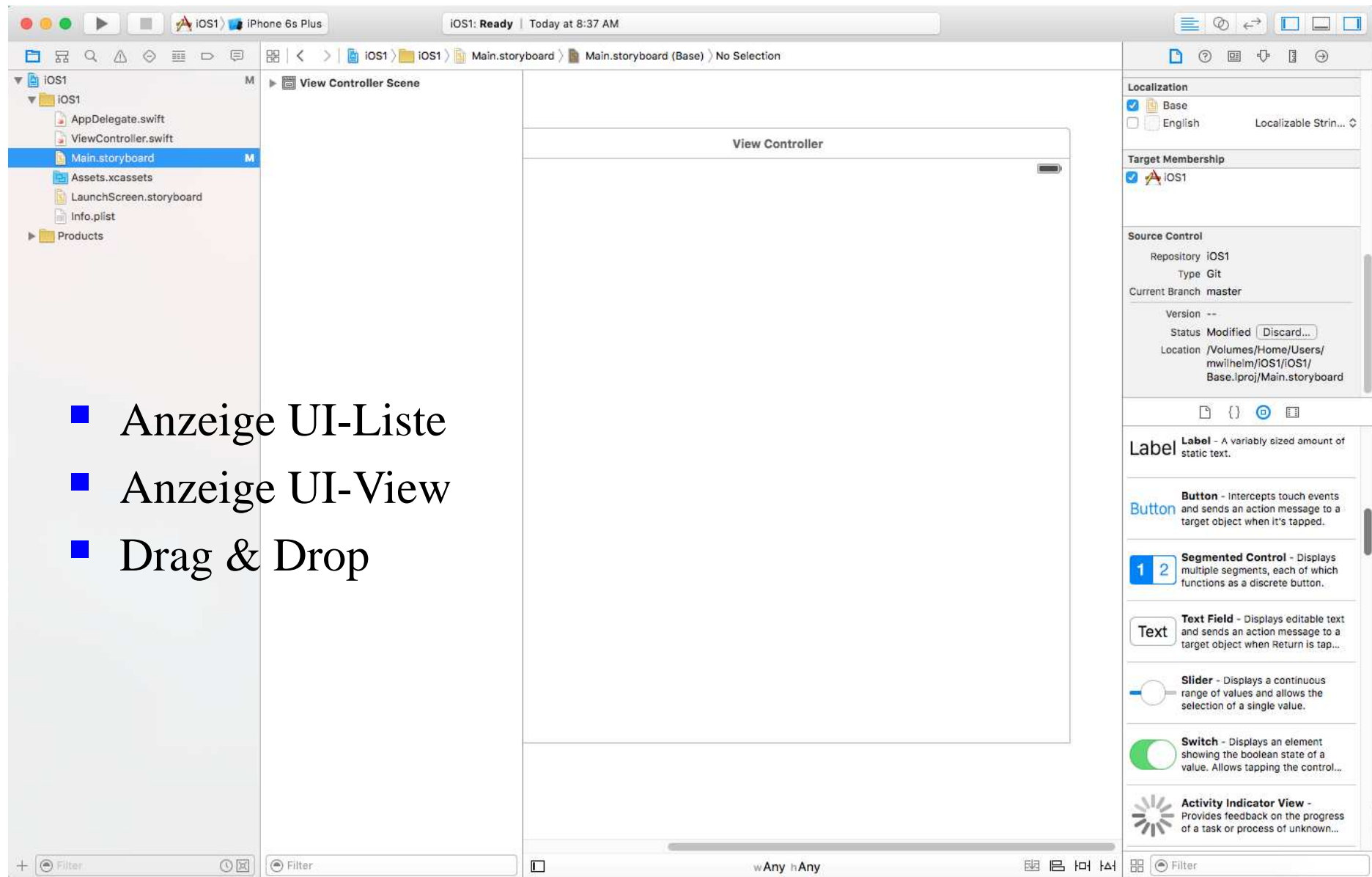
UI-Elemente in xcode



UI-Elemente in xcode

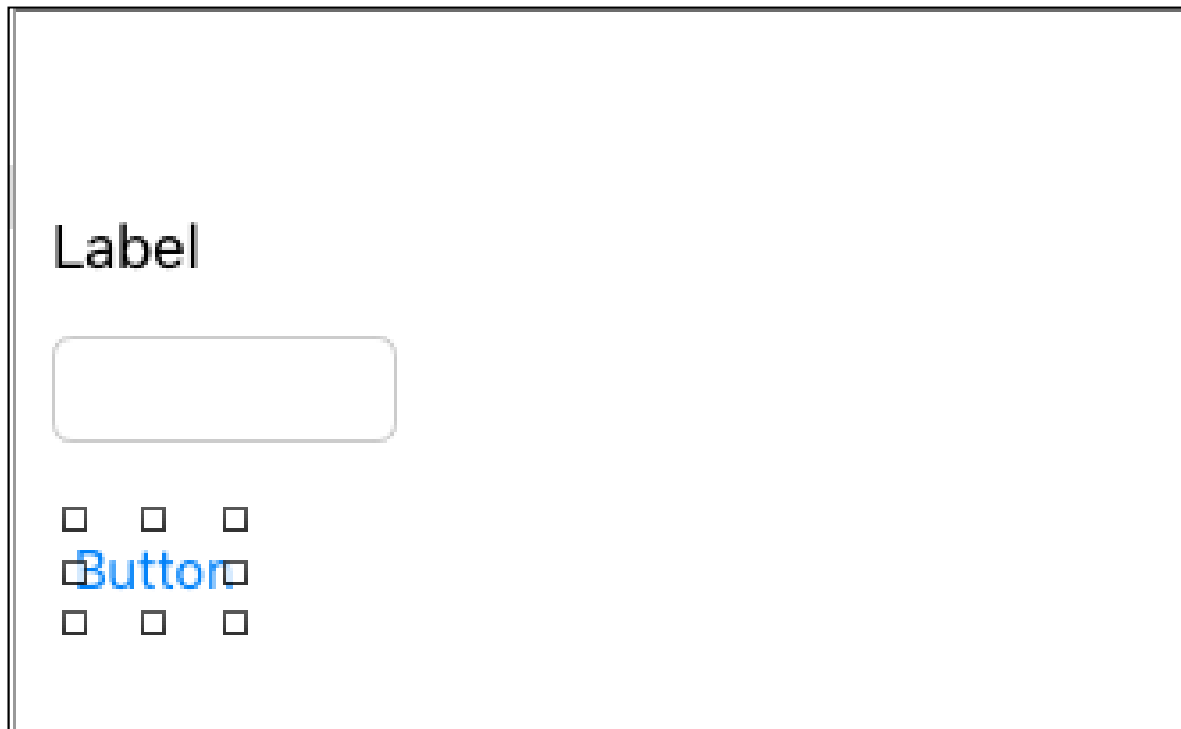


UI-Elemente in den UI-View eintragen



UI-Elemente in den UI-View eintragen

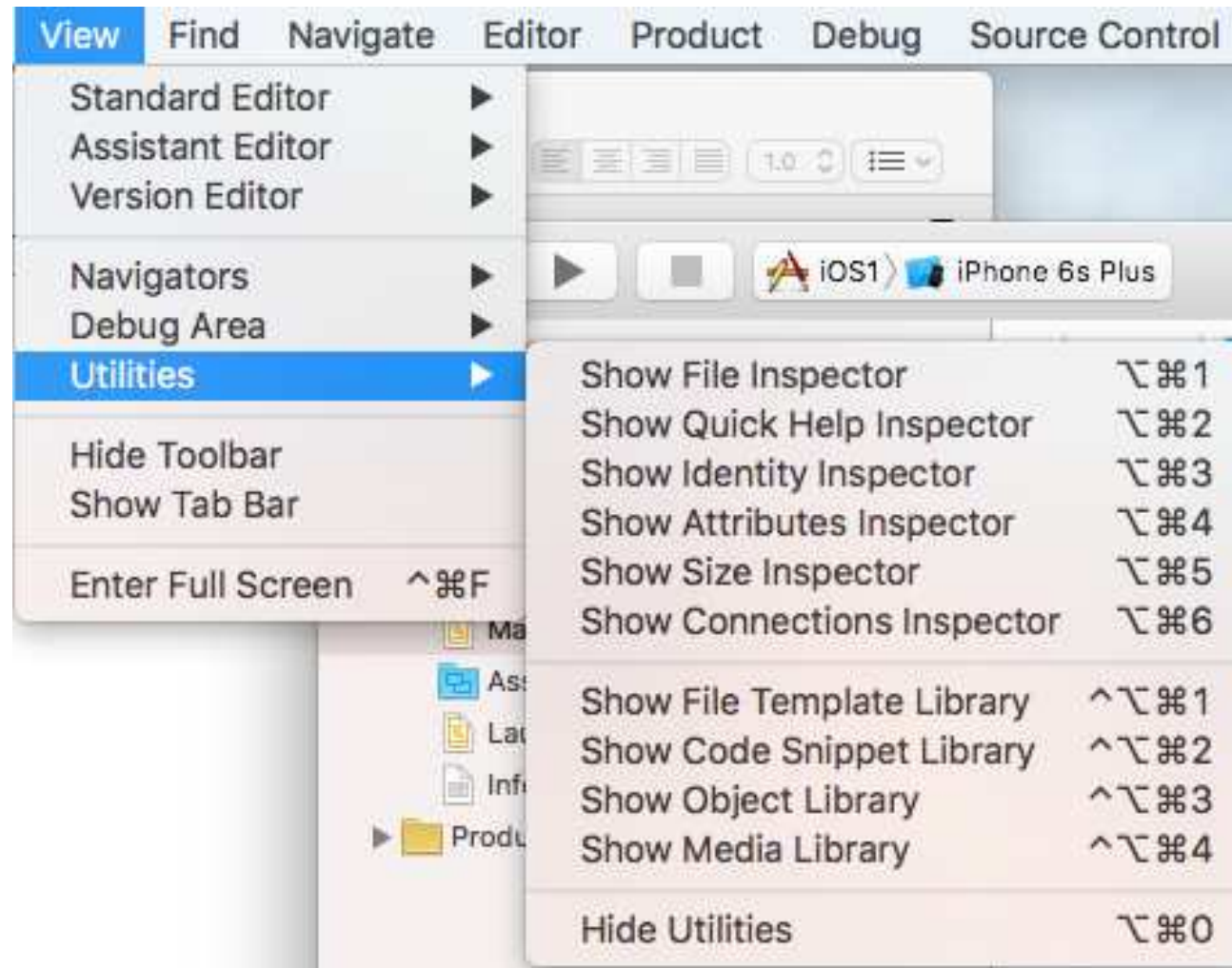
- UI-Elemente mittels Drag & Drop einfügen
- Hier noch ohne Layout



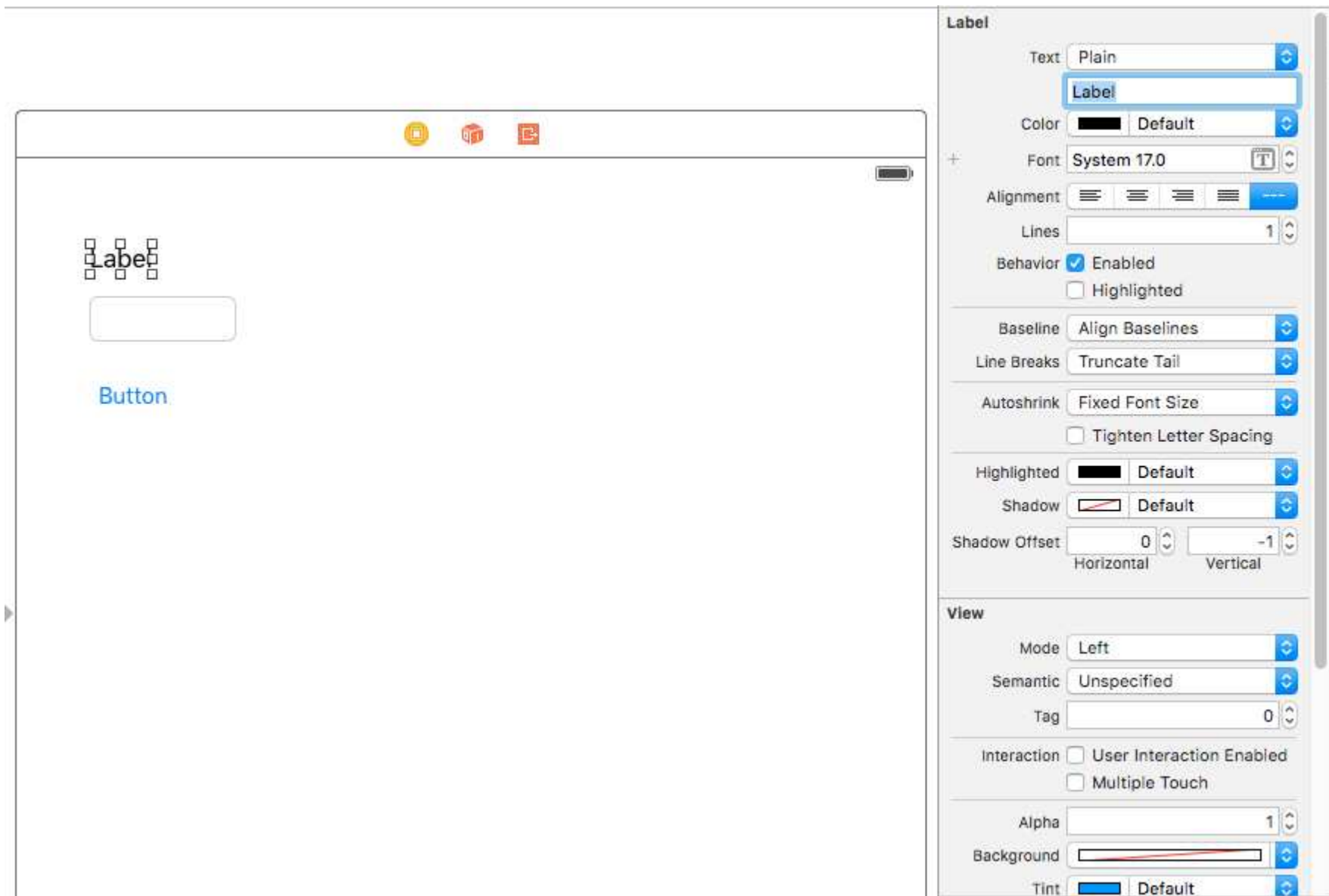
Zielgerät



UI-Elemente ändern: Show Attributes Inspector

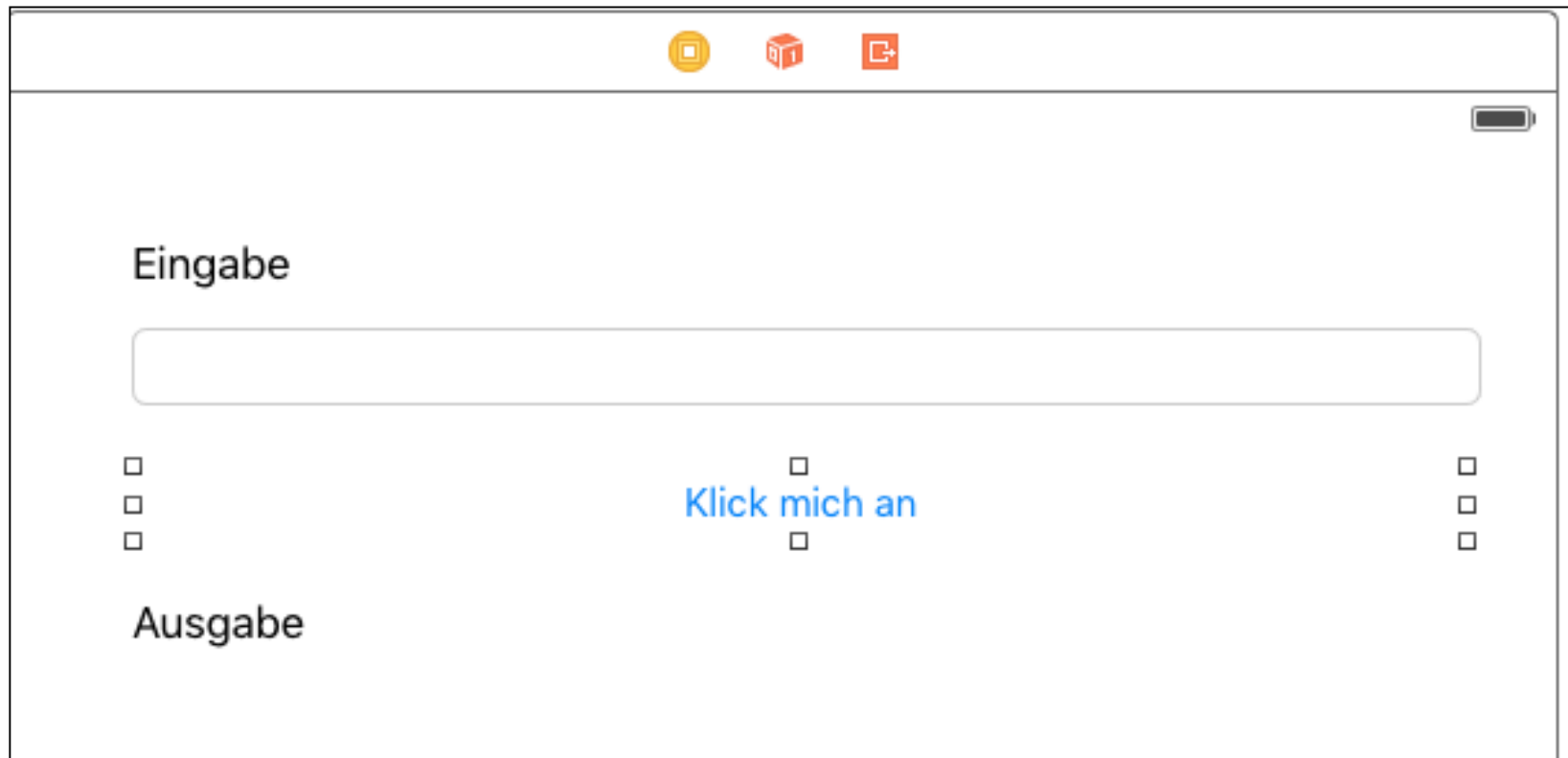


Attribute in UI-Elemente ändern: Name später!

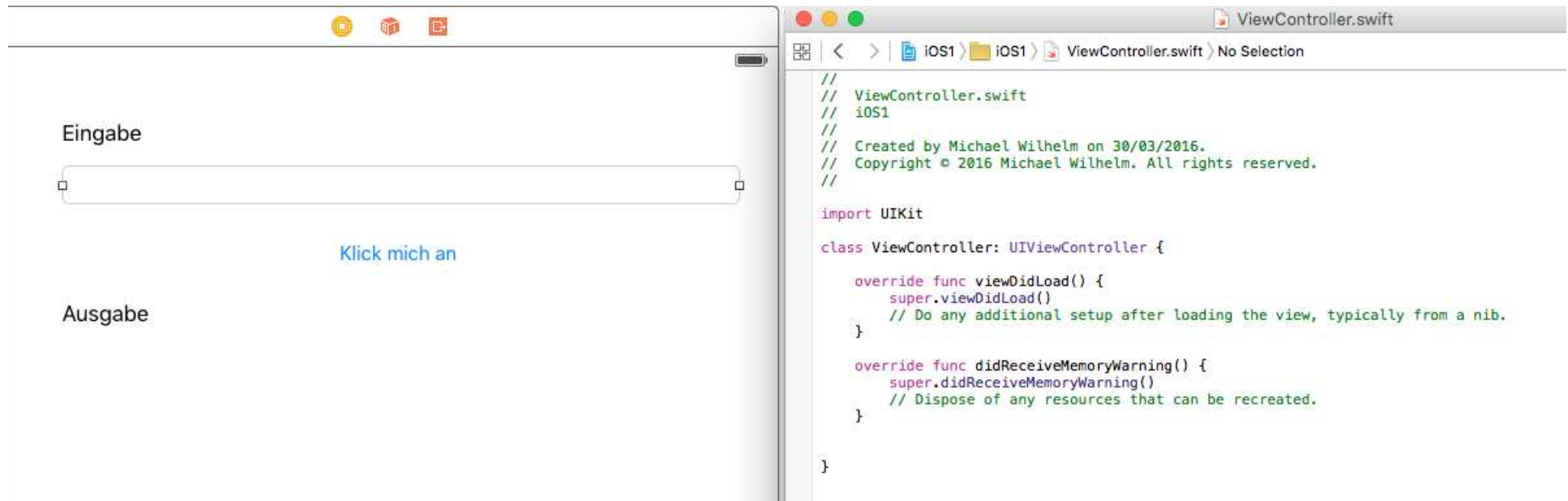


Ändern der Größe / Positionen der UI-Elemente

- UI-Elemente mittels Maus verschieben
- UI-Elemente mittels Maus verkleinern, vergrößern

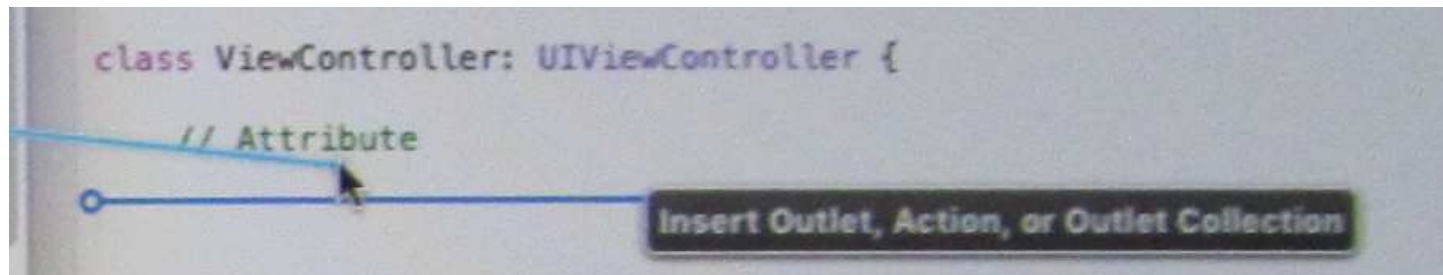
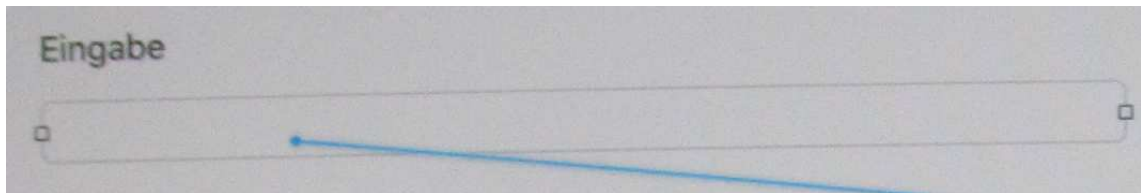
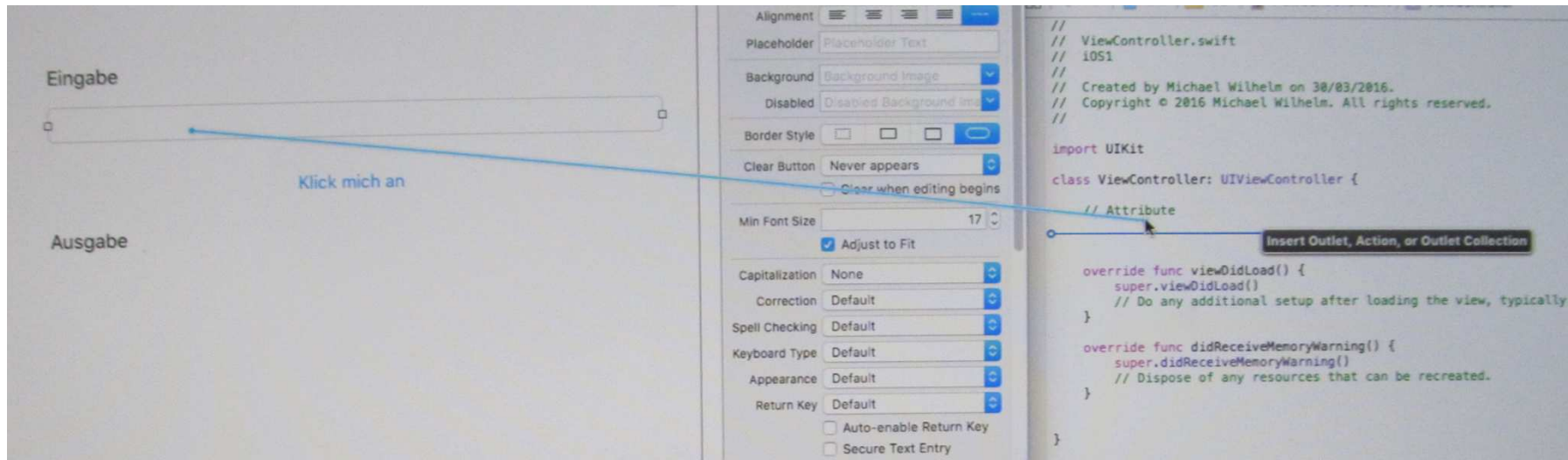


Referenz eines UI-Element in den Quellcode eintragen

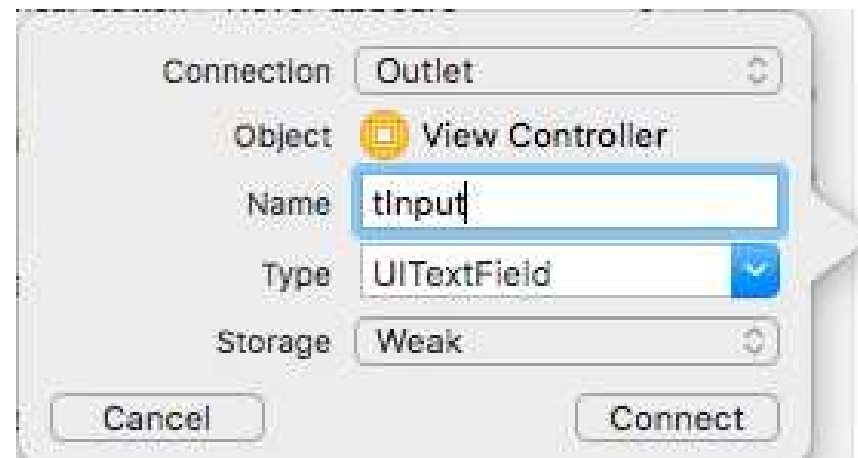
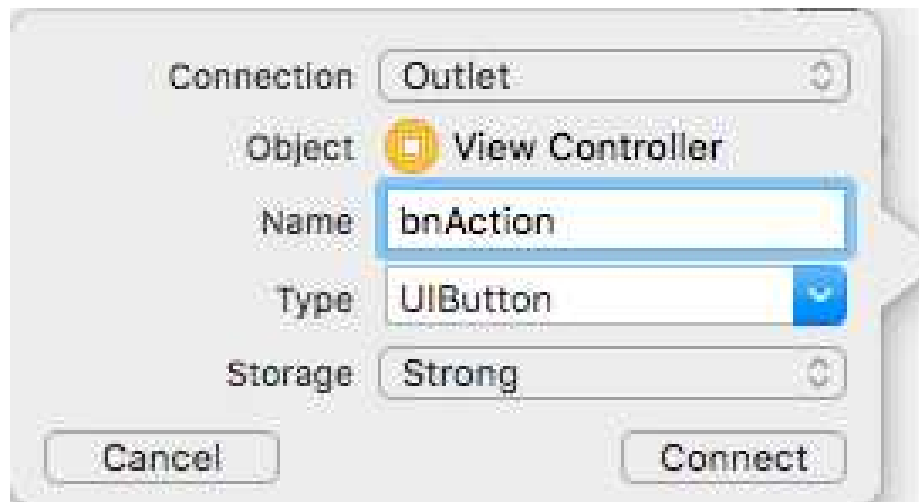
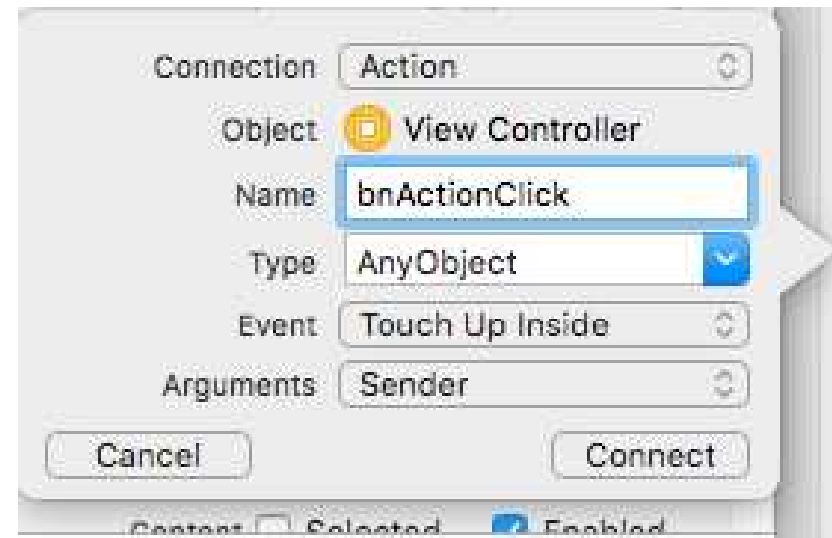
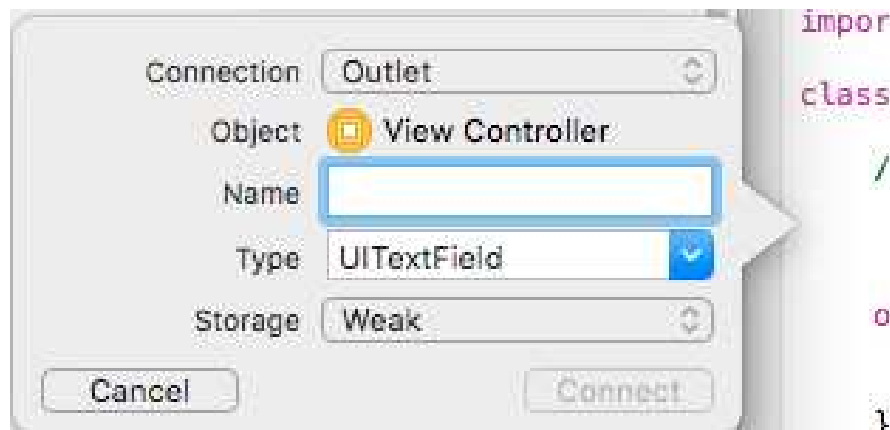


- View und Editor müssen sichtbar sein
- CTRL-Taste drücke
- Linke Maustaste drücken und zum Editor verschieben
- Oben, nach class { „einfügen“

Referenz eines UI-Element in den Quellcode eintragen



Eintragen der Details des Quellcodes



ViewController.swift: class ViewController: UIViewController {

// Attribute

var nr:Int32=0

@IBOutlet var bnAction: UITextField!

@IBOutlet var tInput: UITextField!

@IBOutlet var lblOutput: UILabel!

override func viewDidLoad() { }

override func didReceiveMemoryWarning() { }

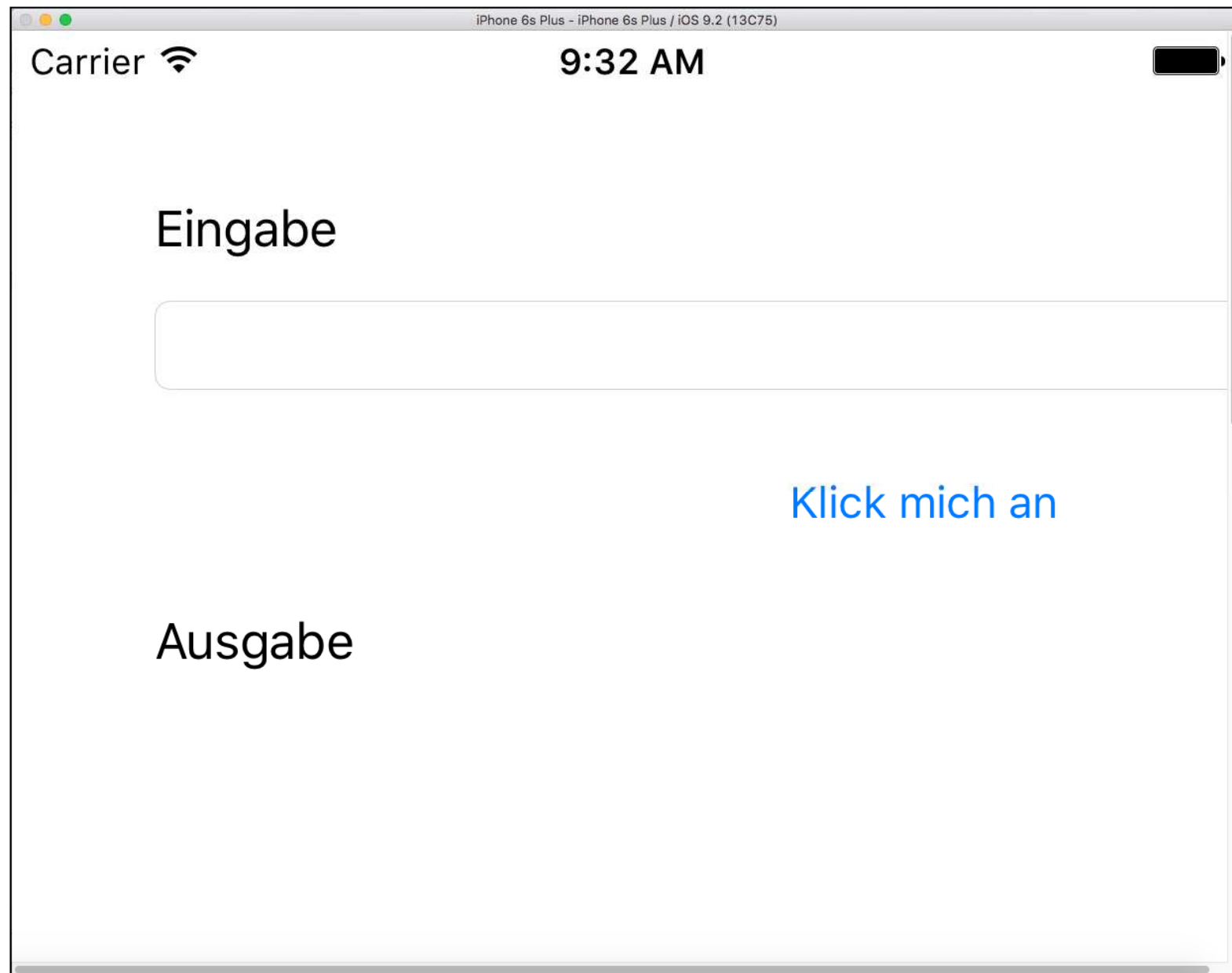
@IBAction func bnActionClick(sender: AnyObject) {

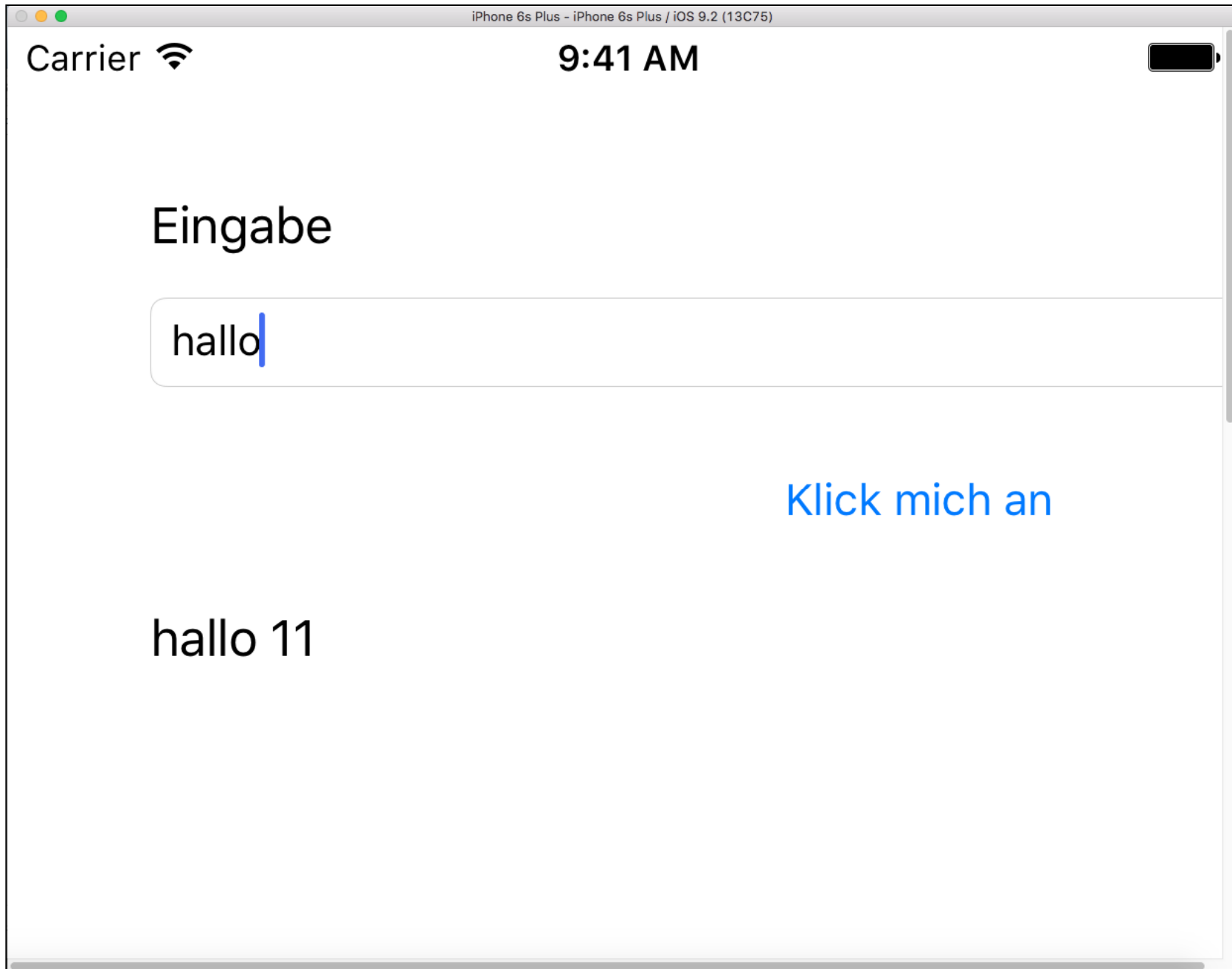
nr++

let input:String=tInput.text!

self.lblOutput.text=input+" "+String(nr)

}

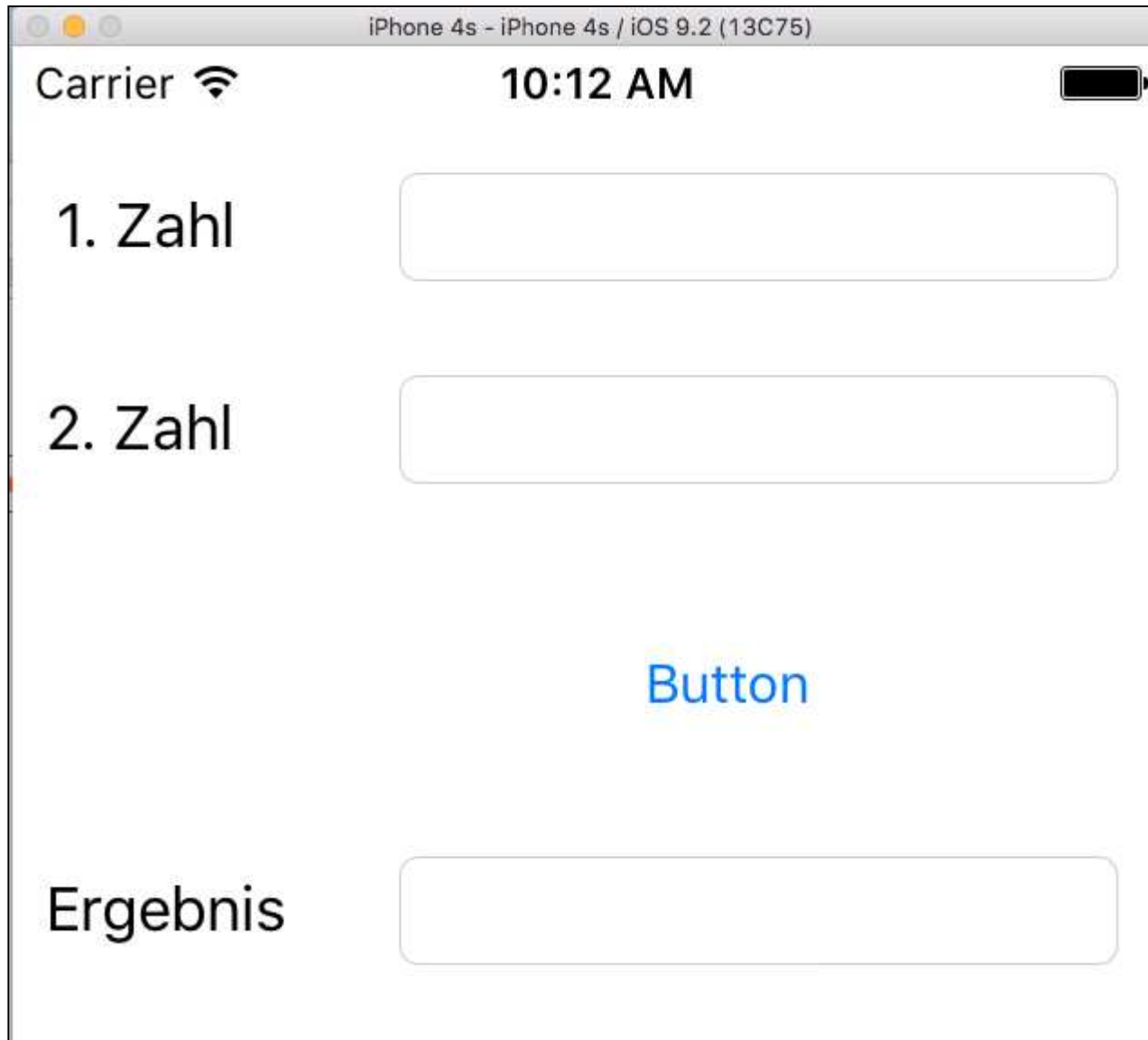




1. UI-Aufgabe

- Taschenrechner
 - add
- UI-Elemente
 - Label 1. Zahl
 - TextField 1. Zahl, uiAttribut
 - Label 2. Zahl
 - TextField 2. Zahl, uiAttribut
 - Button
 - Label Ergebnis
 - TextField Ergebnis, uiAttribut
- Methoden
 - bnActionClick

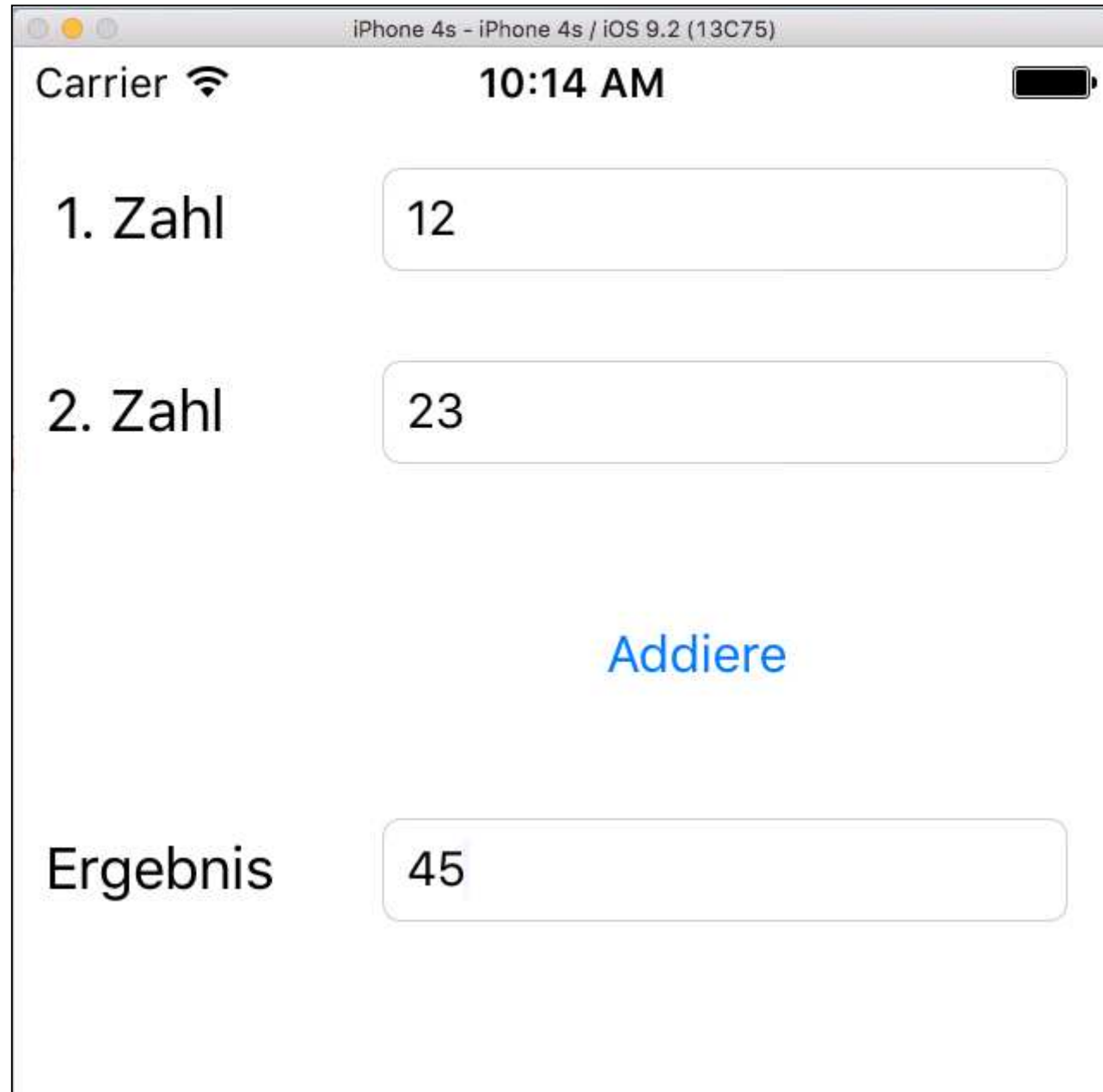
1. UI-Aufgabe: Taschenrechner



The image shows a screenshot of an iPhone 4s simulator running iOS 9.2. The status bar at the top displays "Carrier" with a signal icon, the time "10:12 AM", and a battery level icon. The app interface consists of three input fields and a button. The first input field is labeled "1. Zahl", the second "2. Zahl", and the third "Ergebnis". A blue button labeled "Button" is positioned between the second and third input fields.

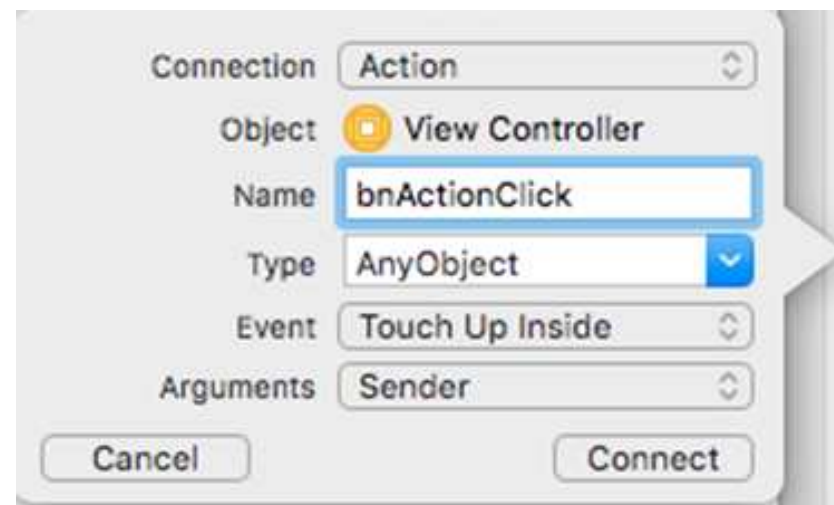
Label	Input Field
1. Zahl	<input type="text"/>
2. Zahl	<input type="text"/>
Ergebnis	<input type="text"/>

Button



1. UI-Aufgabe

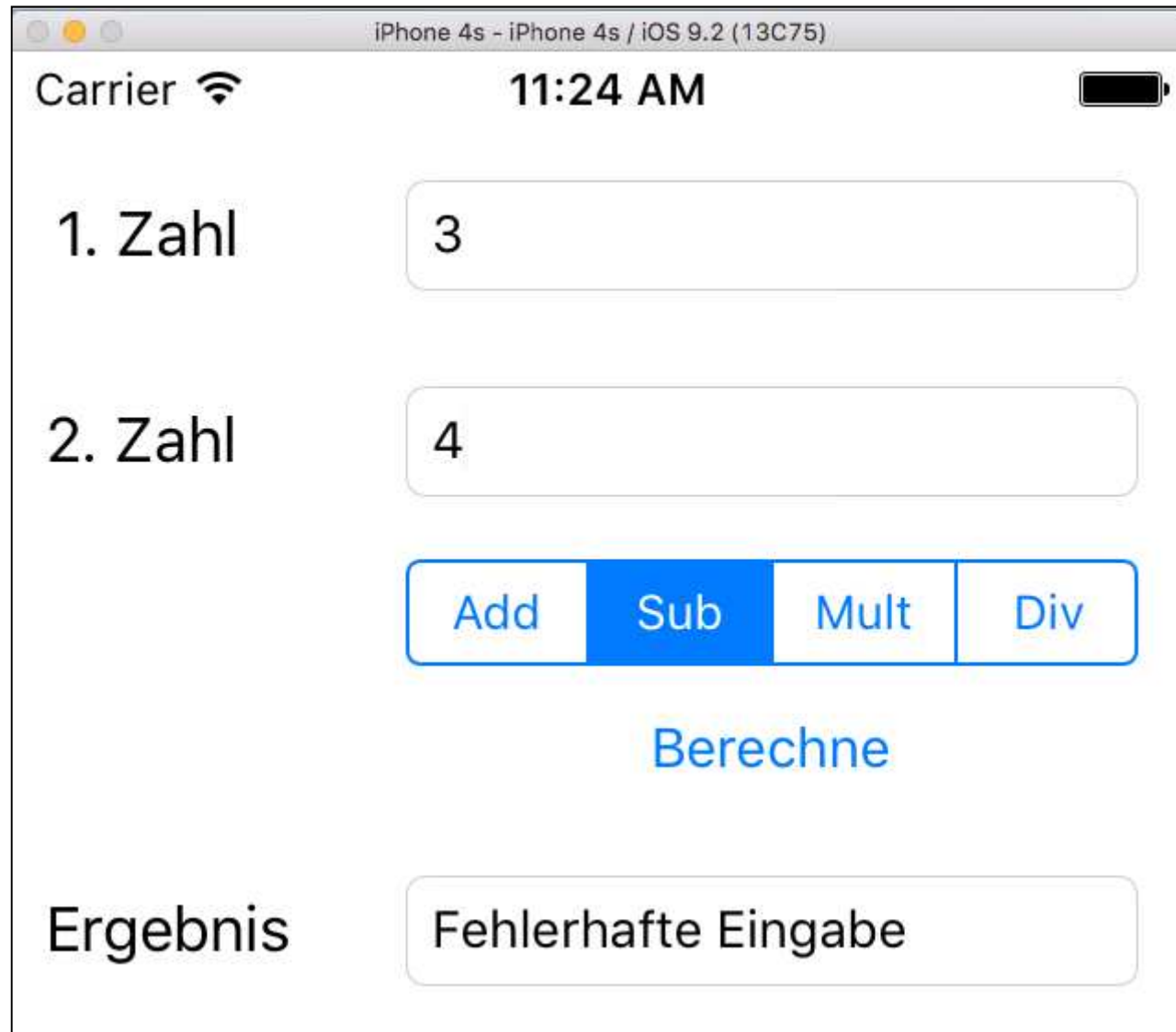
- UI-Elemente in den View eintragen
- UI-Elemente als Referenz eintragen
- Button-Event eintragen



1. UI-Aufgabe: Ablauf

- Ausgabe eines Textes in der Event-Methode
 - `tOutputCStrong.text!="Hallo"`
- Auswerten der Eingangsdaten
 - `let sA:String = tInputAStrong.text!`
- Ausgabe
 - `tOutputCStrong.text!=String(c)`

1. UI-Aufgabe: Erweiterung mit ein SegmentedControl

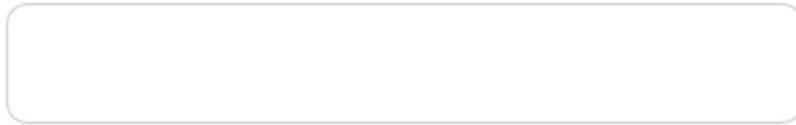


UI-Element UILabel

1. Zahl

- Attribute
 - text
 - font, size etc.
- Methoden
 - keine

UI-Element UITextField



- Attribute
 - text
 - font, size etc.
- Methoden
 - ValueChanged

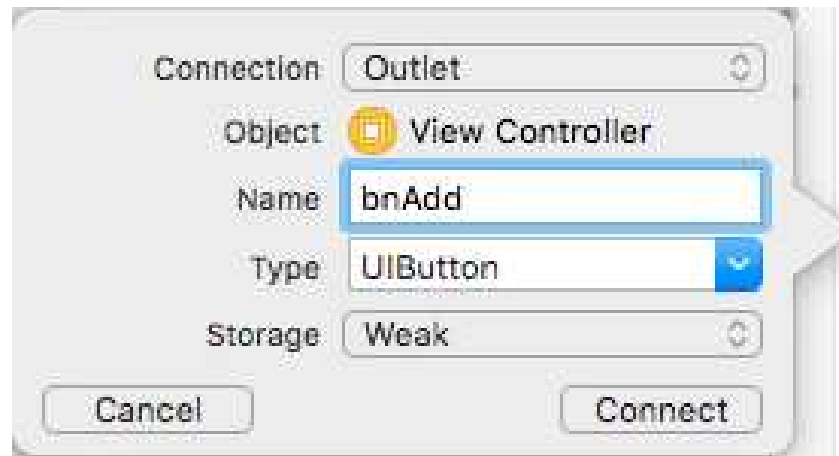
Eintragen als Referenz (strong)



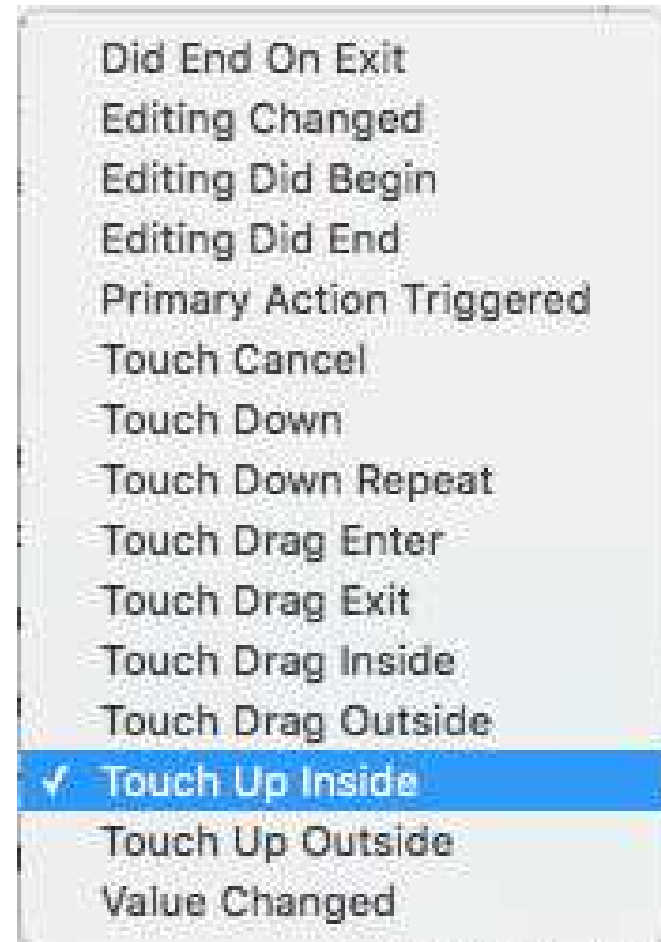
UI-Element UIButton

- Attribute
 - text
 - font, size etc.
- Methoden
 - Touch Up Inside

Eintragen als Referenz (strong)

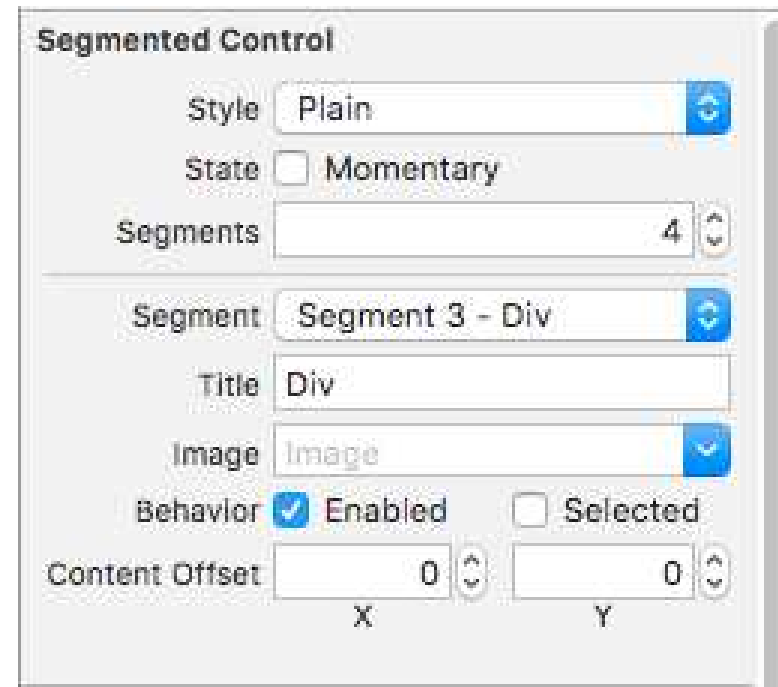
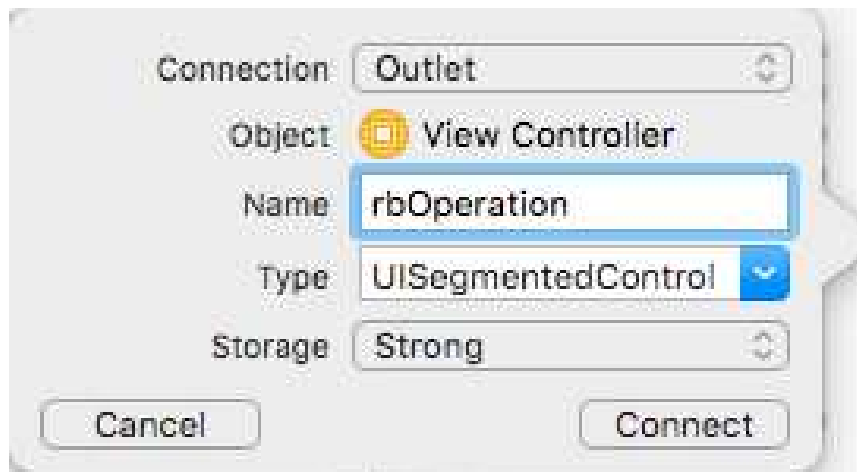


Action



UI-Element UISegmentedControl (RadioButtons)

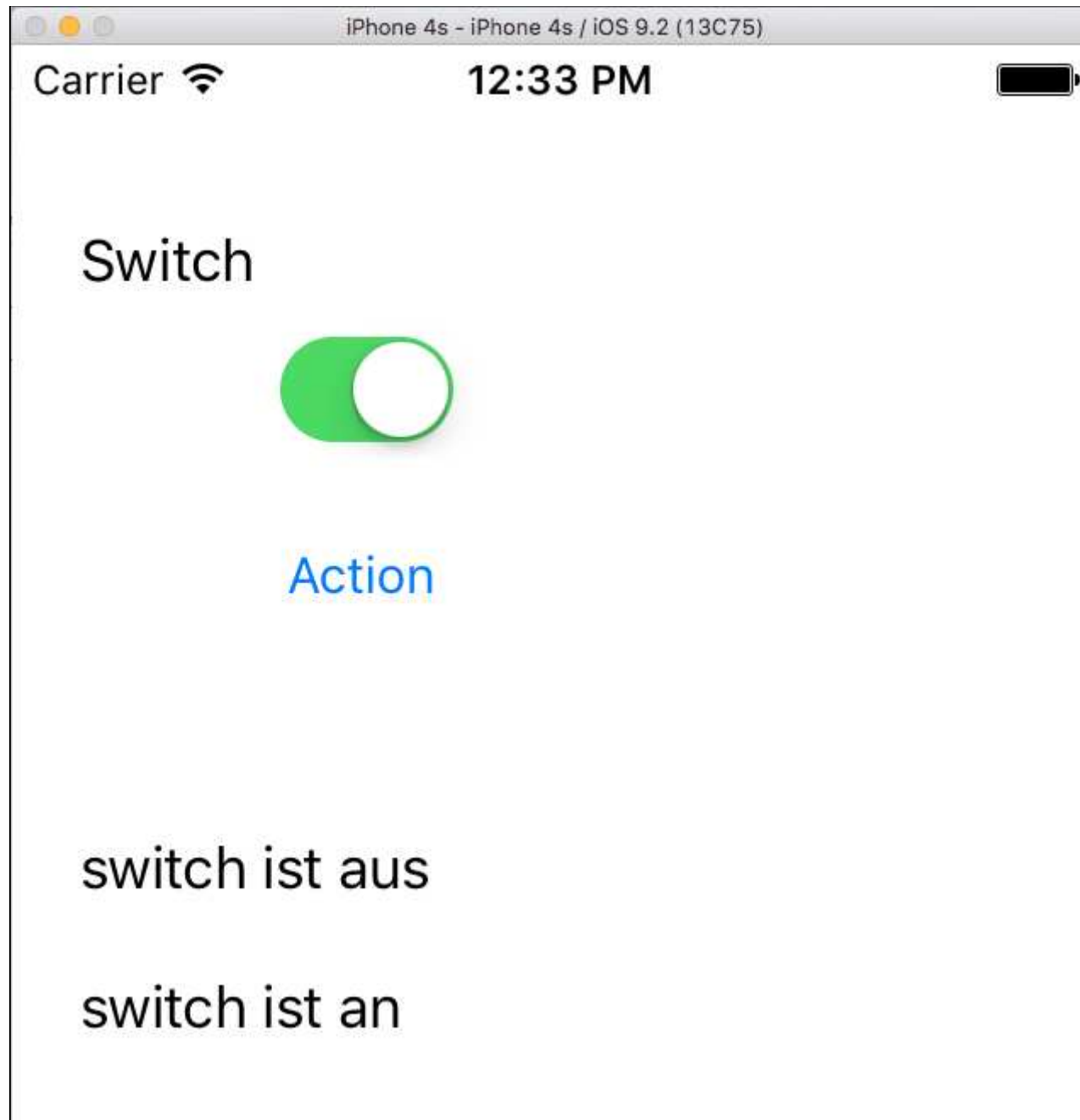
- Attribute
 - Anzahl Buttons
 - text
 - font, size etc.
 - selectedIndex
- Methoden
 - ChangeValue



UI-Element UISwitch

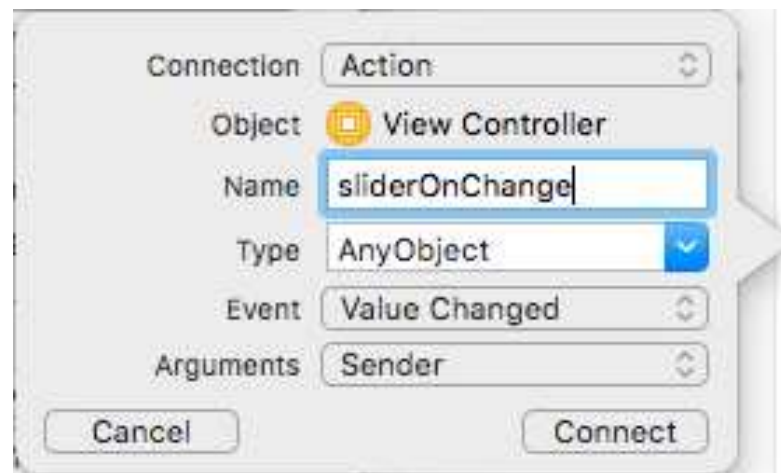
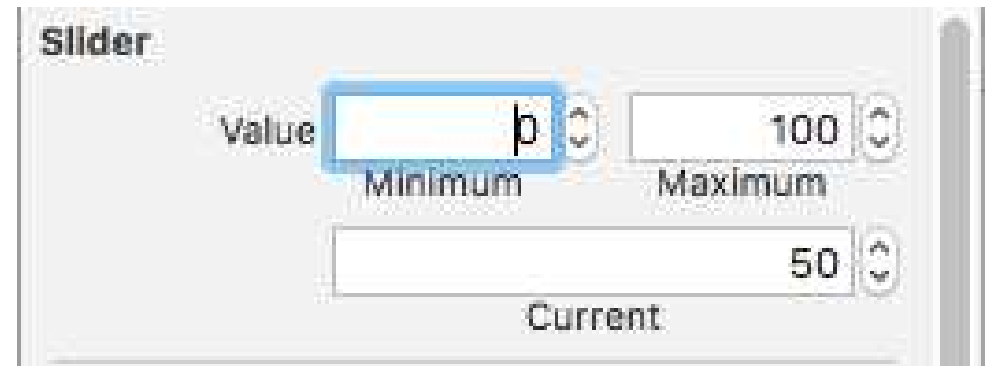


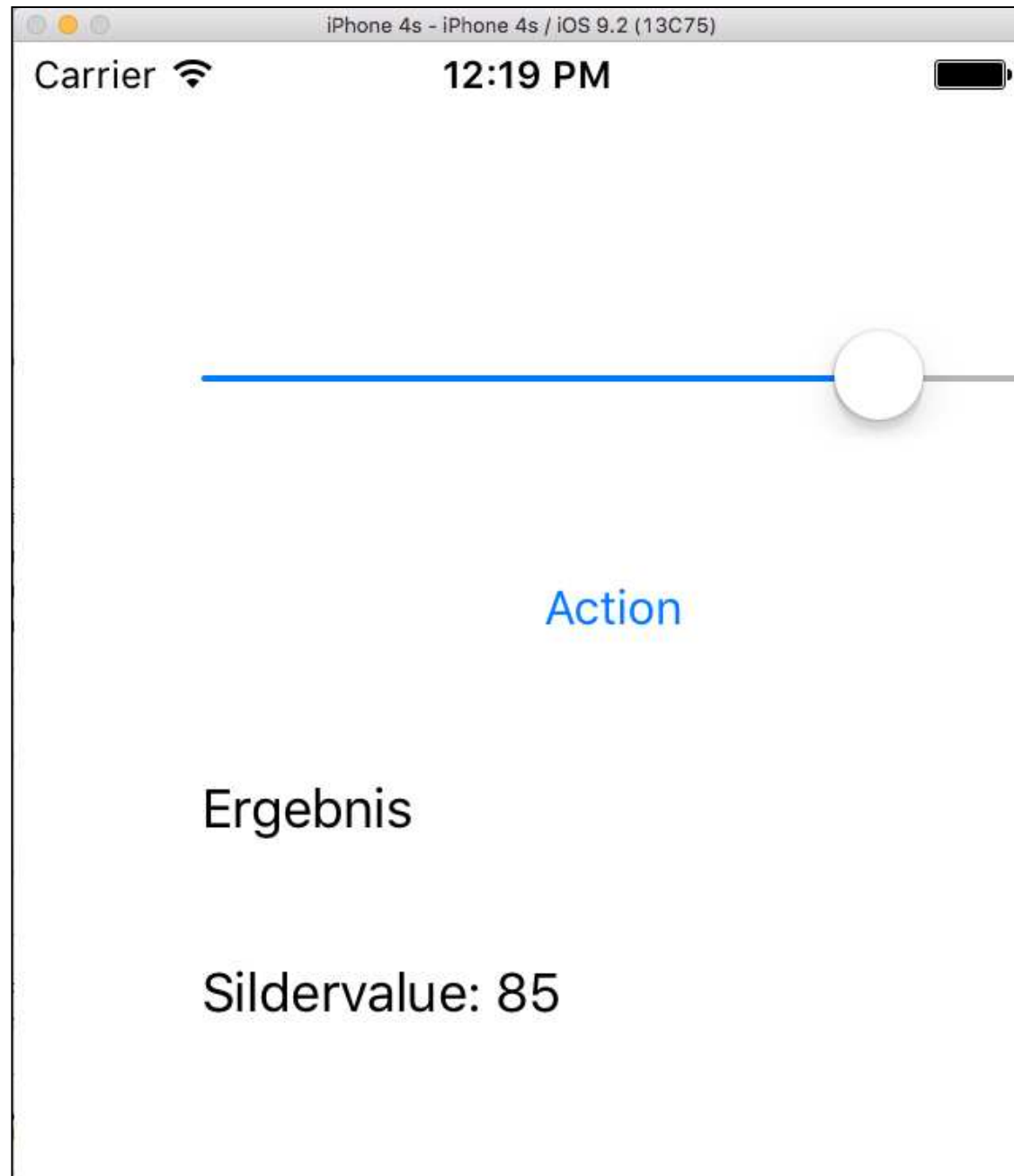
- Attribute
 - on
 - if uiSwitch.on {
 - }
 - font, size etc.
- Methoden
 - ValueChanged



UI-Element UISlider

- Attribute
 - font, size etc.
 - minimum
 - maximum
 - value
- Methoden
 - Value Changed

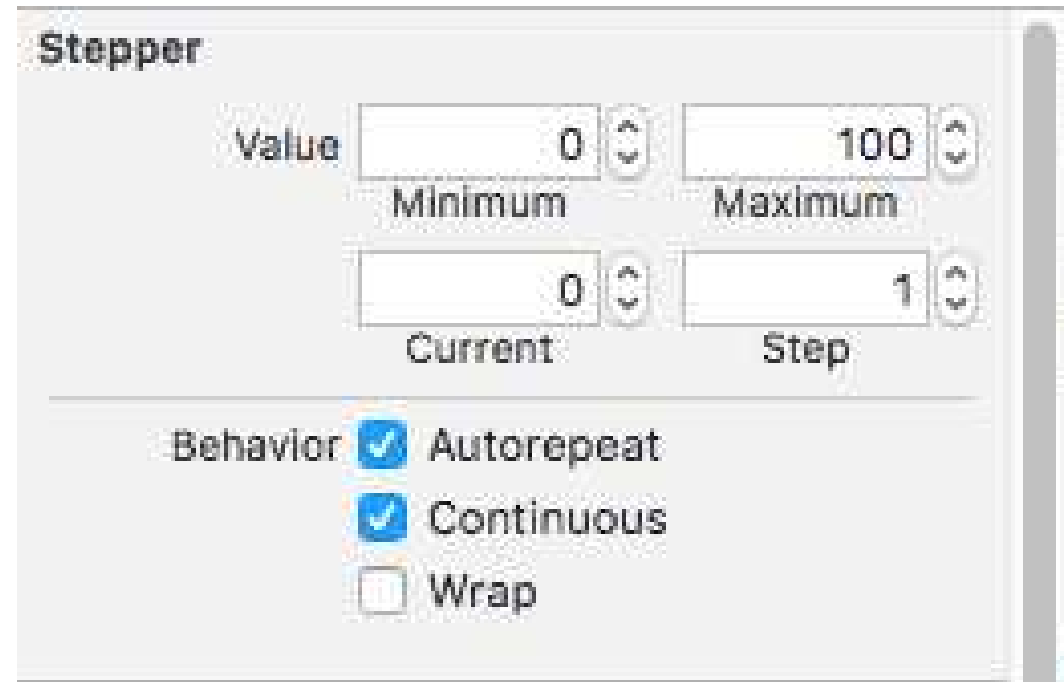


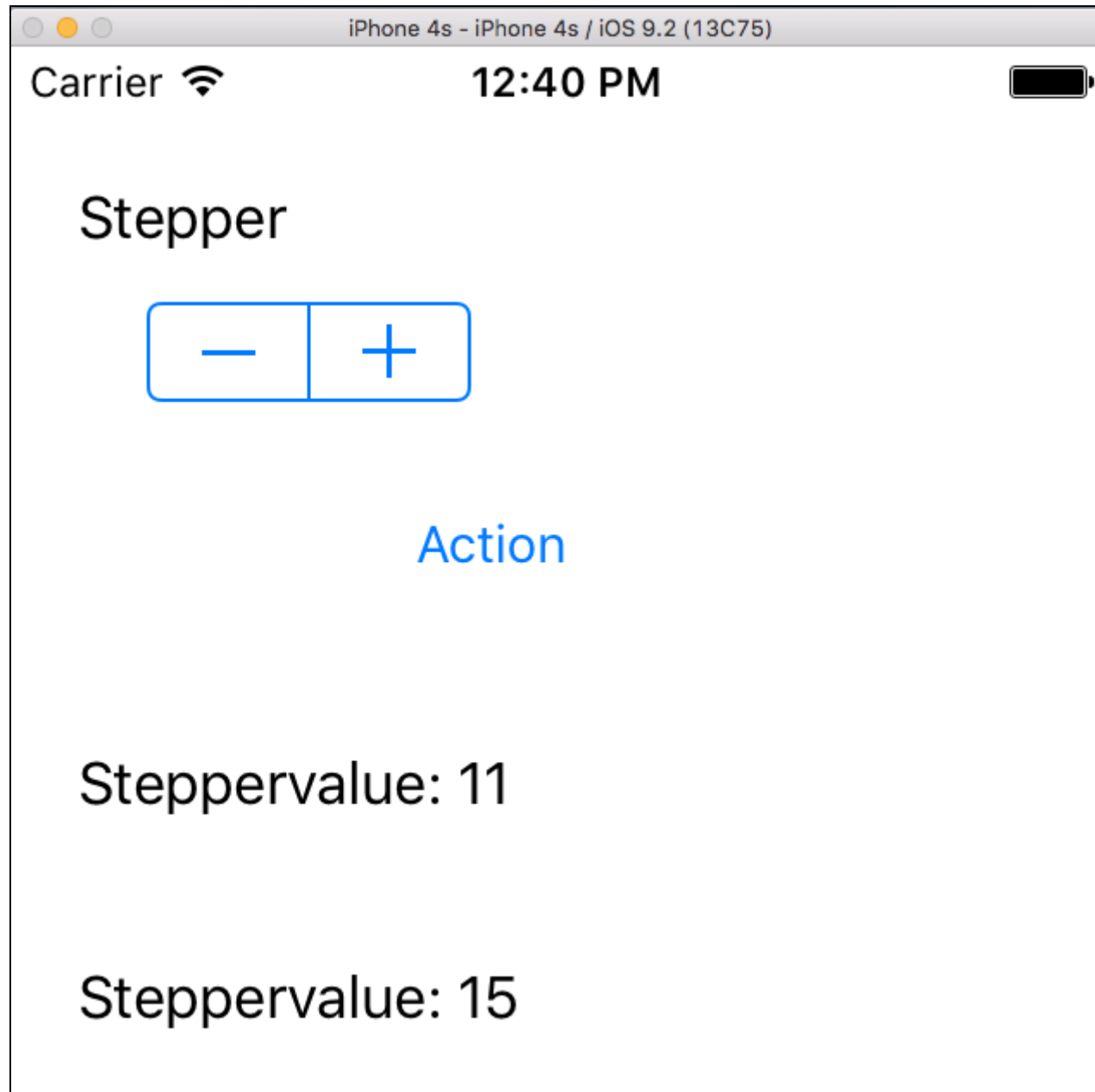


UI-Element UIStepper



- Attribute
 - font, size etc.
 - minimum
 - maximum
 - value
- Methoden
 - ValueChanged





UI-Element UIProgressView

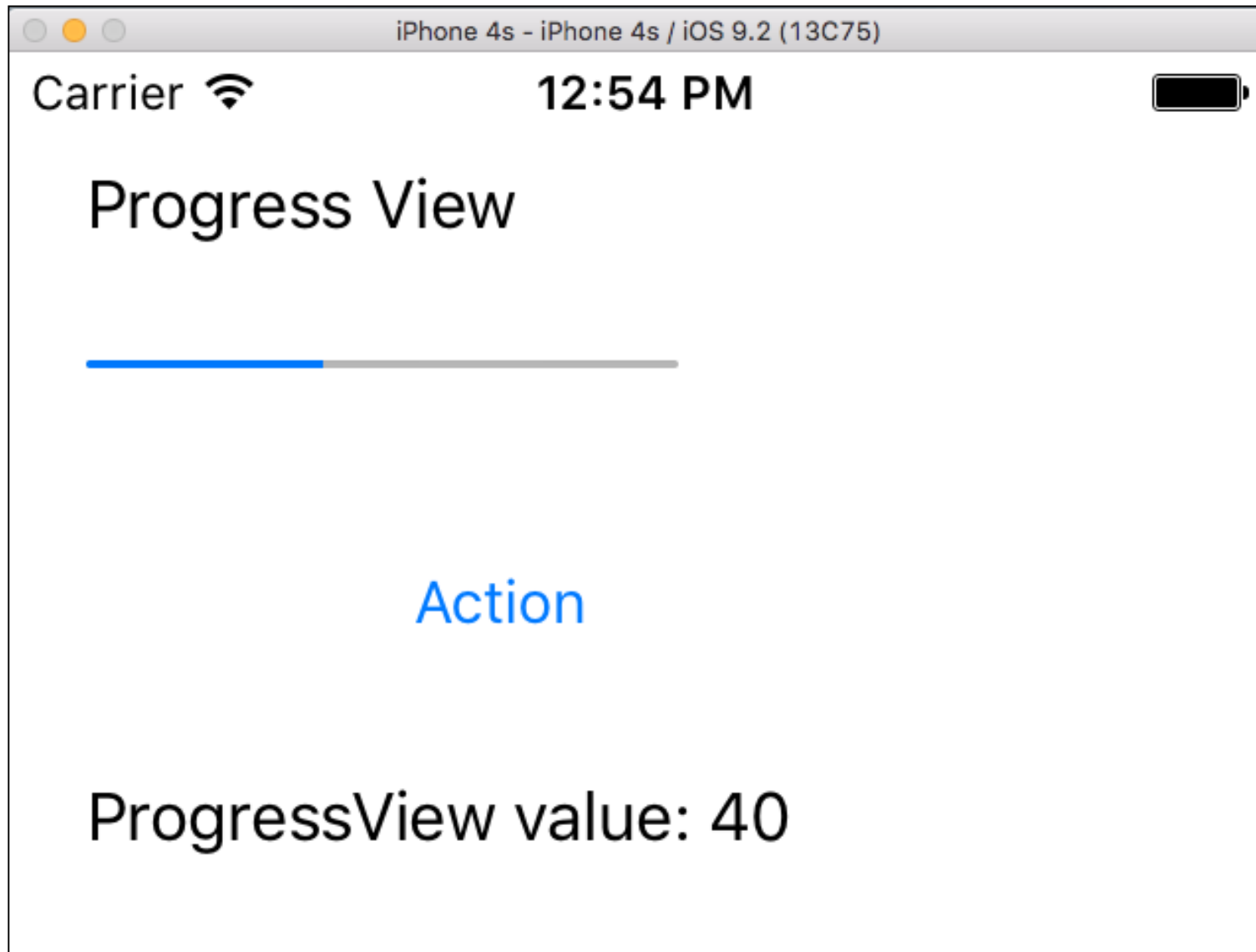
- Attribute

- font, size etc.
- minimum immer 0,0
- maximum immer 1,0
- value

- Methoden

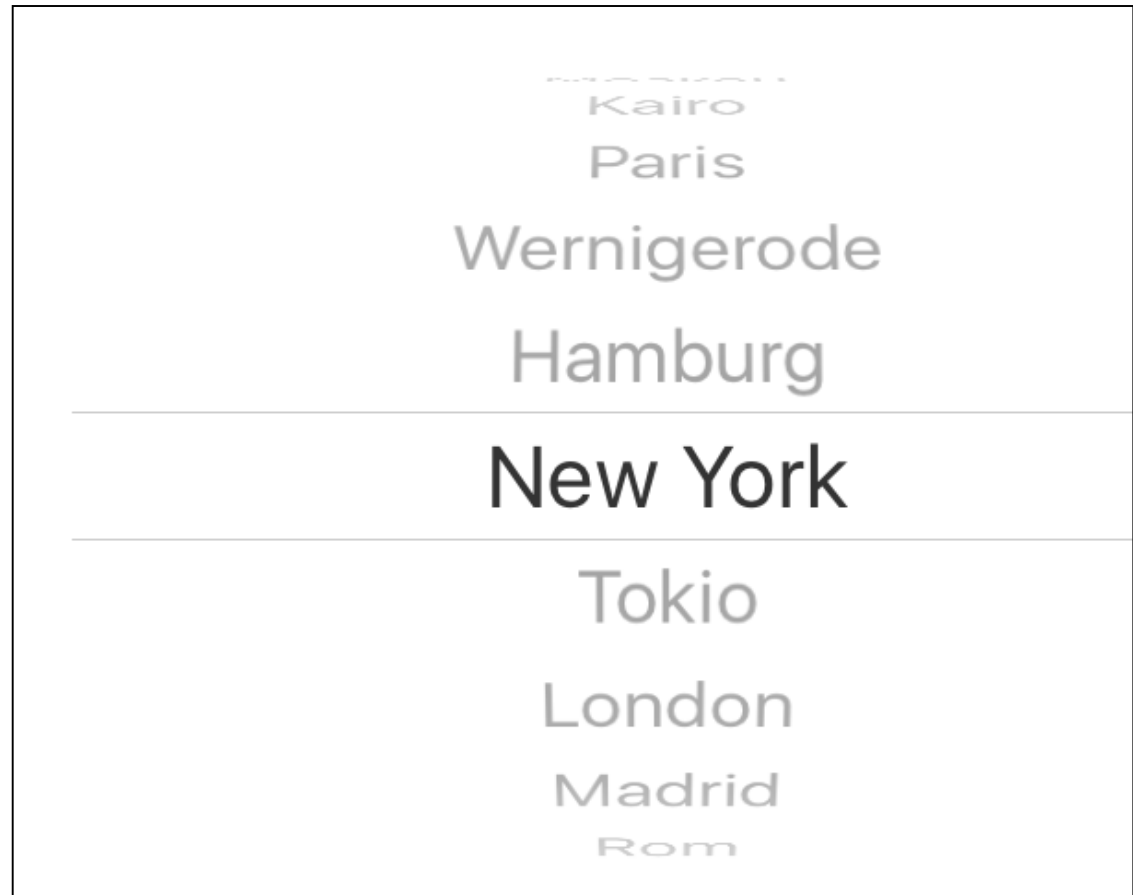
- ValueChanged

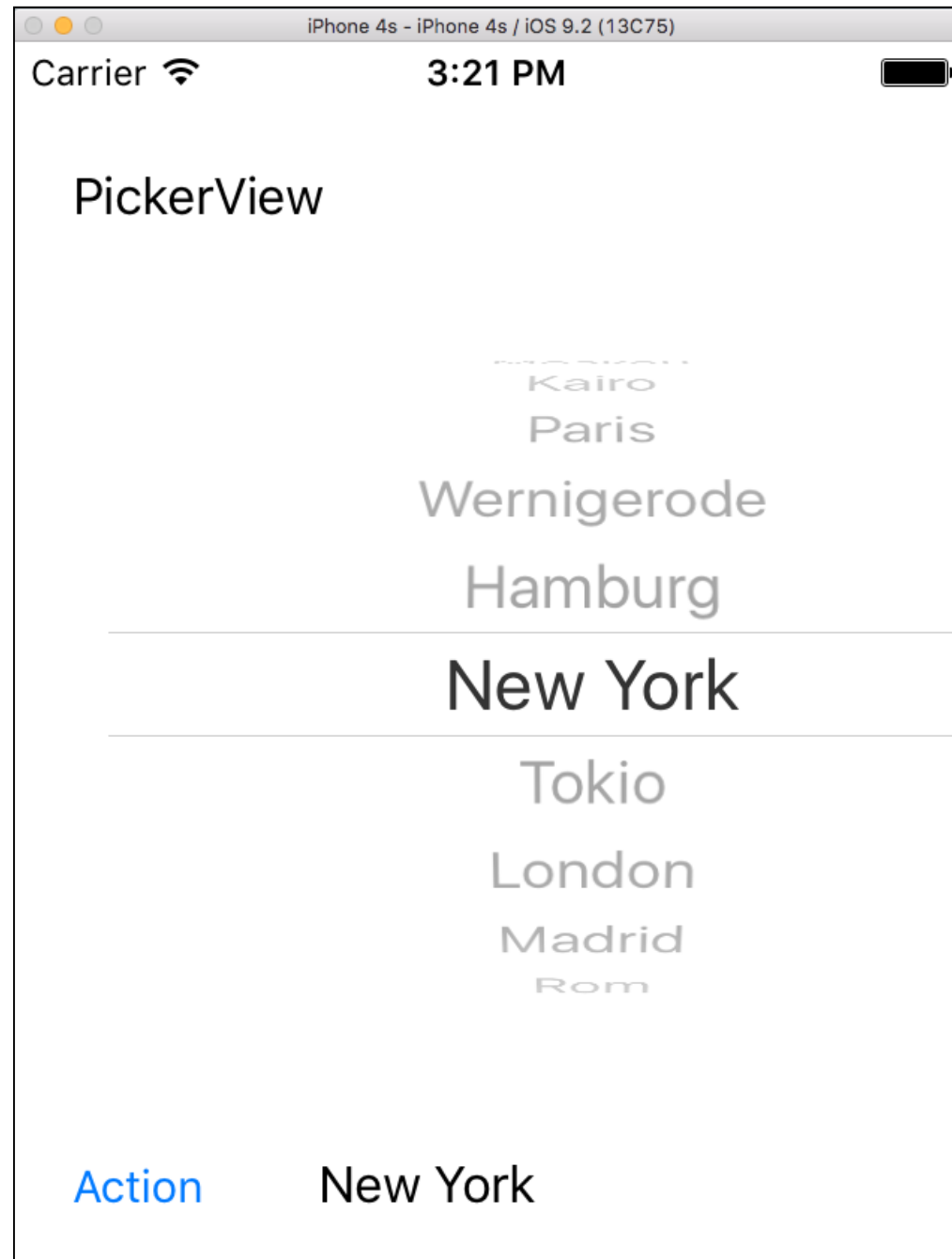




UI-Element UIPickerView, ca. eine JList

- Attribute
 - font, size etc.
 - delegate
- Methoden
 - ValueChanged





```

class ViewController: UIViewController, UIPickerViewDelegate{
    let staedte = ["Brüssel", "Mailand", "Moskau", "Rom"]
    @IBOutlet var uiPickerView: UIPickerView!

    func numberOfComponentsInPickerView(pickerView:UIPickerView)-> Int{
        return 1
    }

    func pickerView(pickerView:UIPickerView,
                     numberOfRowsInComponent component:Int) -> Int{
        return staedte.count
    }

    func pickerView(pickerView:UIPickerView, titleForRow row:Int,
                     forComponent component:Int) -> String?{
        return staedte[row]
    }
}

```

```
class ViewController: UIViewController, UIPickerViewDelegate{
```

```
    func pickerView(pickerView: UIPickerView, didSelectRow row: Int,  
                    inComponent component: Int) {  
        labelErgebnis.text = staedte[row]  
    }
```

```
    override func viewDidLoad() {  
        super.viewDidLoad()  
        // Do any additional setup after loading the view, typically from a nib.  
        self.uiPickerView.delegate=self  
    }
```