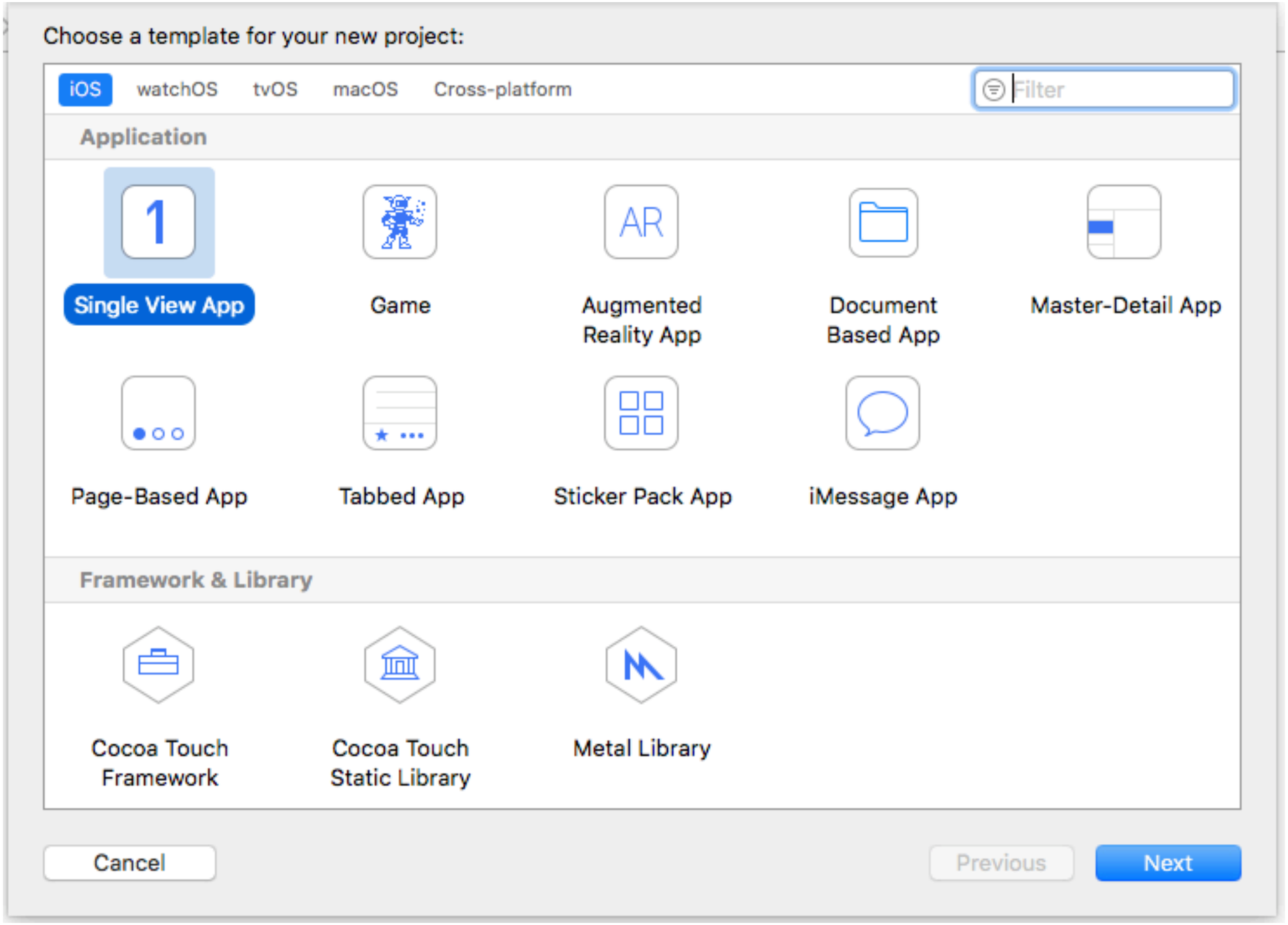
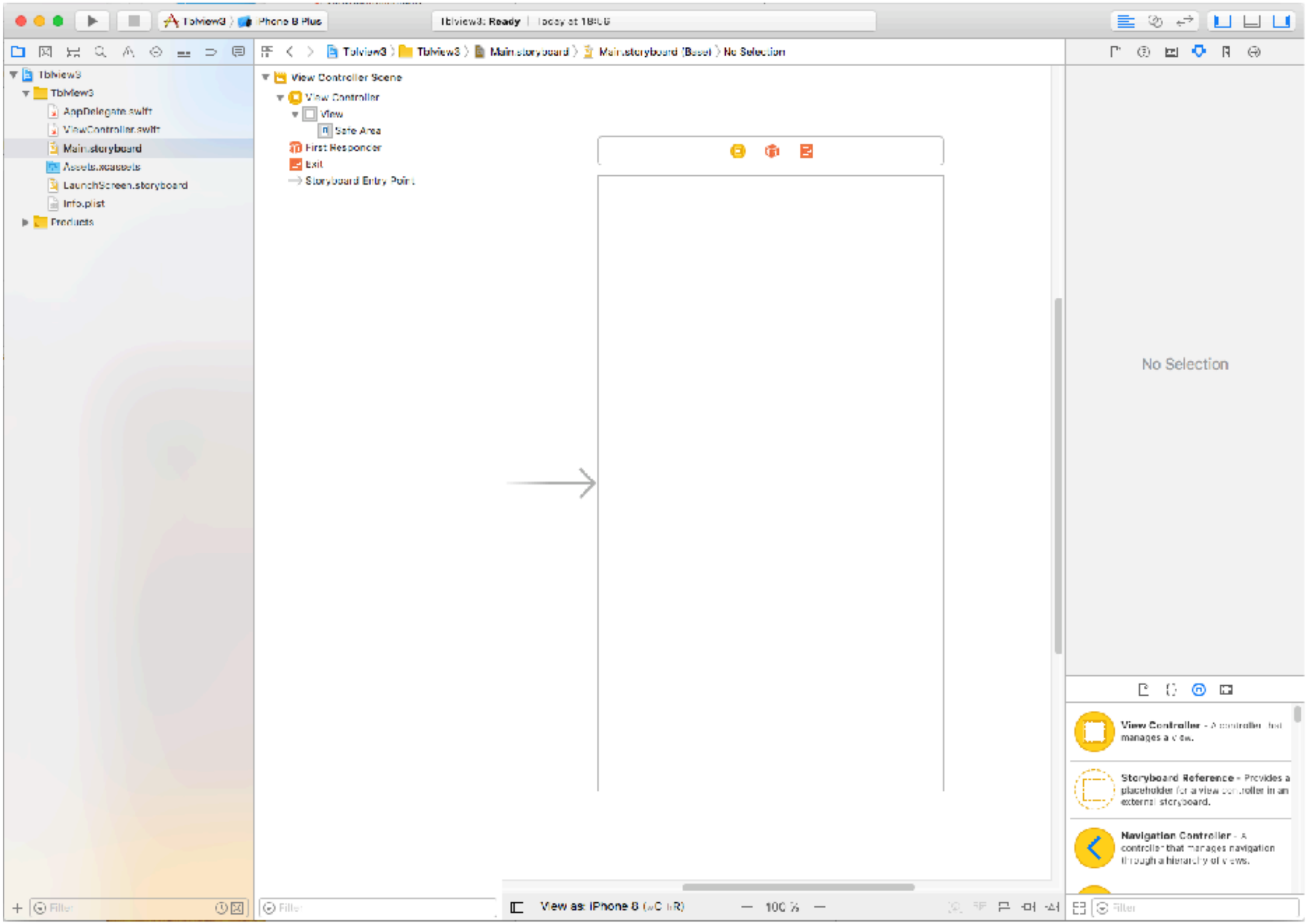
TableView mit einem Navigationskontroller

- Hauptfenster mit einem TableView
- Navigationskontroller
- Editieren mit einem DetailsViewController

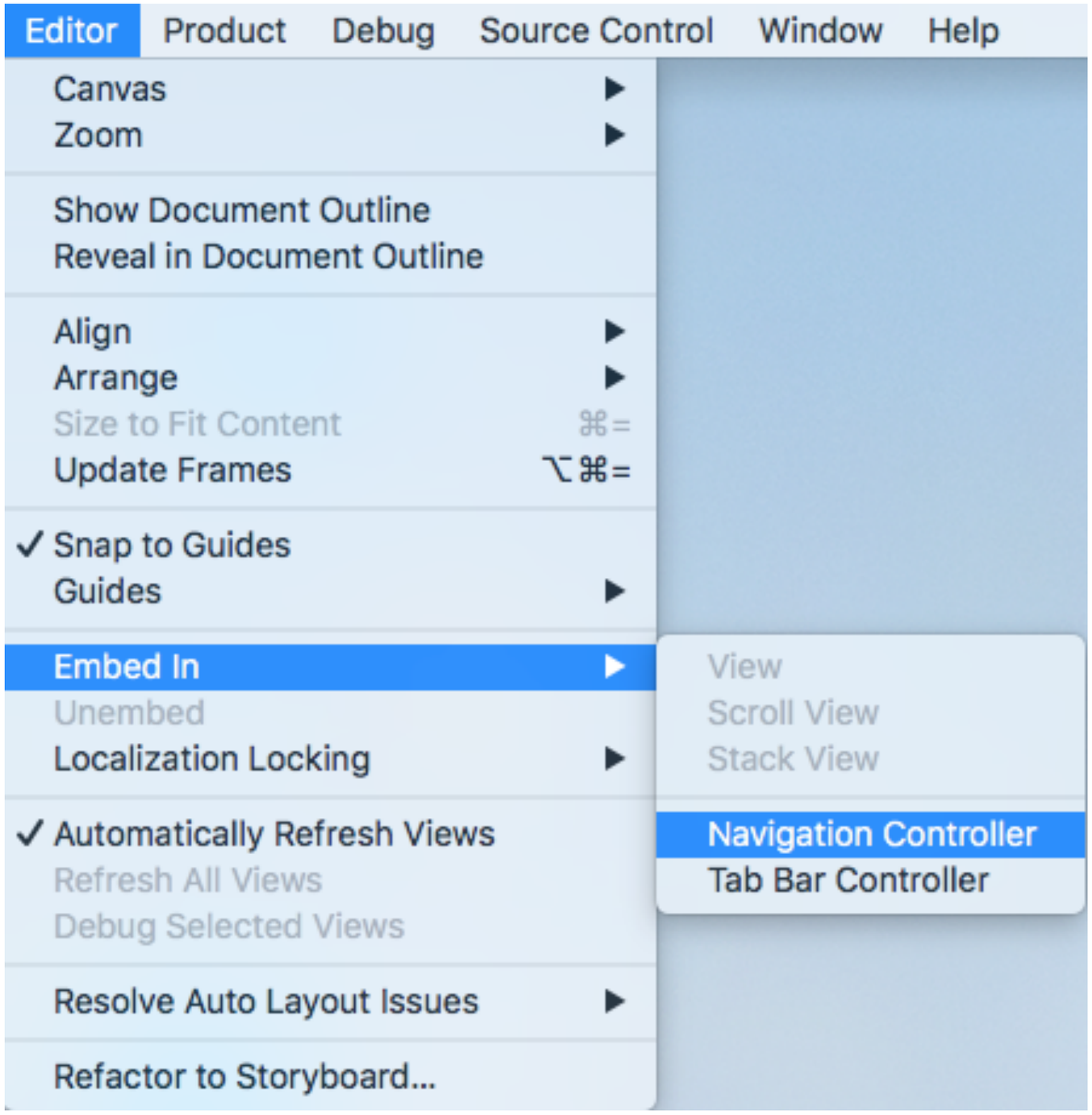
Single View App



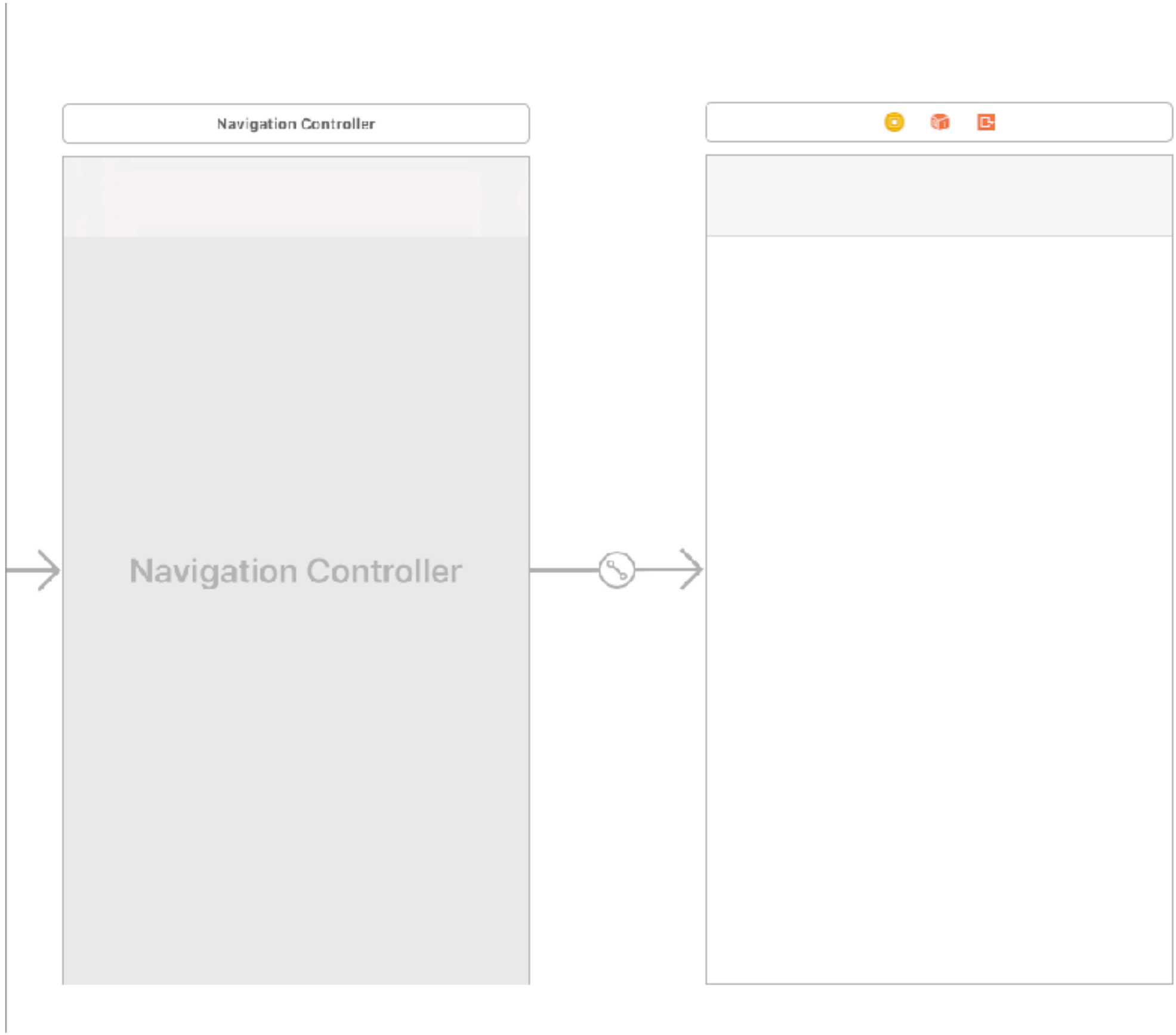
NavigationsController



Einfügen eines NavigationsControllers: Markieren des ViewControllers

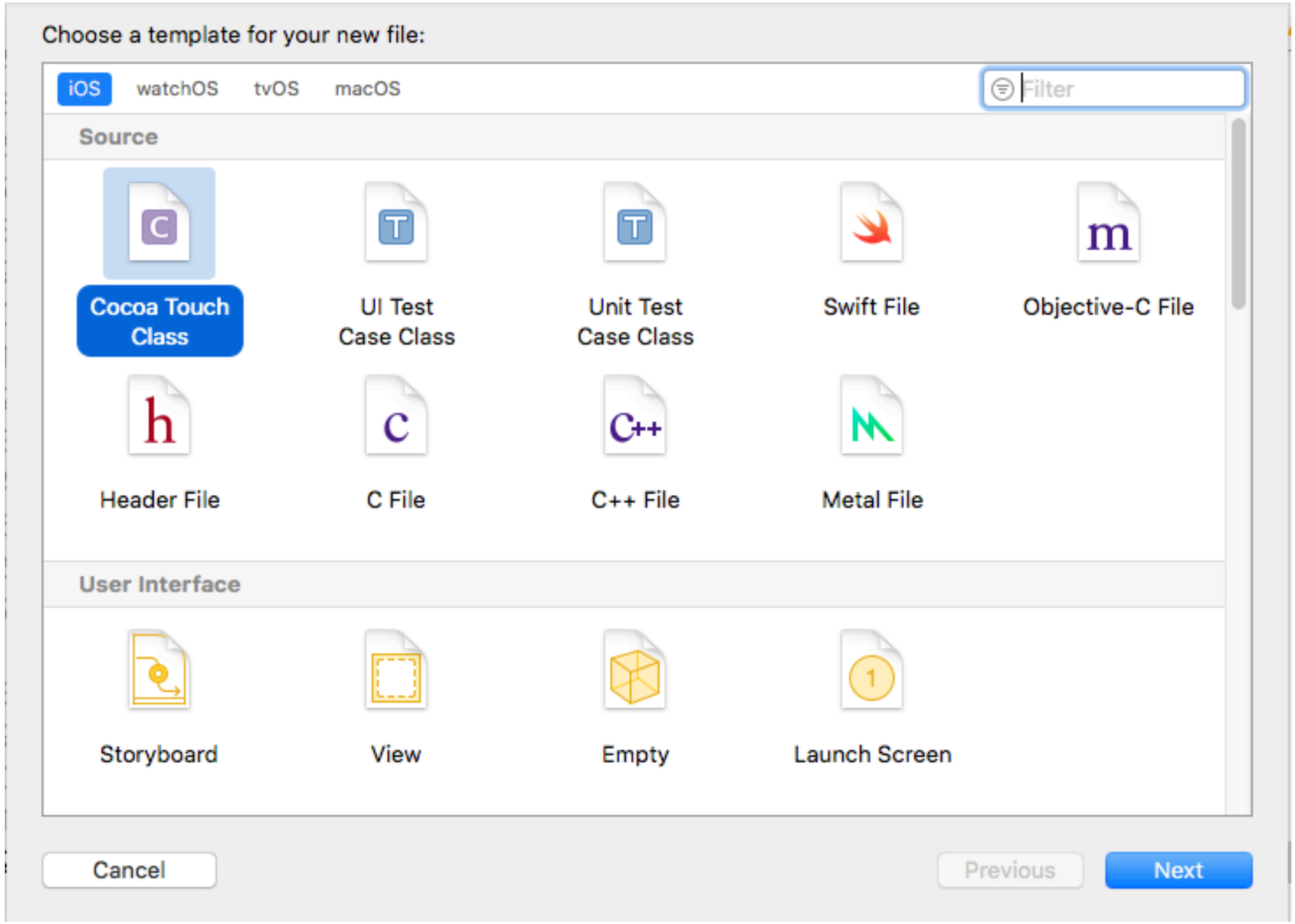


NavigationsController



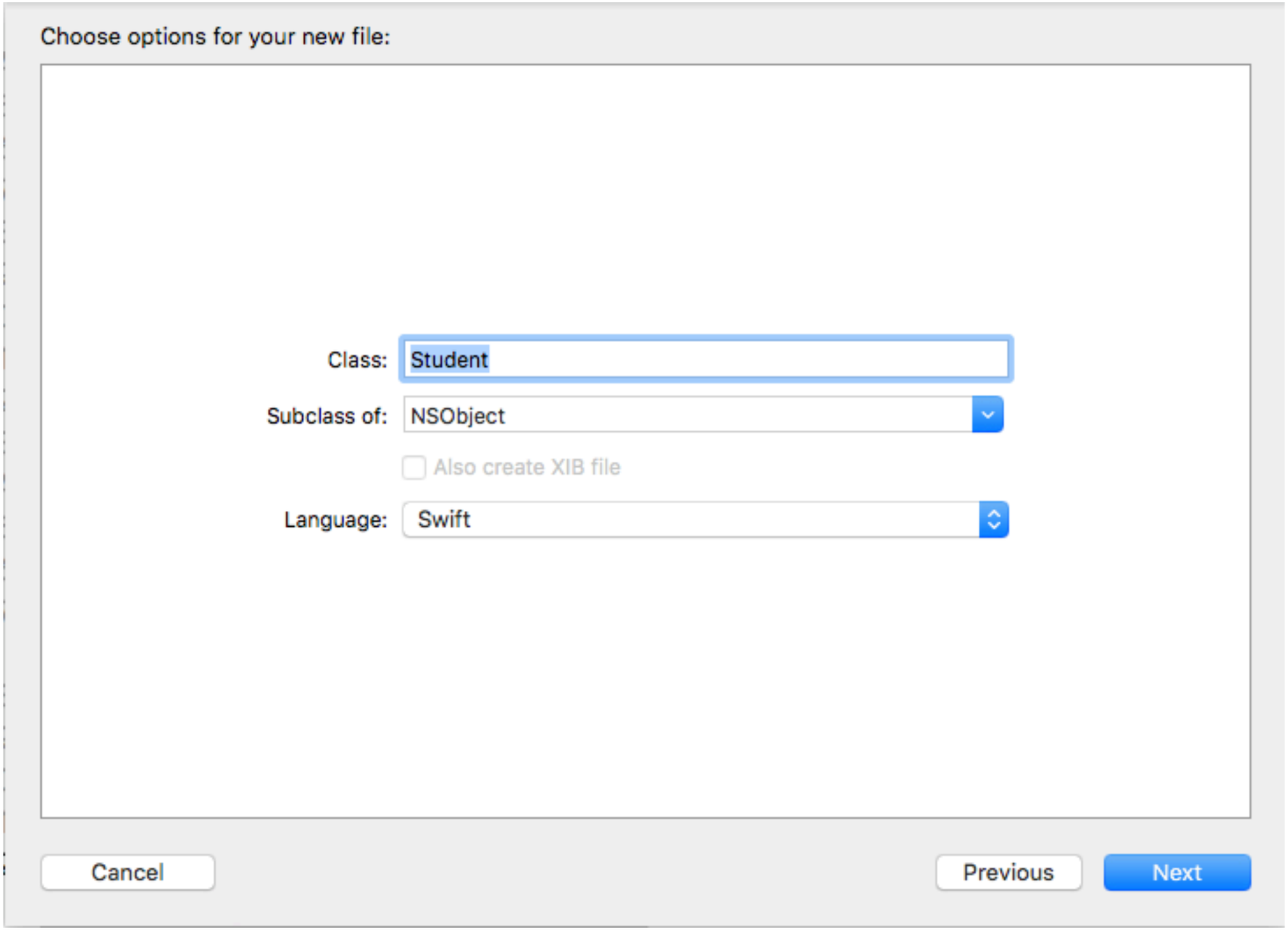
Individuelle Gestaltung der Zellinhalte

Neue Klasse: Student



Individuelle Gestaltung der Zellinhalte

Neue Klasse: Student

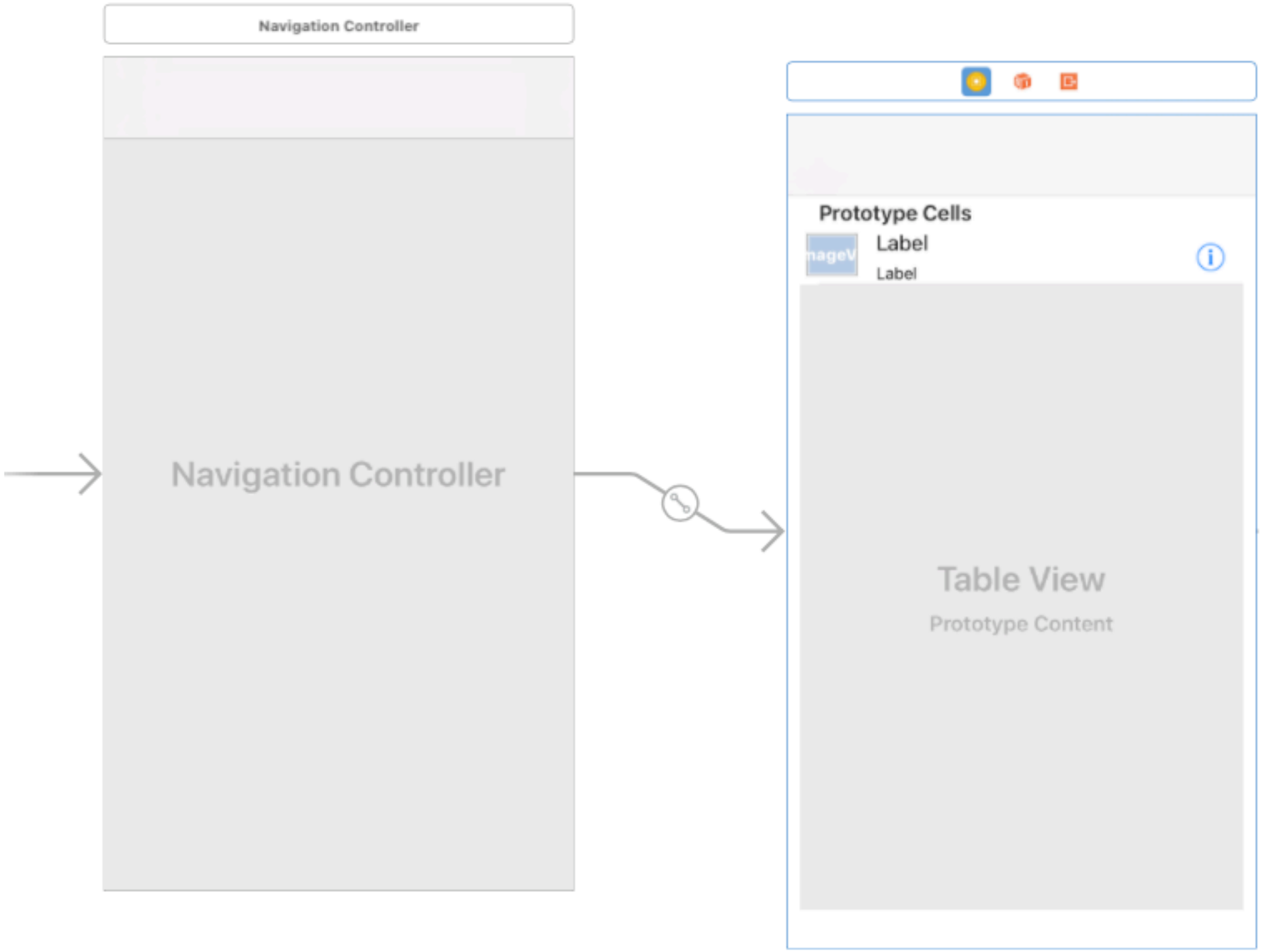


Klasse Student

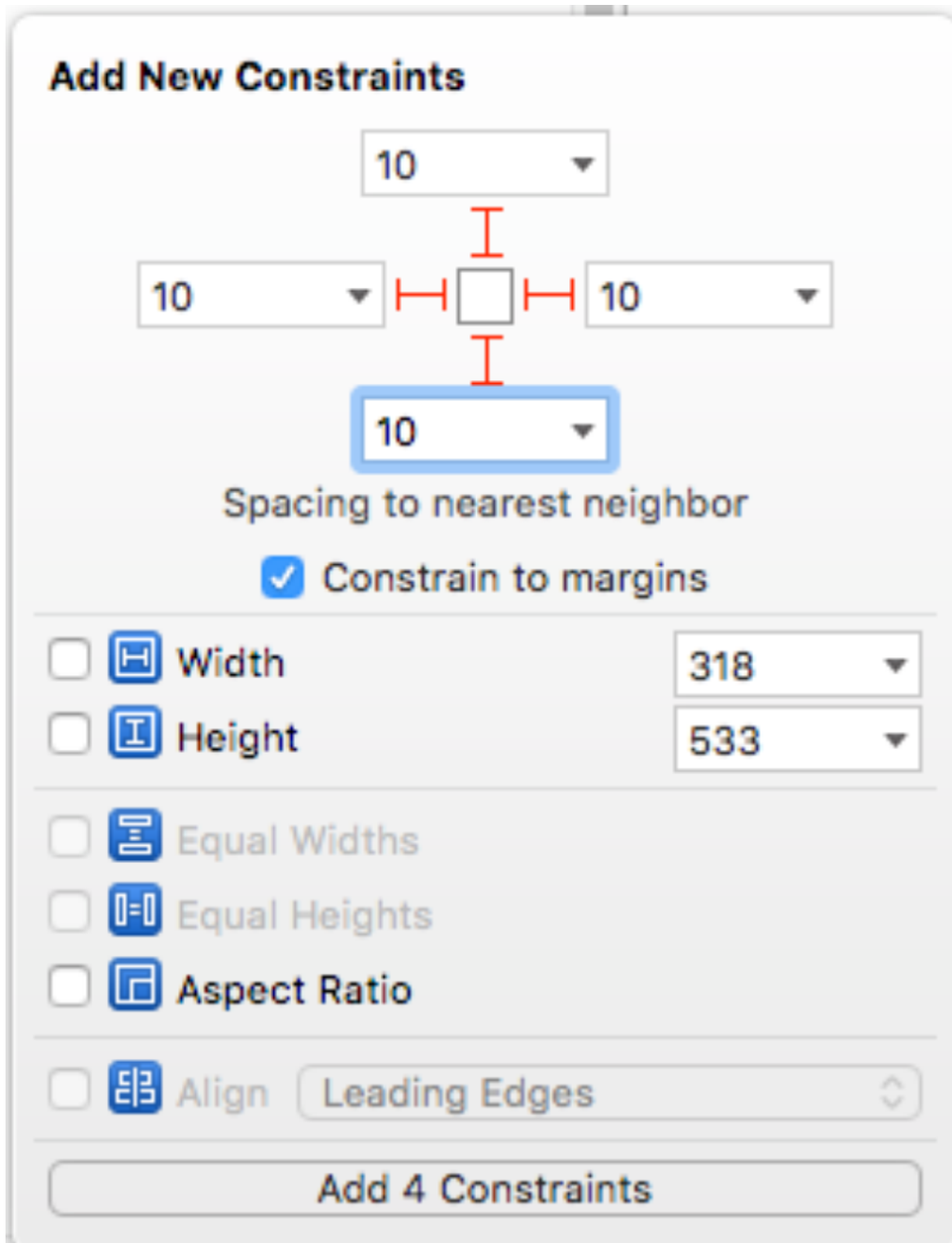
```
class Student: NSObject {
    var name:String=""
    var mnr:Int=0
    var semester:Int=0
    var isMI:Bool=false

    init(_ name:String, _ mnr:Int, _ semester:Int,  _ isMI:Bool){
        self.name=name
        self.mnr=mnr
        self.semester = semester
        self.isMI=isMI
    }
    override var  description :String {
        return "Name: \(name)   Matrnr: \(mnr)  \(isMI)"
    }
}
```

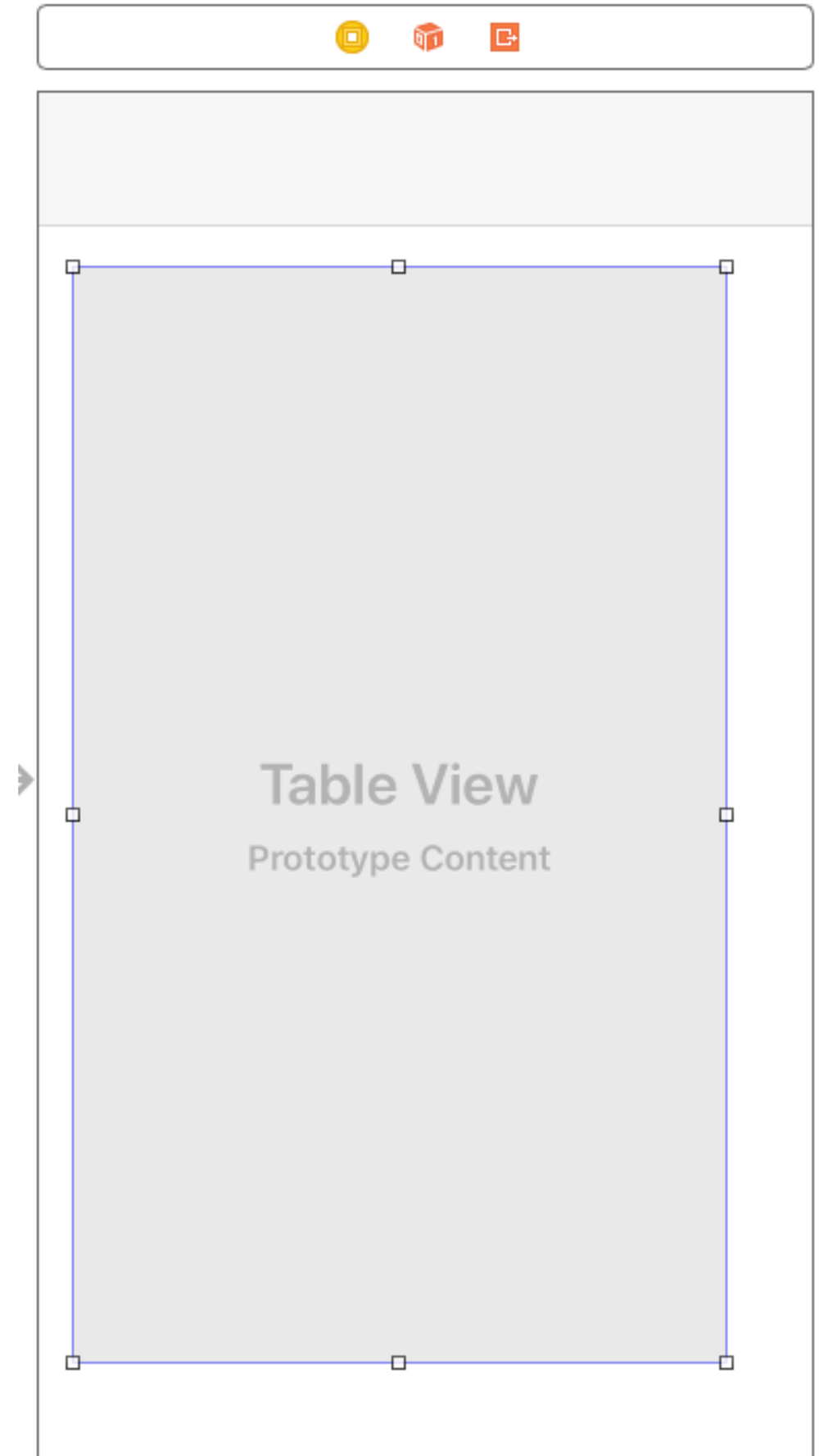
Detail-Darstellung

Einfügen eines Navigationskontrollers:



Einfügen eines TableViews
Outlets in ViewController





Individuelle Gestaltung der Zellinhalte

Protokoll

- UITableViewDelegate, UITableViewDataSource

Methoden

```
// selectedEvent
    func tableView(_: UITableView, didSelectRowAt: IndexPath) {
      let row = (didSelectRowAt as NSIndexPath).row
      labelOutput.text="row: "+String(row)
    }
    func numberOfSectionsInTableView(tableView: UITableView) -> Int {
      return 1
    }
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
      return liste.count
    }
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)
                                                                -> UITableViewCell {
    }
```

Liste der Studierenden

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

    createStudenten()
    tableview.delegate=self
    tableview.dataSource=self
    tableview.tableFooterView = UIView(frame: CGRect.zero)

}
```

Liste der Studierenden

```
private func createStudenten() {
    liste.append(Student("Meier", 25345,1,true))
    liste.append(Student("Müller", 24645,2,true))
    liste.append(Student("Schulze", 23565,5,true))
    liste.append(Student("Kühne", 21645,6,true))
    liste.append(Student("Mattes", 12645,14,true))
    liste.append(Student("Bosch", 21342,7,true))
    liste.append(Student("Hüffner", 23342,4,true))
    liste.append(Student("Brandt", 21342,7,true))
    liste.append(Student("Schmittel", 23342,1,true))
    liste.append(Student("Huffman", 24342,2,false))
    liste.append(Student("Steffan", 23417,2,true))
    liste.append(Student("Fischer", 25342,1,false))
    liste.append(Student("Fleischer", 23342,1,true))
    liste.append(Student("Zwick", 25242,1,false))
    liste.append(Student("Reinhard", 23542,1,true))
}
```

3) Cell-Bilder

http://miwilhelm.de/vorlesungen/5-semester/ios/index.html

TableView3-Projekt:

TableView3.zip

Transparente Bilder:

symboletransparent.zip

semesterbilder.zip

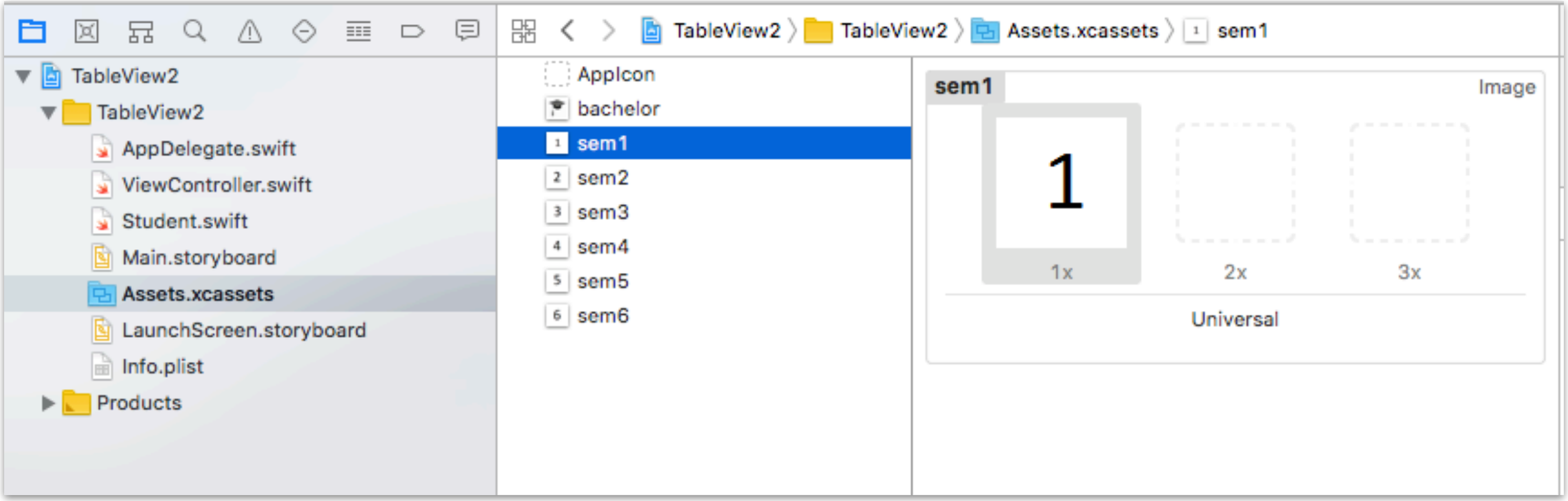
operationsbilder.zip

Overview:

iOSSamples.pdf

3) Cell-Bilder

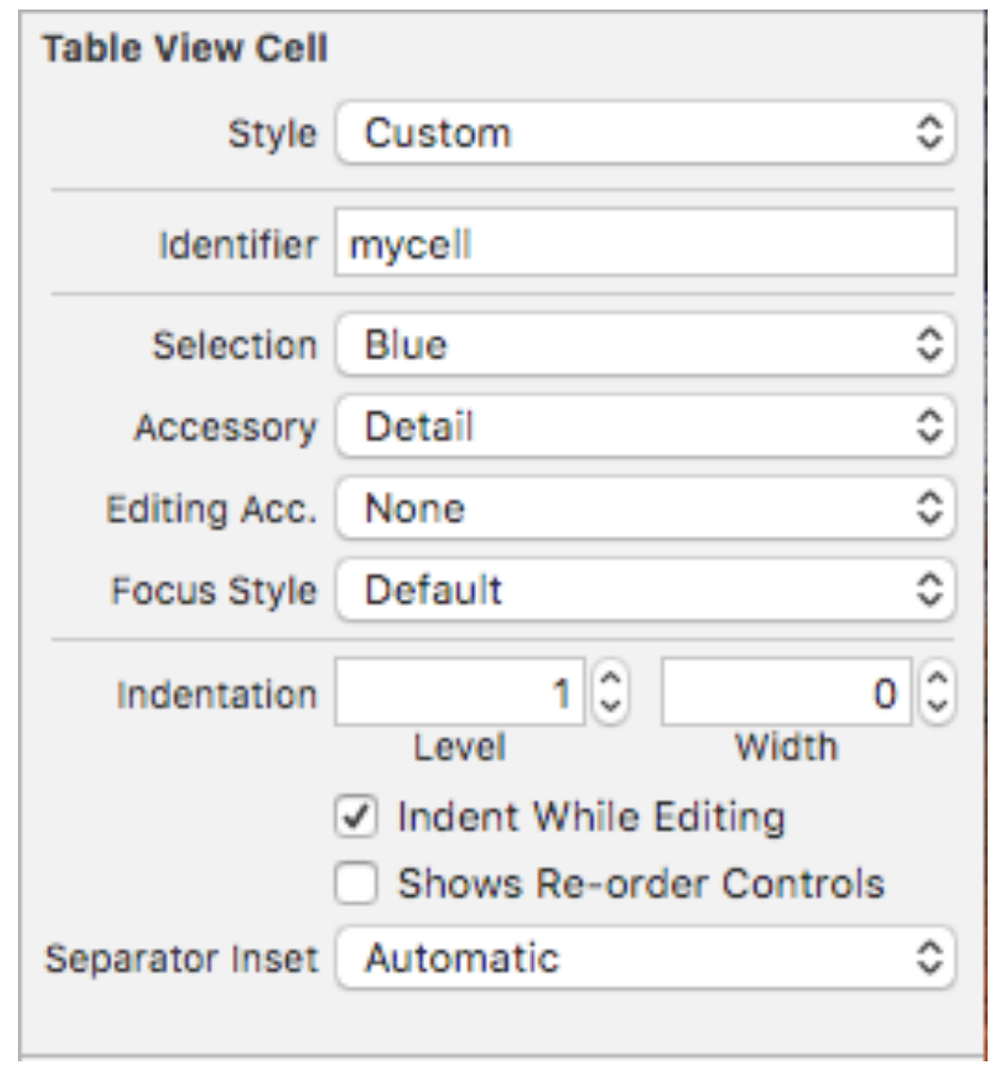
- Safari starten
- Adresse eingeben
- Zip-Dabei runterladen
- entzippen
- Xcode-Projektstruktur: Assets.xcassets
- per Drag & Drop ins Projekt ziehen



Die „cell“ der TableView

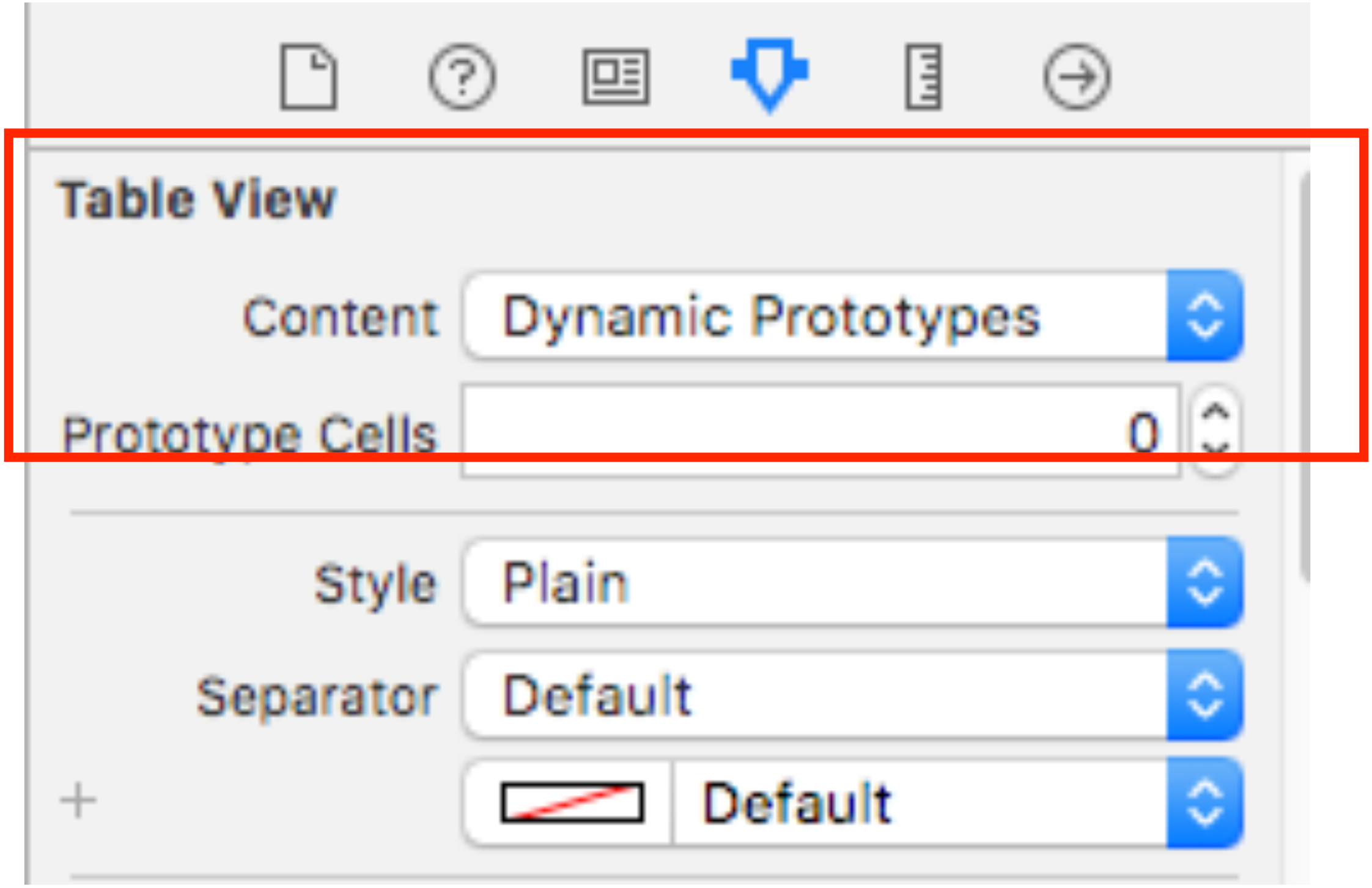
Ablauf:

- Main.storyboard
- Anblicken der Table View
- Wechseln in das Property-Fenster



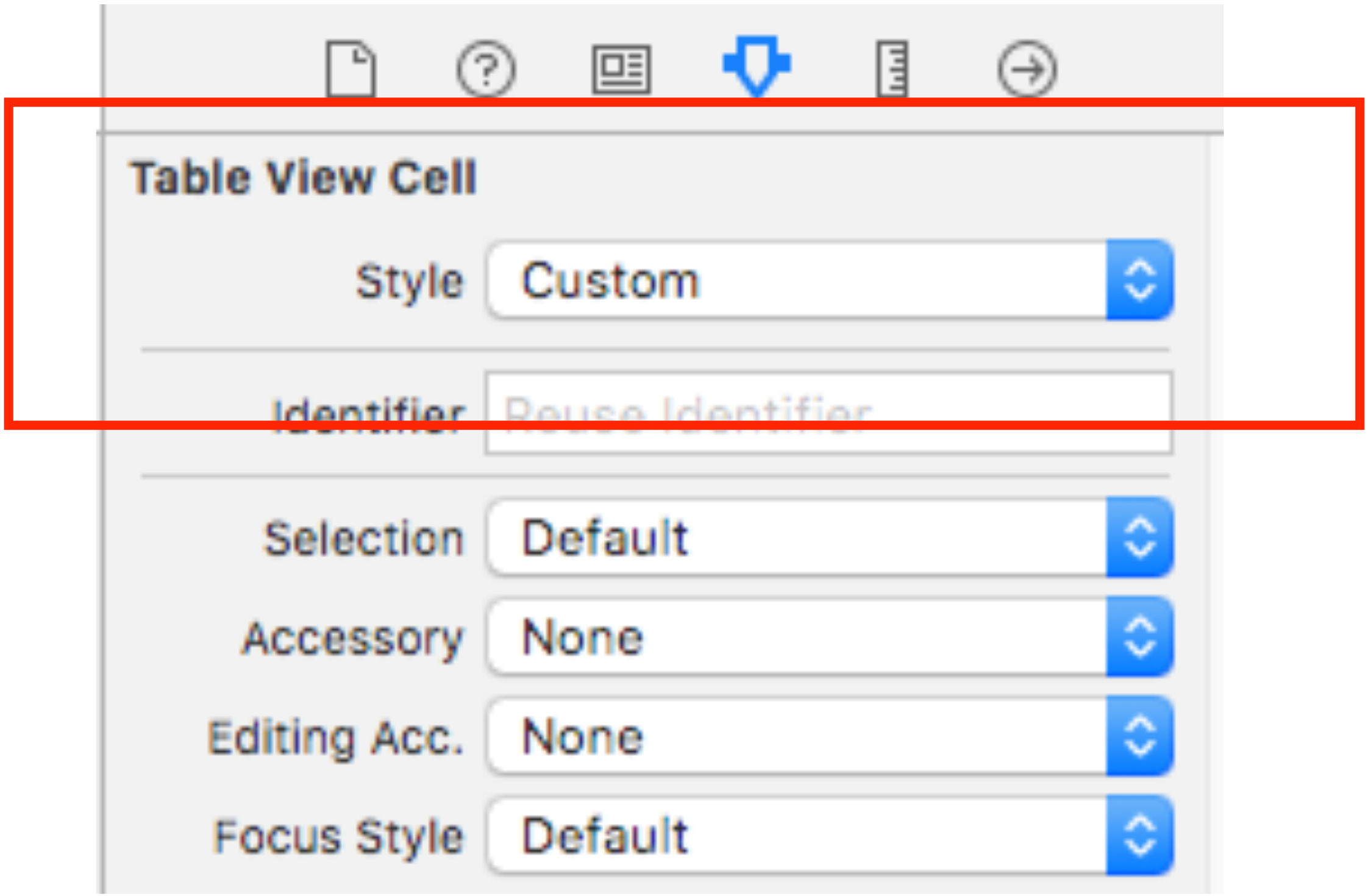
Individuelle Gestaltung der Zellinhalte

Definieren eines Prototypes



Individuelle Gestaltung der Zellinhalte

Definieren eines Prototypes



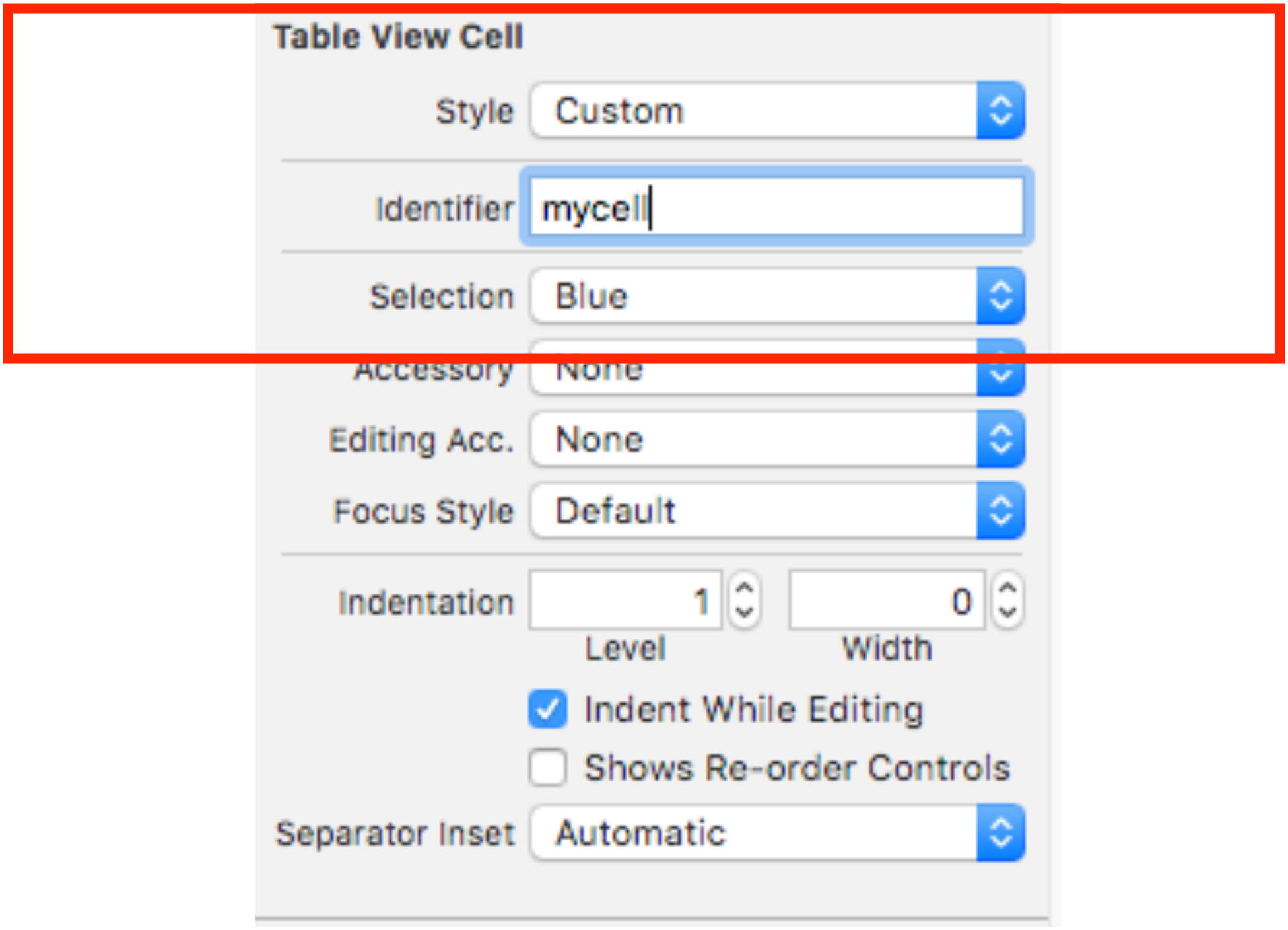
Individuelle Gestaltung der Zellinhalte

- **Eintragen einer Table View Cell**
- **Definieren eines Prototypes**

In diesen Bereich werden die grafischen Elemente eingetragen!

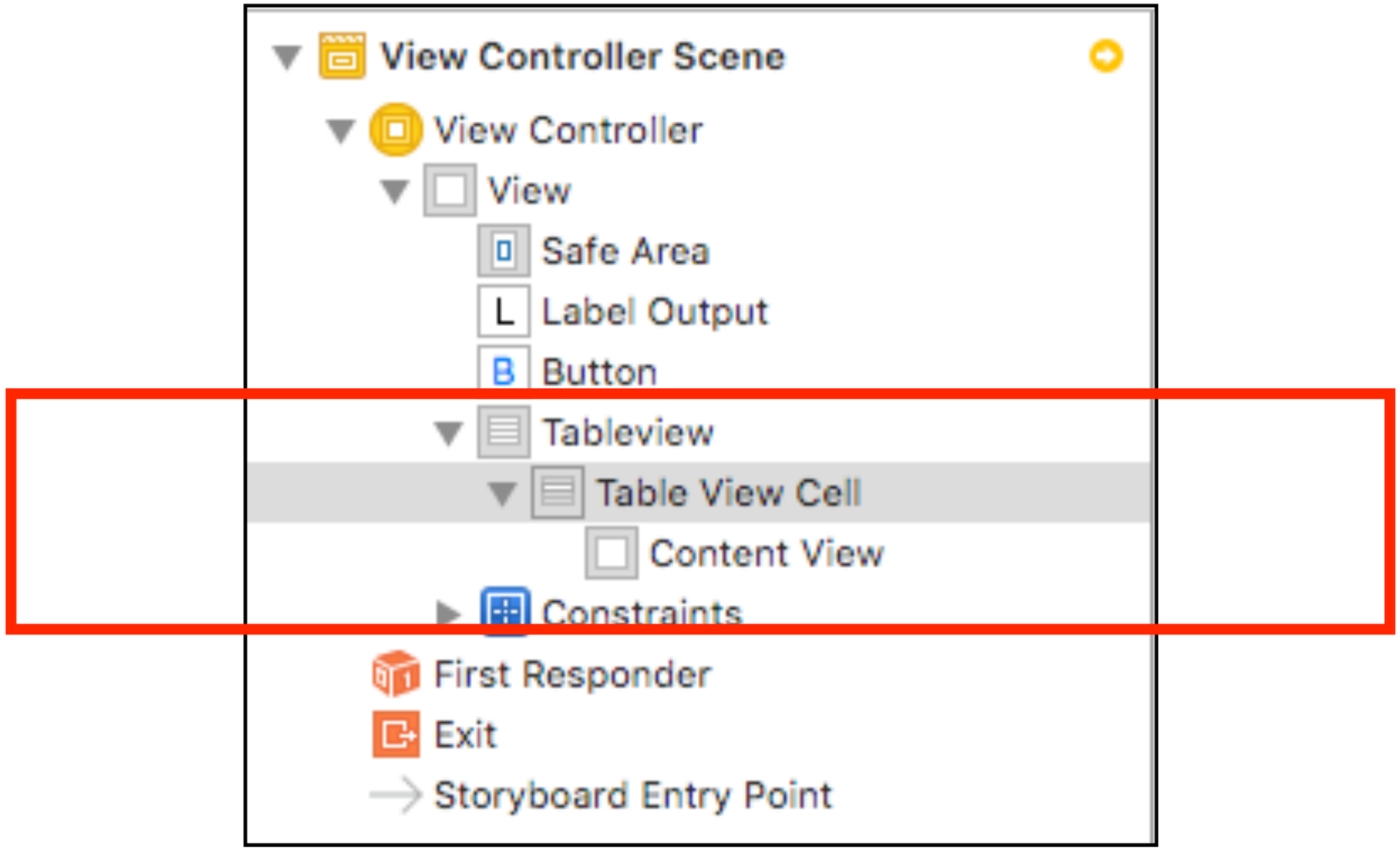
Individuelle Gestaltung der Zellinhalte

Definieren eines Prototypes: Name für später



Individuelle Gestaltung der Zellinhalte

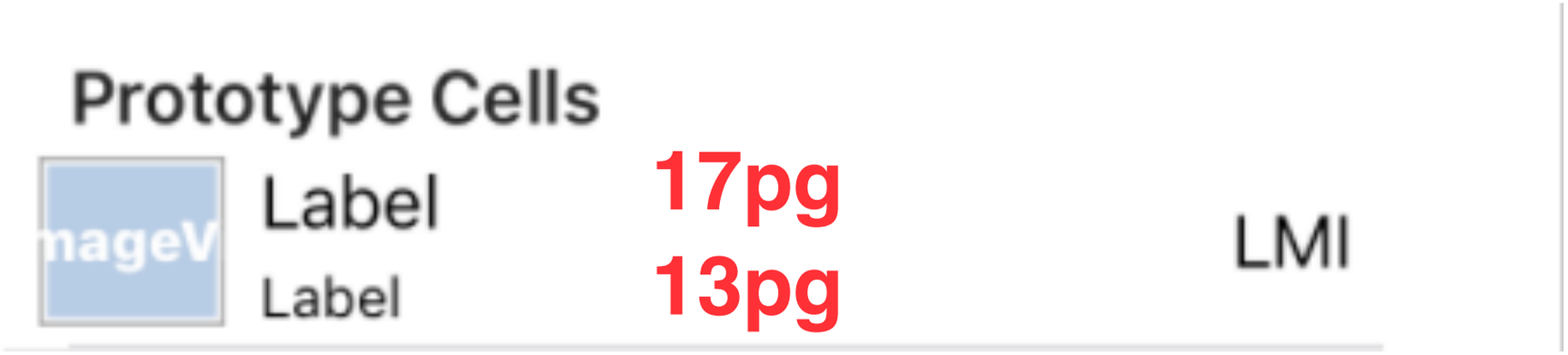
Definieren eines Prototypes: Projektbaum

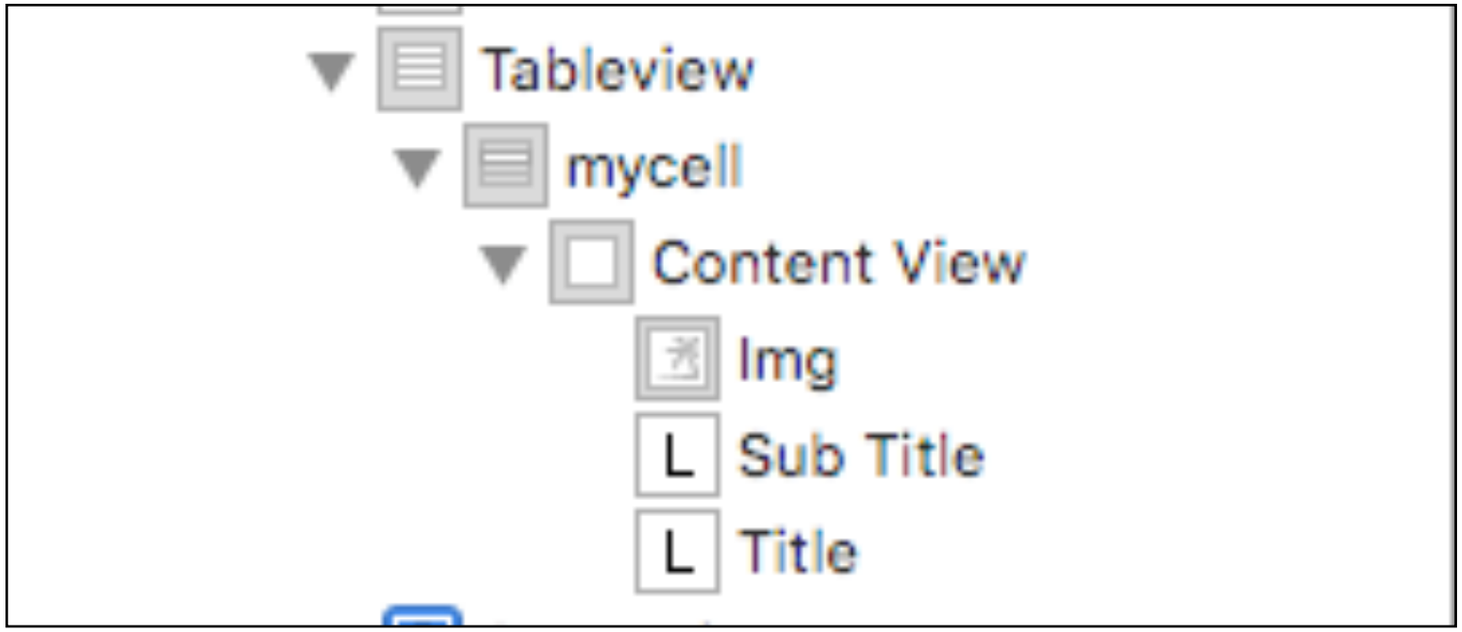


Individuelle Gestaltung der Zellinhalte

Definieren eines Prototypes: Projektbaum

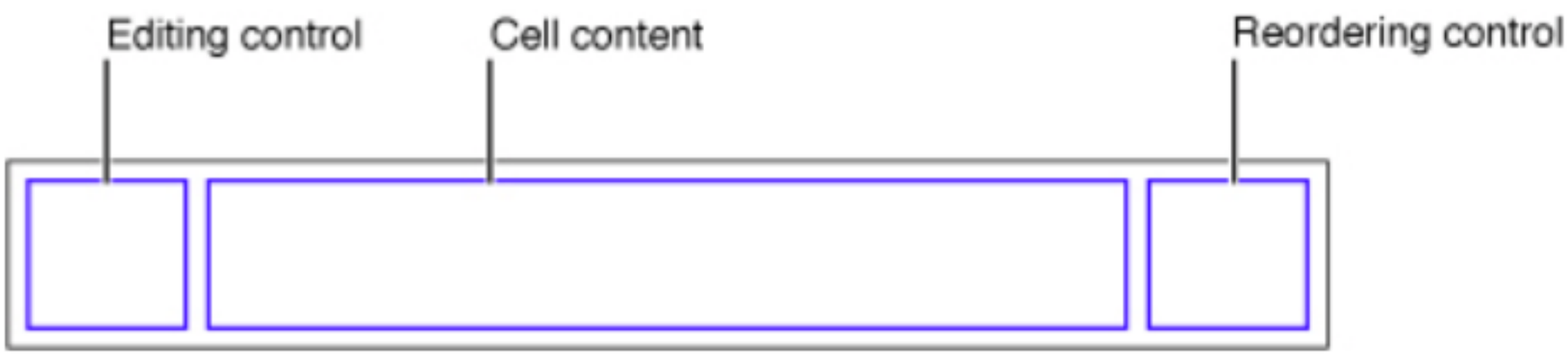
- Drag & Drop der Elemente
- Positionieren mit der Maus



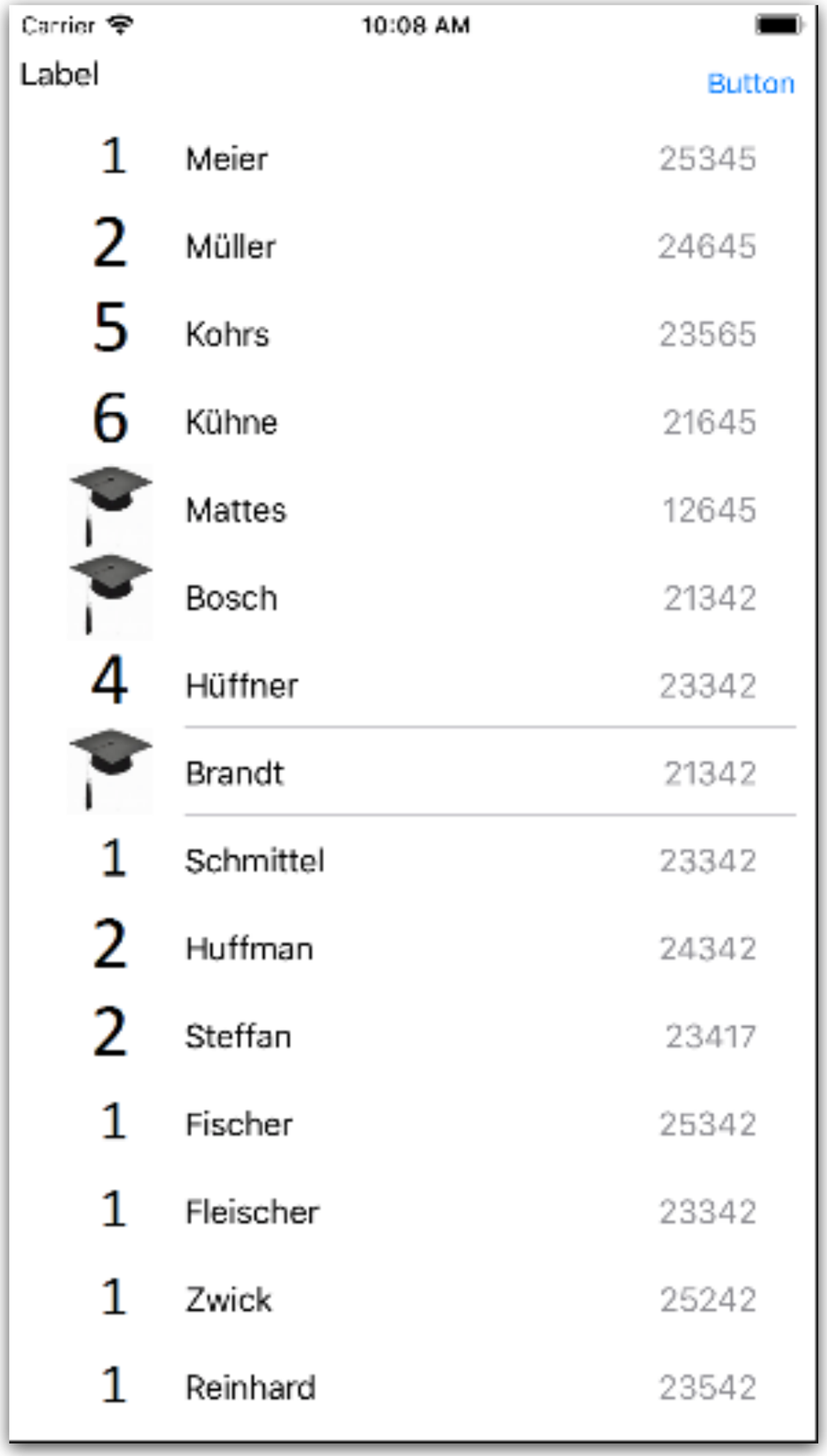


Es fehlt aber die Möglichkeit, die einzelnen Elemente anzusprechen (.default, .value1 etc.)

Cell content
Accessory view


Editing control
Cell content
Reordering control

UITableViewCellStyle.value1:
Bild links, Titel in der Mitte, rechts Zusatzinformation



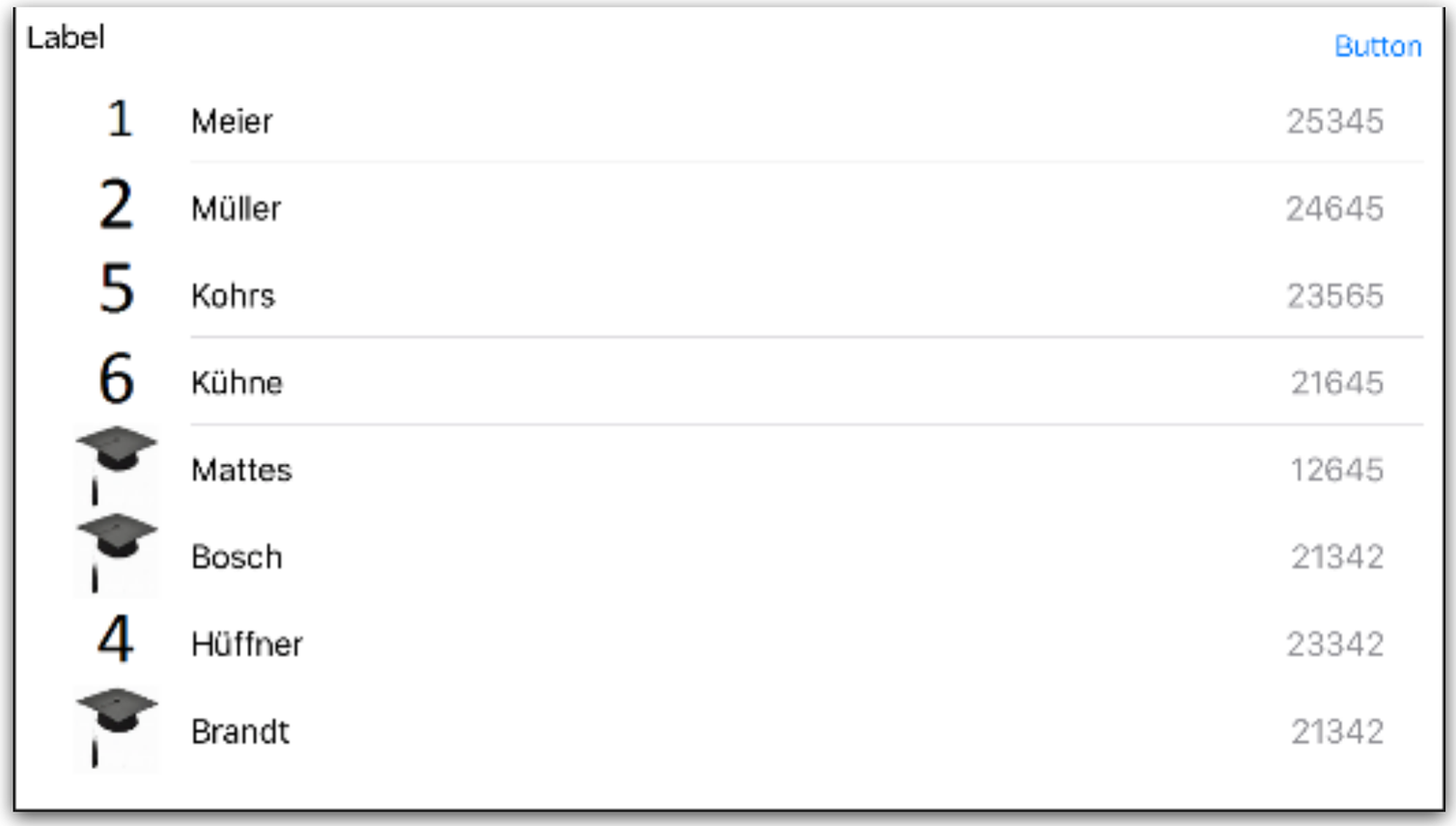
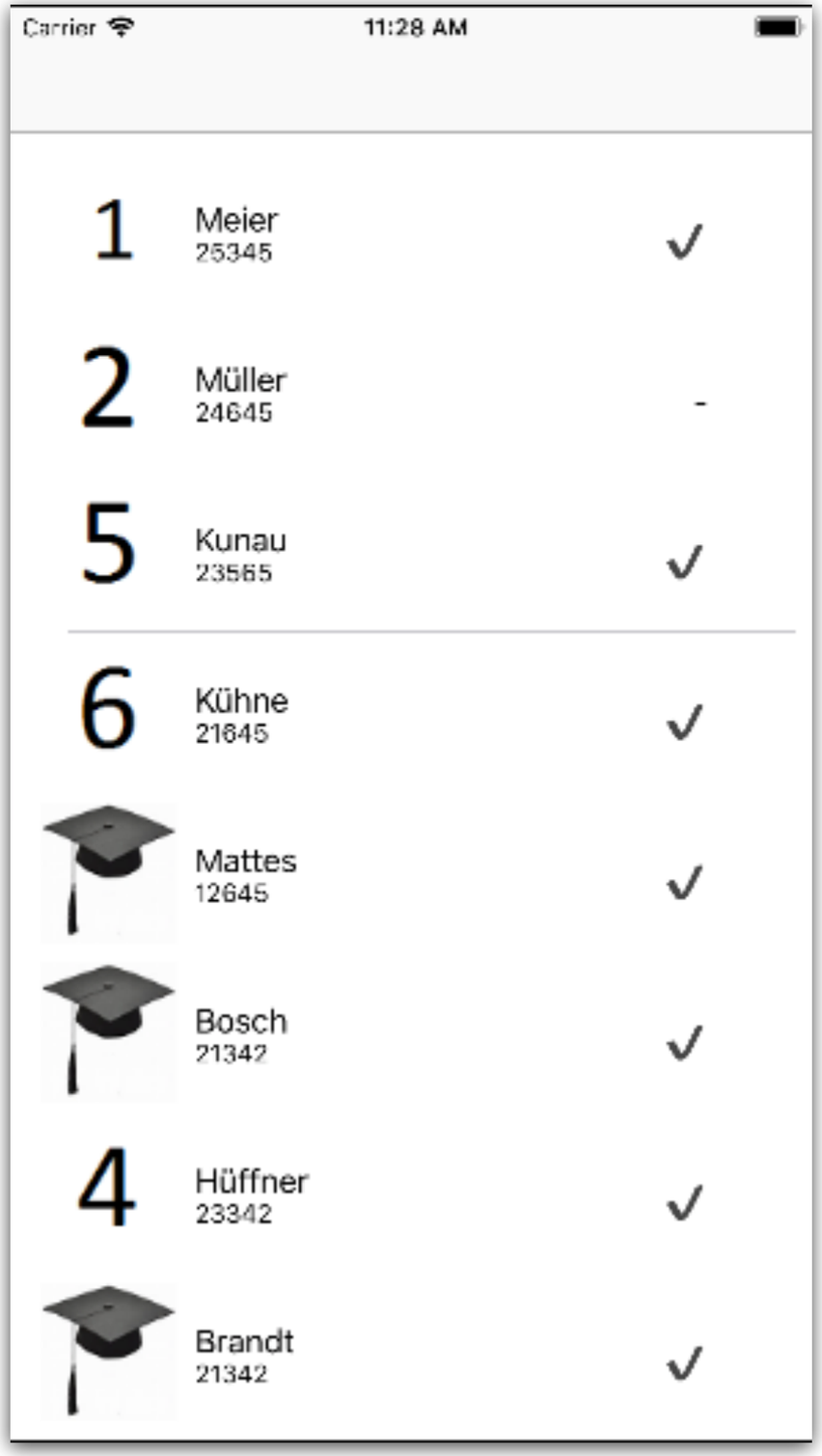


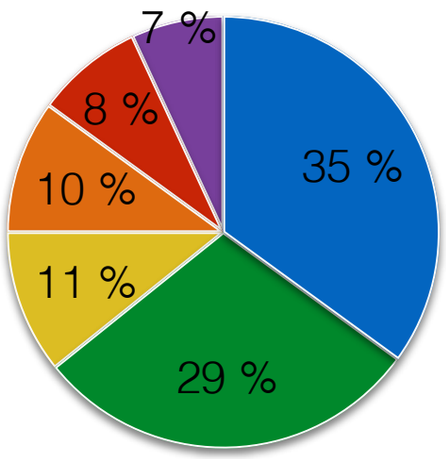
Figure 6-11. A quick demo of the stack view's properties

- The *alignment* option controls how the arranged views are aligned. For example, if it is set to Leading, the stack view aligns the leading edge (i.e. left) of its arranged views along its leading edge.
- The *distribution* option defines the size and position of the arranged views. By default, it is set to *Fill*. In this case, the stack view tries its best to fit all subview in its available space. If it is set to *Fill Equally*, the vertical stack view distributes both labels equally so that they are all the same size along the vertical axis.


Carrier
11:28 AM
1
Meier
25345
2
Müller
24645
-
5
Kunau
23565
6
Kühne
21645
Mattes
12645
Bosch
21342
4
Hüffner
23342
Brandt
21342

1
Meier
25345
2
Müller
24645
-
5
Kunau
23565
6
Kühne
21645

cell-Layout



title

subtitle

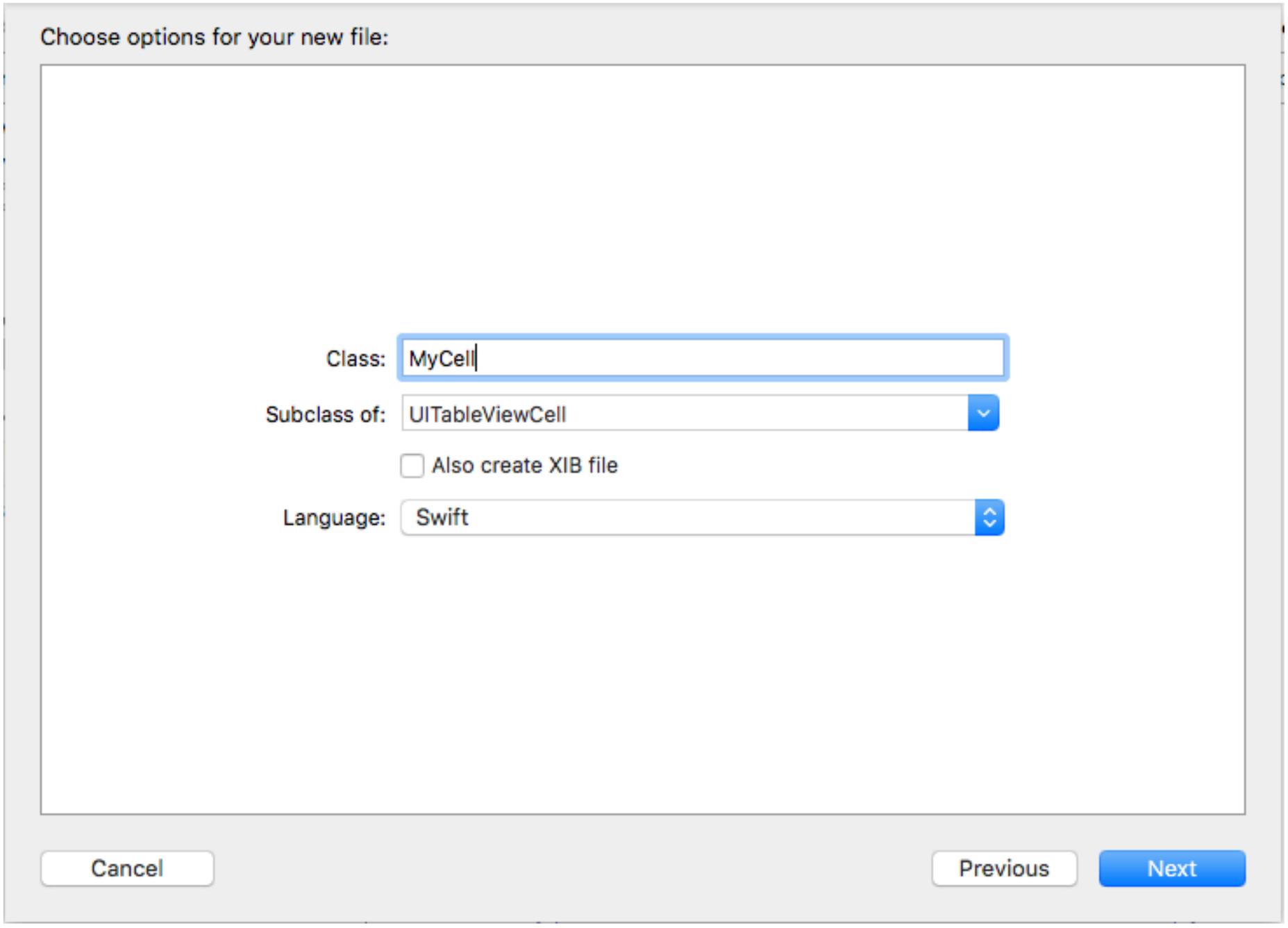
mi

- Image: Top, **Left**, Bottom
- title: Top, **Left**, Bottom
- subtitle: Top, **Left**, Bottom

- mi: Top, **Right**, Bottom

Individuelle Gestaltung der Zellinhalte

Anlegen einer Klasse: Cmd/N: (UITableViewCell)

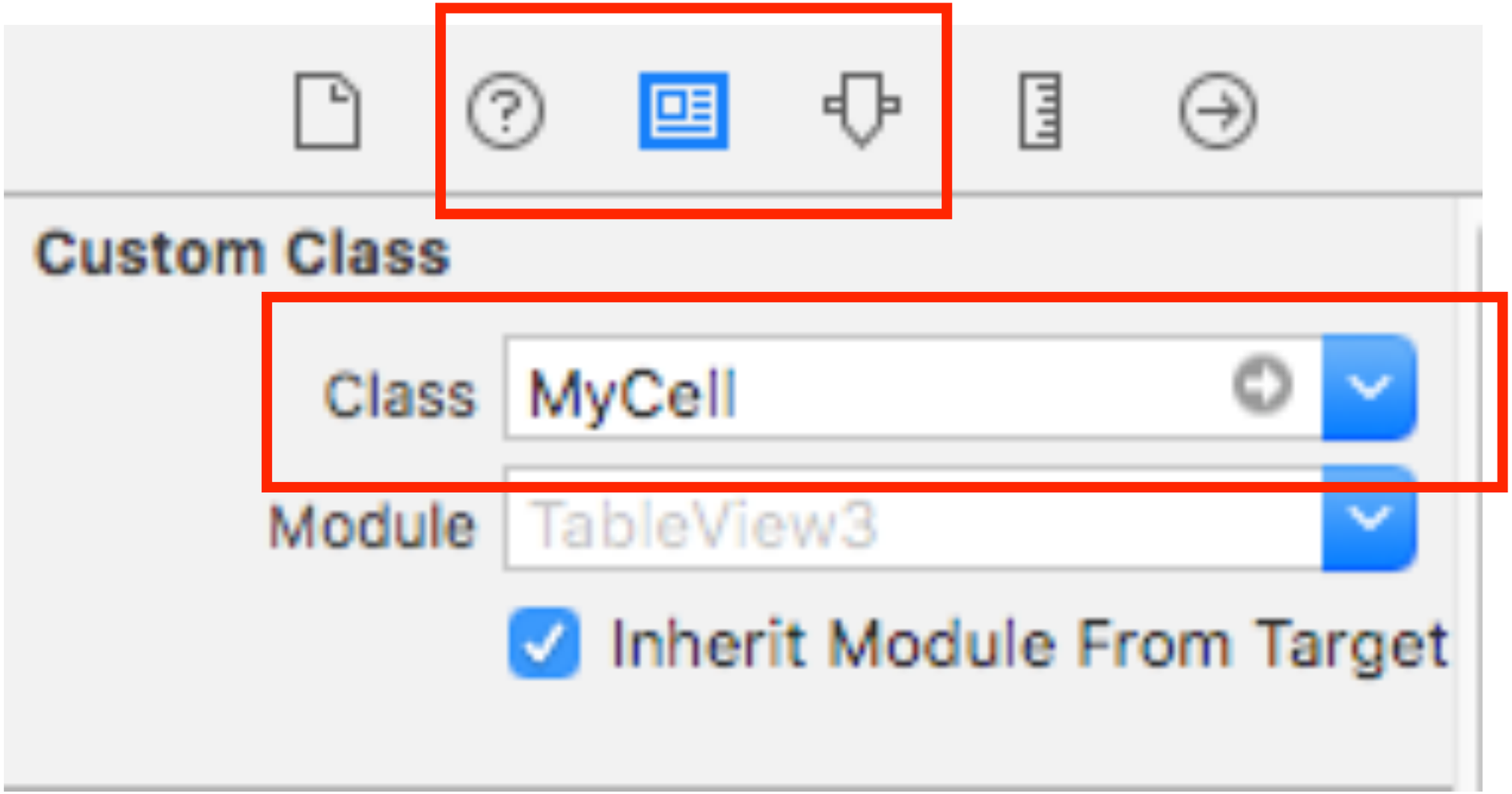


```
class MyCell: UITableViewCell {

    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
    }

    override func setSelected(_ selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }

}
```

Individuelle Gestaltung der Zellinhalte

Zuweisen der Klasse zur Prototyp-Zelle

- MyCell im Projektbaum anklicken



Individuelle Gestaltung der Zellinhalte

Erstellen der Outlets in der Klasse MyCell mittels Drag & Drop

- @IBOutlet var img: UIImageView!
- @IBOutlet var title: UILabel!
- @IBOutlet var subtitle: UILabel!

Individuelle Gestaltung der Zellinhalte

Eintragen des Quellcodes in „drawCell“

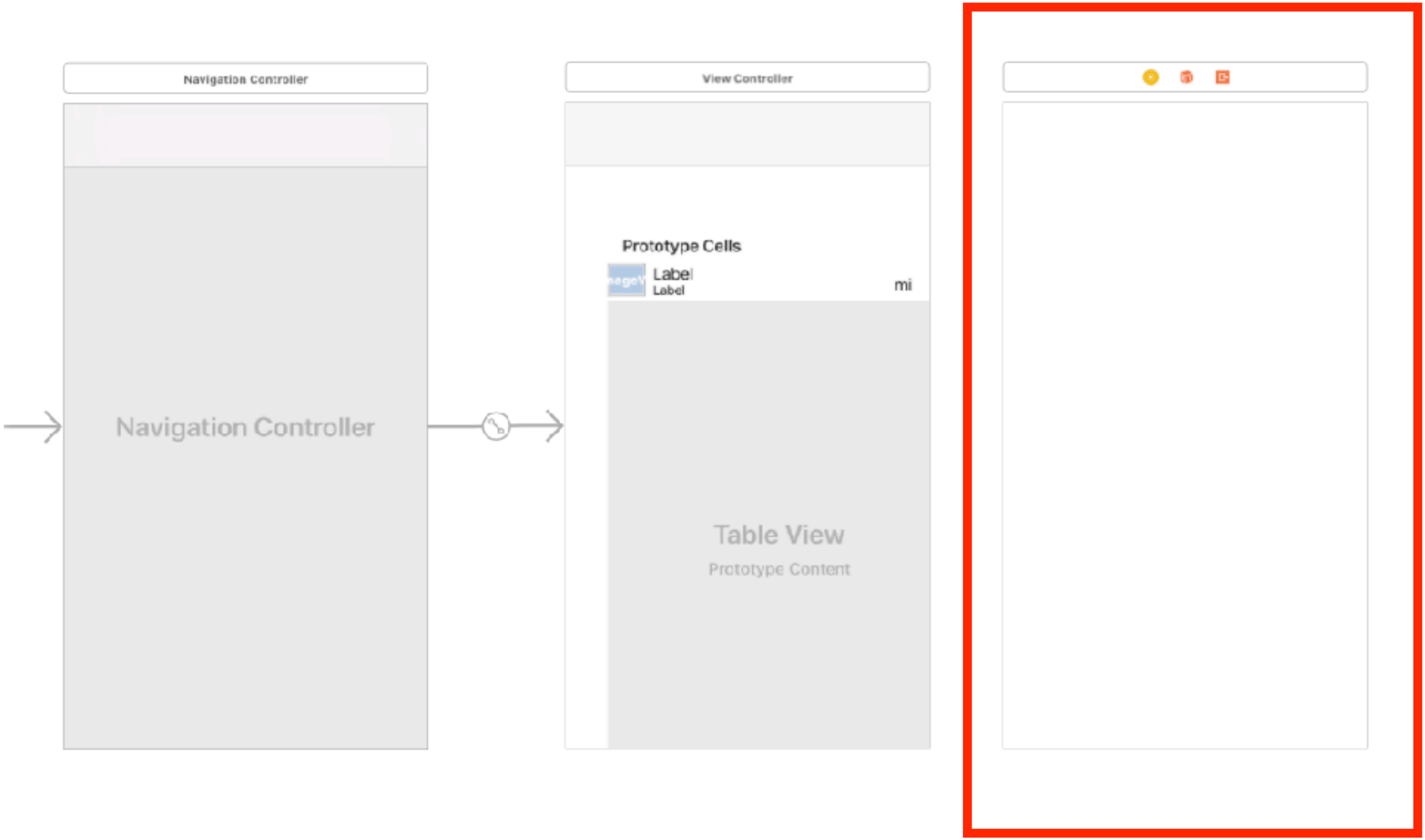
```
func tableView(_ tableView: UITableView, cellForRowAt indexPath:
IndexPath) -> UITableViewCell {
  let cell = tableView.dequeueReusableCell(withIdentifier: "mycell", for:
indexPath) as! MyCell
      let row = (indexPath as NSIndexPath).row
      let student:Student = liste[ row ]
      cell.title!.text = student.name
      cell.subTitle!.text=String(student.mnr)
      switch student.semester {
      case 1:
          cell.img!.image = UIImage(named: "sem1")
     default:
          cell.img!.image = UIImage(named: "bachelor")
      }  // switch

      return cell
}
```

Detaildarstellung

leere Folie

Detail-Darstellung

Einfügen eines neuen Detail-ViewController

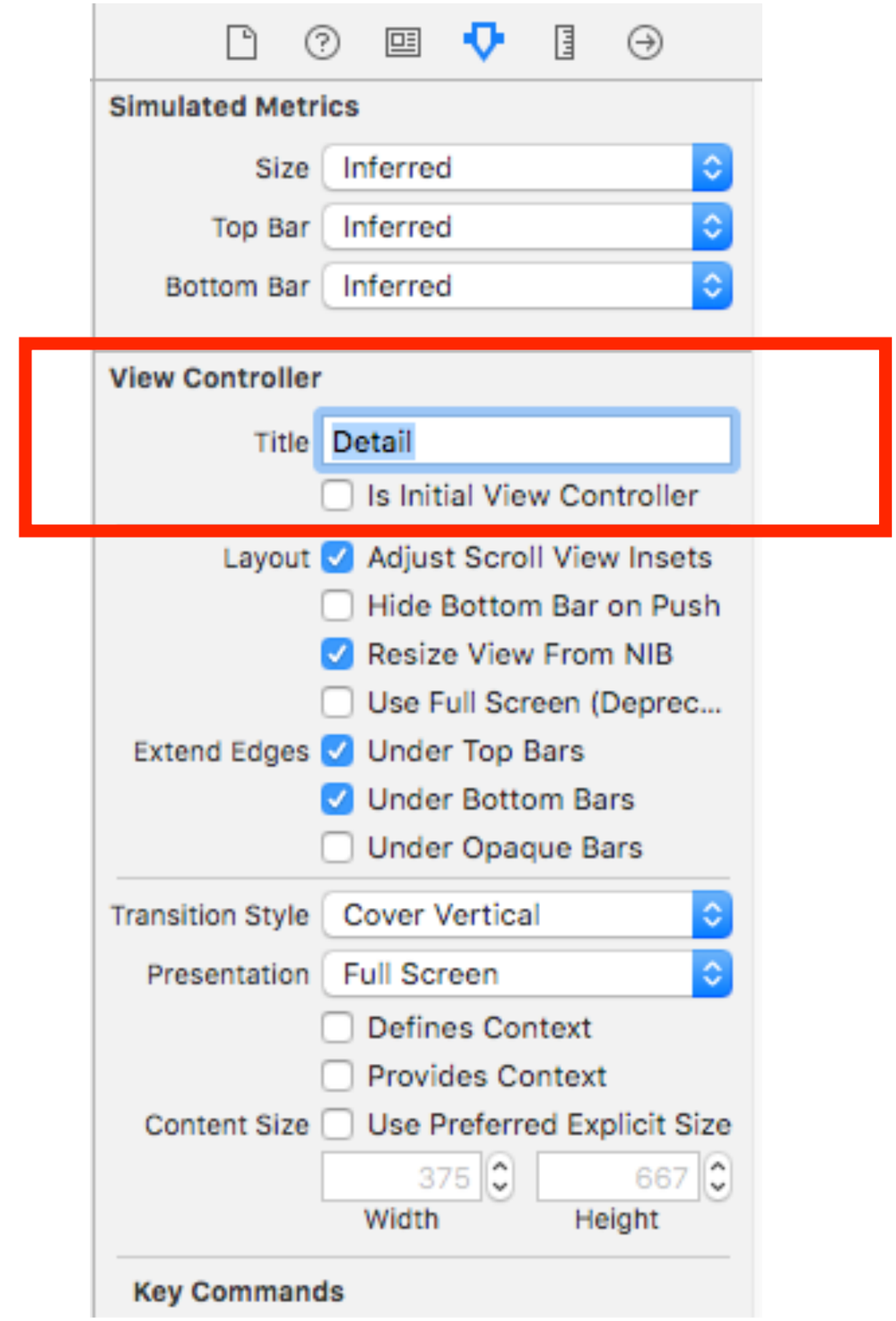


Detail-Darstellung

Einen Namen für den neuen Detail-ViewController vergeben

1. Anklicken des linken Kreises

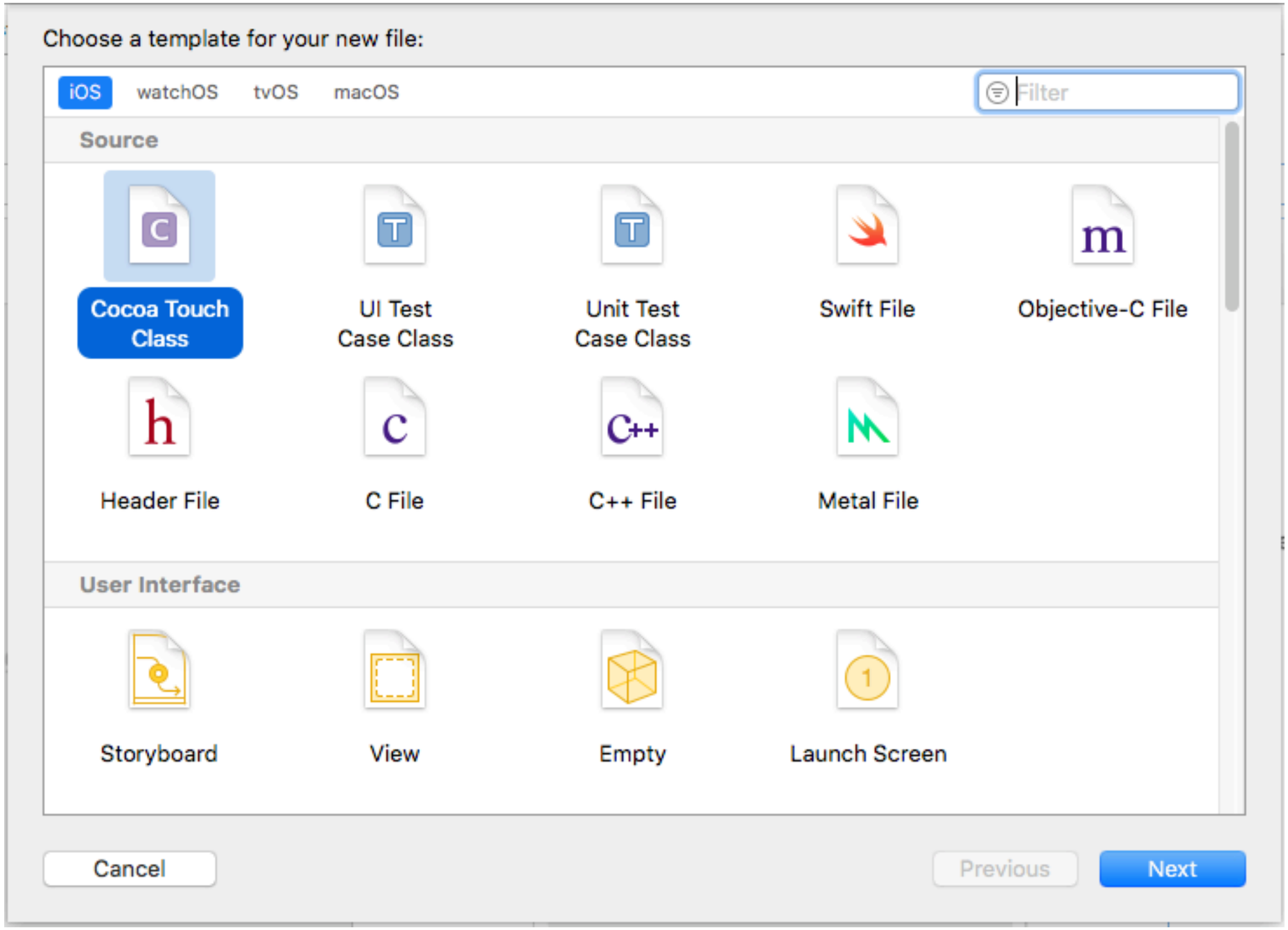
2. Anklicken des Symbols



Detail-Darstellung

Eintragen einer Klasse für den neuen ViewController

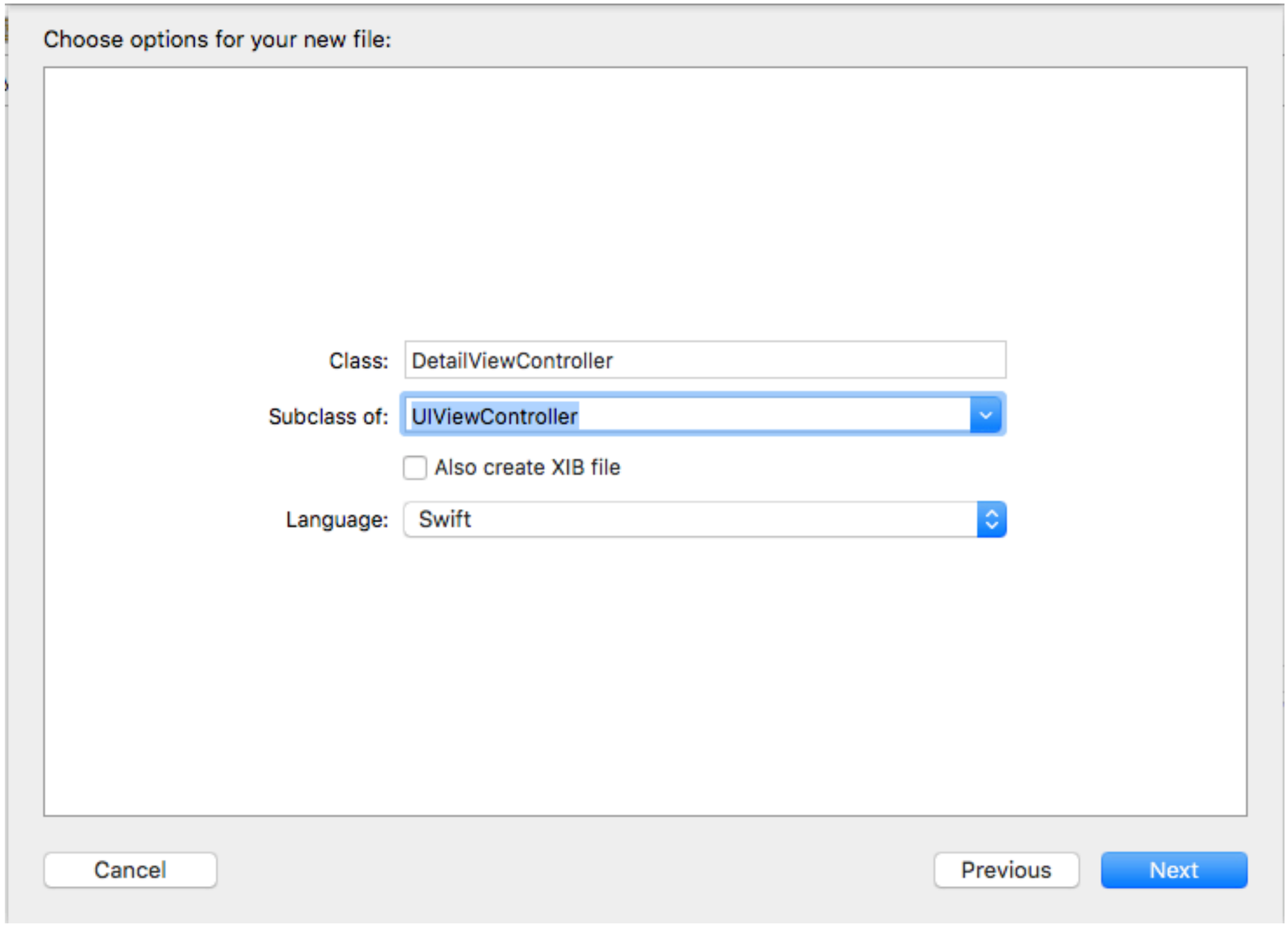
Cmd+N / Cocoa-Klasse

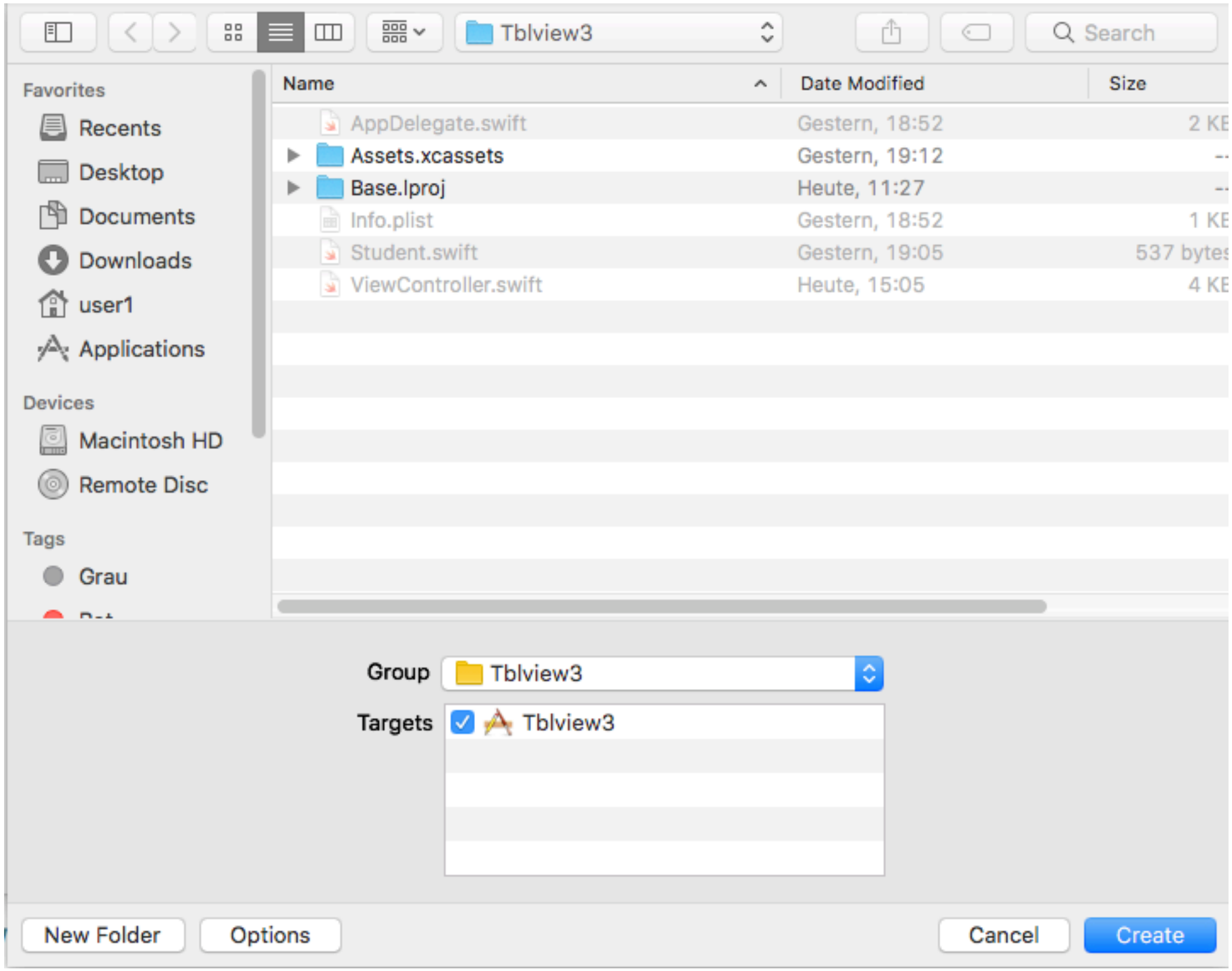


Detail-Darstellung

Eintragen einer Klasse für den neuen ViewController

Cmd+N / Cocoa-Klasse




Tblview3
Search
Favorites
Recents
Desktop
Documents
Downloads
user1
Applications
Devices
Macintosh HD
Remote Disc
Tags
Grau
Name
Date Modified
Size
AppDelegate.swift
Gestern, 18:52
2 KE
Assets.xcassets
Gestern, 19:12
Base.lproj
Heute, 11:27
Info.plist
Gestern, 18:52
1 KE
Student.swift
Gestern, 19:05
537 bytes
ViewController.swift
Heute, 15:05
4 KE
Group
Tblview3
Targets
Tblview3
New Folder
Options
Cancel
Create

Übergabe des aktuellen Studenten an den neuen ViewController

im HauptViewController:

```
override func prepare(for segue: UIStoryboardSegue,
sender: Any?) {
        if let dest = segue.destination as? DetailViewController,
      let cell = sender as? MyCell,
        let indexPath = tableview.indexPath(for: cell) {
        dest.studentData = liste[ indexPath.row ]
    }
}
```

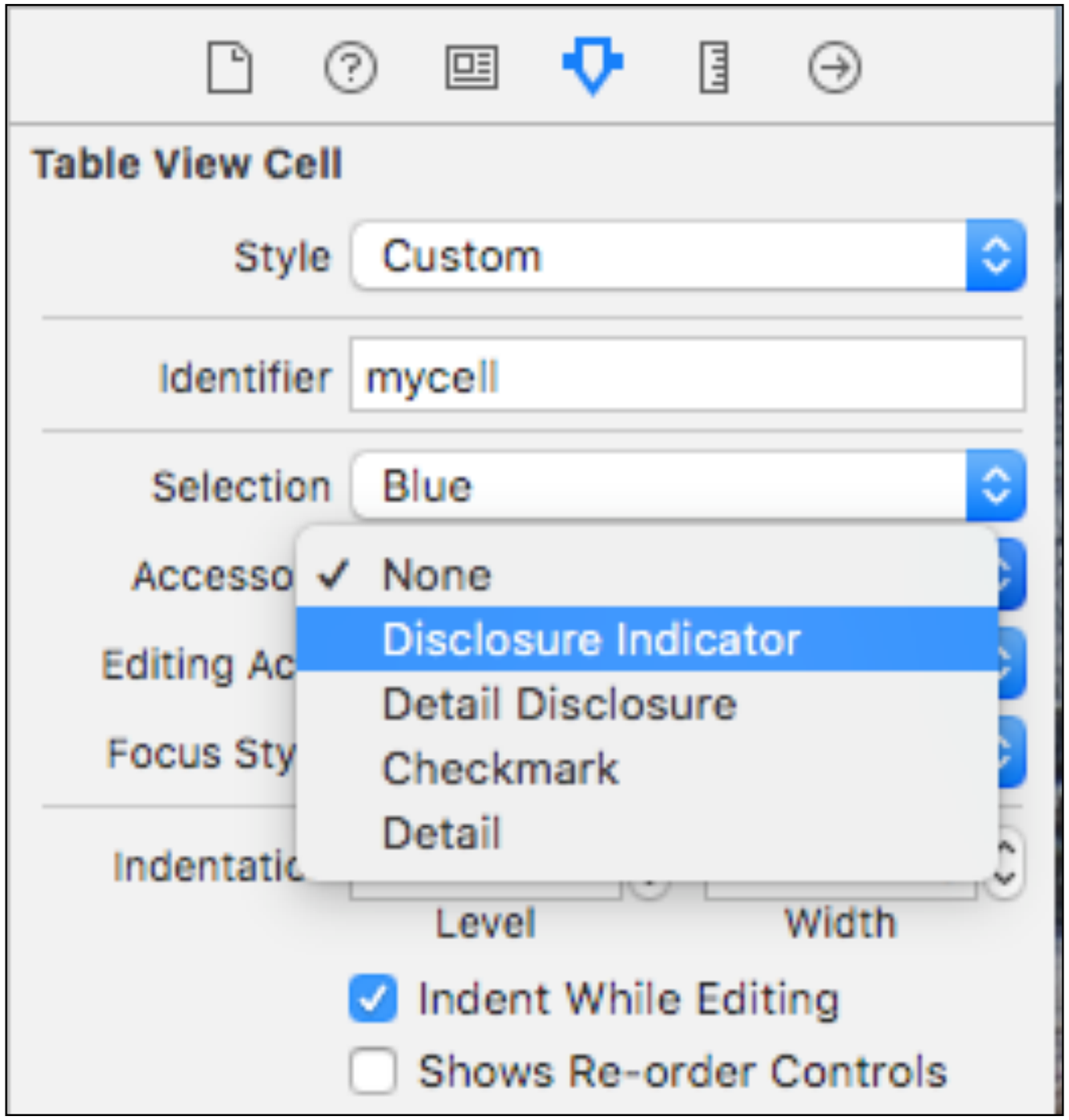
Detail-Darstellung

```
class DetailViewController: UIViewController {

    // public Data
    var studentData:Student!


}
```

Anzeige der Weiterleitung in die Detail-Darstellung

Anklicken der cell

Anzeige der Accessory: Details



Disclosure Indicator

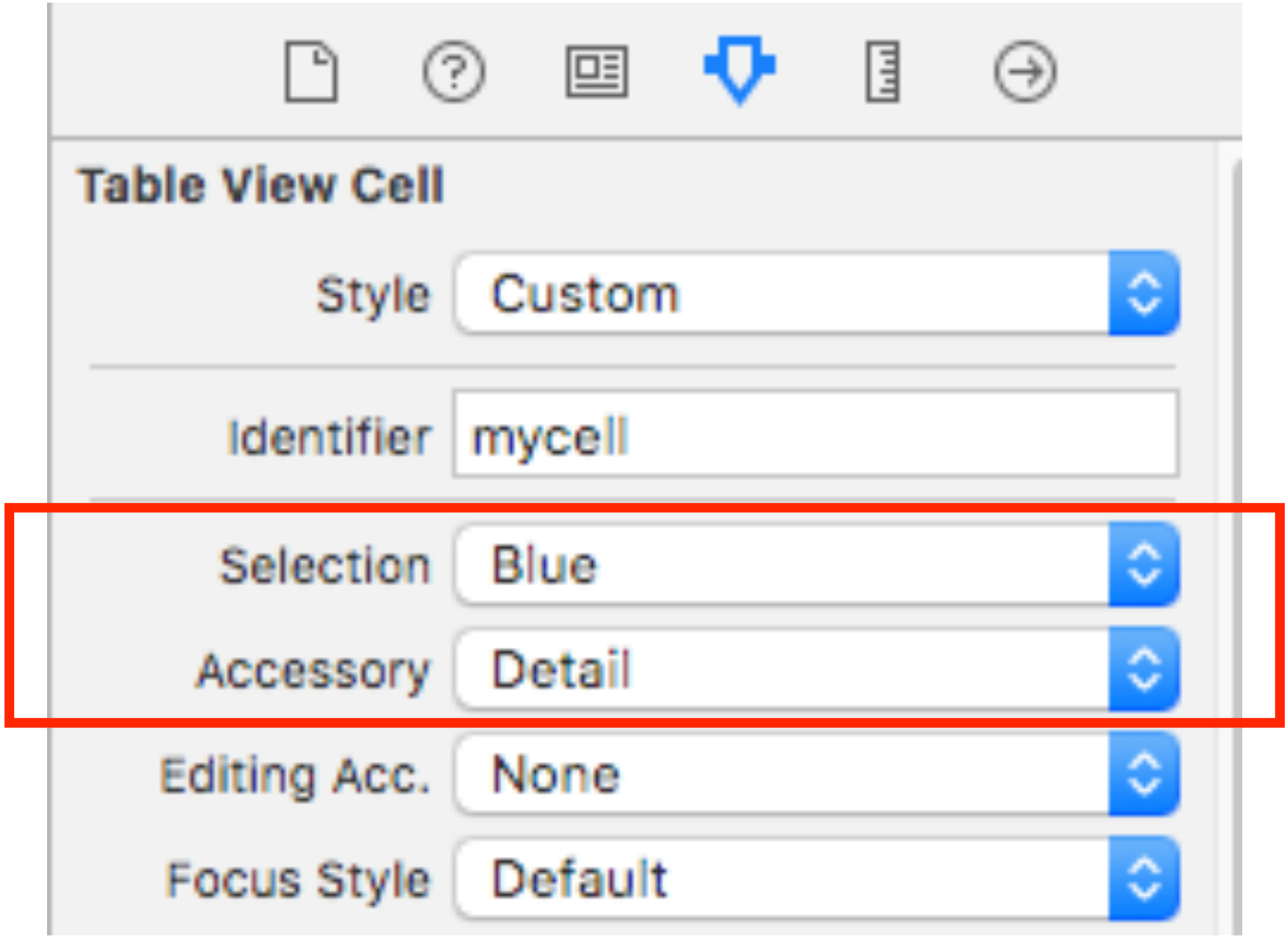
DetailDisclosure

Checkmark

Detail

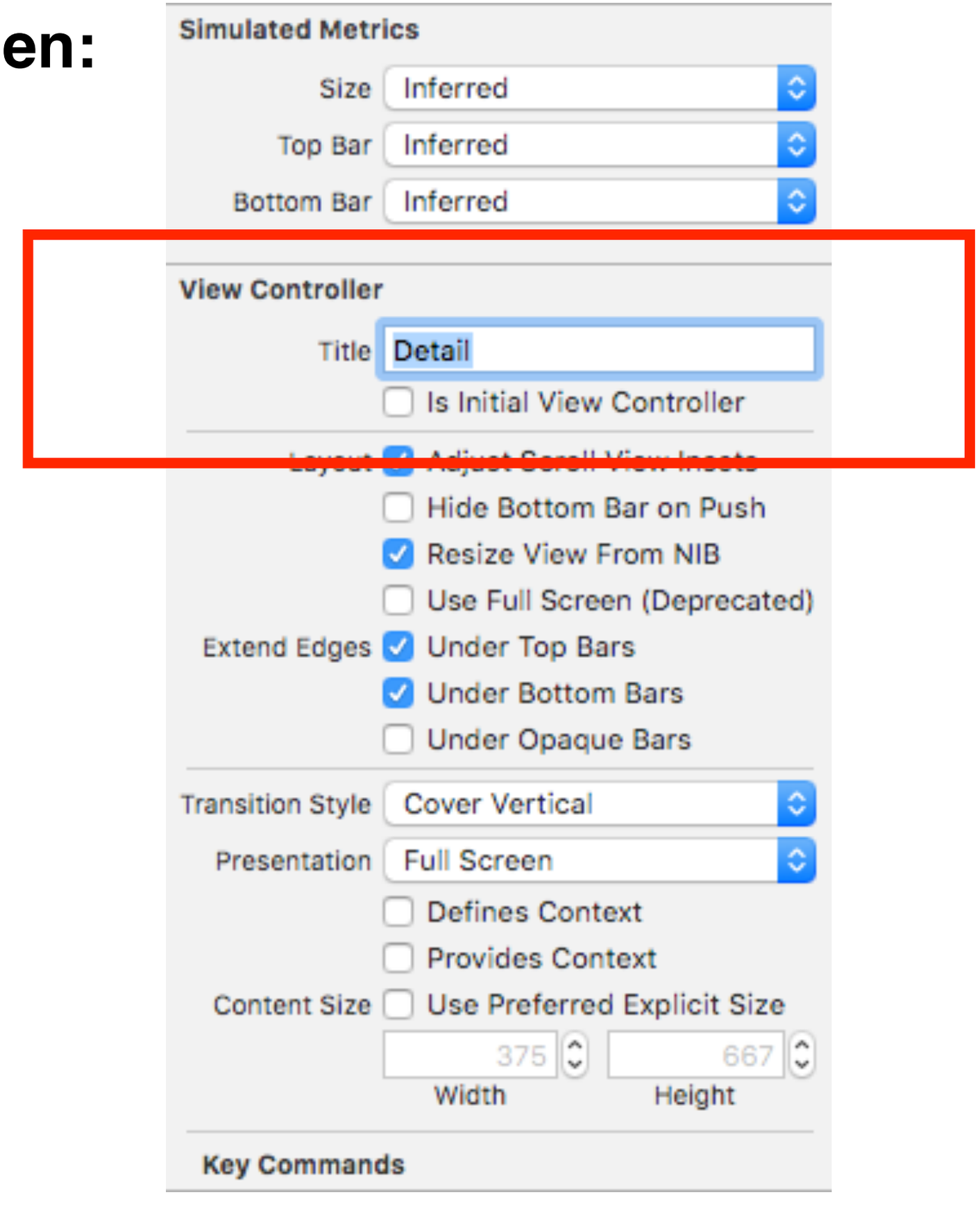
Anzeige der Weiterleitung in die Detail-Darstellung

Anklicken der cell



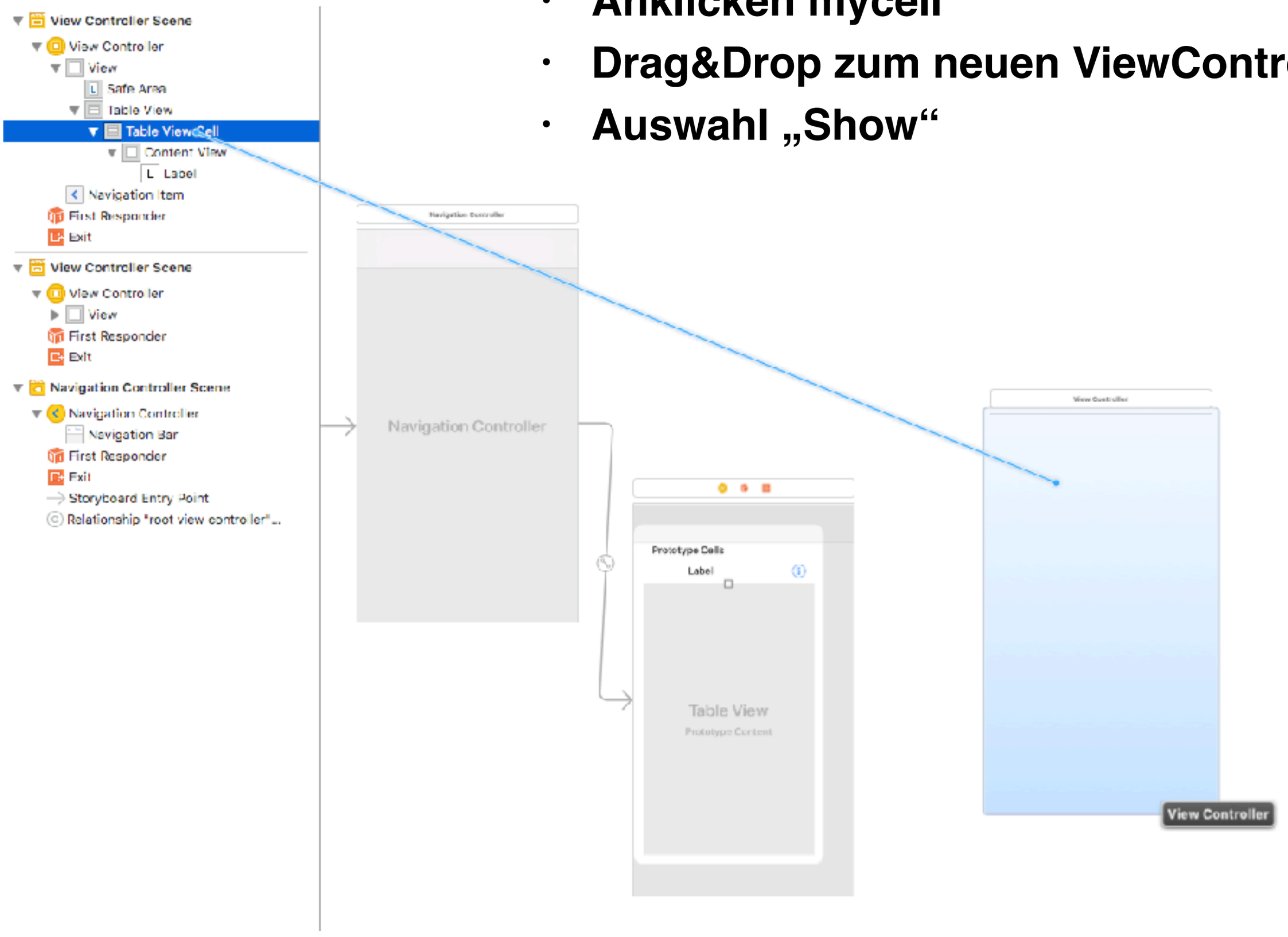
Detail-Darstellung

Namen eintragen:



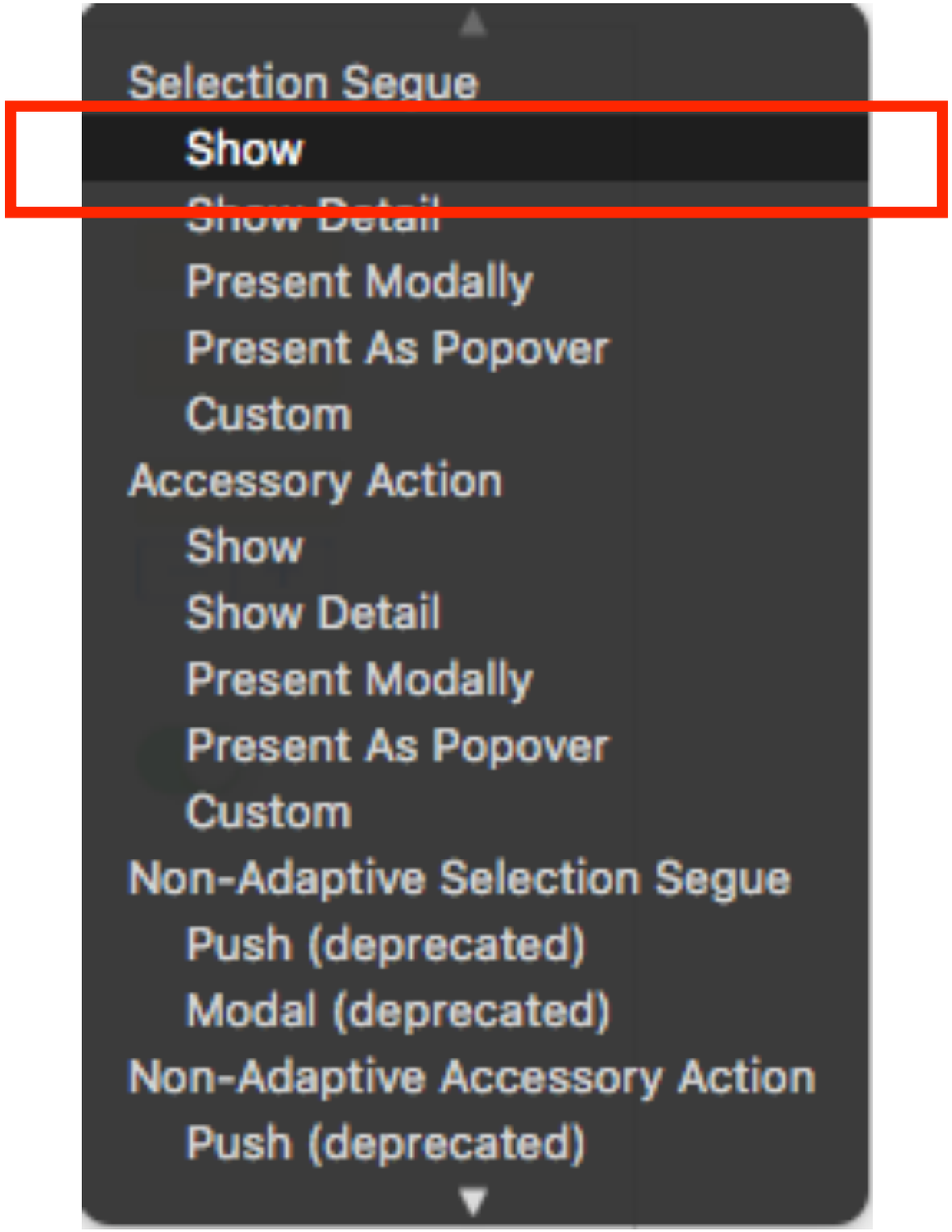
Detail-Darstellung

- **Anklicken mycell**
- **Drag&Drop zum neuen ViewController**
- **Auswahl „Show“**



Detail-Darstellung

Auswahl „Show“

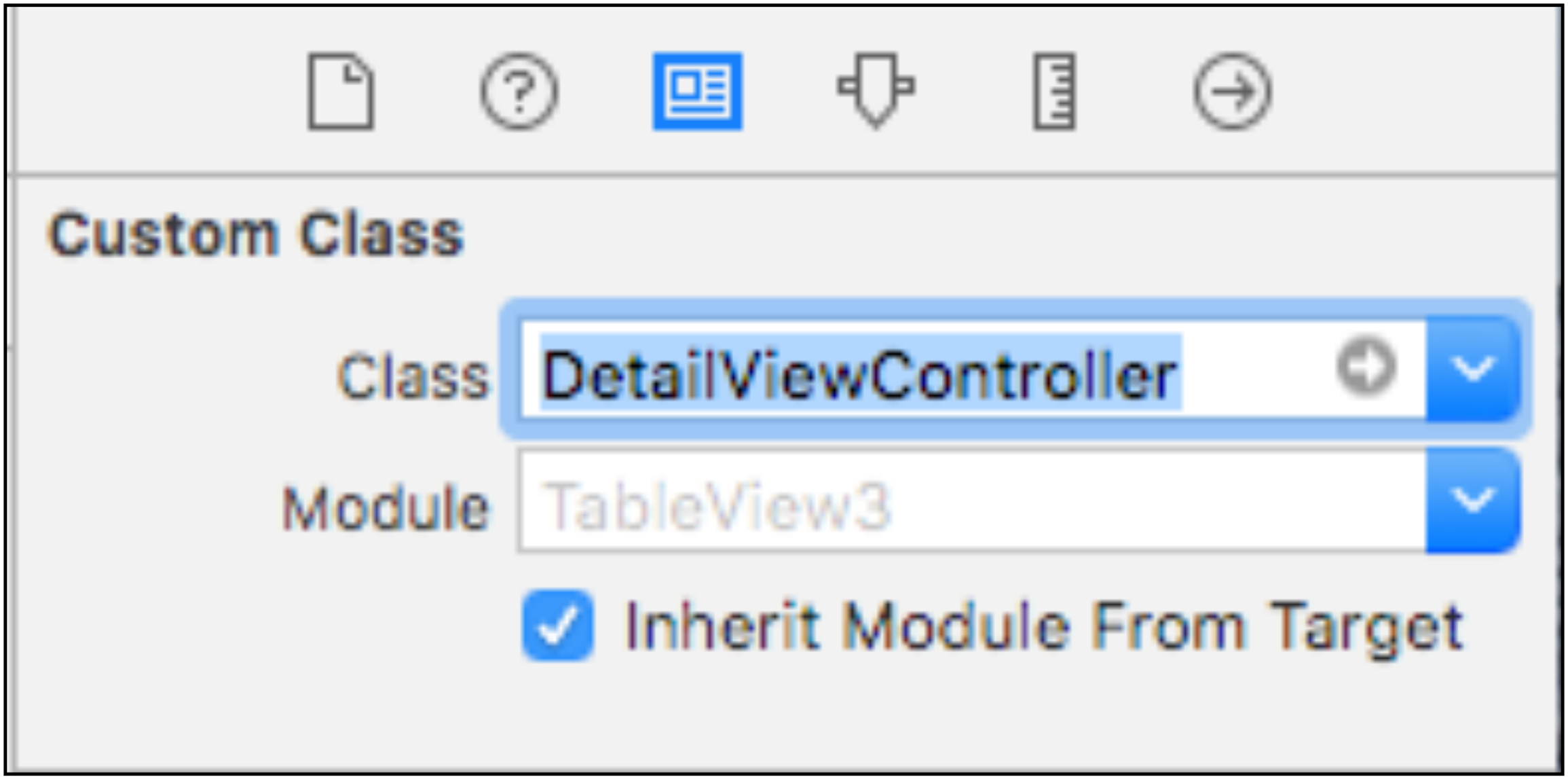


Detail-Darstellung

Zuweisen einer Klasse für den neuen ViewController

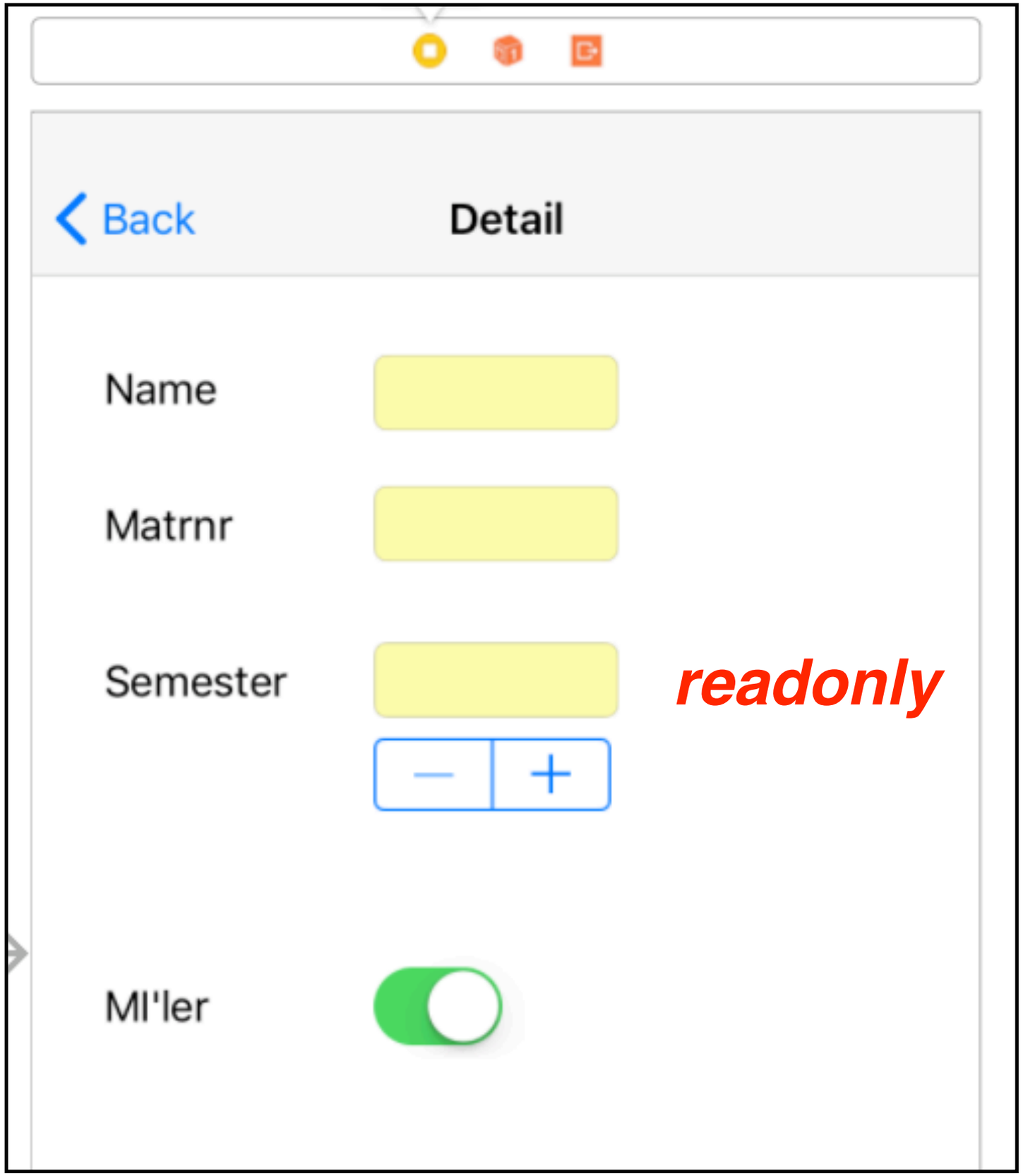
Zweck: Outlets

1. Anklicken des linken Kreises



Detail-Darstellung

- **GUI erstellen**
- **Outlets erstellen**



Detail-Darstellung

```
class DetailViewController: UIViewController {

    // Outlets
    @IBOutlet var tName: UITextField!
    @IBOutlet var tMatrnr: UITextField!
    @IBOutlet var tSemester: UITextField!
    @IBOutlet var switchAbitur: UISwitch!

}
```

Detail-Darstellung

```
class DetailViewController: UIViewController {

    // public Data
    var studentData:Student!

  // Outlets
    @IBOutlet var tName: UITextField!
    @IBOutlet var tMatrnr: UITextField!
    @IBOutlet var tSemester: UITextField!
    @IBOutlet var switchAbitur: UISwitch!


   override func viewDidLoad() {
        super.viewDidLoad()
        tName.text = studentData.name
        tMatrnr.text = String(studentData.mnr)
        tSemester.text = String(studentData.semester)
}
```