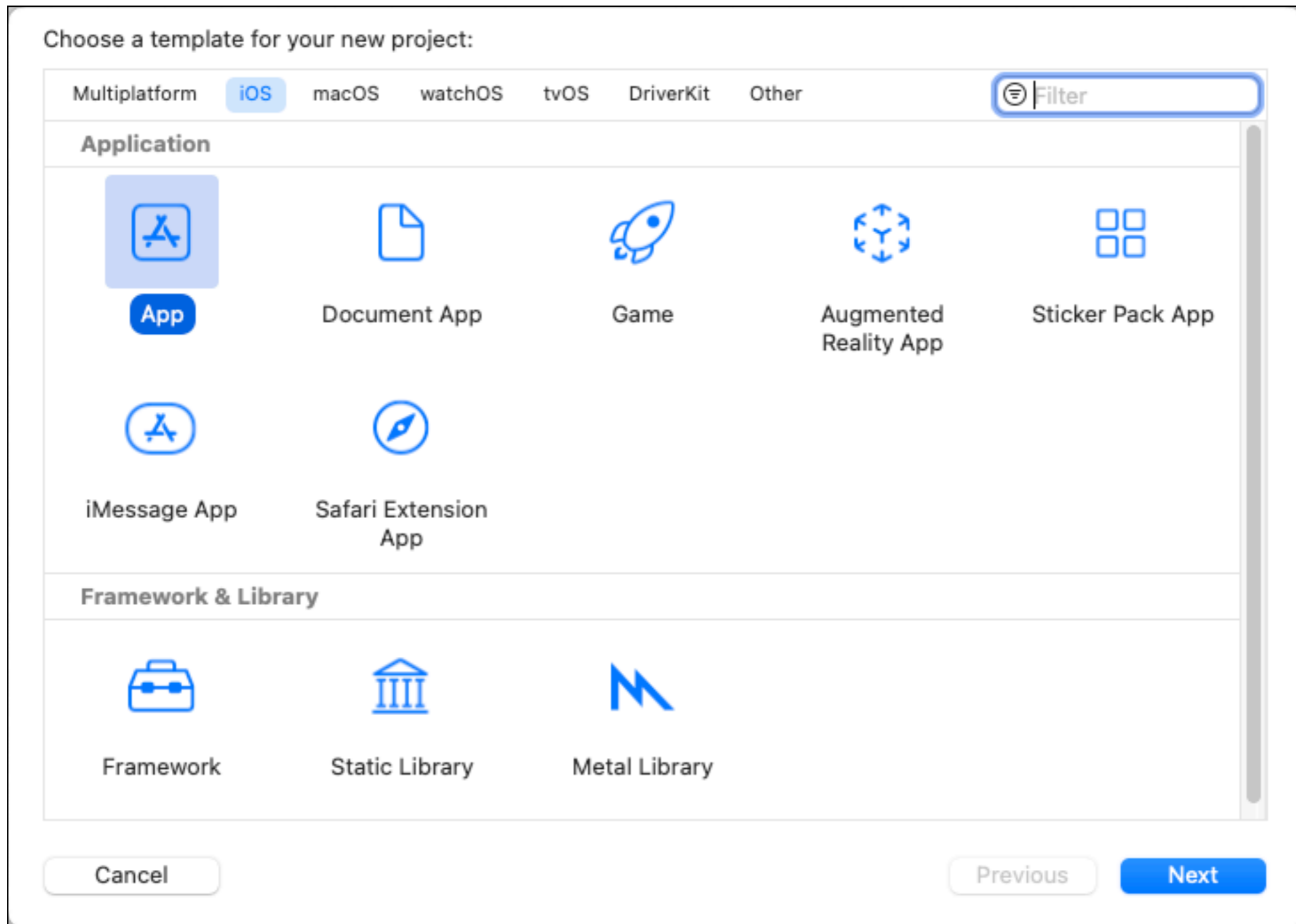


CoreData: iOS mit Datenbanken

- TabbedBar mit
 - Einfügen eines Datensatzes
 - Anzeigen in einem Editor
 - Anzeigen in einer Liste
 - SQL-Konsole
 - Ändern eines Datensatzes
 - Löschen eines Datensatzes

Single View App



Name der App, mit CoreData Option

Choose options for your new project:

Product Name: CoreData01

Team: Add account...

Organization Identifier: hsharz

Bundle Identifier: hsharz.CoreData01

Interface: Storyboard

Language: Swift

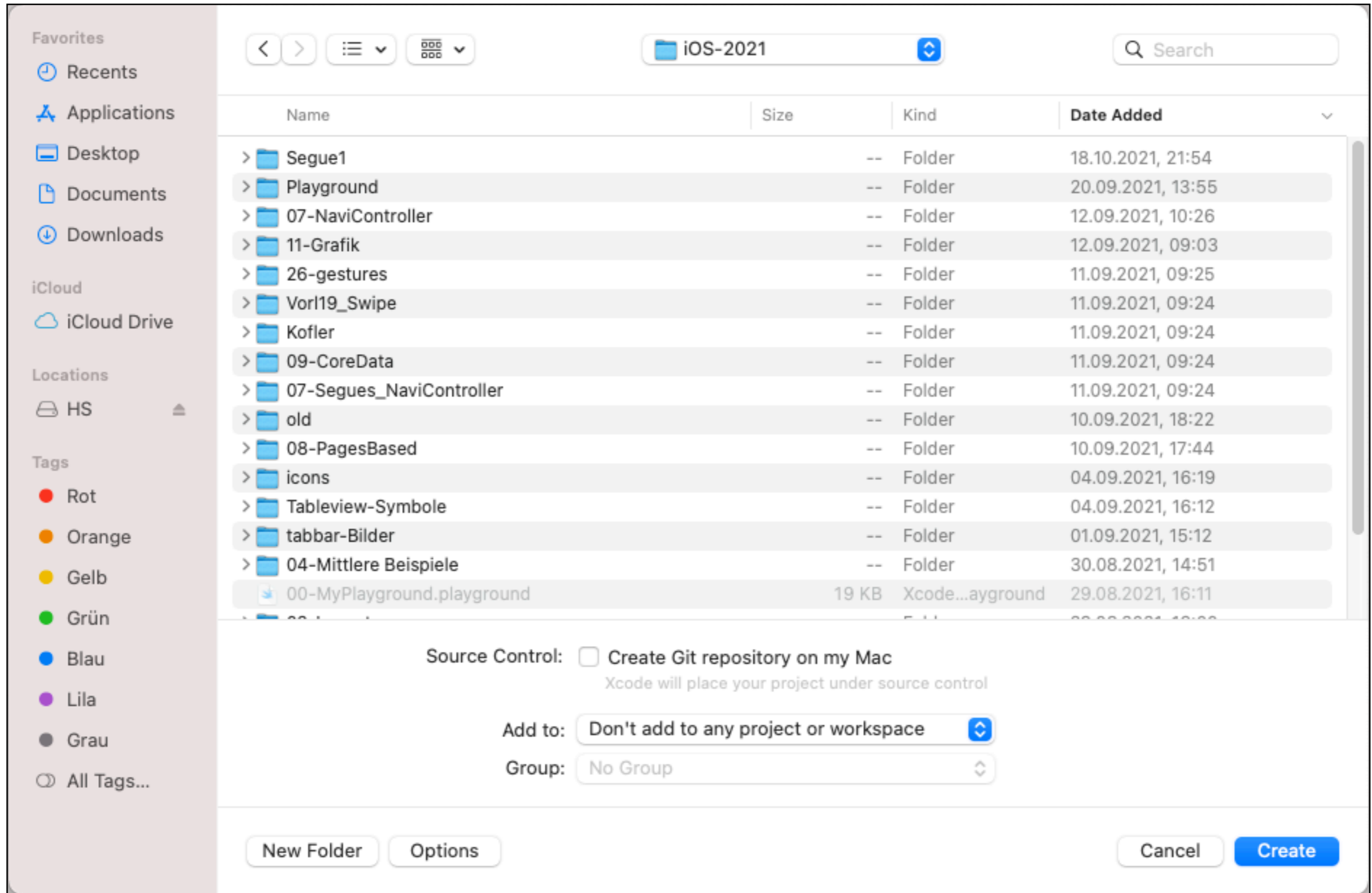
☒ Use Core Data

☐ Host in CloudKit

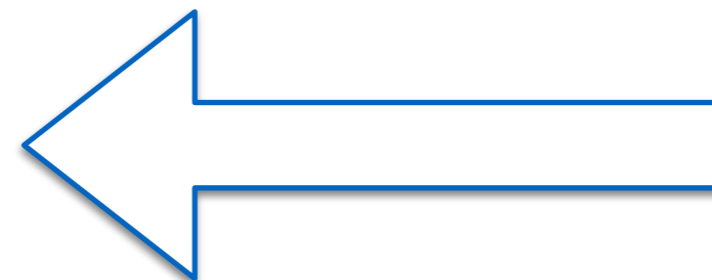
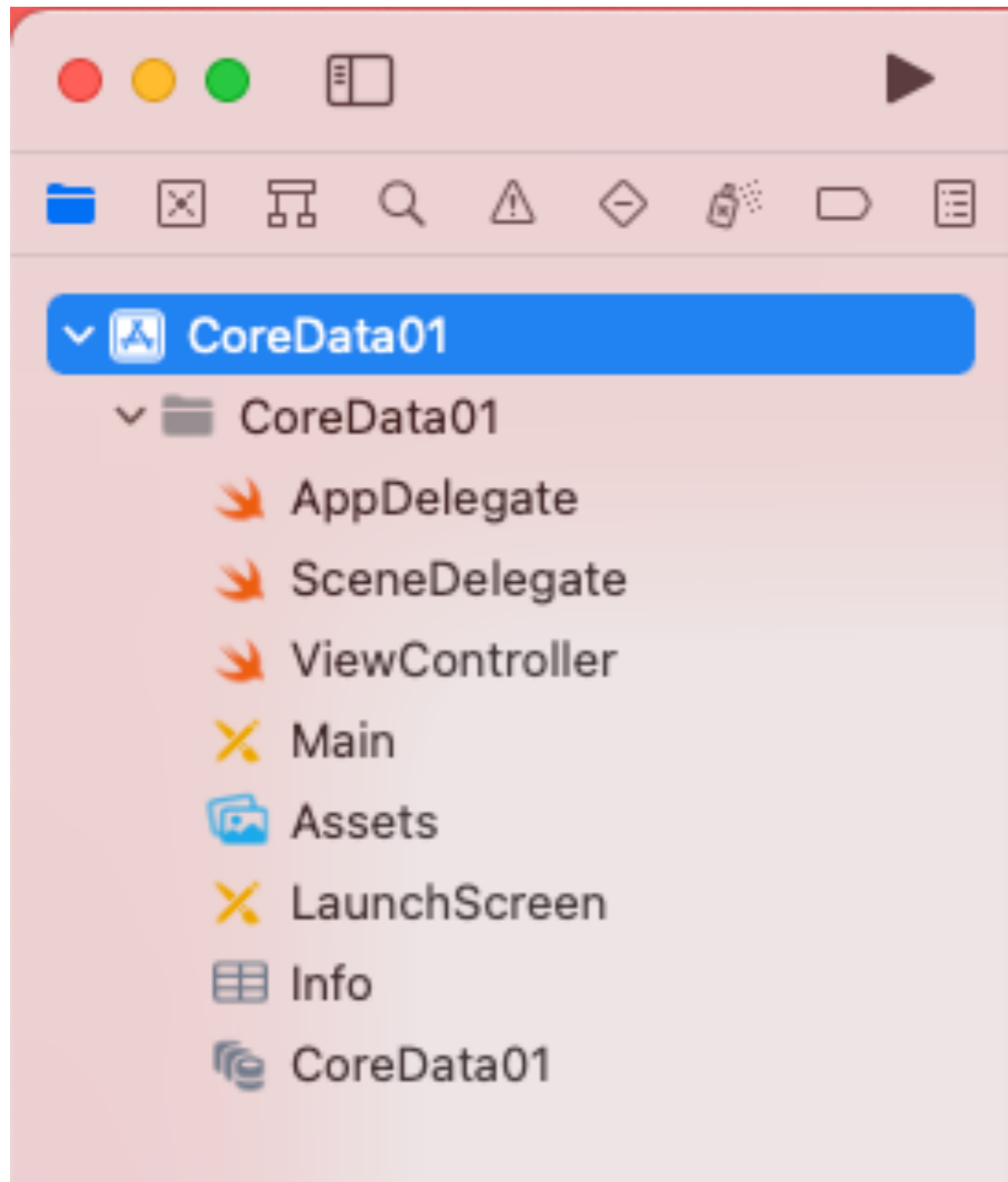
☐ Include Tests

Cancel Previous Next

Speichern im Ordner



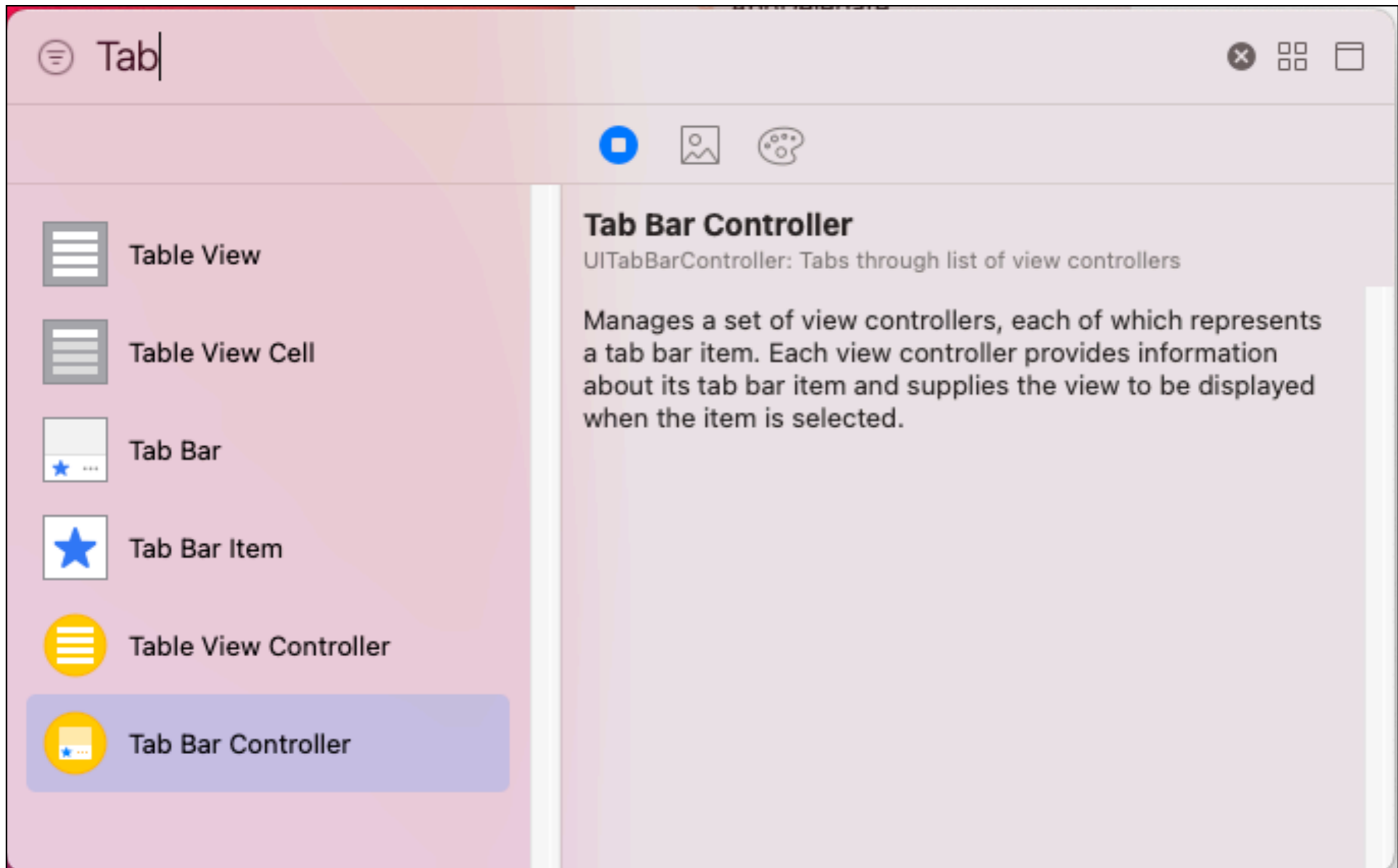
Anzeige der Projektstruktur



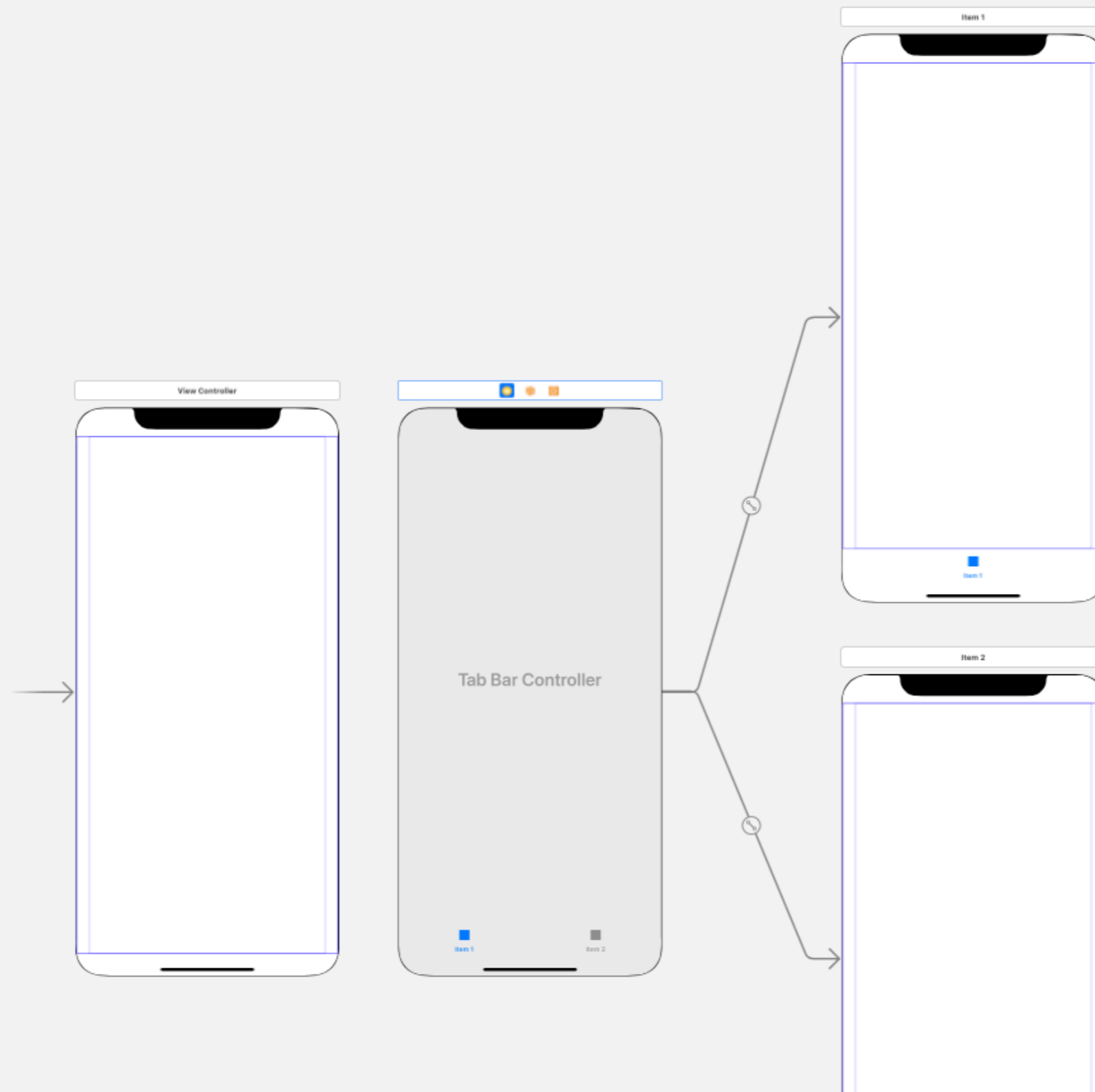
NEU

Einfügen eines Tabbar-Controllers

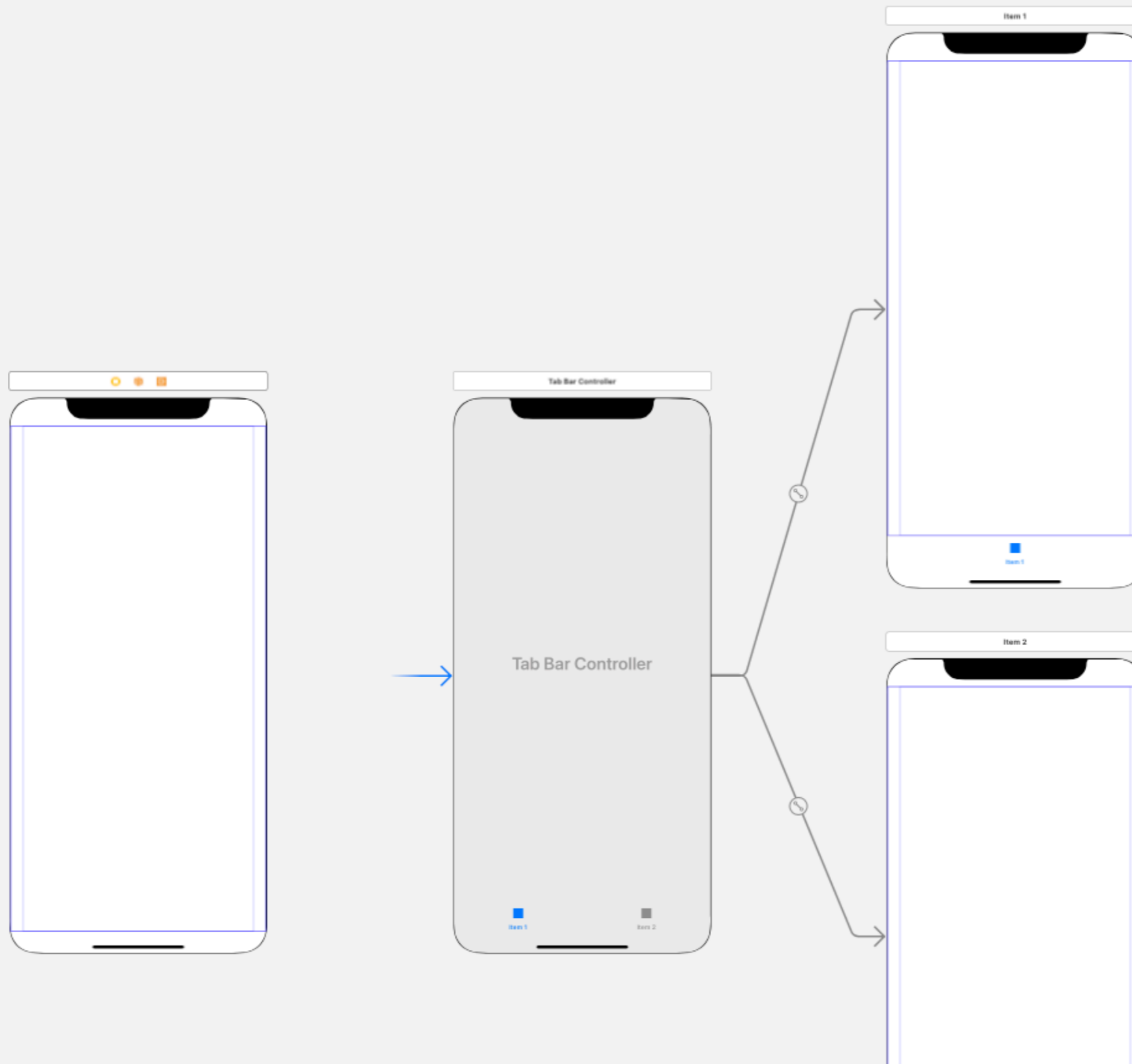
Oder Embedded TabBar Controller



Tabbar-Controllers im Projekt



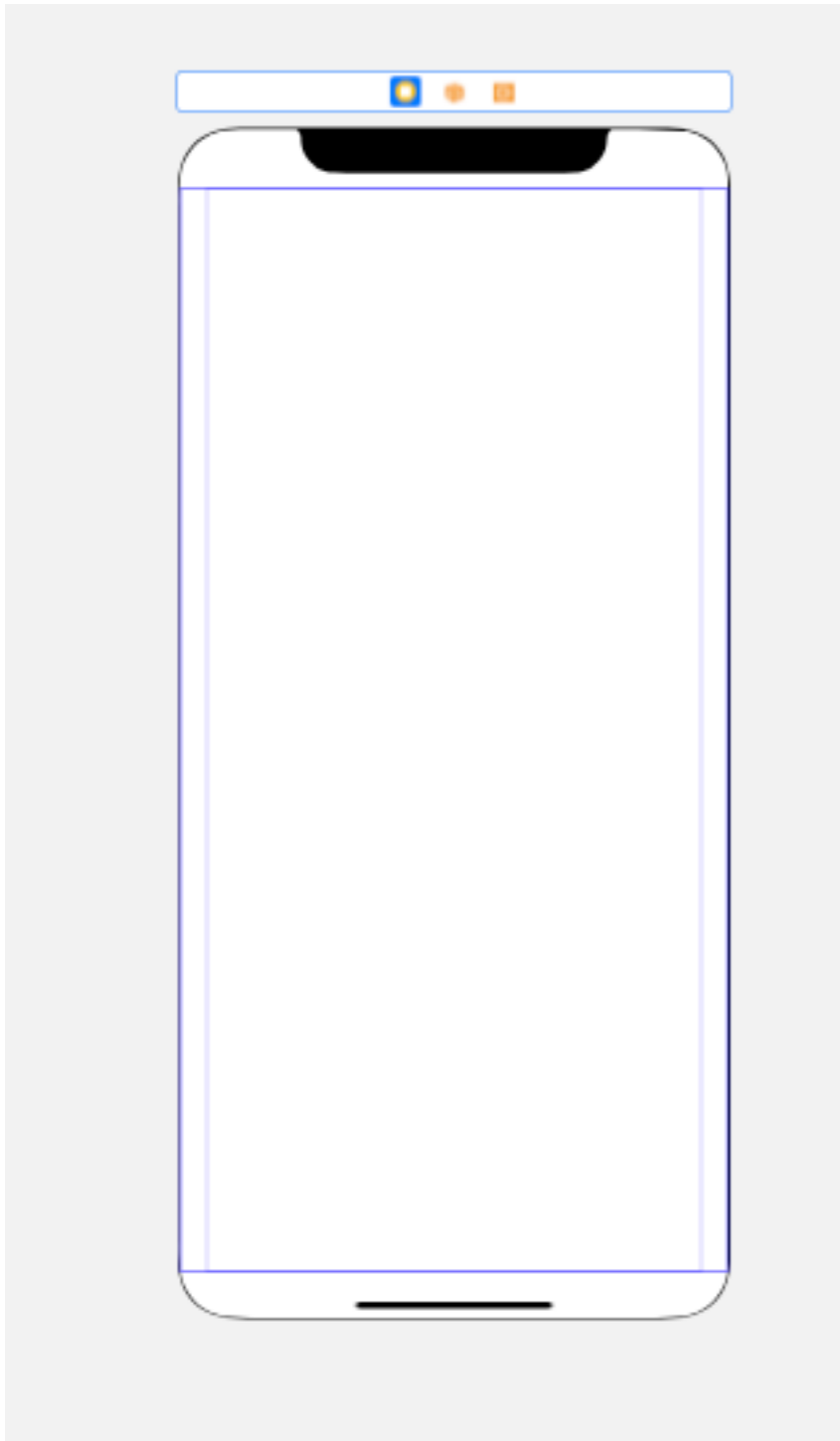
Verschieben des „StartPfeils“



Löschen des ersten ViewControllers

1) Anklicken links oben

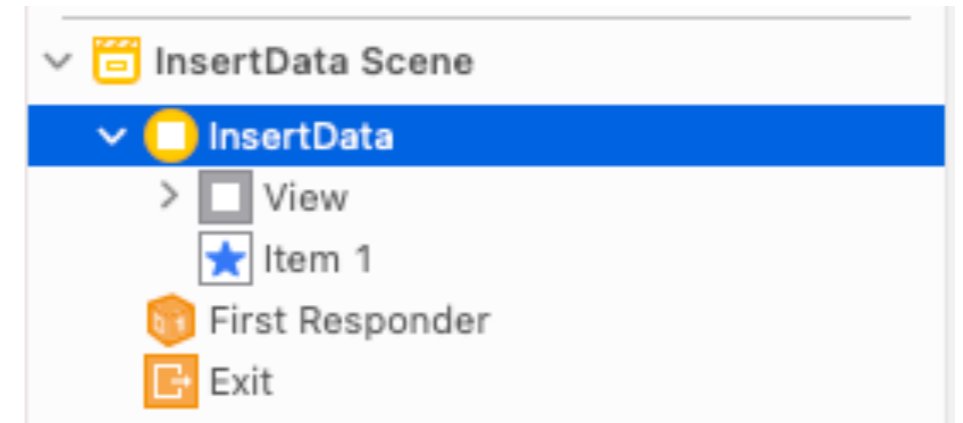
2) Löschen mit Taste Entf



Umbenennen der beiden Tab's

Caption der ViewController

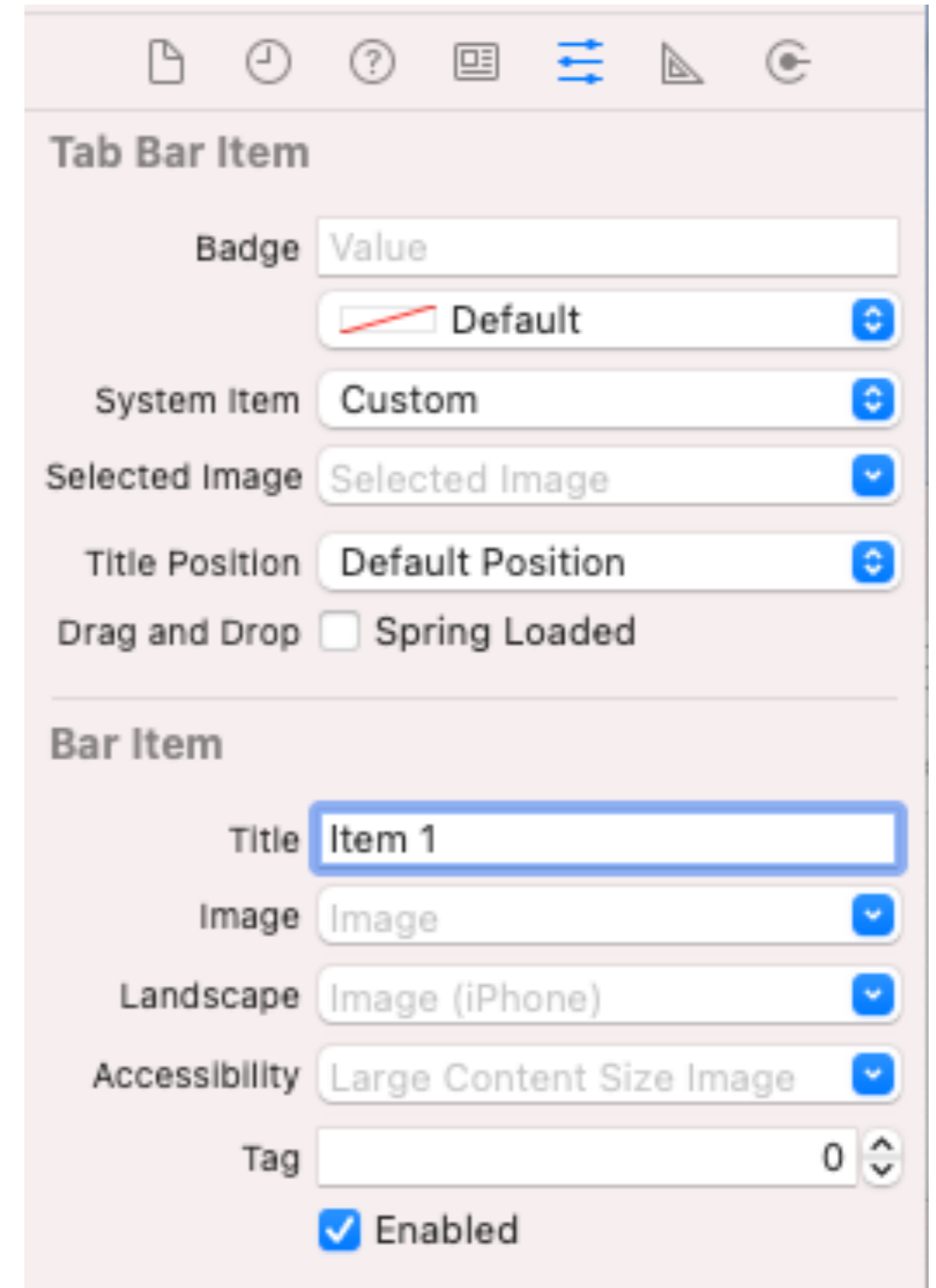
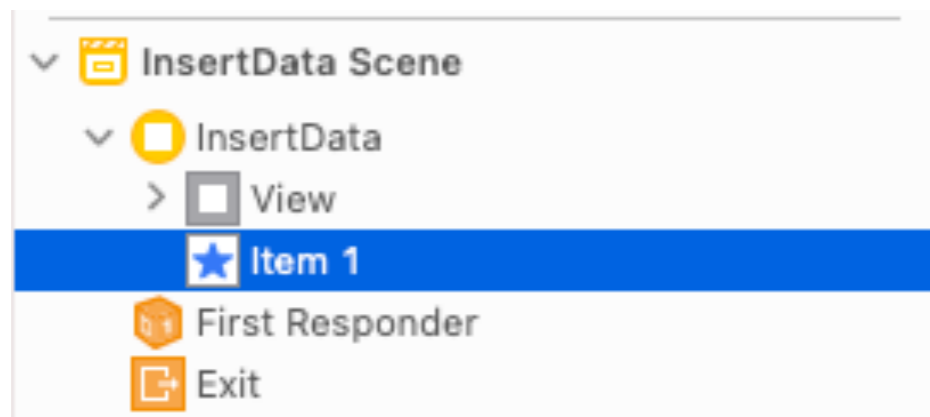
1. „Item 1“ in. InsertData
2. „Item 2“ in „ShowDataEditor“



Umbenennen der beiden Titel

Caption der Tabbar-Items

1. „Item 1“ in. InsertData
2. „Item 2“ in „ShowDataEditor“



CoreData1: UI-Elemente

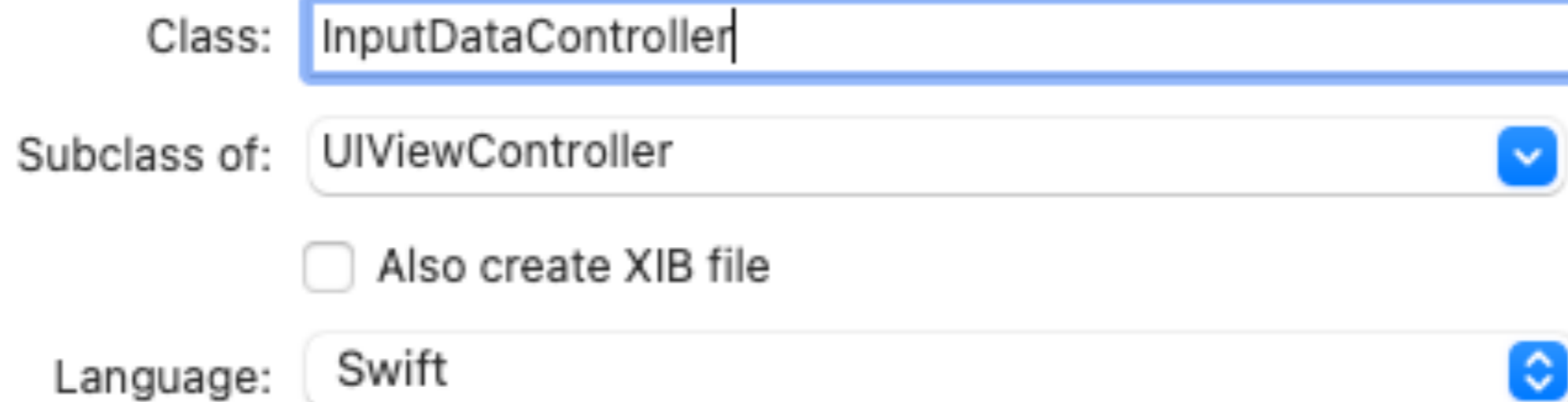
- Label caption (T10, L10,R10)
- Label firstname (T30, L20)
- tfirstname (T23,L75,R40)
- Label lastname (T50, L20)
- tlastname (T37,L60,R40)
- Label empno (T50, L20)
- tempno (T37,L33.5,R40)
- Label department (T50, L20)
- tdepartment (T37,L50,R40)
- Hori-Stack (T90, L10, R10)
 - Button Fontsize: 20, Einfügen
 - Button Fontsize 20, Dummydaten

The image shows a mobile app interface for inserting a new employee. At the top, a title bar contains the text 'Einfügen eines Mitarbeiters' in orange. Below this, there are four input fields arranged vertically, each preceded by a label: 'Vorname', 'Nachname', 'Mitarbeiter-Nr', and 'Department'. The 'Nachname' field has a small square handle on its right side. Below the input fields, there is a horizontal stack of two buttons: 'Einfügen' and 'Dummydaten'. At the bottom of the screen, there is a small blue square icon with the text 'InsertData' below it.

Anlegen ViewController.swift für den 1. ViewController

ViewController.swift für InsertData

- Cmd N
- Cocoa Touch Class
- Class (Name): InputDataController
- SubClass: UIViewController



The screenshot shows the 'New File' dialog in Xcode. The 'Class' field is set to 'InputDataController'. The 'Subclass of' dropdown is set to 'UIViewController'. The 'Also create XIB file' checkbox is unchecked. The 'Language' dropdown is set to 'Swift'.

Class: InputDataController

Subclass of: UIViewController

☐ Also create XIB file


Language: Swift

Anlegen ViewController.swift für den 2. ViewController


ViewController.swift für ShowDataEditor

- Cmd N
- Cocoa Touch Class
- Class (Name): ShowDataEditor
- SubClass: UIViewController

Class:

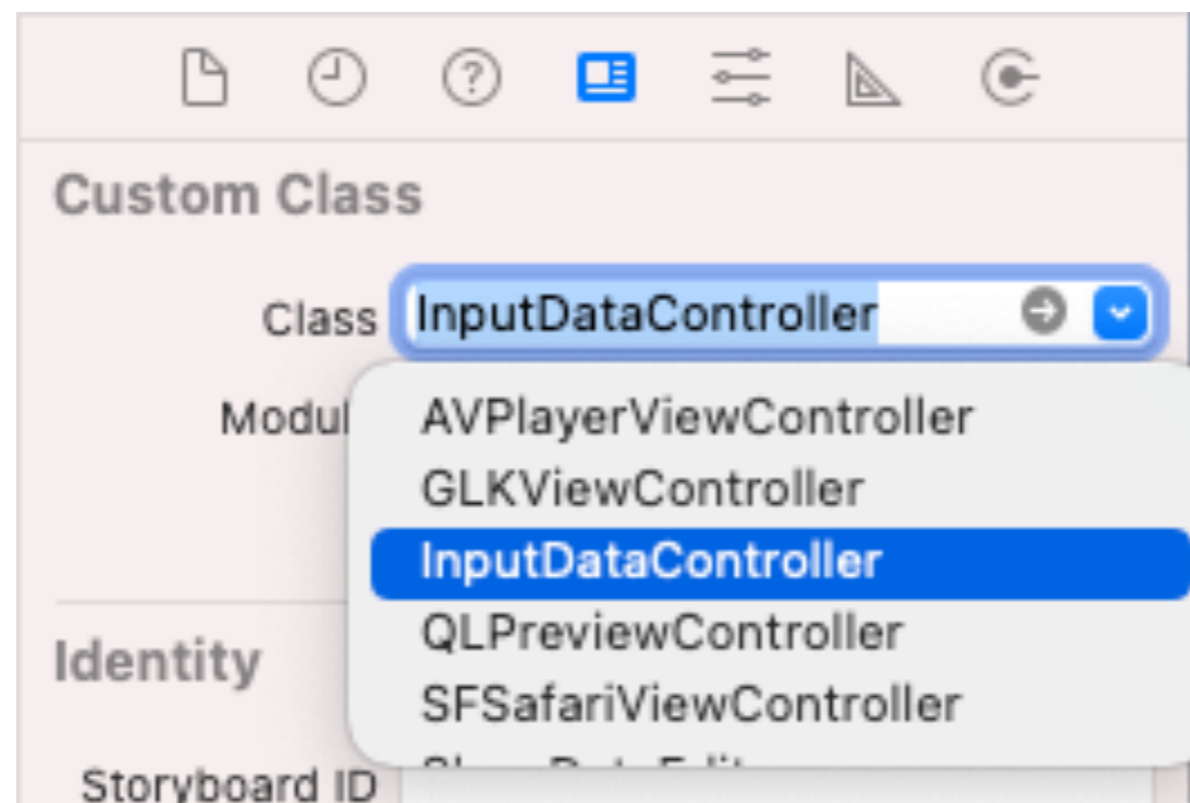
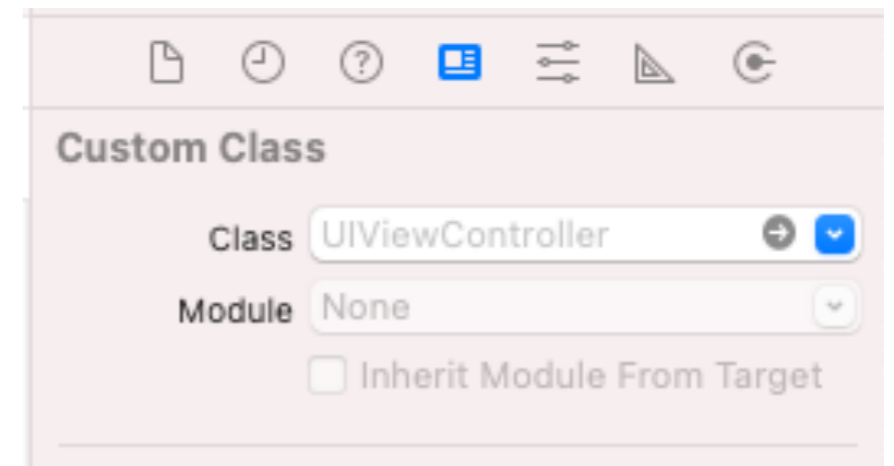
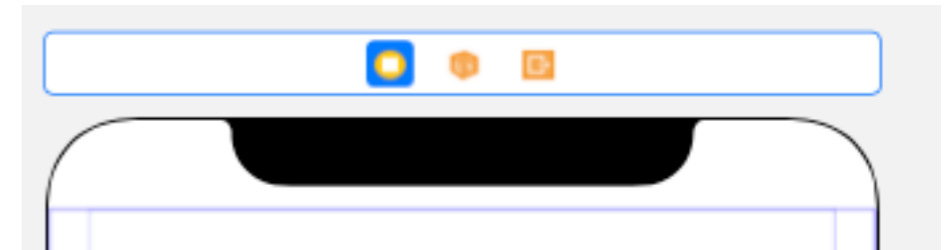
Subclass of: 

☐ Also create XIB file

Language: 

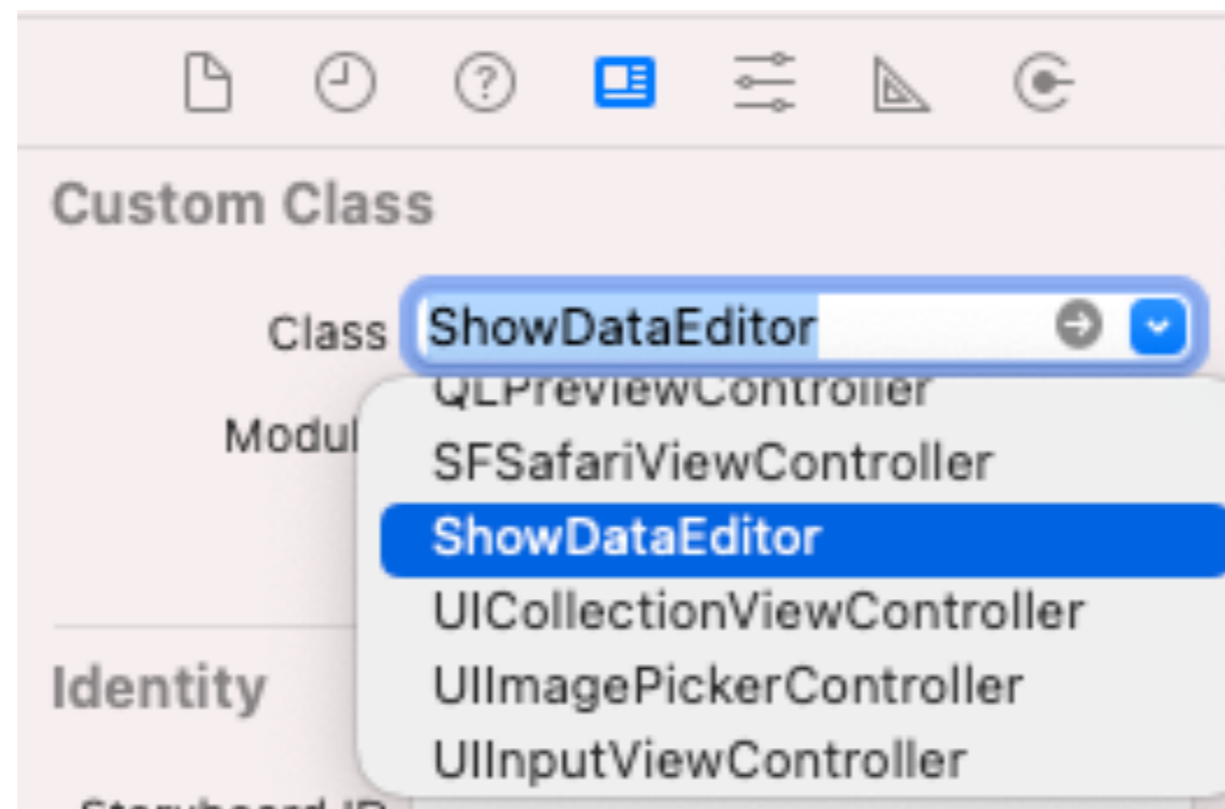
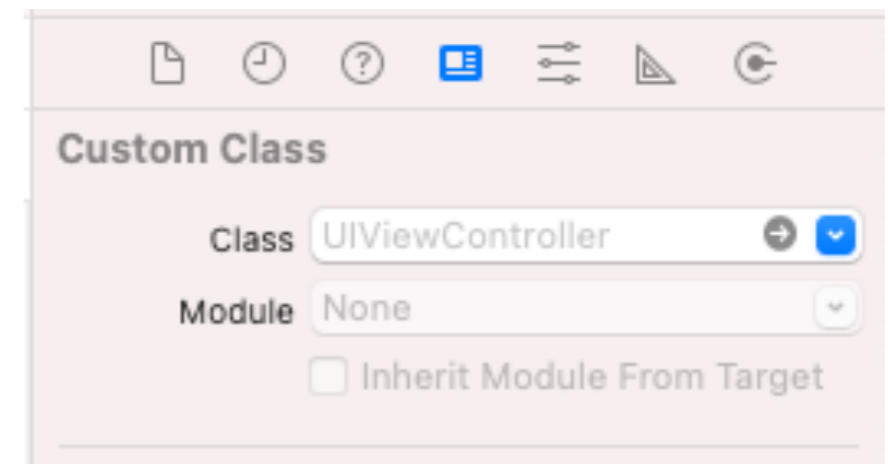
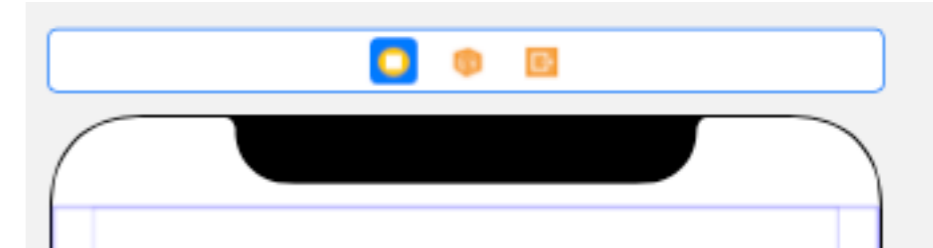
Zuweisen der swift-Datei zum 1. Controller

1. Anklicken des linken Punkt im 1. Controller
2. Anklicken des mittleren Schalters im rechten Property-Fenster
3. Suchen der Swift-Datei in der ComboBox Liste)



Zuweisen der swift-Datei zum 2. Controller

1. Anklicken des linken Punkt im 1. Controller
2. Anklicken des mittleren Schalters im rechten Property-Fenster
3. Suchen der Swift-Datei in der ComboBox Liste)



Referenzen und Events beim 1. Controller

The image shows the Xcode interface with two panels. The left panel displays the Swift code for the `InputDataController` class, which inherits from `UIViewController`. The right panel shows the storyboard for the `Main` scene, featuring a form titled "Einfügen eines Mitarbeiters" with input fields for "Vorname", "Nachname", "Mitarbeiter-Nr", and "Department", and two buttons at the bottom: "Einfügen" and "Dummydaten".

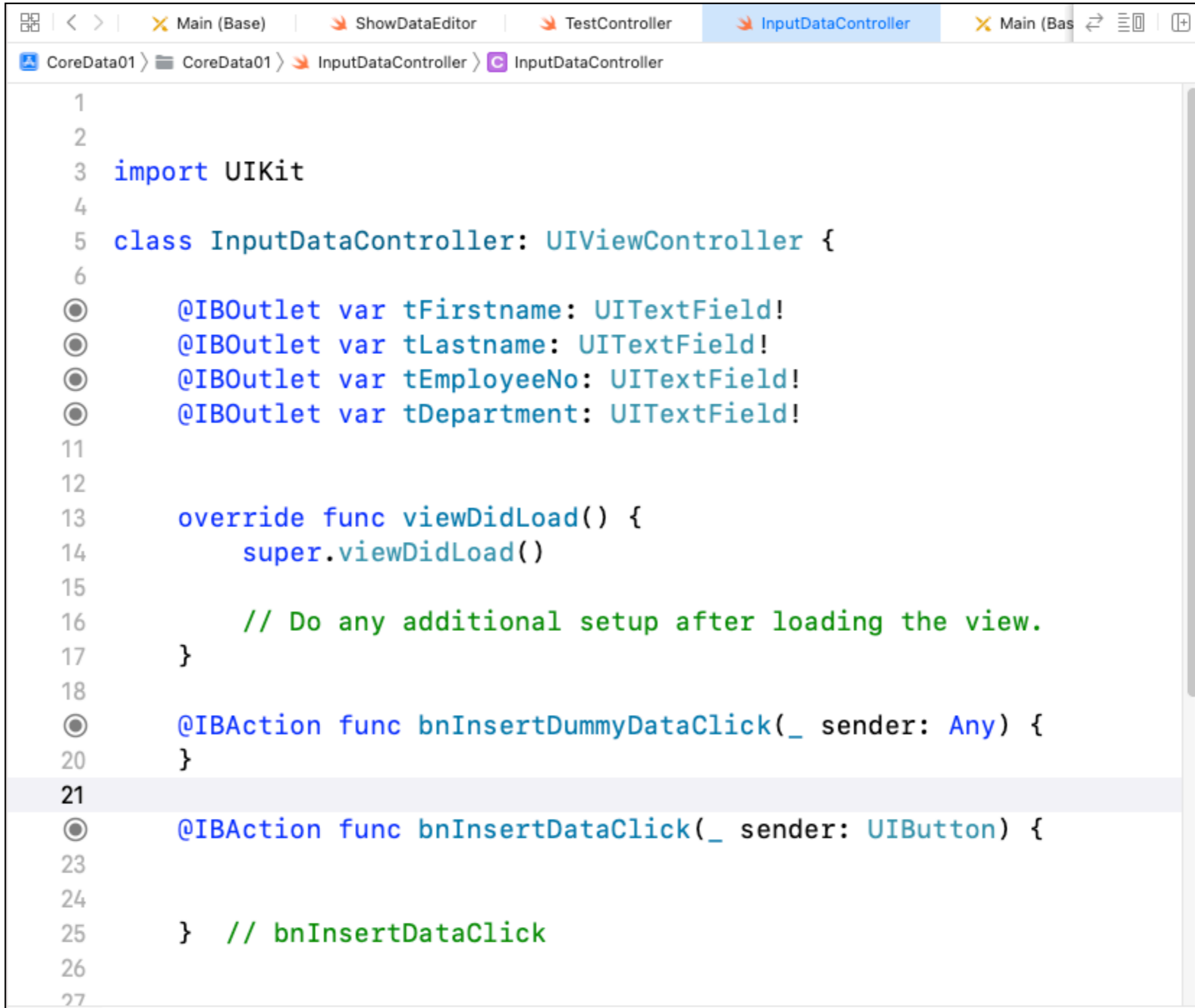
```
1
2
3 import UIKit
4
5 class InputDataController: UIViewController {
6
7
8
9     override func viewDidLoad() {
10         super.viewDidLoad()
11
12         // Do any additional setup after loading the view.
13     }
14
15
16
17
18 }
19
```

The storyboard on the right shows a form titled "Einfügen eines Mitarbeiters" with the following elements:

- Input field for "Vorname"
- Input field for "Nachname"
- Input field for "Mitarbeiter-Nr"
- Input field for "Department"
- Buttons: "Einfügen" and "Dummydaten"

An arrow points from the `viewDidLoad()` method in the code to the storyboard, indicating where the UI setup logic would be implemented.

Referenzen und Events



```
1
2
3 import UIKit
4
5 class InputDataController: UIViewController {
6
7     @IBOutlet var tFirstname: UITextField!
8     @IBOutlet var tLastname: UITextField!
9     @IBOutlet var tEmployeeNo: UITextField!
10    @IBOutlet var tDepartment: UITextField!
11
12
13    override func viewDidLoad() {
14        super.viewDidLoad()
15
16        // Do any additional setup after loading the view.
17    }
18
19    @IBAction func bnInsertDummyDataClick(_ sender: Any) {
20    }
21
22    @IBAction func bnInsertDataClick(_ sender: UIButton) {
23
24
25    } // bnInsertDataClick
26
27
```

CoreData

1. Erstellen der Datenbank

- Tabelle „Employee“
- Tabelle „DBVersion“

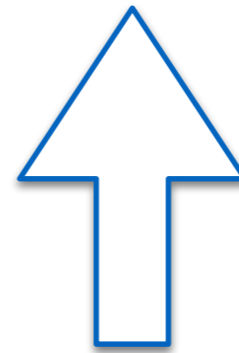
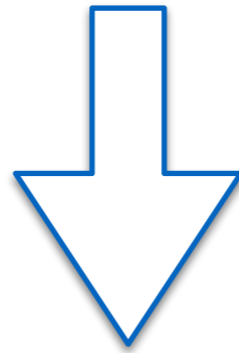
2. Erstellen der Wrapperklassen

3. Datei DBQueries

- Implementieren der DB-Methoden
 - A. Insert-, Delete-, getEmployee
 - B. getDBVersion

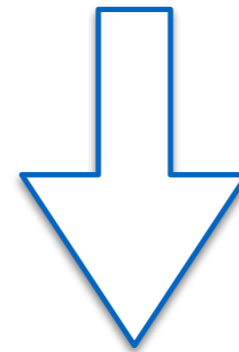
4. .

App



Objekte

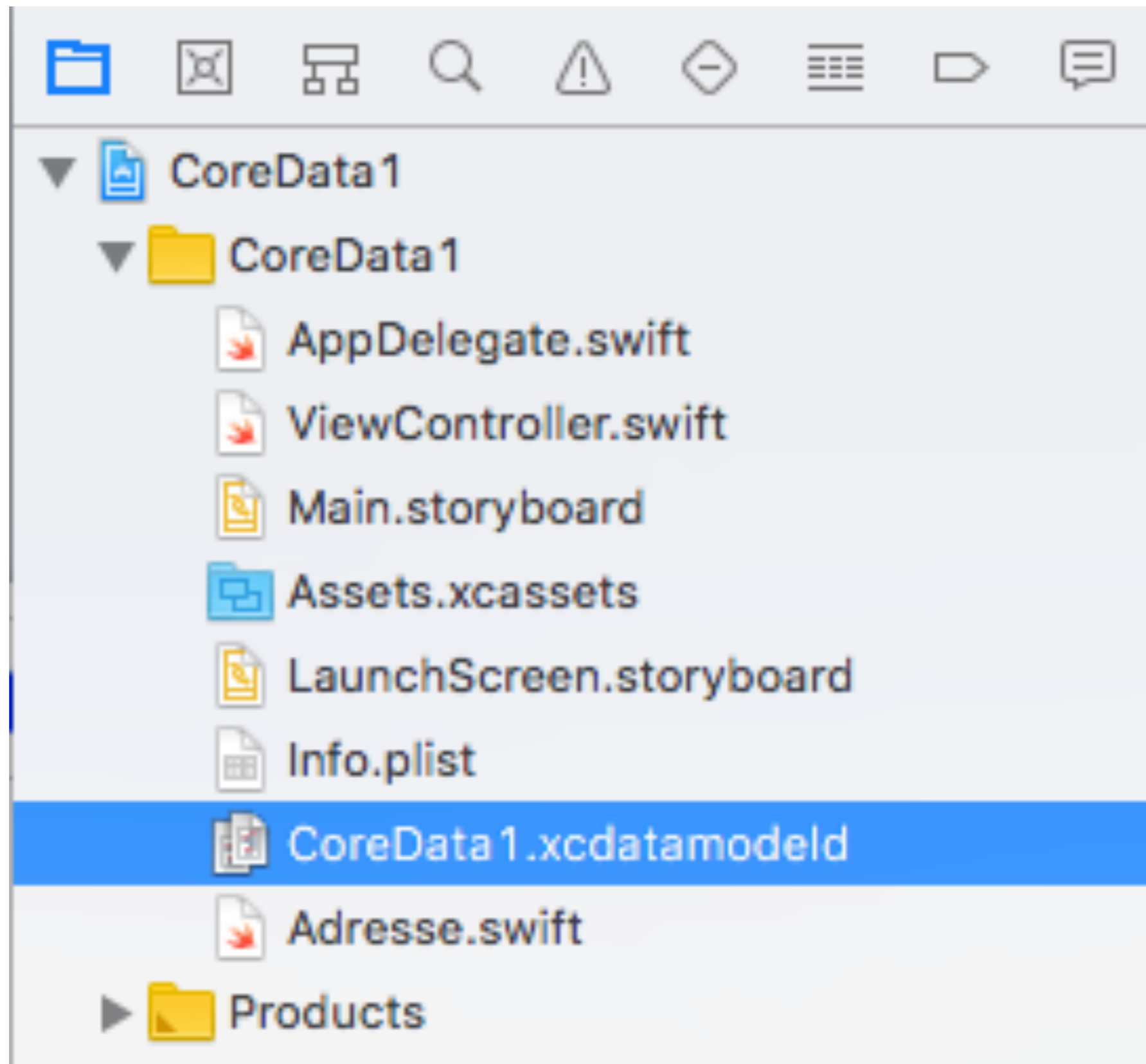
Managed Object Context



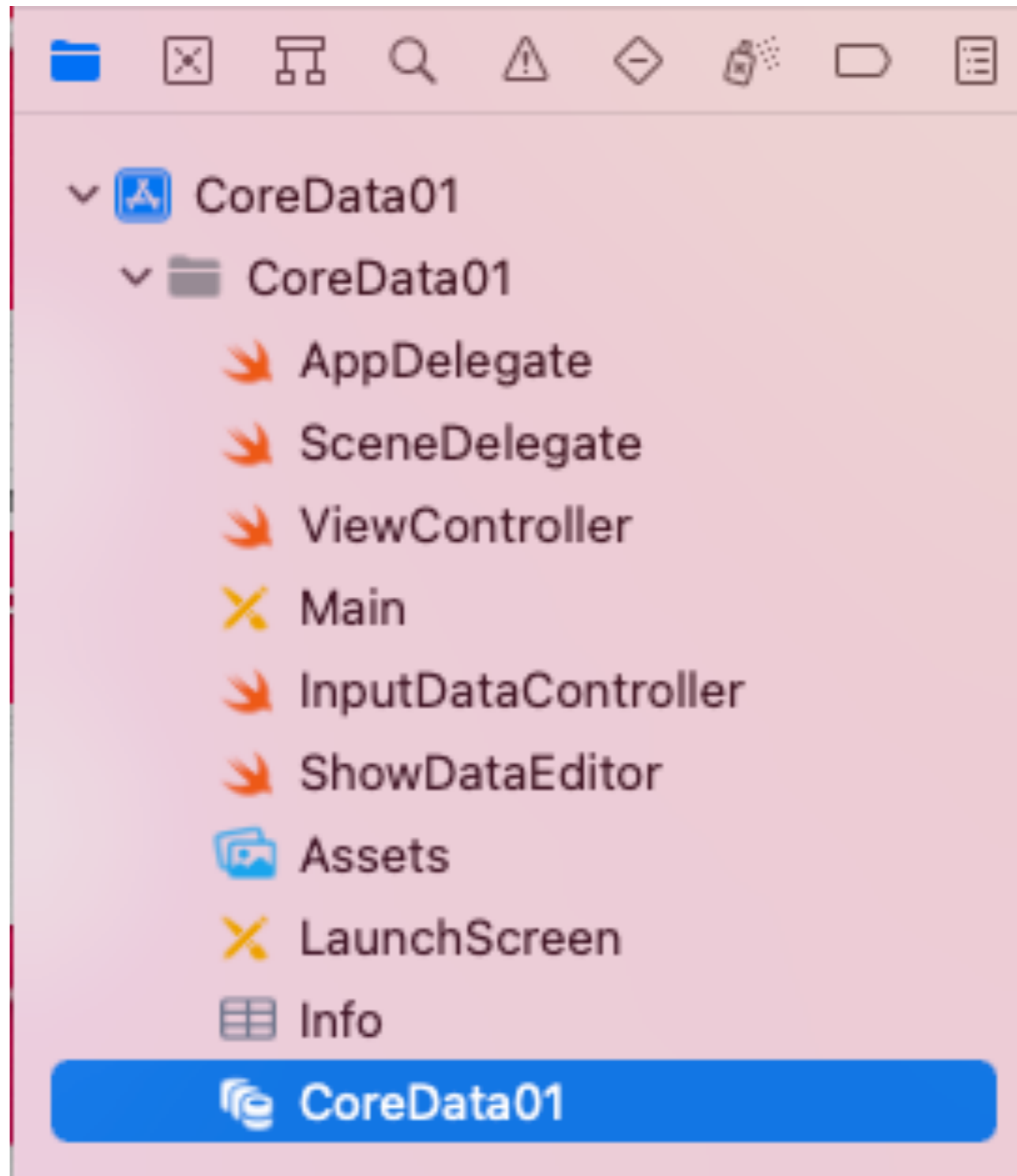
Binärdaten

Core Data Persistent
Container

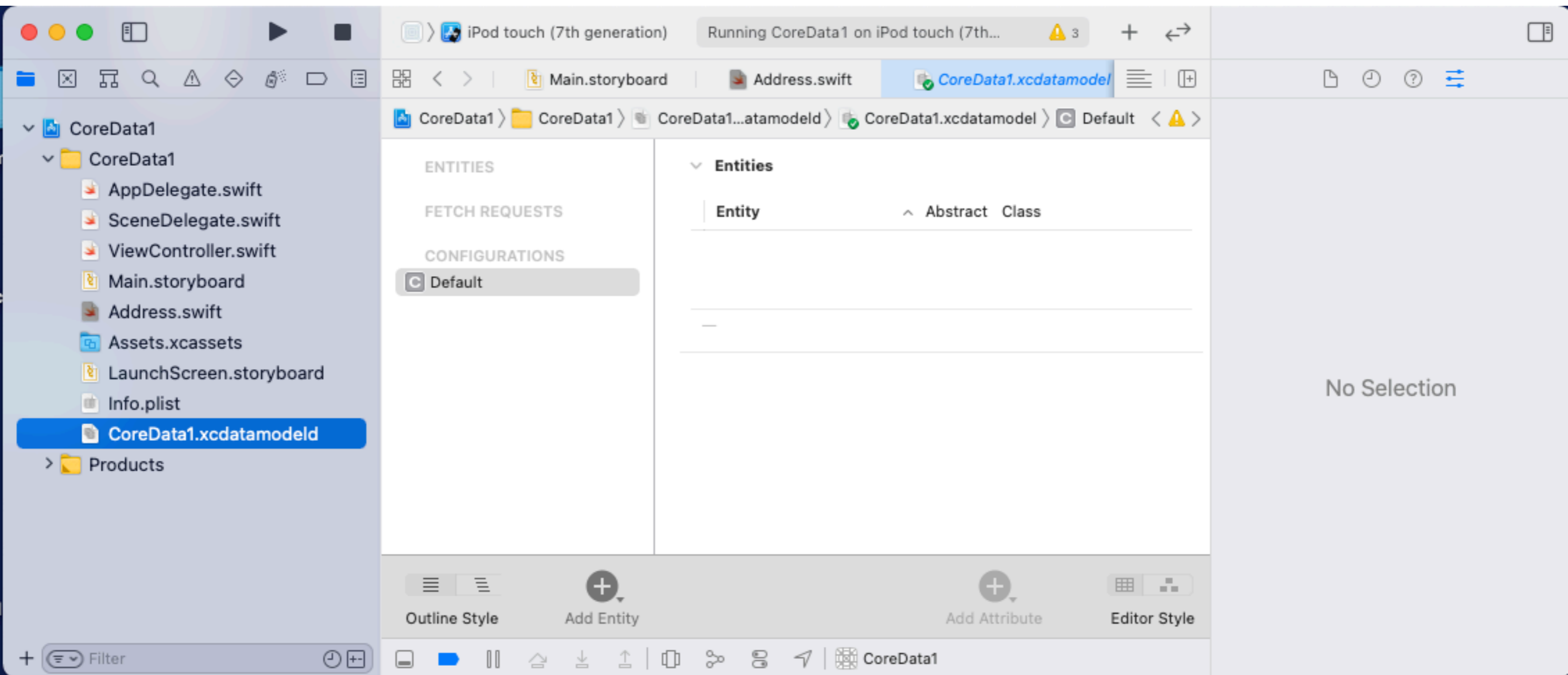
CoreData1:Datenbankmodell



CoreData1:Datenbankmodell



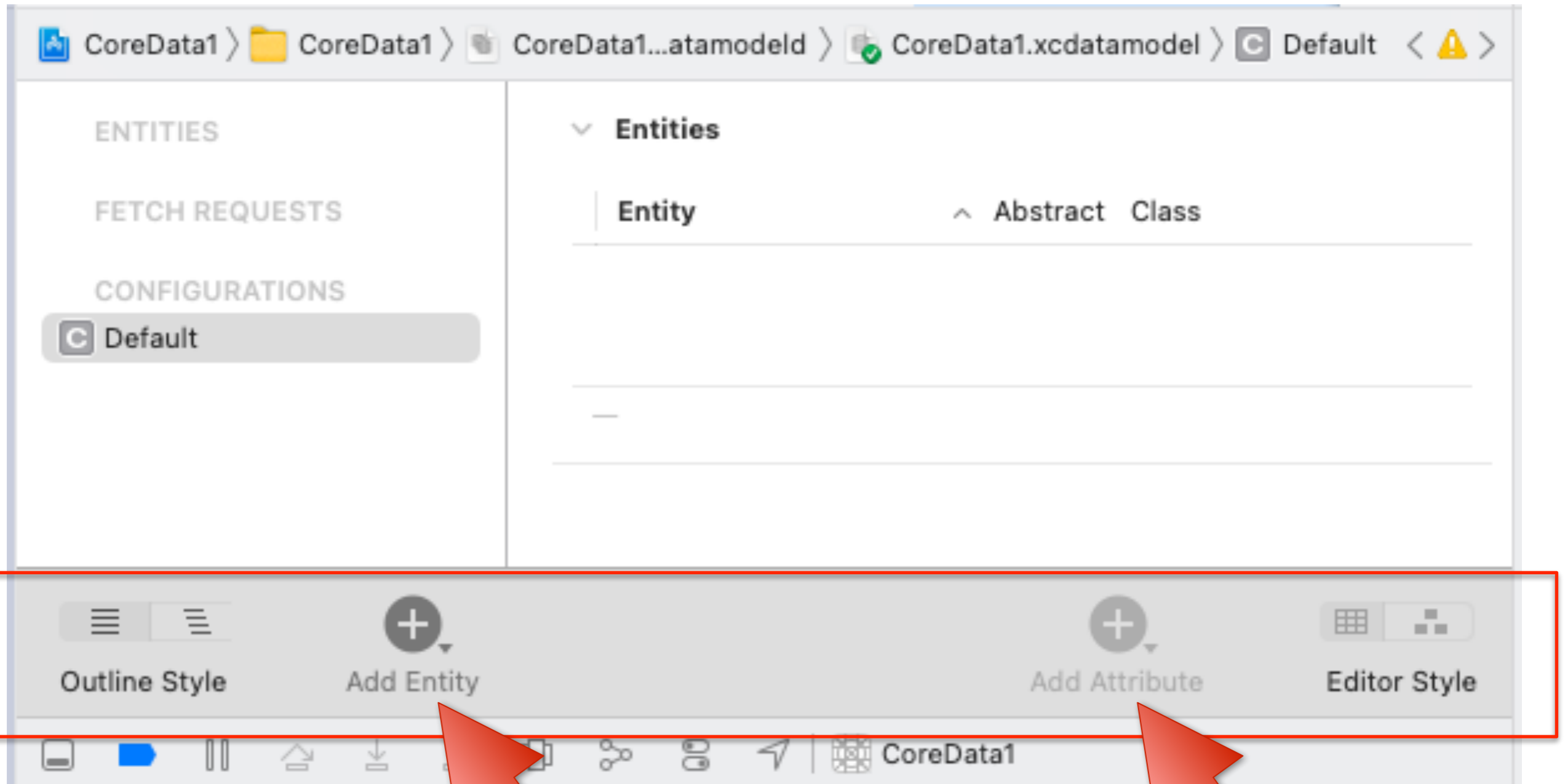
CoreData1:Datenbankmodell



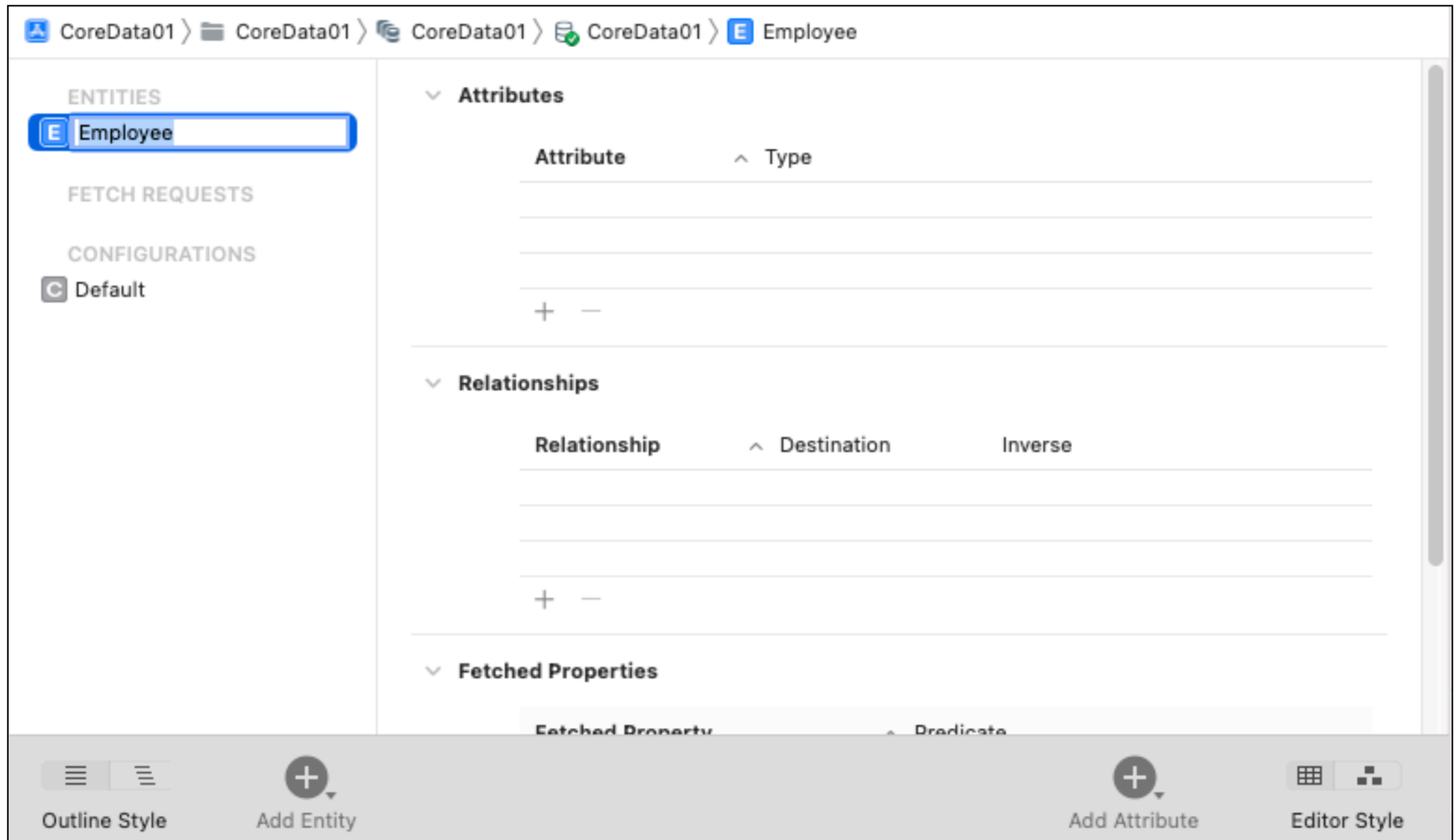
Untere Leiste:

- Add Entity
- Add Attribute

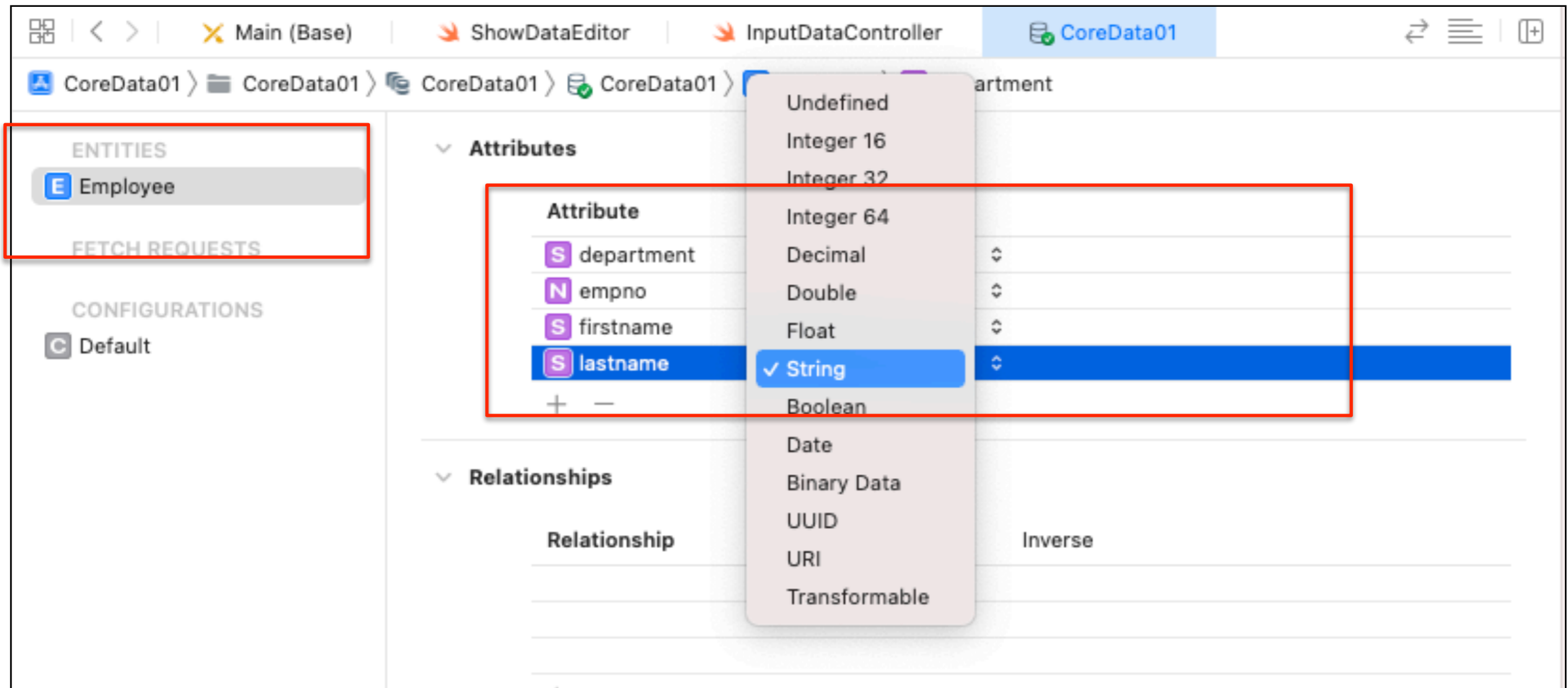
CoreData1:Datenbankmodell



CoreData1:Datenbankmodell: Table „Employee“



CoreData1:Datenbankmodell



CoreData1:Datenbankmodell



This is a screenshot of the "Attribute" configuration window in Xcode's CoreData editor. It shows the settings for the "firstname" attribute.

Attribute

Name:

Type: (dropdown menu)

☐ Optional ☐ Transient

☐ Derived

☐ Allows Cloud Encryption

Default Value: ☒ Default String

Validation: (dropdown) (dropdown)

Min Length Max Length

Reg. Ex.:

Advanced: ☐ Index in Spotlight

☐ Preserve After Deletion

Deprecated

Spotlight ☐ Store in External Record File

User Info

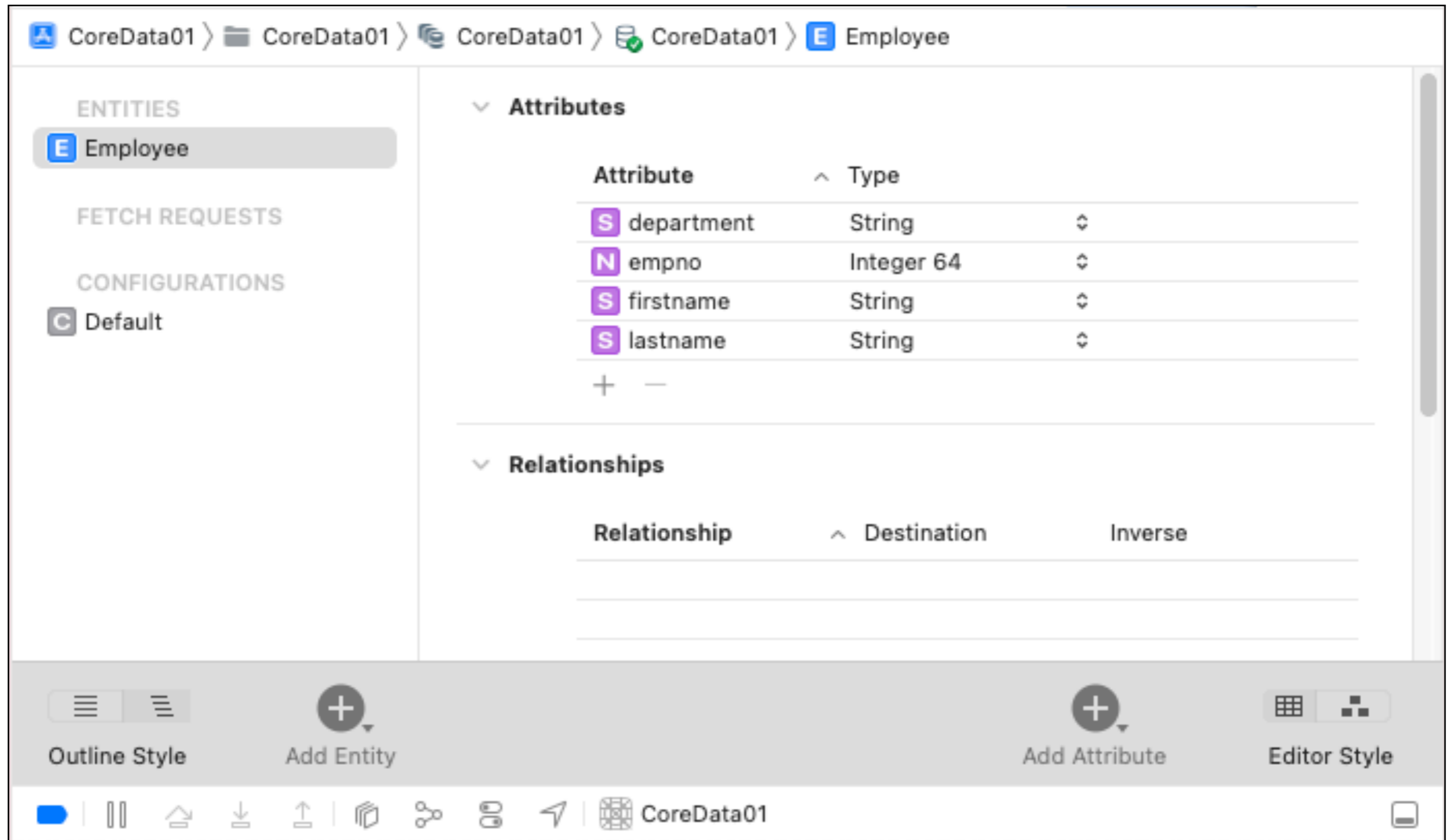
Key	Value

+ -

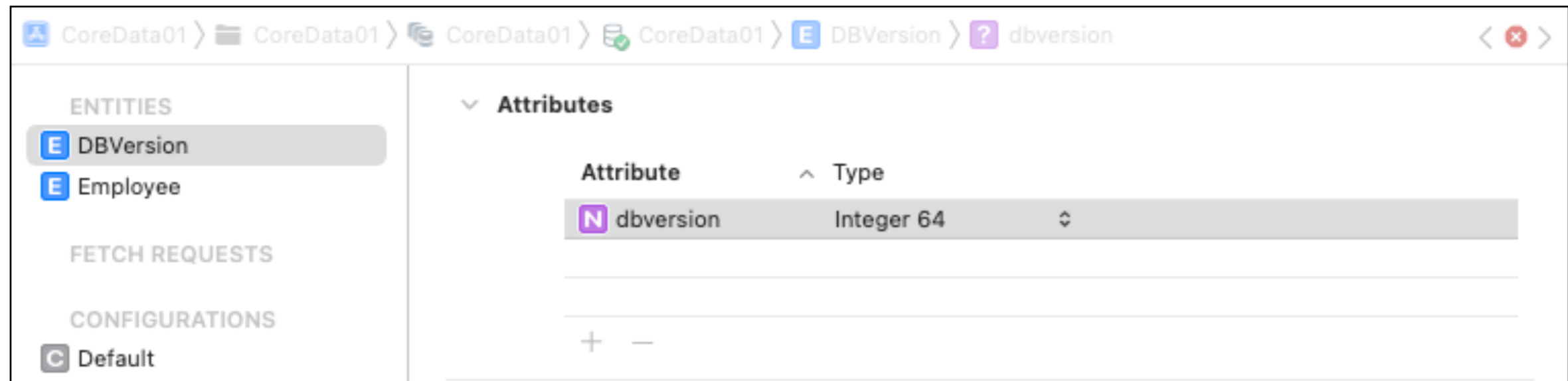
Wichtig

- Optional
- Default Value
- Min Length
- Max Length

CoreData1: Datenbankmodell Employee



CoreData1: Datenbankmodell DBVersion



Attribute

Name

Type ☒

☐ Optional ☐ Transient

☐ Derived

☐ Allows Cloud Encryption

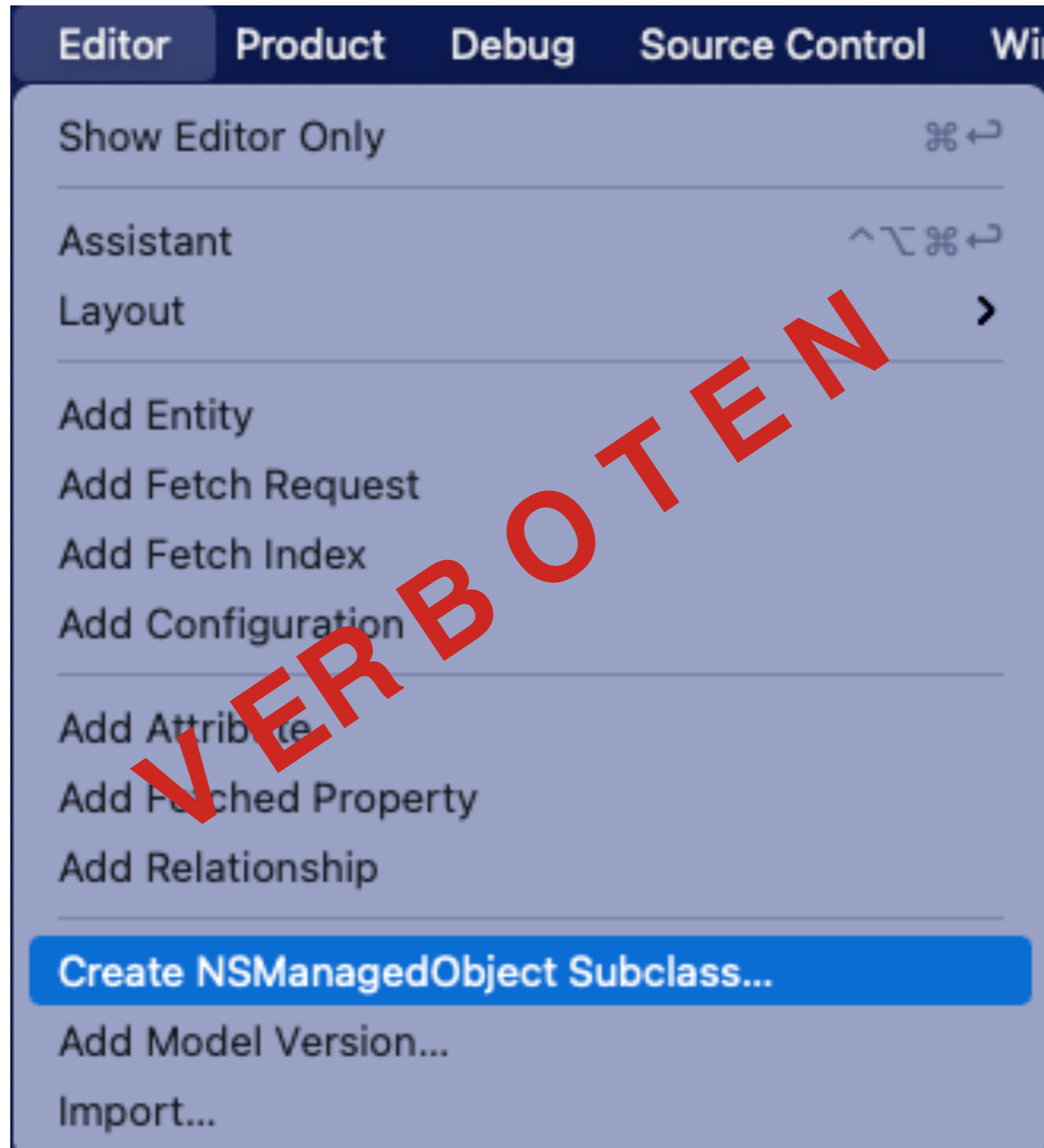
Default Value ☒

☒ Use Scalar Type

Validation ☒ ☒

Minimum Maximum

Erzeugen der Swift-Klassen aus dem Datenbankmodell



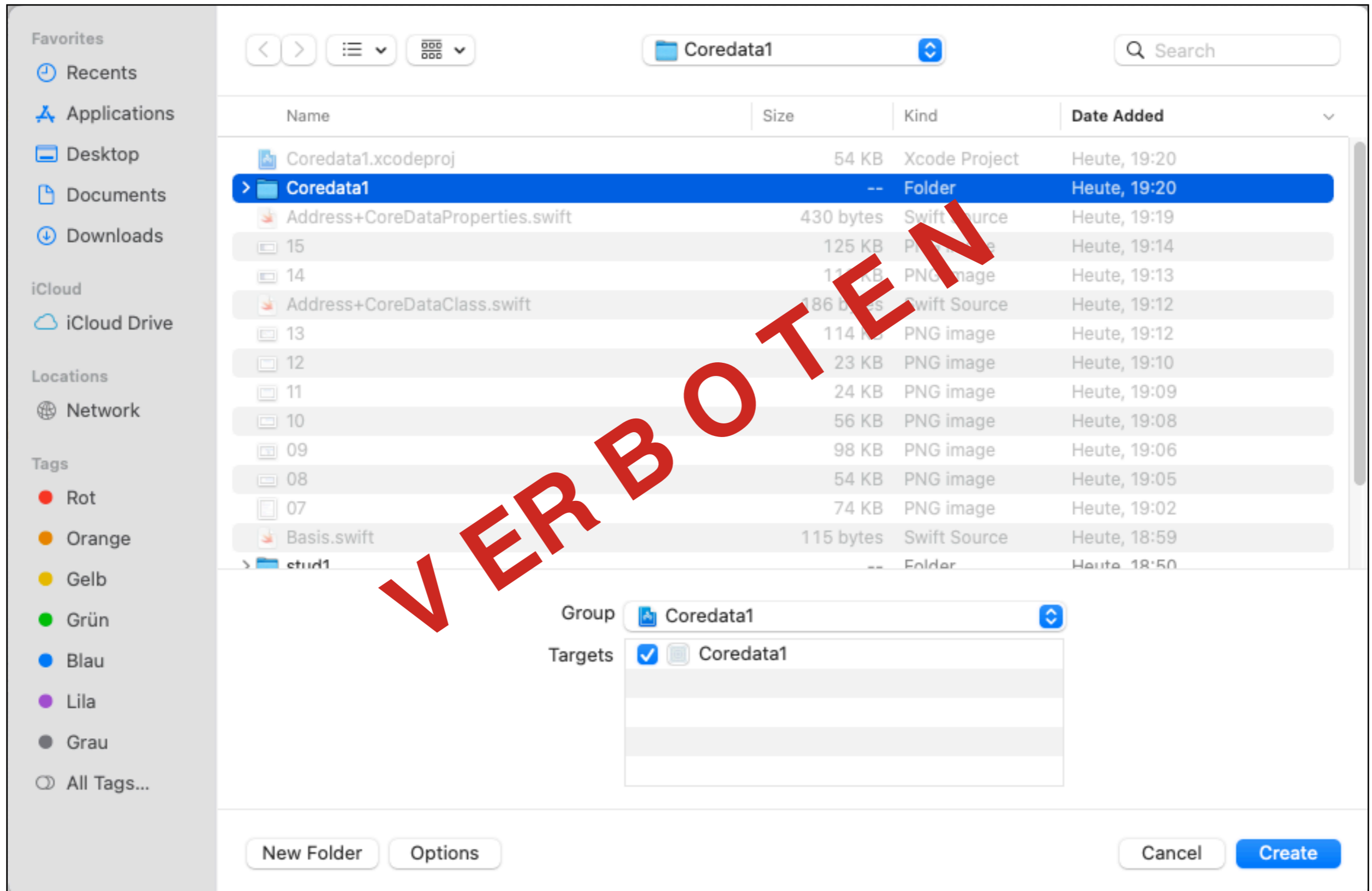
Alle gewünschten Tabellen anklicken

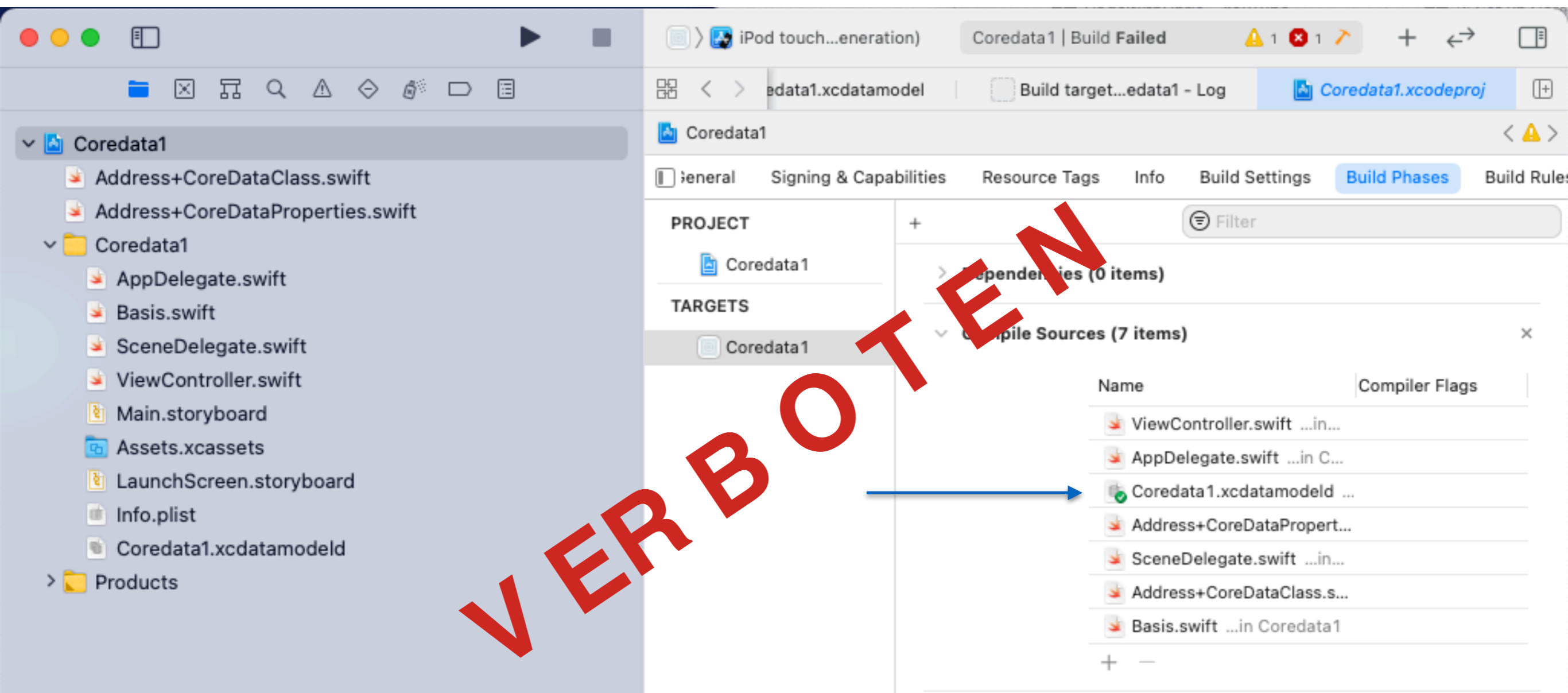
[illegible]

Alle gewünschten Tabellen anklicken

[illegible]

Speichern der neuen Dateien für die SQL-Tabelle





Nun den Eintrag „Coredata1“ löschen

<https://stackoverflow.com/questions/50718018/xcode-10-error-multiple-commands-produce>

Erstellen der Wrapper-Klassen (manuell)

Datei/Klasse: Employee

- Cmd N
- Cocoa Touch Class
- Class (Name): TBL_Employee (Employee ist schon vorhanden)
- SubClass: NSObject

Datei/Klasse: DBVersion

- Cmd N
- Cocoa Touch Class
- Class (Name): TBL_DBVersion
- SubClass: NSObject

Speichern der neuen Dateien für die SQL-Tabellen

```
import UIKit
```

```
class TBL_Employee: NSObject {  
    public var firstname: String = ""  
    public var lastname: String = ""  
    public var empno: Int64 = -1;  
    public var department: String = ""  
}
```

```
class TBL_BVersion: NSObject {  
    public var version: Int64 = -1;  
}
```

Speichern der neuen Dateien für die SQL-Tabellen

```
import UIKit
```

```
class TBL_Employee: NSObject {  
    public var database:NSManagedObject?  
    public var firstname: String = ""  
    public var lastname: String = ""  
    public var empno: Int64 = -1;  
    public var department: String = ""  
}
```

```
    init(firstname:String, lastname:String,  
        empno:Int64 ,department:String) {  
        self.firstname = firstname  
        self.lastname = lastname  
        self.empno = empno  
        self.department = department  
    }
```

Speichern der neuen Dateien für die SQL-Tabellen

```
// toString
override var description :String {
    return "Name: \$(lastname),\$(firstname)
           Number: \$(empno)   Dept: \$(department)"
}
```

Speichern der neuen Dateien für die SQL-Tabellen

```
import UIKit
```

```
class TBL_DBVersion: NSObject {
```

```
    public var database:NSManagedObject?
```

```
    public var version: Int64 = -1;
```

```
}
```

Erstellen der DBQueries-Klasse

Datei/Klasse: DBSQueries

- Cmd N
- Cocoa Touch Class
- Class (Name): DBSQueries
- SubClass: NSObject

Klasse DBSQueries

```
import UIKit

import CoreData

class DBSQueries: NSObject {

    private static let instance = DBSQueries();

    private override init() {

    }

    public static func getInstance ()->DBSQueries {
        return instance;
    }

    let TABLE_EMPLOYEE = "Employee"
    let TABLE_DBVERSION = "DBVersion"

    let FIRSTNAME = "firstname"
    let LASTNAME = "lastname"
    let EMPNO = "empno"
    let DEPARTMENT = "department"
```

Klasse DBSQueries

```
// NSManagedObjectContext
func getConnection()->NSManagedObjectContext?{
    guard let appDelegate = UIApplication.shared.delegate as? AppDelegate
    else {
        print("error")
        return nil
    }

    let context = appDelegate.persistentContainer.viewContext
    return context
}
```

Klasse DBSQueries

```
func insertEmployee(_ emp:TBL_Employee )->Bool {
    if let context = getConnection() {
        guard let tableEmployee = NSEntityDescription.entity(forEntityName:
TABLE_EMPLOYEE, in: context)
        else {
            return false
        }
        let newEmployee = NSManagedObject(entity: tableEmployee,
insertInto: context)

        newEmployee.setValue(emp.firstname, forKey: FIRSTNAME)
        newEmployee.setValue(emp.lastname, forKey: LASTNAME)
        newEmployee.setValue(emp.empno, forKey: EMPNO)
        newEmployee.setValue(emp.department, forKey: DEPARTMENT)

        // save Dataset
        do {
            try context.save()
            print("One employee are saved")
            return true
        } catch {
            print("save by error")
            print(error)
            return false
        }
    }
    else {
        return false
    }
} // insertEmployee
}
```

```

public func loadEmployees() -> Array<TBL_Employee> {
    var employees = [TBL_Employee]()
    if let context = getConnection() {
        let request = NSFetchRequest<NSFetchRequestResult>(entityName: TABLE_EMPLOYEE)

        do {
            let results = try context.fetch(request)
            for res in results {
                if let dbEmployee = res as? NSManagedObject {
                    guard let firstname = dbEmployee.value(forKey: FIRSTNAME) as? String else {
                        return employees
                    }
                    guard let lastname = dbEmployee.value(forKey: LASTNAME) as? String else {
                        return employees
                    }
                    guard let empno = dbEmployee.value(forKey: EMPNO) as? Int64 else {
                        return employees
                    }
                    guard let department = dbEmployee.value(forKey: DEPARTMENT) as? String else {
                        return employees
                    }

                    let emp = TBL_Employee(firstname: firstname, lastname: lastname,
                                           empno: empno, department: department)
                    emp.database = dbEmployee
                    employees.append(emp)
                }
            }
        } catch {
            print("load by error")
            print(error)
        }
    }
    //return employees
    return employees
}

```

Klasse DBSQueries

```
public func deleteEmployee(emp:NSManagedObject) -> Bool {  
    if let context = getConnection() {  
        context.delete(emp)  
        do {  
            try context.save()  
            return true  
        }  
        catch {  
            return false  
        }  
    }  
    else {  
        return false  
    }  
}
```

Klasse DBSQueries

```
public func updateEmployee(emp: NSManagedObject) -> Bool {  
    if let context = getConnection() {  
        do {  
            try context.save()  
            return true  
        }  
        catch {  
            return false  
        }  
    }  
    else {  
        return false  
    }  
}
```

CoreData1: InputDataController

Referenzen

```
@IBOutlet var tFirstname: UITextField!
```

```
@IBOutlet var tLastname: UITextField!
```

```
@IBOutlet var tEmployeeNo: UITextField!
```

```
@IBOutlet var tDepartment: UITextField!
```

CoreData1: InputDataController

```
@IBAction func bnInsertDataClick(_ sender: UIButton) {
    let firstname = tFirstname.text?.trimmingCharacters(
        in: .whitespacesAndNewlines)
    let lastname = tLastname.text?.trimmingCharacters(
        in: .whitespacesAndNewlines)
    let department = tDepartment.text?.trimmingCharacters(
        in: .whitespacesAndNewlines)
    let dummy = tEmployeeNo.text?.trimmingCharacters(
        in: .whitespacesAndNewlines)

    let empno = Int64(dummy!)
    if (empno==nil) {
        print("Fehlerhafte Employee Number")
        return
    }

    var emp = TBL_Employee(firstname: firstname!, lastname: lastname!,
        empno: empno!, department: department!)

    let dbsInstance = DBSQueries.getInstance()

    if !dbsInstance.insertEmployee(emp) {
        Basis.showOKDialog(parent: self, _title: "Fehler",
            _message: "Der Mitarbeiter konnte nicht eingefügt werden")
    }

} // bnInsertDataClick
```

ShowDataEditor Controller

- Label caption (T10, L10,R10)
- Hori-Stack (T10, L 5, R 5)
 - A. Fill Equally (Proportionally)
 - B. Button Anzeigen Fontsize: 15
 - C. Button Sort nach lastname
 - D. Filter Nmber>100
 - E. Filter Firstname like
- TextView:
 - F. tEditor (T10,L10,R10, B10)



ShowDataEditor Controller: Referenzen / Event

CoreData01 > CoreData01 > ShowDataEditor > No Selection

```
1 import UIKit
2
3 class ShowDataEditor: UIViewController {
4     @IBOutlet var tEditor: UITextView!
5
6     override func viewDidLoad() {
7         super.viewDidLoad()
8
9         tEditor.text = "Bitte auf den Schalter klicken"
10    }
11
12
13    @IBAction func bnShowDatasClick(_ sender: UIButton) {
14
15    }
16    @IBAction func bnShowDataSortlastnameClick(_ sender: UIButton) {
17    }
18
19    @IBAction func bnShowEmnpnoGT100Click(_ sender: Any) {
20    }
21    @IBAction func bnShowFilterLikeFirstnameClick(_ sender: Any) {
22    }
23
24 }
25
```

ShowDataEditor Controller: Methoden

```
@IBAction func bnShowDatasClick(_ sender: UIButton) {  
    let dbInstance = DBSQueries.getInstance()  
    let liste:Array<TBL_Employee> =  
        dbInstance.loadEmployees(sortListe: [],predicateListe: [])  
    var s=""  
    for emp in liste {  
        s+=emp.description+"\n"  
    }  
    tEditor.text = s  
}
```

ShowDataEditor Controller: Methoden

```
@IBAction func bnShowDataSortlastnameClick(_ sender: UIButton) {
    let dbsInstance = DBSQueries.getInstance()

    let sortByLastname = NSSortDescriptor.init(key: "lastname",ascending: true)
    let sortByFirstname = NSSortDescriptor.init(key: "firstname",ascending: true)

    let liste:Array<TBL_Employee> = dbsInstance.loadEmployees(
        sortListe: [sortByLastname,sortByFirstname],predicateListe: [])

    var s=""
    for emp in liste {
        s+=emp.description+"\n"
    }
    tEditor.text = s
}
```

ShowDataEditor Controller: Methoden

```
@IBAction func bnShowEmpnoGT100Click(_ sender: Any) {
    let dbsInstance = DBSQueries.getInstance()
    let predicateEmpnoGT100 = NSPredicate(format:"empno>%@", "100")

    let liste:Array<TBL_Employee> = dbsInstance.loadEmployees(
        sortListe: [],predicateListe: [predicateEmpnoGT100])

    var s=""
    for emp in liste {
        s+=emp.description+"\n"
    }
    tEditor.text = s
}
```

ShowDataEditor Controller: Methoden

Filter: firstname = ,Anna‘

```
@IBAction func bnShowFilterLikeFirstnameClick(_ sender: Any) {  
    let dbsInstance = DBSQueries.getInstance()  
    let predicateLikeFirstname1 = NSPredicate(  
        format:"firstname LIKE %@", "Anna")  
  
    let liste:Array<TBL_Employee> = dbsInstance.loadEmployees(  
        sortListe: [],predicateListe: [predicateLikeFirstname1])  
  
    var s=""  
    for emp in liste {  
        s+=emp.description+"\n"  
    }  
    tEditor.text = s  
}
```

ShowDataEditor Controller: Methoden

Filter: firstname contains 'Anna'

```
@IBAction func bnShowFilterLikeFirstnameClick(_ sender: Any) {  
    let dbInstance = DBSQueries.getInstance()  
  
    let predicateLikeFirstname2 = NSPredicate(  
        format:"firstname CONTAINS %@", "Anna")  
  
    let liste:Array<TBL_Employee> = dbInstance.loadEmployees(  
        sortListe: [],predicateListe: [predicateLikeFirstname2])  
  
    var s=""  
    for emp in liste {  
        s+=emp.description+"\n"  
    }  
    tEditor.text = s  
}
```

ShowDataEditor Controller: Methoden

Filter: firstname beginswith 'Anna'

```
@IBAction func bnShowFilterLikeFirstnameClick(_ sender: Any) {  
    let dbsInstance = DBSQueries.getInstance()  
  
    let predicateLikeFirstname3 = NSPredicate(  
        format:"firstname BEGINSWITH %@", "Anna")  
  
    let liste:Array<TBL_Employee> = dbsInstance.loadEmployees(  
        sortListe: [],predicateListe: [predicateLikeFirstname3])  
  
    var s=""  
    for emp in liste {  
        s+=emp.description+"\n"  
    }  
    tEditor.text = s  
}
```

ShowDataEditor Controller: Methoden

Filter: firstname =,Anna'

```
@IBAction func bnShowFilterLikeFirstnameClick(_ sender: Any) {  
    let dbsInstance = DBSQueries.getInstance()  
    let predicateLikeFirstname1 = NSPredicate(  
        format:"firstname LIKE %@", "Anna")  
  
    let liste:Array<TBL_Employee> = dbsInstance.loadEmployees(  
        sortListe: [],predicateListe: [predicateLikeFirstname1])  
  
    var s=""  
    for emp in liste {  
        s+=emp.description+"\n"  
    }  
    tEditor.text = s  
}
```

ShowDataEditor Controller: Methoden

Filter: firstname MATCHES '?nna*'

```
@IBAction func bnShowFilterLikeFirstnameClick(_ sender: Any) {  
    let dbsInstance = DBSQueries.getInstance()  
  
    let patternA="?nna*"
    let predicateLikeFirstname4a = NSPredicate(  
        format:"firstname LIKE %@",patternA)  
  
    let patternB="?nna*"
    let predicateLikeFirstname4c = NSPredicate(  
        format:"firstname MATCHES %@",patternC)  
  
    let liste:Array<TBL_Employee> = dbsInstance.loadEmployees(  
        sortListe: [],predicateListe: [predicateLikeFirstname1])  
  
    var s=""  
    for emp in liste {  
        s+=emp.description+"\n"  
    }  
    tEditor.text = s  
}
```