

Präsentation zur Übungsaufgabe 14b

Frances Bräutigam

Aufgabe

Aggregation der Daten aus einer gegebenen Datenbank, sodass zu den Einkäufen von

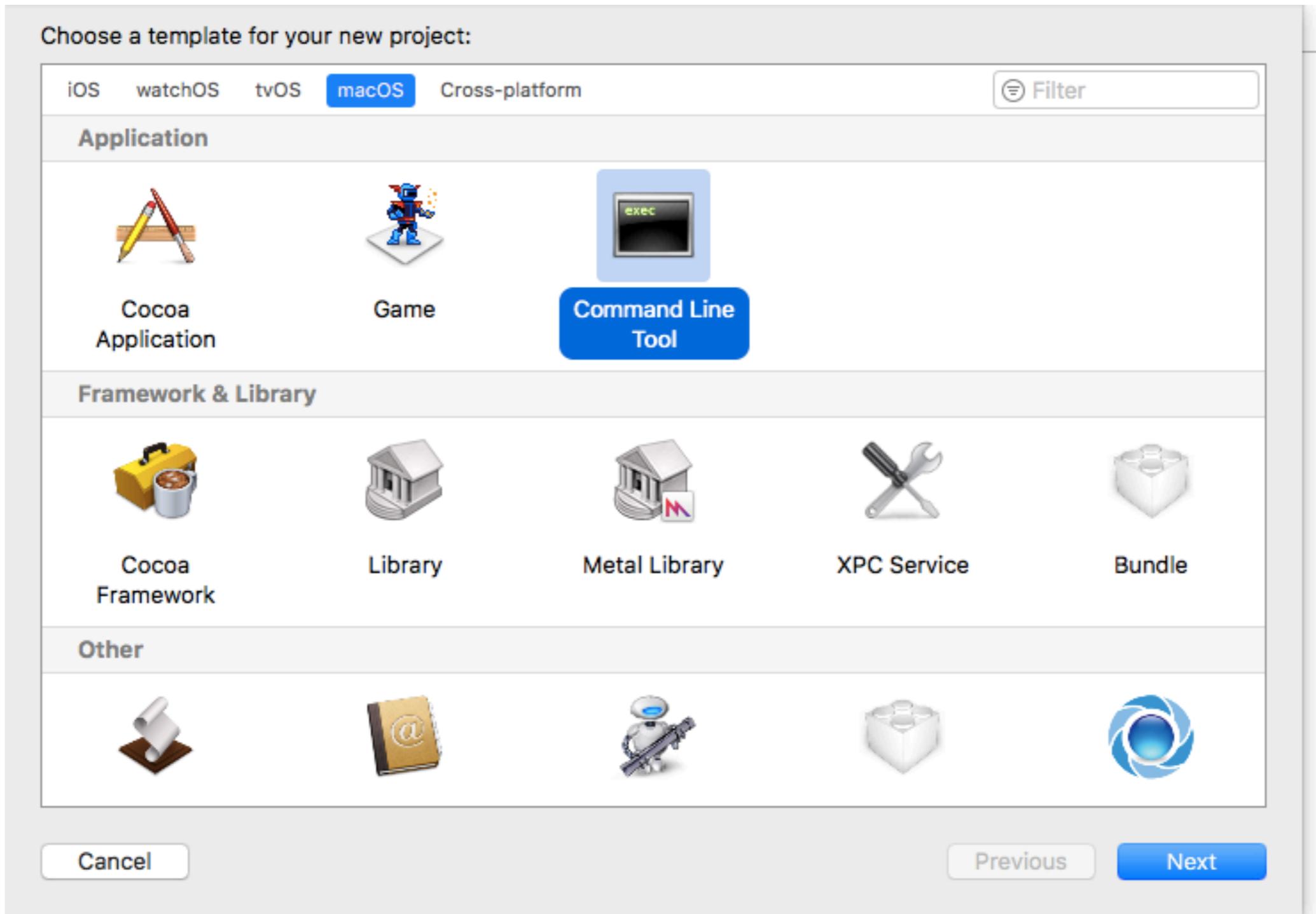
- Michelle Brooks aus New York,
- Lynn Zimmermann aus Frankfurt und
- Lucas Mancini aus Rom

folgende Metadaten ausgegeben werden können:

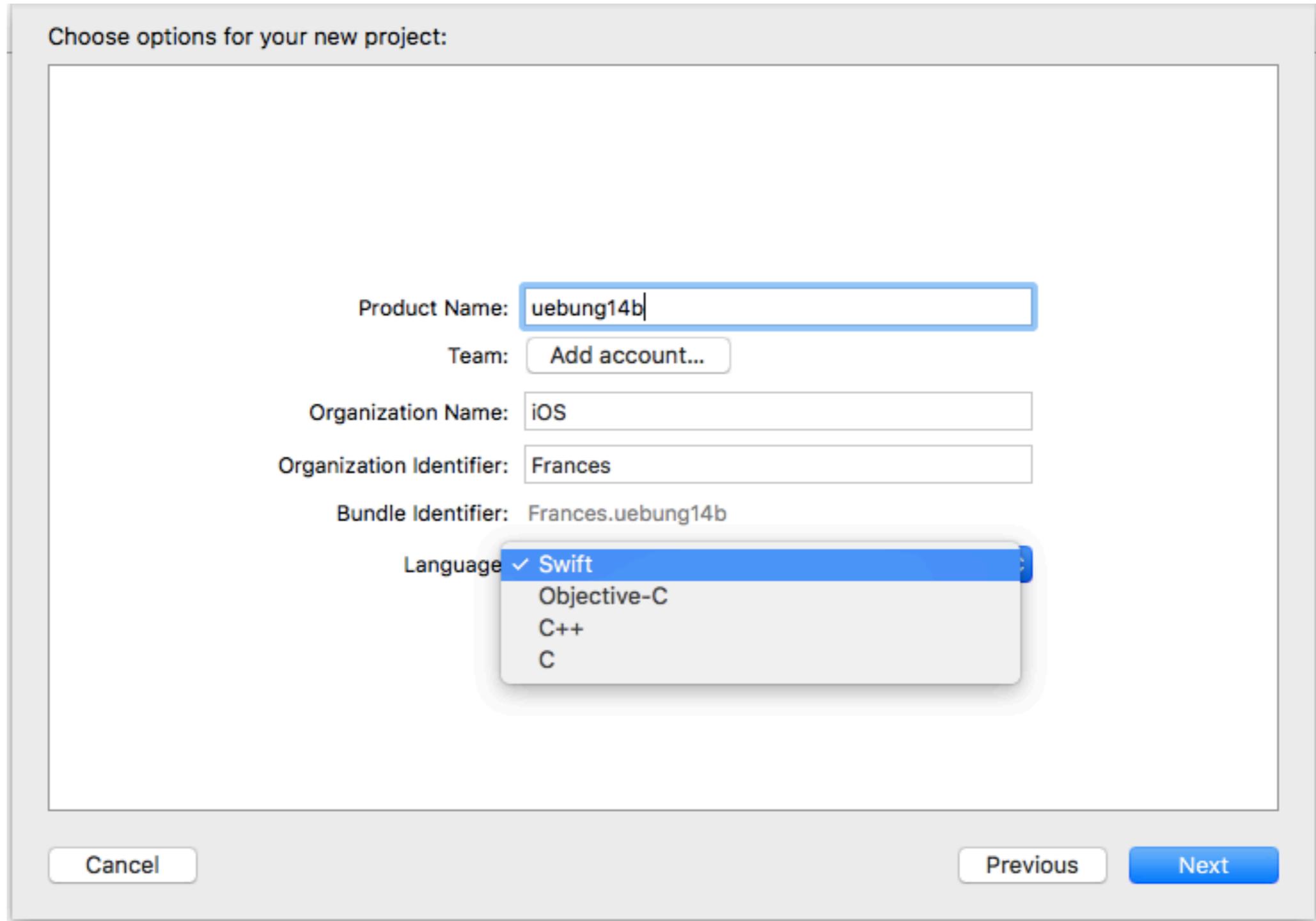
- Artist
- Track
- Album
- Genre
- Name des Medientyps

Aufgabe

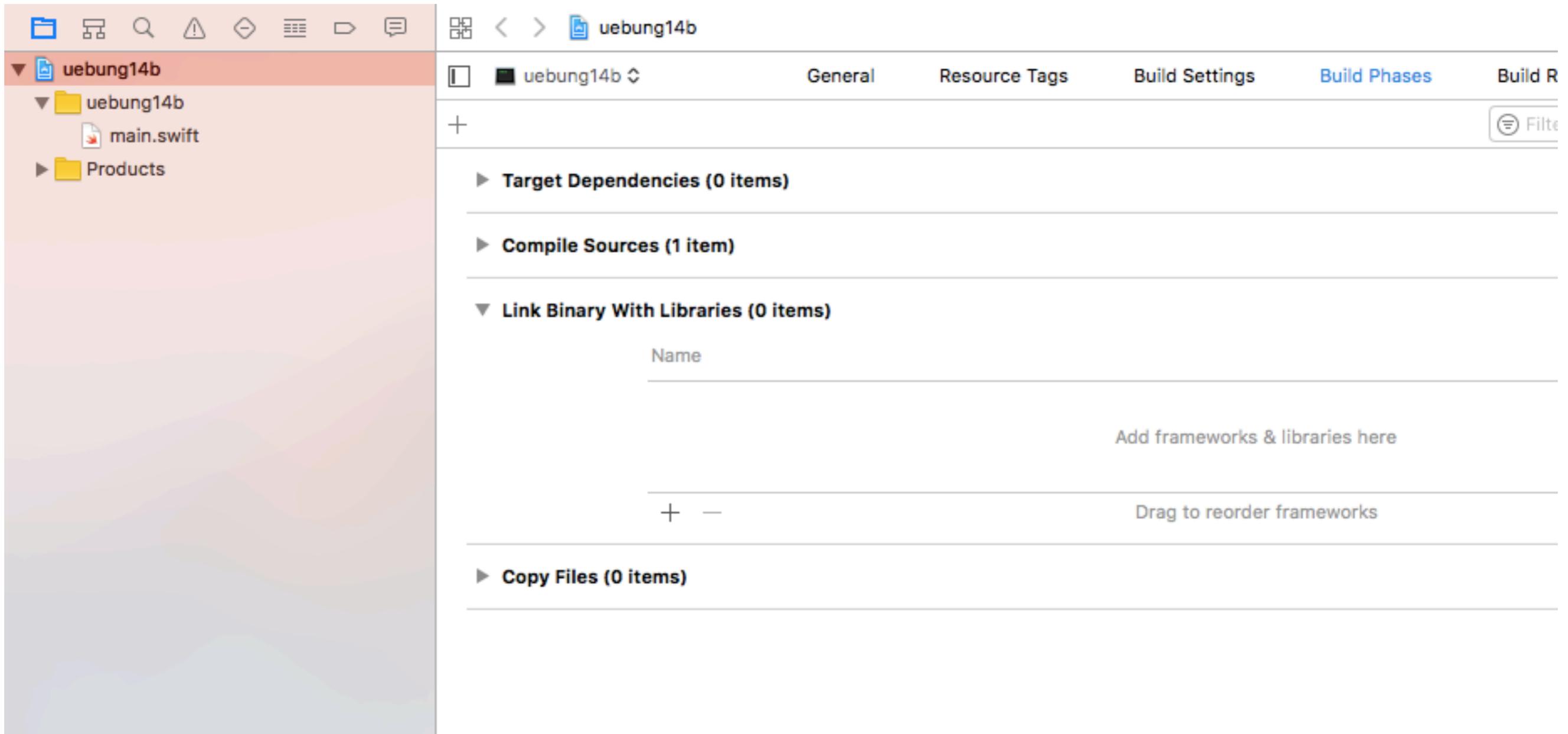
- Ausgabe in einer Tabelle auf der Konsole
- Arbeiten mit Swift
- Erklärung, ob und dann warum ein Wrapper (nicht) verwendet wurde und wie der Aufruf der sqlite3-Methoden sonst funktionieren würde



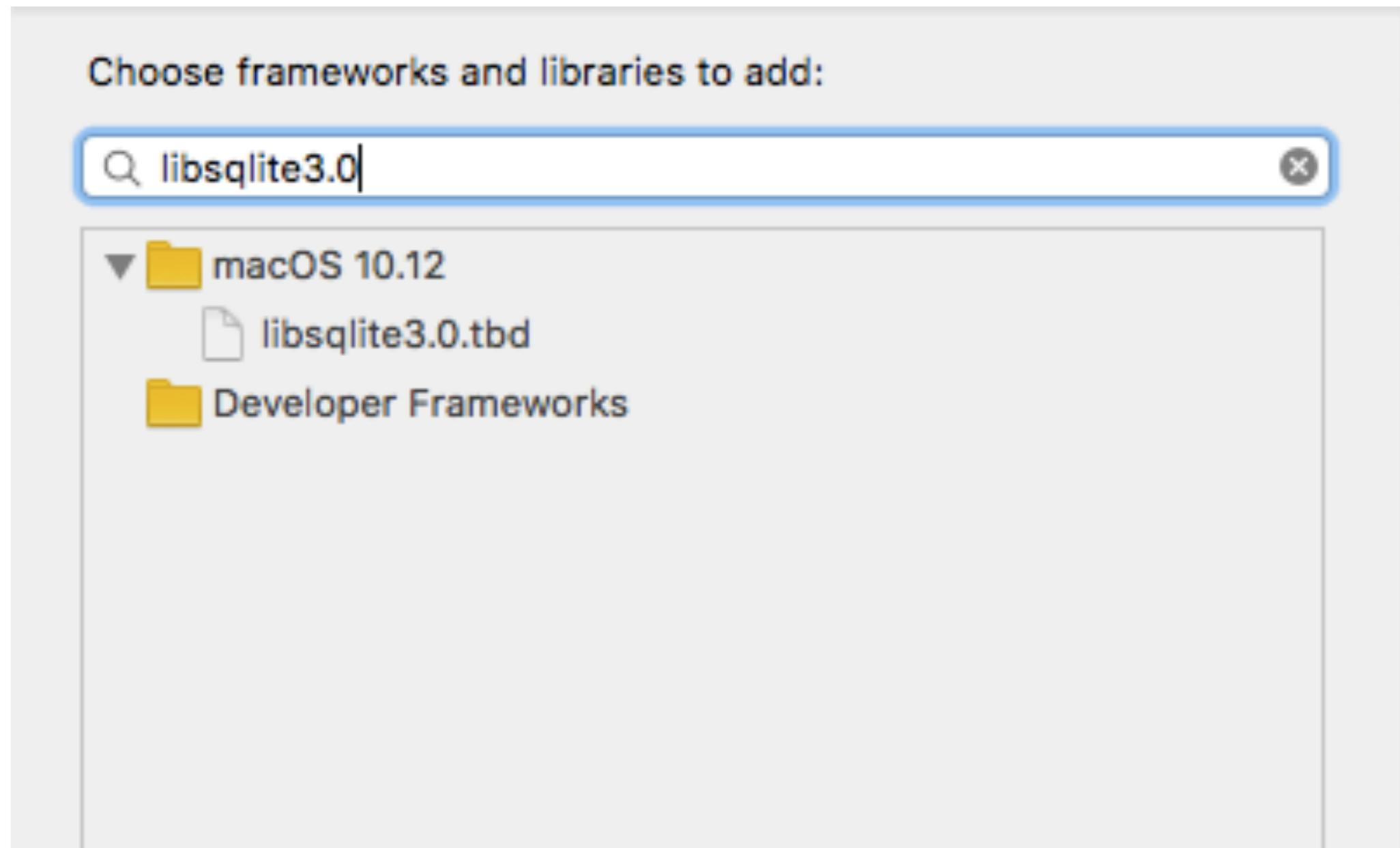
Neues Projekt erstellen —> Konsole



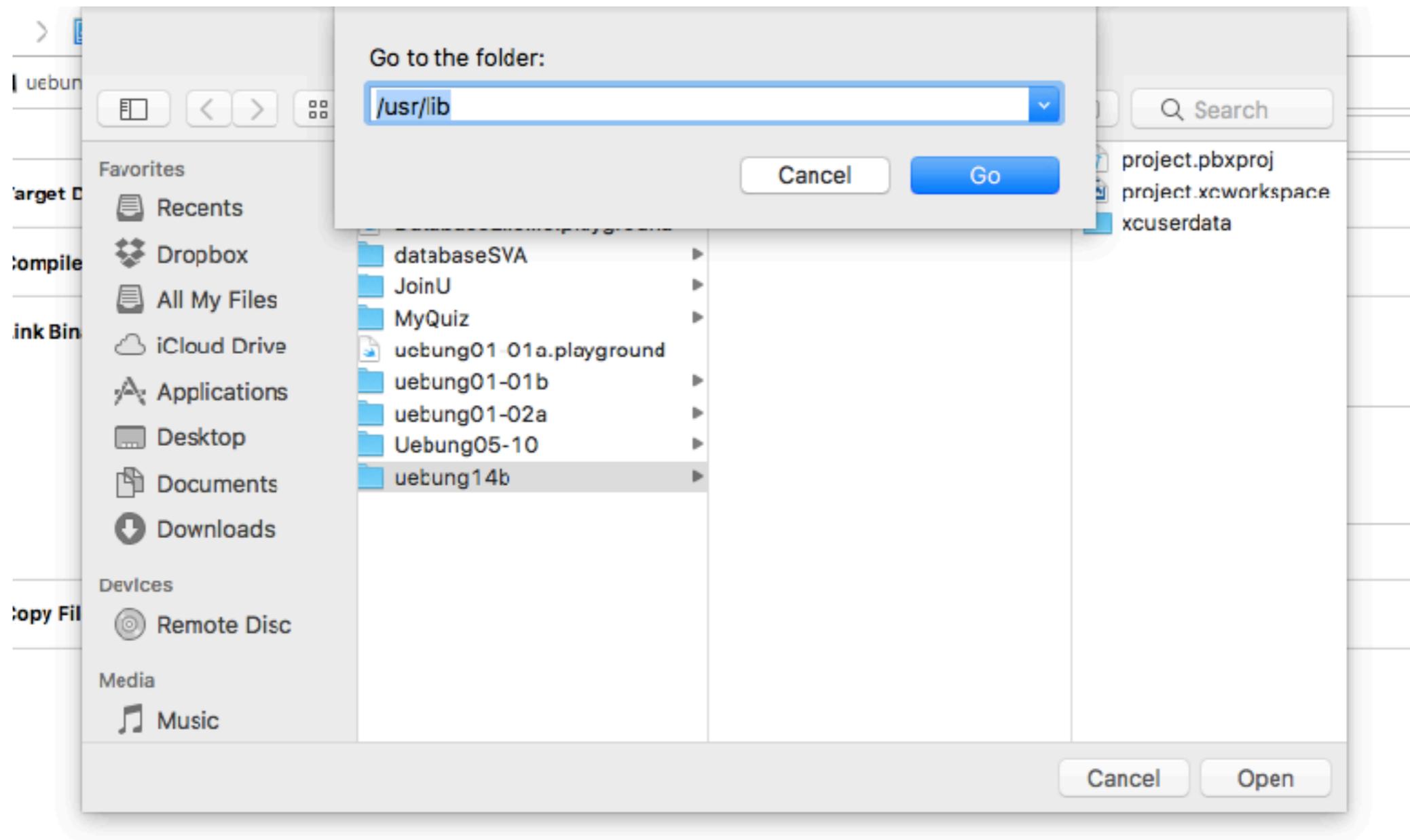
Sicherstellen, dass das Projekt in Swift erstellt wird



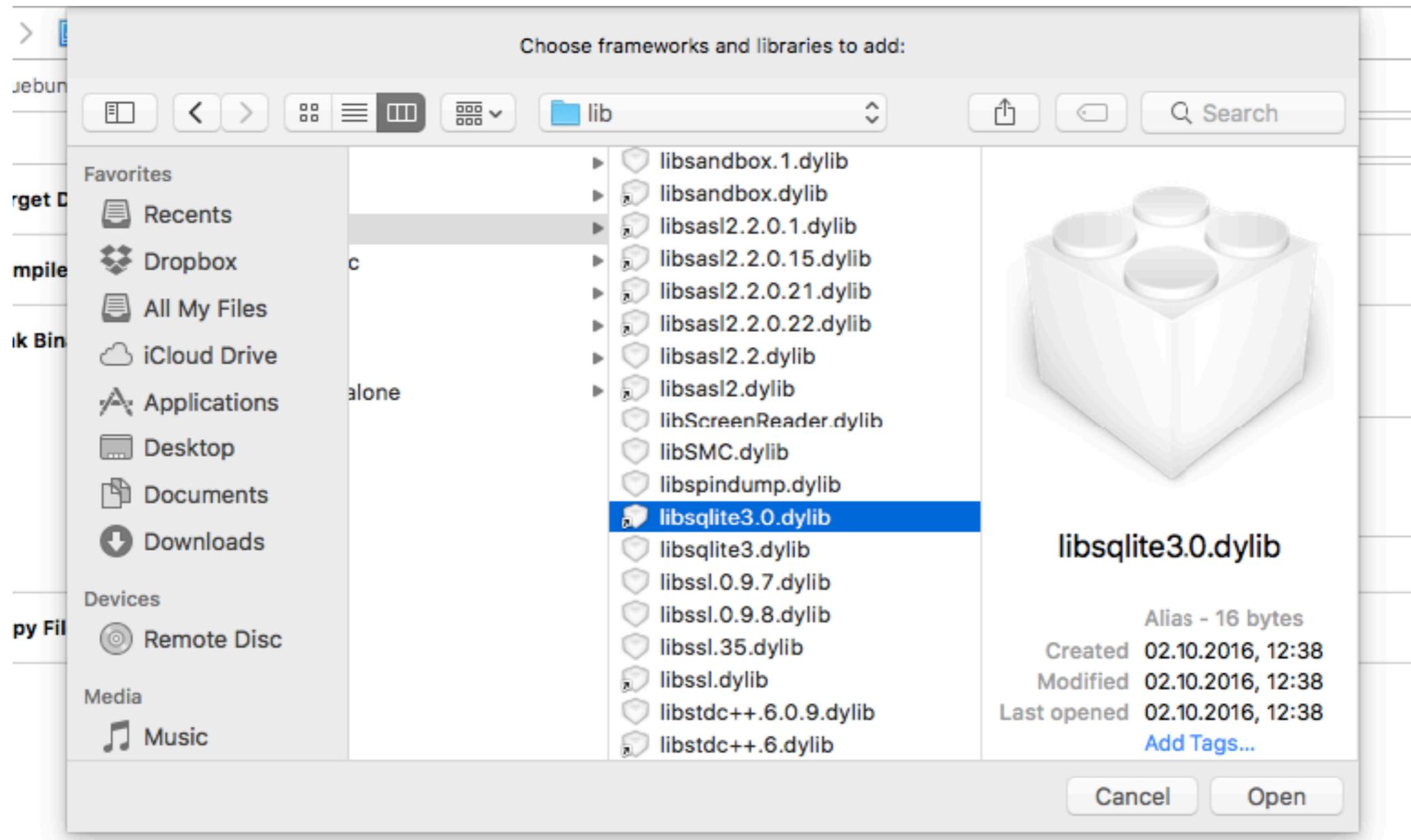
Hinzufügen der dynamischen SQLite Bibliothek: Auf das Plus gehen



Sollte die Bibliothek libsqlite3.0.**dylib** bereits vorhanden sein, auswählen und die nächsten 2 Schritte überspringen. Sonst auf Add other am linken unteren Rand klicken.



Mit `cmd + ctrl + G` öffnet sich der Suchbereich. Nach `/usr/lib` suchen.



Aus der geöffneten Liste die entsprechende Library auswählen. Open klicken.

g14b

General

Resource Tags

Build Settings

Build Phases

Build Rules

Filter

Dependencies (0 items)

Frameworks (1 item)

Libraries (1 item)

Name

Status

 libsqlite3.dylib

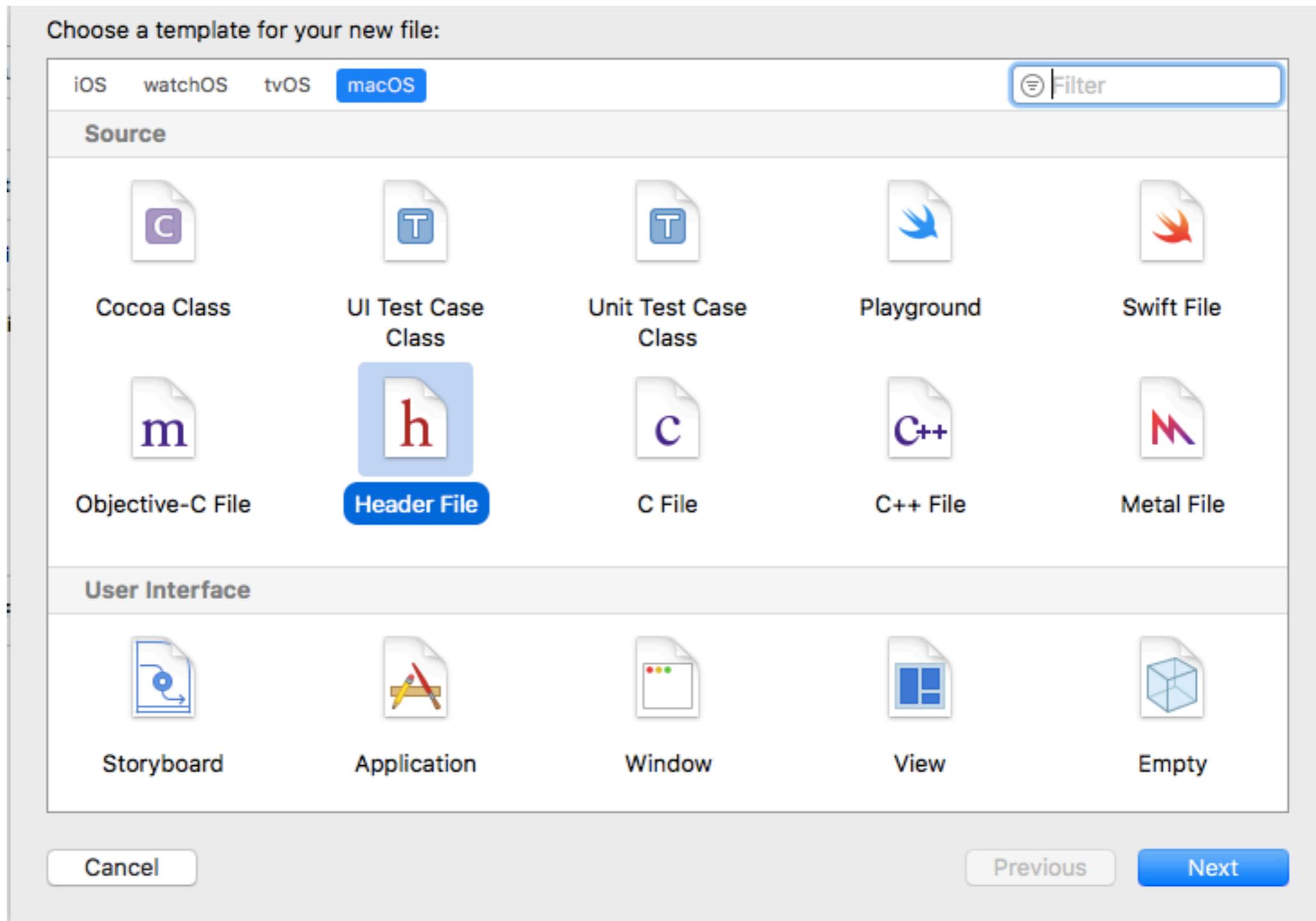
Resolved

+ -

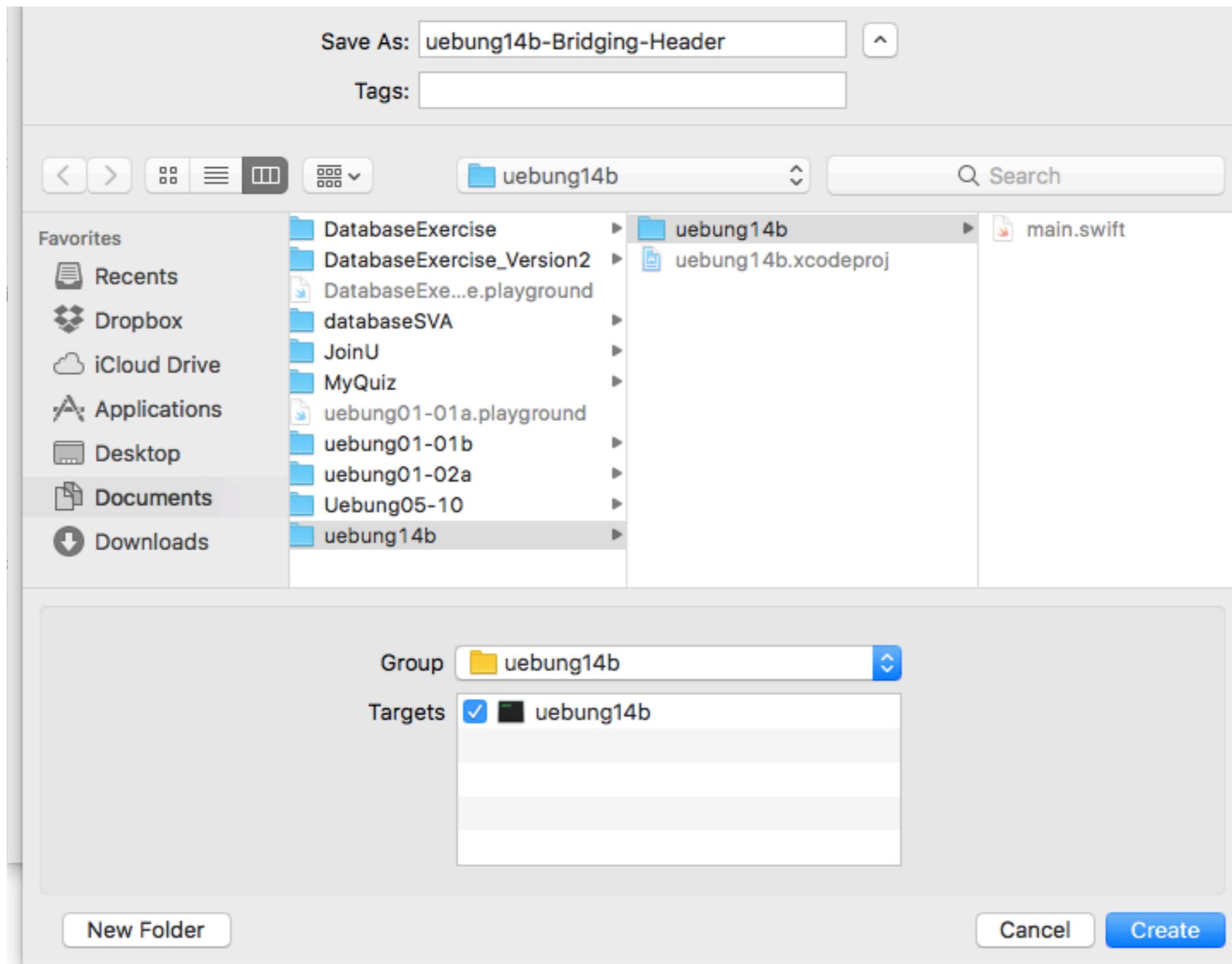
Drag to reorder frameworks

Frameworks (1 item)

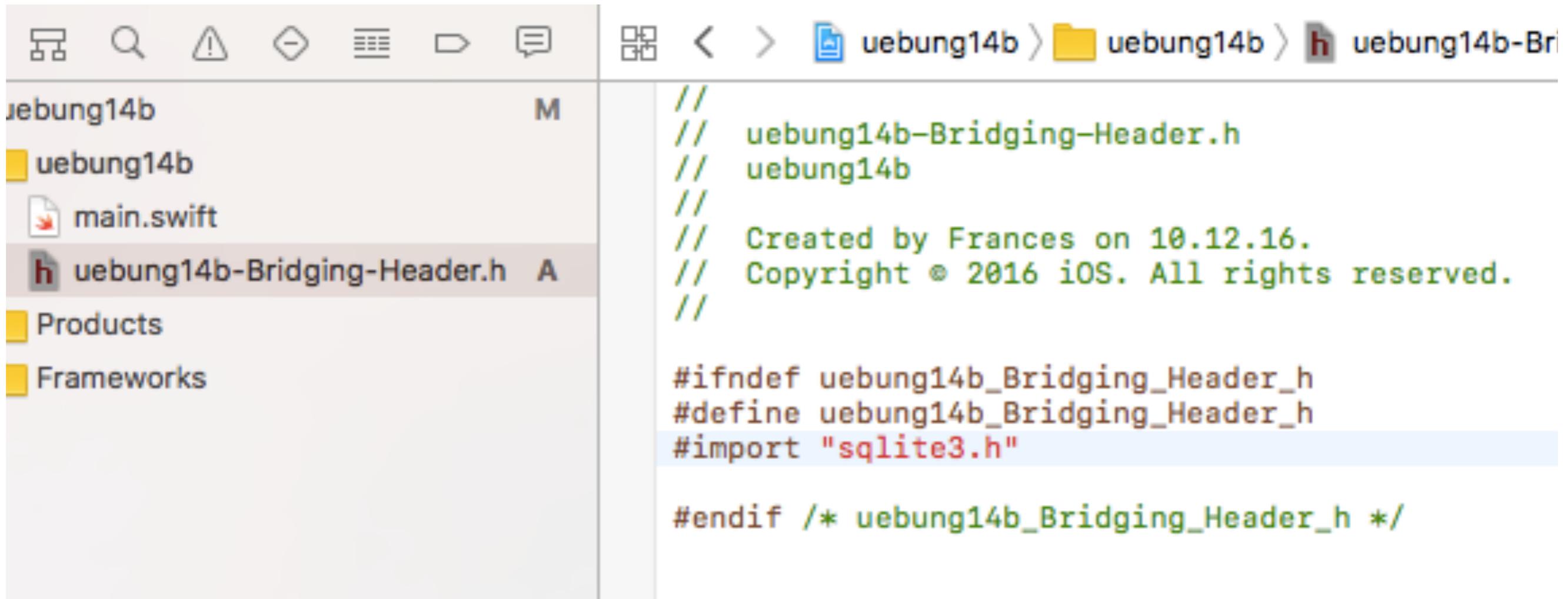
Die Library wurde hinzugefügt.



Neues header-File anlegen



Benennung: <Projektname>-Bridging-Header



```
//  
// uebung14b-Bridging-Header.h  
// uebung14b  
//  
// Created by Frances on 10.12.16.  
// Copyright © 2016 iOS. All rights reserved.  
//  
  
#ifndef uebung14b_Bridging_Header_h  
#define uebung14b_Bridging_Header_h  
#import "sqlite3.h"  
  
#endif /* uebung14b_Bridging_Header_h */
```

Import-Anweisung in der header-Datei ergänzen.
Hinweis: ggf. Müsst ihr unter Build Phase den Bridging
Header noch hinterlegen

```

import Foundation

let dbPath = try! FileManager.default.url(for: .documentDirectory, in: .userDomainMask, appropriateFor:nil, create: false) .
    appendingPathComponent("Chinook_Sqlite.sqlite")

func openDatabase() -> OpaquePointer? {
    var db: OpaquePointer? = nil
    if(sqlite3_open(dbPath.path, &db) == SQLITE_OK) {
        print("Successfully opened connection to database at \(dbPath)")
    } else {
        print("Unable to open database: \(String(cString: sqlite3_errmsg(db)))")
    }
    return db
}

let db = openDatabase()

```

Herstellen einer Verbindung zur Datenbank
(main.swift)

```

let firstname = "Michelle"
let city = "New York"

let customerStatementString = "SELECT c.FirstName, c.LastName, c.City FROM Customer c WHERE FirstName = '\(firstname)' AND
City = '\(city)';"

func customerQuery() -> Bool {
    var customerStatement: OpaquePointer? = nil
    // Auxiliary Bool.
    var booly = false

    if sqlite3_prepare_v2(db, customerStatementString, -1, &customerStatement, nil) == SQLITE_OK {

        while sqlite3_step(customerStatement) == SQLITE_ROW {

            let firstname = String(cString: sqlite3_column_text(customerStatement, 0))
            let lastname = String(cString: sqlite3_column_text(customerStatement, 1))
            let city = String(cString: sqlite3_column_text(customerStatement, 2))

            print("Metadaten für: \(firstname) \(lastname) aus \(city)")
            print("Artist \t\t| Track \t\t| Album \t\t| Genre")
        }
        // Wird nur auf true gesetzt, wenn alle Rows ausgelesen wurden.
        booly = true

    } else {
        print("SELECT statement could not be prepared: \(String(cString: sqlite3_errmsg(db)))")
    }

    sqlite3_finalize(customerStatement)
    return booly
}

while !customerQuery() {
    sleep(1)
}

```

CustomerQuery: Filtern, für welche Person die Abfrage erstellt wird (main.swift)

```

let queryStatementString = "SELECT t.Name, a.Title, art.Name, g.Name, mt.name FROM Customer c, Track t, Album a, Artist art,
Genre g, MediaType mt, InvoiceLine il, Invoice i WHERE FirstName = '\(firstname)' AND City = '\(city)' AND c.CustomerId
= i.CustomerId AND il.InvoiceId = i.InvoiceId AND t.TrackId = il.TrackId AND t.AlbumId = a.AlbumId AND t.GenreId =
g.GenreId AND t.MediaTypeId = mt.MediaTypeId;"

func metaQuery() -> Bool {
    var queryStatement: OpaquePointer? = nil
    // Auxiliary Bool.
    var booly = false

    if sqlite3_prepare_v2(db, queryStatementString, -1, &queryStatement, nil) == SQLITE_OK {

        while sqlite3_step(queryStatement) == SQLITE_ROW {
            let track = String(cString: sqlite3_column_text(queryStatement, 0))
            let album = String(cString: sqlite3_column_text(queryStatement, 1))
            let artist = String(cString: sqlite3_column_text(queryStatement, 2))
            let genre = String(cString: sqlite3_column_text(queryStatement, 3))
            //let mediaType = String(cString: sqlite3_column_text(queryStatement, 4))

            //var tracklength = track.characters.count
            let str = String(repeating: "=", count: 15)

            print("\(artist) \t | \(track) \t| \(album) \t | \(genre)")
            print("\(str) | \(str) | \(str) | \(str)")

        }
        // Wird nur auf true gesetzt, wenn alle Rows ausgelesen wurden.
        booly = true

    } else {
        print("SELECT statement could not be prepared: \(String(cString: sqlite3_errmsg(db)))")
    }

    sqlite3_finalize(queryStatement)
    return booly
}

while !metaQuery() {
    sleep(1)
}

```

metaQuery: Herausfinden der Metadaten für diese Person
(main.swift)

Wrapper

- Diese Aufgabe wurde in Swift OHNE Wrapper gelöst.
- Grundsätzlich ist es aber sinnvoll, einen Wrapper zu verwenden, da die C API dann besser aus Swift angesprochen werden kann.
- Bsp. für einen Wrapper ist FMDB

Worauf ihr noch achten müsst

- `sqlite3_open` gibt auch `SQLITE_OK` zurück, wenn keine Datenbank gefunden wurde. Es wird einfach eine (leere) Datenbank erzeugt. Die `sqlite`-Datei sollte am Besten im Documents-Ordner liegen, dann gibt es keine/weniger Schwierigkeiten.
- Ggf. müsst ihr die Zugriffsrechte für die Datenbank anpassen („`rwxr-xr-x`“ bzw. „`rwxr-xr-x@`“)