

# **Mobile Information Systems**

## **Aufgabe 4**

Fatih Baş  
s911285  
4. Januar 2021

Für die Übung soll eine Fixkosten App entwickelt werden. Für diese werden nur wenige Daten benötigt. Da einzelne Kosten gespeichert werden reicht es aus, diese zu benennen und den Belastungsbetrag zu Speichern. Für mehr Details für den Endbenutzer ist es sinnvoll noch ein Start -und Enddatum hinzuzufügen. In der Tabelle 1 werden die benötigten Daten angezeigt.

Fixkosten	Type
Name	String
Kosten	String
Start Datum	String
End Datum	String
Dauer	String

Tabelle 1: Daten

Diese Daten kann man in einer Klasse zusammenfassen und sollen persistent gespeichert werden.

#### Möglichkeiten Daten Persistent in IOS zu Speichern:

##### **1. UserDefaults:**

UserDefaults ist ein Interface zur UserDefaults Datenbank wo man Key – Value Paare persistent über das Starten und Schließen der App hinweg speichern kann. [1]

##### Vorteile:

- Sehr leichte Nutzung
- Daten stehen in einer normalen Textdatei als Tabelle

##### Nachteile:

- Je mehr Daten vorhanden sind, desto länger dauert das laden aus der Tabelle
- Kann nur die gängigen Datentypen speichern (Int, Double, Float, Bool, String, Array )
- Kann keine Objekte aus selbst erstellten Klassen speichern , dies kann man mit dem NSCoder machen

##### **2. NSCoder:**

NSCoder deklariert die Schnittstelle, die von konkreten Unterklassen verwendet wird, um Objekte und andere Werte zwischen dem Speicher und einem anderen Format zu übertragen. Diese Fähigkeit bildet die Grundlage für die Speicherung von Objekten und Daten auf einem Speichermedium und das Kopieren von Objekten und Datenelementen zwischen verschiedenen Prozessen oder Threads. [2]

### Vorteile:

- Sehr leichte Nutzung
- Daten stehen in einer normalen Textdatei als Tabelle
- Kann beliebige Objekte speichern

### Nachteile:

- Je mehr Daten vorhanden sind, desto länger dauert das laden aus der Tabelle

## 3. **CoreData:**

CoreData ist eine SQLite Datenbank und erstellt gleich zu Beginn eines Projektes ein Datenmodell und Helper Methoden. Mit CoreData lassen sich wesentlich mehr Daten effizient speichern und das Handling ist sehr einfach. [1]

Für die Übung wird die persistente Speicherung der Daten von der Tabelle 1 mit CoreData ausgeführt. Dafür werden folgende Objekte benötigt:

Das *NSManagedObjectModel* lädt und repräsentiert das im xcdatamodel definierte Datenmodell zur Laufzeit. [3]

Der *NSPersistentStoreCoordinator* verwaltet die Persistierung der Daten mittels einem *NSPersistentStore* für SQLite, Binary-Archiven oder In-Memory. Typischerweise wird ein SQLite-Persistent Store mit Speicherung im Dokumentenverzeichnis der Anwendung verwendet. [3]

Die Datenobjekte, die als *NSManagedObject* repräsentiert werden, werden zur Laufzeit in einem *NSManagedObjectContext* verwaltet. [3]

In Abbildung 1 wird der Aufbau von CoreData verdeutlicht.

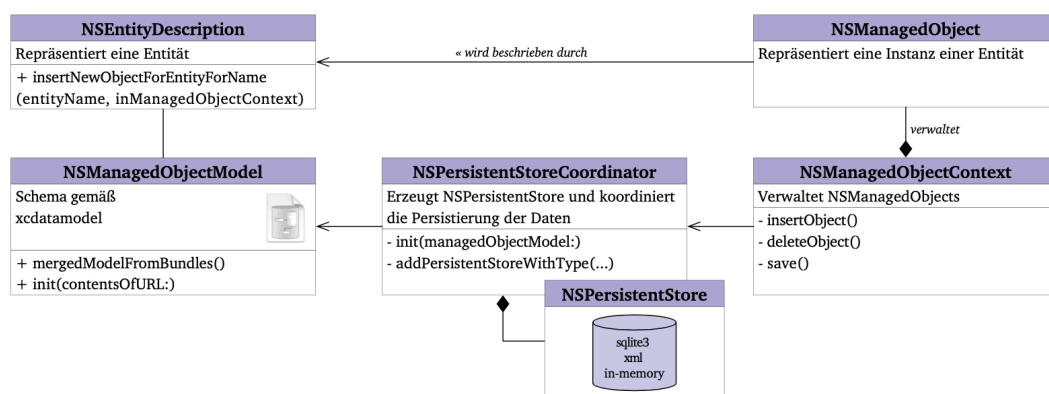


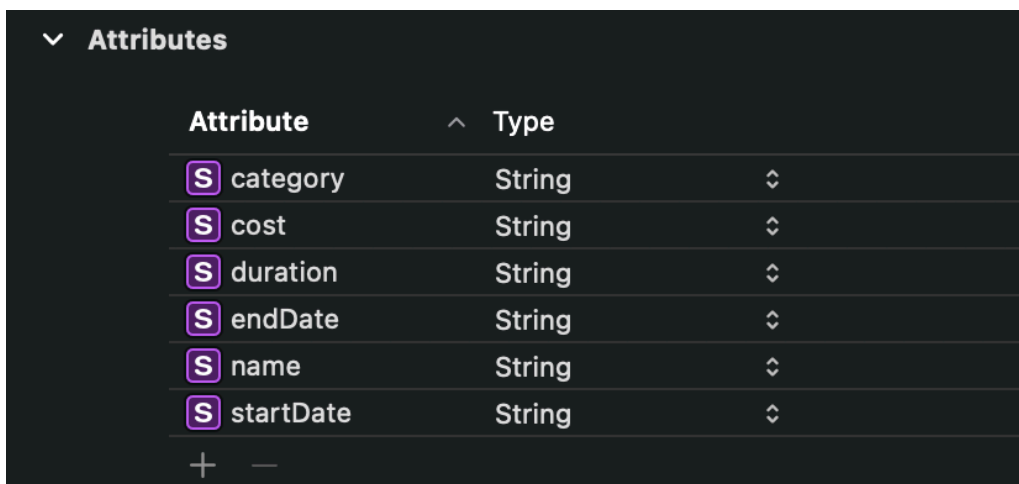
Abbildung 1: CoreData [3]

Damit die Fixkosten sauber angezeigt werden, wird das TableView von IOS verwendet. Dieser bietet die Möglichkeit dynamische Prototypen für Zellen zu konfigurieren. Dafür müssen diese von der *UITableViewCell* - Klasse erben [4]. Für die in Tabelle 1 dargestellten Daten wird deshalb die Klasse *CellItem* erstellt.

```
class CellItem: UITableViewCell {
    @IBOutlet weak var nameOfItem: UILabel!
    @IBOutlet weak var costOfItem: UILabel!
    @IBOutlet weak var startDateOfItem: UILabel!
    @IBOutlet weak var endDateOfItem: UILabel!
    @IBOutlet weak var durationOfItem: UILabel!
}
```

Somit können die benötigten Daten vom User aufgenommen werden. Damit diese nun persistent gespeichert werden muss zunächst ein Datenmodell erstellt werden. Dafür stellt CoreData eine .xcdatamodel - File zur Verfügung. In dieser werden die Entitäten beschrieben.

Entität: CellItemData















Attributes		
Attribute	Type	
 category	String	
 cost	String	
 duration	String	
 endDate	String	
 name	String	
 startDate	String	
+ -		

Abbildung 2: Entitäten

Für das Handeln der Daten wird eine Klasse *CoreDataHandler* erstellt. In dieser Klasse werden die CoreData - Objekte implementiert um die zu Speichernden Daten zu bearbeiten.

```

class CoraDataHandler {
    var carCosts = [CellItemData] ()
    var phoneCosts = [CellItemData] ()
    var homeCosts = [CellItemData] ()
    var insuranceCosts = [CellItemData] ()
    var investCosts = [CellItemData] ()

    var context: NSManagedObjectContext!
    init();
    add();
    insert();
    delete();
    save();
    ...
}

```

#### Quellen:

- [1] <https://mfg.fhstp.ac.at/allgemein/persistentes-lokales-speichern-von-daten-in-swift/>
- [2] <https://developer.apple.com/documentation/foundation/nscoder>
- [3] <https://www.ralfebert.de/ios/coredata/>
- [4] <https://www.ralfebert.de/ios/uitableviewController/>

#### Implementierung:

<https://www.raywenderlich.com/7569-getting-started-with-core-data-tutorial>