Realm. Памятка

Подключение к проекту:

1. в папке проекта через `Terminal` выполнить команду:

```
pod init
```

2. созданный файл `Podfile` открыть для редактирования и внести следующий код:

```
platform :ios, ‘9.0’
use_frameworks!

target ‘Project_name’ do
    pod 'RealmSwift', '~> 0.98'
end
```

3. в папке проекта через `Terminal` выполнить команду:

```
pod install
```

4. для работы с проектом запустить файл “`Project_name.xcworkspace`”.
5. для подключения framework используется команда:

```
import RealmSwift.
```

Термины. Определения:

1. утилита Realm Browser позволяет читать и редактировать базы данных Realm.

2. экземпляр **Realm** - это сердце фреймворка, это точка доступа к основной базе данных, аналогично **управляемому контексту объекта** в Core Data. Создается при помощи инициализатора Realm().

3. **Object (Объект)** - это модель Realm. Процесс создания модели определяет **схему базы данных**. Сначала создается подкласс Object и затем определяются его fields (поля), которые будут храниться аналогично свойствам класса.

4. **Relationships (Связи или отношения)** - это связь с типом one-to-many (один ко многим) между объектами. Обычное свойство типа Object, на которое сохраняется ссылка. Поддерживаются типы связи многие-к-одному и многие-ко-многим через свойство типа List.

5. **Write Transactions (Запись операций)** - любые операции в базе данных, такие как создание, редактирование или удаление объектов, выполняются в рамках операций writes, которые происходят с помощью вызова метода write(_:) у экземляра Realm.

6. **Queries (Запросы)** - предназначены для извлечения объектов из базы данных. Самая простая форма запроса - это вызов метода objects() у экземпляра Realm с входным параметром типа Object, по которому совершается выборка из базы. Для создания более сложных поисковых запросов используются предикаты (фильтры) и дескрипторы сортировки результатов поиска.

7. **Results (Результаты)** - это автоматически обновляемый тип контейнера, получаемый в ответ на объектные запросы. У них много общего с обычными массивами, в том числе синтаксис сабскрипта для захвата элемента в индексе.

Создание первой модели:

1. Создайте новый файл объекта при помощи шаблона `iOS -> Swift File`.

2. Пример реализации класса Specimen:

```
import Foundation
import RealmSwift

class Specimen: Object {
     dynamic var name = ""
     dynamic var specimenDescription = ""
     dynamic var latitude = 0.0
     dynamic var longitude = 0.0
     dynamic var created = NSDate()
}
```

3. Конкретные типы данных `Realm` должны быть инициализированы с начальным значением.

4. Добавьте следующее объявление ниже других свойств:

```
dynamic var category: Category!
```

5. Это устанавливает связь один-ко-многим между `Specimen` и `Category`. Это означает, что каждый `Specimen` может принадлежать только к одной `Category`, но каждая `Category` может иметь много `Specimen`.

Добавление записей:

1. Сначала необходимо импортировать фрейворк `RealmSwift` в исходный файл контроллера:

```
import RealmSwift
```

2. Создать экземпляр Realm:

```
let realm = try! Realm()
```

3. Открыть транзакцию к `realm`:

```
try! realm.write {
}
```

4. Создать экземпляр управляемого объекта и модифицировать его свойства:

```
let newSpecimen = Specimen()
newSpecimen.name = self.nameTextField.text
newSpecimen.category = self.selectedCategory
```

```
    newSpecimen.specimenDescription =
self.descriptionTextField.text
    newSpecimen.latitude =
self.selectedAnnotation.coordinate.latitude
     newSpecimen.longitude =
self.selectedAnnotation.coordinate.longitude
```

5. Добавить объект в `realm`:

```
    realm.add(newSpecimen)
```

Работа с Realm Browser

1. Для быстрого поиска песочницы любого проекта, запущенного на `Simulator`, используйте бесплатную программу `OpenSim`.

2. Для просмотра файлов Realm, необходимо устанвоить программу Realm Browser.

3. После того, как база данных будет открыта в `Realm браузере`, вы увидите `Category` с цифрой `5` напротив. Это означает, что класс содержит пять записей.

Получение записей (запрос данных):

1. Сначала необходимо импортировать фрейворк `RealmSwift` в исходный файл контроллера:

```
import RealmSwift
```

2. У созданного экземпляра `Realm` получить значение свойства `objects`:

```
try! Realm().objects(Managed_Object_Name)
```

3. П. 2 вернет “массив значений” типа `Results<(Managed_Object_Name)>`:

```
var specimens = try! Realm().objects(Specimen)
```

4. Для прохода по всем значениям из полученного массива необходимо организовать аналогичный цикл итераций:

```
for specimen in specimens {
}
```

Получение записей (запрос данных) с сортировкой:

1. Пример выборки объектов `Specimen` с одновременной предварительной сортировкой по свойству `name`:

```
var specimens = try!
Realm().objects(Specimen).sorted("name", ascending: true)
```

Получение записей (запрос данных) с применением предикатов:

1. Необходимо создать предикат (фильтр) с условием выборки:

```
let predicate = NSPredicate(format: "name BEGINSWITH [c] %@", searchString)
```

2. Это пример условия, при котором фильтр накладывается на значения в поле `name`, которые начинаются `BEGINSWITH` с `searchString`.`[c]` без учета регистра:

```
searchResults = realm.objects(Specimen).filter(predicate).sorted("name", ascending: true)
```

Обновление записей:

1. Сначала необходимо импортировать фрейворк `RealmSwift` в исходный файл контроллера:

```
import RealmSwift
```

2. Затем необходимо создать экземпляр `Realm`:

```
let realm = try! Realm()
```

3. Манипуляции, связанные с обновлением свойств управляемого объекта необходимо производить внутри активированной транзакции `write()` следующим образом:

```
    try! realm.write {
        specimen.name = nameTextField.text!
        specimen.category = selectedCategory
        specimen.specimenDescription =
descriptionTextField.text
  }
```