Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Факультет компьютерных систем и сетей
Кафедра информатики

Курсовой проект по дисциплине:
«Программирование»

Пояснительная записка к курсовой работе

Тема работы:
«Приложение с голосовыми заметками на iOS Notevox»

Исполнитель
студент гр. 653501 _____ Линник Г.И.
(подпись дата)

Руководитель
_____ Козуб В.Н.
(подпись дата)

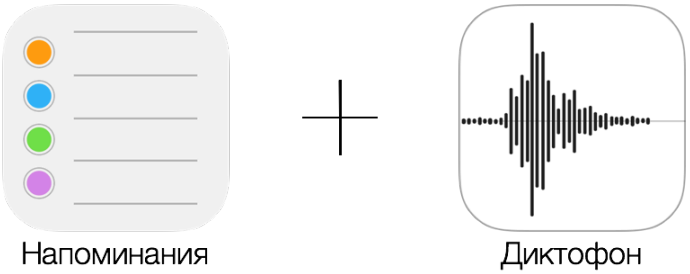
_____
(оценка)

Минск
2017 год

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Поводом к написанию данной курсовой работы стала идея совмещения стандартных приложений «Напоминания» и «Диктофон» в мобильной операционной системе iOS . Важнейшими из требований к современным программным продуктам являются удобство их использования, способность улучшать нашу повседневную жизнь. Также особое внимание в обществе уделяется лицам с ограниченными возможностями, например, имеющими дефекты зрения. Разработанное приложение позволяет на столько просто записать напоминание, как просто поднести телефон к уху. Это экономит время и делает нашу жизнь лучше.



ЦЕЛЬ РАБОТЫ

Целью работы является создание мобильного приложения для операционной системы iOS по записыванию голосовых заметок для автомобилистов, поэтов, людей с ограниченными возможностями, тех, кому неудобно записывать и тратить много времени на написание заметок вручную, а так же публикация в магазинах приложений.

ЗАДАЧИ, РЕШЕННЫЕ ДЛЯ ДОСТИЖЕНИЯ ПОСТАВЛЕННЫХ ЦЕЛЕЙ

Для реализации поставленных целей были решены следующие задачи:

- Изучены языки программирования Objective-C и Swift
- Изучен фреймворк Core Data для работы с базами данных на iOS
- Изучен фреймворк AV Foundation для записи и воспроизведения речевых сообщений.
- Изучена система контроля зависимостей для Swift и Objective-C CocoaPods
- Изучен шаблон проектирования Model-View-Controller, а так же его недостатки в рамках разработки приложений под iOS.
- Проведен анализ магазина приложений AppStore с целью поиска аналогичных программных продуктов.

ПОИСК АНАЛОГОВ В APPSTORE

Перед началом разработки был исследован магазин приложений AppStore. Он исследовался с помощью обычного поиска по ключевым словам, которые могли бы соответствовать мною запрашиваемому функционалу, а именно: напоминания, заметки и запись голоса (reminders, notes, voice reminders, voice notes). В результате поиска конечно же были выявлено множество аналогов, но все они были низкокачественные и у многих отсутствовала ключевая функция воспроизведения записи голоса в самом системном уведомлении о напоминании. Примеры результатов поиска приведены ниже (рис. 1):

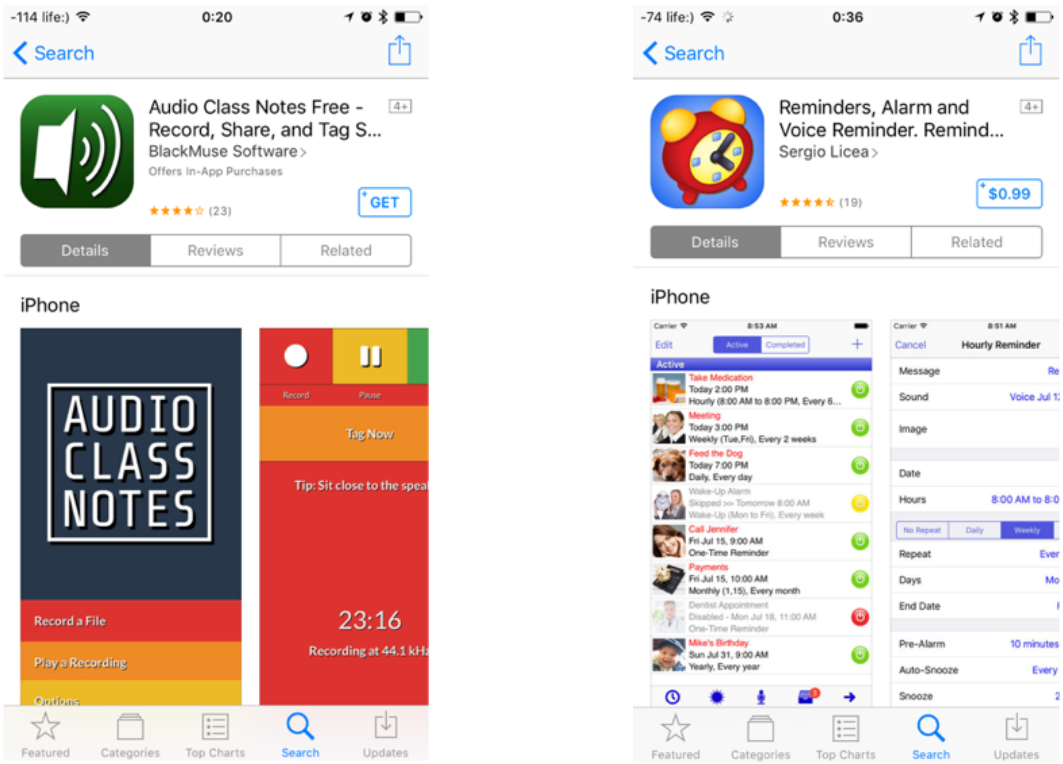


Рисунок 1 — Результаты поиска

Все найденные приложения не отличались хорошим дизайном и пользовательским интерфейсом, а ведь в настоящее время это самая важная часть приложения, потому что при первом знакомстве с программой, пользователь конечно же смотрит на графический интерфейс, и если он ему не понравился, пользователь может сразу же закрыть приложения, даже не смотря на может быть очень продвинутый функционал. Поле исследования результатов поиска, я решил, что данный сегмент достаточно свободный и в нем не со ставит большого труда занять лидирующие позиции, учитывая, что даже у таких гигантов в сфере заметок, как Evernote, тоже отсутствует функция воспроизведения голоса в напоминании. После этого было принято решение о создании собственного приложения с нужным мне функционалом.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ. ДИЗАЙН

Описание работы мобильного приложения

1. Пользователь открывает мобильное приложение и зажимает кнопку записи (рис. 2) или же просто подносит телефон к уху.

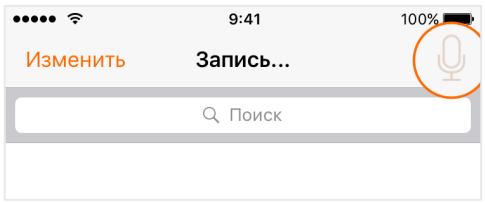


Рисунок 2 — Кнопка записи

2. После этого начинается запись голоса до тех пор, пока пользователь не отпустит кнопку записи или же не отведет телефон от уха.
3. Сразу же после этого, если длительность записи составляет больше 0.5 сек, запись автоматически сохраняется в память телефона со стандартным именем «Без названия» и без

Рисунок 3 — Новая заметка

установленной даты для напоминания (рис. 3).

4. Дальше у пользователя есть возможность изменить название голосовой заметки, установить дату, когда нужно напомнить, выбрать нужно ли воспроизводить записанный голос в уведомлении, если дата напоминания установлена (рис. 4).

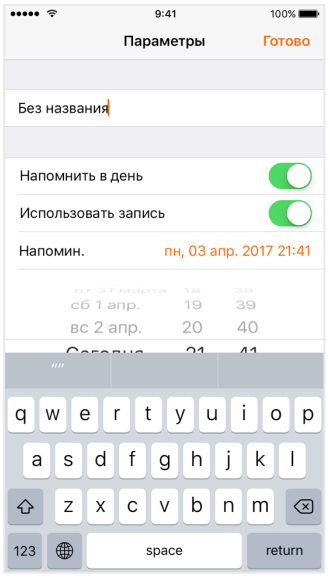


Рисунок 4 — Редактирование заметки

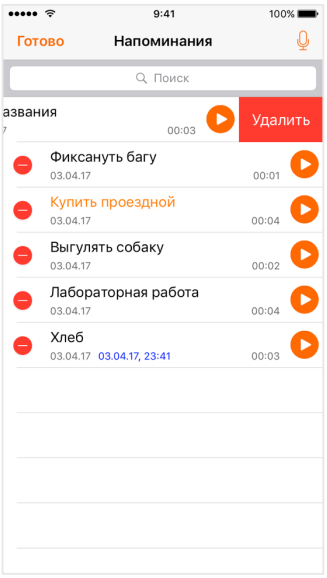


Рисунок 5 — Редактирование списка заметок

5. Также в приложении есть функция редактирования списка заметок и поиска по ним, и возможность выделения важных заметок оранжевым цветом (рис. 5).
6. В разработке используется Xcode 8 с iOS 10 SDK. Для написания проекта используется язык Objective-C. Используется менеджер зависимостей CocoaPods для более удобной установки сторонних библиотек.
7. Для записи голоса используется стандартная библиотека «AVFoundation».
8. Из сторонних библиотек используется «MGSwipeTableCell» для нестандартных ячеек таблицы, т.к. UIKit не предоставляет ячейки с нужным функционалом. MGSwipeTableCell (рис. 6) позволило создать ячейки с поддержкой удобного жеста смахивания вправо

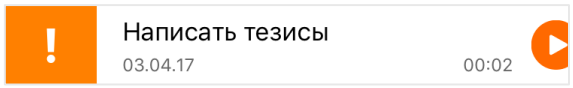


Рисунок 6 — MGSwipeTableCell

для отметки важных заметок.

Иконка приложения

Ещё, стоит отметить, что специально для проекта был разработан оригинальный логотип. Сначала долгое время использовался логотип на рисунке 7.1, но через какое-то время было принято решение переделать логотип, так как он недостаточно передавал смысл приложения. Вскоре пришла идея использовать микрофон на фоне оранжевого круга с графиком амплитуды звуковой волны (рис. 7.2). Но оказалось, что график амплитуды создавал ненужный «шум» на заднем фоне и иконка все так же плохо передавала основной смысл приложения. Тогда на основе этого логотипа был сделан логотип на рисунке 7.3. Здесь уже избавились от «шумных» деталей и пришли к использованию карандаша, который символизирует заметки и что в них можно что-нибудь записать, в качестве ножки микрофона. Данное решение показалось достаточно интересным и поэтому именно этот логотип решили использовать как иконку приложения.


Рисунок 6.1


Рисунок 6.2


Рисунок 6.3

ТЕХНИЧЕСКИЕ АСПЕКТЫ

Objective-C

Как было написано выше, в разработке используется язык программирования Objective-C.

Objective-C — компилируемый объектно-ориентированный язык программирования, используемый корпорацией Apple, построенный на основе языка Си и парадигм Smalltalk. В частности, объектная модель построена в стиле Smalltalk — то есть объектам посылаются сообщения.

Язык Objective-C является надмножеством языка Си, поэтому Си-код полностью понятен компилятору Objective-C.

Компилятор Objective-C входит в GCC и доступен на большинстве основных платформ. Язык используется в первую очередь для Mac OS X (Cocoa) и GNUstep — реализаций объектно-ориентированного интерфейса OpenStep. Также язык используется для iOS (Cocoa Touch).

В начале 1980-х годов было популярно структурное программирование, позволяющее разделить алгоритм на небольшие блоки. Однако, с ростом сложности задач, структурное программирование приводило к снижению качества кода. Приходилось писать всё больше функций, которые очень редко могли использоваться в других программах.

Многие увидели в объектно-ориентированном программировании потенциальное решение возникшей проблемы. С одной стороны, Smalltalk использовали почти все более-менее сложные системы. С другой — использование виртуальных машин повышало требования к ресурсам.

Objective-C был создан Брэдом Коксом в начале 1980-х в его компании Stepstone. Он пытался решить проблему повторного использования кода.

Целью Кокса было создание языка, поддерживающего концепцию software IC, подразумевающей возможность собирать программы из готовых компонентов (объектов), подобно тому как сложные электронные устройства могут быть собраны из набора готовых интегральных микросхем.

При этом язык должен быть простым и основанным на языке C, чтобы облегчить переход разработчиков на него.

Одной из целей было также создание модели, в которой сами классы являются полноценными объектами, поддерживалась бы интроспекция и динамическая обработка сообщений.

Objective-C является расширением C: любая программа на C является программой на Objective-C.

Одной из отличительных черт Objective-C является динамичность: решения, обычно принимаемые на этапе компиляции, здесь откладываются до этапа выполнения.

Objective-C — message-oriented-язык, в то время как C++ — function-oriented: в Objective-C вызовы метода интерпретируются не как вызов функции (хотя к этому обычно все сводится), а как посылка сообщения (с именем и аргументами) объекту, подобно тому, как это происходит в Smalltalk.

Любому объекту можно послать любое сообщение. Объект может вместо обработки сообщения переслать его другому объекту для обработки (делегирование), в частности, так можно реализовать распределённые (то есть находящиеся в различных адресных пространствах и даже на разных компьютерах) объекты.

Привязка сообщения к соответствующей функции происходит на этапе выполнения.

Язык Objective-C поддерживает работу с метаинформацией — так, на этапе выполнения можно узнать класс объекта, список его методов (с типами передаваемых аргументов) и instance-переменных, проверить, является ли класс потомком заданного и поддерживает ли он заданный протокол и т. п.

В языке есть поддержка протоколов (понятия интерфейса объекта и протокола четко разделены). Поддерживается наследование (не множественное); для протоколов поддерживается множественное наследование. Объект может быть унаследован от другого объекта и сразу нескольких протоколов (хотя это скорее не наследование протокола, а его поддержка).

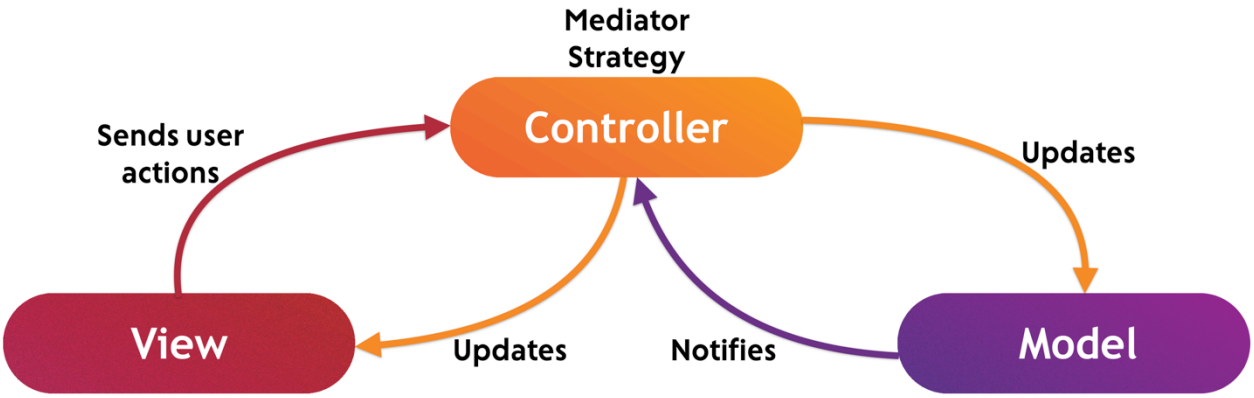
На данный момент язык Objective-C поддерживается компиляторами Clang и GCC (под управлением Windows используется в составе MinGW или cygwin).

Некоторые функции языка перенесены в runtime-библиотеку и сильно зависят от неё. Вместе с компилятором gcc поставляется минимальный вариант такой библиотеки. Также можно свободно скачать runtime-библиотеку компании Apple: Apple’s Objective-C runtime.

Эти две runtime-библиотеки похожи (основные отличия в именах методов). Далее примеры будут ориентироваться на runtime-библиотеку Apple.

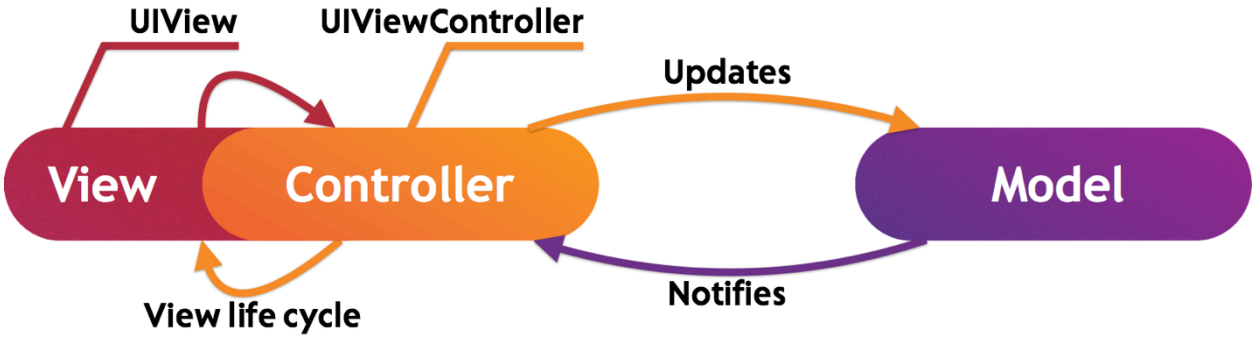
Model-View-Controller

Базовый архитектурный шаблон iOS-приложений, предлагаемый Apple, — это Model-View-Controller, который, что очевидно, — трехслойный. При его использовании все объекты приложения в зависимости от своего назначения принадлежат одному из трех слоев: модели (Model), представлению (View) или управлению (Controller). Каждый архитектурный слой отделен от другого абстрактными границами, через которые осуществляется связь между объектами разных слоев. Основная цель применения этой архитектурной концепции состоит в отделении бизнес-логики (модели) от её визуализации (представления).



Согласно рекомендациям Apple, ядром слоя управления в iOS-приложении выступают контроллеры представления (view controllers). Каждый такой объект отвечает за:

- управление иерархией представлений;
- адаптацию размеров представлений к определяемому устройством пространству отображения;
- обновление содержимого представлений в ответ на изменение данных;
- обработку пользовательского ввода и передачу полученных данных в модельный слой;
- освобождение связанных ресурсов при нехватке доступной оперативной памяти.



Apple даёт некоторые советы по выстраиванию архитектуры приложения, используя MVC, но они слишком общие и неконкретные. Следование этим советам ведет к разрастанию контроллеров представления до неимоверных размеров. Контроллер представления может начать делать не только вышеперечисленные функции, но и выполнять запросы на сервер, формировать запросы в базу данных, планировать локальные уведомления, записывать голос и т.д. То есть то, что совсем выходит за рамки обязанностей контроллеров представления. У разработчиков такое явление получило шутливое название Massive- или Mega-View-Controller (массивный или мегаконтроллер представления). Минусы такой архитектуры очевидны.

- **Высокая сложность поддержки и развития.** И действительно, достаточно легко будет что-либо сломать, из-за сложный и неочевидных потоков данных.
- **Высокий порог вхождения.** Если новый разработчик присоединится к разработке, ему будет достаточно сложно и долго найти нужный метод.
- **Код слабо тестируется.**
- **Код нельзя переиспользовать.**

В ходе разработки, я, как начинающий разработчик на iOS, тоже использовал данную архитектуру приложения и столкнулся с проблемой Massive-View-Controller. Несмотря на то, что программа работала исправно, стало понятно, что дальнейшее развитие проекта может быть затруднено из-за первой проблемы из вышеперечисленного списка. В итоге было принято решение реструктуризировать приложение и передать неположенные функции массивного контроллера представления отдельным сущностям.

Для решения этой проблемы была разработана архитектура, которая представлена на рисунке 7. Я назвал ее Model-View-Controller-Manager. Эта архитектура предполагает создание дополнительной прослойки в связи слоя модели с контроллером представления. Эта дополнительная прослойка состоит из двух сущностей: менеджера работы с Core Data и менеджера работы с заметками. Разделение этой прослойки на две части нужно для того, чтобы при необходимости перехода на другой фреймворк для работы с базами данных, достаточно было переписать сущность для работы с конкретным фреймворком, а сущность для работы с заметками при этом останется неизменной, как и контроллер представления. Так же MVC-M предполагает, в данном случае, вынос функций записи и воспроизведения голоса и планирования локальных уведомлений в ОС тоже в отдельные сущности.

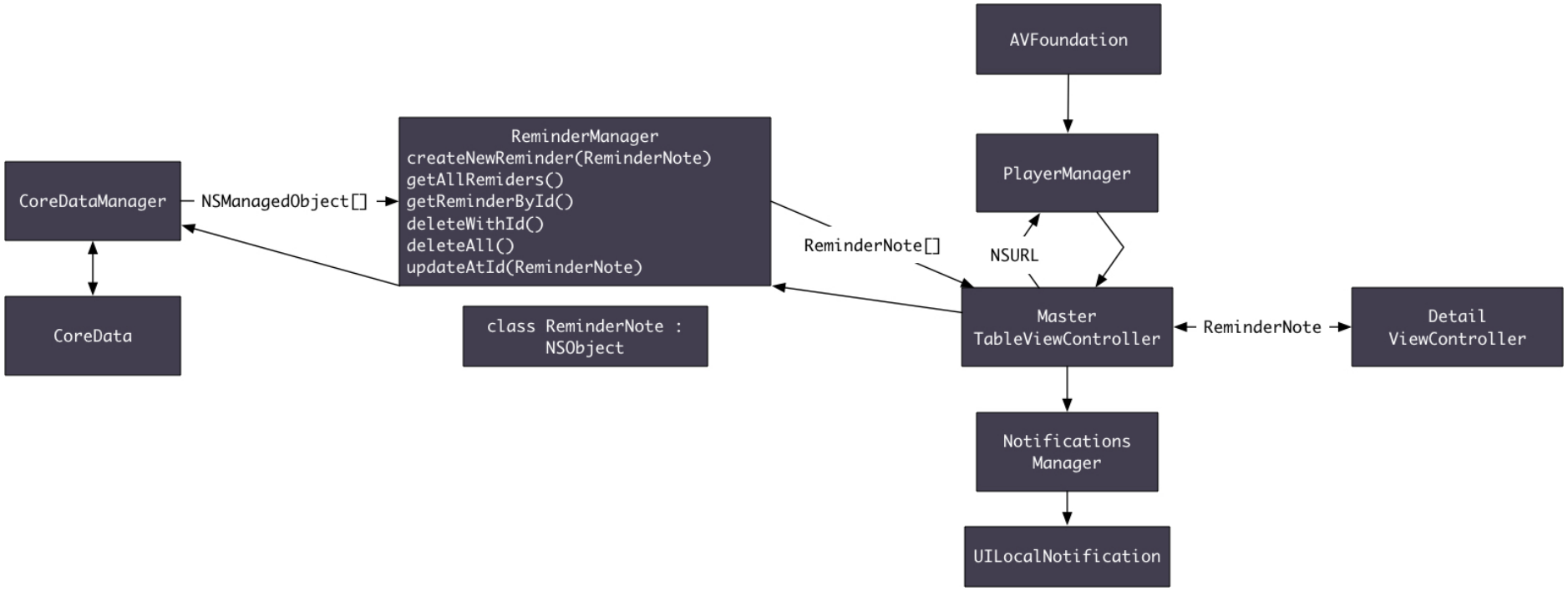


Рисунок 7 — Model-View-Controller-Manager

ЗАКЛЮЧЕНИЕ И ВЫВОДЫ

Подведем итог. Цели, достигнутые в процессе разработки:

- Разработано мобильное приложение на iOS по записыванию голосовых заметок.

Дальнейшие перспективы:

- Полностью переработать графический дизайн мобильного приложения
- Добавить синхронизацию с облачными сервисами
- Добавить распознавание и синтез речи
- Продолжать рефакторинг проекта.

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА И ПО:

Использованное ПО:

- Apple Xcode 8
- Microsoft Word for Mac
- SourceTree
- Prepo
- Adobe Illustrator CC

Список использованных источников:

- Apple Developer:
  - Core Data Programming Guide [Электронный ресурс] — Режим доступа: https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/index.html#//apple_ref/doc/uid/TP40001075-CH2-SW1 — Дата доступа: 19.01.2017
  - AVFoundation [Электронный ресурс] — Режим доступа: https://developer.apple.com/reference/avfoundation?language=objc — Дата доступа: 29.01.2017
  - Programming with Objective-C [Электронный ресурс] — Режим доступа: https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html#//apple_ref/doc/uid/TP40011210-CH1-SW1 — Дата доступа: 15.01.2017
- Wikipedia:
  - Поток выполения [Электронный ресурс] — Режим доступа: https://ru.wikipedia.org/wiki/Поток_выполнения — Дата доступа: 03.04.2017
  - Grand Central Dispatch [Электронный ресурс] — Режим доступа: https://ru.wikipedia.org/wiki/Grand_Central_Dispatch — Дата доступа: 03.04.2017
  - Cocoa Pods [Электронный ресурс] — Режим доступа: Guides https://guides.cocoapods.org — Дата доступа: 13.03.2017
  - [Электронный ресурс] — Режим доступа: https://github.com/MortimerGoro/MGSwipeTableCell — Дата доступа: 13.01.2017

- Apple inc. «Apple Human Interface Guidelines»
- Stanford CS193p Developing Applications for iOS Fall 2013-14
- Objective-C [Электронный ресурс] — Режим доступа: https://ru.wikipedia.org/wiki/Objective-C — Дата доступа: 14.01.2017
- Путь от Massive ViewController до VIPER [Электронный ресурс] — Режим доступа: https://etolstoy.gitbooks.io/the-book-of-viper/content/путь-от-massive-viewcontroller-до-viper.html — Дата доступа: 01.04.2017