Литература

1 Роб, П. Системы баз данных: проектирование, реализация и управление / П. Роб, К. Коронел; пер. с англ. – 5-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2004. – 1040 с.

2 Тейт, Б. Ruby on Rails. Быстрая веб-разработка / Б. Тейт, К. Хиббс. – СПб.: BHV-Петербург, 2008. – 224 с.

3 Кантор, И. Современный учебник JavaScript / И. Кантор. – СПб.: Питер, 2015. – 456 с.

УДК 004.7

Н. С. Андриенко

РЕАЛИЗАЦИЯ IOS-ПРИЛОЖЕНИЯ «ЗАМЕТКИ»

Статья посвящена разработке приложения «Заметки», которое предоставляет полный функционал для создания заметок, поиска по ним, упорядочивание их в группы, редактирования, удаления и изменения пользовательских дел, а также использует для хранения всей пользовательской информации базу данных Core Data, построенной на основе SQLite. Приложение создано при помощи среды разработки XCode для операционной системы Mac OS. Для использования приложения достаточно установить его на устройство и запустить. Приложение не требует особых прав и доступа к интернету.

В связи с большими объемами поступающей информации возникла необходимость в ее корректной обработке. В настоящей работе, используя мобильную операционную систему, разработано приложение «Заметки» позволяющее это делать. Предлагаемое IOS-приложение создано с использованием технологий IOS [1] и языка программирования Swift [2]. Приложение имеет несколько состояний. Состояния зависят друг от друга и меняются строго своей иерархией. Каждое приложение начинает свою работу после нажатия иконки приложения.

Приложение должно быть готово к прекращению работы в любое время и не должно ждать, чтобы сохранить пользовательские данные и выполнить другие важные задачи. Пользователь также, как и система способен удалить приложение из памяти используя многозадачный интерфейс, в этом случае приложение не получит никаких уведомлении об этом.

Разработка приложения велась в среде Xcode доступной для операционной системы Mac OS.

В разработанном приложении реализована возможность создания заметок, поиска по ним, упорядочивание в группы, изменения и удаления. Для создания заметки достаточно нажать на значок + в углу экрана, и пользователь пере направится на экран заполнения заметки. Демонстрация приложения осуществляется с помощью симулятора, который является встроенным средством программной среды XCode. Этот новый инструмент сообщает о проблемах, которые были автоматически выявлены XCode во время запуска приложения, он находит такие трудно отслеживаемые ошибки, которые могут быть не замечены вплоть до выпуска приложения в массы. Инструмент Thread Sanitizeropределяет ошибку «состояния гонки» по изменениям данных и другие ошибки связанные с потоком. Имеется возможность проверить пользовательский интерфейс на наличие ошибок в обновленном View Debugger’e с еще большей точностью.

В ходе выполнения работы программа запускалась на эмуляторах и на физических девайсах с целью выявления ошибок и графических неточностей. При запуске приложения, пользователь несколько секунд наблюдает экран загрузки, после

чего происходит открытие главного экрана приложения. Главный экран, представленный на рисунке 1, был разработан с использованием огромного количества встроенных в систему IOS элементов, главным из которых является Tab Bar, который расположен внизу экрана и позволяет пользователю осуществлять навигацию между экранами, и переходить на другие вкладки. Панель вкладок реализована в классе UITabBar, элементы панели вкладок реализованы в классе UITabBarItem.

Настройка панели инструментов выполняется в Interface Builder. Некоторые виды настроек панели делаются программно.

Рисунок 1 – Главный экран приложения

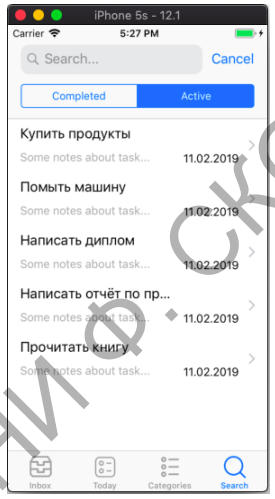

Рисунок 2 – Экран поиска

Каждая вкладка, реализованная в UITabBarItem, имеет наименование по изображению для выбранной и невыбранной вкладки. На вкладке имеется значение для бэйджика. После создания элементов вкладок они добавляются в панель путем изменения свойства items, которое указывает на массив UITabBarItem объектов. При необходимости анимировать добавление вкладок, вместо изменения этого свойства используется метод setItems:animated.

Контент панели вкладок можно изменять во время работы приложения, то есть, пользователь может удалять, добавлять и перестраивать вкладки. Для вывода модального видового представления, которое позволяет пользователю изменить панель вкладок, используется метод beginCustomizingItems. Как только пользователь изменяет Tab Bar, то делегат панели вкладок принимает соответствующее сообщение. Каждая вкладка в панели вкладок имеет пользовательское изображение для выбранного и невыбранного состояния. Эти изображения были заданы при инициализации. На экране заполнения присутствует большое количество элементов таких как, UISwitch, UITextField, UITextVIew. Все перечисленные элементы размещены в статической таблице UITableView.

Ввод даты пользователем реализован через инструмент выбора Date Picker, в котором пользователь самостоятельно выбирает дату, после чего она сохраняется. Использование данного компонента является правилом хорошего тона у IOS разработчиков. Чтобы добавить данный элемент в свой интерфейс необходимо установить вид выбора даты в момент создания данного элемента, установить максимальную и минимальную дату для выбора. При необходимости у данного компонента можно настроить реакцию на пользовательское взаимодействие, в разрабатываемом приложении при изменении даты в пикере она сразу отправляется в строку с итоговой выбранной датой.

У каждого дела также присутствует свой приоритет, который пользователь может выбрать при создании, данная возможность реализована с помощью одного из встроенных элементов дизайна IOS – Action Sheet. Action Sheet отображает набор кнопок,

представляющих несколько альтернативных вариантов выполнения задачи по инициативе пользователя. Для обработки нажатий пользователем кнопок Action Sheet, необходимо назначить делегата, который должен соответствовать протоколу UIActionSheetDelegate. Делегат должен реализовывать сообщение actionSheet:clickedButtonAtIndex в ответ на нажатие кнопки. Для отмены действия в приложении создана кнопка «Отмена», которая позиционируется в нижней части Action sheet.

Пользователь приложения может осуществлять поиск по своим делам. На экране поиска, представленном на рисунке 2, все дела разделены на две части: активные и выполненные. Поиск осуществляется даже по частичным совпадениям, поиск не чувствителен к регистру.

Панель поиска предоставляет интерфейс для поиска информации на основе текста с текстовым полем ввода и кнопками «Поиск» и «Отмена». Панель поиска содержит несколько различных кнопок. Кнопка «Отмена» прекращает операцию поиска. Текст запроса задается в поле Prompt. Запрос выводится прямо поверх панели поиска. В отличие от Placeholder, текст запроса выводится постоянно, вне зависимости от заполнения текстового поля поиска.

Панель поиска нуждается в делегате для обработки пользовательских действий. Поиск осуществляется в базе данных CoreData, в которую сохраняются все пользовательские задачи. Конфигурационный файл базы данных изображен на рисунке 3. Несмотря на то, что Core Data может хранить данные в реляционной базе данных вроде SQLite, Core Data не является СУБД.

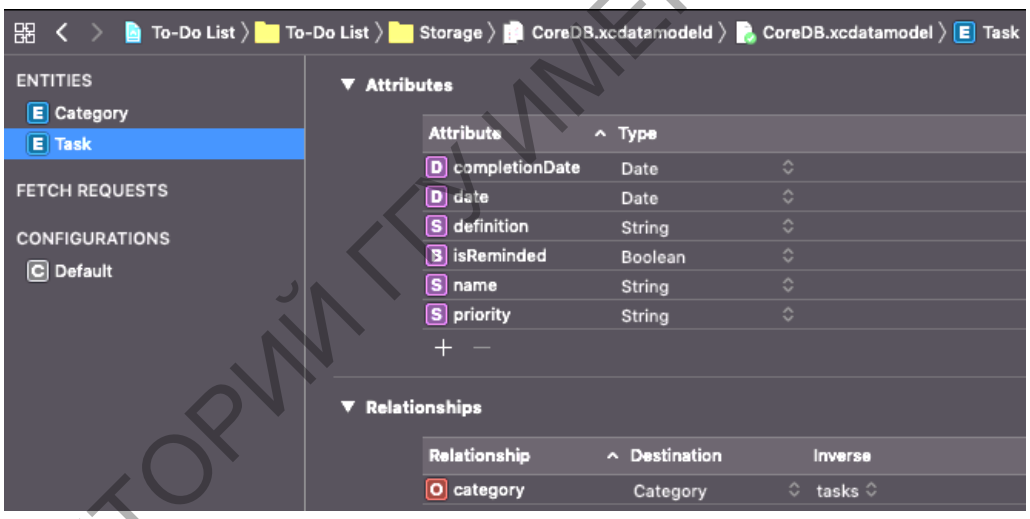

Рисунок 3 – Конфигурационный файл базы данных

Таким образом, в ходе работы было создано приложение «Заметки», которое предоставляет удобный интерфейс. Были обработаны все возможные ошибки. Для использования приложения достаточно установить его на устройство и запустить, приложение не требует особых прав и доступа к интернету. Данный проект использует современные фреймворки, такие как Foundation, UIKit. При написании базы данных использовалась технология Core Data.

Литература

1 Ньюберг, М. Разработка IOS приложений / М. Ньюберг. – СПб.: Питер, 2017. – 564 с.

2 Усов, В. Swift. Основы разработки IOS приложений / В. Усов. – Библиотека программиста.: Питер, 2018. – 464 с.