**ВВЕДЕНИЕ**

С помощью Twitter свою мысль можно выразить так, как диктует время — сжато и оперативно. Оспорить удобства данного сервиса сложно хотя бы потому, что им пользуются более 140 млн. пользователей.

Twitter является важной составляющей нашей интернет жизни. Для кого-то он стал нашим постоянным спутником. Кто-то регулярно проверяет ленту в дороге, кто-то переписывается дома, а для кого-то это уже не только не средство общения, а возможность быстро получить необходимую информацию в сжатом виде. Огромное количество людей подписаны на каналы интересных людей или проектов и регулярно их читают. Twitter очень хорошо смотрится в мобильном формате на iPhone и iPad, но в то же время все популярные клиенты имеют и настольные версии.

Невзирая на то, что официальный сайт Твиттера предоставляет интерфейс, достаточный для полноценного взаимодействия пользователя с ним, существуют и альтернативные оболочки – такие, как расширения, сервисы и программы для разных операционных систем.

Работать с сервисом Twitter можно через Web-интерфейс с использованием браузера, но также возможно для этой цели использовать специальные программы, так называемые Twitter-клиенты. Работа через них считается более безопасной, чем через Web-интерфейс. Такие клиенты есть под разные платформы.

**1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

**1.1 Анализ существующих аналогов**

На данный момент времени существует множество аналогов программных средств для работы с Твиттером. Наиболее популярные: официальное приложение «Twitter» и «TweetBot».

К основным недостаткам большинства таких программ можно отнести то, что они слишком сложные для использования простым пользователем и обладают множеством функций, которые затрудняют работу с программным средством. Зачастую они не обладают интуитивно понятным и лаконичным интерфейсом, который облегчит работу с программным средством. Исходя из вышеперечисленных недостатков, было принято решение создать программное средство – твиттер-клиент под названием «TweetHub».

# 1.2 Постановка задачи

Анализ литературы и предметной области показали, что реализуемое программное средство имеет большое количество аналогов, но каждый из которых имеет свои недостатки.

Основной целью является – разработка программного средства для работы с сайтом twitter.com.

В данном программном средстве необходимо реализовать:

* авторизацию пользователя;
* добавление твита в аккаунт пользователя;
* возможность задать количество отображаемых твитов;
* отображение твитов пользователя;
* связь с поддержкой сайта twitter.com;
* изменение цветовой темы приложения;
* возможность сделать скриншот ленты пользователя.

**1.3 Входные данные**

Входной информацией является:

* имя пользователя;
* количество отображаемых твитов.

**1.4 Выходные данные**

* лента с твитами пользователя;
* добавление твита в аккаунт пользователя;
* сохранение скриншота ленты пользователя.

**2 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**2.1 Разработка спецификаций требований**

Программа будет разрабатываться, и функционировать в операционной системе iOS 10.

iOS 10 - версия операционной системы iOS от корпорации Apple, преемник iOS 9. Система была представлена на конференции WWDC 13 июня 2016 года. Доступна с 13 сентября 2016 года.

Для нормального функционирования проекта необходимо иметь смартфон iPhone 5 с операционной системой iOS 10.

Минимальные требования:

* + - процессор: Apple A6;
    - оперативная память: 1 Гб;
    - операционная система: iOS 10;
    - количество ядер: 2;
    - подключение к сети интернет.

**2.2 Разработка архитектуры программного средства**

Программное средство должно будет брать данные из интернета в режиме реального времени.

STTwitter – стабильная и всесторонняя библиотека Objective-C для Twitter REST API 1.1.

Как версия FOSS для Twitter Fabric TwitterKit, без частей пользовательского интерфейса, но с гораздо большей гибкостью. Также включает мощную консоль Twitter Dev для OS X.

AFNetworking – сетевая библиотека для iOS и Mac OS X. Она построена поверх системы загрузки URL-адресов Foundation, расширяя мощные сетевые абстракции высокого уровня, встроенные в Cocoa. Она имеет модульную архитектуру с хорошо разработанными многофункциональными API-интерфейсами, которые легко использовать.

AFNetworking поддерживает некоторые из самых популярных приложений на iPhone, iPad и Mac.

Интерфейс — это совокупность средств, при помощи которых пользователь взаимодействует с приложением. Для создания лаконичного интерфейса и быстроты работы были использованы следующие компоненты: таблица, ячейки, кнопки, текстовые поля, навигэйшн бар, текстовые метки, экшн меню.

**2.3 Обоснование выбора языка программирования**

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.

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 С, 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 С: любая программа на С 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).

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 языке Objective-C для обозначения объектов используется специальный тип id (это аналог типа Object в Java). Переменная типа id фактически является указателем на произвольный объект. Для обозначения нулевого указателя на объект используется константа nil (= NULL).

При этом вместо id можно использовать и более привычное обозначение с явным указанием класса. В частности последнее позволяет компилятору осуществлять некоторую проверку поддержки сообщения объектами — если компилятор из типа переменной не может сделать вывод о поддержке объектом данного сообщения, то он выдаст предупреждение.

Тем самым язык поддерживает проверку типов, но в нестрогой форме (то есть найденные несоответствия возвращаются как предупреждения, а не ошибки).

**2.4 Разработка схемы программы**

Схема работы программного средства представлена на рисунке 2.1.

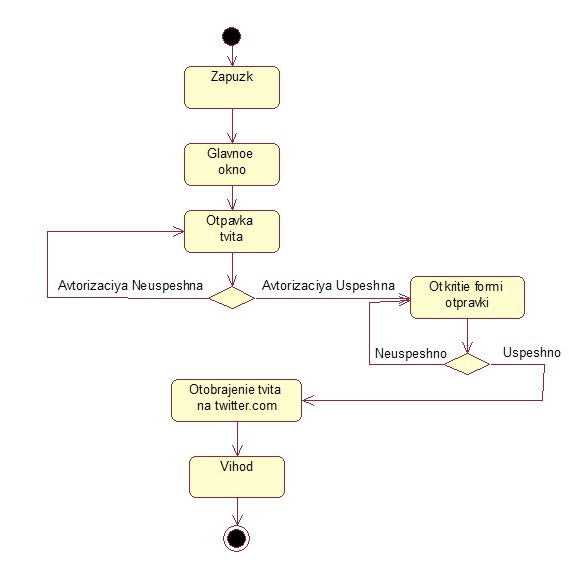


Рисунок 2.1 - Структура программного средства

**2.5 Программная реализация программного средства**

Программное средство предоставляет визуальный интерфейс для доступа к требуемым данным, выполняя работу по пересылке команд между сервером и клиентом, и отображению данных.

Функции:

* авторизацию пользователя;
* добавление твита в аккаунт пользователя;
* возможность задать количество отображаемых твитов;
* отображение твитов пользователя;
* связь с поддержкой сайта twitter.com;
* изменение цветовой темы приложения;
* возможность сделать скриншот ленты пользователя.

**3 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

Тесты существенно различаются по задачам, которые с их помощью решаются, и по используемой технике. Различие задач тестирования приводит, естественным образом, к необходимости использовать весьма разнообразные типы (виды) тестирования. Принято подразделять тестирование на виды по следующим категориям:

* по объектам (элементам) тестирования, часто разделение на виды тестов по данному критерию называют разделением тестирования на уровни;
* по глубине тестирования, то есть разделение тестовых испытаний на типы проводится в зависимости от количества времени и объема тестируемых компонент программного продукта.

1. Модульное тестирование (Автономное или Unit-тестирование).

На данном уровне тестируются по отдельности небольшие элементы системы, максимально отделенные от других элементов и, в то же время, пригодные для тестирования.

2. Комплексное тестирование (Сборочное тестирование, integration testing или interface testing).

На данном уровне тестируются объединенные элементы (компоненты или подсистемы) общей системы, чаще всего некоторая взаимодействующая между собой группа элементов

3. Системное тестирование (system testing).

После того, как система собрана из составляющих компонентов, она должна быть протестирована на соответствие “Системным спецификациям” – реализованы ли все функциональные и нефункциональные требования к разрабатываемой системе.

4. Приемочное тестирование (Приемо-сдаточное тестирование или acceptance testing).

На данном уровне завершенное приложение (система) тестируется Заказчиком, конечными пользователями или соответствующими уполномоченными с целью определения соответствия системы “Требованиям Заказчика” и готовности системы к внедрению

5. Операционное тестирование (Release Testing)

Тестирование в среде эксплуатации позволяет выявить и нефункциональные проблемы, такие как: конфликт с другими системами, смежными в области бизнеса или в программных и электронных окружениях; недостаточная производительность системы в среде эксплуатации и др.

Виды тестирования:

Инсталляционное тестирование (Installation testing). В процессе инсталляционного тестирования проверяется корректность установки и деинсталляции программного продукта в среде максимально приближенной к эксплуатационной. Проверка правильности установки программного продукта должна быть обязательным элементом проекта по тестированию любого продукта. Основная цель состоит в том, чтобы убедиться, что продукт может быть установлен/деинсталлирован при различных условиях – таких как: новая инсталляция, усовершенствование системы (upgrade), установка по умолчанию, полная установка, установка по выбору.

Дымное тестирование (проверка на дым, Smoke testing). Как правило, используется для определения, готова ли программа для проведения более обширного тестирования.

Функциональное тестирование (Functional testing). Проверка соответствия продукта функциональным требованиям и спецификациям.

Функциональные требования включают в себя: Функциональную пригодность, точность, способность к взаимодействию, соответствие стандартам и правилам, защищённость.

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами. В целом, это тестирование того, "Как" система работает. Тестирование нефункциональных требований, то есть: тестирование производительности, защищенности, использования и восстановления.

Регрессионное тестирование (Regression testing). Повторное тестирование после внесения изменений в программное обеспечение или в его окружение (в новой версии приложения), чтобы убедиться в том, что функции, которые работали в предыдущей версии системы, по-прежнему работают так, как ожидалось, а найденные дефекты успешно исправлены (все протестированное ранее тестируется повторно).

Тестирование графического интерфейса пользователя (User Interface testing). Тестирование интерфейса – экранов, кнопок и так далее. Большая часть функциональности ПО реализуется, как правило, через пользовательский интерфейс.

**4 МЕТОДИКА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА**

После запуска приложения, пользователь попадает на главную страницу приложения, которая представлена на рисунке 4.1. В главном окне приложения находятся основные элементы навигации:

* текстовые поля;
* кнопки;

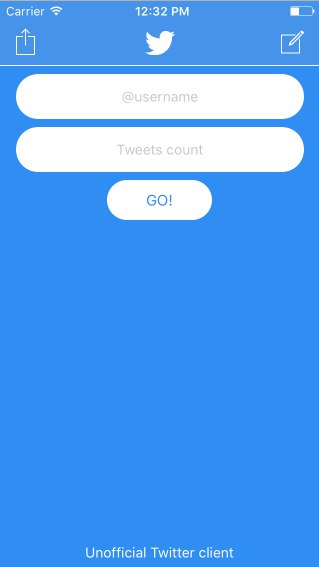


Рисунок 4.1 – Главное окно приложения

Протестируем функцию показа твита пользователя. Для этого введем в первое текстовое поле имя пользователя. Количество твитов вводится по желанию пользователя.

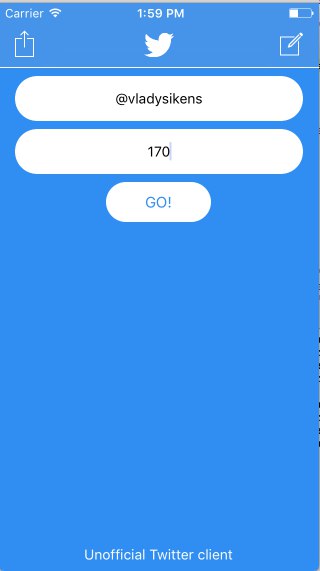


Рисунок 4.2 – Выбор таблицы для работы

После того, как имя пользователя введено, необходимо нажать на кнопку «GO». Далее произойдет переход на форму, которая отображает твиты пользователя. Результат представлен на рисунке 4.3.



Рисунок 4.3 – Страница с твитами пользователя

Протестируем функцию отправки твита. Для этого необходимо предварительно ввести данные пользователя в настройках смартфона iPhone. Если этого не сделать, при попытке отправки твита возникнет ошибка.

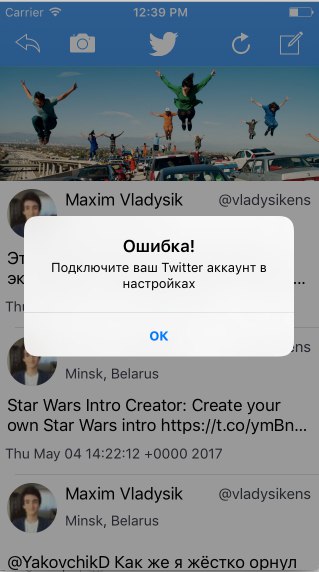


Рисунок 4.4 – Ошибка

На рисунке 4.5 представлен результат ввода данных пользователя, в настройках смартфона iPhone.

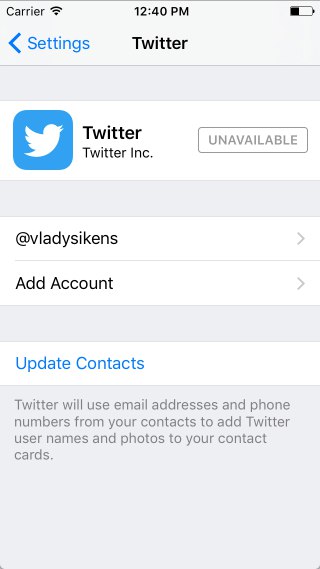


Рисунок 4.5 – Данные пользователя в настройках

После того, как данные пользователя были введены, появится возможность отправить твит.

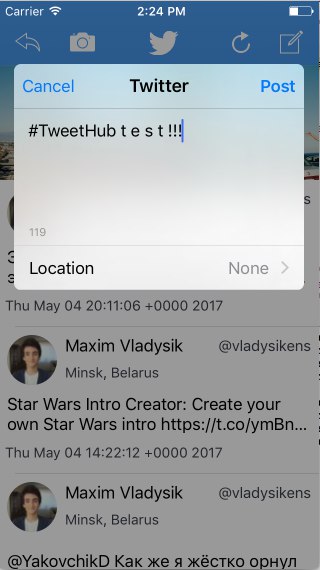


Рисунок 4.6 – Форма отправления твита

Твит был удачно отправлен. Результат отправления твита из приложения «TweetHub» представлен на рисунке 4.7



Рисунок 4.7 – Результат отправления твита из приложения

Протестируем функцию сохранения скриншота. Для этого, находясь в ленте твитов пользователя, необходимо нажать на кнопку с изображением фотоаппарата. Кнопка расположена вверху, слева от логотипа Твиттера. Результатом выполнения действия, будет сохранения скриншота в фотопленку смартфона.

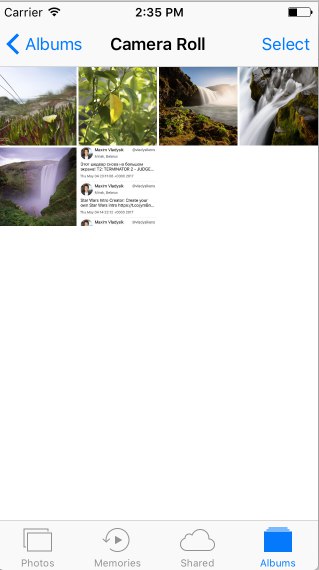


Рисунок 4.8 – Результат сохранения скриншота

**ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте разработано Программное средство для работы с сайтом twitter.com.

В данном программном средстве реализовано;

* экспорт твитов в социальную сеть;
* импорт твитов;
* поддержка пользователя;
* изменение цветовой темы приложения.

Курсовая работа была реализована на объектно-ориентированном языке программирования Objective-C с использованием среды разработки Xcode.

Все задачи, которые ставились в начале проектирования, были реализованы и в некоторых местах дополнены.

В дальнейшем программное средство может быть модифицировано различными элементами управления и настройками, которые позволят сделать данное программное средство более универсальным.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Свободная энциклопедия [Электронный ресурс] / центр информ. Режим доступа: http://ru.wikipedia.org/wiki/Windows\_10.
2. Become an Xcoder / Бутарович. – Питер, 2007. – 240с.
3. Дьюсон Робин. для начинающих разработчиков. СПб.: БХВ-Питер, 2009. – 704 с.
4. Сайт разработчика [Электронный ресурс] центр информ. Режим доступа: http://www.protesting.ru/testing/. Дата доступа: 04.04.2017.
5. Энциклопедия [Электронный ресурс] центр информ. Режим доступа: http://habrahabr.ru/post/169381/ /ARM. Дата доступа: 06.04.2017.
6. Канер С., Фолк Дж., Енг Кек Нгуен. Тестирование программного обеспечения, 2004. – 73 с.