Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

 К ЗАЩИТЕ ДОПУСТИТЬ

\_\_\_\_\_\_\_\_\_\_\_ М. Г. Базанов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

«Twitter - клиент для платформы Android»

по дисциплине

«Системное программное обеспечение вычислительных машин»

БГУИР КР 1-40 02 01 514 ПЗ

Выполнил:                                                                   Руководитель:

студент гр. 650505                                                     Базанов М. Г.

Дик Н. А.

МИНСК 2018

Учреждение образования

«Белорусский государственный университет информатики

и радиоэлектроники»

Факультет: Компьтерных систем и сетей

Кафедра: Электронных вычислительных машин

Специальность: 40 02 01 «Вычислительные машины, системы и сети»

Предмет: Системное программное обеспечение электронных вычислительных машин

УТВЕРЖДАЮ Заведующий кафедрой ЭВМ \_\_\_\_\_\_\_\_\_ Д. И. Самаль

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

ЗАДАНИЕ

по курсовому проектированию студента

Дика Николая Алексеевича

1. Тема проекта: «Twitter - клиент для платформы Android».

2. Дата выдачи задания: 20.02.2018 г.

3. Предоставление студентом готового проекта: 11.06.2018 г.

4. Решаемые задачи и функционал разрабатываемого ПО:

4.1. Вход в личный аккаунт Twitter.

4.2. Возможность просмотра новостной ленты Twitter.

4.3. Написание новых «твитов» в свой аккаунт.

4.4. Просмотр ленты собственных «твитов».

4.5. Возможность осуществления поиска по хэш-тэгам и встречающемуся тексту в «твите» в сети Twitter.

4.6. Просмотр фото и видео в ленте

5. ОС и средства разработки:

5.1. *Android Studio.*

5.2. *Windows* 10.

6. Другое

6.1. Должна быть предусмотрена максимальная длина отправляемого «твита» в соответствии с политикой Twitter,

7. Содержание пояснительной записки:

Введение. 1. Цели и задача курсовой работы 2. Обзор литературы. 3. Структурное проектирование. 4. Функциональное проектирование. 5. Разработка алгоритмов. 6. Руководство пользователя. Заключение. Литература. Приложения.

8. Перечень графического материала:

8.1. Диаграмма классов. Формат А3.

8.2. Схема алгоритма функции searchTimeLine(). Формат А4.

8.3. Схема алгоритма функции getInfo(). Формат А4.

8.4.Схема алгоритма функции putInfo(Result<User> result). Формат А4.

8.5. Листинг программы. Формат А4.

9. Контрольные точки готовности частей проекта:

9.1. 01.03 – введение, глава 1 (постановка задачи), лист задания.

9.2. 01.04 – предварительная структурная схема и диаграмма классов.

9.3. 01.05 – предварительная демонстрация ПО.

9.4. 10.05 – структурная схема, диаграмма классов, главы 1-6.

РУКОВОДИТЕЛЬ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Базанов М. Г.

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дик Н. А.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc484745201)

[1 ЦЕЛИ И ЗАДАЧА КУРСОВОЙ РАБОТЫ 7](#_Toc484745202)

[2 ОБЗОР ЛИТЕРАТУРЫ 8](#_Toc484745203)

[2.1 Обзор модуля Fabric (Twitter Kit) 8](#_Toc484745204)

[2.2 Обзор аналогов 8](#_Toc484745205)

[3 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ 12](#_Toc484745206)

[3.1 Структура входных-выходных данных 12](#_Toc484745207)

[4 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 13](#_Toc484745209)

[4.1 Класс Const.java 13](#_Toc484745210)

[4.2 Класс LoginActivity.java 13](#_Toc484745211)

[4.3 Класс MainActivity.java 13](#_Toc484745212)

[4.4 Класс SearchActivity.java 14](#_Toc484745213)

[4.5 Класс SearchTimeLineFragment.java 14](#_Toc484745214)

[4.6 Класс TimeLineFragment.java 14](#_Toc484745215)

[4.7 Класс HomeActivity.java 14](#_Toc484745216)

[4.8 Файл activity\_login.xml 14](#_Toc484745217)

[4.9 Файл activity\_main.xml 15](#_Toc484745218)

[4.10 Файл activity\_home\_time\_line.xml 15](#_Toc484745219)

[4.11 Файл activity\_search.xml 15](#_Toc484745220)

[4.12 Файл timeline\_fragment.xml 15](#_Toc484745221)

[5 РАЗРАБОТКА АЛГОРИТМОВ 16](#_Toc484745222)

[5.1 Схема функции TweetTimelineListAdapter searchTimeLine() 16](#_Toc484745223)

[5.2 Схема функции void getInfo() 16](#_Toc484745223)

[5.3 Схема функции void putInfo(Result<User> result) 16](#_Toc484745223)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 17](#_Toc484745226)

[6.1 Системные требования 17](#_Toc484745223)

[6.2 Запуск программы 17](#_Toc484745223)

[6.3 Использование программы 17](#_Toc484745223)

[ЗАКЛЮЧЕНИЕ 25](#_Toc484745228)

[ЛИТЕРАТУРА 26](#_Toc484745229)

[ПРИЛОЖЕНИЕ А 27](#_Toc484745230)

[ПРИЛОЖЕНИЕ Б 28](#_Toc484745230)

[ПРИЛОЖЕНИЕ В 29](#_Toc484745231)

[ПРИЛОЖЕНИЕ Г 30](#_Toc484745232)

[ПРИЛОЖЕНИЕ Д 31](#_Toc484745230)

**ВВЕДЕНИЕ**

В данной курсовой работе Twitter-клиент. Для реализации данного клиента использовался язык программирования Java. Вместе с тем использовался модуль Fabric (Twitter Kit).

Twitter-клиент — это клиентское приложение, позволяющее авторизоваться в своем аккаунте социальной сети Twitter, использовать основные функции данной социальной сети.

Социальная сеть Twitter на сегодняшний день является одной из самых известных и пользующихся спросом социальных сетей мирового уровня, таких как Facebook, Telegram, Instagram и т.п.

Twitter создан для того, чтобы люди могли легко и кратко делиться какими-либо новостями, полезными и не очень. Эта социальная сеть очень проста и, наверное, этим и привлекает такое количество людей.

Наряду со стандартными функциями, некоторые веб-серверы имеют дополнительные. Так, к примеру, соответствующее программное обеспечение может фиксировать число обращений пользователей к тому или иному ресурсу, записывать их в отдельный журнал.

# **1 ЦЕЛИ И ЗАДАЧА КУРСОВОЙ РАБОТЫ**

*Цели курсовой работы:* освоение практических навыков проектирования и разработки законченного, отлаженного и протестированного программного продукта; закрепление и углубление теоретических знаний, полученных при изучении курса «Системное программное обеспечение ВМ», изучить и использовать сторонние средства.

*Задача курсовой работы:* проектирование программного обеспечения, a именно Twitter-клиент под платформу Android.

Разрабатываемый клиент должен осуществлять поиск «твитов», вывод новостной и пользовательской ленты, написание новых «твитов». Использовать в программе сторонние библиотеки.

# **2 ОБЗОР ЛИТЕРАТУРЫ**

## **2.1 Обзор модуля Fabric (Twitter Kit)**

При разработке twitter-клиента под платформу Android на языке Java был использован модуль Fabric.   
Данный модель позволяет с легкостью внедрять в свое приложение полезные функции. С его помощью была реализована авторизация в приложении, вывод «твитов», поиск и извлечение информации о пользователе.

Для того, чтобы использовать Fabric, его необходимо установить. А сделать это очень просто в самой Android Studio.  
После установки можно выбрать необходимые вам библиотеки и функции. Одним нажатием кнопки Fabric самостоятельно внедрит в ваше приложение необходимые библиотеки. И после чего еще предложит подробную инструкцию для реализации некоторых функций.

А уже для получения большей информации о возможностях Fabric можно узнать на сайте <https://docs.fabric.io>.

## **2.2 Обзор аналогов**

 На сегодняшний день существует множество Twitter-клиентов. Многие их которых очень похожи, многие из них очень удобны в использовании и пользуются большим спросом, а многие совсем малоизвестны из-за недостатков в функционале и интерфейсе.

В данном разделе будут приведены примеры аналогов. Будут кратко рассмотрены их достоинства и недостатки.

* **Twitter for Android (официальное приложение)**

Удобство чтения ленты. Главным преимуществом официального twitter-клиента всегда была скорость работы, в отличие от многих сторонних приложений, в нем не тормозит пролистывание ленты, даже на большой скорости. В новой версии появилось переключение вкладок с помощью жестов вправо/влево, однако работа с данным обновлением приносит немалые неудобства.

**Работа с отдельным «твитом».** При нажатии на «твит», он открывается в отдельном окне. Отсюда же можно быстро «ретвитнуть» сообщение, ответить автору или добавить его в избранное.

Плюсы:

1. Бесплатный
2. Поддержка изображений с pic.twitter.com
3. Holo-стиль

Минусы:

1. Нет поддержки картинок в ленте
2. Тормоза при переходе с одной вкладки на другую

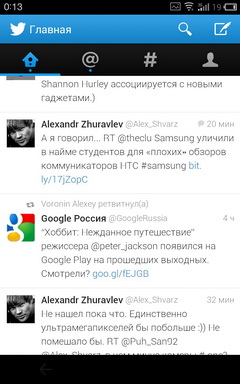


Рисунок 2.2.1 – Twitter-клиент (официальное приложение)

* **Plume**

«Плюм» является одним из самых старых и функциональных твиттер-клиентов на Android. Год назад его даже можно было назвать лучшим.  
  
**Удобство чтения ленты.** Первое на что обращаешь внимание – количество информации, поместившееся на экране. В Plume очень грамотно распорядились местом, в результате пользователям нужно гораздо реже листать ленту. Переключение между вкладками осуществляется с помощью горизонтального свайпа.

**Работа с изображениями.** Клиент поддерживает отображение картинок в ленте, по умолчанию это миниатюры, но в настройках можно включить и большие превью.

**Работа с отдельным «твитом».** При нажатии на «твит», перед вами открывается небольшая панель для работы с ним («ретвит», ответы, добавление в избранное и переход по ссылке). Если «твит» является частью диалога, то он тоже подгружается.

Плюсы:

1. Поддержка встроенного браузера
2. Большие миниатюры в ленте
3. Настройки фотохостинга
4. Множество настроек

Минусы:

1. Скорость и плавность работы

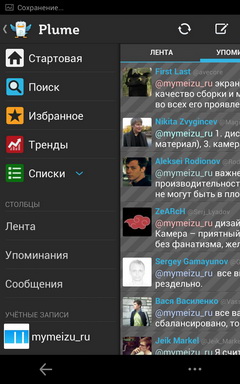


Рисунок 2.2.2 – Plume

* **Robird**

**Удобство чтения ленты.** По умолчанию лента Robird выполнена в темных тонах, но при желании тема меняется в настройках. Разработчик уделил много внимания скорости и плавности работы, поэтому Robird на данный момент можно назвать самым быстрым и плавным twitter-клиентом.

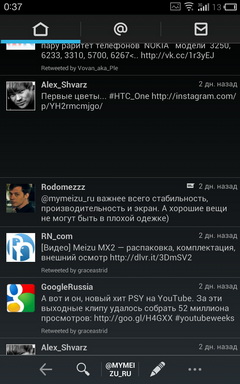


Рисунок 2.2.3 – Robird

**Работа с изображениями.** Как и другие twitter-клиенты, Robird поддерживает почти все популярные фотохостинги, от официального pic.twitter.com до Instagram. Интересно, что Robird корректно работает и несколькими вложенными картинками, кроме того с его помощью вы и сами можете опубликовать сразу несколько изображений. В ленте Robird отображает картинки в виде прямоугольных миниатюр, при нажатии на них картинка открывается полностью.

**Работа с отдельным «твитом».** При нажатии на «твит» он выделяется, а кнопки на SmartBar меняются. Теперь там появляются иконки для работы с «твитом»: быстрый ответ, «ретвит», добавление в избранное. Если вы хотите посмотреть всю переписку, достаточно долгого нажатия на «твит».

Плюсы:

1. Скорость и плавность работы
2. Полная поддержка SmartBar
3. Удобная навигация
4. Поддержка практически всех фотохостингов
5. Поддержка нескольких изображений в ленте

Минусы:

1. Отсутствие встроенного браузер
2. Отсутствие поддержки нескольких аккаунтов

# **3 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ**

## **3.1 Структура входных-выходных данных**

Используемые файлы: «ic\_account\_balance.png», «ic\_search.png», «login\_activity.jpg», «border\_bottom.xml», «border\_et.xml», «border\_layout.xml», «button\_style.xml» .

Файлы с расширением .png и .jpg в качестве иконок некоторых кнопок, а также в качестве фонового изображения Activity. Файлы с расширением .xml служат для описания стилей некоторых виджетов.

# **4 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

## **4.1 Класс Const.java**

В данном классе хранятся константы, необходимые для входа в аккаунт через приложение, а также поля, кторые будут использоваться в качестве ключей для передачи некоторых параметров, присущих пользователю.

Класс Const имеет следующие поля:

String USER\_NAME – ключ имени пользователя.

String USER\_ID – ключ идентификатора пользователя.

String SCREEN\_NAME – ключ имени пользователя.

String LOCATION – ключ местоположения пользователя.

String FOLLOWING\_COUNT – ключ количества подписок пользователя.

String FOLLOWERS\_COUNT – ключ количества подписчиков пользователя.

String TWEET\_COUNT – ключ количества «твитов» пользователя.

String PROFILE\_IMG\_URL – ключ URL фото пользователя.

String PROFILE\_IMG\_BG\_URL – ключ URL фонового фото пользователя.

String SEARCH\_TEXT – ключ текста для поиска пользователя.

String CONSUMER\_KEY – ключ, который дается при регистрации приложения в сети Twitter.

String CONSUMER\_SECRET – секретный пароль, который также дается при регистрации приложения, соответствующий CONSUMER\_KEY.

## **4.2 Класс LoginActivity.java**

LoginActivity – это Activity, которая отображается при запуске приложения. Она реализует вход пользователя в аккаунт.

Она включает в себя один элемент – кнопку входа:

TwitterLoginButton loginButton;

Класс, реализующий данную кнопку принадлежит Twitter библиотеке, подключить которую можно в Android Studio (com.twitter.sdk.android.core).

При нажатии на кнопку происходит перенаправление на сайт Twitter, где пользователю предлагают разрешить авторизацию через Twitter в данном приложении.

После авторизации данные о пользователе передаются в MainActivity для дальнейшей работы с ними.

## **4.3 Класс MainActivity.java**

В MainActivity создаются кнопки FloatingActionButton, которые располагаются в Activity поверх всех слоев и виджетов.

Таких кнопок три:

1. add\_tweet\_btn – кнопка для написания нового твита.
2. search\_btn – кнопка для поиска среди «твитов».
3. home\_btn – кнопка для просмотра новостной ленты.

После вызывается функция, которая принимает данные, переданные ранее из LoginActivity.

Затем создается экземпляр TimeLineFragment, в котором будет отображаться лента «твитов» пользователя.

## **4.4 Класс SearchActivity.java**

В SearchActivity присутсвует поле для ввода текста, по которому будет осуществляться поиск, а также две кнопки:

1. search\_btn – кнопка для поиска.
2. clear\_btn – кнопка для очистки поля для ввода.

При нажатии на кнопку поиска создается экземпляр SearchTimeLineFragment, в котором будет отображаться список «твитов», удовлетворяющих поисковому запросу.

### **4.5 Класс SearchTimeLineFragment.java**

SearchTimeLineFragment – это фрагмент типа ListFragment, который располагается в SearchActivity. В нем создается экземпляр SearchTimeLine в который передается текст, по которому будет осуществляться поиск.

Итоговый список и будет отображать данный фрагмент.

### **4.6 Класс TimeLineFragment.java**

TimeLineFragment – это фрагмент типа ListFragment, который располагается в MainActivity. В нем создается экземпляр UserTimeLine в который передается имя пользователя, чей список твитов и будет отображаться в данном фрагменте.

## **4.7 Класс HomeActivity.java**

В HomeActivity отображается новостная лента пользователя, т.е. новые «твиты» подписок пользователя.

В ней создается экземпляр FixedTweetTimeLine и адаптер TweetTimeLineListAdapter, в который передается имя пользователя, для которого будет отображаться новостная лента в данном фрагменте.

## **4.8 Файл activity\_login.xml**

В данном файле прописана xml разметка LoginActivity.

Она включает в себя фоновое изображение Activity и кнопку для в хода в аккаунт с помощью Twitter.

## **4.9 Файл activity\_main.xml**

В данном файле прописана xml разметка MainActivity.

Она включает в себя:

1. Фоновое изображение страницы пользователя.
2. Главное изображение аккаунта.
3. Имя пользователя.
4. Логин пользователя.
5. Количество «твитов» пользователя.
6. Количество подписок пользователя.
7. Количество подписчиков пользователя.
8. TimeLineFragment, в котором отображается список «твитов» пользователя.

## **4.10 Файл activity\_home\_time\_line.xml**

В данном файле прописана xml разметка HomeActivity. В ней содержится фрагмент, в котором отображается новостная лента.

**4.11 Файл activity\_search.xml**

В данном файле прописана xml разметка SearchActivity. В ней содержится фрагмент, в котором отображается лента «твитов», удовлетворяющая поиску.

## **4.12 Файл timeline\_fragment.xml**

В данном файле прописана xml разметка TimeLineFragment. В ней содержится фрагмент, в котором отображается лента пользователя.

# **5 РАЗРАБОТКА АЛГОРИТМОВ**

С точки зрения алгоритмов, интерес представляют алгоритмы реализующие:

* Создание списка результатов поиска searchTimeLine()
* Получение данных из другой Activity getInfo()
* Передачу данных в другую Activity putInfo(Result<User> result)

## **Схема алгоритма функции** TweetTimelineListAdapter searchTimeLine()

Данная функция реализует создание списка, являющегося результатом поиска среди «твитов» в системе. Разработанная схема представлена на чертеже в Приложении Б.

## **Схема алгоритма функции** void getInfo()

Данная функция принимает данные о пользователе из LoginActivity при входе. Разработанная схема представлена на чертеже в Приложении В.

## **Схема алгоритма функции** void putInfo(Result<User> result)

Данная функция передает данные о пользователе в MainActivity при входе в аккаунт. Разработанная схема представлена на чертеже в Приложении Г.}

# **6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**6.1 Системные требования**

Минимальные системные требования:

* Смартфон на платформе Android версии не менее 4.0
* Доступ в интернет
* Доступ к учетной записи пользователя в социальной сети Twitter.

Приложение сохраняет текущую сессию, что позволяет пользователю проходить авторизацию лишь раз. При необходимости, из учетной записи можно выйти и войти в другой аккаунт.

**6.2 Запуск программы**

Для запуска программы нужно выполнить следующие действия:

1. Вставить CD-диск в дисковод
2. Открыть папку в зависимости от операционной системы
3. Скопировать исполняемый файл *twitter.apk* на Android-устройство
4. На устройстве запустить исполняемый файл клиента
5. При необходимости в настройках разрешить установку приложений не из Play Market.

Все необходимое находится внутри проекта

**6.3 Использование программы**

После запуска приложения на главном экране, представленном на рисунке 6.1, будет располагаться кнопка, позволяющая пользователю войти в приложение с помощью Twitter.



Рисунок 6.1 – Стартовая страница приложения

После нажатия кнопки, пользователь будет перенаправлен на сайт для в хода в учетную запись Twitter ( рисунок 6.2.).

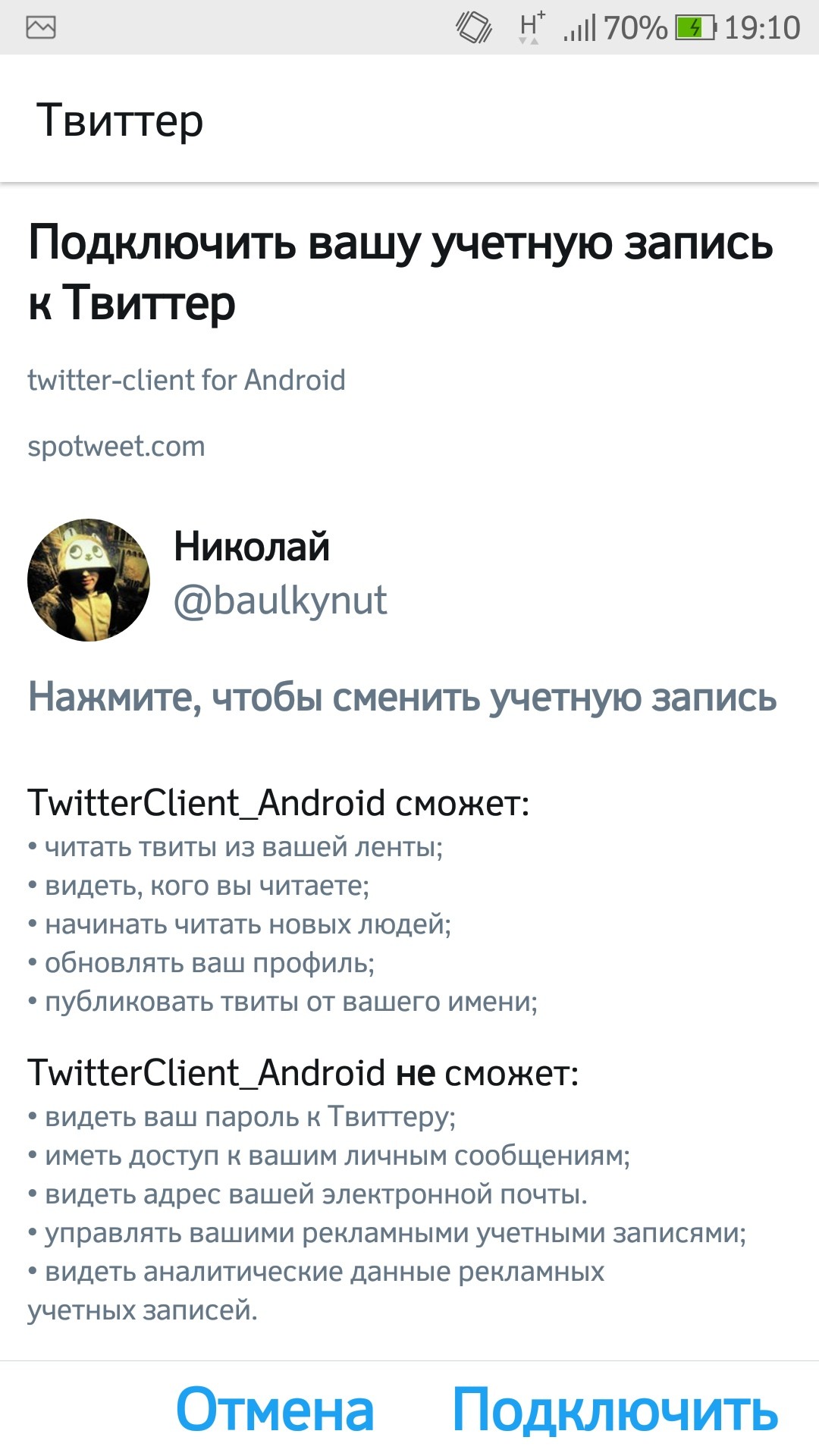


Рисунок 6.2 – Страница браузера для входа в аккаунт

После входа в аккаунт и подтверждения авторизации (разрешения приложению выполнять доступные ему операции) пользователю откроется новая Activity, в которой будет находится информация о пользователе и список его «твитов», как показано на рисунке 6.3. Также будут доступны три кнопки, которые осуществляют написание новых «твитов», поиск среди «твитов» и просмотр новостной ленты.



Рисунок 6.3 – Страница профиля

При нажатии на кнопку с иконкой сообщения, пользователь будет перенаправлен в новую Activity, в которой реализована возможность написания нового «твита». Изображение данной Activity представлено на рисунке 6.4.

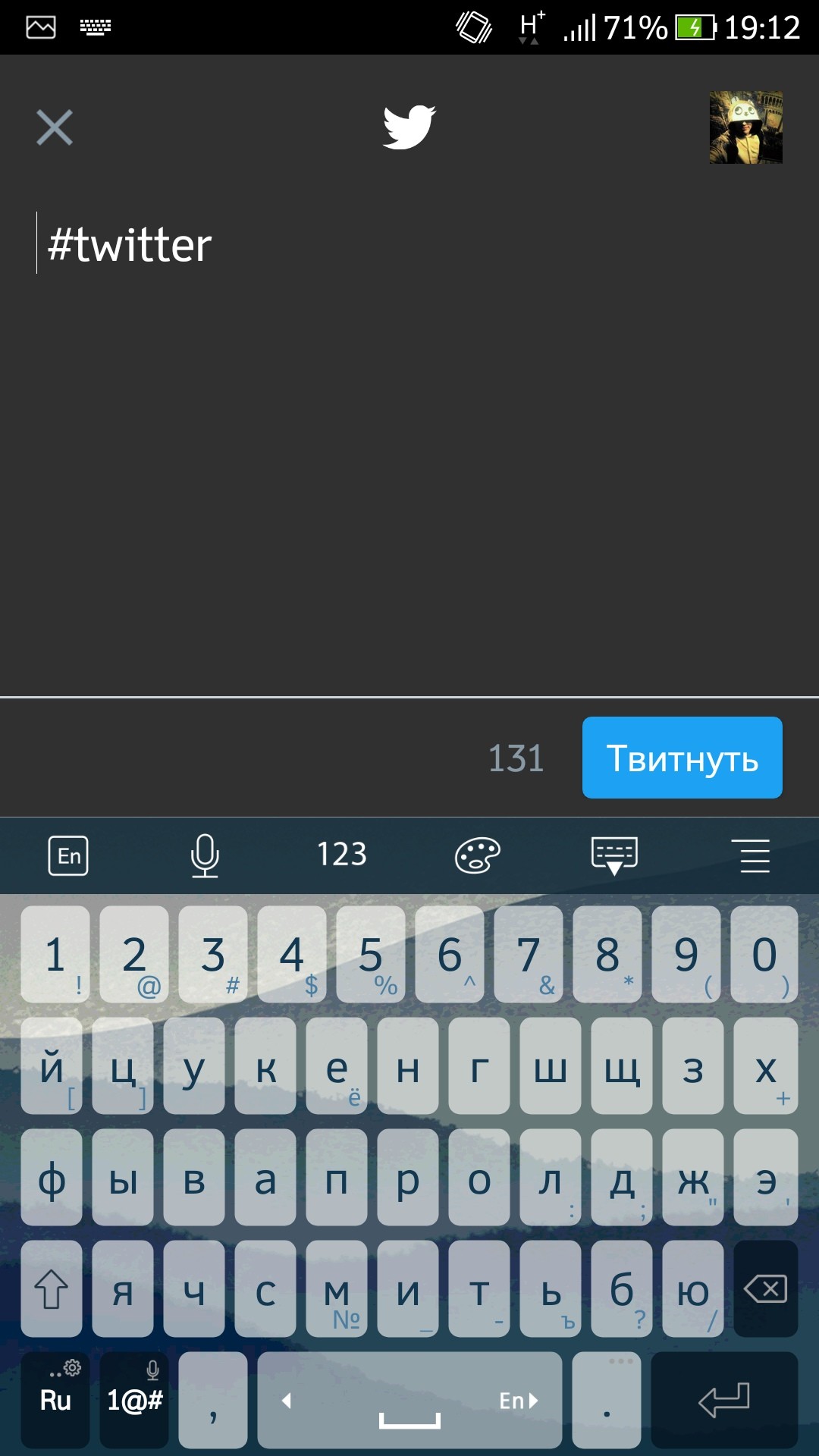


Рисунок 6.4 – Activity для написания нового «твита»

При нажатии на странице профиля кнопки поиска, пользователю откроется Activity для осуществления поиска. Результат поиска среди «твитов» по хэш-тэгу «#twitter» представлен на рисунке 6.5.



Рисунок 6.5 – Результат поиска по хэш-тэгу «#twitter»

При нажатии на странице профиля кнопки Home, пользователю откроется Activity в которой будет отображаться новостная лента, как показано на рисунке 6.6.

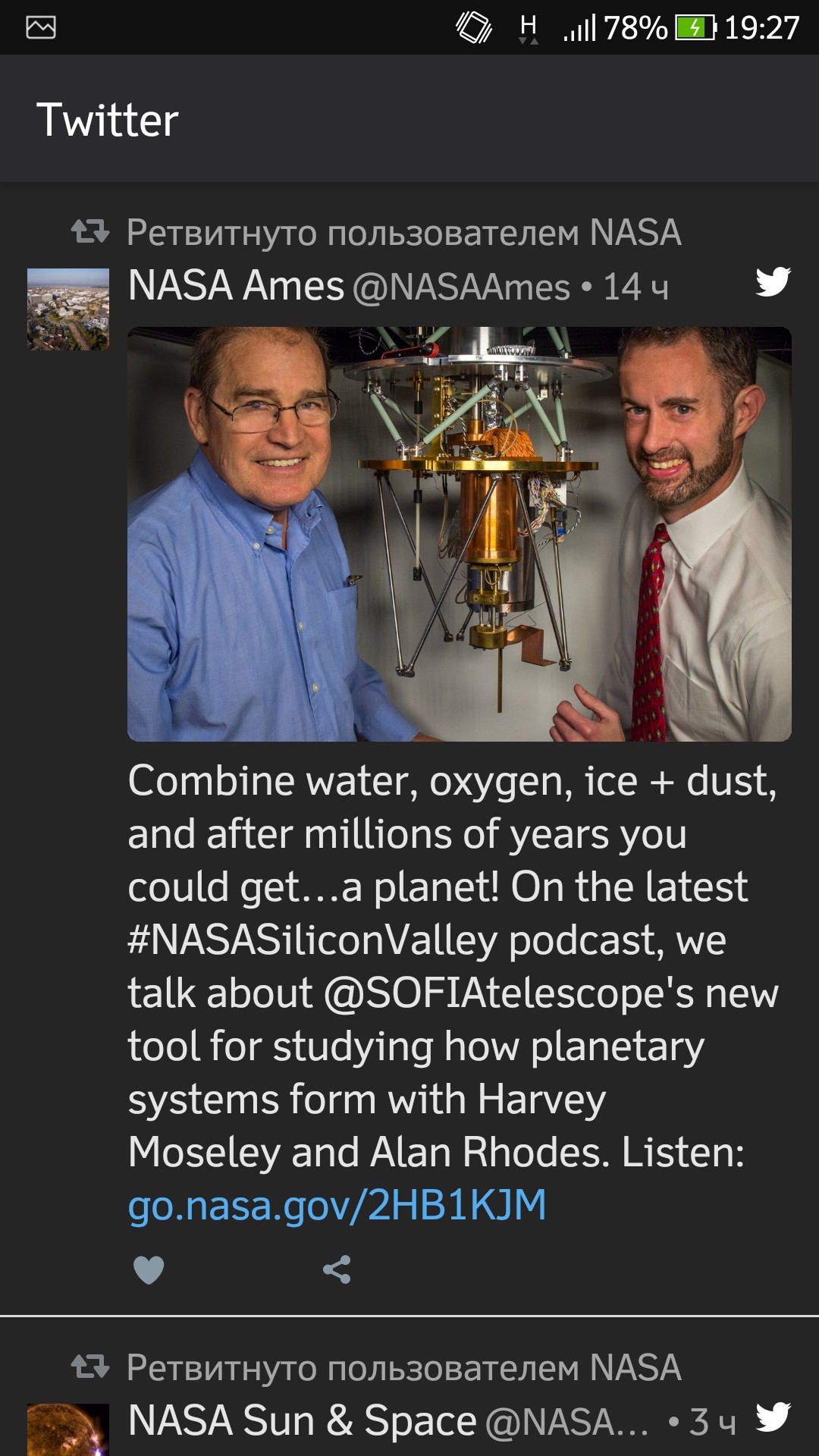


Рисунок 6.6 – Новостная лента

**ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте был разработан «Twitter-клиент под платформу Android», позволяющий просматривать новостную ленту, ленту собственных «твитов», писать новые «твиты», а также осуществлять поиск среди «твитов». К достоинствам данной программы относится простота использования приложения, а также приятный, с эстетической точки зрения, интерфейс. Конечно, по своему функционалу это приложение уступает официальному приложению Twitter, но в дальнейших планах лежит усовершенствование данного приложения, добавления новых необходимых функций и экспериментов в плане графического интерфейса.

В ходе разработки данного клиентского приложения были получены глубокие знания в разработке ПО под платформу Android, также приобретены новые навыки в языке программирования Java.

**ЛИТЕРАТУРА**

1. Интернет-ресурс: <http://developer.android.com>.
2. Интернет-ресурс: <http://startandroid.ru>.
3. Интернет-ресурс (документация Fabric): <https://docs.fabric.io>.
4. Интернет-ресурс: <http://developer.alexanderklimov.ru>.

# **ПРИЛОЖЕНИЕ А**

*(обязательное)*

Диаграмма классов

# **ПРИЛОЖЕНИЕ Б**

*(обязательное)*

Схема алгоритма функции searchTimeLine()

# **ПРИЛОЖЕНИЕ В**

*(обязательное)*

Схема алгоритма функции getInfo()

# **ПРИЛОЖЕНИЕ Г**

*(обязательное)*

Схема алгоритма функции putInfo(Result<User> result)

# **ПРИЛОЖЕНИЕ Д**

*(обязательное)*

Листинг программы

package com.example.usr.viewsapplication;

public class Const {

public static String USER\_NAME = "user\_name";

public static String USER\_ID = "user\_id";

public static String SCREEN\_NAME = "user\_screen\_name";

public static String LOCATION = "user\_location";

public static String FOLLOWING\_COUNT = "user\_following\_count";

public static String FOLLOWERS\_COUNT = "user\_followers\_count";

public static String TWEET\_COUNT = "tweet\_count";

public static String PROFILE\_IMG\_URL = "user\_img\_url";

public static String PROFILE\_IMG\_BG\_URL = "user\_img\_bg\_url";

public static final String SEARCH\_TEXT = "search\_text";

public static final String CONSUMER\_KEY = "uG0NN9yUSGyPqBxddZ3MOmef5";

public static final String CONSUMER\_SECRET = "maI7Sv811LRE4q7wwcOtgMAKktdqdlpWaZ8SklYtHJLV5IoZos";

public static final String OAUTH\_CALLBACK\_SCHEME = "x-oauthflow-twitter";

public static final String OAUTH\_CALLBACK\_HOST = "callback";

public static final String OAUTH\_CALLBACK\_URL = OAUTH\_CALLBACK\_SCHEME + "://" + OAUTH\_CALLBACK\_HOST;

}

package com.example.usr.viewsapplication;

import android.content.Intent;

import android.graphics.Color;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.Gravity;

import android.view.View;

import android.widget.ImageView;

import android.widget.TextView;

import com.squareup.picasso.Picasso;

import com.twitter.sdk.android.core.TwitterCore;

import com.twitter.sdk.android.core.TwitterSession;

import com.twitter.sdk.android.tweetcomposer.ComposerActivity;

import static android.R.drawable.ic\_dialog\_email;

public class MainActivity extends AppCompatActivity {

private final com.example.usr.viewsapplication.vfd vfd = new vfd();

private ImageView ivUserProfileBGImg;

private TextView txtUserScreenName;

private TextView txtUserTagLine;

private TextView txtFollowersCount;

private TextView txtFollowingCount;

private TextView txtTweetCount;

private Bundle extras;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

FloatingActionButton add\_tweet\_btn = new FloatingActionButton.Builder(this)

.withDrawable(getResources().getDrawable(ic\_dialog\_email))

.withButtonColor(Color.parseColor("#098f79"))

.withGravity(Gravity.BOTTOM | Gravity.RIGHT)

.withMargins(0, 0, 16, 16)

.create();

FloatingActionButton search\_btn = new FloatingActionButton.Builder(this)

.withDrawable(getResources().getDrawable(R.drawable.ic\_search))

.withButtonColor(Color.parseColor("#098f79"))

.withGravity(Gravity.BOTTOM | Gravity.RIGHT)

.withMargins(0, 0, 16, 80)

.create();

FloatingActionButton home\_btn = new FloatingActionButton.Builder(this)

.withDrawable(getResources().getDrawable(R.drawable.ic\_account\_balance))

.withButtonColor(Color.parseColor("#098f79"))

.withGravity(Gravity.BOTTOM | Gravity.RIGHT)

.withMargins(0, 0, 16, 150)

.create();

getInfo();

TimeLineFragment timeLineFragment = new TimeLineFragment();

timeLineFragment.setArguments(extras);

getSupportFragmentManager()

.beginTransaction()

.replace(R.id.root\_container, timeLineFragment)

.commit();

add\_tweet\_btn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

final TwitterSession session = TwitterCore.getInstance().getSessionManager()

.getActiveSession();

final Intent intent1 = new ComposerActivity.Builder(MainActivity.this)

.session(session)

.darkTheme()

.hashtags("#twitter")

.createIntent();

startActivity(intent1);

}

});

search\_btn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(MainActivity.this, SearchActivity.class);

startActivity(intent);

}

});

home\_btn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(MainActivity.this, HomeTimeLine.class);

startActivity(intent);

}

});

}

public void getInfo(){

extras = getIntent().getExtras();

String userName = extras.getString(Const.USER\_NAME);

String screenName = extras.getString(Const.SCREEN\_NAME);

String userId = extras.getString(Const.USER\_ID);

String followersCount = extras.getString(Const.FOLLOWERS\_COUNT);

String followingCount = extras.getString(Const.FOLLOWING\_COUNT);

String tweetCount = extras.getString(Const.TWEET\_COUNT);

String profileImgUrl = extras.getString(Const.PROFILE\_IMG\_URL);

String profileImgBgUrl = extras.getString(Const.PROFILE\_IMG\_BG\_URL);

txtUserScreenName = (TextView) findViewById(R.id.txtUserScreenName);

txtUserScreenName.setText(userName);

txtUserTagLine = (TextView) findViewById(R.id.txtUserTagLine);

txtUserTagLine.setText("@" + screenName);

txtFollowersCount = (TextView) findViewById(R.id.txtFollowersCount);

txtFollowersCount.setText(followersCount);

txtFollowingCount = (TextView) findViewById(R.id.txtFollowingCount);

txtFollowingCount.setText(followingCount);

txtTweetCount = (TextView) findViewById(R.id.txtTweetCount);

txtTweetCount.setText(tweetCount);

vfd.ivUserProfileImg = (ImageView) findViewById(R.id.ivUserProfileImg);

Picasso.with(getApplicationContext()).load(profileImgUrl).into(vfd.ivUserProfileImg);

setIvUserProfileBGImg((ImageView) findViewById(R.id.ivUserProfileBGImg));

Picasso.with(getApplicationContext()).load(profileImgBgUrl).into(getIvUserProfileBGImg());

}

public ImageView getIvUserProfileBGImg() {

return ivUserProfileBGImg;

}

public void setIvUserProfileBGImg(ImageView ivUserProfileBGImg) {

this.ivUserProfileBGImg = ivUserProfileBGImg;

}

}

package com.example.usr.viewsapplication;

import android.content.Intent;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.util.Log;

import android.widget.Toast;

import com.twitter.sdk.android.Twitter;

import com.twitter.sdk.android.core.Callback;

import com.twitter.sdk.android.core.Result;

import com.twitter.sdk.android.core.TwitterException;

import com.twitter.sdk.android.core.TwitterSession;

import com.twitter.sdk.android.core.identity.TwitterLoginButton;

import com.twitter.sdk.android.core.models.User;

import retrofit2.Call;

public class LoginActivity extends AppCompatActivity {

private TwitterLoginButton loginButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

loginButton = (TwitterLoginButton) findViewById(R.id.twitter\_login\_button);

loginButton.setCallback(new Callback<TwitterSession>() {

@Override

public void success(Result<TwitterSession> result) {

TwitterSession session = result.data;

String msg = "@" + session.getUserName() + " logged in!" ;

Toast.makeText(getApplicationContext(), msg, Toast.LENGTH\_LONG).show();

Call<User> user = Twitter.getApiClient(session).getAccountService().verifyCredentials(true, false);

user.enqueue(new Callback<User>() {

@Override

public void success(Result<User> result) {

putInfo(result);

}

@Override

public void failure(TwitterException exception) {

Log.d("TwitterKit", "Login with Twitter failure", exception);

}

});

}

@Override

public void failure(TwitterException exception) {

Log.d("TwitterKit", "Login with Twitter failure", exception);

}

});

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

loginButton.onActivityResult(requestCode, resultCode, data);

}

public void putInfo(Result<User> result){

Intent intent = new Intent(LoginActivity.this, MainActivity.class);

User userInfo = result.data;

intent.putExtra(Const.USER\_NAME, userInfo.name );

intent.putExtra(Const.SCREEN\_NAME, userInfo.screenName);

intent.putExtra(Const.LOCATION, userInfo.location);

intent.putExtra(Const.FOLLOWING\_COUNT, String.valueOf(userInfo.friendsCount));

intent.putExtra(Const.FOLLOWERS\_COUNT, String.valueOf(userInfo.followersCount));

intent.putExtra(Const.TWEET\_COUNT, String.valueOf(userInfo.statusesCount));

intent.putExtra(Const.PROFILE\_IMG\_URL, userInfo.profileImageUrl.replace("", ""));

//intent.putExtra(Const.PROFILE\_IMG\_BG\_URL, userInfo.profileBannerUrl.replace("", ""));

intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);

startActivity(intent);

}

}

package com.example.usr.viewsapplication;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.EditText;

import android.widget.ImageButton;

public class SearchActivity extends AppCompatActivity implements View.OnClickListener {

private EditText et\_search;

private ImageButton search\_btn;

private ImageButton clear\_btn;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_search);

et\_search = (EditText) findViewById(R.id.et\_search);

search\_btn = (ImageButton) findViewById(R.id.search\_btn);

search\_btn.setOnClickListener(this);

clear\_btn = (ImageButton) findViewById(R.id.clear\_btn);

clear\_btn.setOnClickListener(this);

}

@Override

public void onClick(View v) {

switch (v.getId()){

case R.id.search\_btn:

Bundle bundle = new Bundle();

bundle.putString(Const.SEARCH\_TEXT, et\_search.getText().toString());

SearchTimeLineFragment searchTimeLineFragment = new SearchTimeLineFragment();

searchTimeLineFragment.setArguments(bundle);

getSupportFragmentManager()

.beginTransaction()

.replace(R.id.search\_timeline, searchTimeLineFragment)

.commit();

break;

case R.id.clear\_btn:

et\_search.setText("");

break;

}

}

}

package com.example.usr.viewsapplication;

import android.os.Bundle;

import android.support.v4.app.ListFragment;

import android.support.v4.widget.SwipeRefreshLayout;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Toast;

import com.twitter.sdk.android.core.Callback;

import com.twitter.sdk.android.core.Result;

import com.twitter.sdk.android.core.TwitterException;

import com.twitter.sdk.android.core.models.Tweet;

import com.twitter.sdk.android.tweetui.TimelineResult;

import com.twitter.sdk.android.tweetui.TweetTimelineListAdapter;

import com.twitter.sdk.android.tweetui.UserTimeline;

public class TimeLineFragment extends ListFragment {

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

View rootView = inflater.inflate(R.layout.timeline\_fragment, container, false);

String screenName = getArguments().getString(Const.SCREEN\_NAME);

final UserTimeline userTimeline = new UserTimeline

.Builder()

.includeRetweets(true)

.includeReplies(true)

.screenName(screenName)

.build();

final TweetTimelineListAdapter adapter = new TweetTimelineListAdapter.Builder(getActivity())

.setTimeline(userTimeline)

.setViewStyle(R.style.tw\_\_TweetDarkWithActionsStyle)

.build();

setListAdapter(adapter);

final SwipeRefreshLayout swipeLayout = (SwipeRefreshLayout) rootView.findViewById(R.id.swipe\_layout);

swipeLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {

@Override

public void onRefresh() {

swipeLayout.setRefreshing(true);

adapter.refresh(new Callback<TimelineResult<Tweet>>() {

@Override

public void success(Result<TimelineResult<Tweet>> result) {

swipeLayout.setRefreshing(false);

}

@Override

public void failure(TwitterException exception) {

Toast.makeText(getActivity().getApplicationContext(), "Failed to retrieve timeline",Toast.LENGTH\_LONG);

}

});

}

});

return rootView;}}

package com.example.usr.viewsapplication;

import android.animation.AnimatorSet;

import android.animation.ObjectAnimator;

import android.app.Activity;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.drawable.BitmapDrawable;

import android.graphics.drawable.Drawable;

import android.view.Gravity;

import android.view.MotionEvent;

import android.view.View;

import android.view.ViewGroup;

import android.view.animation.AccelerateInterpolator;

import android.view.animation.OvershootInterpolator;

import android.widget.FrameLayout;

public class FloatingActionButton extends View {

Context context;

Paint mButtonPaint;

Paint mDrawablePaint;

Bitmap mBitmap;

boolean mHidden = false;

public FloatingActionButton(Context context) {

super(context);

this.context = context;

init(Color.WHITE);

}

public void setFloatingActionButtonColor(int FloatingActionButtonColor) {

init(FloatingActionButtonColor);

}

public void setFloatingActionButtonDrawable(Drawable FloatingActionButtonDrawable) {

mBitmap = ((BitmapDrawable) FloatingActionButtonDrawable).getBitmap();

invalidate();

}

public void init(int FloatingActionButtonColor) {

setWillNotDraw(false);

setLayerType(View.LAYER\_TYPE\_SOFTWARE, null);

mButtonPaint = new Paint(Paint.ANTI\_ALIAS\_FLAG);

mButtonPaint.setColor(FloatingActionButtonColor);

mButtonPaint.setStyle(Paint.Style.FILL);

mButtonPaint.setShadowLayer(10.0f, 0.0f, 3.5f, Color.argb(100, 0, 0, 0));

mDrawablePaint = new Paint(Paint.ANTI\_ALIAS\_FLAG);

invalidate();

}

@Override

protected void onDraw(Canvas canvas) {

setClickable(true);

canvas.drawCircle(getWidth() / 2, getHeight() / 2, (float) (getWidth() / 2.6), mButtonPaint);

canvas.drawBitmap(mBitmap, (getWidth() - mBitmap.getWidth()) / 2,

(getHeight() - mBitmap.getHeight()) / 2, mDrawablePaint);

}

@Override

public boolean onTouchEvent(MotionEvent event) {

if (event.getAction() == MotionEvent.ACTION\_UP) {

setAlpha(1.0f);

} else if (event.getAction() == MotionEvent.ACTION\_DOWN) {

setAlpha(0.6f);

}

return super.onTouchEvent(event);

}

public void hideFloatingActionButton() {

if (!mHidden) {

ObjectAnimator scaleX = ObjectAnimator.ofFloat(this, "scaleX", 1, 0);

ObjectAnimator scaleY = ObjectAnimator.ofFloat(this, "scaleY", 1, 0);

AnimatorSet animSetXY = new AnimatorSet();

animSetXY.playTogether(scaleX, scaleY);

animSetXY.setInterpolator(new AccelerateInterpolator());

animSetXY.setDuration(100);

animSetXY.start();

mHidden = true;

}

}

public void showFloatingActionButton() {

if (mHidden) {

ObjectAnimator scaleX = ObjectAnimator.ofFloat(this, "scaleX", 0, 1);

ObjectAnimator scaleY = ObjectAnimator.ofFloat(this, "scaleY", 0, 1);

AnimatorSet animSetXY = new AnimatorSet();

animSetXY.playTogether(scaleX, scaleY);

animSetXY.setInterpolator(new OvershootInterpolator());

animSetXY.setDuration(200);

animSetXY.start();

mHidden = false;

}

}

public boolean isHidden() {

return mHidden;

}

static public class Builder {

private FrameLayout.LayoutParams params;

private final Activity activity;

int gravity = Gravity.BOTTOM | Gravity.RIGHT; // default bottom right

Drawable drawable;

int color = Color.WHITE;

int size = 0;

float scale = 0;

public Builder(Activity context) {

scale = context.getResources().getDisplayMetrics().density;

// The calculation (value \* scale + 0.5f) is a widely used to convert to dps to pixel units

// based on density scale

// see developer.android.com (Supporting Multiple Screen Sizes)

size = (int) (72 \* scale + 0.5f); // default size is 72dp by 72dp

params = new FrameLayout.LayoutParams(size, size);

params.gravity = gravity;

this.activity = context;

}

/\*\*

\* Sets the gravity for the FAB

\*/

public Builder withGravity(int gravity) {

this.gravity = gravity;

return this;

}

/\*\*

\* Sets the margins for the FAB in dp

\*/

public Builder withMargins(int left, int top, int right, int bottom) {

params.setMargins((int) (left \* scale + 0.5f), (int) (top \* scale + 0.5f),

(int) (right \* scale + 0.5f), (int) (bottom \* scale + 0.5f));

return this;

}

/\*\*

\* Sets the FAB drawable

\* @param drawable

\*/

public Builder withDrawable(final Drawable drawable) {

this.drawable = drawable;

return this;

}

/\*\*

\* Sets the FAB color

\*/

public Builder withButtonColor(final int color) {

this.color = color;

return this;

}

/\*\*

\* Sets the FAB size in dp

\*/

public Builder withButtonSize(int size) {

size = (int) (size \* scale + 0.5f);

params = new FrameLayout.LayoutParams(size, size);

return this;

}

public FloatingActionButton create() {

final FloatingActionButton button = new FloatingActionButton(activity);

button.setFloatingActionButtonColor(this.color);

button.setFloatingActionButtonDrawable(this.drawable);

params.gravity = this.gravity;

ViewGroup root = (ViewGroup) activity.findViewById(android.R.id.content);

root.addView(button, params);

return button;

}

}

}