УДК 004.413

Гребнева Д.М.
к.п.н., доцент кафедры информационных технологий
Филиал Российского государственного профессионально-педагогического университета в г. Нижний Тагил,
г. Нижний Тагил, Россия

РАЗРАБОТКА СИСТЕМЫ ОПОВЕЩЕНИЙ СТУДЕНТОВ ВУЗА ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ

Аннотация

В статье описан процесс решения задачи создания автоматизированной системы оповещений студентов ВУЗа о значимых событиях. Представлены этапы разработки новостного мобильного приложения с возможностью отправки push-уведомлений. Подробно описан процесс написания программного кода и оформление интерфейса мобильного приложения в среде разработки Android Studio.

Ключевые слова: мобильное приложение, Android, система оповещений, push-уведомления.

Grebneva D.M.
Ph. D., associate Professor of information technologies Department Branch of the Russian State Vocational and Pedagogical University in Nizhny Tagil
Nizhny Tagil, Russia

DESIGN OF THE STUDENT'S ALERTS SYSTEM FOR MOBILE DEVICES

Abstract

The article describes the process of solving the task of creating an automated system for alerting university students about significant events. The stages of developing a mobile news application with the ability to send push notifications are presented. A detailed description of the process of writing code and the design of the interface of the mobile application in the development environment of Android Studio.

Keywords: mobile application, Android, system of notifications, push-notifications.

В современных условиях трудно представить себе человека без мобильного телефона, планшетного компьютера, смартфона или любого другого портативного мультимедийного устройства. Мы привыкли к тому, что такое устройство всегда под рукой, и не только как средство общения. Как правило, мобильное устройство имеет много полезных функций, таких как калькулятор, органайзер, конвертер, календарь, часы и многое другое.

Для расширения функциональности смартфонов разрабатываются специальные мобильные приложения, под которыми понимают программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах [8]. В связи с доступностью и всеобщим распространением смартфонов и планшетных компьютеров, мобильные приложения активно используются в различных сферах деятельности человека. Например, все большую популярность приобретают системы автоматического оповещения разных категорий населения о тех или иных событиях. Данные системы используются в маркетинговых кампаниях, для напоминания клиентам о записи на прием у врача, о приближение какого-либо значимого события и др. В процессе обучения в Вузе также нередко возникает необходимость оперативно оповестить студентов о чем-либо: о старостате, о ближайших спортивных или научных мероприятиях и др.

Автоматически отправлять оповещения можно с помощью мобильных приложений. На сегодняшний день наиболее популярными являются мобильные приложения под операционной системой Android.

Создадим приложение, которое будет получать RSS ленту с сайта Нижнетагильского государственного социально-педагогического института. Для программирования будем использовать среду разработки AndroidStudio. Выберем File-> New-> New Project (рис. 1).

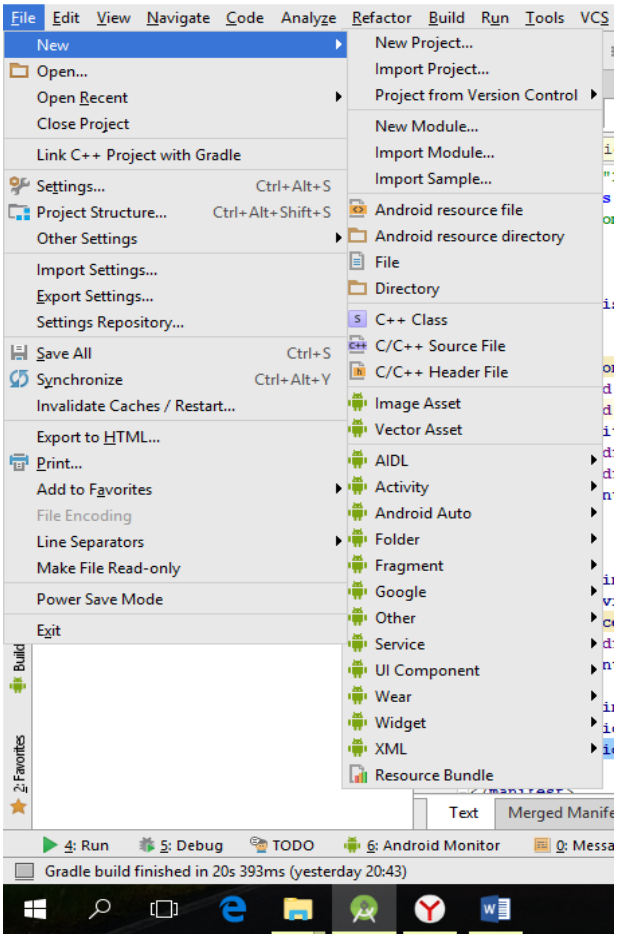

Рис. 1. Создание нового проекта

В открывшемся диалоге в качестве имени проекта вводим RSSFeed, нажимаем кнопку Next (рис. 2).

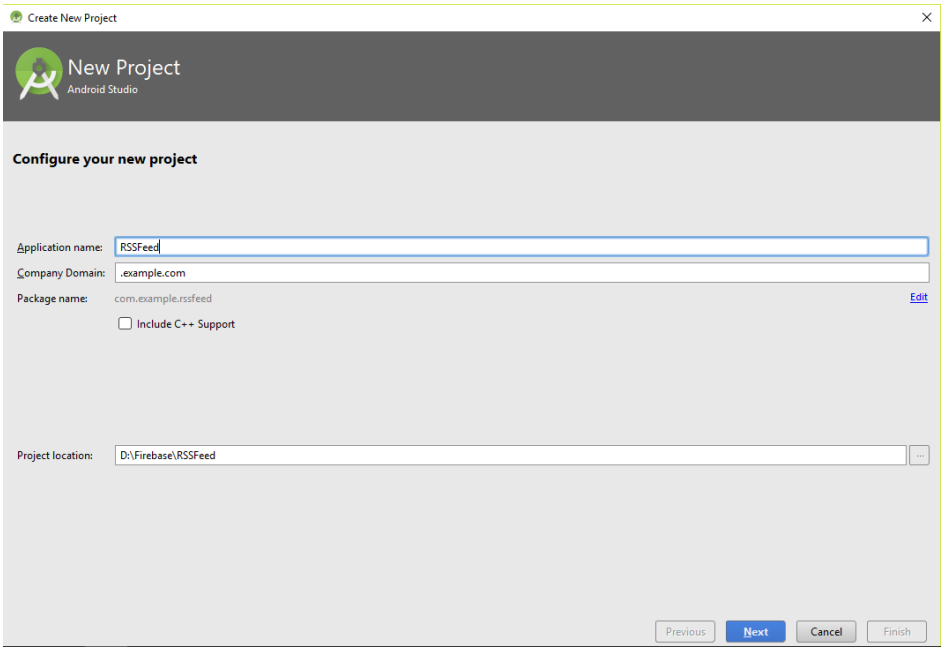

Рис. 2. Создание нового проекта

В следующем окне нам предлагают выбрать целевую платформу. Выбираем Android 4.1, нажимаем кнопку Next, оставляя остальные настройки по умолчанию нажимаем еще раз Next и Finish.

Создадим java класс RSSFeed.java. Обратите внимание, созданный нами класс, наследующий Activity фактически представляет собой визуальный пользовательский интерфейс – окно (рис. 3).

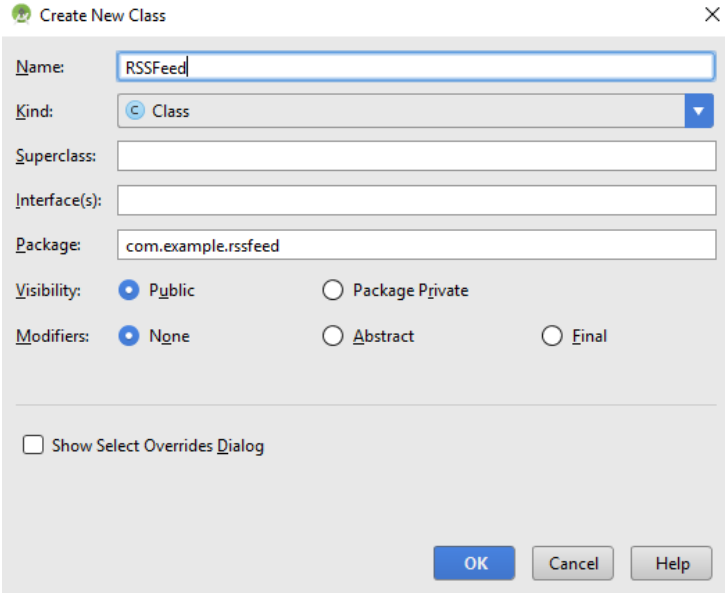

Рис. 3. Создание класса RSSFeed.java

На данном этапе наш проект состоит из трех файлов: RSSFeed.java, который содержит исходный код класса RSSFeed; файл разметки res/layout/activity_main.xml, содержащий информацию о дизайне

пользовательского интерфейса и Android UI компонентах; файл манифеста AndroidManifest.xml, в котором хранится основные параметры проекта (название пакета, запускаемая при старте программы деятельность (Activity класс), компоненты приложения, процессы, разрешения, минимально необходимый уровень API).

Для начала настроим интерфейс нашего будущего приложения. Щелкнем два раза в дереве проекта по файлу res/layout/activity_main.xml. В центральной части Android Studio откроется визуальный редактор. В моем случае на окне уже имеется компонент LinearLayout, у которого задана вертикальная ориентация. Под окном редактирования присутствуют вкладки Design и Text, позволяющие переключиться от визуального редактора к редактору xml файла (рис. 4).

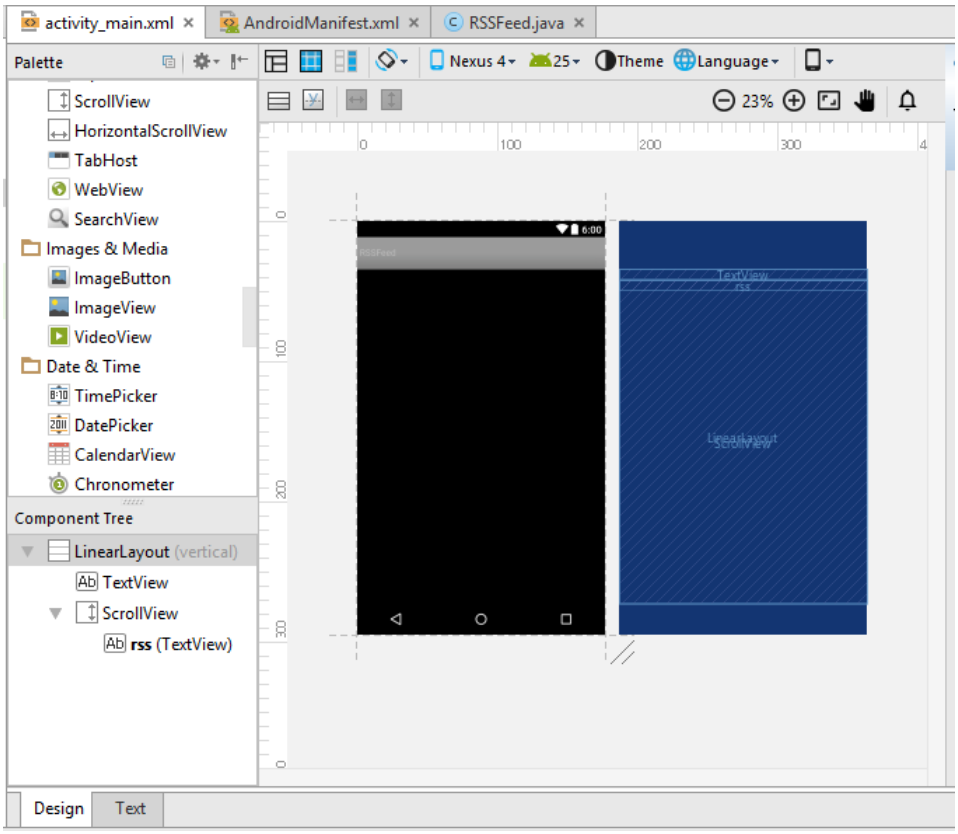

Рис. 4. Файл разметки activity_main.xml

Программный код файла разметки activity_main.xml приведен на рис. 5.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
<TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text=""
        android:autoLink="web"/>
<ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

<TextView android:id="@+id/rss"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
</ScrollView>
</LinearLayout>
```

Рис. 5. Код файла разметки activity_main.xml

Настройка AndroidManifest.xml. RSS ленту будем получать с удаленного сервера, поэтому приложение должно иметь соответствующие разрешения. Щелкнем в дереве проектов по файлу AndroidManifest.xml, перейдем на вкладку Permissions, нажмем на кнопку Add, выберем Uses Permission, нажмем ОК. В поле Name введем android.permission.INTERNET. В результате этих действий в файле AndroidManifest.xml появится строка:

```
<uses-permission
android:name="android.permission.INTERNET"/>
```

После этого приложение может работать с сокетами и получать информацию из сети. Файл AndroidManifest.xml должен иметь вид как на рис. 6.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.rssfeed">
<uses-permission android:name="android.permission.INTERNET"/>
<application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name" >
<activity
            android:name=".RSSFeed"
            android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>

</manifest>
```

Рис. 6. Код файла AndroidManifest.xml

Для того, чтобы организовать оповещение в виде RSS ленты, необходимо импортировать несколько классов, которые изображены на рис. 7.

- javax.xml.parsers.SAXParser;
- javax.xml.parsers.SAXParserFactory;
- org.xml.sax.InputSource;
- org.xml.sax.XMLReader;
- org.xml.sax.helpers.DefaultHandler.

Рис.7. Классы импорта

Внутри класса RSSFeed определим строковую переменную, куда будем загружать RSS ленту, перед заголовком метода onCreate.

String rssResult = "";

Добавим в конец этого метода переменную rss, связанную со вторым элементом TextView. С помощью этой переменной мы будем отображать информацию на экране.

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

Загрузим из интернет RSS ленту, для этого воспользуемся классом URL, в конструкторе которого укажем адрес файла с лентой, например https://www.ntspi.ru/about_academy/academy_news/rss/

В процессе чтения может произойти ошибка ввода/вывода, поэтому Java требует помещать такие потенциально опасные команды в блок try/catch.

```
    try {
    URL rssUrl = new
URL("https://www.ntspi.ru/about_academy/academy_news/rss/");
    } catch (IOException e) {rss.setText(e.getMessage());}
```

Рис.8. Код блока try/catch

Для того, чтобы программное приложение исправно функционировало, необходимо подключить дополнительные библиотеки: android.widget.TextView/ java.net.URL /java.io.IOException.

Далее, мы воспользовались статическим методом newInstance и создали объект SAXParserFactory.

```
SAXParserFactory factory = SAXParserFactory.newInstance();
Создадим объект SAXParser с помощью newSAXParser:
SAXParser saxParser = factory.newSAXParser();
```

Рис.9. Код объекта SAXParserFactory

Таким образом, мы с помощью объекта URL установили связь с удаленной страницей, а для организации считывания потока данных создали объект InputSource и открыли поток данных для чтения с помощью метода openStream():

InputSourceinputSource = newInputSource(rssUrl.openStream());

Созданный таким образом объект мы передаем XMLReader для разбора данных

xmlReader.parse(inputSource);

После этой команды происходит считывание и парсинг файла файла RSS ленты, в процессе которого внутри методов класса RSSHandler заполняется данными строка rssResult. Все что нам осталось, передать текст из этой строки в элемент TextView на экране.

Класс AsyncTask позволяет выполнять фоновые операции в отдельном потоке и публиковать результаты этих операций для UI thread.

Когда выполняется асинхронная задача, то она проходит через 4 шага:

1. OnPreExecute(), вызывается в UI thread перед тем, как задача начнет выполняться.

2. DoInBackground(Params...), вызывается в фоновом потоке задачи сразу после того, как завершит работу onPreExecute().

3. OnProgressUpdate(Progress...), вызывается в UI thread после вызова publishProgress(Progress...). Этот метод используется для отображения прогресса любой формы в интерфейсе пользователя.

4. OnPostExecute(Result), вызывается в UI thread после завершения фоновых вычислений. Результат фонового вычисления передается на этом шаге как параметр.

Таким образом, в результате работы мы считали удаленный файл, провели его разбор, в процессе которого сформировали строку rssResult, а затем отобразили эту строку на экране с помощью TextView.

Сервис Firebase доступен в меню Tools. Открываем и видим все инструменты, доступные для добавления в наше приложение. Выбираем его в списке. Открылось меню со списком шагов по внедрению сервиса в проект (рис. 10).

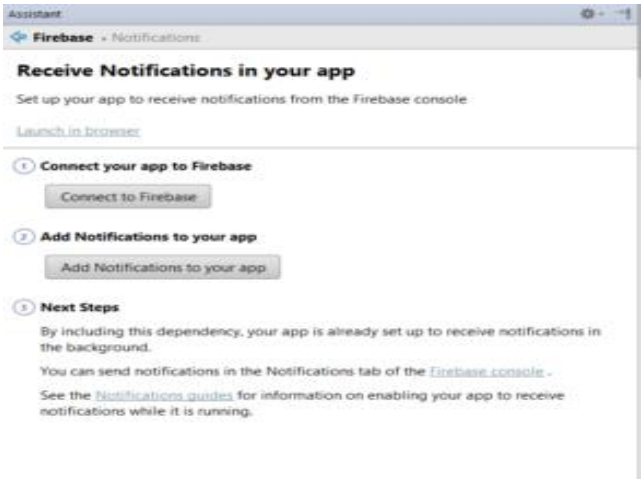

Рис. 10. Меню Notification

На первом этапе необходимо установить связь приложения с Firebase.

Далее осуществляется подключение библиотек Notifications в приложение.

На следующем этапе мы уже можем видеть информацию о том, что приложение уже настроено для получения уведомлений в фоновом режиме.

Для проверки исправности работы мобильного приложения, необходимо перейти в консоль Firebase и отправить уведомление.

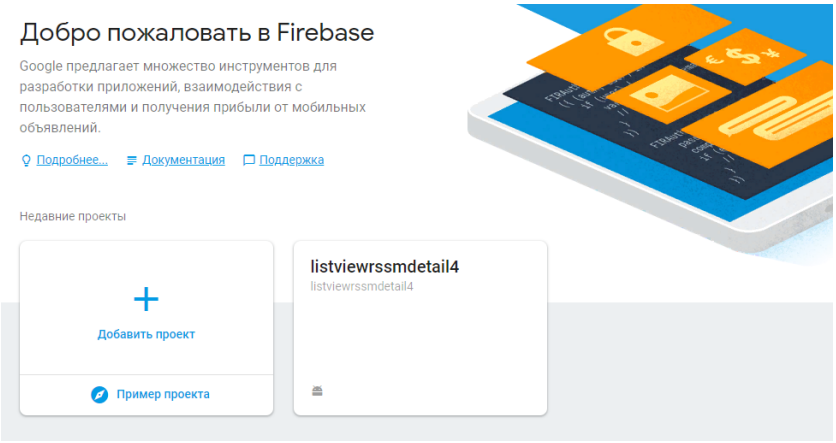

Рис. 11. Консоль Firebase

Здесь можно создать новое сообщение. Вводим его текст. Можно ввести заголовок, но это необязательно, он не будет показан на устройстве (рис. 12).

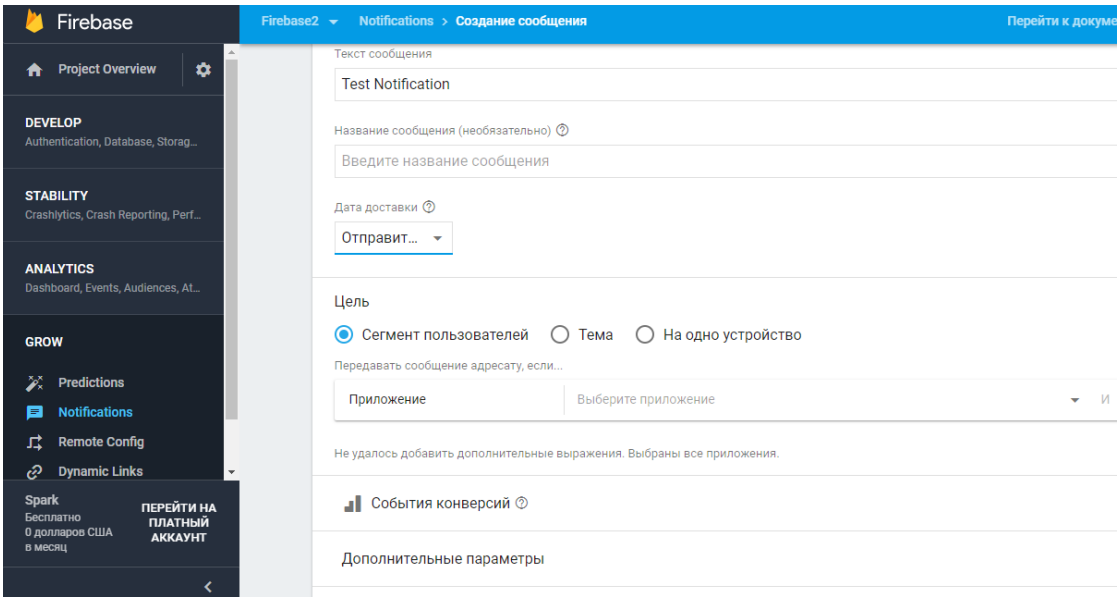

Рис.12. Меню отправки сообщений

Указываем дату доставки сообщения – можно отправить сейчас или запланировать на срок до одного месяца. Указываем таргетинг – можно отправить уведомление на все устройства с установленным приложением или только на определенные, на основании токена.

Создадим в приложении такой класс сервиса, унаследованный от класса FirebaseMessagingService (рис. 12).

```
import android.app.NotificationManager;
import android.content.Context;
import android.content.Intent;
import android.graphics.BitmapFactory;
import android.media.RingtoneManager;
import android.net.Uri;
import android.support.v4.app.NotificationCompat;
import com.google.firebase.messaging.FirebaseMessagingService;
import com.google.firebase.messaging.RemoteMessage;

public class MyFirebaseMessagingService extends FirebaseMessagingService {

    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {

        sendNotification(remoteMessage.getNotification().getBody());

    }
    private void sendNotification(String messageBody) {
        Intent intent = new Intent(this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

        Uri defaultSoundUri= RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);

        NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this)
                .setSmallIcon(R.drawable.ic_launcher)
                .setLargeIcon(BitmapFactory.decodeResource(this.getResources(), R.drawable.ic_launcher))
                .setContentTitle(this.getString(R.string.app_name))
                .setContentText(messageBody)
                .setAutoCancel(true)
                .setSound(defaultSoundUri);

        NotificationManager notificationManager =
                (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

        notificationManager.notify(0, notificationBuilder.build());
    }
}
```

Рис. 13. Описание класса FirebaseMessagingService

Метод onMessageReceived выполняется при получении сообщения с сервера и получает на вход объект класса RemoteMessage, который хранит в себе всю информацию о сообщении, доступную через геттеры. Вот все поля, которые имеются в этом классе. В данном случае нас интересует только метод внутреннего класса Notification getBody, который содержит текст уведомления. Мы его будем передавать методу sendNotification, в котором мы создаем уведомление для отображения на устройстве. Здесь создаем интент, который будет выполняться при нажатии на уведомление. В данном случае у нас будет открываться главное активити, но можно прописать здесь открытие любого другого активити, или другое действие, переход по внешней ссылке, например. Созданием уведомления занимается notificationBuilder, он устанавливает маленькую иконку, которая отображается в панели уведомлений, большую иконку для открытого уведомления, заголовок уведомления, текст уведомлении – его мы передаем в методе onMesageReceived, далее методом setAutoCancel устанавливаем уничтожение уведомления при касании, а методом setSound задаем сигнал для уведомления. Его мы получаем здесь из класса RingtoneManager с флагом, указывающим тим сигнала. Это может быть notification – сигнал уведомления, alarm – сигнал будильника, rington – сигнал звонка. Но лучше не злоупотреблять терпением пользователей и оставить по умолчанию звук уведомления, или вообще без звука. Уведомление построили, теперь для его отправки используем NotificationManager – это системный сервис Android, который управляет всеми уведомлениями. Экземпляр NotificationManager создается при помощи вызова метода getSystemService(), а затем, когда надо показать уведомление пользователю, вызывается метод

notify(), которому мы передаем идентификатор (в данном случае не используется и равен 0) и созданное уведомление.

Перед тем, как проверить работу сервиса, его нужно прописать в манифесте (рис. 14).

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.rssfeed"
>


<uses-permission android:name="android.permission.INTERNET"/>


<application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name" >
<activity
            android:name=".RSSFeed"
            android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<service
            android:name=".MyFirebaseMessagingService">
<intent-filter>
<action android:name="com.google.firebase.MESSAGING_EVENT"/>
</intent-filter>
</service>
</application>

</manifest>
```

Рис. 7. Описание сервиса в манифесте

Теперь запускаем приложение на устройстве и идем в консоль Firebase для создания и отправки тестового уведомления. Уведомление приходит на устройство и тогда, когда приложение активно.

Мобильное приложение было разработано для смартфонов под управлением AndroidOS 4.1 и выше.

После запуска приложения, откроется главное окно, в котором находится стартовая кнопка загрузки ленты новостей (рис. 15).

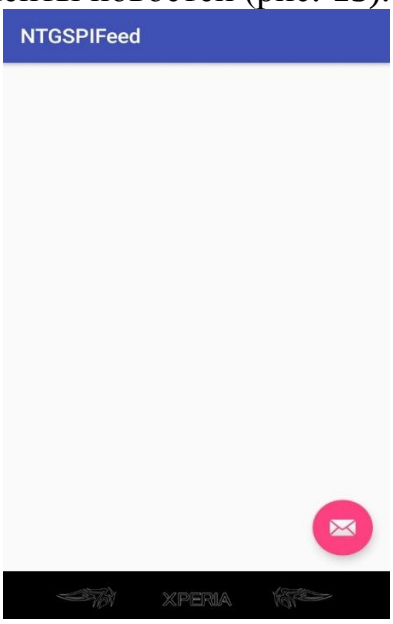

Рис. 8. Главное окно приложения

После нажатия стартовой кнопки, откроется следующее окно приложения с новостной лентой (RSS лента) сайта Нижнетагильского Социально-Педагогического Института (рис. 16).

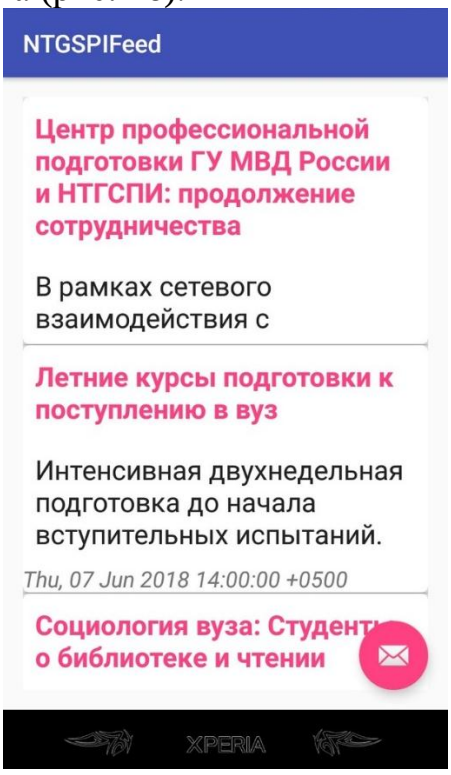

Рис. 9. RSS лента НТГСПИ

Переход по новостям, для прочтения представлен следующим. Мы можем увидеть дату публикации, вплоть до точного времени, а также сам текст новостей (рис. 17).

Рис. 10. Переход для прочтения новостей

С помощью сервиса Firebase было отправлено тестовое уведомление, которое было получено тестируемыми смартфонами. Уведомления работают без ошибок (рис. 18).

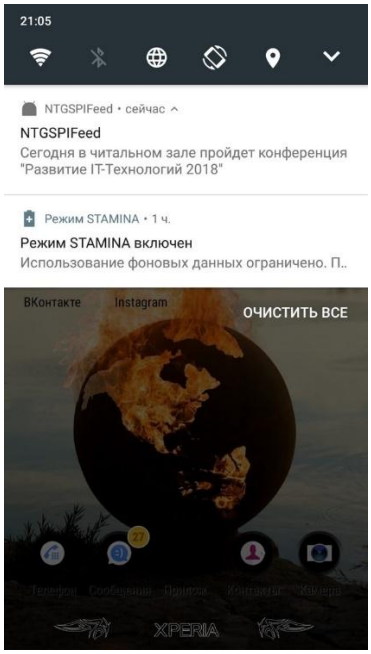

Рис. 11. Тестовое уведомление

Тестируемые смартфоны:

– Sony Xperia XA – версияOS Android 7.0;
– Sony Xperia XA1 – версия OS Android 7.0;
– Samsung Galaxy J2 Prime –версия OS Android 6.0;
– Huawei Honor 5A – версия OS Android 5.1;
– Lenovo A2010-a – версия OS Android 5.1

Разработанная система может применятся для оповещения студентов ВУЗа о значимых событиях. Мобильное приложение оповещений студентов Вуза для ОС Android разработано средствами технологий XML и Java в интегрированной среде разработки Android Studio.

ЛИТЕРАТУРА

1. Блог по разработке Android [Электронный ресурс] URL: http: //android-zone. Info (дата обращения: 29.09.18).

2. Блог разработчика Android [Электронный ресурс] URL: http: //androidengineer. ru/ (дата обращения: 29.09.18).

3. Вейл Э. HTML5. Разработка приложений для мобильных устройств / Э. Вейл. Спб.: Питер, 2014. – 930 с.

4. Голощапов А. Л. Google Android: программирование для мобильных устройств / А. Л. Голощапов. СПб.: БХВ-Петербург, 2011. – 448 с.

5. Дейтел П. Android для разработчиков / П. Дейтел. СПб.: Питер, 2015. – 384 с.

6. Дизайн для Android. [Электронный ресурс] URL: http://developer.alexanderklimov.ru/android/design/basic.php (дата обращения: 15.03.18).

7. Майер Р. Программирование приложений для планшетных компьютеров и смартфонов / Р. Майер. М.:Эксмо, 2011. – 672 с.

8. МакГрат М. Создание приложений для Android для начинающих / М. МакГрат. – М.: Эксмо, 2016. – 192 с.