УДК 004

Внедрение приложения файлового менеджера для любого приложения на Android смартфон

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассмотрен способ внедрения файлового менеджера для любого приложения на Android устройстве. Приложение будет разработано в среде разработки Android Studio на языке программирования JavaScript. Практическим результатом является внедрение файлового менеджера в приложение, с возможностью создания, редактирования и удаления файлов и папок.
Ключевые слова: Android, Android Studio, приложение, File manager.

Applications for any application on an Android smartphone

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes how to implement a file manager for any application for Android devices. The application will be developed in the Android Studio development environment in the JavaScript programming language. The practical result is the introduction of a file manager into the application, with the ability to create, edit and delete files and folders.
Keywords: Android, Android Studio, application, File manager.

Файловый менеджер — программа, с помощью которой пользователь имеет доступ к файловой системе и файлами. Файловый менеджер позволяет выполнять наиболее простые операции над файлами — создание, открытие/проигрывание/просмотр, редактирование, перемещение, переименование, копирование, удаление, поиск файлов.

Цель данной статьи является реализация пользовательских функций, с помощью которых можно видеть весь список файлов и каталогов, создавать файлы и каталоги, а также удалять его.

Сьянов С.Л., Хасанов Р.А. в своей статье предоставили алгоритмы генерации и взаимодействия мультимедиа данных [1]. Так же Ибляминов Ф.Ф., Кутепова Л.М., Тимербулатова И.Р., Яшина Н.Г охарактеризовали некоторые инструменты, используемые для создания мобильных приложений, в том числе и Android Studio [2]. В статье Захаров В.Б.,

Мальковский М.Г., Мостяев А.И. рассмотрели перенос нескольких приложений с Windows на мобильные платформы [3].

Исходя из требований клиента или требований функциональных возможностей приложения для Android, иногда может возникнуть необходимость показывать определенные файлы, т.е. только изображения, документы или PDF-файлы. Это можно сделать с помощью приложения «FileViewer» по умолчанию, но существуют некоторые ограничения. Поэтому в таких случаях всегда желательно использовать индивидуальный подход для изучения конкретных файлов в соответствии с необходимостью.

Чтобы внедрить пользовательский проводник в Android приложение, сначала добавьте разрешения в файл «AndroidManifest.xml» этого приложения (рис.1).

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

Рисунок 1 – Добавление разрешений

Для просмотров файлов и каталогов используем виджет «ListView». Помещаем его в XML-файл макета приложения (рис.2).

```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/rl_lvListRoot"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</ListView>
```

Рисунок 2 – Добавление виджета

Создаем файл шаблона для разметки элемента «ListView». Здесь создается пользовательский макет для отображения значка, имени и даты изменения файла или каталога (рис.3).

```
</RelativeLayout  xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >

    <CheckBox
        android:id="@+id/lr_cbCheck"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/lr_ivFileIcon"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:focusable="false" />

    <ImageView
        android:id="@+id/lr_ivFileIcon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@+id/lr_cbCheck"
        android:src="@drawable/ic_launcher"
        android:clickable="false"
        android:focusable="false" />

    <TextView
        android:id="@+id/lr_tvFileName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_toRightOf="@+id/lr_ivFileIcon"
        android:text="TextView"
        android:clickable="false"
        android:focusable="false" />

    <TextView
        android:id="@+id/lr_tvdate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/lr_tvFileName"
        android:layout_marginLeft="10dp"
        android:layout_toRightOf="@+id/lr_ivFileIcon"
        android:text="TextView"
        android:focusable="false"
        android:clickable="false" />

</RelativeLayout>
```

Рисунок 3 – Создание макета

Создаём пользовательский адаптер, расширив класс «BaseAdapter» для привязки с «ListView» (рис.4).

```
public class ListAdapter extends BaseAdapter {
    private List<String> m_item;
    private List<String> m_path;
    public ArrayList<Integer> m_selectedItem;
    Context m_context;
    Boolean m_isRoot;
    public ListAdapter(Context p_context,List<String> p_item, List<String> p_path,Boolean p_isRoot) {
        m_context=p_context;
        m_item=p_item;
        m_path=p_path;
        m_selectedItem=new ArrayList<Integer>();
        m_isRoot=p_isRoot;
    }

    @Override
    public int getCount() {
        return m_item.size();
    }

    @Override
    public Object getItem(int position) {
        return m_item.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(final int p_position, View p_convertView, ViewGroup p_parent)
    {...}

    class ViewHolder
    {...}

    private int setFileImageType(File m_file)
    {
        int m_lastIndex=m_file.getAbsolutePath().lastIndexOf(".");
        String m_filepath=m_file.getAbsolutePath();
        if (m_file.isDirectory())
            return R.drawable.folderopened_yellow;
        else
        {...}
    }

    String getLastDate(int p_pos)
    {...}
}
```

Рисунок 4 – Создание пользовательского адаптера

Далее нужно вывести все каталоги и файлы из корневой папки внешнего хранилища (рис.5).

```
private String m_root=Environment.getExternalStorageDirectory().getPath();

public void getDirFromRoot(String p_rootPath)
        {
        m_item = new ArrayList<String>();
        Boolean m_isRoot=true;
        m_path = new ArrayList<String>();
        File m_file = new File(p_rootPath);
        if(!p_rootPath.equals(m_root))
        {
        m_item.add("../");
        m_path.add(m_file.getParent());
        }
        m_curDir=p_rootPath;
        Arrays.sort(m_filesArray);
        for(int i=0; i < m_filesArray.length; i++)
        {
        File file = m_filesArray[i];
        if(file.isDirectory())
        {
        m_item.add(file.getName());
        }
        else
        {
        m_files.add(file.getName());
        m_filesPath.add(file.getPath());
        }
        }
        for(String m_AddFile:m_files)
        {
        m_item.add(m_AddFile);
        }
        for(String m_AddPath:m_filesPath)
        {
        m_path.add(m_AddPath);
        }
        m_RootList.setOnItemClickListener(new OnItemClickListener() {

@Override
public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
        File m_isFile=new File(m_path.get(position));
        if(m_isFile.isDirectory())
        {
        getDirFromRoot(m_isFile.toString());
        }
        else
        {
        Toast.makeText(RootListActivity.this, "This is File", Toast.LENGTH_SHORT).show();
        }
        }
        });
        }
```

Рисунок 5 – Вывод каталогов и файлов

Добавляем возможность удалять файлы или каталоги (рис.6).

```
void deleteFile()
        {
        for(int m_delItem : m_listAdapter.m_selectedItem)
        {
        File m_delFile =new File(m_path.get(m_delItem));
        Log.d("file",m_path.get(m_delItem));
        boolean m_isDelete=m_delFile.delete();
        Toast.makeText(RootListActivity.this, "File(s) Deledted", Toast.LENGTH_SHORT).show();
        getDirFromRoot(m_curDir);
        }
        }
```

Рисунок 6 – Добавление функции удаления

Результатом данной статьи является внедрение файлового менеджера в приложение для устройства с Android. Данный проводник обладает всеми функциями присущим файловым менеджерам, такими как поиск, создание, редактирование, удаление папок и фалов.

Библиографический список

1. Сьянов С.Л., Хасанов Р.А. Использование языка Java для создания мультимедийных приложений на android // Информатика. Общие вопросы информатики. 2017. № 7. С. 110-134.
2. Ибляминов Ф.Ф., Кутепова Л.М., Тимербулатова И.Р., Яшина Н.Г. Инструментальные средства разработки мобильных приложений // Научное обозрение. 2017. № 9. С. 22-25.
3. Захаров В.Б., Мальковский М.Г., Мостяев А.И. Java или альтернативы? Опыт переноса приложений на платформу android // Научные труды SWORLD. 2015. №1. С. 15-27.