В статье рассматривается структура Android-приложений, которая включает четыре основных компонента. Кроме этого, в статье освещается назначение файла манифеста приложения, а также рассматривается жизненный цикл одного из наиболее важных компонентов Android приложения – Activity.

The structure of the Android-application consists of four main components that can be combined with each other – Activity, Service, Intent Receiver, and Content Provider. Information about which of these components are used in a particular application is stored in a special file – AndroidManifest.xml.

Due to this structure, the application can use the elements of another application, thus saving resources. For example, if your application needs to use a calculator and another application already has a suitable. You can call it to work, instead of developing own.

To enable the use of components of other applications, the system Android should be able to launches the application, which contains a required component and initialize it is objects. Therefore, the Android application do not has single point of entry to launch the application as opposed to other systems applications [1].

Activity is a visual interface for a user task. For example, an application for sending text messages may has one activity to display a list of contacts, the other activity to edit it and other activity to view old messages or change the settings. All of these activities form a single user interface, but do not depend on each other. Each of them has been implemented as a subclass of the base Activity class.

An application may consist of one activity and a few. The transition from one activity to another occurs when the current activity launches the next one.

Each activity has a window to display. Typically, the window is full screen size, but can be located over other windows. Activity may also have additional windows, for example, pop-up dialogs that require user, or windows that contain certain information which is required by the user.

Services do not have a visual interface, because they work in the background. For example, the service can play background music when the user is working in another application, or can receive data from the network or something to calculate and reports the result to activity when it is needed. Each service is inherited from base classes Service.

You can join an already running service (or run it, if it has not worked). You can interact with the service
through the management interface [1].

**Broadcast receivers** – are components that do nothing but receive and respond to broadcast messages [1]. Most of the messages generated by the system, for example, the message about changing the time zone, low battery, photographing or change the setting language. Applications can also send broadcast messages, for example, to communicate about downloading the required data to the device and availability for use it to other applications.

An application can have any number of broadcast receivers that are responding to any communication which it considers important. All broadcast receivers are inherited from a base class BroadcastReceiver.

Broadcast receivers do not have the user interface. However, they can launch the activity on response to their receiving information or they can use the class NotificationManager to alert the user.

**Content providers** – can make the data available to other applications. Data can be stored in the file system into SQLite database, or into any other form. Content providers are inherited from a base class ContentProvider, implements a standard set of methods for obtaining and storing data of a given type. However, the application can not call these methods directly. Instead, they use the object ContentResolver, which can interact with any of the content provider [2].

Activities, services, and broadcast receivers are activated with asynchronous message are called intent. **Intent** is the object of a class Intent, which is the content of the message [1]. For example, intent may submit a request activity to show the image of the user or allow the user to edit some text. The intent launches declared action for the broadcast receivers.

**Manifest** is XML-file which always has the name AndroidManifest.xml for all applications has the following functions [2]:

- Declares the name of a package of Java-application, which is a unique identifier for the application.
- Describes the application components – activities, services, broadcast receivers intent and content providers which are contained in the application. It calls the classes that implement each of the components, and declare their intentions.
- Declares the minimum Android API, which is required by the application.
- Lists the libraries that the application should be connected.

Elements manifest, application and uses-sdk are required. Other elements are used as needed.

The operating system controls the life cycle of the application. Activity can be in one of three states [1]:

- **Active or running** is an activity in the foreground of the screen (at the top of the stack).
- **Paused** is an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of your activity). A paused activity is completely alive, but can be killed by the system in extreme low memory situations.
- **Stopped** is an activity is completely obscured by another activity. It still retains all state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.

If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.

The following diagram shows the important state paths of an Activity (fig. 1) [3].
Moving from state to state, activity notifies about it by calling the following protected methods [1]:

- `onCreate(Bundle savedInstanceState)` – called when the activity is first created.
- `onStart()` – called when the activity becomes visible to the user.
- `onRestart()` – called when the activity has been stopped and is restarting again.
- `onResume()` – called when the activity starts interacting with the user.
- `onPause()` – called when the current activity is being paused and the previous activity is being resumed.
- `onStop()` – called when the activity is no longer visible to the user.
- `onDestroy()` – called before the activity is destroyed by the system (either manually or by the system to conserve memory).

In this paper, Android platform is introduced for developing mobile applications that are not inferior to the PC applications. It is expected that Android will offer the most ubiquitous solution for future mobile applications with the continuous improvement in mobile devices.

REFERENCES