# Пользовательский интерфейс

Данный небольшой цикл статей будет посвящен написанию собственных приложений для операционной системы Android. Мы последовательно разберемся с созданием программы и со всевозможными подводными камнями, которые могут поджидать пользователя, решившегося на такой смелый, но похвальный шаг. В первой статье мы поговорим о пользовательском интерфейсе.

Для начала, нужно определиться с целью создания своего приложения. Для примера мы будем создавать программу, которая при запуске отображает местоположение устройства на Google Maps. При этом, открыв меню, можно попасть к перечню дополнительных функций. Основной из них будет отображение остановок общественного транспорта и его маршрутов. Также попытаемся воплотить в жизнь поиск по карте подходящего маршрута и визуальное отображение общественного транспорта согласно расписания его движения. Главная цель написания данного приложения – это приобретение опыта для будущих программных изысков. Но всё по порядку.

Итак, как вывести на экран Google Maps? Компания Google уже предоставила владельцам Android-девайсов готовый интерфейс для работы со своими картами, потому давайте разберемся с основными видами интерфейса операционной системы Android. Существуют такие типы пользовательской оболочки:

Linear Layout – первая и основная схема отображения элементов пользовательского интерфейса. Всё содержимое отображается одно за другим, то есть линейно – просто и практично.

Relative Layout – также одна из основных схем отображения пользовательской оболочки, отличие заключается в том, что она позволяет выравнивать элементы по отношению друг к другу.

Table Layout – отображение пользовательской оболочки в виде таблицы. Структура описания данной схемы схожа с составлением таблицы в HTML при помощи тега <table>.

Grid View – схема для отображения элементов в виде сетки, которая состоит из столбцов и ячеек.

Tab Layout – достаточно распространенная и весьма удобная схема, суть которой заключается в создании закладок для разных функций.

List View – отображение элементов в виде списка с возможностью его прокрутки пользователем, также очень часто встречается во всех Android-программах. Самый простой пример – это «Список контактов» на Android-смартфоне.

Схемы интерйефса – это основа того, что будет видеть пользователь. Существует много различных «виджетов» для упрощения процесса создания приложения. Внимание, данные «виджеты» не то же самое, что известные всем виджеты для рабочего стола! Так вот, суть данных «виджетов» в том, что для отображения какой-либо информации можно использовать виджет, который запросит и выведет в установленном месте необходимые данные, для примера – дату рождения. Приведем краткий список возможных виджетов интерфейса:

Date Picker – может служить для воплощения примера с датой рождения, или же, естественно, любой другой даты.

Time Picker – аналогично предыдущему, только служит для выбора времени, будь-то время напоминания, будильника или чего бы то ни было.

Auto Complete – поле для ввода текстовой информации с возможность автоматического завершения недопечатанных слов, также достаточно удобен и распространен.

Spinner – отображение выпадающего списка с различными элементами.

Google Map View – то, что мы и собираемся использовать, виджет для работы с картами от Google.

Gallery – отображение горизонтального списка каких-либо элементов с возможностью их перелистывания, простейший пример – стандартная фотогалерея Android.

Web View – виджет для отображения web-страниц.

А также стандартные элементы-виджеты, вроде кнопок, полей для ввода текста, «галочек» и так далее.

Конечно же, мы привели далеко не полный список схем интерфейса и возможных виджетов, а лишь некоторые из них для более наглядного примера.

Итак, что нам будет нужно для совершения задуманного? Конечно, это упомянутый Map View, который послужит основой приложения, кроме того это схемы Relative Layout, Tab Layout, Table Layout, Spinner, List View и стандартные элементы оболочки. Так придется научиться делать собственные меню, но не будем забегать вперед.

Что ж, приступим. Создаем новый проект в IDE. Назовем его CityBus, хотя, конечно, название особой роли не играет. Важно, что для использования Map View нужно использовать Android и Google API. Тогда все нужные для работы библиотеки будут подключены к проекту автоматически в процессе сборки.

Создадим приложение, которое при запуске будет выводить на экран Google Maps, с возможностью масштабирования при помощи соответствующих кнопок. Для работы с Google[понадобится Maps API Key, так что предварительно его нужно получить](http://fans-android.com/poluchenie-klyucha-google-maps-api).

Для использования библиотек Google Maps объявляем их в *AndroidManifest.xml*. Нашему приложению для работы понадобится соединение с интернетом, поэтому не забываем присвоить ему соответствующие права. В манифесте добавляем между тегами <application> следующую строку:

*<uses-library android:name=»com.google.android.maps» />*

Обратите внимание, что добавлять данную строку нужно именно в корень тегов <application>, например в конце, перед закрывающим тегом </application>.

А вот после </application> добавляем следующее:

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

Для освобождения дополнительного пространства на экране для отображения Google Maps уберем строку состояния в нашем приложении. В том же самом .xml-файле ищем нашу Activity и добавляем вот такой параметр:

*<activity android:name=».CityBus»*

*android:label=»@string/app\_name»*

*android:theme=»@android:style/Theme.NoTitleBar»>*

Теперь приступим к редактированию файла *main.xml*. Корневым у нас будет Map View. Заменяем содержимое файла *main.xml*, поменям apiKey на свой ключ, следующим:

*<?xml version=»1.0″ encoding=»utf-8″?>*

*<com.google.android.maps.MapView*

*xmlns:android=»http://schemas.android.com/apk/res/android»*

*android:id=»@+id/mapview»*

*android:layout\_width=»fill\_parent»*

*android:layout\_height=»fill\_parent»*

*android:clickable=»true»*

*android:apiKey=»Сюда* *нужно* *вставить* *ранее* *сгенерированный Map API Key»*

*/>*

Открываем *CityBus.java*. Изменяем объявление класса приложения CityBus и указываем наследование MapActivity:

*public class CityBus extends MapActivity {*

После введения данной строки Eclipse любезно сообщит об ошибке в коде – однако, это не повод для беспокойства. Нажимаем сочетание клавиш Ctlr + Shift + O, автоматически добавляя строчку:

*import com.google.android.maps.MapActivity;*

Переопределяем метод isRouteDisplayed(), добавив его переопредление сразу после метода onCreate():

*@Override*

*protected boolean isRouteDisplayed() {*

*return false;*

*}*

Данное переопределение – обязательно, без него приложение работать не будет. Осталось добавить в OnCreate() отрисовку элементов для масштабирования карты Google. Например вот так:

*package com.training.citybus;*

*import android.os.Bundle;*

*import com.google.android.maps.MapActivity;*

*public class CityBus extends MapActivity {*

*/\*\* Called when the activity is first created. \*/*

*@Override*

*public void onCreate(Bundle savedInstanceState) {*

*super.onCreate(savedInstanceState);*

*setContentView(R.layout.main);*

*MapView mapView = (MapView) findViewById(R.id.mapview);*

*mapView.setBuiltInZoomControls(true);*

*}*

*@Override*

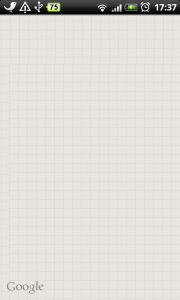
*protected boolean isRouteDisplayed() {*

*return false;*

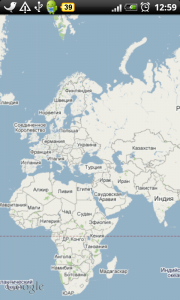
*}*

*}*

Теперь созданное приложение можно запускать для тестирования. В теории все должно было бы получится как и хотелось, но на деле получаем не совсем то, что планировали:

[](http://fans-android.com/wp-content/uploads/2012/10/maps.png)

То есть вроде бы всё работает, но сама карта не отображается. Разберемся, в чем проблема. А проблема в *AndroidManifest.xml*, где мы поставили теги внутри другого тега – а это ошибка. Мы приводим такие примеры для наглядности. Итак, исправив ошибки, и запустим приложение еще раз, наблюдаем на экране устройства всё, как и нужно:

[](http://fans-android.com/wp-content/uploads/2012/10/works.png)

Кнопки увеличения и уменьшения масштаба присутствуют, они просто не отобразились на снимках экрана. Но в целом мы получили запланированный результат. В следующей статье мы[разберемся, как реализовать в приложении меню, и, в том числе. кнопку «Выход»](http://fans-android.com/napisanie-prilozhenij-dlya-android-chast-2-sozdanie-menyu).

Мы продолжаем цикл статей о создании собственных приложений для ОС Android. На этот раз мы затронем тему создания в приложении меню и наполнения его дополнительными функциями.

В предыдущей статье мы уже [разобрались с основами создания пользовательского интерфейса](http://fans-android.com/napisanie-prilozhenij-dlya-android-chast-1-polzovatelskij-interfejs). На этот раз мы попытаемся добавить в наше приложение основное меню, которое можно открыть нажатием кнопки «Меню» на девайсе, а также дополнительное контекстное меню, которое можно вызвать длительным нажатием на рабочую область. Напомним, что за основу мы взяли Google Maps.

Сначала давайте ознакомимся, какие виды меню доступны для приложений в операционной системе Android. А доступно их три вида:

Options Menu – это основное меню программы, которое вызывается нажатием соответствующей кнопки на мобильном девайсе. Естественно, данное меню присутствует практически в любом приложении. Options Menu может состоять из пиктограмм двух категорий:

Icon Menu, или главное меню, которое способно отобразить до шести иконок. Если иконок больше шести, то последняя преобразовывается в пресловутое «Еще…». Данная кнопка же, в свою очередь, ведет в Expanded Menu.

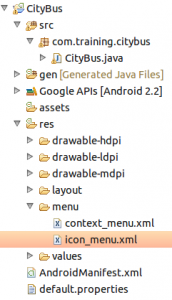
Expanded Menu – вертикальное меню, которое содержит шестую, а также все остальные иконки, которые не поместились в главном меню (Icon Menu).

Context Menu – это всплывающее меню, вызываемое длительным нажатием на рабочую область или какой-либо элемент интерфейса, по сути же – аналог клика правой кнопкой мышки на персональном компьютере.

Submenu – всплывающее меню, которое при необходимости вызывается нажатием на Options Menu или Context Menu. Submenu не может иметь собственных подменю.

Для достижения поставленных целей нам понадобятся такие виды меню, как Options Menu и Context Menu. Открываем наш проект. Нам потребуется иконка для создаваемого меню, так что создаем её или берем из любого источника. Меню опций у нас будет всего с одним пунктом, а именно – «Выход».

Во-первых, меню следует описать в отдельном файле xml. Описания всех меню приложения находятся в директории res/menu, так что если такой папки нет – создаем её. Внутри директории создаем файл icon\_menu.xml:

[](http://fans-android.com/wp-content/uploads/2012/10/res_menu_duos.png)

Заполняем данный файл. Корневым тегом является xml-тег menu, а сами элементы меню описываются дочерними тегами item. Item имеет три параметра:

Id – идентификатор элемента меню.

Icon – очевидно, это иконка элемента меню.

Title – название элемента меню.

Созданный xml-файл будет иметь следующий вид:

<?xml version=»1.0″ encoding=»utf-8″?>

<menu xmlns:android=»http://schemas.android.com/apk/res/android»>

    <item android:id=»@+id/quit»

          android:icon=»@drawable/ic\_menu\_exit»

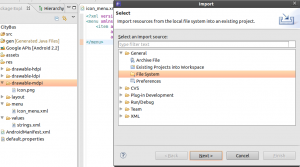
android:title=»@string/quit» />

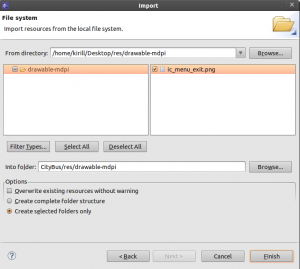
</menu>

В файл res/values/strings.xmlдобавляем строчку «quit»:

<string name=»quit»>Выйти</string>

Не забываем добавить иконки, созданные или взятые из стороннего источника. Импортируем их во все три папки drawable. Для вызова диалогового окна импорта, кликаем правой кнопкой на папке и выбираем пункт «Import».

[](http://fans-android.com/wp-content/uploads/2012/10/import.png)

[](http://fans-android.com/wp-content/uploads/2012/10/import2.png)

Меню создано и необходимые ресурсы с ним связаны. Осталось задействовать его непосредственно в приложении. Это не слишком сложно. В java-проекте добавляем вот такие строки:

@Override

    public boolean onCreateOptionsMenu(Menu menu) {

        MenuInflater inflater = getMenuInflater();

        inflater.inflate(R.menu.icon\_menu, menu);

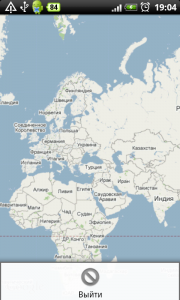
return true;

    }

В Eclipse всё несколько проще – нажав сочетание клавиш Ctrl + Shift + O нужные классы будут автоматически добавлены в создаваемый проект.

Метод onCreateOptionsMenu вызывается путем нажатия кнопки «Меню» пользователем. Переопределяем его и добавляем созданный в .xml-файле контент.

Если всё было сделано как следует, при запуске приложения на экране будет отображена вот такая вот картина:

[](http://fans-android.com/wp-content/uploads/2012/10/exit_menu.png)

Осталось добавить созданной кнопке «Выход» соответствующий функционал – то есть завершение приложения по нажатию. В CityBus.java добавляем такие строки:

@Override

    public boolean onOptionsItemSelected(MenuItem item) {

        // Handle item selection

        switch (item.getItemId()) {

        case R.id.quit:

            quit();

            return true;

        default:

            return super.onOptionsItemSelected(item);

}

    }

В данном отрывке кода мы проводим переопределение onOptionsItemSelected и перехватываем нажатие на элементы меню, после чего проводится проверка, какой именно элемент меню был нажат. Так как вызывается метод quit() класса CityBus, добавляем его в код следующими строками:

private void quit() {

            finish();

}

Здесь ничего сложного уже нет — finish() завершает выполнение приложения, то есть выходит из него. Именно то, что нам нужно. Проверяем код в работе – если всё сделано правильно, то ошибок возникнуть не должно.

На этом всё. Мы создали простенькое приложение на основе Google Maps с возможностью масштабирования карты и выхода.