МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное автономное образовательное учреждение
высшего профессионального образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

ПРОГРАММИРОВАНИЕ МОБИЛЬНЫХ УСТРОЙСТВ

Методические указания к выполнению лабораторных работ

Санкт-Петербург

2020

Лабораторная работа №1

Развертывание средств разработки приложений для мобильных устройств

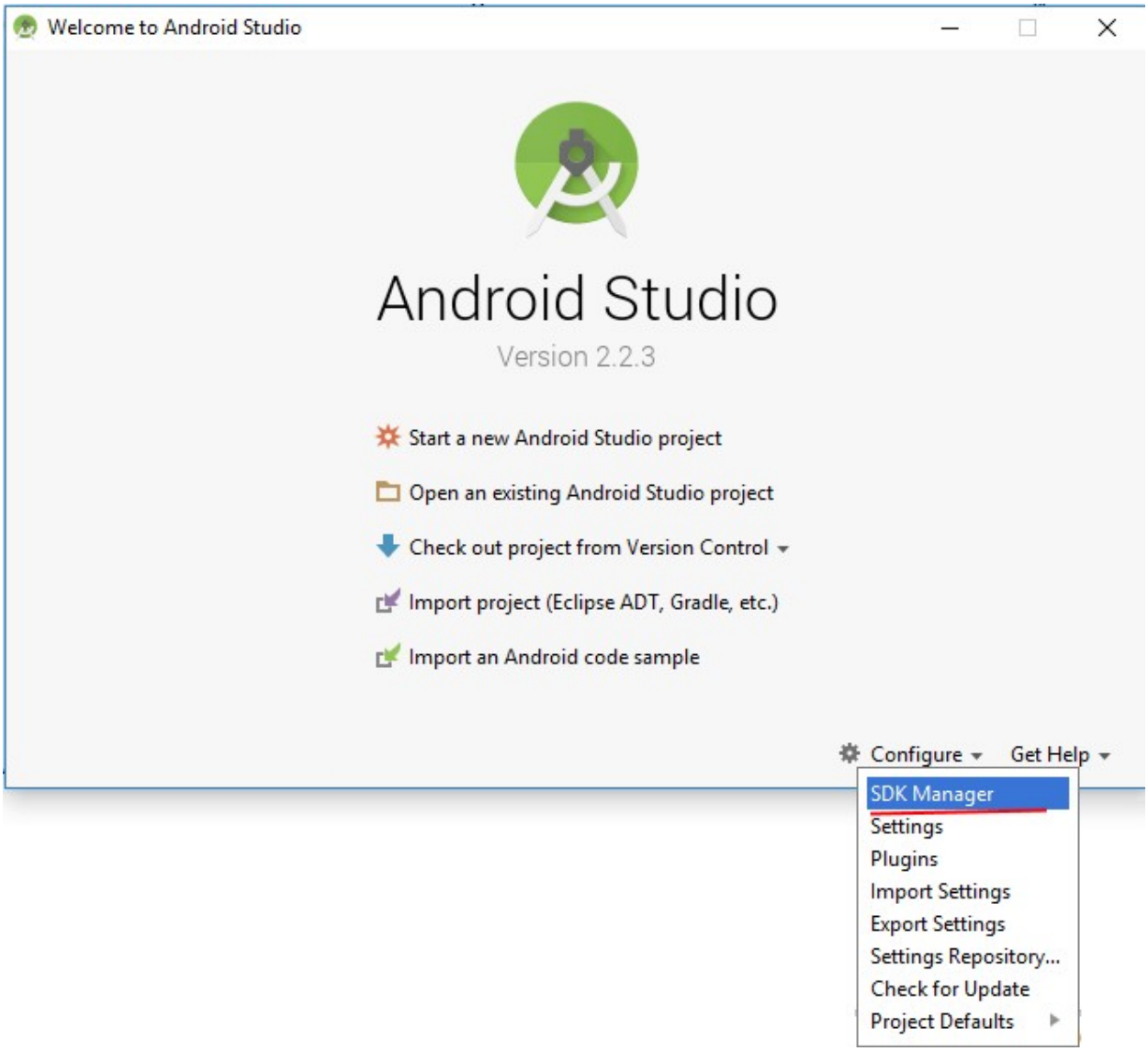
Задание: Выбрать тип и версию планируемой к использованию интегрированной среды разработки, соответствующую типам и версиям мобильной ОС и ОС на персональном компьютере (например, Android Studio или Qt-Creator для мобильной ОС Android), установить среду разработки, выбрать и установить версию SDK (NDK) (при необходимости - несколько версий), образ системы для эмулятора, проверить возможность работы средств разработки с реальным устройством и (или) эмулятором устройства. Обосновать выбор версии средств разработки и SDK (NDK) и технологии отладки (эмулятор или (и) реальное устройство).

Цель работы: Освоение процесса развертывания средств разработки мобильных приложений, ознакомление со средствами разработки и отладки.

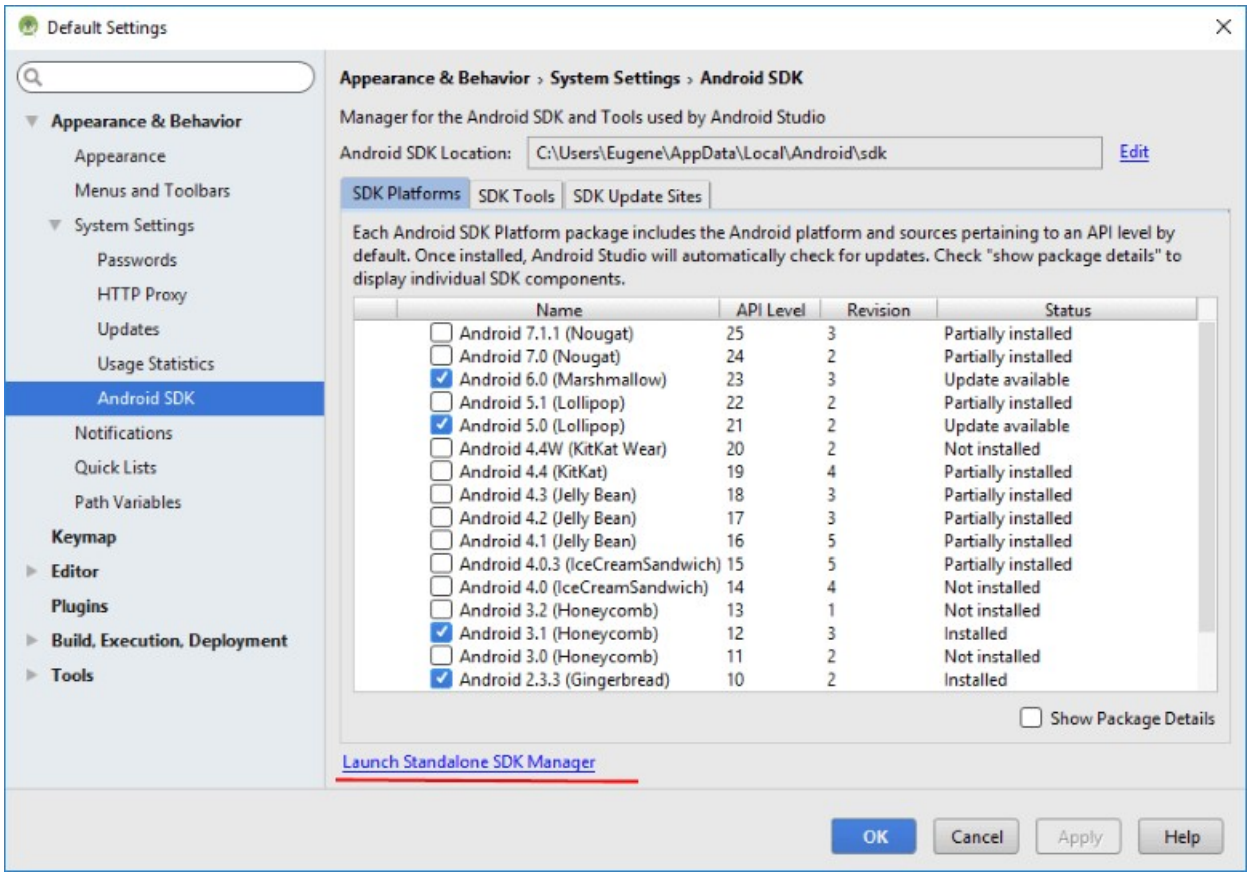
Настройка Android Studio и Android SDK

Все программы, которые создаются под Android с помощью Java, зависит от Android SDK – если создается приложение под определенную версию, например, для Android Nougat, то должны быть установлены соответствующие инструменты SDK. Это надо учитывать при разработке.

По умолчанию, если Android Studio запускается в первый раз, то открывается некоторое начальное меню. В самом низу стартового экрана программы найдем кнопку "Configure" и нажмем на нее:

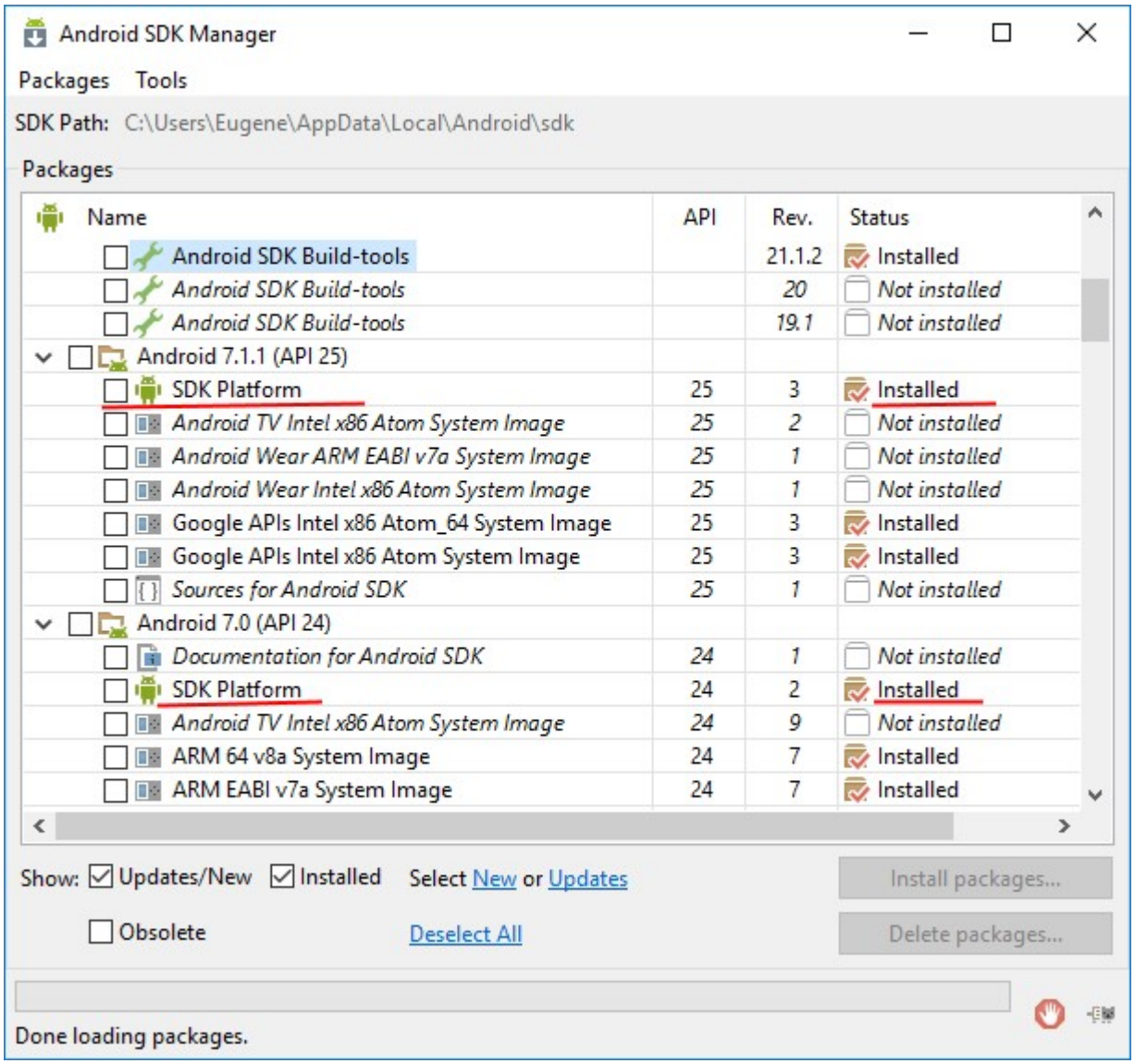


Далее в выпадающем меню нажмем на пункт "SDK Manager". После этого откроется окно с настройками для Android SDK Manager:



Здесь можно увидеть, какие версии SDK установлены. Но каждая версия предполагает широкий круг компонентов, это в том числе и инструменты для разработки под телевизоры, под смарт-часы и т.д. Не все эти инструменты могут понадобиться, поэтому нет смысла полностью устанавливать все версии SDK.

Для более детального просмотра всех компонентов по каждой платформе внизу окна есть ссылка Launch Standalone SDK Manager. Эта ссылка опять же запустит Android SDK Manager, но в отдельном окне:



Структурно SDK Manager построен следующим образом. Сначала идет секция **Tools**. Она содержит пакеты, необходимые для разработки приложения.

Далее идут секции плафторм Android. Здесь детально можно посмотреть, какие пакеты для каждой платформы установлены. В данном случае нас прежде всего будет интересовать пункт **SDK Platform**. SDK Platform содержит весь основной функционал, который используется при разработке. Данный пункт можно выделить для всех тех платформ, под которые собираются собираться приложения.

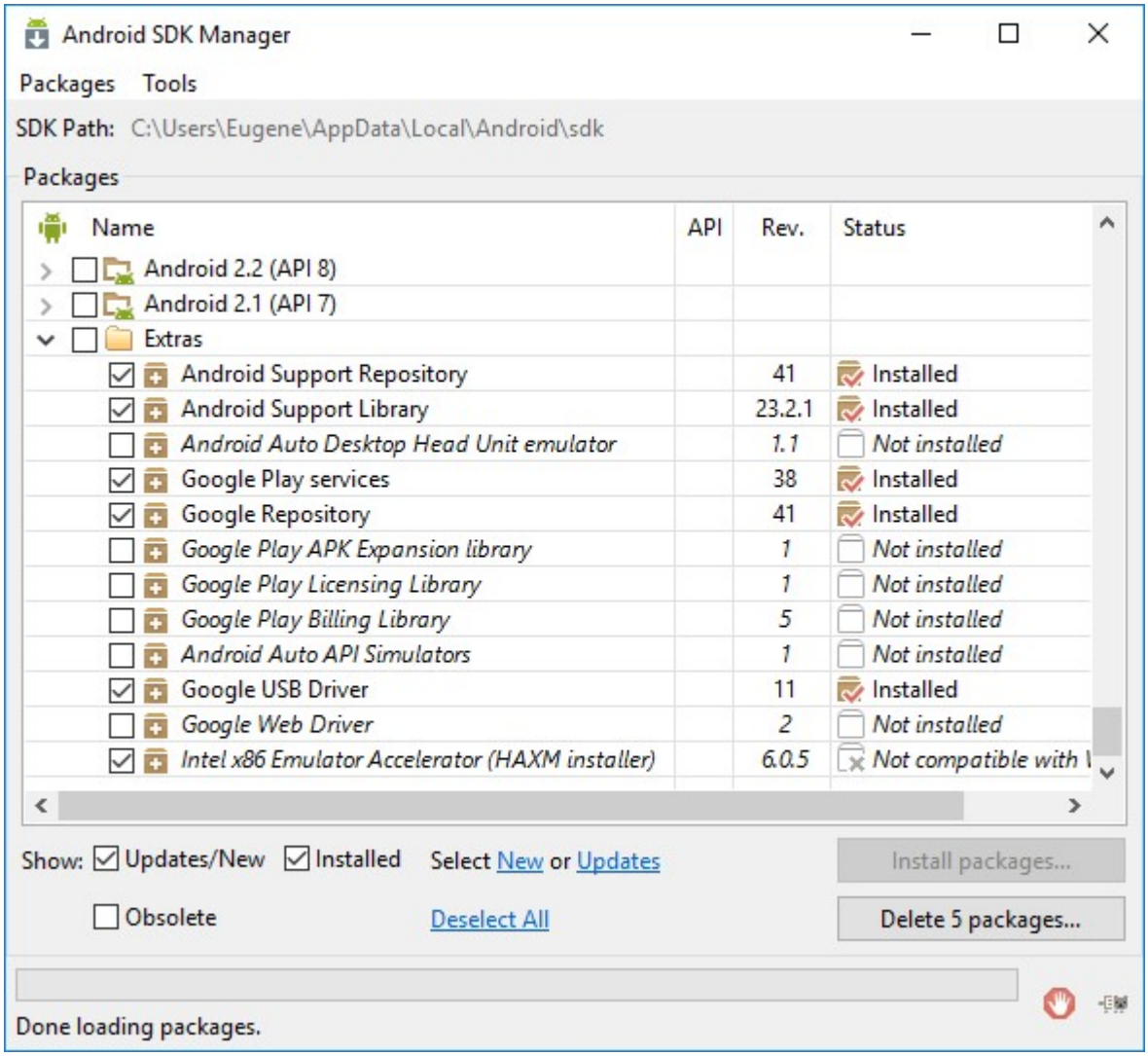
Кроме SDK Platform каждая платформы, как правило, содержит еще ряд компонентов:

- **Google APIs**: различные API от Google, в частности, интерфейсы по работе с Google Maps, другими сервисами Google;
- **ARM EABI v7a System Image**: образ системы, эмулирующий работу процессора ARM, необходимый для запуска эмулятора;
- **Intel x86 Atom System Image / Intel x86 Atom_64 System Image**: образ системы (для 86-х и 64-х платформ) от компании Intel, также необходимый для запуска эмулятора;
- **Google APIs ARM EABI v7a System Image**: Google API для образа системы ARM EABI v7a System Image;

- **Google APIs Intel x86 Atom System Image / Intel x86 Atom_64 System Image**: Google API для образа системы Intel x86 Atom System Image или Intel x86 Atom_64 System Image.

Если не планируется использование эмулятора, то данные компоненты не столь важны. И наоборот, если тестирование будет происходить на эмуляторе, то следует установить для этого образ системы, который соответствует текущей рабочей машине, и соответствующее для него Google API.

После всех платформ в SDK Manager идет секция Extras. Здесь перечислены дополнительные пакеты, не привязанные к платформам:



Если вы хотите использовать для тестирования приложений смартфон от Google – Nexus 5/5X и 6/6P или Google Pixel, то обязательно надо установить пакет **Google Usb Driver**, который представляет драйвер для работы с данными смартфонами. Если же предполагается использовать смартфон другого производителя, то в этом случае надо будет установить USB-драйвер непосредственно от этого производителя. Как правило, при подключении смартфона система сама пытается установить драйвер.

Если планируется использование эмулятора, то также следует установить пакет **Intel x86 Emulator Accelerator (HAXM installer).**

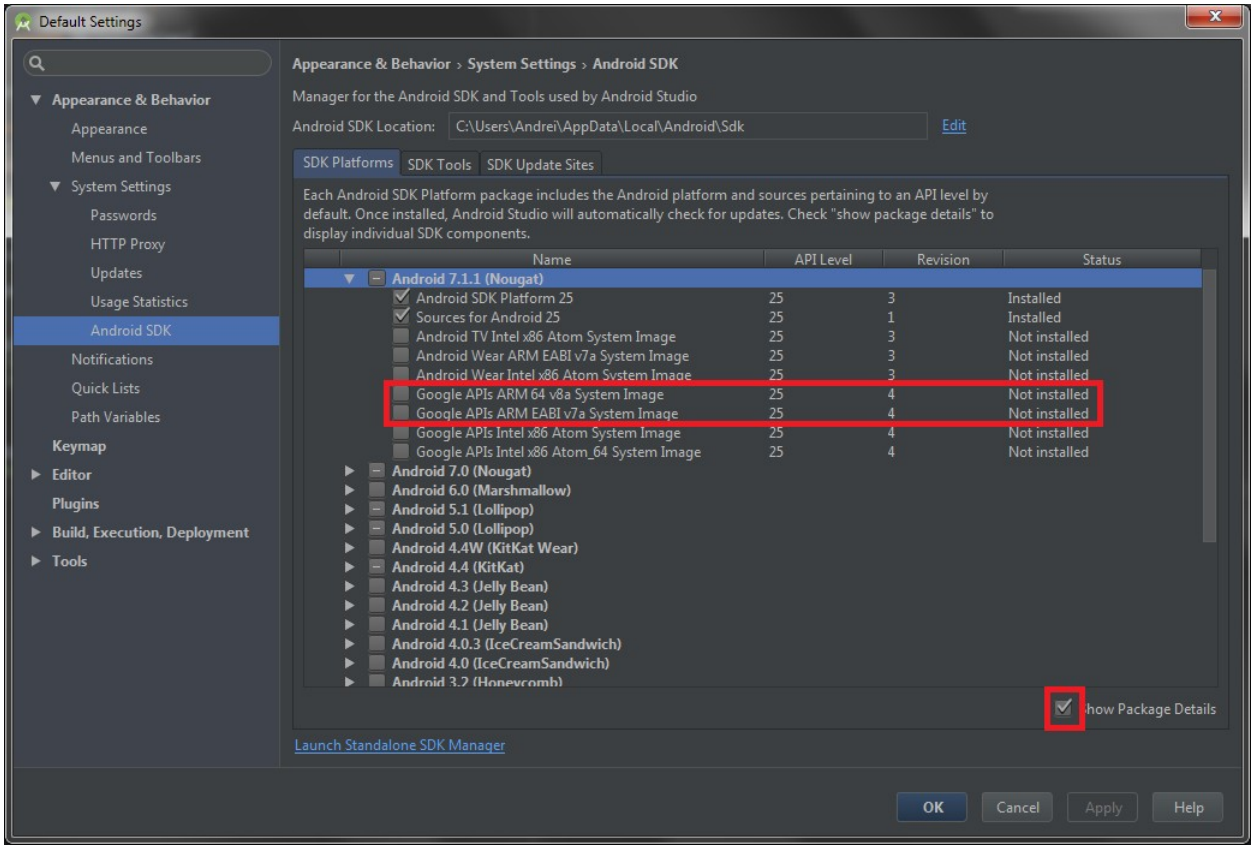
Если в процессе разработки или тестирования будут возникать проблемы с какими-то версиями ОС Android, то вполне вероятно корень проблем заключается в отсутствии нужных компонентов для определенных платформ. В этом случае можно будет из Android Studio запустить SDK Manager и вручную установить недостающие компоненты.

Впоследствии при каждом запуске Android Studio будут появляться всплывающие уведомления о доступности обновлений по платформам, что довольно удобно и что позволит позволит проектам не отставать от последних обновлений от Google.

Если впоследствии потребуется вызвать SDK Manager, то мы можем сделать это из самой Android Sudio через панель инструментов.

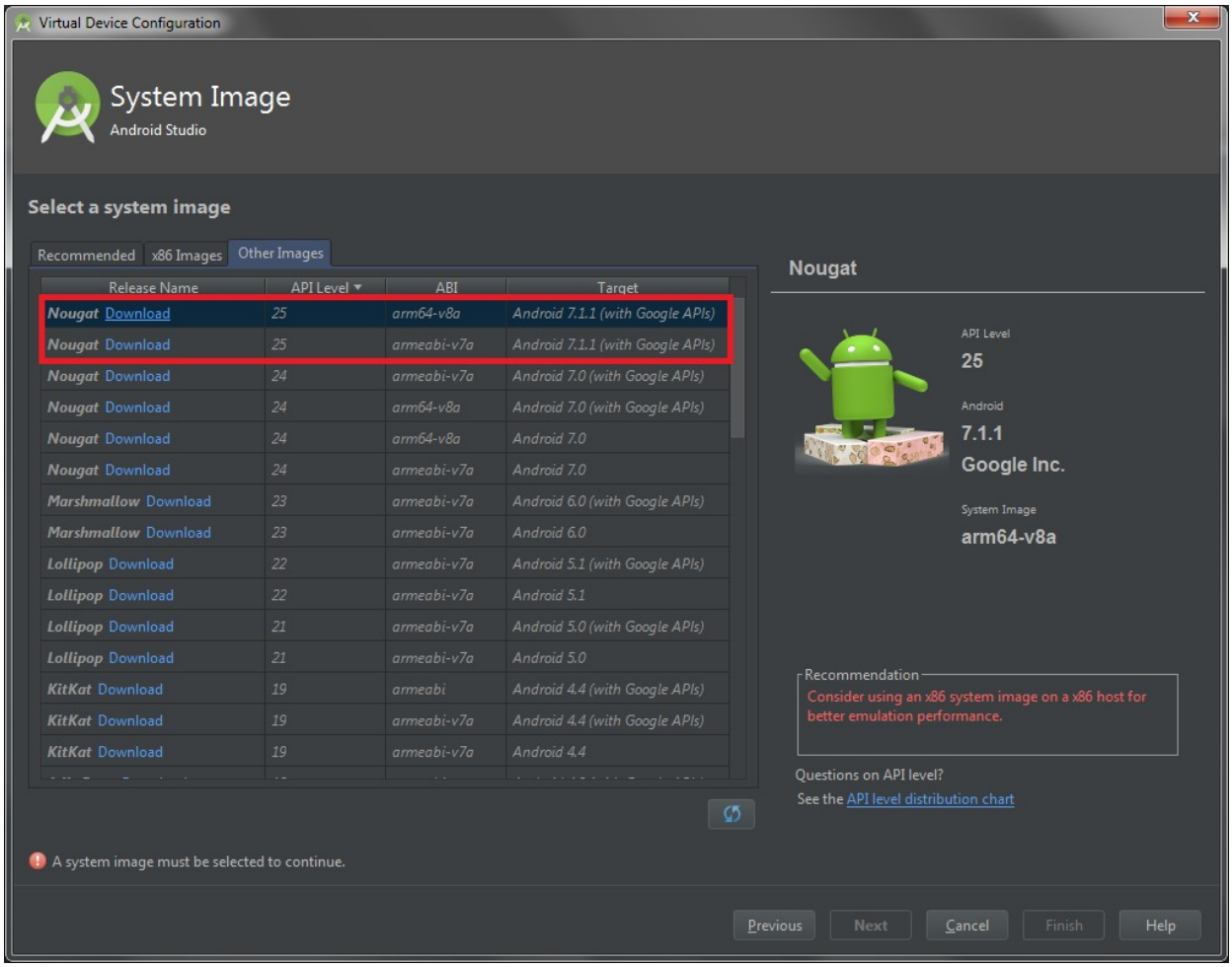
Настройка Andoid Studio для эмуляции на процессорах AMD

1. Для начала нужно открыть окно «Android SDK», которое находится в меню: Tools -> Android -> SDK Manager.
2. Для того, чтобы увидеть полный список пакетов доступных для скачивания и установки, поставте галочку в нижнем правом углу «Show Package Details». Необходимыми пакетами являются «ARM EABI v7a System Image» или «ARM 64 v8a System Image». Скачивать пакеты необходимо для той версии Android, для которой будет вестись разработка.



3. Вторым шагом является настройка и создание виртуального устройства. Менеджер виртуальных устройств находится в меню: Tools -> Android -> AVD Manager.
4. Создав новое виртуальное устройство («Create virtual device») и выбрав нужное из списка, нажмите кнопку «Next». В появившемся окне на вкладке «Other Images» необходимо скачать образы, соответствующие версиям API и Android, которые были выбраны в предыдущем пункте.
5. Для завершения настройки необходимо перезапустить Android Studio.

Последующую работу необходимо будет вести с созданным устройством.



Лабораторная работа №2

Освоение средств разработки, создание приложения для мобильного устройства

Задание: Создать простое приложение, выполнить сборку приложения в среде разработки и запуск на мобильном устройстве или эмуляторе. Описать результаты в отчете, включая исходные тексты приложения.

Цель работы: Освоение процесса сборки и отладки приложения для мобильного устройства.

Лабораторная работа №3

Создание приложения с различными интерфейсными элементами

Задание: Создать простое приложение с использованием различных интерфейсных элементов, выполнить сборку и запуск приложения. Убедиться в корректном отображении интерфейсных элементов в основном окне. Описать результаты в отчете (в т.ч. тексты xml для activity).

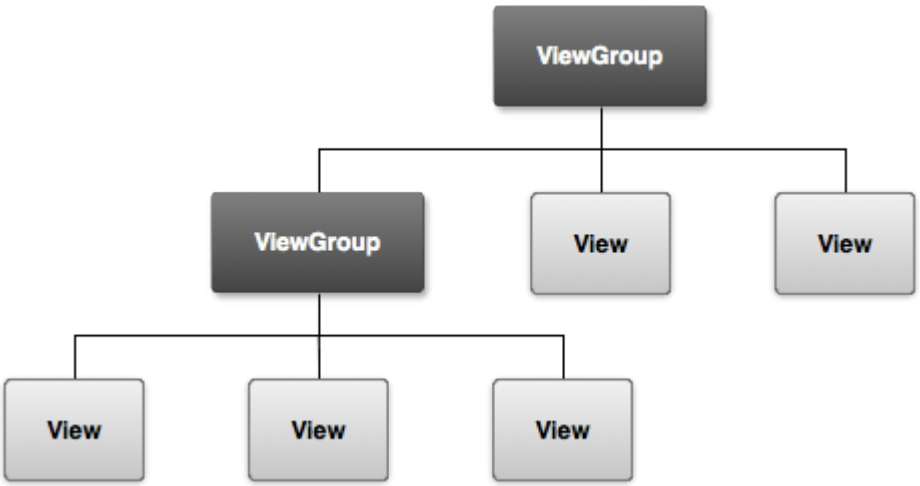
Цель работы: Освоение возможностей создания приложения с различными типами

интерфейсных элементов для мобильного устройства.

Графический интерфейс пользователя

Графический интерфейс пользователя представляет собой иерархию объектов android.view.View и android.view.ViewGroup. Каждый объект ViewGroup представляет контейнер, который содержит и упорядочивает дочерние объекты View. В частности, к контейнерам относят такие элементы как RelativeLayout, LinearLayout, GridLayout и ряд других.

Простые объекты View представляют собой элементы управления и прочие виджеты, например, кнопки, текстовые поля и т.д., через которые пользователь взаимодействует с программой:



Ключевым компонентом для создания визуального интерфейса в приложении Android является activity (активность). Нередко activity ассоциируется с отдельным экраном или окном приложения, а переключение между окнами будет происходить как перемещение от одной activity к другой. Приложение может иметь одну или несколько activity.

Все объекты activity представляют собой объекты класса android.app.Activity, которая содержит базовую функциональность для всех activity.

Жизненный цикл приложения

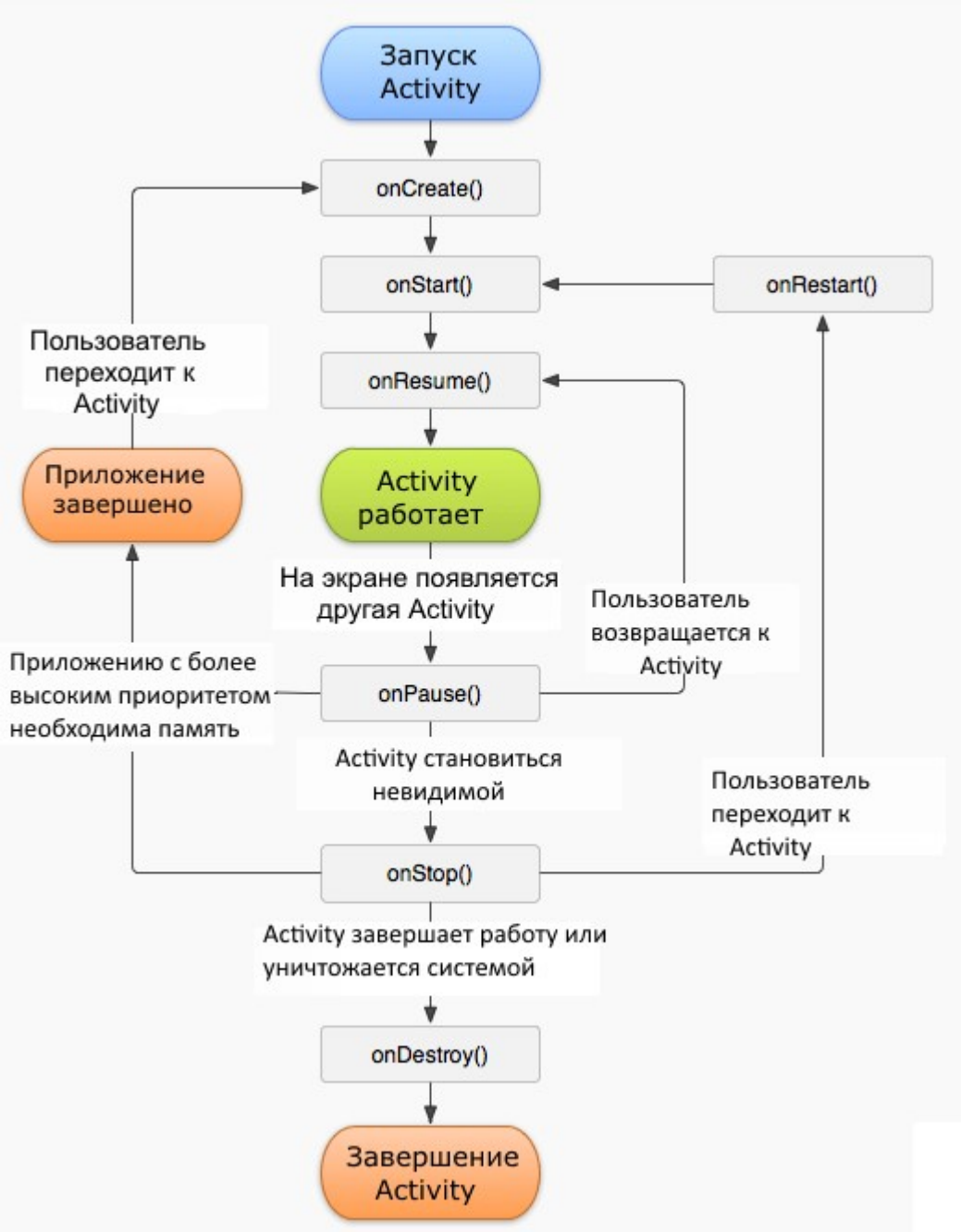
Все приложения Android имеют строго определенный системой жизненный цикл. При запуске пользователем приложения система дает этому приложению высокий приоритет. Каждое приложение запускается в виде отдельного процесса, что позволяет системе давать одним процессам более высокой приоритет, в отличие от других. Благодаря этому, например, при работе с одними приложениями не блокировать входящие звонки. После прекращения работы с приложением, система освобождает все связанные ресурсы и переводит приложение в разряд низкоприоритетного и закрывает его.

Все объекты activity, которые есть в приложении, управляются системой в виде стека activity, который называется back stack. При запуске новой activity она помещается поверх стека и выводится на экран устройства, пока не появится новая activity. Когда текущая activity заканчивает свою работу (например, пользователь уходит из приложения), то она удаляется из стека, и возобновляет работу та activity, которая ранее была второй в стеке.

После запуска activity проходит через ряд событий, которые обрабатываются системой и для обработки которых существует ряд обратных вызовов:

```
protected void onCreate(Bundle saveInstanceState);
protected void onStart();
protected void onRestoreInstanceState(Bundle saveInstanceState);
protected void onRestart();
protected void onResume();
protected void onPause();
protected void onSaveInstanceState(Bundle saveInstanceState);
protected void onStop();
protected void onDestroy();
```

Схематично взаимосвязь между всеми этими обратными вызовами можно представить следующим образом:



onCreate – первый метод, с которого начинается выполнение activity. В этом методе activity переходит в состояние Created. Этот метод обязательно должен быть определен в классе activity. В нем производится первоначальная настройка activity. В частности, создаются объекты визуального интерфейса. Этот метод получает объект Bundle, который содержит прежнее состояние activity, если оно было сохранено. Если activity заново создается, то данный объект имеет значение null. Если же activity уже ранее была создана, но находилась в приостановленном состоянии, то bundle содержит связанную с activity информацию.

В методе **onStart** осуществляется подготовка к выводу activity на экран устройства. Как правило, этот метод не требует переопределения, а всю работу производит встроенный код. После завершения работы метода activity отображается на экране, вызывается метод onResume, а activity переходит в состояние Resumed.

После завершения метода **onStart** вызывается метод **onRestoreInstanceState**, который призван восстанавливать сохраненное состояние из объекта Bundle, который передается в качестве параметра. Но следует учитывать, что этот метод вызывается только тогда, когда Bundle не равен null и содержит ранее сохраненное состояние. Так, при первом запуске приложения этот объект Bundle будет иметь значение null, поэтому и метод onRestoreInstanceState не будет вызываться.

А при вызове метода **onResume** activity переходит в состояние Resumed, а пользователь может с ней взаимодействовать. И собственно activity остается в этом состоянии, пока она не потеряет фокус, например, в следствии переключения на другую activity или просто из-за выключения экрана устройства.

Если пользователь решит перейти к другой activity, то система вызывает метод **onPause**. В этом методе можно освобождать используемые ресурсы, приостанавливать процессы, например, воспроизведение аудио, анимаций, останавливать работу камеры (если она используется) и т.д., чтобы они меньше сказывались на производительность системы.

Но надо учитывать, что на работу данного метода отводится очень мало времени, поэтому не стоит здесь сохранять какие-то данные, особенно если при этом требуется обращение к сети, например, отправка данных по интернету, или обращение к базе данных.

После выполнения этого метода activity становится невидимой, не отображается на экране, но она все еще активна. И если пользователь решит вернуться к этой activity, то система вызовет снова метод onResume, и activity снова появится на экране.

Другой вариант работы может возникнуть, если вдруг система видит, что для работы активных приложений необходимо больше памяти. И система может сама завершить полностью работу activity, которая невидима и находится в фоне. Либо пользователь может нажать на кнопку Back (Назад). В этом случае у activity вызывается метод onStop.

Метод **onSaveInstanceState** вызывается после метода onPause, но до вызова onStop. В onSaveInstanceState производится сохранение состояния приложения в передаваемый в качестве параметра объект Bundle.

В методе **onStop**, activity переходит в состояние Stopped. В методе onStop следует особождать используемые ресурсы, которые не нужны пользователю, когда он не взаимодействует с activity. Здесь также можно сохранять данные, например, в базу данных.

При этом во время состояния Stopped activity остается в памяти устройства, сохраняется состояние всех элементов интерфейса. К примеру, если в текстовое поле EditText был введен какой-то текст, то после возобновления работы activity и перехода ее в состояние Resumed мы вновь увидим в текстовом поле ранее введенный текст.

Если после вызова метода onStop пользователь решит вернуться к прежней activity, тогда система вызовет метод onRestart. Если же activity вовсе завершила свою работу, например, из-за закрытия приложения, то вызывается метод onDestroy.

Завершается работа активности вызовом метода **onDestroy**, который возникает либо, если система решит убить activity, либо при вызове метода finish.

Также следует отметить, что при изменении ориентации экрана система завершает activity и затем создает ее заново, вызывая метод onCreate.

```
b.putInt("key", 1); // Нужный id
intent.putExtras(b); // Добавляем id в intent
startActivity(intent);
finish();
```

Затем получаем идентификатор в новом Activity:

```
Bundle b = getIntent().getExtras();
int value = 0;
if(b != null)
    value = b.getInt("key");
```

Пример №2:

```
Intent intent = new Intent(getBaseContext(), SignoutActivity.class);
intent.putExtra("EXTRA_SESSION_ID", sessionId);
startActivity(intent);
```

Доступ к этому intent в следующей Activity:

```
String s = getIntent().getStringExtra("EXTRA_SESSION_ID");
```

Подробнее о том, как работают объекты Intent и фильтры объектов Intent, можно в официальной документации.

Лабораторная работа №4

Создание нескольких экранов в приложении

Задание: Создать простое приложение с использованием различных интерфейсных элементов, размещенных на нескольких экранах (Activity) выполнить сборку и запуск приложения. Убедиться в корректном отображении интерфейсных элементов в основном окне. Описать результаты в отчете (в т.ч. тексты xml для activity).

Цель работы: Освоение возможностей создания приложения с различными типами интерфейсных элементов для мобильного устройства.

Лабораторная работа №5

Организация навигации в многоэкранном приложении

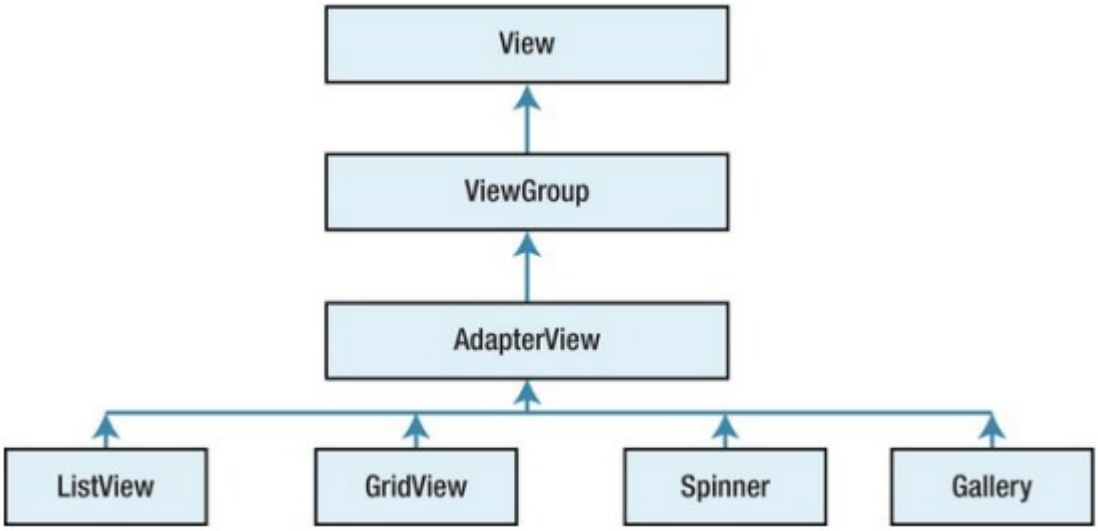
Задание: Организовать навигацию между несколькими Activity (не менее 3-х) при помощи обработки нажатий на интерфейсные элементы, выполнить сборку, отладку и запуск приложения. Навигация должна позволять возможность перехода из любой экранной формы (Activity) в любую другую, возможны различные способы организации (переход из основной и обратно, прямой переход из каждой в каждую, циклический переход по цепочке) Описать результаты в отчете (в т.ч. исх. тексты java и xml для activity).

Цель работы: Получение навыков организации навигации в многоэкранных мобильных приложениях

Адаптеры и списки

Android представляет широкую палитру элементов, которые представляют списки. Все они является наследниками класса android.widget.AdapterView. Это такие виджеты как

ListView, GridView, Spinner. Они могут выступать контейнерами для других элементов управления.



При работе со списками работа ведется с тремя компонентами. Первый – это сами элементы списков (ListView, GridView), которые отображают данные. Второй – это источник данных – массив, объект ArrayList, база данных и т.д., в котором находятся сами отображаемые данные. Третий – это адаптеры – специальные компоненты, которые связывают источник данных с элементом списка.

Рассмотрим связь элемента ListView с источником данных с помощью одного из таких адаптеров – класса ArrayAdapter.

Класс ArrayAdapter представляет собой простейший адаптер, который связывает массив данных с набором элементов TextView, из которых, к примеру, может состоять ListView. То есть в данном случае источником данных выступает массив объектов. ArrayAdapter вызывает у каждого объекта метод toString() для приведения к строковому виду и полученную строку устанавливает в элемент TextView.

Пример использования:

Файл разметки:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView android:id="@+id/countriesList"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </ListView>

</RelativeLayout>
```

Исходный код Activity:

```
public class MainActivity extends AppCompatActivity {
    // набор данных, которые свяжем со списком
    String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили", "Уругвай"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // получаем элемент ListView
        ListView countriesList = (ListView) findViewById(R.id.countriesList);

        // создаем адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_1, countries);

        // устанавливаем для списка адаптер
        countriesList.setAdapter(adapter);
    }
}
```

Для создания адаптера использовался следующий конструктор `ArrayAdapter(this, android.R.layout.simple_list_item_1, countries)`, где:

- this: текущий объект Activity;
- android.R.layout.simple_list_item_1: файл разметки списка, который фреймворк представляет по умолчанию. Он находится в папке Android SDK по пути *platforms/[android-номер_версии]/data/res/layout*;
- countries: массив данных. Здесь необязательно указывать именно массив, это может быть список ArrayList<T>.

Создание сложного списка с кнопками:

Определим следующий класс Product (данный класс хранит название, количество продукта, а также единицу измерения, объекты этого класса будут выводиться в список):

```
public class Product {
    private String name;
    private int count;
    private String unit;

    Product(String name, String unit) {
        this.name = name;
        this.count = 0;
        this.unit = unit;
    }

    public String getUnit() {
        return this.unit;
    }

    public void setCount(int count) {
        this.count = count;
    }

    public int getCount() {
        return count;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }
}
```

В папку res/layout добавим новый файл list_item.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="16dp">

    <TextView android:id="@+id/nameView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:textSize="18sp" />
    <TextView android:id="@+id/countView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:textSize="18sp" />
    <Button android:id="@+id/addButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="+"/>
    <Button
        android:id="@+id/removeButton"
        android:layout_width="0dp"
```

```
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="-"/>
</LinearLayout>
```

Здесь определены два текстовых поля для вывода названия и количества продукта и две кнопки для добавления и удаления одной единицы продукта. Теперь необходимо добавить класс адаптера ProductAdapter:

```
class ProductAdapter extends ArrayAdapter<Product> {
    private LayoutInflater inflater;
    private int layout;
    private ArrayList<Product> productList;

    ProductAdapter(Context context, int resource, ArrayList<Product> products) {
        super(context, resource, products);
        this.productList = products;
        this.layout = resource;
        this.inflater = LayoutInflater.from(context);
    }

    public View getView(int position, View convertView, ViewGroup parent) {

        final ViewHolder viewHolder;
        if (convertView == null) {
            convertView = inflater.inflate(this.layout, parent, false);
            viewHolder = new ViewHolder(convertView);
            convertView.setTag(viewHolder);
        } else {
            viewHolder = (ViewHolder) convertView.getTag();
        }
        final Product product = productList.get(position);

        viewHolder.nameView.setText(product.getName());
        viewHolder.countView.setText(formatValue(product.getCount(), product.getUnit()));

        viewHolder.removeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int count = product.getCount() - 1;
                if (count < 0) count = 0;
                product.setCount(count);
                viewHolder.countView.setText(formatValue(count, product.getUnit()));
            }
        });
        viewHolder.addButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int count = product.getCount() + 1; product.setCount(count);
                viewHolder.countView.setText(formatValue(count, product.getUnit()));
            }
        });

        return convertView;
    }

    private String formatValue(int count, String unit) {
        return String.valueOf(count) + " " + unit;
    }

    private class ViewHolder {
        final Button addButton, removeButton;
        final TextView nameView, countView;
```

```
        ViewHolder(View view) {
            addButton = (Button) view.findViewById(R.id.addButton);
            removeButton = (Button) view.findViewById(R.id.removeButton);
            nameView = (TextView) view.findViewById(R.id.nameView);
            countView = (TextView) view.findViewById(R.id.countView);
        }
    }
}
```

Для каждой кнопки определен обработчик нажатия, в котором уменьшается, либо увеличивается количество продукта на единицу и затем изменяется текст в соответствующем текстовом поле. Далее в файле activity_main.xml необходимо определить элемент ListView:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ListView
        android:id="@+id/productList"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>
```

И изменить логику MainActivity:

```
public class MainActivity extends AppCompatActivity {

    ArrayList<Product> products = new ArrayList();

    ListView productList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        if (products.size() == 0) {
            products.add(new Product("Картофель", "кг."));
            products.add(new Product("Чай", "шт."));
            products.add(new Product("Яйца", "шт."));
            products.add(new Product("Молоко", "л."));
            products.add(new Product("Макароны", "кг."));
        }
        productList = (ListView) findViewById(R.id.productList);
        ProductAdapter adapter = new ProductAdapter(this, R.layout.list_item, products);
        productList.setAdapter(adapter);
    }
```

Лабораторная работа №6

Организация доступа к аппаратным компонентам мобильного устройства

Задание: Дополнить приложение, разработанное в рамках одной из предыдущих лабораторных работ, добавив в него работу с аппаратным обеспечение мобильного устройства (камера, светодиод подсветки, датчики и т.п.)

Цель работы: Получение навыков использования в разрабатываемых мобильных приложениия доступа к компонентам аппаратного обеспечения мобильных устройств.

Пример работы с камерой

Пример в официальной документации: https://developer.android.com/training/camera/photobasics.html

Основные объекты, которые используются при работе с камерой: Camera, SurfaceView, SurfaceHolder.

Camera используется, чтобы получить изображение с камеры. А чтобы это изображение в приложении отобразить, используется SurfaceView.

Работа с surface ведется не напрямую, а через посредника – SurfaceHolder. Именно с этим объектом умеет работать Camera. Также, SurfaceHolder будет сообщать нам о том, что surface готов к работе, изменен или более недоступен.

В манифест добавьте права на камеру, запись звука и запись на sd-карту!

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

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

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

strings.xml:

```
<resources>
    <string name="take_picture">Picture</string>
    <string name="start_record">Start</string>
    <string name="stop_record">Stop</string>
</resources>
```

main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">
    <SurfaceView
        android:id="@+id/surfaceView"
        android:layout_width="200dp"
        android:layout_height="150dp"
```

```
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true">
    </SurfaceView>
    <Button android:id="@+id/btnTakePicture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@id/surfaceView"
        android:layout_marginRight="10dp"
        android:layout_marginTop="10dp"
        android:onClick="onClickPicture"
        android:text="@string/take_picture">
    </Button>
    <Button android:id="@+id/btnStartRecord"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/btnTakePicture"
        android:layout_toRightOf="@id/btnTakePicture"
        android:onClick="onClickStartRecord"
        android:text="@string/start_record">
    </Button>
    <Button android:id="@+id/btnStopRecord"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/btnStartRecord"
        android:layout_toRightOf="@id/btnStartRecord"
        android:onClick="onClickStopRecord"
        android:text="@string/stop_record">
    </Button>
</RelativeLayout>
```

MainActivity.java:

```
public class MainActivity extends Activity {

    SurfaceView surfaceView;
    Camera camera;
    MediaRecorder mediaRecorder;

    File photoFile;
    File videoFile;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        File pictures = Environment
                .getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
        photoFile = new File(pictures, "myphoto.jpg");
        videoFile = new File(pictures, "myvideo.3gp");

        surfaceView = (SurfaceView) findViewById(R.id.surfaceView);

        SurfaceHolder holder = surfaceView.getHolder();
        holder.addCallback(new SurfaceHolder.Callback() {
            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                try {
                    camera.setPreviewDisplay(holder);
                    camera.startPreview();
                } catch (Exception e) {
```

```
                e.printStackTrace();
            }
        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format,
                                   int width, int height) {
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
        }
    });

}

@Override
protected void onResume() {
    super.onResume();
    camera = Camera.open();
}

@Override
protected void onPause() {
    super.onPause();
    releaseMediaRecorder();
    if (camera != null)
        camera.release();
    camera = null;
}

public void onClickPicture(View view) {
    camera.takePicture(null, null, new PictureCallback() {
        @Override
        public void onPictureTaken(byte[] data, Camera camera) {
            try {
                FileOutputStream fos = new FileOutputStream(photoFile);
                fos.write(data);
                fos.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });

}

public void onClickStartRecord(View view) {
    if (prepareVideoRecorder()) {
        mediaRecorder.start();
    } else {
        releaseMediaRecorder();
    }
}

public void onClickStopRecord(View view) {
    if (mediaRecorder != null) {
        mediaRecorder.stop();
        releaseMediaRecorder();
    }
}

private boolean prepareVideoRecorder() {
```

```
        camera.unlock();

        mediaRecorder = new MediaRecorder();

        mediaRecorder.setCamera(camera);
        mediaRecorder.setAudioSource(MediaRecorder.AudioSource.CAMCORDER);
        mediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
        mediaRecorder.setProfile(CamcorderProfile
                .get(CamcorderProfile.QUALITY_HIGH));
        mediaRecorder.setOutputFile(videoFile.getAbsolutePath());
        mediaRecorder.setPreviewDisplay(surfaceView.getHolder().getSurface());

        try {
            mediaRecorder.prepare();
        } catch (Exception e) {
            e.printStackTrace();
            releaseMediaRecorder();
            return false;
        }
        return true;
    }

    private void releaseMediaRecorder() {
        if (mediaRecorder != null) {
            mediaRecorder.reset();
            mediaRecorder.release();
            mediaRecorder = null;
            camera.lock();
        }
    }
}
```

В onCreate настраивается Activity так, чтобы оно было без заголовка и в полный экран. Затем мы определяется surface и его SurfaceHolder. Далее для SurfaceHolder создается callback объект типа HolderCallback (реализует интерфейс SurfaceHolder.Callback), через который SurfaceHolder будет сообщать о состояниях surface. В нем должно быть переопределено 3 метода:

- surfaceCreated – surface создан. Можно дать камере объект holder с помощью метода setPreviewDisplay и начать транслировать изображение методом startPreview.
- surfaceChanged – был изменен формат или размер surface. В этом случае останавливается просмотр (stopPreview), камера настраивается с учетом поворота устройства (setCameraDisplayOrientation), и снова запускается просмотр.
- surfaceDestroyed – surface более недоступен.

В onResume осуществляется доступ к камере, используя метод open.

В onPause камера освобождается методом release, чтобы другие приложения могли ее использовать.

onClickPicture – это обработчик нажатия на кнопку Picture. Здесь происходит снимок. Для этого необходимо вызвать метод takePicture. Этот метод асинхронный, для получения результата используются callback'и:

- ShutterCallback, сработает сразу после того, как камера сделает снимок. Сюда можно повесить звук затвора, например.
- PictureCallback, вернет нам сырую raw картинку.

- PictureCallback, вернет нам готовую сжатую jpeg картинку. В его методе onPictureTaken происходит получение byte-массива. Это и есть готовое фото, которое необходимо записать в файл.

onClickStartRecord – обработчик нажатия на кнопку Start. Здесь включается запись видео с камеры. Для этого подготавливается MediaRecorder в методе prepareVideoRecorder. Метод вернет значение Boolean, по которому определяется удалось ли подготовить объект. Если он готов, стартует запись методом start. Если при подготовке возникли проблемы, то освобождаются ресурсы.

onClickStopRecord – обработчик кнопки Stop. Здесь останавливается запись видео методом stop и освобождаются ресурсы. Данный метод может выдавать RuntimeException, если что-то не так с аудио/видео, которое подали MediaRecorder'у на вход. Эту ошибку необходимо обрабатывать и удалять в таком случае записываемый файл, т.к. там будут некорректные данные.

prepareVideoRecorder – метод подготовки MediaRecorder к записи. Сначала методом unlock снимается монопольный доступ с камеры, чтобы MediaRecorder мог ее использовать. Затем создается объект MediaRecorder. Далее идут 6 методов его настройки:

- setCamera – предоставляем камеру, которая будет использована для записи
- setAudioSource: указывается источник звука – CAMCORDER
- setVideoSource: указывается источник видео – CAMERA
- setProfile: указывается профиль записи. В профиле содержаться такие данные как: тип контейнера, аудио/видео кодек, битрейт и пр.
- setOutputFile: указывается путь к файлу, куда будет записано видео
- setPreviewDisplay: указывается surface для показа превью в процессе записи. Если ранее для камеры уже указали surface в ее методе setPreviewDisplay, то этот метод вызывать необязательно.

Рекомендуется вызывать эти 6 методов именно в таком порядке, иначе велика вероятность неудачной записи.

Лабораторная работа №7

Использование геолокационных сервисов

Задание: Дополнить приложение, разработанное в рамках одной из предыдущих лабораторных работ, добавив в него работу с поиском текущего местоположения устройства.

Цель работы: Получение навыков использования в разрабатываемых мобильных приложения геолокационных сервисов.

Лабораторная работа №8

Освоение работы с базами данных в мобильном приложении

Задание: Разработать и отладить мобильное приложение, обеспечивающее подключение и работу с базой данных (например, SQLite для ОС Android). Приложение должно реализовывать создание, чтение, изменение и удаление отдельных записей.

Цель работы: Получение навыков работы с базами данных в разрабатываемых мобильных приложениях.

В Android существует встроенная поддержка базы данных SQLite. Поддерживаются все функции SQLite, предоставляется API оболочки с совместимым интерфейсом.

SQLite – компактная встраиваемая реляционная база данных. Слово «встраиваемый» (embedded) означает, что SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том устройстве, на котором исполняется программа. SQLite поддерживает динамическое типизирование данных.

Тип	Описание
NULL	пустое значение
INTEGER	целочисленное значение
REAL	значение с плавающей точкой
TEXT	строки или символы в кодировке UTF-8, UTF-16BE или UTF-16LE
NUMERIC	булевы значения, а также время и дата
BLOB	бинарные данные

База данных SQLite имеет некоторые особенности, о которых важно помнить:

- Не поддерживается проверка типов данных. Это значит, что Вы случайно сможете записать данные не того типа, например, поместить строку в колонку, предназначенную для целых чисел;
- Не поддерживается ссылочная целостность: нет поддержки FOREIGN KEY и конструкций JOIN;
- Полная поддержка Unicode отключена по умолчанию;
- Не поддерживается тип boolean. Используйте числа 0 для false и 1 для true;
- Не используйте тип BLOB для хранения данных (картинки) в Android. Лучше хранить в базе путь к изображениям, а сами изображения хранить в файловой системе.

Для упрощения работы с базами данных SQLite в Android нередко применяется класс SQLiteOpenHelper. Для использования создать необходимо создать класса-наследник от SQLiteOpenHelper, переопределив как минимум два его метода:

- onCreate(): вызывается при попытке доступа к базе данных, но когда еще эта база данных не создана;

- onUpgrade(): вызывается, когда необходимо обновление схемы базы данных. Здесь можно пересоздать ранее созданную базу данных в onCreate(), установив соответствующие правила преобразования от старой бд к новой.

Пример реализации такого класса:

```
public class DataBaseHelper extends SQLiteOpenHelper {

    // Название БД
    private static final String DATABASE_NAME = "userstore.db";
    // Версия БД
    private static final int SCHEMA = 1;
    // Название таблицы в БД
    public static final String TABLE = "users";
    // Названия столбцов
    public static final String COLUMN_ID = "_id";
    public static final String COLUMN_NAME = "name";
    public static final String COLUMN_AGE = "age";

    public DataBaseHelper(Context context) {
        super(context, DATABASE_NAME, null, SCHEMA);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE IF NOT EXISTS " + TABLE + " (" + COLUMN_ID + " INTEGER
PRIMARY KEY AUTOINCREMENT, " + COLUMN_NAME + " TEXT, " + COLUMN_AGE + " INTEGER);");
        db.execSQL("INSERT INTO " + TABLE + " (" + COLUMN_NAME + ", " + COLUMN_AGE + ")
VALUES ("Tom", 23);");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE);
        onCreate(db);
    }
}
```

Если база данных отсутствует или ее версия (которая задается в переменной SCHEMA) выше текущей, то срабатывает метод onCreate().

Для выполнения запросов к базе данных потребуется объект SQLiteDatabase, который представляет базу данных. Метод onCreate() получает в качестве параметра базу данных приложения.

Чтобы получить объект базы данных, надо использовать метод getReadableDatabase() (получение базы данных для чтения) или getWritableDatabase() (для записи соответственно).

```
dbHelper = new DataBaseHelper(this);
// Открываем подключение
db = dbHelper.getReadableDatabase();
// Получаем данные из БД в виде курсора
userCursor = db.rawQuery("SELECT * FROM " + dbHelper.TABLE, null);
if (userCursor.moveToFirst())
{
    do {
        Log.d(TAG, "Name:" + userCursor.getString(0) + "\nAge:" + userCursor.getInt(1));
    }
    while (userCursor.moveToNext());
}

userCursor.close();
db.close();
```

Для выполнения запросов к SQLite используется метод execSQL(). Он принимает sql-выражение CREATE TABLE, которое создает таблицу. Также при необходимости можно выполнить и другие запросы, например, добавить какие-либо начальные данные.

В методе onUpgrade() происходит обновление схемы БД. В данном случае для примера использован примитивный поход с удалением предыдущей базы данных с помощью sql-выражения DROP и последующим ее созданием. Но в реальности если будет необходимо сохранить данные, этот метод может включать более сложную логику - добавления новых столбцов, удаление ненужных, добавление дополнительных данных и т.д.

Android предоставляет различные способы для осуществления запросов к объекту SQLiteDatabase. В большинстве случаев можно применять метод rawQuery(), который принимает два параметра: SQL-выражение SELECT и дополнительный параметр, задающий параметры запроса.

После выполнения запроса rawQuery() возвращает объект Cursor, который хранит результат выполнения SQL-запроса.

Класс Cursor предлагает ряд методов для управления выборкой, в частности:

- getCount(): получает количество извлеченных из базы данных объектов;
- Методы moveToFirst() и moveToNext() позволяют переходить к первому и к следующему элементам выборки. Метод isAfterLast() позволяет проверить, достигнут ли конец выборки;
- Методы get*(columnIndex) (например, getLong(), getString()) позволяют по индексу столбца обратиться к данному столбцу текущей строки.

Дополнительно для управления курсором в Android имеется класс CursorAdapter. Он позволяет адаптировать полученный с помощью курсора набор к отображению в списковых элементах наподобие ListView. Как правило, при работе с курсором используется подкласс CursorAdapter – SimpleCursorAdapter. Хотя можно использовать и другие адаптеры, типа ArrayAdapter.

```
userAdapter = new SimpleCursorAdapter(this,
android.R.layout.two_line_list_item, userCursor, headers, new
int[]{android.R.id.text1, android.R.id.text2}, 0);
userList.setAdapter(userAdapter);
```

Конструктор класса SimpleCursorAdapter принимает шесть параметров:

1. Первым параметром выступает контекст, с которым ассоциируется адаптер, например, текущая activity;
2. Второй параметр – ресурс разметки интерфейса, который будет использоваться для отображения результатов выборки;
3. Третий параметр – курсор;
4. Четвертый параметр – список столбцов из выборки, которые будут отображаться в разметке интерфейса;
5. Пятый параметр – элементы внутри ресурса разметки, которые будут отображать значения столбцов из четвертого параметра;
6. Шестой параметр – флаги, задающие поведения адаптера.

```
public class DBManager {

    private DatabaseHelper dbHelper;

    private Context context;

    private SQLiteDatabase database;

    public DBManager(Context c) {
        context = c;
    }

    public DBManager open() throws SQLException {
        dbHelper = new DatabaseHelper(context);
        database = dbHelper.getWritableDatabase();
        return this;
    }

    public void close() {
        dbHelper.close();
    }

    public void insert(String name, String desc) { ContentValues
        contentValue = new ContentValues();
        contentValue.put(DatabaseHelper.SUBJECT, name);
        contentValue.put(DatabaseHelper.DESC, desc);
        database.insert(DatabaseHelper.TABLE_NAME, null, contentValue);
    }

    public Cursor fetch() {
        String[] columns = new String[]{DatabaseHelper._ID, DatabaseHelper.SUBJECT,
DatabaseHelper.DESC};
        Cursor cursor = database.query(DatabaseHelper.TABLE_NAME, columns, null, null,
null, null, null);
        if (cursor != null) {
            cursor.moveToFirst();
        }
        return cursor;
    }

    public int update(long _id, String name, String desc) {
        ContentValues contentValues = new ContentValues();
        contentValues.put(DatabaseHelper.SUBJECT, name);
        contentValues.put(DatabaseHelper.DESC, desc);
        int i = database.update(DatabaseHelper.TABLE_NAME, contentValues,
DatabaseHelper._ID + " = " + _id, null);
        return i;
    }

    public void delete(long _id) {
        database.delete(DatabaseHelper.TABLE_NAME, DatabaseHelper._ID + "=" + _id, null);
    }
}
```

При использовании CursorAdapter и его подклассов следует учитывать, что выборка курсора должна включать целочисленный столбец с названием _id, который должен быть уникальным для каждого элемента выборки. Значение этого столбца при нажатии на элемент списка затем передается в метод обработки onListItemClick(), благодаря чему мы можем по id идентифицировать нажатый элемент.

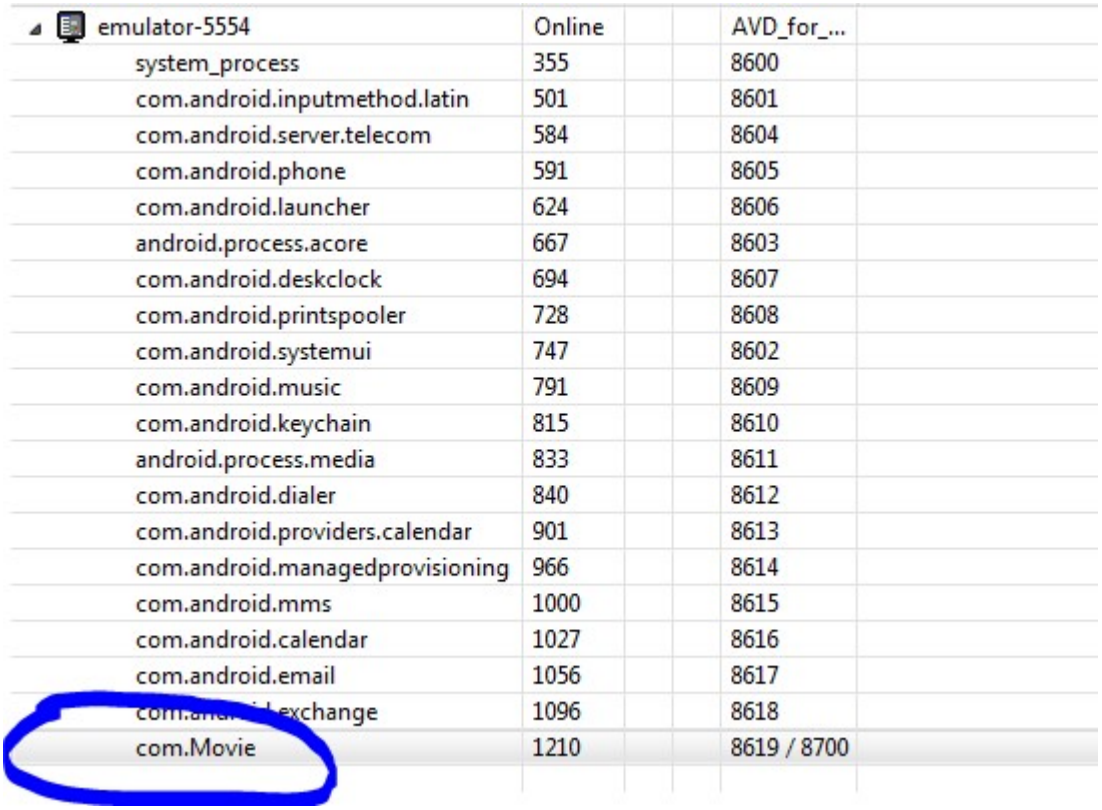
После завершения работы, курсор должен быть закрыт методом close().

Более подробная статья об особенностях работы с базой данных в Android.

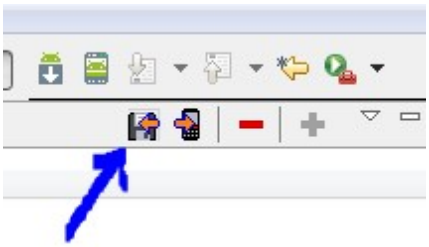
Просмотр баз данных в Android Studio:

Для просмотра баз данных при работе с Android Studio необходимо использовать SQLiteBrowser (http://sqlitebrowser.org/).

1. Открыть DDMS Tools -> Android -> Android Device Monitor
2. Нажмите на свое устройство слева. Вы должны увидеть свое приложение:



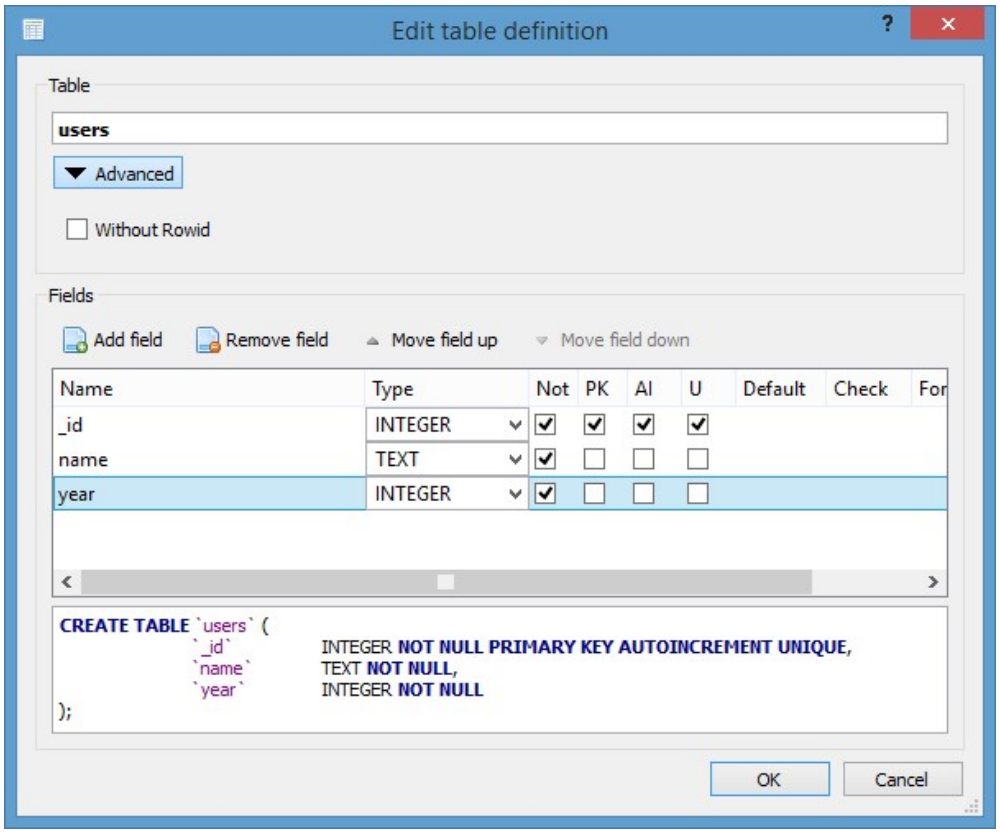
3. Перейдите в Проводник файлов (одна из вкладок справа), перейдите в */data/data/databases*
4. Выберите базу данных, просто щелкнув по ней.
5. Перейдите в верхний правый угол окна Android Device Monitor. Нажмите кнопку «pull a file from the device»:



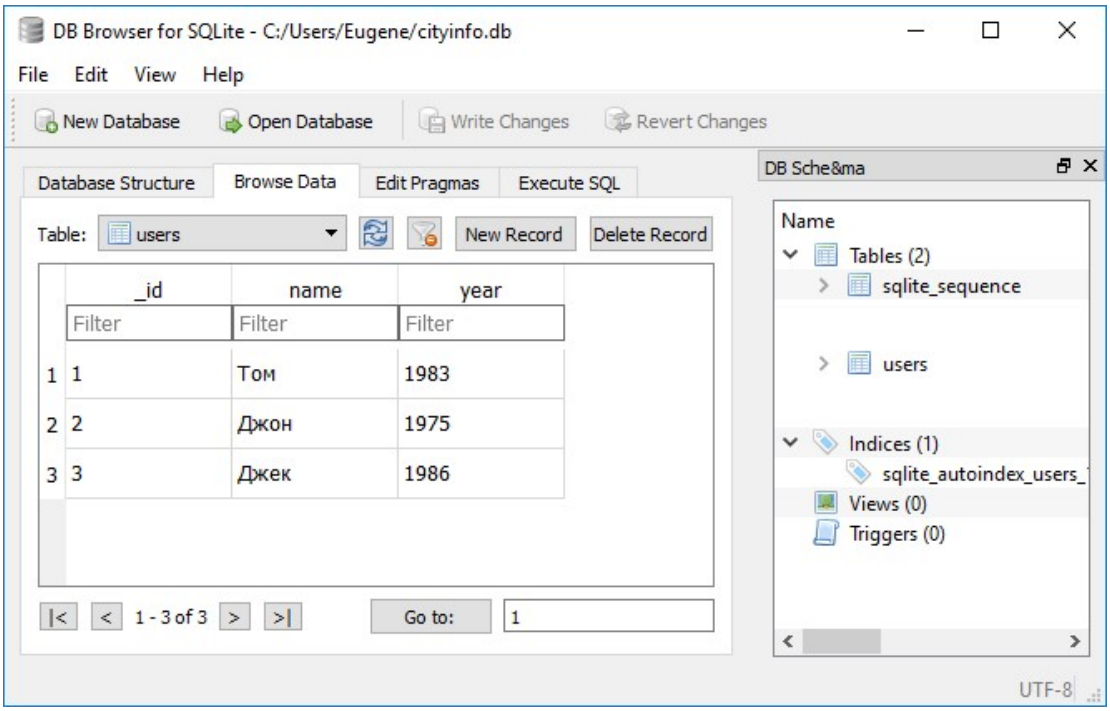
6. Откроется окно с вопросом, где вы хотите сохранить файл базы данных. Сохраните его в любом месте на своем компьютере.
7. Теперь откройте SQLiteBrowser, который вы установили. Нажмите «open database», перейдите к местоположению, в котором был сохранен файл базы данных, и откройте его. Теперь вы можете просматривать содержимое своей базы данных.

Использование существующей БД SQLite

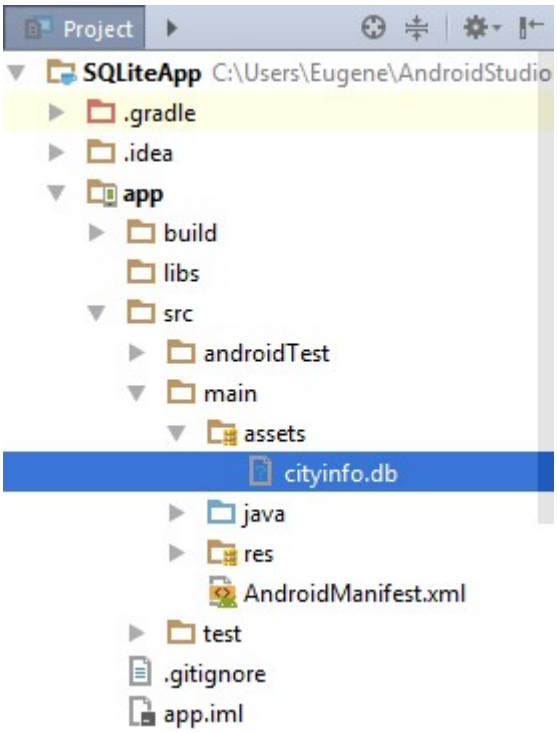
Для начала создадим базу данных SQLite при помощи SQLiteBrowser.



Сразу же в программе добавим несколько элементов в созданную таблицу:



После создания таблицы нужно добавить в проект в Android Studio папку assets, а в папку assets – только что созданную базу данных. Для этого нужно перейти к полному определению проекта, нажать на папку main правой кнопкой мыши и в меню выбрать New -> Directory: Название добавляемой папку assets и затем скопируйте в нее базу данных:



В таком случае код класса DatabaseHelper должен выглядеть следующем образом:

```
class DataBaseHelper extends SQLiteOpenHelper {
    // полный путь к базе данных
    private static String DB_PATH;
    private static String DB_NAME = "cityinfo.db";
    // версия базы данных
    private static final int SCHEMA = 1;
    // название таблицы в бд
    static final String TABLE = "users";
    // названия столбцов
    static final String COLUMN_ID = "_id";
    static final String COLUMN_NAME = "name";
    static final String COLUMN_YEAR = "year";
    private Context myContext;

    DataBaseHelper(Context context) {
        super(context, DB_NAME, null, SCHEMA);
        this.myContext = context;
        DB_PATH = context.getFilesDir().getPath() + DB_NAME;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }

    void create_db() {
        InputStream myInput = null;
        OutputStream myOutput = null;
        try {
            File file = new File(DB_PATH);
            if (!file.exists()) {
                this.getReadableDatabase();
                // получаем локальную бд как поток
                myInput = myContext.getAssets().open(DB_NAME);
```

```
                // Путь к новой бд
                String outFileName = DB_PATH;

                // Открываем пустую бд
                myOutput = new FileOutputStream(outFileName);

                // побайтово копируем данные
                byte[] buffer = new byte[1024];
                int length;
                while ((length = myInput.read(buffer)) > 0) {
                    myOutput.write(buffer, 0, length);
                }

                myOutput.flush();
                myOutput.close();
                myInput.close();
            }
        } catch (IOException ex) {
            Log.d("DatabaseHelper", ex.getMessage());
        }
    }

    public SQLiteDatabase open() throws SQLException {

        return SQLiteDatabase.openDatabase(DB_PATH, null, SQLiteDatabase.OPEN_READWRITE);
    }
}
```

Лабораторная работа №9

Работа с сетевыми сервисами в мобильных приложениях

Задание: Дополнить приложение разработанное в рамках одной из предыдущих лабораторных работ, добавив в него работу с внешним сетевым сервисом (взаимодействие с сервером).

Обязательная часть: Реализовать взаимодействие с внешним сервисом. Если не выполняется рекомендательная часть работы, то реализуется взаимодействие с уже готовым внешним сервисом (например, «Вконтакте», «Instagram», «Twitter»)
Рекомендуется: Реализовать серверную часть самомостоятельно на любом языке высокого уровня. При работе сервер должен открывать REST сервис для работы с внешними клиентами. При такой реализации сервер будет выступать в роли backend'а, а мобильное приложение в роли frontend'а.
Не обязательная часть: Реализовать всю работу с серверной частью асинхронно.

Цель работы: Получение практических навыков использования сетевых сервисов при разработке приложений для мобильных устройств.

Для работы с сетью необходимо использовать пакеты из java.net.*.

Элемент <uses-permission> запрашивает разрешение, которые приложению должны быть предоставлены системой для его нормального функционирования. Разрешения предоставляются во время установки приложения, а не во время его работы.

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

Запрашивают разрешения на доступ к интернету, информацию о сетевых соединениях и получение сообщений о загрузке устройства.

AndroidManifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="by.andreidanilevich.temp_chat"
    android:versionCode="1"
    android:versionName="1.0" >

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

    <uses-sdk android:minSdkVersion="16"
        android:targetSdkVersion="22" />

    <application android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

```
        </activity>
        <activity android:name=".ChatActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait" >
        </activity>

        <receiver
            android:name=".AutoRun"
            android:enabled="true"
            android:exported="false" >
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>

        <service android:name=".BackgroundService" />
    </application>

</manifest>
```

chat.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#999999" >

    <ListView android:id="@+id/lv"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginBottom="60dp"
        android:scrollbars="none"
        android:stackFromBottom="true"
        android:transcriptMode="alwaysScroll" >
    </ListView>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_alignParentBottom="true"
        android:background="#ffffff" >

        <EditText android:id="@+id/et"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="top"
            android:layout_weight="1"
            android:ems="10" >
        </EditText>

        <Button android:id="@+id/bt"
            android:layout_width="50dp"
            android:layout_height="wrap_content"
            android:layout_gravity="top"
            android:onClick="send"
            android:text=">" />
    </LinearLayout>

</RelativeLayout>
```

main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"

    <Spinner android:id="@+id/spinner_author"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Spinner android:id="@+id/spinner_client"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button android:id="@+id/open_chat_btn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="open_chat" />

    <Button
        android:id="@+id/open_chat_reverse_btn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="open_chat_reverse" />

    <Button android:id="@+id/delete_server_chat"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:onClick="delete_server_chats"
        android:text="Удалить все чаты на сервере!" />

    <Button android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="delete_local_chats"
        android:text="Удалить все чаты на этом устройстве!" />

</LinearLayout>
```

list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#999999"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp" >

        <TextView android:id="@+id/author"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
```

```
            android:gravity="left"
            android:textColor="#c6c6c6"
            android:textSize="12sp" />

        <TextView android:id="@+id/client"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="left"
            android:textColor="#c6c6c6"
            android:textSize="12sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="5dp"
        android:layout_marginTop="5dp" >

        <TextView android:id="@+id/list_author"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="left"
            android:textColor="#0000ff"
            android:textSize="14sp"
            android:textStyle="bold" />

        <TextView android:id="@+id/list_client"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="left"
            android:textColor="#ffff00"
            android:textSize="14sp"
            android:textStyle="bold" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="5dp" >

        <TextView
            android:id="@+id/list_author_time"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="right"
            android:textColor="#c6c6c6"
            android:textSize="12sp" />

        <TextView
            android:id="@+id/list_client_time"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="right"
            android:textColor="#c6c6c6"
            android:textSize="12sp" />
    </LinearLayout>
```

```
</LinearLayout>
```

BackgroundService.java

```
public class BackgroundService extends Service {

    // ИМЯ СЕРВЕРА (url зарегистрированного нами сайта)
    // например http://l29340eb.bget.ru
    String server_name = "http://l29340eb.bget.ru";

    SQLiteDatabase chatDBlocal;
    HttpURLConnection conn;
    Cursor cursor;
    Thread thr;
    ContentValues new_mess;
    Long last_time; // время последней записи в БД, отсекаем по нему что нам
    // тянуть с сервера, а что уже есть

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    public void onStart(Intent intent, int startId) {

        Log.i("chat", "+ BackgroundService - запуск сервиса");

        chatDBlocal = openOrCreateDatabase("chatDBlocal.db",
                Context.MODE_PRIVATE, null);
        chatDBlocal
                .execSQL("CREATE TABLE IF NOT EXISTS chat (_id integer primary key
autoincrement, author, client, data, text)");

        // создадим и покажем notification
        // это позволит стать сервису "бессмертным"
        // и будет визуально видно в трее
        Intent iN = new Intent(getApplicationContext(), MainActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP
                | Intent.FLAG_ACTIVITY_SINGLE_TOP);
        PendingIntent pI = PendingIntent.getActivity(getApplicationContext(),
                0, iN, PendingIntent.FLAG_CANCEL_CURRENT);
        Notification.Builder bl = new Notification.Builder(
                getApplicationContext());

        bl.setContentIntent(pI)
                .setSmallIcon(R.drawable.ic_launcher)
                .setLargeIcon(
                        BitmapFactory.decodeResource(getApplicationContext()
                                .getResources(), R.drawable.ic_launcher))
                .setAutoCancel(true)
                .setContentTitle(getResources().getString(R.string.app_name))
                .setContentText("работаю...");

        Notification notification = bl.build();
        startForeground(101, notification);

        startLoop();
    }

    // запуск потока, внутри которого будет происходить
    // регулярное соединение с сервером для чтения новых
    // сообщений.
    // если сообщения найдены - отправим броадкаст для обновления
    // ListView в ChatActivity
```

```
    private void startLoop() {

        thr = new Thread(new Runnable() {

            // ansver = ответ на запрос
            // lnk = линк с параметрами
            String ansver, lnk;

            public void run() {

                while (true) { // стартуем бесконечный цикл

                    // глянем локальную БД на наличие сообщщений чата
                    cursor = chatDBlocal.rawQuery(
                            "SELECT * FROM chat ORDER BY data", null);

                    // если какие-либо сообщения есть - формируем запрос
                    // по которому получим только новые сообщения
                    if (cursor.moveToLast()) {
                        last_time = cursor.getLong(cursor
                                .getColumnIndex("data"));
                        lnk = server_name + "/chat.php?action=select&data="
                                + last_time.toString();

                        // если сообщений в БД нет - формируем запрос
                        // по которому получим всё
                    } else {
                        lnk = server_name + "/chat.php?action=select";
                    }

                    cursor.close();

                    // создаем соединение ---------------------------------->
                    try {
                        Log.i("chat",
                                "+ BackgroundService --------------- ОТКРОЕМ
СОЕДИНЕНИЕ");

                        conn = (HttpURLConnection) new URL(lnk)
                                .openConnection();
                        conn.setReadTimeout(10000);
                        conn.setConnectTimeout(15000);
                        conn.setRequestMethod("POST");
                        conn.setRequestProperty("User-Agent", "Mozilla/5.0");
                        conn.setDoInput(true);
                        conn.connect();

                    } catch (Exception e) {
                        Log.i("chat", "+ BackgroundService ошибка: " + e.getMessage());
                    }
                    // получаем ответ ---------------------------------->
                    try {
                        InputStream is = conn.getInputStream();
                        BufferedReader br = new BufferedReader(
                                new InputStreamReader(is, "UTF-8"));
                        StringBuilder sb = new StringBuilder();
                        String bfr_st = null;
                        while ((bfr_st = br.readLine()) != null) {
                            sb.append(bfr_st);
                        }

                        Log.i("chat", "+ BackgroundService - полный ответ сервера:\n"
                                + sb.toString());
                        // сформируем ответ сервера в string
```

```
                    // обрежем в полученном ответе все, что находится за "]"
                    // это необходимо, т.к. json ответ приходит с мусором
                    // и если этот мусор не убрать - будет невалидным
                    answer = sb.toString();
                    answer = answer.substring(0, answer.indexOf("]") + 1);

                    is.close(); // закроем поток
                    br.close(); // закроем буфер

                } catch (Exception e) {
                    Log.i("chat", "+ BackgroundService ошибка: " + e.getMessage());
                } finally {
                    conn.disconnect();
                    Log.i("chat",
                            "+ BackgroundService ---------- ЗАКРОЕМ
СОЕДИНЕНИЕ");
                }

                // запишем ответ в БД ---------------------------------->
                if (answer != null && !answer.trim().equals("")) {

                    Log.i("chat",
                            "+ BackgroundService ---------- ответ содержит JSON:");

                    try {
                        // ответ превратим в JSON массив
                        JSONArray ja = new JSONArray(answer);
                        JSONObject jo;

                        Integer i = 0;

                        while (i < ja.length()) {

                            // разберем JSON массив построчно
                            jo = ja.getJSONObject(i);

                            Log.i("chat",
                                    "================>>> "
                                            + jo.getString("author")
                                            + " | "
                                            + jo.getString("client")
                                            + " | " + jo.getLong("data")
                                            + " | " + jo.getString("text"));

                            // создадим новое сообщение new_mess = new
                            ContentValues(); new_mess.put("author",
                            jo.getString("author")); new_mess.put("client",
                            jo.getString("client")); new_mess.put("data",
                            jo.getLong("data")); new_mess.put("text",
                            jo.getString("text"));
                            // запишем новое сообщение в БД
                            chatDBlocal.insert("chat", null, new_mess);
                            new_mess.clear();

                            i++;

                            // отправим броадкаст для ChatActivity
                            // если она открыта - она обновить ListView
                            sendBroadcast(new Intent(
                                    "by.andreidanilevich.action.UPDATE_ListView"));
                        }
                    } catch (Exception e) {
                        // если ответ сервера не содержит валидный JSON
                        Log.i("chat",
```

```
                                    "+ BackgroundService ---------- ошибка ответа
сервера:\n"
                                            + e.getMessage());
                        }
                    } else {
                        // если ответ сервера пустой
                        Log.i("chat",
                                "+ BackgroundService ---------- ответ не содержит
JSON!");
                    }

                    try {
                        Thread.sleep(15000);
                    } catch (Exception e) {
                        Log.i("chat",
                                "+ BackgroundService - ошибка процесса: "
                                        + e.getMessage());
                    }
                }
            }
        });

        thr.setDaemon(true);
        thr.start();

    }
}
```

ChatActivity.java

```
public class ChatActivity extends Activity {

    // ИМЯ СЕРВЕРА (url зарегистрированного нами сайта)
    // например http://l29340eb.bget.ru
    String server_name = "http://l29340eb.bget.ru";

    ListView lv; // полоса сообщений
    EditText et;
    Button bt;
    SQLiteDatabase chatDBlocal;
    String author, client;
    INSERTtoChat insert_to_chat; // класс отправляет новое сообщение на сервер
    UpdateReceiver upd_res; // класс ждет сообщение от сервиса и получив его -
    // обновляет ListView

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.chat);

        // получим 2 переменные по которым будем отбирать инфу из БД:
        // author - от чьего имени идет чат
        // client - с кем чатимся
        Intent intent = getIntent();
        author = intent.getStringExtra("author");
        client = intent.getStringExtra("client");

        Log.i("chat", "+ ChatActivity - открыт author = " + author
                + " | client = " + client);

        lv = (ListView) findViewById(R.id.lv);
        et = (EditText) findViewById(R.id.et);
        bt = (Button) findViewById(R.id.bt);

        chatDBlocal = openOrCreateDatabase("chatDBlocal.db",
```

```
                Context.MODE_PRIVATE, null);
        chatDBlocal
                .execSQL("CREATE TABLE IF NOT EXISTS chat (_id integer primary key
autoincrement, author, client, data, text)");

        // Создаём и регистрируем широковещательный приёмник

        upd_res = new UpdateReceiver();
        registerReceiver(upd_res, new IntentFilter(
                "by.andreidanilevich.action.UPDATE_ListView"));

        create_lv();
    }

    // обновим lv = заполним нужные позиции в lv информацией из БД
    @SuppressLint("SimpleDateFormat")
    public void create_lv() {

        Cursor cursor = chatDBlocal.rawQuery(
                "SELECT * FROM chat WHERE author = "" + author
                        + "" OR author = "" + client + "" ORDER BY data", null);
        if (cursor.moveToFirst()) {
            // если в базе есть элементы соответствующие
            // нашим критериям отбора

            // создадим массив, создадим hashmap и заполним его результатом
            // cursor
            ArrayList<HashMap<String, Object>> mList = new ArrayList<HashMap<String,
Object>>();
            HashMap<String, Object> hm;

            do {
                // мое сообщение !!!
                // если автор сообщения = автор
                // и получатель сообщения = клиент
                if (cursor.getString(cursor.getColumnIndex("author")).equals(
                        author)
                        && cursor.getString(cursor.getColumnIndex("client"))
                        .equals(client)) {

                    hm = new HashMap<>();
                    hm.put("author", author);
                    hm.put("client", "");
                    hm.put("list_client", "");
                    hm.put("list_client_time", "");
                    hm.put("list_author",
                            cursor.getString(cursor.getColumnIndex("text")));
                    hm.put("list_author_time", new SimpleDateFormat(
                            "HH:mm - dd.MM.yyyy").format(new Date(cursor
                            .getLong(cursor.getColumnIndex("data")))));
                    mList.add(hm);

                }

                // сообщение мне !!!!!!!
                // если автор сообщения = клиент
                // и если получатель сообщения = автор
                if (cursor.getString(cursor.getColumnIndex("author")).equals(
                        client)
                        && cursor.getString(cursor.getColumnIndex("client"))
                        .equals(author)) {

                    hm = new HashMap<>();
                    hm.put("author", "");
```

```
                hm.put("client", client);
                hm.put("list_author", "");
                hm.put("list_author_time", "");
                hm.put("list_client",
                        cursor.getString(cursor.getColumnIndex("text")));
                hm.put("list_client_time", new SimpleDateFormat(
                        "HH:mm - dd.MM.yyyy").format(new Date(cursor
                        .getLong(cursor.getColumnIndex("data")))));
                mList.add(hm);

            }

        } while (cursor.moveToNext());

        // покажем lv
        SimpleAdapter adapter = new SimpleAdapter(getApplicationContext(),
                mList, R.layout.list, new String[]{"list_author",
                "list_author_time", "list_client",
                "list_client_time", "author", "client"},
                new int[]{R.id.list_author, R.id.list_author_time,
                        R.id.list_client, R.id.list_client_time,
                        R.id.author, R.id.client});

        lv.setAdapter(adapter);
        cursor.close();

    }

    Log.i("chat",
            "+ ChatActivity ======================== обновили поле чата");

}

public void send(View v) {
    // запишем наше новое сообщение
    // вначале проверим на пустоту

    if (!et.getText().toString().trim().equals("")) {

        // кнопку сделаем неактивной
        bt.setEnabled(false);

        // если чтото есть - действуем!
        insert_to_chat = new INSERTtoChat();
        insert_to_chat.execute();

    } else {
        // если ничего нет - нечего и писать
        et.setText("");
    }
}

// отправим сообщение на сервер

private class INSERTtoChat extends AsyncTask<Void, Void, Integer> {

    HttpURLConnection conn;
    Integer res;
    string myUrlAddress = "/chat.php?action=insert&author=";

    protected Integer doInBackground(Void... params) {

        try {
```

```
            // соберем линк для передачи новой строки
            String post_url = server_name
                    + myUrlAddress
                    + URLEncoder.encode(author, "UTF-8")
                    + "&client="
                    + URLEncoder.encode(client, "UTF-8")
                    + "&text="
                    + URLEncoder.encode(et.getText().toString().trim(),
                    "UTF-8");

            Log.i("chat",
                    "+ ChatActivity - отправляем на сервер новое сообщение: "
                            + et.getText().toString().trim());

            URL url = new URL(post_url);
            conn = (HttpURLConnection) url.openConnection();
            conn.setConnectTimeout(10000); // ждем 10сек
            conn.setRequestMethod("POST");
            conn.setRequestProperty("User-Agent", "Mozilla/5.0");
            conn.connect();

            res = conn.getResponseCode();
            Log.i("chat", "+ ChatActivity - ответ сервера (200 - все ОК): "
                    + res.toString());

        } catch (Exception e) {
            Log.i("chat",
                    "+ ChatActivity - ошибка соединения: " + e.getMessage());

        } finally {
            // закроем соединение
            conn.disconnect();
        }
        return res;
    }

    protected void onPostExecute(Integer result) {

        try {
            if (result == 200) {
                Log.i("chat", "+ ChatActivity - сообщение успешно ушло.");
                // сбросим набранный текст
                et.setText("");
            }
        } catch (Exception e) {
            Log.i("chat", "+ ChatActivity - ошибка передачи сообщения:\n"
                    + e.getMessage());
            Toast.makeText(getApplicationContext(),
                    "ошибка передачи сообщения", Toast.LENGTH_SHORT).show();
        } finally {
            // активируем кнопку
            bt.setEnabled(true);
        }
    }
}

// ресивер приёмник ждет сообщения от BackgroundService
// если сообщение пришло, значит есть новая запись в БД - обновим ListView
public class UpdateReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.i("chat",
                "+ ChatActivity - ресивер получил сообщение - обновим ListView");
        create_lv();
```

```
        }
    }

    // выходим из чата
    public void onBackPressed() {
        Log.i("chat", "+ ChatActivity - закрыт");
        unregisterReceiver(upd_res);
        finish();
    }
}
```

MainActivity.java

```
public class MainActivity extends Activity {

    // ИМЯ СЕРВЕРА (url зарегистрированного нами сайта)
    // например http://l29340eb.bget.ru
    String server_name = "http://l29340eb.bget.ru";

    Spinner spinner_author, spinner_client;
    String author, client;
    Button open_chat_btn, open_chat_reverse_btn, delete_server_chat;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Log.i("chat", "+ MainActivity - запуск приложения");

        open_chat_btn = (Button) findViewById(R.id.open_chat_btn);
        open_chat_reverse_btn = (Button) findViewById(R.id.open_chat_reverse_btn);
        delete_server_chat = (Button) findViewById(R.id.delete_server_chat);

        // запустим BackgroundService
        this.startService(new Intent(this, BackgroundService.class));

        // заполним 2 выпадающих меню для выбора автора и получателя сообщения
        // 5 мужских и 5 женских имен
        // установим слушателей
        spinner_author = (Spinner) findViewById(R.id.spinner_author);
        spinner_client = (Spinner) findViewById(R.id.spinner_client);

        spinner_author.setAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_spinner_item, new String[]{"Петя",
                "Вася", "Коля", "Андрей", "Сергей", "Оля", "Лена",
                "Света", "Марина", "Наташа"}));
        spinner_client.setAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_spinner_item, new String[]{"Петя",
                "Вася", "Коля", "Андрей", "Сергей", "Оля", "Лена",
                "Света", "Марина", "Наташа"}));
        spinner_client.setSelection(5);

        open_chat_btn.setText("Открыть чат: "
                + spinner_author.getSelectedItem().toString() + " > "
                + spinner_client.getSelectedItem().toString());
        open_chat_reverse_btn.setText("Открыть чат: "
                + spinner_client.getSelectedItem().toString() + " > "
                + spinner_author.getSelectedItem().toString());

        spinner_author
                .setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
                    public void onItemSelected(AdapterView<?> parent,
                                               View itemSelected, int
selectedItemPosition,
```

```
                                            long selectedId) {

                        author = spinner_author.getSelectedItem().toString();

                        open_chat_btn.setText("Открыть чат: "
                                + spinner_author.getSelectedItem().toString()
                                + " > "
                                + spinner_client.getSelectedItem().toString());
                        open_chat_reverse_btn.setText("Открыть чат: "
                                + spinner_client.getSelectedItem().toString()
                                + " > "
                                + spinner_author.getSelectedItem().toString());
                    }

                    public void onNothingSelected(AdapterView<?> parent) {
                    }
                });

        spinner_client
                .setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
                    public void onItemSelected(AdapterView<?> parent,
                                               View itemSelected, int
selectedItemPosition,
                                               long selectedId) {

                        client = spinner_client.getSelectedItem().toString();

                        open_chat_btn.setText("Открыть чат: "
                                + spinner_author.getSelectedItem().toString()
                                + " > "
                                + spinner_client.getSelectedItem().toString());
                        open_chat_reverse_btn.setText("Открыть чат: "
                                + spinner_client.getSelectedItem().toString()
                                + " > "
                                + spinner_author.getSelectedItem().toString());
                    }

                    public void onNothingSelected(AdapterView<?> parent) {
                    }
                });
    }

    // откроем чат с выбранным автором и получателем
    public void open_chat(View v) {
        // быстрая проверка
        if (author.equals(client)) {
            // если автор и получатель одинаковы
            // чат не открываем
            Toast.makeText(this, "author = client !", Toast.LENGTH_SHORT)
                    .show();
        } else {
            // откроем нужный чат author > client
            Intent intent = new Intent(MainActivity.this, ChatActivity.class);
            intent.putExtra("author",  author);
            intent.putExtra("client",  client);
            startActivity(intent);
        }
    }

    // откроем чат с выбранным автором и получателем, только наоборот
    public void open_chat_reverse(View v) {
        // быстрая проверка
        if (author.equals(client)) {
            // если автор и получатель одинаковы
```

```
            // чат не открываем
            Toast.makeText(this, "author = client !", Toast.LENGTH_SHORT)
                    .show();
        } else {
            // откроем нужный чат client > author
            Intent intent = new Intent(MainActivity.this, ChatActivity.class);
            intent.putExtra("author",  client);
            intent.putExtra("client",  author);
            startActivity(intent);
        }
    }

    // отправим запрос на сервер о удалении таблицы с чатами
    public void delete_server_chats(View v) {

        Log.i("chat", "+ MainActivity - запрос на удаление чата с сервера");

        delete_server_chat.setEnabled(false);
        delete_server_chat.setText("Запрос отправлен. Ожидайте...");

        DELETEfromChat delete_from_chat = new DELETEfromChat();
        delete_from_chat.execute();
    }

    // удалим локальную таблицу чатов
    // и создадим такуюже новую
    public void delete_local_chats(View v) {

        Log.i("chat", "+ MainActivity - удаление чата с этого устройства");

        SQLiteDatabase chatDBlocal;
        chatDBlocal = openOrCreateDatabase("chatDBlocal.db",
                Context.MODE_PRIVATE, null);
        chatDBlocal.execSQL("drop table chat");
        chatDBlocal
                .execSQL("CREATE TABLE IF NOT EXISTS chat (_id integer primary key
autoincrement, author, client, data, text)");

        Toast.makeText(getApplicationContext(),
                "Чат на этом устройстве удален!", Toast.LENGTH_SHORT).show();
    }

    // отправим запрос на сервер о удалении таблицы с чатами
    // если он пройдет - таблица будет удалена
    // если не пройдет (например нет интернета или сервер недоступен)
    // - покажет сообщение
    private class DELETEfromChat extends AsyncTask<Void, Void, Integer> {

        Integer res;
        HttpURLConnection conn;

        protected Integer doInBackground(Void... params) {

            try {
                URL url = new URL(server_name + "/chat.php?action=delete");
                conn = (HttpURLConnection) url.openConnection();
                conn.setConnectTimeout(10000); // ждем 10сек
                conn.setRequestMethod("POST");
                conn.setRequestProperty("User-Agent", "Mozilla/5.0");
                conn.connect();
                res = conn.getResponseCode();
                Log.i("chat", "+ MainActivity - ответ сервера (200 = ОК): "
                        + res.toString());
```

```
            } catch (Exception e) {
                Log.i("chat",
                        "+ MainActivity - ответ сервера ОШИБКА: "
                                + e.getMessage());
            } finally {
                conn.disconnect();
            }

            return res;
        }

        protected void onPostExecute(Integer result) {

            try {
                if (result == 200) {
                    Toast.makeText(getApplicationContext(),
                            "Чат на сервере удален!", Toast.LENGTH_SHORT)
                            .show();
                }
            } catch (Exception e) {
                Toast.makeText(getApplicationContext(),
                        "Ошибка выполнения запроса.", Toast.LENGTH_SHORT)
                        .show();
            } finally {
                // сделаем кнопку активной
                delete_server_chat.setEnabled(true);
                delete_server_chat.setText("Удалить все чаты на сервере!");
            }
        }
    }

    public void onBackPressed() {
        Log.i("chat", "+ MainActivity - выход из приложения");
        finish();
    }
}
```

AutoRun.java

```
public class AutoRun extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals("android.intent.action.BOOT_COMPLETED")) {

            // получили boot_completed - запустили BackgroundService
            context.startService(new Intent(context, BackgroundService.class));
            Log.i("chat", "+ AutoRun - отработал");
        }
    }
}
```

В фоне работает BackgroundService.java, который, в отдельном потоке, каждые 15 секунд делает запрос на сервер. Если ответ сервера содержит новые сообщения, BackgroundService.java записывает их в локальную БД и отправляет сообщение ChatActivity.java о необходимости обновить ListView, с сообщениями. ChatActivity.java (если она в этот момент открыта) получает сообщение и обновляет содержимое ListView из локальной БД.

Отправка нового сообщения из ChatActivity.java происходит сразу на сервер, минуя BackgroundService.java. При этом сообщение НЕ записывается в локальную БД! Там оно появится только после получения его назад в виде ответа сервера.

Создавая новое сообщение, мы передаем запросом на сервер: имя автора сообщения, имя получателя сообщения, текст сообщения. Получая эту запись назад, в виде ответа сервера, мы получаем то, что отправляли + четвертый параметр: время получения сообщения сервером.

Реализация REST-сервиса на C#

HTTP протокол описывает взаимодействие между двумя компьютерами (клиентом и сервером), построенное на базе сообщений, называемых запрос (Request) и ответ (Response). Каждое сообщение состоит из трех частей: стартовая строка, заголовки и тело. При этом обязательной является только стартовая строка.

Стартовые строки для запроса и ответа имеют различный формат — стартовая строка запроса, выглядит так:

METHOD URI HTTP/*VERSION*,

где METHOD — это как раз метод HTTP-запроса, URI — идентификатор ресурса, VERSION — версия протокола.

Заголовки — это набор пар имя-значение, разделенных двоеточием. В заголовках передается различная служебная информация: кодировка сообщения, название и версия браузера, адрес, с которого пришел клиент (Referrer) и так далее.

Тело сообщения — это, собственно, передаваемые данные. В ответе передаваемыми данными, как правило, является html-страница, которую запросил браузер, а в запросе, например, в теле сообщения передается содержимое файлов, загружаемых на сервер. Но как правило, тело сообщения в запросе вообще отсутствует.

В стартовой строке запроса присутствует такой параметр, как URI (Uniform Resource Identifier) — единообразный идентификатор ресурса. Ресурс — это, как правило, файл на сервере, но вообще ресурсом может являться и какой-либо абстрактный объект ('/blogs/webdev/' — указывает на блок, а не на конкретный файл).

Тип HTTP-запроса (также называемый HTTP-метод) указывает серверу на то, какое действие мы хотим произвести с ресурсом. Изначально (в начале 90-х) предполагалось, что клиент может хотеть от ресурса только одно — получить его, однако сейчас по протоколу HTTP можно создавать посты, редактировать профиль, удалять сообщения и многое другое. И эти действия сложно объединить термином «получение».

Для разграничения действий с ресурсами на уровне HTTP-методов и были придуманы следующие варианты:

- GET — получение ресурса;
- POST — создание ресурса;
- PUT — обновление ресурса;
- DELETE — удаление ресурса.

Обратите внимание на тот факт, что спецификация HTTP не обязывает сервер понимать все методы (которых на самом деле гораздо больше, чем 4) — обязателен только GET, а также не указывает серверу, что он должен делать при получении запроса с тем или иным методом. А это значит, что сервер в ответ на запрос DELETE /index.php HTTP/1.1 *не*

обязан удалять страницу index.php на сервере, так же как на запрос GET /index.php HTTP/1.1 *не обязан* возвращать вам страницу index.php, например, он может ее удалить.

REST (Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. Вообще REST охватывает более широкую область, нежели HTTP — его можно применять и в других сетях с другими протоколами. REST описывает принципы взаимодействия клиента и сервера, основанные на понятиях «ресурса» и «запроса». Для веб-служб, построенных с учётом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «RESTful».

В отличие от веб-сервисов (веб-служб) на основе SOAP, не существует «официального» стандарта для RESTful веб-API. Дело в том, что REST является архитектурным стилем, в то время как SOAP является протоколом. Несмотря на то, что REST не является стандартом сам по себе, большинство RESTful-реализаций используют стандарты, такие как HTTP, URL, JSON и XML.

Пример реализации на C# с использованием WCF:

App.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />
  </startup>

  <system.serviceModel>
    <services>
      <service name="Server.Services.Client">
        <endpoint address="" binding="webHttpBinding"
contract="Contracts.Interfaces.IClient" />
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8084/client/" />
          </baseAddresses>
        </host>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

app.manifest

```
<?xml version="1.0" encoding="utf-8"?>
<assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1">
  <assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
        <requestedExecutionLevel level="requireAdministrator" uiAccess="false" />
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

Program.cs

```
using System;
```

```
using System.ServiceModel.Web;

namespace Server
{
    internal class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        private static void Main()
        {
            try
            {
                using (var hostClient = new WebServiceHost(typeof(Services.Client)))
                {
                    hostClient.Open();
                    Console.ReadKey();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
                Console.ReadKey();
            }
        }
    }
}
```

IClient.cs

```
using System.ServiceModel;
using System.ServiceModel.Web;

namespace Contracts.Interfaces
{
    [ServiceContract]
    public interface IClient
    {
        /// <summary>
        /// Вернуть информацию о всех актуальных терминалах.
        /// </summary>
        [WebInvoke(BodyStyle = WebMessageBodyStyle.Bare, Method = "GET", ResponseFormat =
WebMessageFormat.Json, UriTemplate = "getterminalinfo")]
        [OperationContract]
        int GetTerminalsInfo();

        /// <summary>
        /// Вернуть все данные конкретного терминала.
        /// </summary>
        [WebInvoke(BodyStyle = WebMessageBodyStyle.Bare, Method = "GET", ResponseFormat =
WebMessageFormat.Json, UriTemplate = "getdata/{terminalId}")]
        [OperationContract]
        object GetDataByTerminalId(string terminalId);
    }
}
```

Client.cs

```
using Contracts.Interfaces;

namespace Server.Services
{
    internal class Client : IClient
    {
        public int GetTerminalsInfo()
```

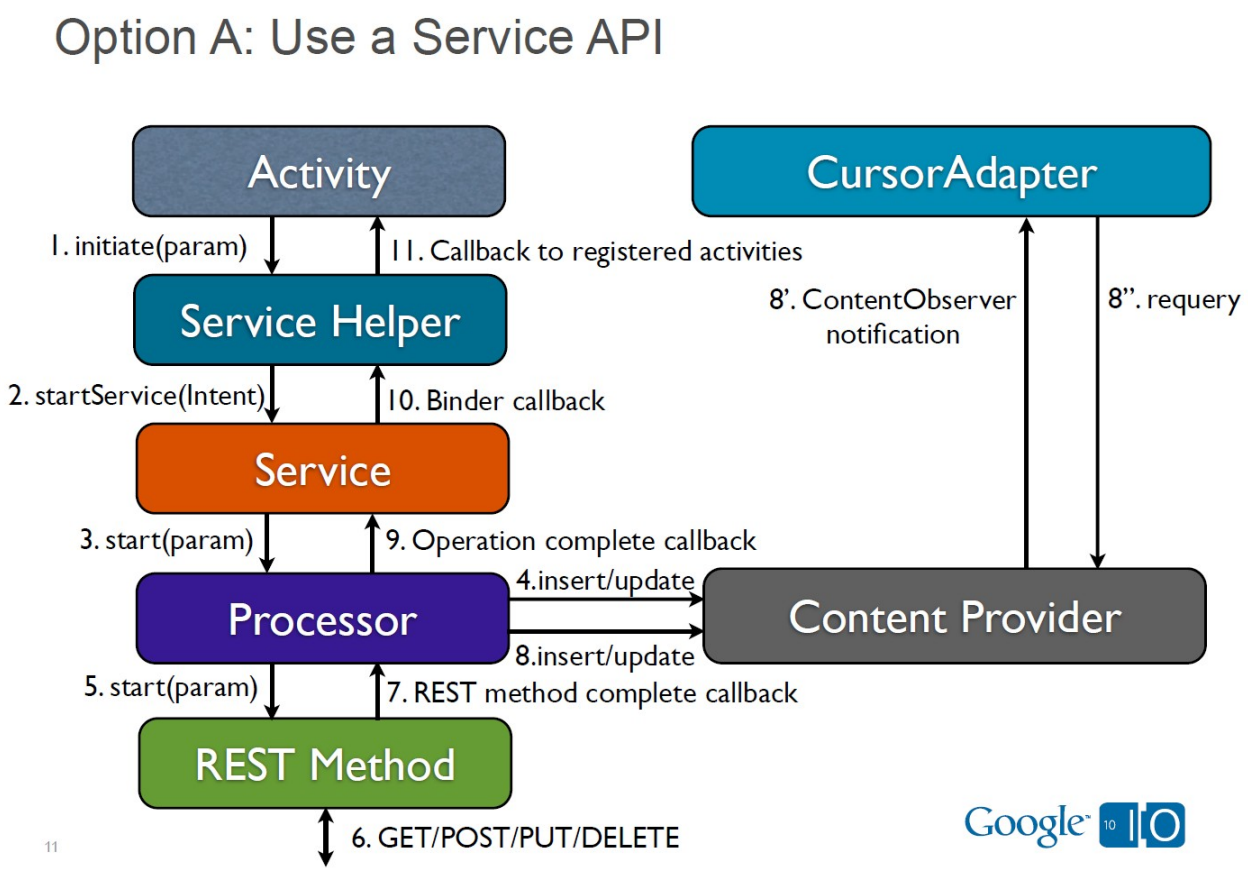
```
        {
            // TODO ...
            return 1;
        }

        public object GetDataByTerminalId(string terminalId)
        {
            // TODO ...
            return null;
        }
    }
}
```

Реализация REST-клиентов на Android

Основные проблемы связаны с тем, что время жизни Activity в Android совершенно непредсказуемо. В любой момент ваша Activity может быть приостановлена или даже удалена из памяти. Соответственно, какие-либо длительные операции в Activity делать нельзя, т.к. Activity запросто может "отвалиться" до завершения этих операций. В "лучшем" случае, будут потеряны результаты работы. В худшем – произойдет рассинхронизация данных на сервере и клиенте. Отсюда - необходимость все длительные операции запускать в сервисе. Сервис гораздо более живуч, чем Activity, его система без веских причин останавливать не будет.

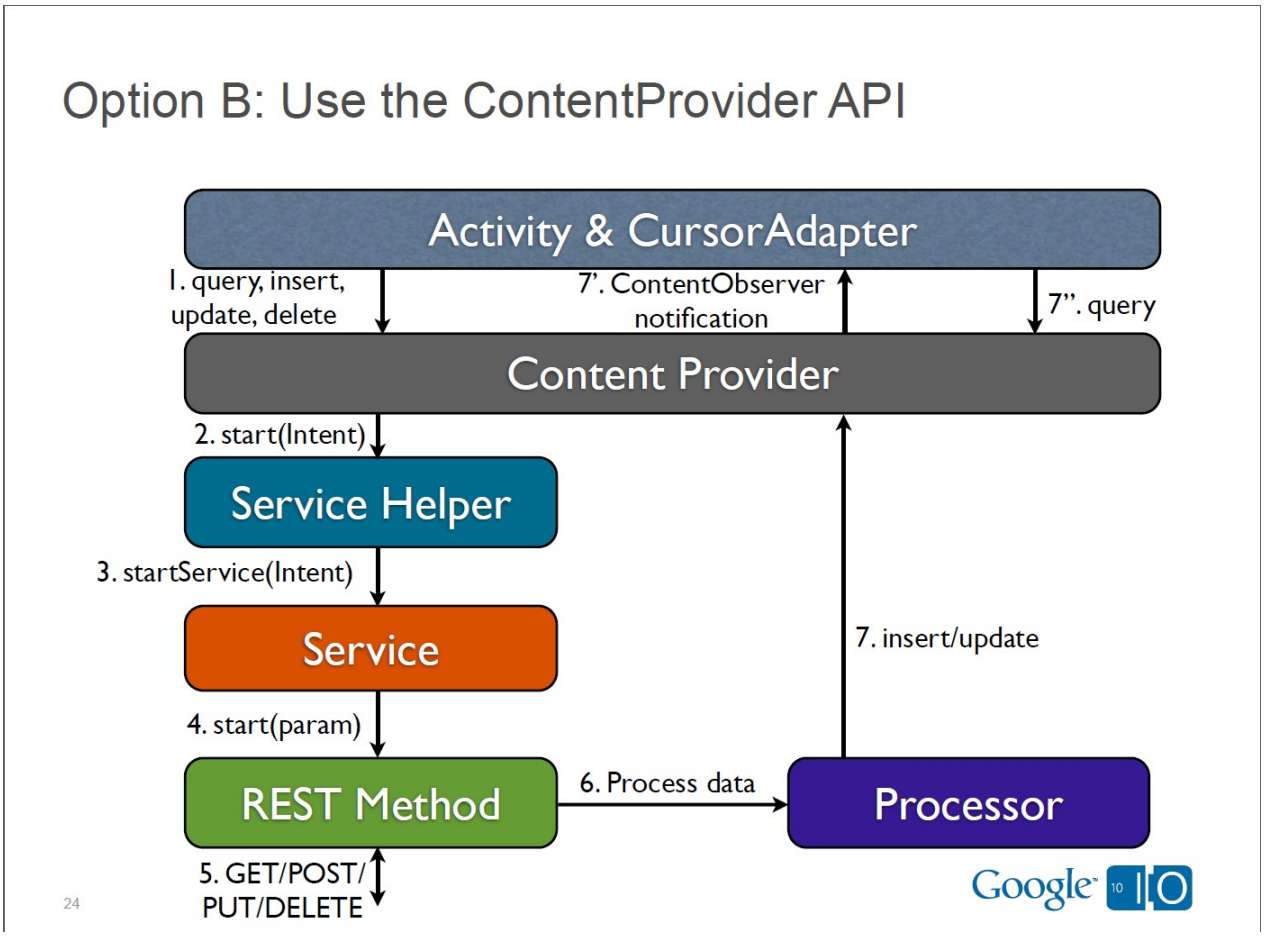
Pattern A

Используется Service API: Activity -> Service -> Content Provider. В данном варианте Activity работает с API Android Servcie. При необходимости послать REST-запрос Activity создает Service, Service асинхронно посылает запросы к REST-серверу и сохраняет результаты в Content Provider (sqlite). Activity получает уведомление о готовности данных и считывает результаты из Content Provider (sqlite).



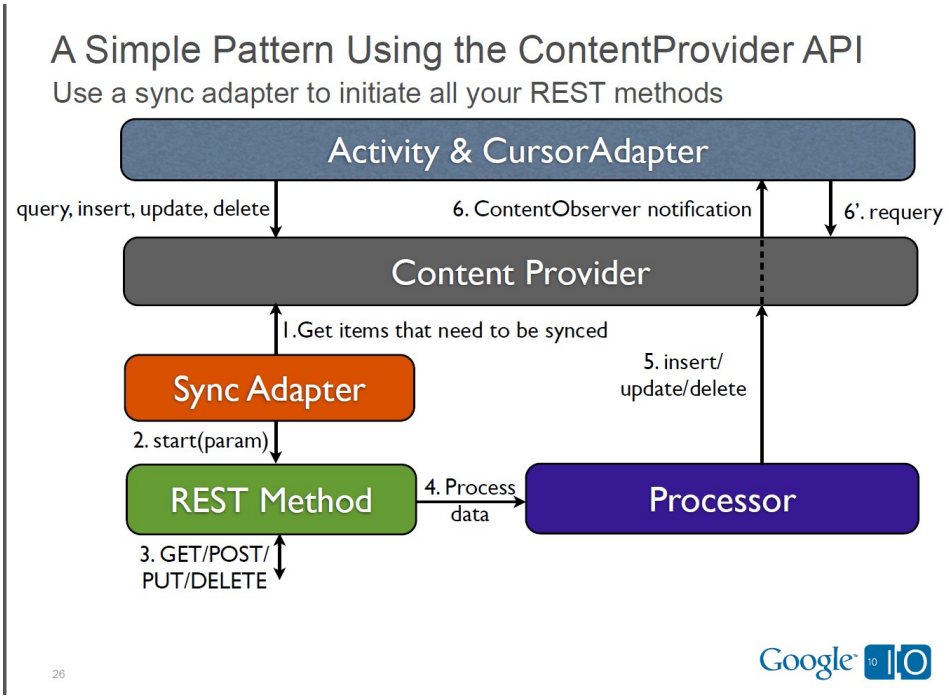
Pattern B

Используется ContentProvider API: Activity -> Content Provider -> Service. В этом случае Activity работает с API Content Provider, который выступает фасадом для сервиса. Данный подход основан на схожести Content Provider API и REST API: GET REST эквивалентен select-запросу к базе данных, POST REST эквивалентен insert, PUT REST ~ update, DELETE REST ~ delete. Результаты Activity так же загружает из sqlite.



Pattern C

Используется Content Provider API + SyncAdapter: Activity -> Content Provider -> Sync Adapter. Вариация подхода "B", в котором вместо сервиса используется собственный Sync Adapter. Activity дает команду Content Provider, который переадресовывает ее в Sync Adapter. Sync Adapter вызывается из Sync Manager, но не сразу, а в "удобный" для системы момент. Т.е. возможны задержки в исполнении команд.



Полезные ссылки:

http://square.github.io/retrofit/

http://loopj.com/android-async-http/

СПИСОК ЛИТЕРАТУРЫ

1. Пол Дейтел, Харви Дейтел, Александер Уолд. Android для разработчиков. = Android 6 for Programmers: An App-Driven Approach. — Спб.: «Питер», 2016. — 512 с. — ISBN 978-5-496-02371-9.
2. Андерс Ёранссон. Эффективное использование потоков в операционной системе Android. Технологии асинхронной обработки данных = Efficient Android Threading. — М.: «ДМК Пресс», 2015. — 304 с. — ISBN 978-5-97060-168-6.

ПРИЛОЖЕНИЕ А

Список всех uses-permission

Официальнаядокументация

```
<uses-permission android:name="android.permission.ACCESS_CHECKIN_PROPERTIES" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_SURFACE_FLINGER" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCOUNT_MANAGER" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
<uses-permission android:name="android.permission.BATTERY_STATS" />
<uses-permission android:name="android.permission.BIND_ACCESSIBILITY_SERVICE"/>
<uses-permission android:name="android.permission.BIND_APPWIDGET" />
<uses-permission android:name="android.permission.BIND_DEVICE_ADMIN" />
<uses-permission android:name="android.permission.BIND_INPUT_METHOD" />
<uses-permission android:name="android.permission.BIND_REMOTEVIEWS" />
<uses-permission android:name="android.permission.BIND_TEXT_SERVICE"/>
<uses-permission android:name="android.permission.BIND_VPN_SERVICE"/>
<uses-permission android:name="android.permission.BIND_WALLPAPER" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BRICK" />
<uses-permission android:name="android.permission.BROADCAST_PACKAGE_REMOVED" />
<uses-permission android:name="android.permission.BROADCAST_SMS" />
<uses-permission android:name="android.permission.BROADCAST_STICKY" />
<uses-permission android:name="android.permission.BROADCAST_WAP_PUSH" />
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.CALL_PRIVILEGED" />
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.CHANGE_COMPONENT_ENABLED_STATE" />
<uses-permission android:name="android.permission.CHANGE_CONFIGURATION" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.CLEAR_APP_CACHE" />
<uses-permission android:name="android.permission.CLEAR_APP_USER_DATA" />
<uses-permission android:name="android.permission.CONTROL_LOCATION_UPDATES" />
<uses-permission android:name="android.permission.DELETE_CACHE_FILES" />
<uses-permission android:name="android.permission.DELETE_PACKAGES" />
<uses-permission android:name="android.permission.DEVICE_POWER" />
<uses-permission android:name="android.permission.DIAGNOSTIC" />
<uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
<uses-permission android:name="android.permission.DUMP" />
<uses-permission android:name="android.permission.EXPAND_STATUS_BAR" />
<uses-permission android:name="android.permission.FACTORY_TEST" />
<uses-permission android:name="android.permission.FLASHLIGHT" />
<uses-permission android:name="android.permission.FORCE_BACK" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.GET_PACKAGE_SIZE" />
<uses-permission android:name="android.permission.GET_TASKS" />
<uses-permission android:name="android.permission.GLOBAL_SEARCH" />
<uses-permission android:name="android.permission.HARDWARE_TEST" />
<uses-permission android:name="android.permission.INJECT_EVENTS" />
<uses-permission android:name="android.permission.INSTALL_LOCATION_PROVIDER" />
<uses-permission android:name="android.permission.INSTALL_PACKAGES" />
<uses-permission android:name="android.permission.INTERNAL_SYSTEM_WINDOW" />
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES" />
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
<uses-permission android:name="android.permission.MANAGE_APP_TOKENS" />
<uses-permission android:name="android.permission.MASTER_CLEAR" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
```

```
<uses-permission android:name="android.permission.MODIFY_PHONE_STATE" />
<uses-permission android:name="android.permission.MOUNT_FORMAT_FILESYSTEMS" />
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
<uses-permission android:name="android.permission.NFC" />
<uses-permission android:name="android.permission.PERSISTENT_ACTIVITY"/>
<uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />
<uses-permission android:name="android.permission.READ_CALENDAR" />
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_FRAME_BUFFER" />
<uses-permission android:name="android.permission.READ_HISTORY_BOOKMARKS" />
<uses-permission android:name="android.permission.READ_INPUT_STATE" />
<uses-permission android:name="android.permission.READ_LOGS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.READ_PROFILE"/>
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.READ_SOCIAL_STREAM"/>
<uses-permission android:name="android.permission.READ_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.READ_SYNC_STATS" />
<uses-permission android:name="android.permission.READ_USER_DICTIONARY"/>
<uses-permission android:name="android.permission.REBOOT" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.RECEIVE_MMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.RECEIVE_WAP_PUSH" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.REORDER_TASKS" />
<uses-permission android:name="android.permission.RESTART_PACKAGES" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.SET_ACTIVITY_WATCHER" />
<uses-permission android:name="android.permission.SET_ALARM" />
<uses-permission android:name="android.permission.SET_ALWAYS_FINISH" />
<uses-permission android:name="android.permission.SET_ANIMATION_SCALE" />
<uses-permission android:name="android.permission.SET_DEBUG_APP" />
<uses-permission android:name="android.permission.SET_ORIENTATION" />
<uses-permission android:name="android.permission.SET_POINTER_SPEED" />
<uses-permission android:name="android.permission.SET_PREFERRED_APPLICATIONS"/>
<uses-permission android:name="android.permission.SET_PROCESS_LIMIT" />
<uses-permission android:name="android.permission.SET_TIME" />
<uses-permission android:name="android.permission.SET_TIME_ZONE" />
<uses-permission android:name="android.permission.SET_WALLPAPER" />
<uses-permission android:name="android.permission.SET_WALLPAPER_HINTS" />
<uses-permission android:name="android.permission.SIGNAL_PERSISTENT_PROCESSES" />
<uses-permission android:name="android.permission.STATUS_BAR" />
<uses-permission android:name="android.permission.SUBSCRIBED_FEEDS_READ" />
<uses-permission android:name="android.permission.SUBSCRIBED_FEEDS_WRITE" />
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<uses-permission android:name="android.permission.UPDATE_DEVICE_STATS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
<uses-permission android:name="android.permission.USE_SIP" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_APN_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_CALENDAR" />
<uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_GSERVICES" />
<uses-permission android:name="android.permission.WRITE_HISTORY_BOOKMARKS" />
<uses-permission android:name="android.permission.WRITE_PROFILE"/>
<uses-permission android:name="android.permission.WRITE_SECURE_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_SMS" />
<uses-permission android:name="android.permission.WRITE_SOCIAL_STREAM"/>
<uses-permission android:name="android.permission.WRITE_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_USER_DICTIONARY"/>
```

```
<uses-permission android:name="com.android.alarm.permission.SET_ALARM"/>
<uses-permission android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
<uses-permission android:name="com.android.browser.permission.WRITE_HISTORY_BOOKMARKS"/>
```