**Разработка мобильных приложений**

**Лабораторная работа №2**

**Сетевое взаимодействие**

В предыдущей работе вы создали Android-приложение с базовым интерфейсом и тестовыми данными. В данной работе вы подключите приложение к источнику реальных данных по погоде. При этом вы познакомитесь с сетевыми возможностями Android, разрешениями приложений и параллельным выполнением задач.

Для получения данных о погоде вы будете использовать сайт <http://openweathermap.org/API>. Сайт предоставляет бесплатные открытые данные о погоде. Перейдите на сайт и ознакомьтесь с тем, какие виды данных могут быть получены.

Некоторые примеры запросов к сайту на получение текущей погоды представлены на странице <http://openweathermap.org/current>. Для запросов используется простой протокол на базе HTTP, все параметры запроса задаются параметрами URL, поэтому вы можете тестировать запросы и наблюдать их результаты непосредственно в браузере. Форматом данных по умолчанию является JSON, но, используя дополнительный параметр, можно получить данные в XML.

Например, чтобы получить текущую погоду по Астрахани, используется следующий запрос:

<http://api.openweathermap.org/data/2.5/weather?q=Astrakhan>

Для получения тех же данных в формате XML можно использовать запрос:

<http://api.openweathermap.org/data/2.5/weather?q=Astrakhan&mode=xml>

Можно запросить данные на русском языке:

<http://api.openweathermap.org/data/2.5/weather?q=Astrakhan&lang=ru>

К сожалению, русификация сервиса неполная, поэтому рекомендуется использовать английский язык.

Для получения данных в метрической системе используйте параметр units=metric:

<http://api.openweathermap.org/data/2.5/weather?q=Astrakhan&units=metric>

Можно также посмотреть расшифровки всех кодов погодных явлений, использующихся в результатах запросов:

<http://openweathermap.org/weather-conditions>

Поэкспериментируйте с запросами к сервису, просмотрев погоду по различным городам и с различными параметрами.

Для разрабатываемого приложения потребуются данные по прогнозу со следующими параметрами:

* Период: 1 неделя
* Формат: JSON
* Местоположение: Астрахань
* Система: метрическая

Сформируйте соответствующий запрос и выполните его в браузере, просмотрите результаты

Вся последующая работа будет происходить в коде фрагмента PlaceholderFragment. Чтобы код класса MainActivity не разрастался, желательно вынести код фрагмента в отдельный файл. Кроме того, имя класса фрагмента не соответствует его функциональному назначению.

* Переименуйте PlaceholderFragment в ForecastFragment. Рекомендуется делать это, используя рефакторинг Rename.
* Перенесите класс ForecastFragment в отдельный файл, для этого рекомендуется использовать рефакторинг Move Type to New File.

Теперь необходимо реализовать загрузку сведений о погоде с сервера openweathermap.org. Поскольку сетевое взаимодействие не может выполняться в основном потоке приложения, нам потребуется сперва создать класс на основе AsyncTask для создания вспомогательного процесса.

* В классе ForecastFragment создайте класс FetchWeatherTask, наследующий от класса AsyncTask. В качестве всех трех типов параметров пока можно использовать Void, в дальнейшем пропишем реальные типы данных.

Более подробная информация об использовании AsyncTask:

<http://developer.android.com/intl/ru/training/articles/perf-anr.html>

<http://developer.android.com/reference/android/os/AsyncTask.html>

<http://developer.android.com/intl/ru/guide/components/processes-and-threads.html>

* В классе FetchWeatherTask создайте метод doInBackground(). В этом методе реализуйте код, выполняющий подключение к сервису погоды, используя сформированный вами ранее запрос, и получение данных с сервера (результатов запроса). Результат запроса (весь полученный json-документ) сохраните в виде строки и выведите в журнал в виде отладочной информации.

Более подробная информация об использовании HttpURLConnection:

<http://developer.android.com/intl/ru/training/basics/network-ops/connecting.html>

<http://developer.android.com/intl/ru/reference/java/net/HttpURLConnection.html>

* Добавьте код, реализующий запуск потока на базе FetchWeatherTask в метод onCreateView класса ForecastFragment, чтобы подключение к серверу и получение данных выполнялись автоматически при запуске приложения.
* Откройте файл конфигурации приложения AndroidManifest.xml и добавьте разрешение на доступ приложения в сеть Интернет.

Более подробная информация об использовании разрешений:

<http://developer.android.com/intl/ru/guide/topics/security/permissions.html#permissions>

<http://developer.android.com/intl/ru/guide/topics/manifest/uses-permission-element.html>

* Запустите приложение. Просмотрите журнал, вы должны увидеть результат выполнения запроса (json-строку) в журнале. Это означает, что получение данных с сервера выполнено успешно. Полученные с сервера данные пока не отображаются на экране, поскольку этот функционал вы еще не реализовали. Полученный json-документ необходимо разобрать на элементы и отобразить их в списке.

Предварительно добавим в приложение возможность обновлять данные не только при запуске, но и по запросу пользователя. Для этого добавим соответствующий пункт в меню приложения.

* Создайте новую компоновку меню с именем forecast\_fragment.xml
* В этом меню создайте элемент с ID action\_refresh и подписью Обновить. Не забывайте использовать строковые ресурсы для хранения всех текстовых строк.
* Подключите созданное меню к фрагменту ForecastFragment, для этого:
* В методе onCreateView() класса ForecastFragment добавьте вызов setHasOptionsMenu(true), чтобы указать системе, что у данного фрагмента имеется собственное меню.
* В классе ForecastFragment реализуйте метод onCreateOptionsMenu, в котором произведите генерацию меню на основе компоновки forecast\_fragment.xml
* В классе ForecastFragment реализуйте метод onOptionsItemSelected, обрабатывающий выбор пользователем одного из пунктов меню. В этом методе проверьте IT выбранного пользователем пункта (он должен совпадать с указанным вами при создании пункта меню Обновить ID action\_refresh) и, в случае совпадения, реализуйте создание и запуск потока на базе FetchWeatherTask.

Более подробная информация об использовании меню:

<http://developer.android.com/intl/ru/training/basics/actionbar/adding-buttons.html>

<http://developer.android.com/intl/ru/guide/topics/ui/menus.html>

В настоящее время выполняется запрос к серверу по фиксированному городу. Необходимо сделать так, чтобы пользователь имел возможность указывать город, по которому показывать прогноз погоды. Первый шаг к этому – модифицировать построение запроса к серверу таким образом, чтобы использовалась не фиксированная строка запроса, а собираемая из параметров. Для этого вы используете специальный класс Uri.Builder, имеющий возможность конструировать запросы.

Пример использования класса Uri.Builder для построения строки URL вида <http://api.openweathermap.org/data/2.5/weather?q=Astrakhan&mode=xml>:

Uri.Builder builder = Uri.parse("http://api.openweathermap.org/data/2.5/weather").buildUpon();

String urlString = builder.

appendQueryParameter("q", "Astrakhan").

appendQueryParameter("mode", "xml").

build().toString();

Здесь сперва на основе постоянной части адреса создается объект класса Uri.Builder, который затем используется для добавления параметров q и mode.

Более подробная информация об использовании Uri.Builder:

<http://developer.android.com/intl/ru/reference/android/net/Uri.Builder.html>

Измените FetchWeatherTask таким образом, чтобы он при запуске принимал название города, для которого необходимо выполнить запрос. Для этого:

* Создайте в классе FetchWeatherTask константы для базового URL и для наименований параметров.
* Измените тип первого из параметров AsyncTask с Void на String. Соответственно измените на String параметры метода doInBackground()
* Используйте Uri.Builder для построения URL запроса к серверу. Название города будет передаваться первым параметром метода doInBackground(), получить его можно будет как params[0]. Для названий параметров запроса используйте ранее созданные константы, значения можно подставлять непосредственно.
* Модифицируйте запуск процесса FetchWeatherTask - вызов метода execute(), передав в качестве параметра значение “Astrakhan”. В дальнейшем значение этого параметра будем брать из настроек приложения.
* Для проверки выведите сконструированную строку запроса в журнал.

Следующий шаг – выбрать из полученного json-документа данные, которые необходимо отобразить пользователю. Напомним, что наша цель – показать пользователю набор строк вида:

Пятница - Солнечно - 25/18

В строке отображается день недели, общее состояние погоды в этот день, а также минимальная и максимальная температура в этот день.

Рассмотрим подробнее ответ сервера:

{    
   **"cod"**:"200",  
   **"message"**:0.0045,  
   **"city"**:{    
      **"id"**:580497,  
      **"name"**:"Astrakhan",  
      **"coord"**:{    
         **"lon"**:48.04076,  
         **"lat"**:46.349682  
      },  
      **"country"**:"RU",  
      **"population"**:0,  
      **"sys"**:{    
         **"population"**:0  
      }  
   },  
   **"cnt"**:5,  
   **"list"**:[    
      {    
         **"dt"**:1411718400,  
         **"temp"**:{    
            **"day"**:15.04,  
            **"min"**:10.7,  
            **"max"**:16.13,  
            **"night"**:11.09,  
            **"eve"**:14.07,  
            **"morn"**:13  
         },  
         **"pressure"**:1028.56,  
         **"humidity"**:98,  
         **"weather"**:[    
            {    
               **"id"**:500,  
               **"main"**:"Rain",  
               **"description"**:"light rain",  
               **"icon"**:"10d"  
            }  
         ],  
         **"speed"**:3.81,  
         **"deg"**:312,  
         **"clouds"**:76,  
         **"rain"**:2.5  
      },

Json-документ представляет собой структуру пар вида «имя: значение», в которой значениями могут быть:

- непосредственные значения;

- вложенные массивы (заключаются в []);

- вложенные json-документы (заключаются в {}).

В данном документе мы видим, что в документ входят элементы “cod”, “message”, “cnt”, вложенный json-документ "city", вложенный массив “list”. Каждый элемент массива “list” является json-документом, содержащим данные по прогнозу на определенный день. Документ прогноза на день содержит, в частности:

- значение “dt” – дату в формате Unix Timestamp (количество секунд, прошедших с 1 января 1970);

- вложенный json-документ со сведениями о температуре (минимум, максимум, температура в разное время суток – днем, ночью и т.д.);

- значение “pressure” – атмосферное давление;

- значение “humidity” – влажность;

- вложенный массив “weather”, содержащий единственный элемент со сведениями о погоде: “id” – идентификатор типа погоды, “main” – общий тип погоды, “description” – описание погоды, “icon” – имя файла с символом погоды.

Для обработки json-документов в Android предусмотрены классы JSONObject и JSONArray. Объект класса JSONObject представляет собой json-документ, из которого можно получать отдельные элементы либо вложенные документы и массивы. Он может создаваться на основе строки с json. Основные операции:

getString(“element”) – получить строковое значение элемента “element”

getInteger(“element”) – получить числовое значение элемента “element”

getDouble(“element”) – получить числовое значение элемента “element”

getJSONObject(“element”) – получить вложенный json-документ с именем “element”

getJSONArray(“element”) – получить вложенный массив с именем “element”

Например, для приведенного документа вызов obj.getInteger(“cod”) вернет значение 200, а вызов obj.getJSONObject(“city”) – новый JSONObject с информацией о городе.

Объект класса JSONArray представляет собой массив, в который могут входить значения, вложенный массивы или вложенные json-документы. Для доступа к элементам массива используются методы, аналогичные методам JSONObject, только вместо имени элемента параметром метода задается порядковый номер элемента в массиве, например, вызов getInteger(3) вернет целое число на позиции 3 в массиве, а вызов getJSONObject(5) – вложенный JSONObject на позиции 5 в массиве.

Для отображения в строке потребуются значения элементов из массива “list”: “dt” – дата (переведенная в читаемый формат), “min” и “max” из объекта “temp” – минимум и максимум температуры, “main” из массива “weather” – описание погоды.

* Создайте в классе FetchWeatherTask метод вида

String[] parseWeatherJson(String weatherJson, int numDays),

возвращающий массив строк вида «дата – описание – мин/макс», разбирая заданный json-документ (параметр weatherJson) для указанного количества дней (параметр numDays). В этом методе, используя комбинацию JSONObject и JSONArray, реализуйте извлечение указанных значений из json-документа.

Для преобразования даты в читаемый формат сперва умножьте ее на 1000, чтобы привести к стандарту задания даты в Java, после чего используйте нижеприведенный код:

Date date = new Date(time \* 1000);

SimpleDateFormat format = new SimpleDateFormat("E, dd-MM");

String result = format.format(date).toString();

* Перед возвратом полученного массива строк предварительно выведите каждую строку в журнал.
* Измените определение класса FetchWeatherTask, заменив третий параметр AsyncTask (тип результата выполнения потока) на массив строк String[]. Аналогично измените на String[] возвращаемый тип метода doInBackground(). В методе doInBackground() отправьте считанную с сервера строку с json-документом о прогнозе на обработку в метод parseWeatherJson(), результат этого метода – массив строк – верните как результат работы метода doInBackground().
* Запустите программу, получите данные с сервера и проверьте в журнале корректность формируемых строк.

Последний шаг – отображение полученного списка строк в ListView. Для этого:

* Определите метод onPostExecute() в классе FetchWeatherTask, кторый будет принимать параметром массив строк, полученный в результате выполнения метода doInBackground(). Получите адаптер, используемый для отображения данных в ListView (для этого вам придется изменить его с локальной переменной метода onCreateView() на поле класса ForecastFragment), очистите его и поместите в него все строки из массива.
* Запустите приложение, вы должны увидеть реальный прогноз погоды, полученный через Интернет. В случае успеха удалите из исходного текста все выводы отладочной информации в журнал.