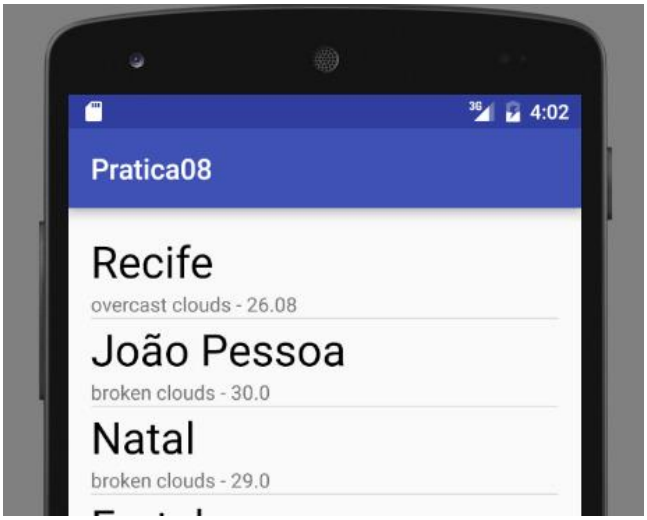
INSTITUTO FED. DE EDUCAÇÃO, CIÊNC. E TEC. DE PERNAMBUCO
CURSO: TEC. EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS
PROFESSOR: RAMIDE DANTAS
ASSUNTO: ACESSO À REDE COM VOLLEY

Aula Prática 08

Parte 1: Preparação

Passo 1: Esta prática é baseada na prática 3 (ListView), sendo usado um ListView (ou RecyclerView) como mostrado na figura abaixo (já com as modificações da prática).



Passo 2: Adicione a permissão de acesso a Internet à aplicação. Siga os passos descritos na prática 7 (Acesso à Rede) caso precise.

Passo 3: Para poder usar a API Volley, modifique o script gradle do app adicionando a dependência como descrito a seguir:

```
...
dependencies {
    ...
    compile 'com.android.volley:volley:1.0.0'
}
```

Parte 2: Refatorando

Passo 1: Modifique a classe City usada na prática:

- A propriedade “info” passa a se chamar “weather”;
- Adicione um método `setWeather()`, para setar essa propriedade;
- O construtor da classe deve receber agora só o valor de “name”.

Passo 2: Em MainActivity, faça com o as cidades criadas no array `cities` recebam apenas o nome.

Passo 3: Em MainActivity, modifique o método `onCreate()` como a seguir:

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

    this.queue = Volley.newRequestQueue(this);

    ListView listView = (ListView)findViewById(R.id.list_view);
    listView.setAdapter(new CityArrayListAdapter(this, queue,
                R.layout.city_listitem, cities
        )
    );

    ...

}
```

Nesse método está sendo criado o `RequestQueue` (fila de requisições), um dos componentes principais da API Volley. É ele que gerencia o envio e recebimento de requisições de rede, permitindo inclusive cancelar requisições. Normalmente só há um objeto por aplicação, devendo ficar de preferência num Singleton.

Parte 3: Carregando da rede com Volley

Passo 1: Na classe `CityArrayListAdapter`, faça as seguintes modificações:

- Adicione a fila de requisições (propriedade `RequestQueue queue`;)
- Modifique o construtor para inicializar essa propriedade com uma fila recebida via parâmetro.

Passo 2: Ainda em `CityArrayListAdapter`, modifique o método `getView()` como descrito a seguir:

```
@Override
public View getView(int position, View view, ViewGroup parent) {
    //Trecho que cria o View ou pega o ViewHolder
    ...

    holder.cityName.setText(cities[position].getName());

    if (cities[position].getWeather() != null) {
        holder.cityInfo.setText(cities[position].getWeather());
    } else {
        final City city = cities[position];
        final TextView weather = holder.cityInfo;
        weather.setText("Loading weather...");
        loadInBackground(weather, city);
    }

    return listItem;

}
```

Esse trecho verifica se as condições climáticas atuais (weather) foram setadas, caso contrário dispara uma carga em segundo plano usando o método privada `loadInBackground()`, descrito a seguir.

Passo 3: Crie o método `loadInBackground()` na classe `CityArrayListAdapter`, como a seguir:

```
private void loadInBackground(final TextView weatherView, final City city) {

    Uri.Builder builder = new Uri.Builder();
    builder.scheme("http");
    builder.authority("api.openweathermap.org");
    builder.appendPath("data/2.5/weather");
    builder.appendQueryParameter("q", city.getName());
    builder.appendQueryParameter("mode", "json");
    builder.appendQueryParameter("units","metric");
    builder.appendQueryParameter("cnt","" + 1);
    builder.appendQueryParameter("APPID", "SUA CHAVE AQUI");

    JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET,
                                    builder.build().toString(), null,
            new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    String weatherStr = getWeather(response);

                    if (weatherStr != null) {
                        city.setWeather(weatherStr);
                        weatherView.setText(weatherStr);
                    } else {
                        weatherView.setText("Erro!");
                    }
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            weatherView.setText("Erro!");
        }
    });

    queue.add(request);
}
```

Esse método constrói uma URL para a requisição HTTP que será usada para carregar as condições climáticas atuais. Assim como na prática 7, use a chave que você criou no site do OpenWeatherMap para o campo APPID.

Em seguida é criada uma requisição do tipo `JsonObjectRequest`, que faz parte da API Volley (outros tipos incluem `ImageRequest` e `StringRequest`). Ao criar essa requisição, associamos um listener que é invocado quando a requisição termina (com sucesso ou falha). Esse listener dá o tratamento adequado, no nosso caso, invocando o parser JSON e atualizando o correspondente. Em caso de falha, uma mensagem de erro é exibida no lugar onde apareceria a previsão do tempo.

Passo 4: Cancele a requisições que estiverem pendentes quando a atividade for parada, para evitar problemas de acesso indevido a UI. Em `MainActivity`, adicione o tratador `onStop()`:

```
@Override
public void onStop() {
    super.onStop();
    queue.cancelAll(this);
}
```

Passo 5: Crie o método `getWeather()` listado abaixo em `CityArrayListAdapter`:

```
    private String getWeather(JSONObject forecastJson) {
        final String OWM_WEATHER = "weather";
        final String OWM_TEMPERATURE = "temp";
        final String OWM_MAIN = "main";
        final String OWM_DESCRIPTION = "description";

        try {
            JSONObject weatherObject = forecastJson.getJSONArray(OWM_WEATHER).
                                                              getJSONObject(0);
            JSONObject mainObject = forecastJson.getJSONObject(OWM_MAIN);

            String description = weatherObject.getString(OWM_DESCRIPTION);
            double temp = mainObject.getDouble(OWM_TEMPERATURE);

            return description + " - " + temp;
        } catch (JSONException e) {
            e.printStackTrace();
            return null;
        }
    }
```

Esse método é uma versão simplificada do parser JSON usado na prática 7 para interpretar a resposta do servidor. Ele cria uma string simples contendo a descrição das condições climáticas atuais seguida da temperatura.

Parte 4: Desafio – Carregando imagens em segundo plano.

Passo 1: Modifique o layout dos itens da lista para incluir um `ImageView` na parte direita do layout. Dê um identificador adequado a esse `ImageView`.

Passo 2: Modifique o `ViewHolder` em `CityArrayListAdapter` para conter uma referência para esse `ImageView`. Modifique `getView()` para setar essa referência de acordo.

Passo 3: Procure por imagens na Internet (ícones) para alguns tipos de condições climáticas (ensolarado, chovendo, etc.). Pegue as URLs dessas imagens.

Passo 4: Modifique `loadInBackground()` de forma que ao obter o retorno do serviço, uma imagem adequada (das selecionadas no passo 3) seja carregada em segundo plano. Por exemplo, se a descrição contiver a string “cloud”, a imagem de uma nuvem é carregada no `ImageView`, como descrito a seguir.

- Utilize um `ImageLoader`: crie no construtor de `CityArrayListAdapter` passando o RequestQueue e uma classe de cache, como abaixo:

```
loader = new ImageLoader(queue, new ImageLoader.ImageCache() {
      private final LruCache<String, Bitmap> mCache = new LruCache<String, Bitmap>(10);
      public void putBitmap(String url, Bitmap bitmap) { mCache.put(url, bitmap); }
      public Bitmap getBitmap(String url) { return mCache.get(url); }
});
```

- Para carregar as imagens, use o `ImageLoader` como no código abaixo:

```
loader.get(url, ImageLoader.getImageListener(holder.cityImg,
                android.R.drawable.btn_star, android.R.drawable.ic_dialog_alert));
```

O trecho acima ficaria em `loadInBackground()`. A API Volley disponibiliza outras duas formas de carregar imagens: `ImageRequest` e `NetworkImageView`. ImageRequest lhe dá mais controle sobre o que fazer com a imagem retornada. Já NetworkImageView pode substituir por completo o componente ImageView: basta setar a URL que ele carrega a imagem em segundo plano.