CV 2

ListView, Adapter

https://github.com/codepath/android_guides/wiki/Using-a-BaseAdapter-with-ListView

- ListView
- BaseAdapter
- model
- ViewHolder
- RecyclerView - lepší, rychlejší, ale složitější

Networking

- obrázky - Picasso & Fresco
- JSON (GSON)
- HttpURLConnection
- OkHttpClient
- Retrofit 2

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

Obrázky - Picasso

https://github.com/square/picasso

```
implementation ‘com.squareup.picasso:picasso:2.5.2'
```

```
Picasso.with(context)
  .load(“https://…”)
  .into(imageView)
```

Zadání

- Připravte si Picasso

- V ListView zobrazte i obrázek ke jménu

Obrázky - Fresco

http://frescolib.org/

- Facebook

- ashmem vs. OutOfMemory

- ProgressiveJPEG

```
<com.facebook.drawee.view.SimpleDraweeView
    …
  />

Uri.parse(“…”)

view: SimpleDraweeView

view.setImageURI(uri)
```

JSON - GSON

https://github.com/google/gson

- GSON je knihovna pro zpracování JSON do objektu a naopak

JSON - GSON

```
implementation 'com.google.code.gson:gson:2.8.2'
```

```
val gson = Gson()
val user = User()
val json: String = gson.toJson(user)
val user2: User = gson.fromJson(json, User::class.java)
```

Zadání

- Vytvořte třídu Student, která bude mít 2 Stringy - name a surname.
- Inicializujte s hodnotami a převeďte do JSONu pomocí GSONu
- Výsledek zobrazte jako Toast

```
Toast.makeText(this, "ahoj", Toast.LENGTH_LONG).show()
```

Řešení

```
data class Student(val name: String, val surname:
String)

val s = Student(“Radim”, “Vaculik”)
val json = Gson().toJson(s)

Toast.makeText(this, json, Toast.LENGTH_LONG).show()
```

HttpUrlConnection

https://developer.android.com/reference/java/net/HttpURLConnection.html

```
    URL url = new URL(“http:/…/“);
    HttpURLConnection urlConnection =
(HttpURLConnection) url.openConnection();
    try {
      InputStream in = new
BufferedInputStream(urlConnection.getInputStream());
      readStream(in);
    } finally {
      urlConnection.disconnect();
    }
```

HttpUrlConnection

https://[illegible]per.android.com/reference/java/net/HttpURLConnection.html

```
   URL url = new URL("…/");
   HttpURLConnection urlConnection =
(HttpURLConnection) url.openConnection();
   try {
     InputStream in
BufferedInputStream(urlConnection.getInputStream());
     readStream(in);
   } finally {
     urlConnection.disconnect();
   }
```

NOPE

OkHttpClient - GET

http://square.github.io/okhttp/

```
val client = OkHttpClient()

val request = new Request.Builder()
        .url(url)
        .build()

val response = client
        .newCall(request)
        .execute()

val body = response.body().string()
```

OkHttpClient - POST

http://square.github.io/okhttp/

```
val client = OkHttpClient()

val request = new Request.Builder()
      .url(url)
      .post(RequestBody.create(.., json))
      .build()

val response = client
      .newCall(request)
      .execute()

val body = response.body().string();
```

Retrofit 2

https://square.github.io/retrofit/

- knihovna pro HTTP requests
- Request, Response, Call
- Synchronní a asynchronní

Retrofit 2

https://square.github.io/retrofit/

```
interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(@Path("user") String user)
}
```

```
val retrofit = Retrofit.Builder()
    .baseUrl("https://api.github.com/")
    .build()

val service = retrofit.create(GitHubService.class)


val repos: Call<List<Repo>> = service.listRepos("octocat");
```

Retrofit 2

https://square.github.io/retrofit/

```
val repos: Call<List<Repo>> = service.listRepos("octocat");
```

Synchronní (nechceš používat)

```
repos.execute()
```

Asynchronní (chceš používat)

```
repos.enqueue(object : Callback<User> {
    override fun onResponse(call: Call<User>?, response: Response<User>?) {
        val user = response?.body?
    }

    override fun onFailure(call: Call<User>?, t: Throwable?) {
        // Error handling
    }
}
```

Zadání

https://developer.github.com/v3/users/

- Aplikace, která zobrazí fotku uživatele z Githubu

- https://api.github.com/users/octocat

- avatar_url