



MOBILNE APLIKACIJE

Vežbe 4

GUI II

2022/2023

Sadržaj

1. Adapteri	3
1.1 Pravljenje adaptera	3
2.1 Pravljenje toolbar-a	6
3. Toasts / Snackbars	10
4. Navigation Drawer	12
5. Spinner	15
6. Dijalozi	16
7. Domaći	17

1. Adapteri

Adapteri povezuju poglede i izvore podataka (podaci iz baze, sa interneta..).

Moguće je koristiti neke od predefinisanih adaptera (*BaseAdapter*, *ArrayAdapter*, *CursorAdapter*) ili napraviti *Custom Adapters*, adaptere koji povezuju proizvoljan pogled i izvor podataka.

ArrayAdapter povezuju *TextView* pogled (ili pogled koji sadrži *TextView* pogled) i niz ili kolekciju. Automatski se poziva *toString()* metoda svakog objekta u nizu ili kolekciji i njena povratna vrednost se prikazuje u pogledu.

1.1 Pravljenje adaptera

Kreiramo novu klasu, *CinemaAdapter*, koja nasleđuje neki od postojećih adaptera. U našem primeru nasleđuje *BaseAdapter* (slika 1), koji je sposoban da kao izvor podataka iskoristi listu ili niz. Dobijamo metode koje moramo da redefinišemo, da bi naš adapter ispravno radio.

```
public class CinemaAdapter extends BaseAdapter{
    private Activity activity;

    public CinemaAdapter(Activity activity) { this.activity = activity; }
```

Slika 1. Kreiranje adaptera

Metoda *getCount()* vraća ukupan broj elemenata u listi, koja treba da se prikaže.

```
/*...*/
@Override
public int getCount() {
    return Mokap.getCinemas().size();
}
```

Slika 2. Metoda *getCount()*

Metoda *getItem(int position)* vraća pojedinačan element na osnovu njegove pozicije.

```
/*...*/
@Override
public Object getItem(int position) {
    return Mokap.getCinemas().get(position);
}
```

Slika 3. Metoda *getItem(int position)*

Metoda *getItemId(int position)* vraća jedinstveni identifikator.

```

/*...*/
@Override
public long getItemId(int position) {
    return position;
}

```

Slika 4. Metoda *getItemId(int position)*

Metoda *getView* popunjava *ListView* sa podacima, tako što adapter, koji čuva listu elemenata, iterira kroz elemente i redom popunjava *ListView*. Adapter zna koliko iteracija treba da ima jer poseduje metodu *getCount()*.

U metodi *getView* *position* je pozicija elementa u listi elemenata, koju čuva adapter, a *parent* je roditelj na kog će *view* biti postavljen. U prvom *if-u* vršimo inicijalizaciju *convertView*-a na kreirani layout *cinema_list*, koji će prikazivati sve elemente.

```

/*...*/
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    View vi=convertView;
    Cinema cinema = Mokap.getCinemas().get(position);

    if(convertView==null)
        vi = activity.getLayoutInflater().inflate(R.layout.cinema_list, root: null);

    TextView name = (TextView)vi.findViewById(R.id.name);
    TextView description = (TextView)vi.findViewById(R.id.description);
    ImageView image = (ImageView)vi.findViewById(R.id.item_icon);

    name.setText(cinema.getName());
    description.setText(cinema.getDescription());

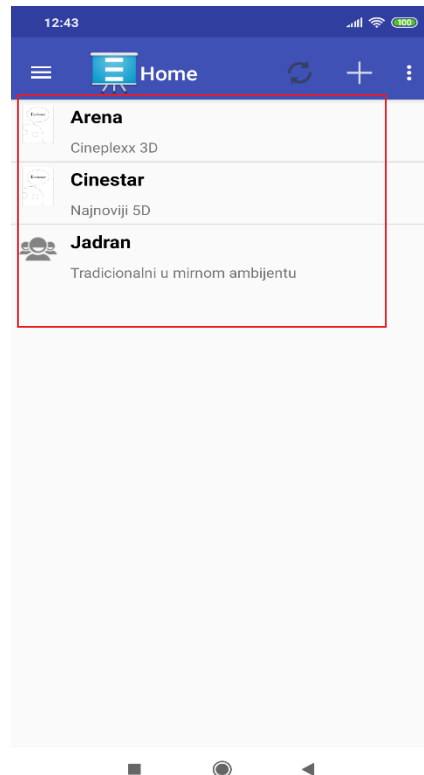
    if (cinema.getAvatar() != -1){
        image.setImageResource(cinema.getAvatar());
    }

    return vi;
}

```

Slika 5. Metoda *getView*

Kao rezultat dobijamo prikazanu listu svih filmova (slika 6).



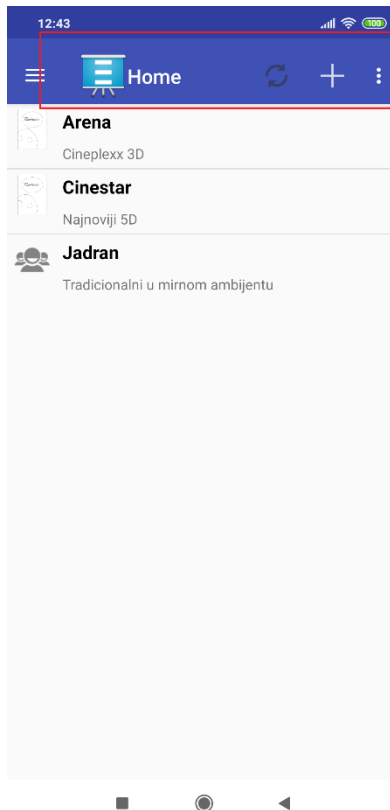
Slika 6. Lista filmova

2. Toolbar

Toolbar je element GUI-a koji se uglavnom nalazi na vrhu ekrana i obezbeđuje navigaciju, izvršavanje akcija, promenu pogleda, *branding* aplikacije.

2.1 Pravljenje toolbar-a

Na slici 7 se nalazi primer *toolbar*-a koji ćemo kreirati.



Slika 7. Primer *toolbar*-a

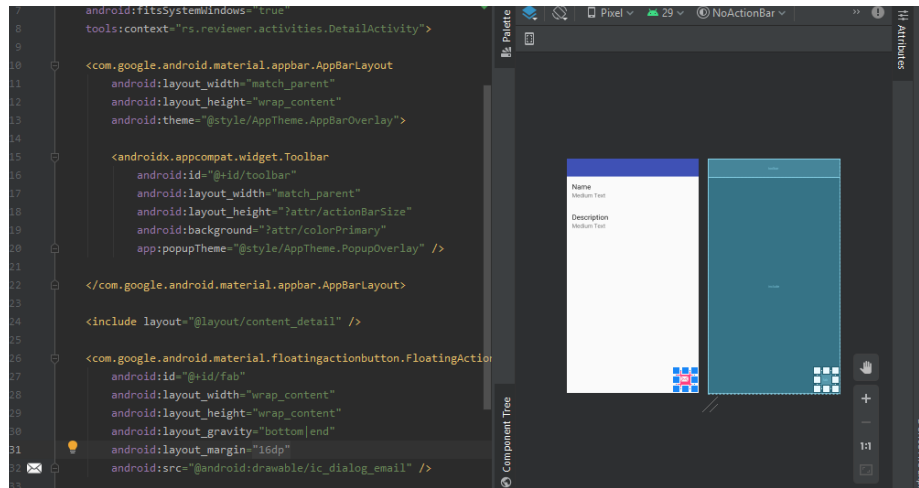
Kreiramo *layout* i u njemu definišemo element `<androidx.appcompat.widget.Toolbar>` (slika 8).

Toolbar se nalazi unutar *AppBarLayout*-a, tj. unutar elementa:

`<com.google.android.material.appbar.AppBarLayout>`.

AppBarLayout je vertikalni *LinearLayout* koji obezbeđuje mnoga svojstva *Material design*-a. *Material design* je skup principa za vizuelni dizajn, dizajn pokreta i dizajn interakcija. Sve aplikacije koje su dizajnirane po ovim principima pružaju korisnicima konzistentno iskustvo i obezbeđuju da korišćenje aplikacija bude intuitivno. *AppBarLayout layout* se koristi kao dete *CoordinatorLayout*-a.

Toolbar možemo ručno da definišemo ili u *Design* režimu da ga prevučemo iz palete (Containers > Toolbar).



Slika 8. Kreiranje *Toolbar*-a

Sledeći korak je da kreiramo klasu, koja nasleđuje *AppCompatActivity*. U metodi *onCreate* dobivamo *toolbar* (linija 17) i pozivamo metodu *setSupportActionBar()* kojoj prosleđujemo dobavljeni *toolbar* (slika 9).

```

10
11 public class DetailActivity extends AppCompatActivity {
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_detail);
17         Toolbar toolbar = findViewById(R.id.toolbar);
18         setSupportActionBar(toolbar);
19
20         TextView tvName = findViewById(R.id.tvName);
21         TextView tvDescr = findViewById(R.id.tvDescr);
22
23         tvName.setText(getIntent().getStringExtra( name: "name"));
24         tvDescr.setText(getIntent().getStringExtra( name: "descr"));
25     }
26
27 }
28

```

Slika 9. Postavljenje *Toolbar*-a

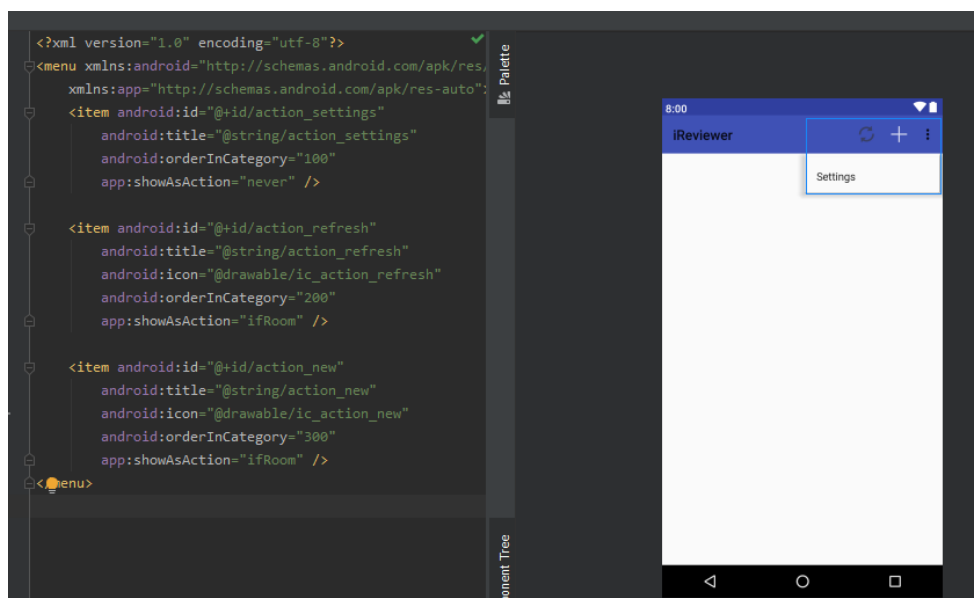
Postavljanje ikonica na toolbar i povezivanje sa akcijama

Da bismo mogli da izvršavamo akcije klikom na određene dugmiće iz *toolbar*-a, prvo kreiramo meni kao resurs. U ovom primeru kreirali smo meni sa 3 stavke: *settings*, *new* i *refresh* (slika 10) i to će biti ikonice koje će se prikazati na *toolbar*-u. Za svaku stavku smo definisali atribut *app:showAsAction*. Ovaj atribut govori kad i kako će stavka menija da bude prikazana.

Vrednosti koje ovaj atribut može da ima su:

- *ifRoom*
- *withText*
- *never*
- *always*
- *collapseActionView*

Ako atribut postavimo na *ifRoom* to znači da će ta stavka menija biti prikazana samo ako postoji dovoljno prostora u meniju. U suprotnom ova stavka će biti smeštena u padajući meni, kao što smo za stavku *settings* eksplicitno uradili sa postavljanjem atributa na vrednost *never*.



Slika 10. Menu stavke

Da bismo kreirali meni postavili na *toolbar*, redefinišemo metodu *onCreateOptionsMenu* (slika 11).

```
@Override
public void onCreateOptionsMenu(@NonNull Menu menu, @NonNull MenuInflater inflater) {
    super.onCreateOptionsMenu(menu, inflater);

    // ovo korostimo ako je nasa arhitektura takva da imamo jednu aktivnost
    // i vise fragmentaa gde svaki od njih ima svoj menu unutar toolbar-a
    menu.clear();
    inflater.inflate(R.menu.activity_itemdetail, menu);
}
```

Slika 11. Metoda *onCreateOptionsMenu*

Za povezivanje dugmića iz *toolbar*-a sa akcijama, koristimo metodu *onOptionsItemSelected* (slika 12). Ova metoda vraća *MenuItem* na koji je korisnik kliknuo. Taj *MenuItem* u sebi sadrži i identifikator, uz pomoć god znamo tačno na koje dugme iz *toolbar*-a smo kliknuli.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();


    if(id == R.id.action_refresh){
        Toast.makeText(getActivity(), text: "Refresh App", Toast.LENGTH_SHORT).show();
    }
    if(id == R.id.action_new){
        Toast.makeText(getActivity(), text: "Create Text", Toast.LENGTH_SHORT).show();
    }
    return super.onOptionsItemSelected(item);
}
```

Slika 12. Metoda *onOptionsItemSelected*

3. Toasts / Snackbars

Toast je *pop-up* poruka koja automatski nestaje posle određenog vremena. Ovakve poruke korisniku daju povratne informacije da je neka akcija izvršena ili eventualno da je došlo do neke greške.

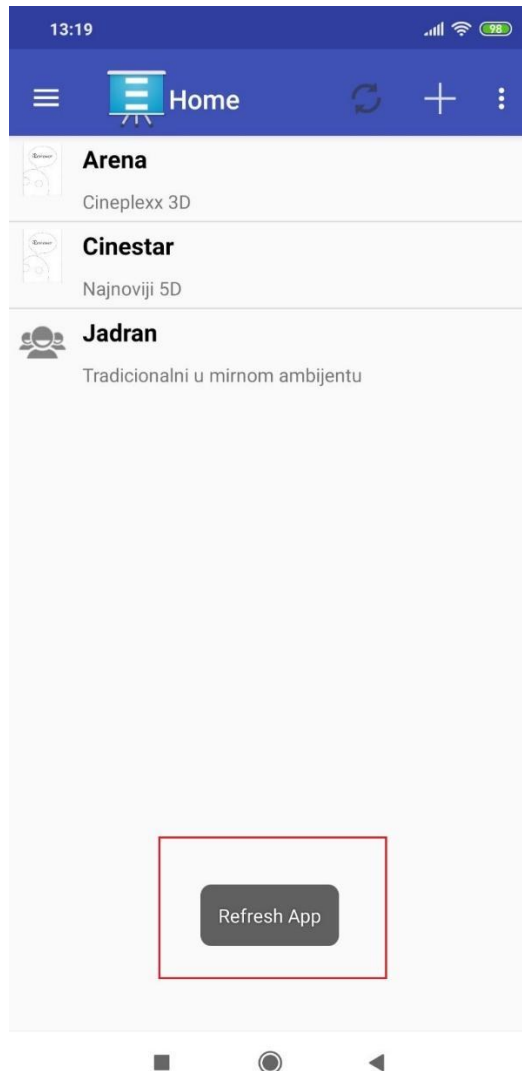
Na slici se nalazi metoda koja dodeljuje akcije dugmičima iz menija (slika 13). Kada korisnik klikne na dugme *refresh* u tom trenutku će se prikazati *toast* sa porukom „Refresh App“ (slika 14), a kada klikne na dugme *new* prikazaće se poruka „Create Text“.



```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    if(id == R.id.action_refresh){
        Toast.makeText(getActivity(), text: "Refresh App", Toast.LENGTH_SHORT).show();
    }
    if(id == R.id.action_new){
        Toast.makeText(getActivity(), text: "Create Text", Toast.LENGTH_SHORT).show();
    }
    return super.onOptionsItemSelected(item);
}
```

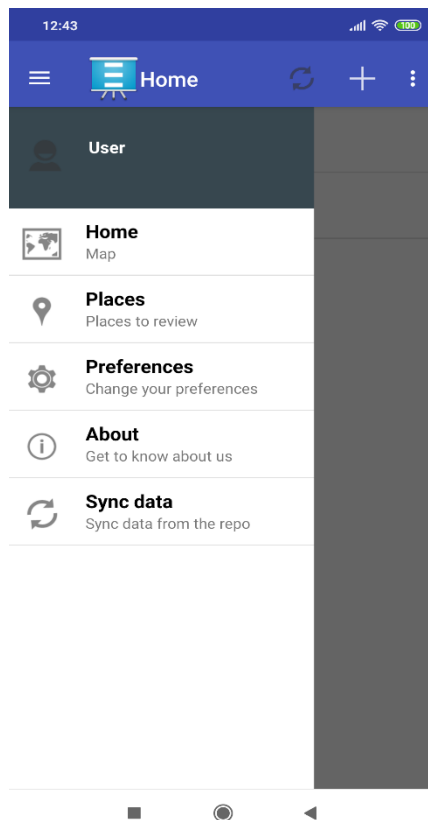
Slika 13. Kreiranje toast-a



Slika 14. Primer *pop-up* poruke

4. Navigation Drawer

Na slici 15 se nalazi primer *Navigation Drawer*-a koji ćemo kreirati.



Slika 15. Primer *Navigation Drawer*-a

Prvo deklarišemo *DrawerLayout* raspored (slika 16) i u njega smeštamo ostale pogledе. Dodajemo jedan pogled koji sadrži glavni sadržaj aktivnosti (kod posle komentara *The main content view*) i drugi pogled koji sadrži fioke za navigaciju (kod posle komentara *The navigation drawer*). Posebno se izdvaja prva fioka koja predstavlja profil (kod posle komentara *Profile Box*).

```

<androidx.drawerlayout.widget.DrawerLayout
    android:id="@+id/drawerLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <!-- The main content view -->
    <RelativeLayout
        android:id="@+id/mainContent"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!-- The navigation drawer -->
    <RelativeLayout
        android:layout_width="280dp"
        android:layout_height="match_parent"
        android:id="@+id/drawerPane"
        android:layout_gravity="start">

        <!-- Profile Box -->

```

Slika 16. *DrawerLayout*

Na slici 17 se nalazi početak *drawer*-a tj. deo za profil (*Profile Box*).

```

<RelativeLayout
    android:id="@+id/profileBox"
    android:layout_width="match_parent"
    android:layout_height="100dp"
    android:background="@color/material_blue_grey_800"
    android:padding="8dp" >

    <ImageView
        android:id="@+id/avatar"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:src="@drawable/ic_action_person"
        android:layout_marginTop="15dp"
        android:contentDescription="@string/image" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="42dp"
        android:layout_centerVertical="true"
        android:layout_marginStart="15dp"
        android:layout_toEndOf="@+id/avatar"
        android:orientation="vertical" >

```

Slika 17. Profile box

Nakon dela za profil, slede ostale stavke, koje smo programski kreirali u metodi *prepareMenu* (slika 18) koja se nalazi u klasi *MainActivity*.

```

private void prepareMenu(ArrayList<NavItem> mNavItems ){
    mNavItems.add(new NavItem(getString(R.string.home), getString(R.string.home_long), R.drawable.ic_action_map));
    mNavItems.add(new NavItem(getString(R.string.places), getString(R.string.places_long), R.drawable.ic_action_place));
    mNavItems.add(new NavItem(getString(R.string.preferences), getString(R.string.preferences_long), R.drawable.ic_action_se));
    mNavItems.add(new NavItem(getString(R.string.about), getString(R.string.about_long), R.drawable.ic_action_about));
    mNavItems.add(new NavItem(getString(R.string.sync_data), getString(R.string.sync_data_long), R.drawable.ic_action_refres));
}

```

Slika 18. Metoda *prepareMenu*

Da bismo povezali kreirane stavke sa pogledom, pravimo adapter *DrawerListAdapter* koji nasleđuje *BaseAdapter*. U metodi *onCreate* (slika 19), klase *MainActivity*, prosleđujemo kreirane stavke adapteru (linija 44) i taj adapter postavljamo na *drawer* (linija 51).

```

51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

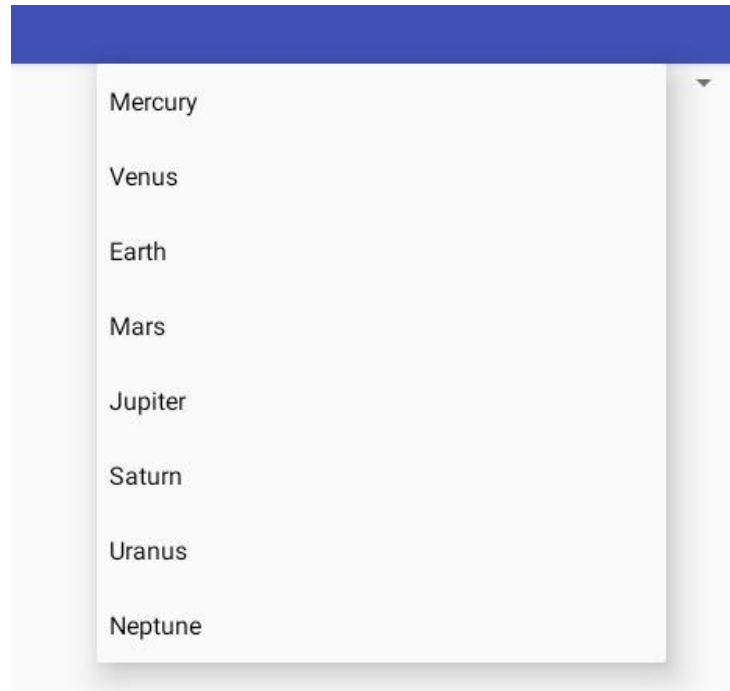
Slika 19. Postavljanje adaptera na *drawer*

5. Spinner

Spinner omogućava odabir jedne od više ponuđenih vrednosti. Na slikama 20 i 21 prikazan je izgled ovog elementa.



Slika 20. Izgled *spinner*-a sa izabranom opcijom



Slika 21. Izgled *spinner*-a sa ponuđenim opcijama

Za kreiranje elementa neohodno je definisati ga unutar XML datoteke kao na slici 22.

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1" />
```

Slika 22. Definisanje *spinner*-a

Ponuđene opcije mogu se kreirati unutar *values* foldera kao *string-array* (slika 23.).

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="spinner_option">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
        <item>Jupiter</item>
        <item>Saturn</item>
        <item>Uranus</item>
        <item>Neptune</item>
    </string-array>
</resources>
```

Slika 23. Datoteka sa predefinisanim vrednostima *spinner*-a

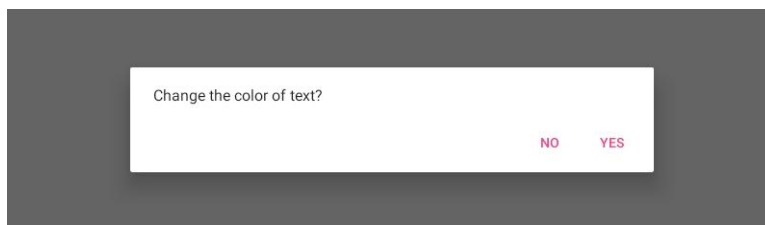
Za povezivanje prethodno prikazanih vrednosti, koriste se adapteri. Na slici 24. iskorišćen je *ArrayAdapter* kom se prosleđuje ugrađen *layout* - *simple_spinner_item* i element sa definisanim vrednostima - *spinner_option*.

```
Spinner spinner = view.findViewById(R.id.spinner);
// Create an ArrayAdapter using the string array and a default spinner layout
ArrayAdapter<String> adapter = new ArrayAdapter<>(getActivity(),
    android.R.layout.simple_spinner_item, getResources().getStringArray(R.array.spinner_option));
// Specify the layout to use when the list of choices appears
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
// Apply the adapter to the spinner
spinner.setAdapter(adapter);
```

Slika 24. Povezivanje *spinner*-a sa predefinisanim vrednostima

6. Dijalozi

Dijalozi su komponente koje predstavljaju male prozore. Prikazuju se korisniku uglavnom kada se traži potvrda neke akcije ili unos nekih dodatnih podataka. Primer jednostavnog dijaloga dat je na slici 25.



Slika 25. Dijalog sa "da" i "ne" opcijom

Postoje izvedene klase poput *AlertDialog* i *DatePickerDialog* koje nasleđuju baznu klasu *Dialog*. Svaka od ovih klasa poseduje predefinisane *layout*-e i elemente koje je moguće prilagođavati. Na slici 26. prikazana je implementacija *AlertDialog*-a. Pomoću *setMessage* prosleđuje se tekst dijaloga, a *setPositiveButton* i *setNegativeButton* postavljaju dugmad za potvrđivanje i odustajanje. Da bi se dijalog prikazao potrebno je kreirati ga sa *create()*, a zatim pozvati i metodu *show()*.

```
AlertDialog.Builder dialog = new AlertDialog.Builder(requireActivity());
dialog.setMessage("Change the color of the text?")
    .setCancelable(false)
    .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            Log.v( tag: "item", (String) parent.getItemAtPosition(position));
            ((TextView) parent.getChildAt( index: 0)).setTextColor(Color.MAGENTA);
        }
    })
    .setNegativeButton("No", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) { dialog.cancel(); }
    });
alertDialog = dialog.create();
alertDialog.show();
```

Slika 26. Implementacija *AlertDialog*-a

7. Domaći

Domaći se nalazi na *Canvas-u* (canvas.ftn.uns.ac.rs) na putanji *Вежбе/04 Вежбе/04 Задачамак.pdf*.

Primer možete preuzeti na sledećem linku: <https://gitlab.com/antesevicceca/mobilne-aplikacije>

Za dodatna pitanja možete se obratiti asistentima:

- Svetlana Antešević (svetlanaantesevic@uns.ac.rs)
- Jelena Matković (matkovic.jelena@uns.ac.rs)