

# MoneyMate

*Aplicación Android de finanzas personales*



**Autora:** Lucía Solís González

**Curso:** Desarrollo de aplicaciones multiplataforma

**Centro:** IES El Majuelo

## ÍNDICE

1. Resumen del proyecto.....	3
2. Requisitos hardware y software.....	3
2.1. Requisitos de hardware.....	3
2.2. Requisitos de software.....	3
3. Guía de la instalación del entorno de desarrollo.....	4
3.1. Descargar Android Studio para Mac.....	4
3.2. Instalar Android Studio en Mac.....	4
4. Puesta en marcha del entorno de desarrollo y la aplicación.....	5
5. Explicación del funcionamiento de la aplicación.....	11
5.1. Inicio de sesión y registro.....	12
5.2. Menú principal.....	12
5.3. Visualización de transacciones.....	12
5.4. Visualización de estadísticas.....	12
6. Explicación del código o procesos más complejos o interesantes.....	13
6.1. MainActivity (Inicio de sesión).....	15
6.2. RegistroActivity (Registrar usuario).....	16
6.3. MenuActivity (Menú principal).....	18
6.4. AgregarTransaccionActivity (Aregar ingreso o gasto).....	21
6.5. EditarUsuarioActivity (Modificar información del usuario).....	24
6.6. TransaccionesActivity (Lista de transacciones).....	27
6.7. EstadisticasActivity (Gráficos y presupuesto mensual).....	30
7. Mayores dificultades encontradas.....	36
7.1. Integración con Firebase Realtime Database.....	36
7.2. Procesamiento de fechas.....	36
7.3. Implementación de gráficos personalizados.....	36
7.4. Interactividad y cambios dinámicos.....	36
7.5. Adaptación del diseño a distintos dispositivos.....	36

## 1. Resumen del proyecto

**MoneyMate** es una aplicación Android desarrollada en **Kotlin** que permite a los usuarios llevar un control detallado de sus finanzas personales. La aplicación facilita el registro de ingresos y gastos, permite categorizarlos, visualizar estadísticas mediante gráficos y establecer presupuestos mensuales.

Utiliza **Firebase** como backend para la autenticación de usuarios y el almacenamiento de datos en tiempo real, lo que permite la sincronización entre dispositivos. La app está pensada para ser ligera, intuitiva y útil tanto para estudiantes como para adultos con interés en organizar su economía.

## 2. Requisitos hardware y software

### 2.1. Requisitos de hardware

- **Procesador:** Intel i5 o superior, Chip Apple M1 o superior.
- **RAM:** Mínimo 8 GB (se recomienda 16 GB, especialmente para usar el emulador).
- **Almacenamiento:** Al menos 8 GB libres para Android Studio, el SDK de Android y el emulador.
- **Resolución de pantalla:** Mínimo 1280 x 800, recomendado 1920 x 1080.

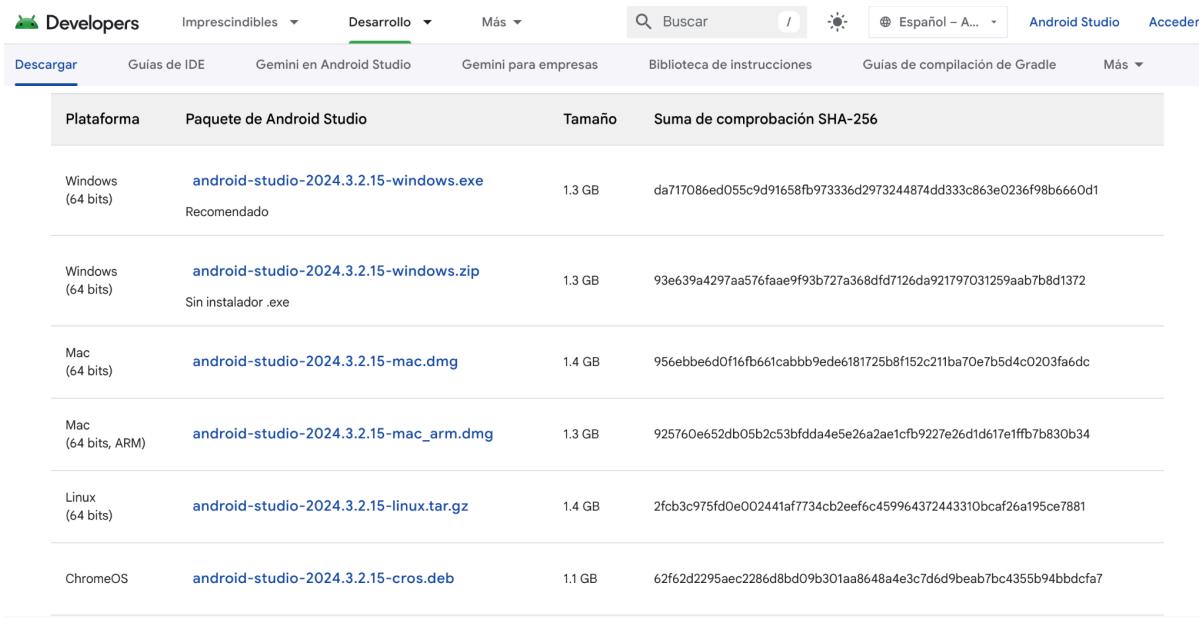
### 2.2. Requisitos de software

- **Sistema operativo:** Windows 10 de 64 bits o superior, macOS 12 o superior, o cualquier distribución Linux de 64 bits.
- **Android Studio:** Descargar la versión más reciente disponible en la página web de [Android Developers](#).
- **Java Development Kit (JDK):** Versión 11 o superior (incluido en Android Studio).
- **Android SDK:** Imprescindible para compilar, probar y generar APKs/APP Bundles (incluido en Android Studio).
- **Kotlin:** Lenguaje de programación (incluido en Android Studio).
- **Firebase:** Iniciar sesión con una cuenta de Google y crear un proyecto de Firebase.
- **Firebase SDKs:** Para integrarlo en la aplicación se necesitan agregar los SDKs de Firebase correspondientes (Realtime Database).

### 3. Guía de la instalación del entorno de desarrollo

#### 3.1. Descargar Android Studio para Mac

Enlace: <https://developer.android.com/studio?hl=es-419>

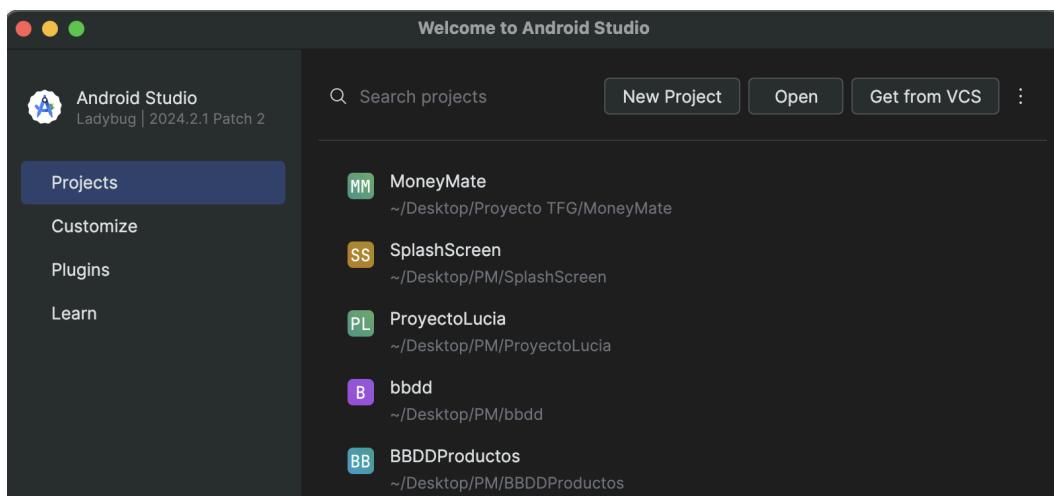


Plataforma	Paquete de Android Studio	Tamaño	Suma de comprobación SHA-256
Windows (64 bits)	<a href="#">android-studio-2024.3.2.15-windows.exe</a> Recomendado	1.3 GB	da717086ed055c9d91658fb973336d2973244874dd333c863e0236f98b6660d1
Windows (64 bits)	<a href="#">android-studio-2024.3.2.15-windows.zip</a> Sin instalador .exe	1.3 GB	93e639a4297aa576faae9f93b727a368dfd7126da921797031259aab7b8d1372
Mac (64 bits)	<a href="#">android-studio-2024.3.2.15-mac.dmg</a>	1.4 GB	956ebbe6d0f16fb661cabbb9ede6181725b8f152c211ba70e7b5d4c0203fa6dc
Mac (64 bits, ARM)	<a href="#">android-studio-2024.3.2.15-mac_arm.dmg</a>	1.3 GB	925760e652db05b2c53bfdda4e5e26a2ae1cfb9227e26d1d617e1ffb7b830b34
Linux (64 bits)	<a href="#">android-studio-2024.3.2.15-linux.tar.gz</a>	1.4 GB	2fc3c975fd0e002441af7734cb2eef6c459964372443310bcfa26a195ce7881
ChromeOS	<a href="#">android-studio-2024.3.2.15-cros.deb</a>	1.1 GB	62f62d2295aec2286d8bd09b301aa8648a4e3c7d6d9beab7bc4355b94bbdcfa7

#### 3.2. Instalar Android Studio en Mac

Para instalar Android Studio en Mac, seguí estos pasos:

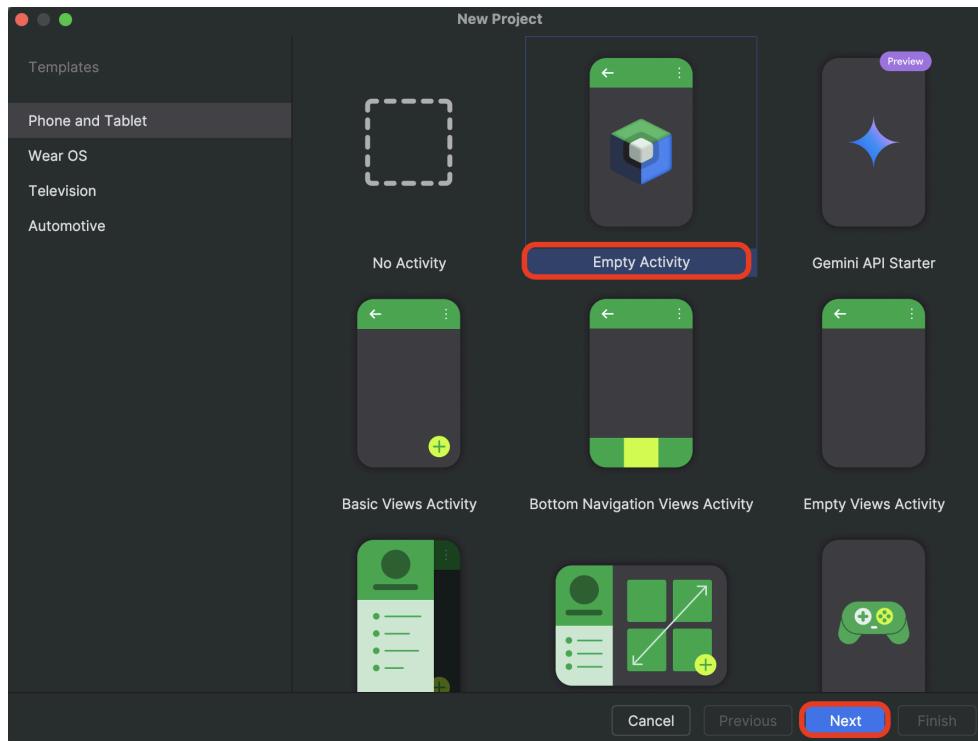
1. Ejecutar el archivo descargado DMG de Android Studio.
2. Arrastrar Android Studio a la carpeta **Aplicaciones** y, luego, iniciararlo.
3. Elegir si quería importar configuraciones anteriores de Android Studio y hacer clic en **OK**.
4. Completar el **Setup Wizard** de Android Studio, que incluye la descarga de los componentes del SDK de Android que se necesitan para el desarrollo.



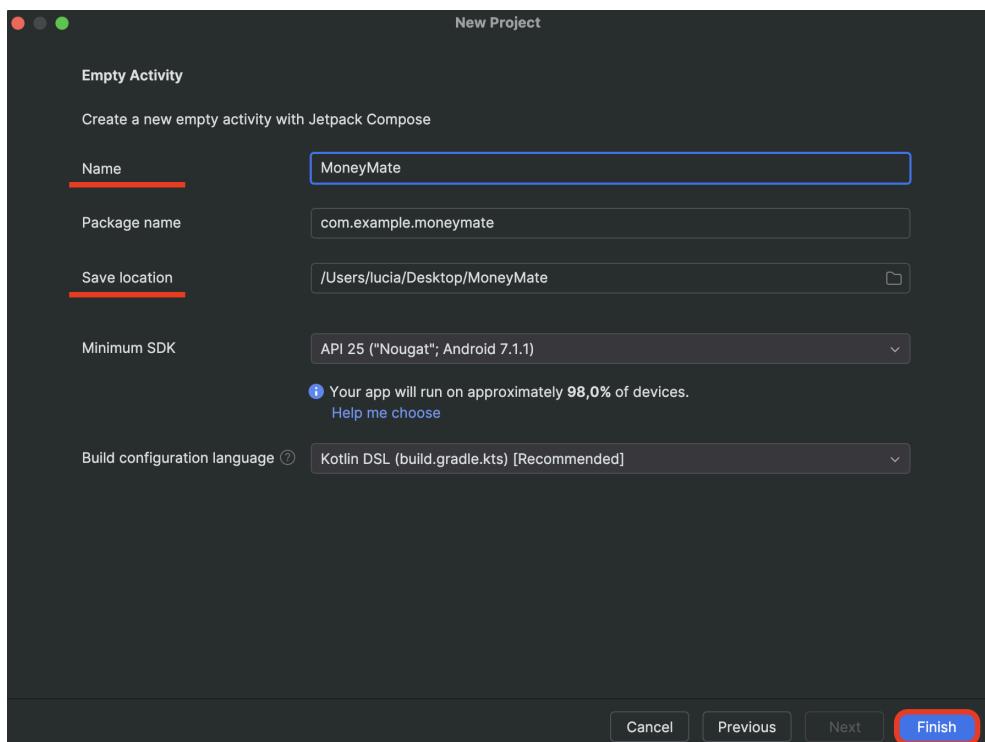
## 4. Puesta en marcha del entorno de desarrollo y la aplicación

Una vez instalado Android Studio, seguí estos pasos para crear un nuevo proyecto:

1. Pulsé en **New Project > Empty Activity > Next**

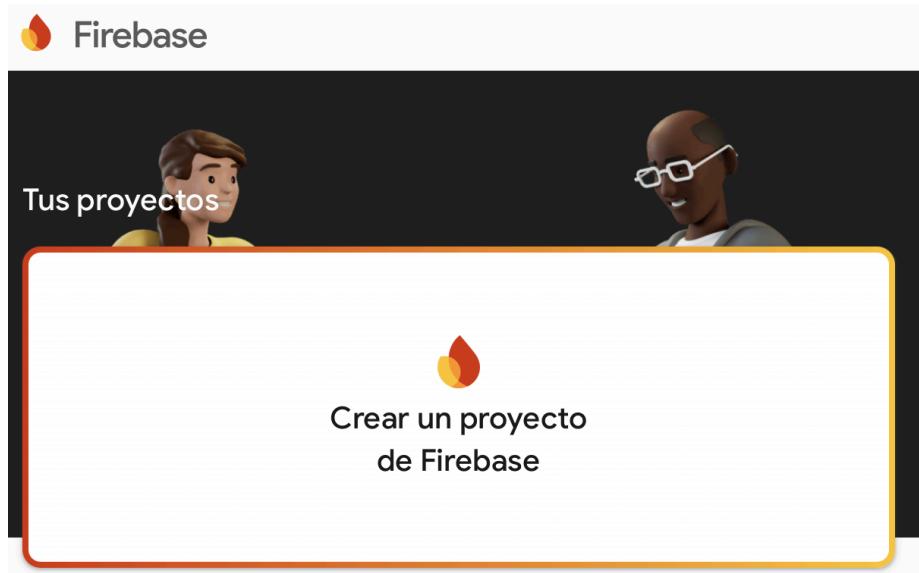


2. Elegí el nombre del proyecto y la ubicación, luego le di a **Finish**.

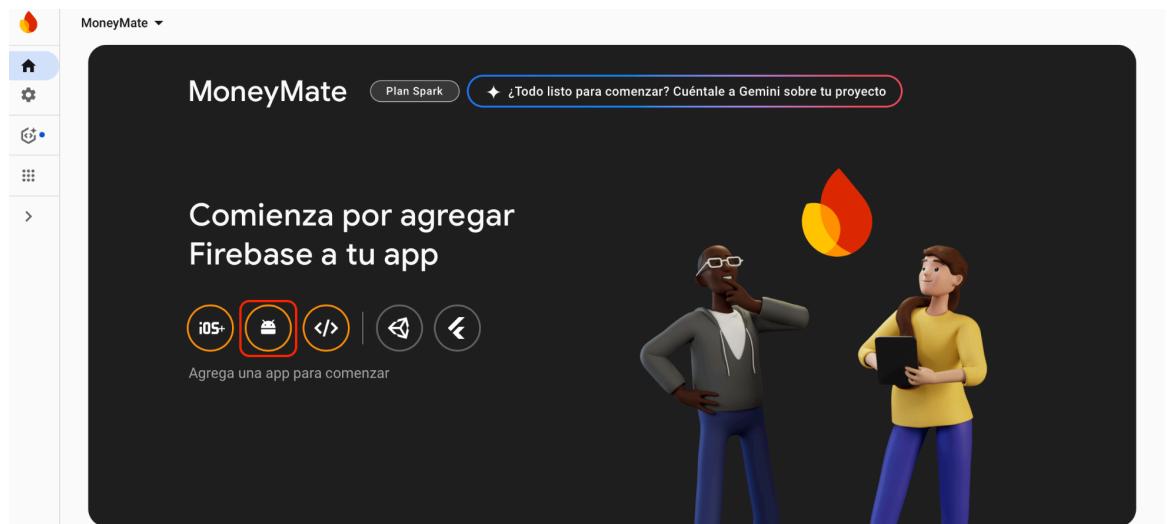


## Proyecto de Desarrollo de Aplicaciones Multiplataforma

3. Una vez creado el proyecto en Android Studio, inicié sesión en Firebase y le di a crear un proyecto. Le puse nombre y seguí los pasos.



4. Una vez creado pulsé *Android* para agregar la app, puse el nombre del paquete Android (*namespace = "com.example.moneymate"*), que se encuentra en *build.gradle.kts (:app)*, descargué *google-services.json* y lo arrastré a la carpeta *app* dentro de *Project* en Android Studio.



## **× Agregar Firebase a tu app para Android**

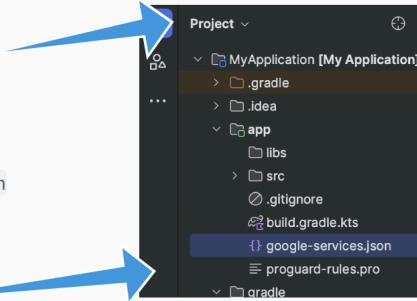
## 2 Descargar y, luego, agregar el archivo de configuración

Instrucciones para Android Studio a continuación | [Unity](#)  [C++](#) 

## ↓ Descargar google-services.json

Cambia a la vista Proyecto en Android Studio para ver el directorio raíz de tu proyecto.

Mueve el archivo descargado `google-services.json` al directorio raíz de tu módulo (nivel de app).



Siguiente

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

5. Luego, seleccioné **Realtime Database** para crear mi base de datos y dejé la configuración por defecto. Una vez creada le cambié las reglas a **true** para poder editar la base de datos.

The screenshot shows the Firebase console interface for the project "MoneyMate". The left sidebar has icons for Home, Settings, Functions, and a grid (selected). The top navigation bar says "MoneyMate" and "Descripción general del proyecto > Productos y funciones de Firebase". Below the navigation are two cards: "Realtime Database" (selected) and "Storage". The "Realtime Database" card has a yellow gradient icon, the text "Realtime Database", "Almacena y sincroniza datos en tiempo real", and a "Crear una base de datos" button.

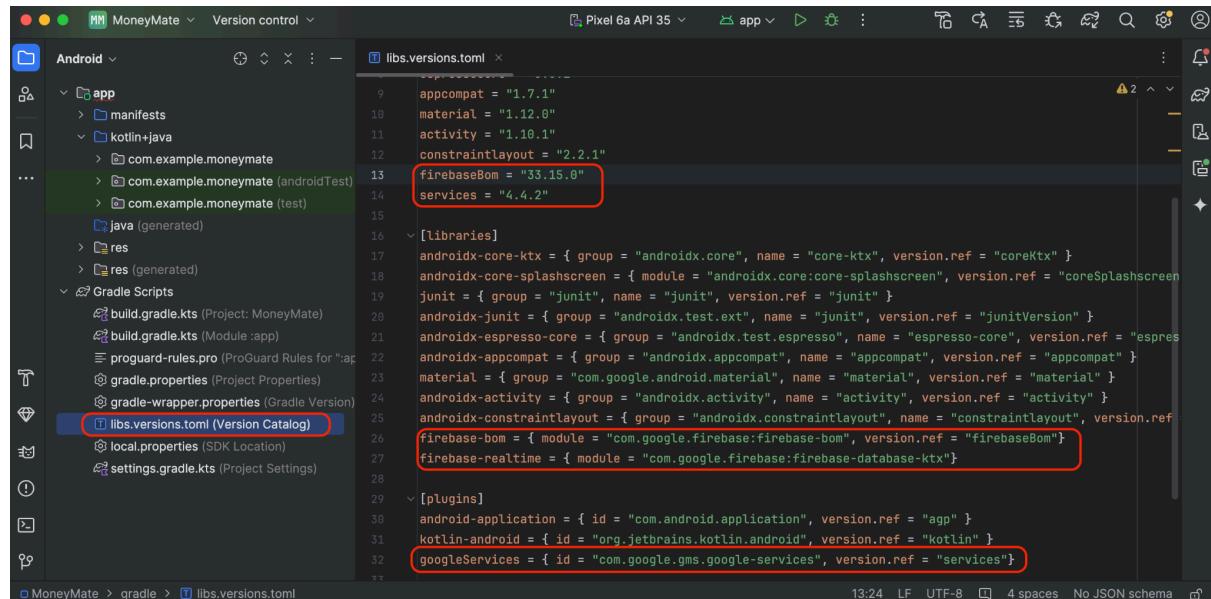
The screenshot shows the "Realtime Database" creation page. The left sidebar is the same as the previous screen. The main area has a dark background with a blue server icon and user network icons. It says "Realtime Database", "Almacena y sincroniza datos en tiempo real", "Crear una base de datos" (button highlighted with a red box), and "Preguntarle a Gemini".

The screenshot shows the "Realtime Database" rules editor. The left sidebar is the same. The top navigation bar has tabs: "Datos", "Reglas" (selected), "Copias de seguridad", "Uso", and "Extensions". Below the tabs is a "Zona de pruebas de reglas" button. The main area shows a code editor with the following rules:

```
1 ▾ {  
2 ▾   "rules": {  
3       ".read": true,  
4       ".write": true  
5     }  
6   }.
```

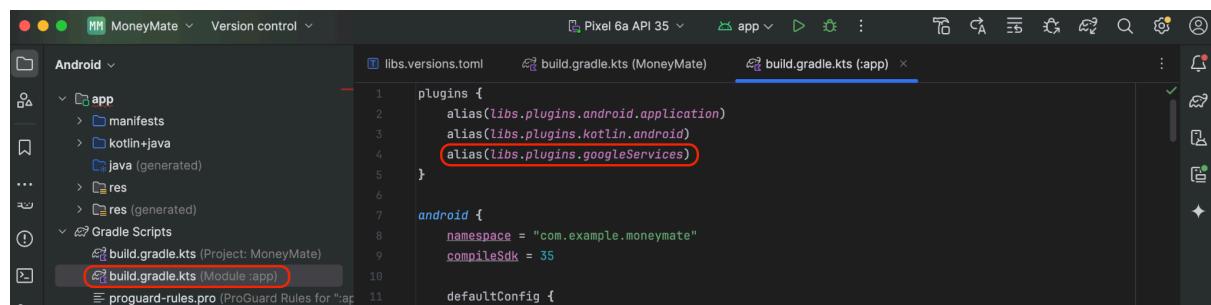
## Proyecto de Desarrollo de Aplicaciones Multiplataforma

6. Ya solo queda añadir las librerías necesarias para Firebase. Primero las añado en **libs.versions.toml**, dentro de *Gradle Scripts*, y sincronizo.

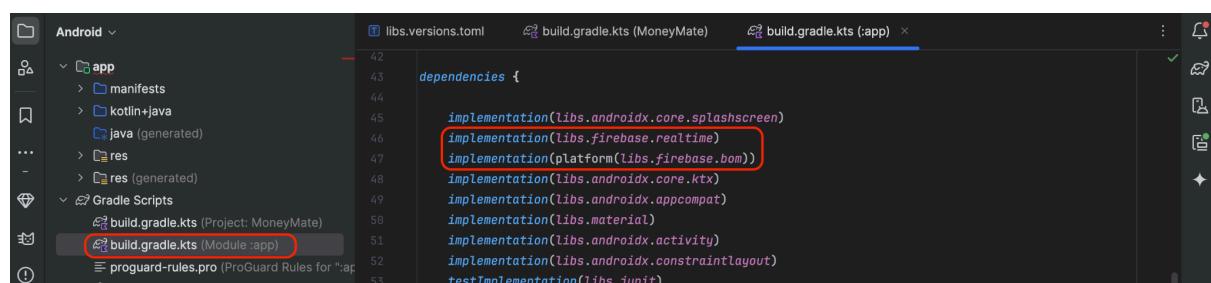


```
9 appcompat = "1.7.1"
10 material = "1.12.0"
11 activity = "1.10.1"
12 constraintlayout = "2.2.1"
13 firebaseBom = "33.15.0"
14 services = "4.4.2"
15
16 [libraries]
17 androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref = "coreKtx" }
18 androidx-core-splashscreen = { module = "androidx.core:core-splashscreen", version.ref = "coreSplashscreen" }
19 junit = { group = "junit", name = "junit", version.ref = "junit" }
20 androidx-junit = { group = "androidx.test.ext", name = "junit", version.ref = "junitVersion" }
21 androidx-espresso-core = { group = "androidx.test.espresso", name = "espresso-core", version.ref = "espressoVersion" }
22 androidx-appcompat = { group = "androidx.appcompat", name = "appcompat", version.ref = "appcompat" }
23 material = { group = "com.google.android.material", name = "material", version.ref = "material" }
24 androidx-activity = { group = "androidx.activity", name = "activity", version.ref = "activity" }
25 androidx-constraintlayout = { group = "androidx.constraintlayout", name = "constraintlayout", version.ref = "constraintlayout" }
26 firebase-bom = { module = "com.google.firebaseio:firebase-bom", version.ref = "firebaseBom" }
27 firebase-realm = { module = "com.google.firebaseio:firebase-database-ktx" }
28
29 [plugins]
30 android-application = { id = "com.android.application", version.ref = "agp" }
31 kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref = "kotlin" }
32 googleServices = { id = "com.google.gms.google-services", version.ref = "services" }
```

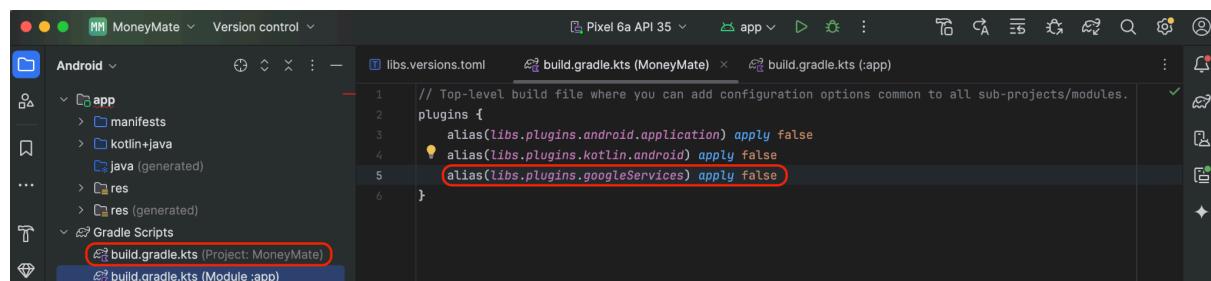
7. Luego en **build.gradle.kts (Module :app)** y en **build.gradle.kts (Project: MoneyMate)** añado lo siguiente y después le doy a sincronizar:



```
1 plugins {
2     alias(libs.plugins.android.application)
3     alias(libs.plugins.kotlin.android)
4     alias(libs.plugins.googleServices)
5 }
6
7 android {
8     namespace = "com.example.moneymate"
9     compileSdk = 35
10
11 defaultConfig {
```



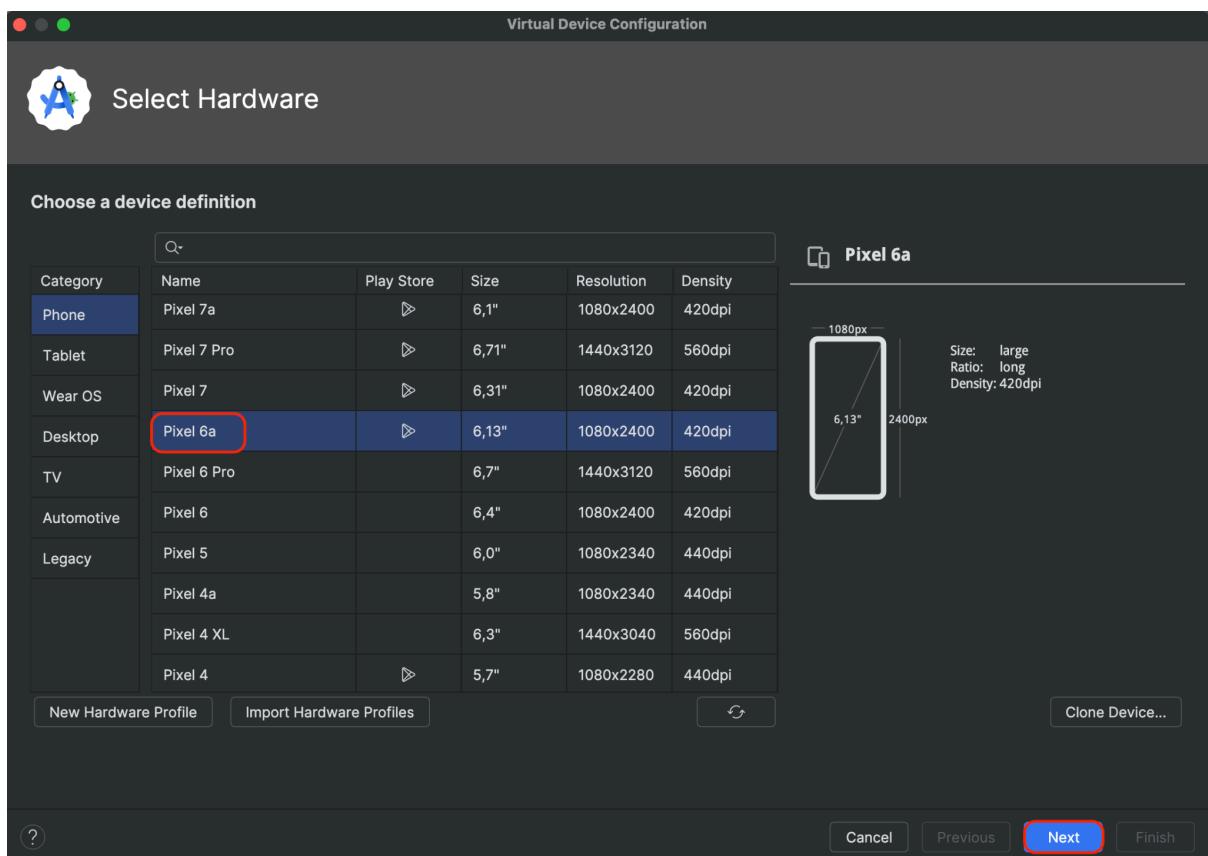
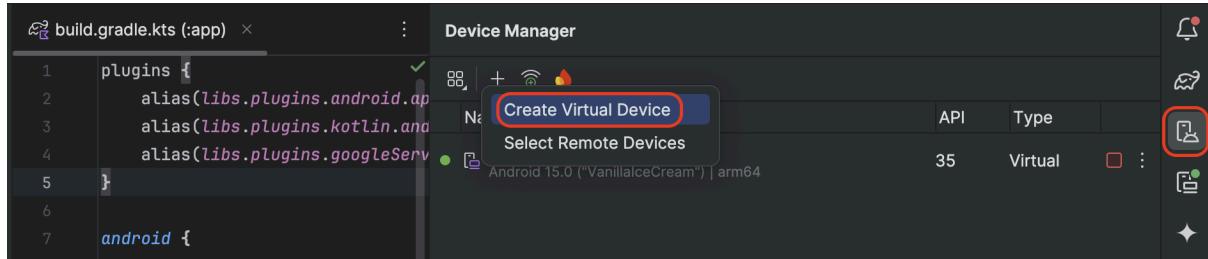
```
42 dependencies {
43     implementation(libs.appcompat)
44     implementation(libs.core.ktx)
45     implementation(libs.material)
46     implementation(libs.constraintlayout)
47     implementation(libs.firebase.core)
48     implementation(libs.firebase.storage)
49     implementation(libs.firebase.database)
50     implementation(libs.firebase.functions)
51     implementation(libs.firebase.crashlytics)
52     implementation(libs.firebase.messaging)
53 }
```

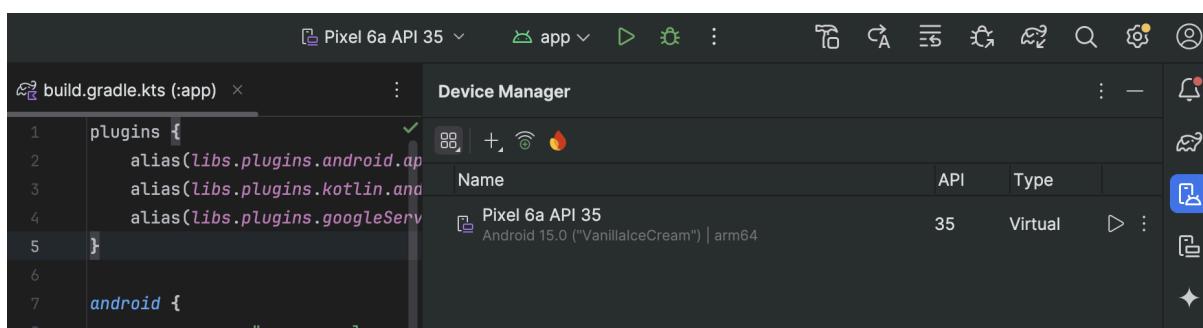
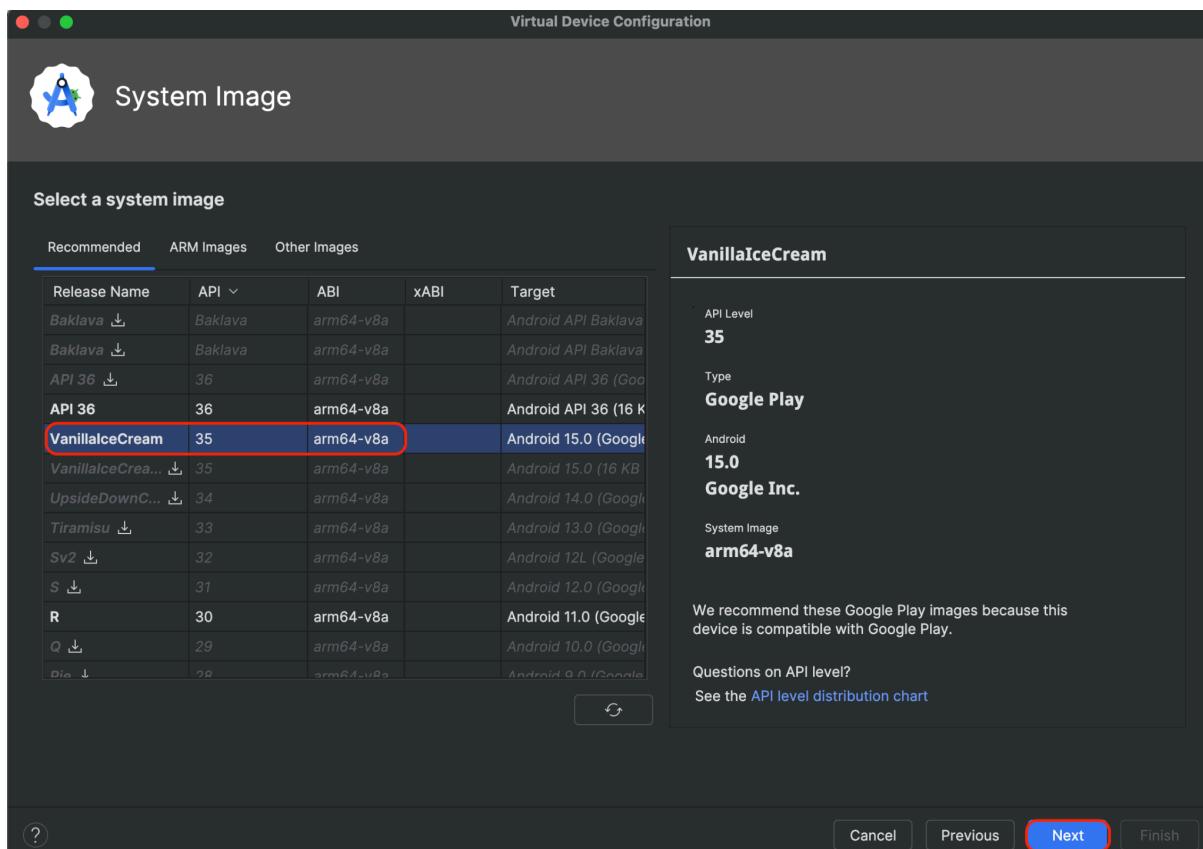


```
1 // Top-level build file where you can add configuration options common to all sub-projects/modules.
2 plugins {
3     alias(libs.plugins.android.application) apply false
4     alias(libs.plugins.kotlin.android) apply false
5     alias(libs.plugins.googleServices) apply false
6 }
```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

8. Al terminar la sincronización, creé un nuevo emulador para ir probando la aplicación. Pulsé en **Device Manager** y le di a **Create Virtual Device**. Elegí **Phone Pixel 6a** y **API 35** como imágenes del sistema.





Ya estaría todo lo necesario para empezar a crear el código.

## 5. Explicación del funcionamiento de la aplicación

Mi aplicación, **MoneyMate**, está diseñada para ayudar a los usuarios a gestionar sus finanzas personales de una forma clara, visual e intuitiva. Su funcionamiento se divide en cuatro secciones principales: inicio de sesión o registro del usuario, menú principal, visualización de transacciones y visualización de estadísticas. Gracias al uso de Firebase, los datos se sincronizan automáticamente en la nube, lo que permite acceder a la información desde distintos dispositivos con una misma cuenta.

### 5.1. Inicio de sesión y registro

Al iniciar la aplicación, el usuario se encuentra con una pantalla de autenticación. Utilizo el **correo** del usuario como identificador, el cual guardo en **Firebase Realtime Database**, esto me permite asegurar que cada usuario tenga acceso únicamente a sus propios datos financieros. Si el usuario no tiene una cuenta, puede registrarse directamente desde la app.

### 5.2. Menú principal

Una vez autenticado, el usuario accede a la pantalla principal donde puede ver su **saldo total** junto con los **ingresos** y **gastos** totales, además de registrar **transacciones**. También tengo un pequeño desplegable con las opciones de **Editar usuario** y **Cerrar sesión**.

Cada transacción contiene los siguientes campos: *Tipo* (ingreso o gasto), *Cantidad*, *Categoría* (como alimentación, transporte, ocio, etc.) y *Fecha*. Estas transacciones se almacenan por usuario en **Firebase Realtime Database**. Esta estructura me permite sincronizar los datos entre dispositivos en tiempo real.

### 5.3. Visualización de transacciones

La aplicación incluye una pantalla donde poder consultar todas las **transacciones** realizadas por el usuario. Estas transacciones se muestran en una lista cronológica con la información clave de cada operación. Además, esta pantalla incorpora una funcionalidad de selección que brinda al usuario la posibilidad de mostrar únicamente los **ingresos** o los **gastos**, según su preferencia. Esto simplifica la interpretación de sus finanzas, al permitir enfocarse en una categoría concreta del movimiento económico.

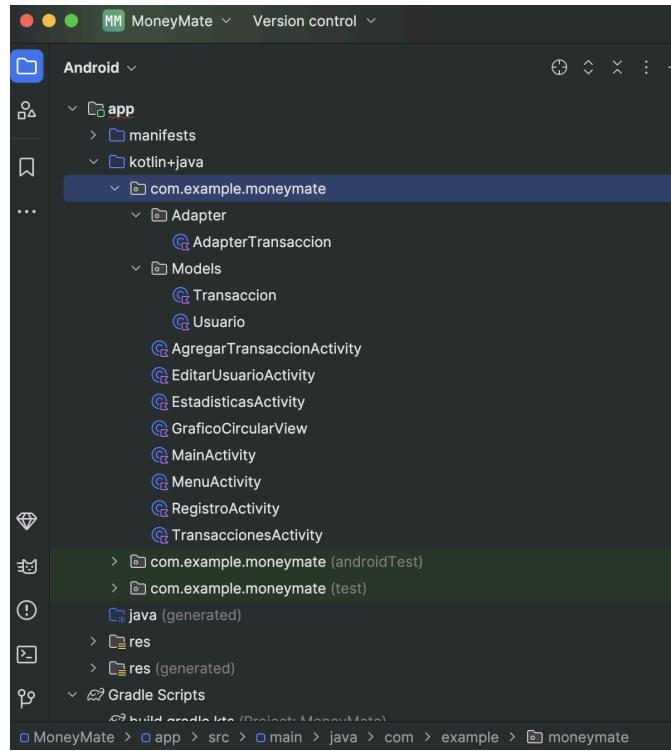
### 5.4. Visualización de estadísticas

Para ofrecer una visión más global, he incluido gráficos dinámicos a partir de las transacciones almacenadas. Un **gráfico de barras vertical** que representa los gastos e ingresos mensuales y un **gráfico circular** que muestra el porcentaje gastado de cada categoría, en el mes actual. Ambos gráficos se actualizan automáticamente cuando se registran nuevas transacciones, utilizando la información obtenida desde Firebase.

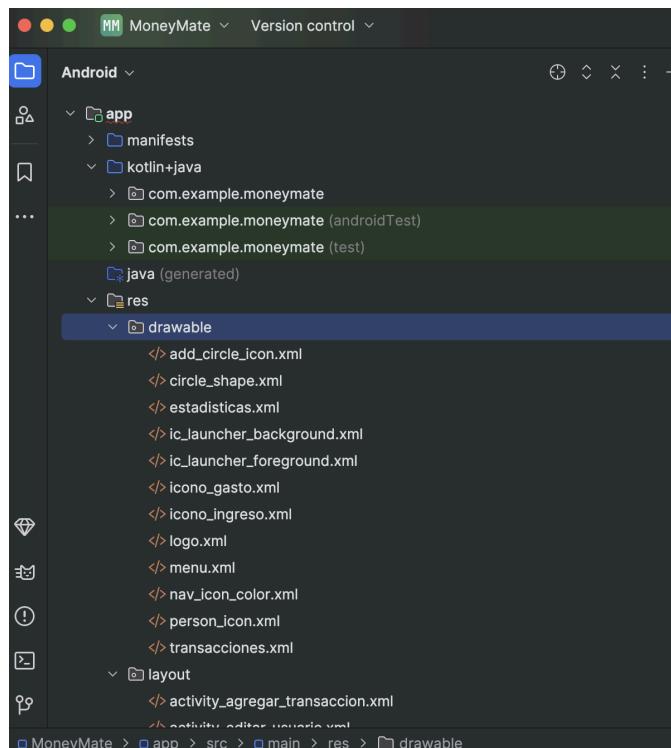
Además, he añadido la opción de establecer un **presupuesto mensual**. A medida que el usuario va registrando gastos, se actualiza en tiempo real el porcentaje gastado respecto al presupuesto fijado, y se muestra mediante una barra de progreso.

## 6. Explicación del código o procesos más complejos o interesantes

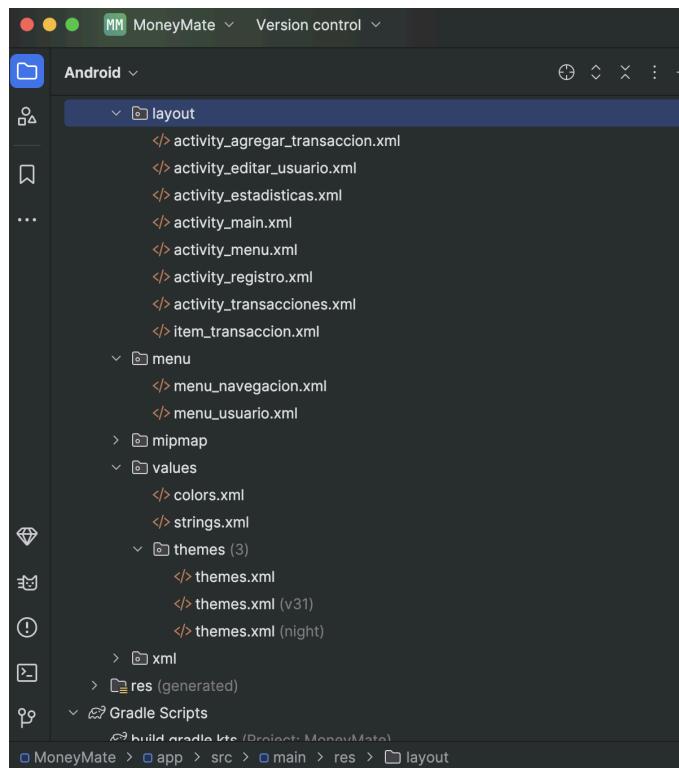
Mi aplicación se estructura de la siguiente forma:



Dentro del paquete **kotlin+java** tengo toda la lógica de mi app. Cree las ***data class*** de **Usuario** y **Transacción**, un **adapter** para manejar una lista de transacciones en un **RecyclerView** y todas las **activities** necesarias para la funcionalidad de la aplicación.



## Proyecto de Desarrollo de Aplicaciones Multiplataforma

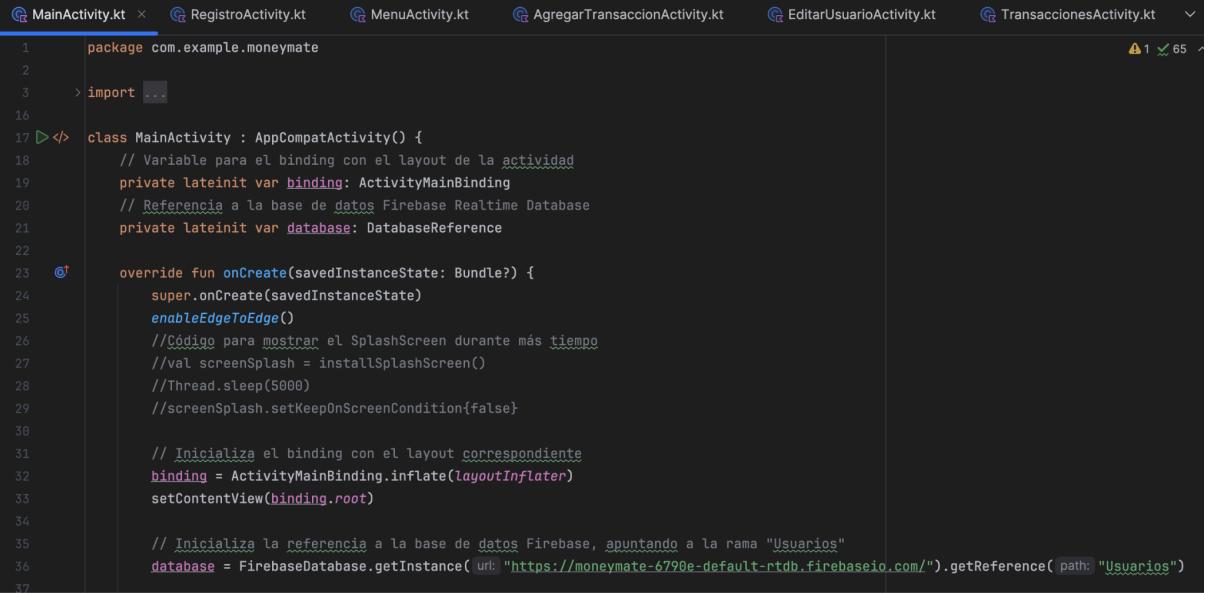


En la carpeta **res** de mi proyecto, guardo todos los recursos que utilizo para la aplicación:

- **drawable:** Aquí se encuentran los iconos que he utilizado.
- **layout:** Contiene todos los archivos XML correspondientes a las interfaces de mis Activitys.
- **menu:** Tengo los XML para el menú de navegación y el menú de usuario.
- **values:** Almacena los colores, las cadenas de texto (strings) y los temas que he definido para la app.

Una vez clara la estructura, voy a explicar el código que considero más complejo, que sería el de las **Activitys**. Para facilitar la comprensión, mostraré el **código comentado** y su **resultado** correspondiente:

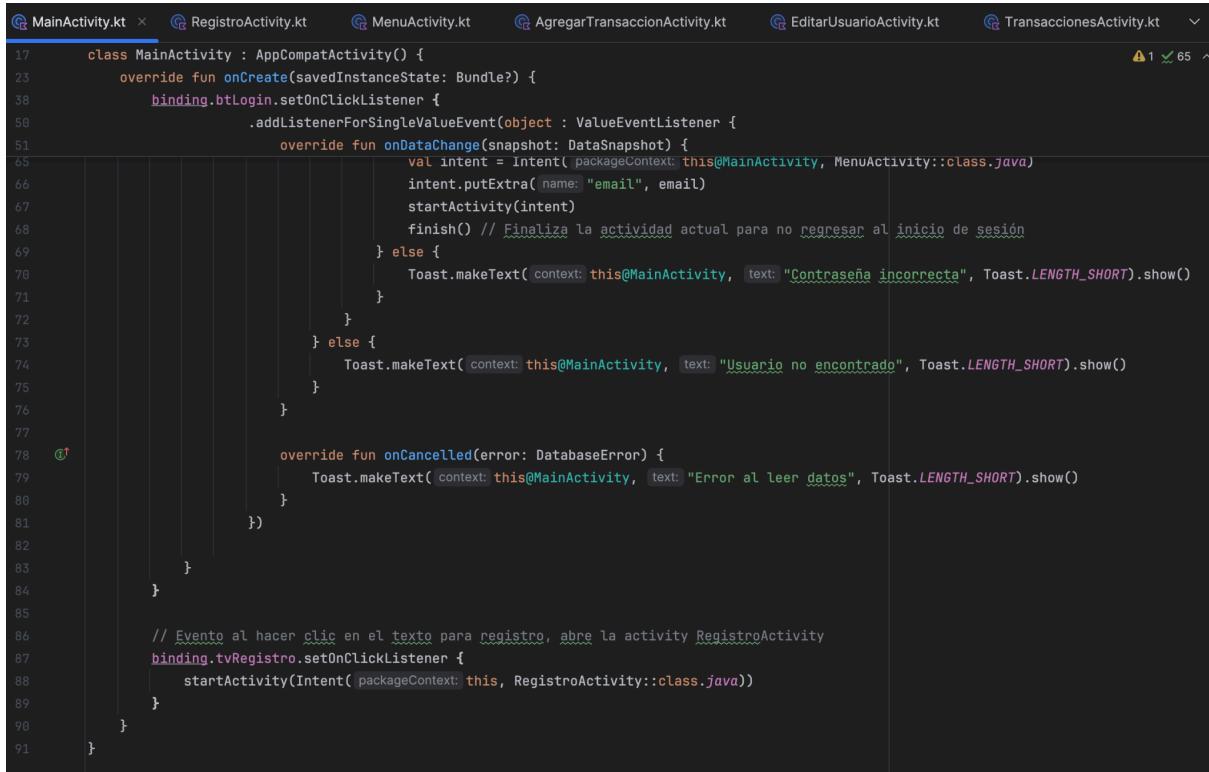
## 6.1. MainActivity (Inicio de sesión)



```

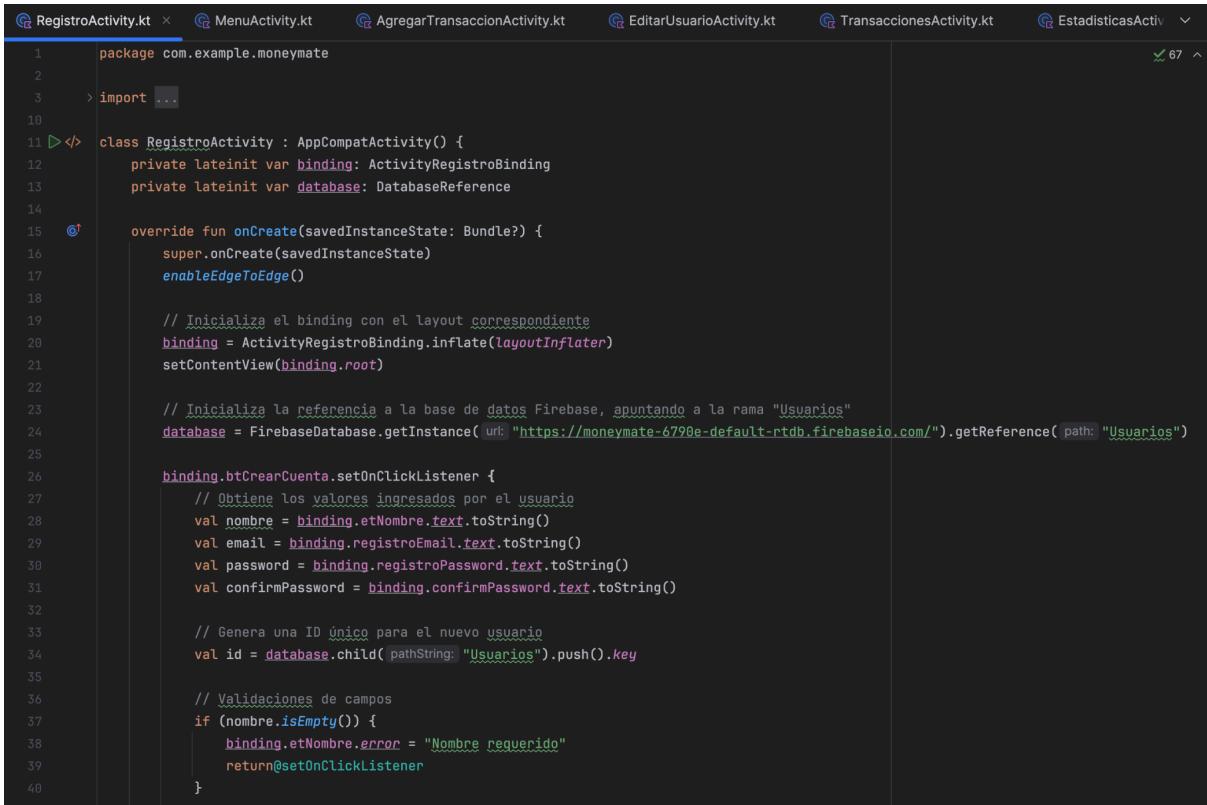
1 package com.example.moneymate
2
3 > import ...
4
5
6 <> class MainActivity : AppCompatActivity() {
7     // Variable para el binding con el layout de la actividad
8     private lateinit var binding: ActivityMainBinding
9     // Referencia a la base de datos Firebase Realtime Database
10    private lateinit var database: DatabaseReference
11
12
13    override fun onCreate(savedInstanceState: Bundle?) {
14        super.onCreate(savedInstanceState)
15        enableEdgeToEdge()
16        // Código para mostrar el SplashScreen durante más tiempo
17        //val screenSplash = installSplashScreen()
18        //Thread.sleep(5000)
19        //screenSplash.setKeepOnScreenCondition{false}
20
21        // Inicializa el binding con el layout correspondiente
22        binding = ActivityMainBinding.inflate(layoutInflater)
23        setContentView(binding.root)
24
25        // Inicializa la referencia a la base de datos Firebase, apuntando a la rama "Usuarios"
26        database = FirebaseDatabase.getInstance(url: "https://moneymate-6790e-default.firebaseio.com/").getReference(path: "Usuarios")
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
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
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
127
128
129
129
130
131
132
133
134
135
136
137
137
138
139
139
140
141
142
143
144
145
146
147
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
217
218
219
219
220
221
222
223
224
225
226
227
227
228
229
229
230
231
232
233
234
235
235
236
237
237
238
239
239
240
241
242
243
244
245
245
246
247
247
248
249
249
250
251
252
253
253
254
255
255
256
256
257
257
258
258
259
259
260
260
261
261
262
262
263
263
264
264
265
265
266
266
267
267
268
268
269
269
270
270
271
271
272
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1
```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma



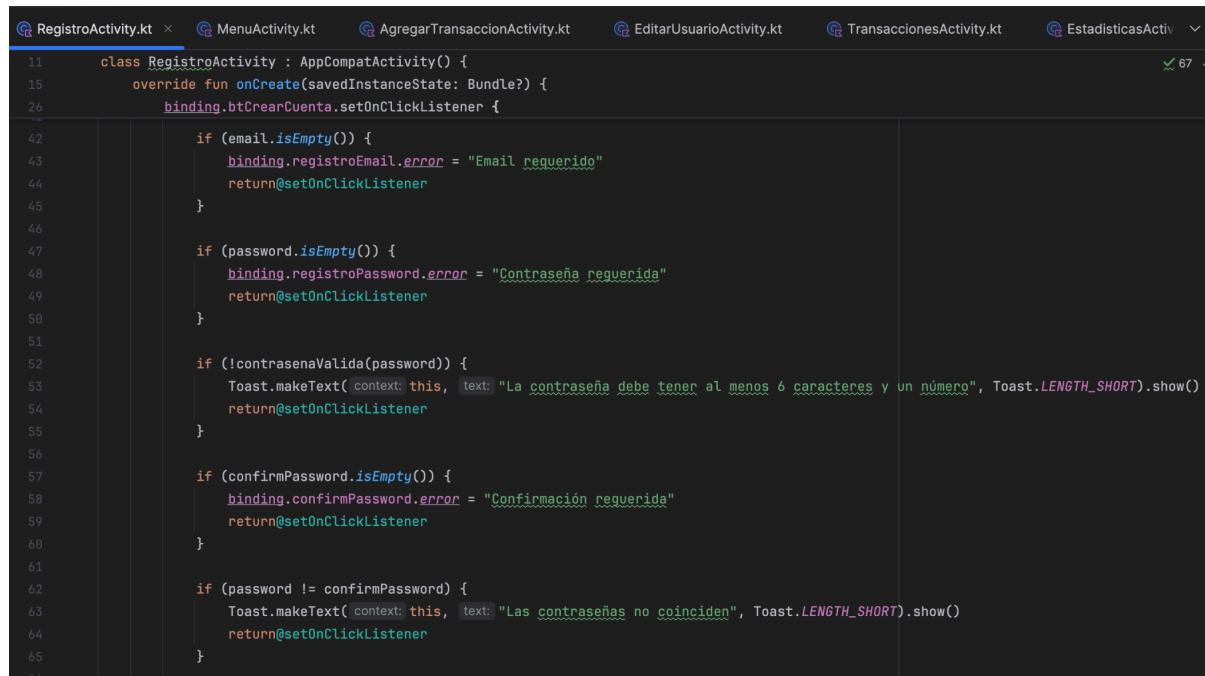
```
17 class MainActivity : AppCompatActivity() {
23     override fun onCreate(savedInstanceState: Bundle?) {
24         binding.btLogin.setOnClickListener {
25             addListenerForSingleValueEvent(object : ValueEventListener {
26                 override fun onDataChange(snapshot: DataSnapshot) {
27                     val intent = Intent(packageContext: this@MainActivity, MenuActivity::class.java)
28                     intent.putExtra(name: "email", email)
29                     startActivity(intent)
30                     finish() // Finaliza la actividad actual para no regresar al inicio de sesión
31                 } else {
32                     Toast.makeText(context: this@MainActivity, text: "Contraseña incorrecta", Toast.LENGTH_SHORT).show()
33                 }
34             }
35             override fun onCancelled(error: DatabaseError) {
36                 Toast.makeText(context: this@MainActivity, text: "Error al leer datos", Toast.LENGTH_SHORT).show()
37             }
38         })
39     }
40
41     // Evento al hacer clic en el texto para registro, abre la activity RegistroActivity
42     binding.tvRegistro.setOnClickListener {
43         startActivity(Intent(packageContext: this, RegistroActivity::class.java))
44     }
45 }
```

## 6.2. RegistroActivity (Registrar usuario)

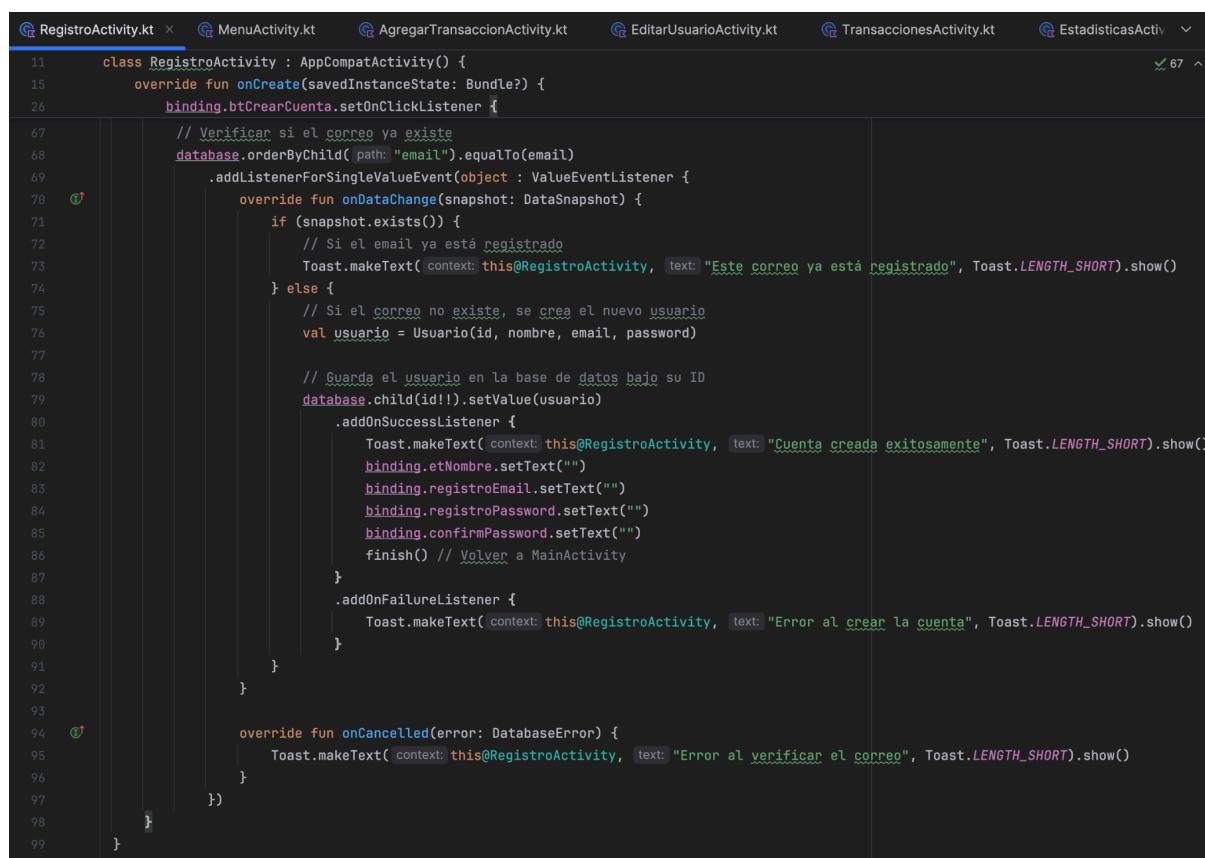


```
1 package com.example.moneymate
2
3 > import ...
4
5 class RegistroActivity : AppCompatActivity() {
6     private lateinit var binding: ActivityRegistroBinding
7     private lateinit var database: DatabaseReference
8
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         enableEdgeToEdge()
12
13         // Inicializa el binding con el layout correspondiente
14         binding = ActivityRegistroBinding.inflate(layoutInflater)
15         setContentView(binding.root)
16
17         // Inicializa la referencia a la base de datos Firebase, apuntando a la rama "Usuarios"
18         database = FirebaseDatabase.getInstance(url: "https://moneymate-6790e-default.firebaseio.com/").getReference(path: "Usuarios")
19
20         binding.btCrearCuenta.setOnClickListener {
21             // Obtiene los valores ingresados por el usuario
22             val nombre = binding.etNombre.text.toString()
23             val email = binding.registroEmail.text.toString()
24             val password = binding.registroPassword.text.toString()
25             val confirmPassword = binding.confirmPassword.text.toString()
26
27             // Genera una ID única para el nuevo usuario
28             val id = database.child(pathString: "Usuarios").push().key
29
30             // Validaciones de campos
31             if (nombre.isEmpty()) {
32                 binding.etNombre.error = "Nombre requerido"
33                 return@setOnClickListener
34             }
35         }
36     }
37 }
```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

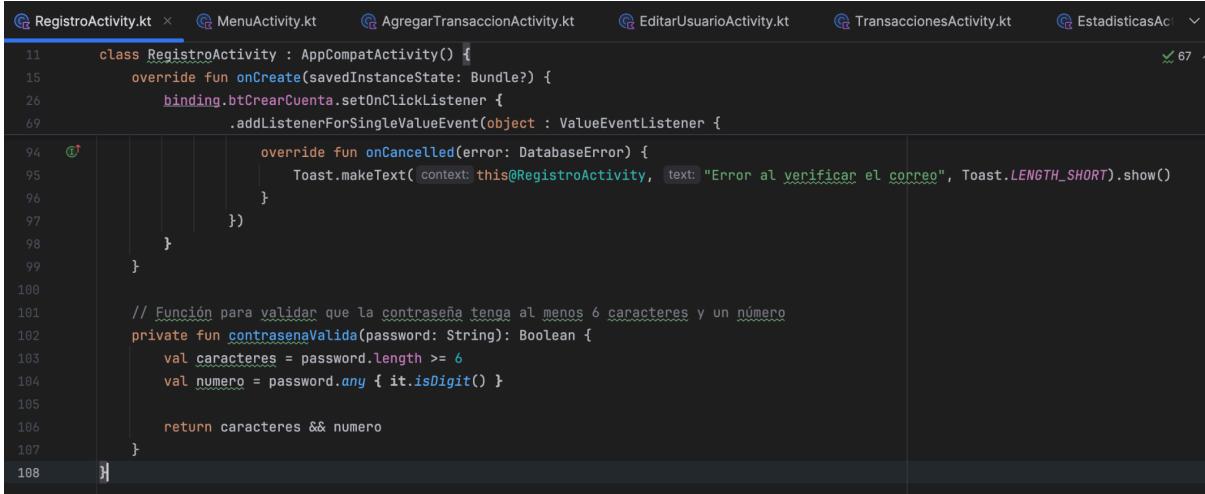


```
11     class RegistroActivity : AppCompatActivity() {
12         override fun onCreate(savedInstanceState: Bundle?) {
13             binding.btCrearCuenta.setOnClickListener {
14
15                 if (email.isEmpty()) {
16                     binding.registroEmail.error = "Email requerido"
17                     return@setOnClickListener
18                 }
19
20                 if (password.isEmpty()) {
21                     binding.registroPassword.error = "Contraseña requerida"
22                     return@setOnClickListener
23                 }
24
25                 if (!contraseñaValida(password)) {
26                     Toast.makeText(context: this, text: "La contraseña debe tener al menos 6 caracteres y un número", Toast.LENGTH_SHORT).show()
27                     return@setOnClickListener
28                 }
29
30                 if (confirmPassword.isEmpty()) {
31                     binding.confirmPassword.error = "Confirmación requerida"
32                     return@setOnClickListener
33                 }
34
35                 if (password != confirmPassword) {
36                     Toast.makeText(context: this, text: "Las contraseñas no coinciden", Toast.LENGTH_SHORT).show()
37                     return@setOnClickListener
38                 }
39             }
40         }
41     }
```



```
11     class RegistroActivity : AppCompatActivity() {
12         override fun onCreate(savedInstanceState: Bundle?) {
13             binding.btCrearCuenta.setOnClickListener {
14
15                 // Verificar si el correo ya existe
16                 database.orderByChild(path: "email").equalTo(email)
17                     .addListenerForSingleValueEvent(object : ValueEventListener {
18                         override fun onDataChange(snapshot: DataSnapshot) {
19                             if (snapshot.exists()) {
20                                 // Si el email ya está registrado
21                                 Toast.makeText(context: this@RegistroActivity, text: "Este correo ya está registrado", Toast.LENGTH_SHORT).show()
22                             } else {
23                                 // Si el correo no existe, se crea el nuevo usuario
24                                 val usuario = Usuario(id, nombre, email, password)
25
26                                 // Guarda el usuario en la base de datos bajo su ID
27                                 database.child(id!!).setValue(usuario)
28                                     .addOnSuccessListener {
29                                         Toast.makeText(context: this@RegistroActivity, text: "Cuenta creada exitosamente", Toast.LENGTH_SHORT).show()
30                                         binding.etNombre.setText("")
31                                         binding.registroEmail.setText("")
32                                         binding.registroPassword.setText("")
33                                         binding.confirmPassword.setText("")
34                                         finish() // Volver a MainActivity
35                                     }
36                                     .addOnFailureListener {
37                                         Toast.makeText(context: this@RegistroActivity, text: "Error al crear la cuenta", Toast.LENGTH_SHORT).show()
38                                     }
39                             }
40                         }
41                     }
42
43                     override fun onCancelled(error: DatabaseError) {
44                         Toast.makeText(context: this@RegistroActivity, text: "Error al verificar el correo", Toast.LENGTH_SHORT).show()
45                     }
46                 })
47             }
48         }
49     }
```

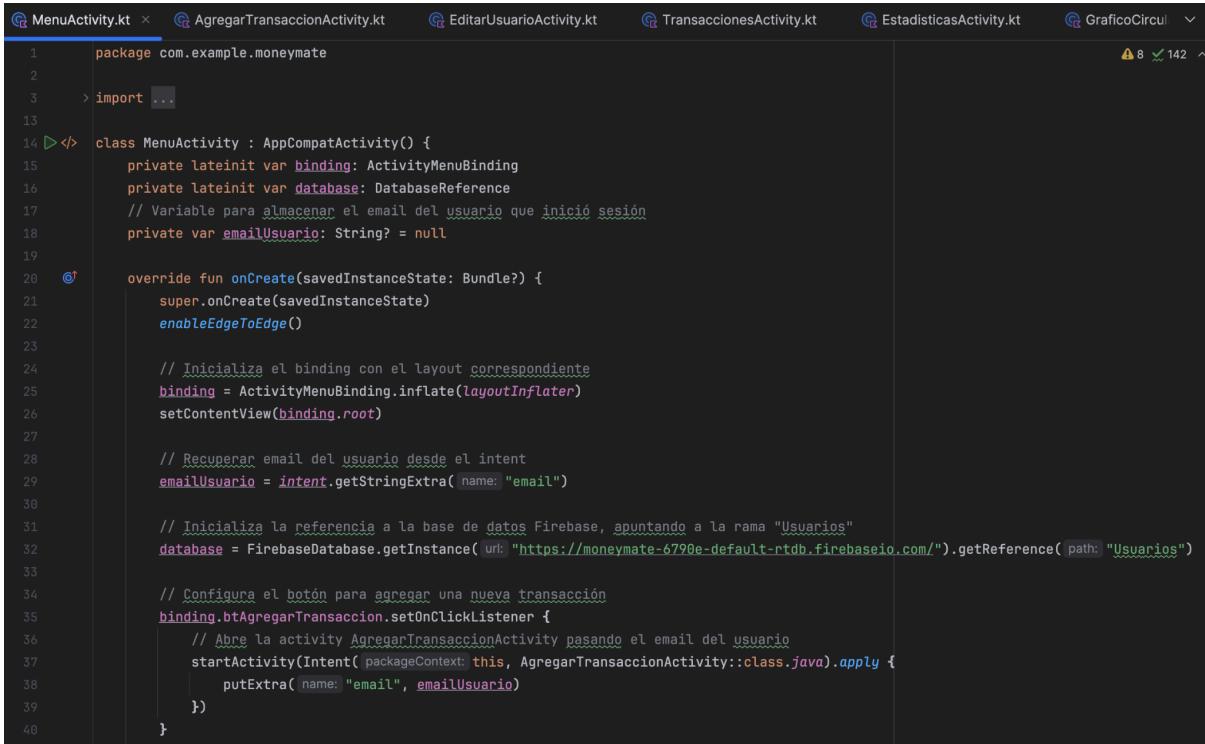
## Proyecto de Desarrollo de Aplicaciones Multiplataforma



```
RegistroActivity.kt  MainActivity.kt  AgregarTransaccionActivity.kt  EditarUsuarioActivity.kt  TransaccionesActivity.kt  EstadisticasActivity.kt  67 ^

11     class RegistroActivity : AppCompatActivity() {
15         override fun onCreate(savedInstanceState: Bundle?) {
16             binding.btCrearCuenta.setOnClickListener {
17                 .addSingleValueEventListener(object : ValueEventListener {
18
19                     @Override
20                     void onCancelled(DatabaseError error) {
21                         Toast.makeText(context: this@RegistroActivity, text: "Error al verificar el correo", Toast.LENGTH_SHORT).show()
22                     }
23
24                 })
25             }
26         }
27
28         // Función para validar que la contraseña tenga al menos 6 caracteres y un número
29         private fun contrasenaValida(password: String): Boolean {
30             val caracteres = password.length >= 6
31             val numero = password.any { it.isDigit() }
32
33             return caracteres && numero
34         }
35     }
36
37     67 ^
```

### 6.3. MenuActivity (Menú principal)



```
MenuActivity.kt  AgregarTransaccionActivity.kt  EditarUsuarioActivity.kt  TransaccionesActivity.kt  EstadisticasActivity.kt  GraficoCircularActivity.kt  8 142 ^

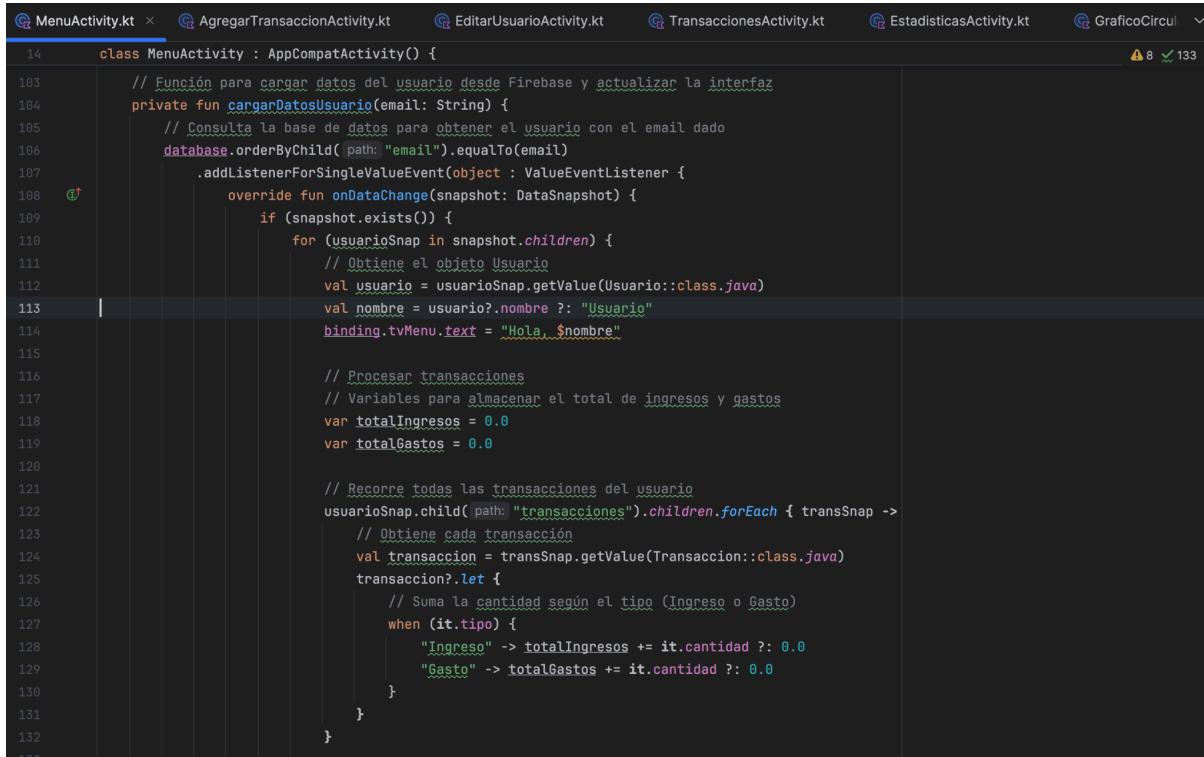
1 package com.example.moneymate
2
3 > import ...
4
5 > class MenuActivity : AppCompatActivity() {
6     private lateinit var binding: ActivityMenuBinding
7     private lateinit var database: DatabaseReference
8     // Variable para almacenar el email del usuario que inició sesión
9     private var emailUsuario: String? = null
10
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         enableEdgeToEdge()
14
15         // Inicializa el binding con el layout correspondiente
16         binding = ActivityMenuBinding.inflate(layoutInflater)
17         setContentView(binding.root)
18
19         // Recuperar email del usuario desde el intent
20         emailUsuario = intent.getStringExtra(name: "email")
21
22         // Inicializa la referencia a la base de datos Firebase, apuntando a la rama "Usuarios"
23         database = FirebaseDatabase.getInstance(url: "https://moneymate-6790e-default.firebaseio.com/").getReference(path: "Usuarios")
24
25         // Configura el botón para agregar una nueva transacción
26         binding.btAgregarTransaccion.setOnClickListener {
27             // Abre la actividad AgregarTransaccionActivity pasando el email del usuario
28             startActivity(Intent(packageContext: this, AgregarTransaccionActivity::class.java).apply {
29                 putExtra(name: "email", emailUsuario)
30             })
31         }
32     }
33
34     8 142 ^
```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

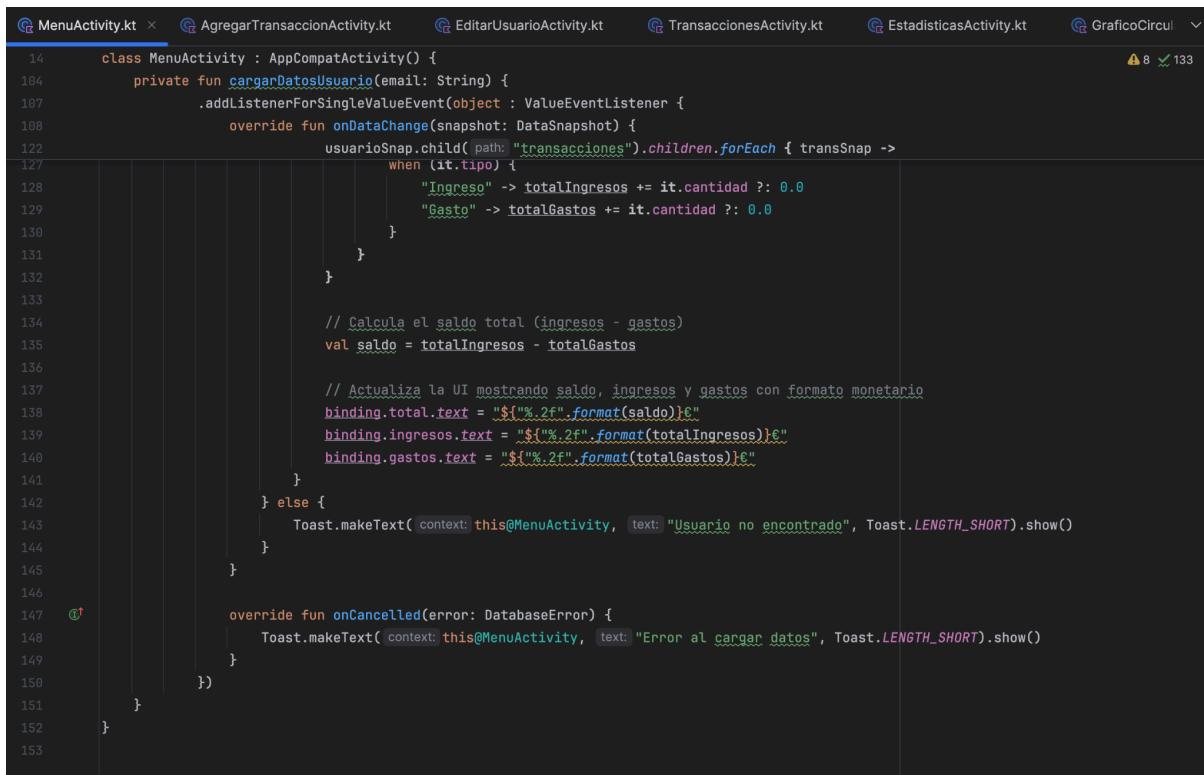
```
MenuActivity.kt x AgregarTransaccionActivity.kt EditarUsuarioActivity.kt TransaccionesActivity.kt EstadisticasActivity.kt GraficoCirculo.kt
14  class MenuActivity : AppCompatActivity() {
20      override fun onCreate(savedInstanceState: Bundle?) {
42          // Configura el ícono de usuario para que muestre un menú desplegable con opciones
43          binding.icono.setOnClickListener {
44              val popupMenu = PopupMenu(context: this, binding.icono)
45              // Infla el menú definido en recursos (menu_usuario) / Inflar = convertir XML en objetos reales de la UI
46              popupMenu.menuInflater.inflate(R.menu.menu_usuario, popupMenu.menu)
47
48              // Maneja las opciones seleccionadas en el menú
49              popupMenu.setOnMenuItemClickListener { item ->
50                  when (item.itemId) {
51                      // Abre la actividad EditarUsuarioActivity para editar los datos del usuario
52                      R.id.opcion_editar -> {
53                          startActivity(Intent(packageContext: this, EditarUsuarioActivity::class.java).apply {
54                              putExtra(name: "email", emailUsuario)
55                          })
56                          true
57                      }
58                      // Cierra sesión y finaliza esta actividad
59                      R.id.opcion_cerrar_sesion -> {
60                          startActivity(Intent(packageContext: this, MainActivity::class.java))
61                          finish()
62                          true
63                      }
64                      else -> false
65                  }
66              }
67              // Muestra el menú desplegable
68              popupMenu.show()
69          }
70      }
71  }
```

```
MenuActivity.kt x AgregarTransaccionActivity.kt EditarUsuarioActivity.kt TransaccionesActivity.kt EstadisticasActivity.kt GraficoCirculo.kt
14  class MenuActivity : AppCompatActivity() {
20      override fun onCreate(savedInstanceState: Bundle?) {
71          // Configura el listener para el menú de navegación inferior
72          binding.btNavegacion.setOnItemSelectedListener { item ->
73              when (item.itemId) {
74                  R.id.nav_menu -> true // Ya está en esta sección
75                  R.id.nav_transacciones -> {
76                      // Abre la actividad para ver transacciones
77                      startActivity(Intent(packageContext: this, TransaccionesActivity::class.java).apply {
78                          putExtra(name: "email", emailUsuario)
79                      })
80                      true
81                  }
82                  R.id.nav_estadisticas -> {
83                      // Abre la actividad de estadísticas
84                      startActivity(Intent(packageContext: this, EstadisticasActivity::class.java).apply {
85                          putExtra(name: "email", emailUsuario)
86                      })
87                      true
88                  }
89                  else -> false
90              }
91          }
92      }
93
94      // Método que se ejecuta cuando la actividad vuelve a estar en primer plano
95      override fun onResume() {
96          super.onResume()
97          // Si el email del usuario no es nulo, carga sus datos para actualizar la UI
98          emailUsuario?.let {
99              cargarDatosUsuario(it)
100         }
101     }
102 }
```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

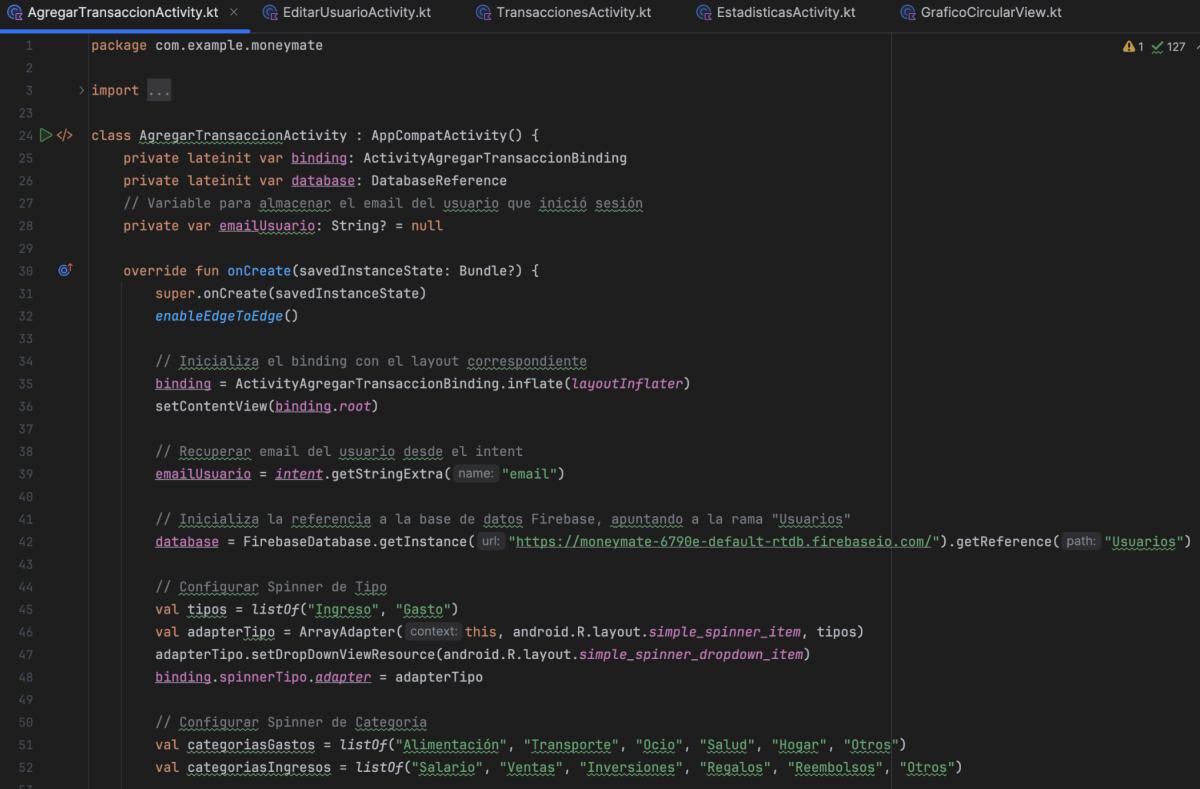


```
14 class MainActivity : AppCompatActivity() {
103     // Función para cargar datos del usuario desde Firebase y actualizar la interfaz
104     private fun cargarDatosUsuario(email: String) {
105         // Consulta la base de datos para obtener el usuario con el email dado
106         database.orderByChild("email").equalTo(email)
107             .addListenerForSingleValueEvent(object : ValueEventListener {
108                 override fun onDataChange(snapshot: DataSnapshot) {
109                     if (snapshot.exists()) {
110                         for (usuarioSnap in snapshot.children) {
111                             // Obtiene el objeto Usuario
112                             val usuario = usuarioSnap.getValue(Usuario::class.java)
113                             val nombre = usuario?.nombre ?: "Usuario"
114                             binding.tvMenu.text = "Hola, $nombre"
115
116                             // Procesar transacciones
117                             // Variables para almacenar el total de ingresos y gastos
118                             var totalIngresos = 0.0
119                             var totalGastos = 0.0
120
121                             // Recorre todas las transacciones del usuario
122                             usuarioSnap.child("transacciones").children.forEach { transSnap ->
123                                 // Obtiene cada transacción
124                                 val transaccion = transSnap.getValue(Transaccion::class.java)
125                                 transaccion?.let {
126                                     // Suma la cantidad según el tipo (Ingreso o Gasto)
127                                     when (it.tipo) {
128                                         "Ingreso" -> totalIngresos += it.cantidad ?: 0.0
129                                         "Gasto" -> totalGastos += it.cantidad ?: 0.0
130                                     }
131                                 }
132                             }
133
134                             // Calcula el saldo total (ingresos - gastos)
135                             val saldo = totalIngresos - totalGastos
136
137                             // Actualiza la UI mostrando saldo, ingresos y gastos con formato monetario
138                             binding.total.text = "${("%.2f".format(saldo))}€"
139                             binding.ingresos.text = "${("%.2f".format(totalIngresos))}€"
140                             binding.gastos.text = "${("%.2f".format(totalGastos))}€"
141
142                         } else {
143                             Toast.makeText(context, this@MainActivity, text: "Usuario no encontrado", Toast.LENGTH_SHORT).show()
144                         }
145                     }
146
147                 override fun onCancelled(error: DatabaseError) {
148                     Toast.makeText(context, this@MainActivity, text: "Error al cargar datos", Toast.LENGTH_SHORT).show()
149                 }
150             }
151         }
152     }
153 }
```



```
14 class MainActivity : AppCompatActivity() {
104     private fun cargarDatosUsuario(email: String) {
105         .addListenerForSingleValueEvent(object : ValueEventListener {
106             override fun onDataChange(snapshot: DataSnapshot) {
107                 usuarioSnap.child("transacciones").children.forEach { transSnap ->
108                     when (it.tipo) {
109                         "Ingreso" -> totalIngresos += it.cantidad ?: 0.0
110                         "Gasto" -> totalGastos += it.cantidad ?: 0.0
111                     }
112                 }
113
114                 // Calcula el saldo total (ingresos - gastos)
115                 val saldo = totalIngresos - totalGastos
116
117                 // Actualiza la UI mostrando saldo, ingresos y gastos con formato monetario
118                 binding.total.text = "${("%.2f".format(saldo))}€"
119                 binding.ingresos.text = "${("%.2f".format(totalIngresos))}€"
120                 binding.gastos.text = "${("%.2f".format(totalGastos))}€"
121
122             } else {
123                 Toast.makeText(context, this@MainActivity, text: "Usuario no encontrado", Toast.LENGTH_SHORT).show()
124             }
125         }
126
127         override fun onCancelled(error: DatabaseError) {
128             Toast.makeText(context, this@MainActivity, text: "Error al cargar datos", Toast.LENGTH_SHORT).show()
129         }
130     }
131 }
```

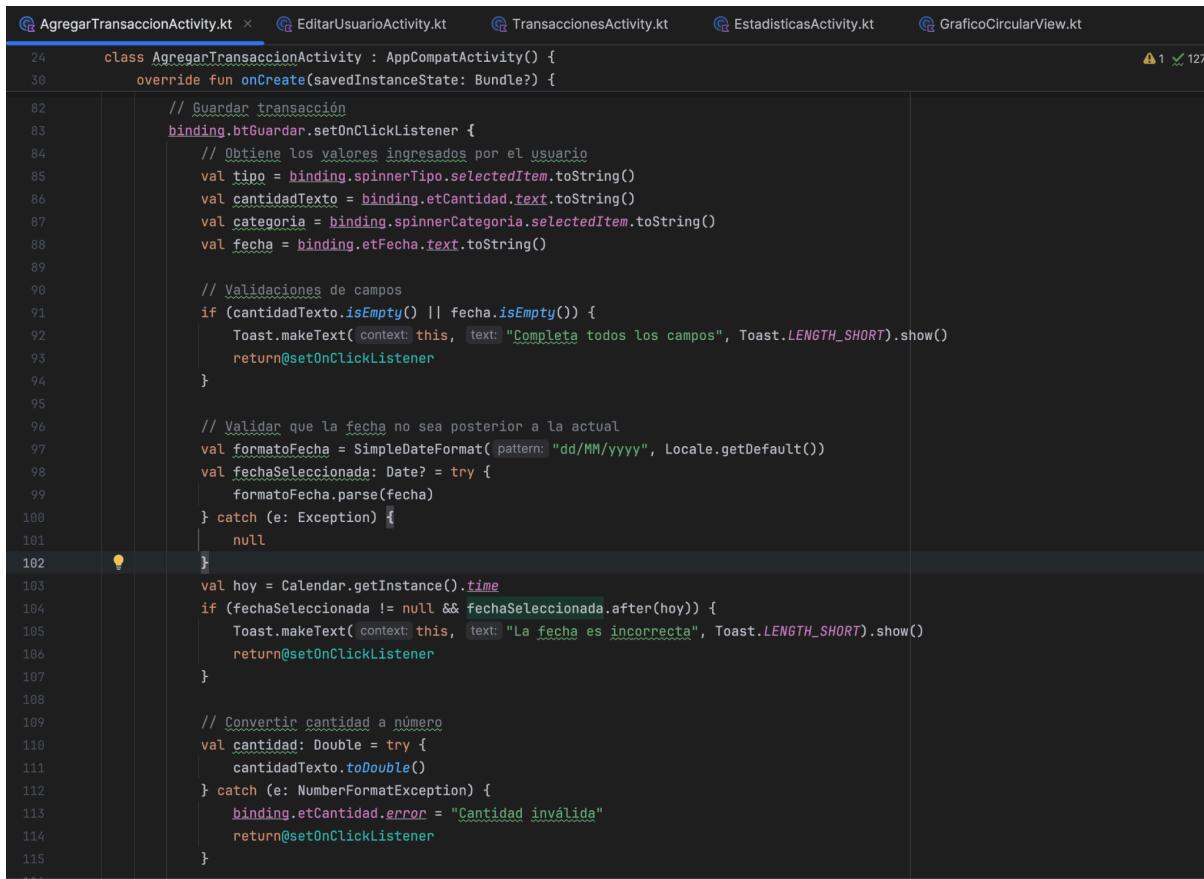
## 6.4. AgregarTransaccionActivity (Aregar ingreso o gasto)



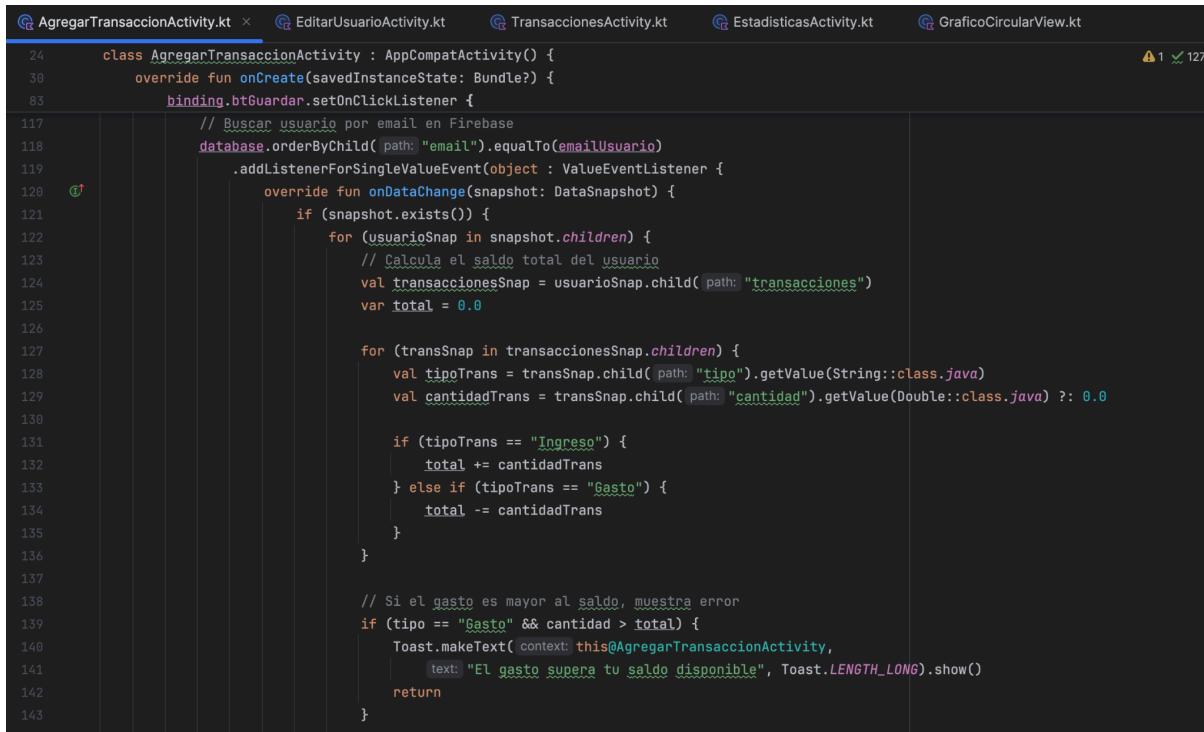
```

1 package com.example.moneymate
2
3 > import ...
4
5 <> class AgregarTransaccionActivity : AppCompatActivity() {
6     private lateinit var binding: ActivityAgregarTransaccionBinding
7     private lateinit var database: DatabaseReference
8     // Variable para almacenar el email del usuario que inició sesión
9     private var emailUsuario: String? = null
10
11    override fun onCreate(savedInstanceState: Bundle?) {
12        super.onCreate(savedInstanceState)
13        enableEdgeToEdge()
14
15        // Inicializa el binding con el layout correspondiente
16        binding = ActivityAgregarTransaccionBinding.inflate(layoutInflater)
17        setContentView(binding.root)
18
19        // Recuperar email del usuario desde el intent
20        emailUsuario = intent.getStringExtra(name: "email")
21
22        // Inicializa la referencia a la base de datos Firebase, apuntando a la rama "Usuarios"
23        database = FirebaseDatabase.getInstance(url: "https://moneymate-6790e-default.firebaseio.com/").getReference(path: "Usuarios")
24
25        // Configurar Spinner de Tipo
26        val tipos = listOf("Ingreso", "Gasto")
27        val adapterTipo = ArrayAdapter(context: this, android.R.layout.simple_spinner_item, tipos)
28        adapterTipo.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
29        binding.spinnerTipo.adapter = adapterTipo
30
31        // Configurar Spinner de Categoría
32        val categoriasGastos = listOf("Alimentación", "Transporte", "Ocio", "Salud", "Hogar", "Otros")
33        val categoriasIngresos = listOf("Salario", "Ventas", "Inversiones", "Regalos", "Reembolsos", "Otros")
34
35    }
36
37    override fun onCreateOptionsMenu(menu: Menu): Boolean {
38        menuInflater.inflate(R.menu.menu_agregar_transaccion, menu)
39        return true
40    }
41
42    override fun onOptionsItemSelected(item: MenuItem): Boolean {
43        when (item.itemId) {
44            R.id.action_guardar -> {
45                // Guardar transacción en la base de datos
46                val transaction = Transaction(
47                    tipo = binding.spinnerTipo.selectedItem.toString(),
48                    categoria = binding.spinnerCategoria.selectedItem.toString(),
49                    fecha = binding.etFecha.text.toString(),
50                    cantidad = binding.etCantidad.text.toString()
51                )
52                database.push().setValue(transaction)
53            }
54        }
55        return super.onOptionsItemSelected(item)
56    }
57
58    companion object {
59        const val name = "email"
60    }
61
62    data class Transaction(
63        val tipo: String,
64        val categoria: String,
65        val fecha: String,
66        val cantidad: String
67    )
68
69    data class Usuario(
70        val nombre: String,
71        val apellido: String,
72        val email: String
73    )
74
75    data class Estadistica(
76        val categoria: String,
77        val cantidad: String
78    )
79
80    data class GraficoCircularView(
81        val titulo: String,
82        val datos: List<Estadistica>
83    )
84
85    data class DateRange(
86        val inicio: Date,
87        val fin: Date
88    )
89
90    data class Reporte(
91        val fecha: Date,
92        val ingresos: Double,
93        val gastos: Double
94    )
95
96    data class ReporteResumen(
97        val fecha: Date,
98        val total: Double
99    )
100
101    data class ReportePorCategoría(
102        val categoria: String,
103        val cantidad: Double
104    )
105
106    data class ReportePorMes(
107        val mes: String,
108        val cantidad: Double
109    )
110
111    data class ReportePorAño(
112        val año: String,
113        val cantidad: Double
114    )
115
116    data class ReportePorSemana(
117        val semana: String,
118        val cantidad: Double
119    )
120
121    data class ReportePorDía(
122        val día: String,
123        val cantidad: Double
124    )
125
126    data class ReportePorHora(
127        val hora: String,
128        val cantidad: Double
129    )
130
131    data class ReportePorMinuto(
132        val minuto: String,
133        val cantidad: Double
134    )
135
136    data class ReportePorSegundo(
137        val segundo: String,
138        val cantidad: Double
139    )
140
141    data class ReportePorMilisegundo(
142        val milisegundo: String,
143        val cantidad: Double
144    )
145
146    data class ReportePorMicrosegundo(
147        val microsegundo: String,
148        val cantidad: Double
149    )
150
151    data class ReportePorNanosegundo(
152        val nanosegundo: String,
153        val cantidad: Double
154    )
155
156    data class ReportePorPicosegundo(
157        val picosegundo: String,
158        val cantidad: Double
159    )
160
161    data class ReportePorFemosegundo(
162        val femosegundo: String,
163        val cantidad: Double
164    )
165
166    data class ReportePorPetosegundo(
167        val petosegundo: String,
168        val cantidad: Double
169    )
170
171    data class ReportePorExasegundo(
172        val exasegundo: String,
173        val cantidad: Double
174    )
175
176    data class ReportePorZetasegundo(
177        val zetasegundo: String,
178        val cantidad: Double
179    )
180
181    data class ReportePorYotasegundo(
182        val yotasegundo: String,
183        val cantidad: Double
184    )
185
186    data class ReportePorZettasegundo(
187        val zettasegundo: String,
188        val cantidad: Double
189    )
190
191    data class ReportePorExottasegundo(
192        val exottasegundo: String,
193        val cantidad: Double
194    )
195
196    data class ReportePorZettottasegundo(
197        val zettottasegundo: String,
198        val cantidad: Double
199    )
199
200    data class ReportePorExottottasegundo(
201        val exottottasegundo: String,
202        val cantidad: Double
203    )
204
205    data class ReportePorZettottottasegundo(
206        val zettottottasegundo: String,
207        val cantidad: Double
208    )
209
210    data class ReportePorExottottottasegundo(
211        val exottottottasegundo: String,
212        val cantidad: Double
213    )
214
215    data class ReportePorZettottottottasegundo(
216        val zettottottottasegundo: String,
217        val cantidad: Double
218    )
219
220    data class ReportePorExottottottottasegundo(
221        val exottottottottasegundo: String,
222        val cantidad: Double
223    )
224
225    data class ReportePorZettottottottottasegundo(
226        val zettottottottottasegundo: String,
227        val cantidad: Double
228    )
229
230    data class ReportePorExottottottottottasegundo(
231        val exottottottottottasegundo: String,
232        val cantidad: Double
233    )
234
235    data class ReportePorZettottottottottottasegundo(
236        val zettottottottottottasegundo: String,
237        val cantidad: Double
238    )
239
240    data class ReportePorExottottottottottottasegundo(
241        val exottottottottottottasegundo: String,
242        val cantidad: Double
243    )
244
245    data class ReportePorZettottottottottottottasegundo(
246        val zettottottottottottottasegundo: String,
247        val cantidad: Double
248    )
249
250    data class ReportePorExottottottottottottottasegundo(
251        val exottottottottottottottasegundo: String,
252        val cantidad: Double
253    )
254
255    data class ReportePorZettottottottottottottottasegundo(
256        val zettottottottottottottottasegundo: String,
257        val cantidad: Double
258    )
259
260    data class ReportePorExottottottottottottottottasegundo(
261        val exottottottottottottottottasegundo: String,
262        val cantidad: Double
263    )
264
265    data class ReportePorZettottottottottottottottottasegundo(
266        val zettottottottottottottottottasegundo: String,
267        val cantidad: Double
268    )
269
270    data class ReportePorExottottottottottottottottottasegundo(
271        val exottottottottottottottottottasegundo: String,
272        val cantidad: Double
273    )
274
275    data class ReportePorZettottottottottottottottottottasegundo(
276        val zettottottottottottottottottottasegundo: String,
277        val cantidad: Double
278    )
279
280    data class ReportePorExottottottottottottottottottottasegundo(
281        val exottottottottottottottottottottasegundo: String,
282        val cantidad: Double
283    )
284
285    data class ReportePorZettottottottottottottottottottottasegundo(
286        val zettottottottottottottottottottottasegundo: String,
287        val cantidad: Double
288    )
289
290    data class ReportePorExottottottottottottottottottottottasegundo(
291        val exottottottottottottottottottottottasegundo: String,
292        val cantidad: Double
293    )
294
295    data class ReportePorZettottottottottottottottottottottottasegundo(
296        val zettottottottottottottottottottottottasegundo: String,
297        val cantidad: Double
298    )
299
300    data class ReportePorExottottottottottottottottottottottottasegundo(
301        val exottottottottottottottottottottottottasegundo: String,
302        val cantidad: Double
303    )
304
305    data class ReportePorZettottottottottottottottottottottottottasegundo(
306        val zettottottottottottottottottottottottottasegundo: String,
307        val cantidad: Double
308    )
309
310    data class ReportePorExottottottottottottottottottottottottottasegundo(
311        val exottottottottottottottottottottottottottasegundo: String,
312        val cantidad: Double
313    )
314
315    data class ReportePorZettottottottottottottottottottottottottottasegundo(
316        val zettottottottottottottottottottottottottottasegundo: String,
317        val cantidad: Double
318    )
319
320    data class ReportePorExottottottottottottottottottottottottottottasegundo(
321        val exottottottottottottottottottottottottottottasegundo: String,
322        val cantidad: Double
323    )
324
325    data class ReportePorZettottottottottottottottottottottottottottottasegundo(
326        val zettottottottottottottottottottottottottottottasegundo: String,
327        val cantidad: Double
328    )
329
330    data class ReportePorExottottottottottottottottottottottottottottottasegundo(
331        val exottottottottottottottottottottottottottottottasegundo: String,
332        val cantidad: Double
333    )
334
335    data class ReportePorZettottottottottottottottottottottottottottottottasegundo(
336        val zettottottottottottottottottottottottottottottottasegundo: String,
337        val cantidad: Double
338    )
339
340    data class ReportePorExottottottottottottottottottottottottottottottottasegundo(
341        val exottottottottottottottottottottottottottottottottasegundo: String,
342        val cantidad: Double
343    )
344
345    data class ReportePorZettottottottottottottottottottottottottottottottottasegundo(
346        val zettottottottottottottottottottottottottottottottottasegundo: String,
347        val cantidad: Double
348    )
349
350    data class ReportePorExottottottottottottottottottottottottottottottottottasegundo(
351        val exottottottottottottottottottottottottottottottottasegundo: String,
352        val cantidad: Double
353    )
354
355    data class ReportePorZettottottottottottottottottottottottottottottottottottasegundo(
356        val zettottottottottottottottottottottottottottottottottasegundo: String,
357        val cantidad: Double
358    )
359
360    data class ReportePorExottottottottottottottottottottottottottottottottottasegundo(
361        val exottottottottottottottottottottottottottottottasegundo: String,
362        val cantidad: Double
363    )
364
365    data class ReportePorZettottottottottottottottottottottottottottottottottottasegundo(
366        val zettottottottottottottottottottottottottottottasegundo: String,
367        val cantidad: Double
368    )
369
370    data class ReportePorExottottottottottottottottottottottottottottottottottasegundo(
371        val exottottottottottottottottottottottottottasegundo: String,
372        val cantidad: Double
373    )
374
375    data class ReportePorZettottottottottottottottottottottottottottottottottasegundo(
376        val zettottottottottottottottottottottottottasegundo: String,
377        val cantidad: Double
378    )
379
380    data class ReportePorExottottottottottottottottottottottottottottottottasegundo(
381        val exottottottottottottottottottottottasegundo: String,
382        val cantidad: Double
383    )
384
385    data class ReportePorZettottottottottottottottottottottottottasegundo(
386        val zettottottottottottottottottottottasegundo: String,
387        val cantidad: Double
388    )
389
390    data class ReportePorExottottottottottottottottottottottasegundo(
391        val exottottottottottottottottottasegundo: String,
392        val cantidad: Double
393    )
394
395    data class ReportePorZettottottottottottottottottasegundo(
396        val zettottottottottottottottasegundo: String,
397        val cantidad: Double
398    )
399
400    data class ReportePorExottottottottottottasegundo(
401        val exottottottottottasegundo: String,
402        val cantidad: Double
403    )
404
405    data class ReportePorZettottottottasegundo(
406        val zettottottottasegundo: String,
407        val cantidad: Double
408    )
409
410    data class ReportePorExottottasegundo(
411        val exottottasegundo: String,
412        val cantidad: Double
413    )
414
415    data class ReportePorZettottasegundo(
416        val zettottasegundo: String,
417        val cantidad: Double
418    )
419
420    data class ReportePorExottasegundo(
421        val exottasegundo: String,
422        val cantidad: Double
423    )
424
425    data class ReportePorZettasegundo(
426        val zettasegundo: String,
427        val cantidad: Double
428    )
429
430    data class ReportePorExottasegundo(
431        val exottasegundo: String,
432        val cantidad: Double
433    )
434
435    data class ReportePorZettasegundo(
436        val zettasegundo: String,
437        val cantidad: Double
438    )
439
440    data class ReportePorExottasegundo(
441        val exottasegundo: String,
442        val cantidad: Double
443    )
444
445    data class ReportePorZettasegundo(
446        val zettasegundo: String,
447        val cantidad: Double
448    )
449
450    data class ReportePorExottasegundo(
451        val exottasegundo: String,
452        val cantidad: Double
453    )
454
455    data class ReportePorZettasegundo(
456        val zettasegundo: String,
457        val cantidad: Double
458    )
459
460    data class ReportePorExottasegundo(
461        val exottasegundo: String,
462        val cantidad: Double
463    )
464
465    data class ReportePorZettasegundo(
466        val zettasegundo: String,
467        val cantidad: Double
468    )
469
470    data class ReportePorExottasegundo(
471        val exottasegundo: String,
472        val cantidad: Double
473    )
474
475    data class ReportePorZettasegundo(
476        val zettasegundo: String,
477        val cantidad: Double
478    )
479
480    data class ReportePorExottasegundo(
481        val exottasegundo: String,
482        val cantidad: Double
483    )
484
485    data class ReportePorZettasegundo(
486        val zettasegundo: String,
487        val cantidad: Double
488    )
489
490    data class ReportePorExottasegundo(
491        val exottasegundo: String,
492        val cantidad: Double
493    )
494
495    data class ReportePorZettasegundo(
496        val zettasegundo: String,
497        val cantidad: Double
498    )
499
500    data class ReportePorExottasegundo(
501        val exottasegundo: String,
502        val cantidad: Double
503    )
504
505    data class ReportePorZettasegundo(
506        val zettasegundo: String,
507        val cantidad: Double
508    )
509
510    data class ReportePorExottasegundo(
511        val exottasegundo: String,
512        val cantidad: Double
513    )
514
515    data class ReportePorZettasegundo(
516        val zettasegundo: String,
517        val cantidad: Double
518    )
519
520    data class ReportePorExottasegundo(
521        val exottasegundo: String,
522        val cantidad: Double
523    )
524
525    data class ReportePorZettasegundo(
526        val zettasegundo: String,
527        val cantidad: Double
528    )
529
530    data class ReportePorExottasegundo(
531        val exottasegundo: String,
532        val cantidad: Double
533    )
534
535    data class ReportePorZettasegundo(
536        val zettasegundo: String,
537        val cantidad: Double
538    )
539
540    data class ReportePorExottasegundo(
541        val exottasegundo: String,
542        val cantidad: Double
543    )
544
545    data class ReportePorZettasegundo(
546        val zettasegundo: String,
547        val cantidad: Double
548    )
549
550    data class ReportePorExottasegundo(
551        val exottasegundo: String,
552        val cantidad: Double
553    )
554
555    data class ReportePorZettasegundo(
556        val zettasegundo: String,
557        val cantidad: Double
558    )
559
560    data class ReportePorExottasegundo(
561        val exottasegundo: String,
562        val cantidad: Double
563    )
564
565    data class ReportePorZettasegundo(
566        val zettasegundo: String,
567        val cantidad: Double
568    )
569
570    data class ReportePorExottasegundo(
571        val exottasegundo: String,
572        val cantidad: Double
573    )
574
575    data class ReportePorZettasegundo(
576        val zettasegundo: String,
577        val cantidad: Double
578    )
579
580    data class ReportePorExottasegundo(
581        val exottasegundo: String,
582        val cantidad: Double
583    )
584
585    data class ReportePorZettasegundo(
586        val zettasegundo: String,
587        val cantidad: Double
588    )
589
590    data class ReportePorExottasegundo(
591        val exottasegundo: String,
592        val cantidad: Double
593    )
594
595    data class ReportePorZettasegundo(
596        val zettasegundo: String,
597        val cantidad: Double
598    )
599
599
600    data class ReportePorExottasegundo(
601        val exottasegundo: String,
602        val cantidad: Double
603    )
604
605    data class ReportePorZettasegundo(
606        val zettasegundo: String,
607        val cantidad: Double
608    )
609
610    data class ReportePorExottasegundo(
611        val exottasegundo: String,
612        val cantidad: Double
613    )
614
615    data class ReportePorZettasegundo(
616        val zettasegundo: String,
617        val cantidad: Double
618    )
619
620    data class ReportePorExottasegundo(
621        val exottasegundo: String,
622        val cantidad: Double
623    )
624
625    data class ReportePorZettasegundo(
626        val zettasegundo: String,
627        val cantidad: Double
628    )
629
630    data class ReportePorExottasegundo(
631        val exottasegundo: String,
632        val cantidad: Double
633    )
634
635    data class ReportePorZettasegundo(
636        val zettasegundo: String,
637        val cantidad: Double
638    )
639
640    data class ReportePorExottasegundo(
641        val exottasegundo: String,
642        val cantidad: Double
643    )
644
645    data class ReportePorZettasegundo(
646        val zettasegundo: String,
647        val cantidad: Double
648    )
649
650    data class ReportePorExottasegundo(
651        val exottasegundo: String,
652        val cantidad: Double
653    )
654
655    data class ReportePorZettasegundo(
656        val zettasegundo: String,
657        val cantidad: Double
658    )
659
660    data class ReportePorExottasegundo(
661        val exottasegundo: String,
662        val cantidad: Double
663    )
664
665    data class ReportePorZettasegundo(
666        val zettasegundo: String,
667        val cantidad: Double
668    )
669
670    data class ReportePorExottasegundo(
671        val exottasegundo: String,
672        val cantidad: Double
673    )
674
675    data class ReportePorZettasegundo(
676        val zettasegundo: String,
677        val cantidad: Double
678    )
679
680    data class ReportePorExottasegundo(
681        val exottasegundo: String,
682        val cantidad: Double
683    )
684
685    data class ReportePorZettasegundo(
686        val zettasegundo: String,
687        val cantidad: Double
688    )
689
690    data class ReportePorExottasegundo(
691        val exottasegundo: String,
692        val cantidad: Double
693    )
694
695    data class ReportePorZettasegundo(
696        val zettasegundo: String,
697        val cantidad: Double
698    )
699
699
700    data class ReportePorExottasegundo(
701        val exottasegundo: String,
702        val cantidad: Double
703    )
704
705    data class ReportePorZettasegundo(
706        val zettasegundo: String,
707        val cantidad: Double
708    )
709
710    data class ReportePorExottasegundo(
711        val exottasegundo: String,
712        val cantidad: Double
713    )
714
715    data class ReportePorZettasegundo(
716        val zettasegundo: String,
717        val cantidad: Double
718    )
719
720    data class ReportePorExottasegundo(
721        val exottasegundo: String,
722        val cantidad: Double
723    )
724
725    data class ReportePorZettasegundo(
726        val zettasegundo: String,
727        val cantidad: Double
728    )
729
730    data class ReportePorExottasegundo(
731        val exottasegundo: String,
732        val cantidad: Double
733    )
734
735    data class ReportePorZettasegundo(
736        val zettasegundo: String,
737        val cantidad: Double
738    )
739
740    data class ReportePorExottasegundo(
741        val exottasegundo: String,
742        val cantidad: Double
743    )
744
745    data class ReportePorZettasegundo(
746        val zettasegundo: String,
747        val cantidad: Double
748    )
749
750    data class ReportePorExottasegundo(
751        val exottasegundo: String,
752        val cantidad: Double
753    )
754
755    data class ReportePorZettasegundo(
756        val zettasegundo: String,
757        val cantidad: Double
758    )
759
760    data class ReportePorExottasegundo(
761        val exottasegundo: String,
762        val cantidad: Double
763    )
764
765    data class ReportePorZettasegundo(
766        val zettasegundo: String,
767        val cantidad: Double
768    )
769
770    data class ReportePorExottasegundo(
771        val exottasegundo: String,
772        val cantidad: Double
773    )
774
775    data class ReportePorZettasegundo(
776        val zettasegundo: String,
777        val cantidad: Double
778    )
779
780    data class ReportePorExottasegundo(
781        val exottasegundo: String,
782        val cantidad: Double
783    )
784
785    data class ReportePorZettasegundo(
786        val zettasegundo: String,
787        val cantidad: Double
788    )
789
790    data class ReportePorExottasegundo(
791        val exottasegundo: String,
792        val cantidad: Double
793    )
794
795    data class ReportePorZettasegundo(
796        val zettasegundo: String,
797        val cantidad: Double
798    )
799
799
800    data class ReportePorExottasegundo(
801        val exottasegundo: String,
802        val cantidad: Double
803    )
804
805    data class ReportePorZettasegundo(
806        val zettasegundo: String,
807        val cantidad: Double
808    )
809
810    data class ReportePorExottasegundo(
811        val exottasegundo: String,
812        val cantidad: Double
813    )
814
815    data class ReportePorZettasegundo(
816        val zettasegundo: String,
817        val cantidad: Double
818    )
819
820    data class ReportePorExottasegundo(
821        val exottasegundo: String,
822        val cantidad: Double
823    )
824
825    data class ReportePorZettasegundo(
826        val zettasegundo: String,
827        val cantidad: Double
828    )
829
830    data class ReportePorExottasegundo(
831        val exottasegundo: String,
832        val cantidad: Double
833    )
834
835    data class ReportePorZettasegundo(
836        val zettasegundo: String,
837        val cantidad: Double
838    )
839
840    data class ReportePorExottasegundo(
841        val exottasegundo: String,
842        val cantidad: Double
843    )
844
845    data class ReportePorZettasegundo(
846        val zettasegundo: String,
847        val cantidad: Double
848    )
849
849
850    data class ReportePorExottasegundo(
851        val exottasegundo: String,
852        val cantidad: Double
853    )
854
855    data class ReportePorZettasegundo(
856        val zettasegundo: String,
857        val cantidad: Double
858    )
859
860    data class ReportePorExottasegundo(
861        val exottasegundo: String,
862        val cantidad: Double
863    )
864
865    data class ReportePorZettasegundo(
866        val zettasegundo: String,
867        val cantidad: Double
868    )
869
870    data class ReportePorExottasegundo(
871        val exottasegundo: String,
872        val cantidad: Double
873    )
874
875    data class ReportePorZettasegundo(
876        val zettasegundo: String,
877        val cantidad: Double
878    )
879
879
880    data class ReportePorExottasegundo(
881        val exottasegundo: String,
882        val cantidad: Double
883    )
884
885    data class ReportePorZettasegundo(
886        val zettasegundo: String,
887        val cantidad: Double
888    )
889
890    data class ReportePorExottasegundo(
891        val exottasegundo: String,
892        val cantidad: Double
893    )
894
895    data class ReportePorZettasegundo(
896        val zettasegundo: String,
897        val cantidad: Double
898    )
899
899
900    data class ReportePorExottasegundo(
901        val exottasegundo: String,
902        val cantidad: Double
903    )
904
905    data class ReportePorZettasegundo(
906        val zettasegundo: String,
907        val cantidad: Double
908    )
909
910    data class ReportePorExottasegundo(
911        val exottasegundo: String,
912        val cantidad: Double
913    )
914
915    data class ReportePorZettasegundo(
916        val zettasegundo: String,
917        val cantidad: Double
918    )
919
920    data class ReportePorExottasegundo(
921        val exottasegundo: String,
922        val cantidad: Double
923    )
924
925    data class ReportePorZettasegundo(
926        val zettasegundo: String,
927        val cantidad: Double
928    )
929
930    data class ReportePorExottasegundo(
931        val exottasegundo: String,
932        val cantidad: Double
933    )
934
935    data class ReportePorZettasegundo(
936        val zettasegundo: String,
937        val cantidad: Double
938    )
939
940    data class ReportePorExottasegundo(
941        val exottasegundo: String,
942        val cantidad: Double
943    )
944
945    data class ReportePorZettasegundo(
946        val zettasegundo: String,
947        val cantidad: Double
948    )
949
949
950    data class ReportePorExottasegundo(
951        val exottasegundo: String,
952        val cantidad: Double
953    )
954
955    data class ReportePorZettasegundo(
956        val zettasegundo: String,
957        val cantidad: Double
958    )
959
960    data class ReportePorExottasegundo(
961        val exottasegundo: String,
962        val cantidad: Double
963    )
964
965    data class ReportePorZettasegundo(
966        val zettasegundo: String,
967        val cantidad: Double
968    )
969
970    data class ReportePorExottasegundo(
971        val exottasegundo: String,
972        val cantidad: Double
973    )
974
975    data class ReportePorZettasegundo(
976        val zettasegundo: String,
977        val cantidad: Double
978    )
979
979
980    data class ReportePorExottasegundo(
981        val exottasegundo: String,
982        val cantidad: Double
983    )
984
985    data class ReportePorZettasegundo(
986        val zettasegundo: String,
987        val cantidad: Double
988    )
989
990    data class ReportePorExottasegundo(
991        val exottasegundo: String,
992        val cantidad: Double
993    )
994
995    data class ReportePorZettasegundo(
996        val zettasegundo: String,
997        val cantidad: Double
998    )
999
999
1000    data class ReportePorExottasegundo(
1001        val exottasegundo: String,
1002        val cantidad: Double
1003    )
1004
1005    data class ReportePorZettasegundo(
1006        val zettasegundo: String,
1007        val cantidad: Double
1008    )
1009
1010    data class ReportePorExottasegundo(
1011        val exottasegundo: String,
1012        val cantidad: Double
1013    )
1014
1015    data class ReportePorZettasegundo(
1016        val zettasegundo: String,
1017        val cantidad: Double
1018    )
1019
1020    data class ReportePorExottasegundo(
1021        val exottasegundo: String,
1022        val cantidad: Double
1023    )
1024
1025    data class ReportePorZettasegundo(
1026        val zettasegundo: String,
1027        val cantidad: Double
1028    )
1029
1029
1030    data class ReportePorExottasegundo(
1031        val exottasegundo: String,
1032        val cantidad: Double
1033    )
1034
1035    data class ReportePorZettasegundo(
1036        val zettasegundo: String,
1037        val cantidad: Double
1038    )
1039
1040    data class ReportePorExottasegundo(
1041        val exottasegundo: String,
1042        val cantidad: Double
1043    )
1044
1045    data class ReportePorZettasegundo(
1046        val zettasegundo: String,
1047        val cantidad: Double
1048    )
1049
1049
1050    data class ReportePorExottasegundo(
1051        val exottasegundo: String,
1052        val cantidad: Double
1053    )
1054
1055    data class ReportePorZettasegundo(
1056        val zettasegundo: String,
1057        val cantidad: Double
1058    )
1059
1060    data class ReportePorExottasegundo(
1061        val exottasegundo: String,
1062        val cantidad: Double
1063    )
1064
1065    data class ReportePorZettasegundo(
1066        val zettasegundo: String,
1067        val cantidad: Double
1068    )
1069
1069
1070    data class ReportePorExottasegundo(
1071        val exottasegundo: String,
1072        val cantidad: Double
1073    )
1074
1075    data class ReportePorZettasegundo(
1076        val zettasegundo: String,
1077        val cantidad: Double
1078    )
1079
1080    data class ReportePorExottasegundo(
1081        val exottasegundo: String,
1082        val cantidad: Double
1083    )
1084
1085    data class ReportePorZettasegundo(
1086        val zettasegundo: String,
1087        val cantidad: Double
1088    )
1089
1089
1090    data class ReportePorExottasegundo(
1091        val exottasegundo: String,
1092        val cantidad: Double
1093    )
1094
1095    data class ReportePorZettasegundo(
1096        val zettasegundo: String,
1097        val cantidad: Double
1098    )
1099
1100    data class ReportePorExottasegundo(
110
```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

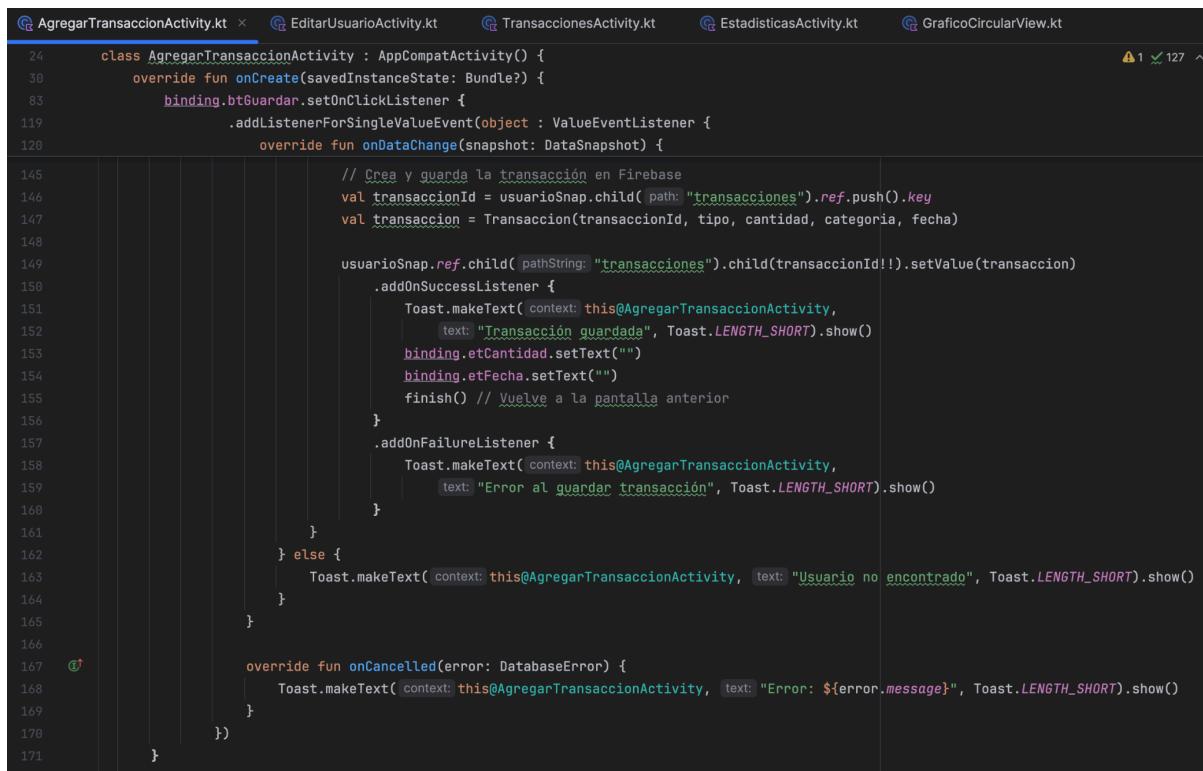


```
24  class AgregarTransaccionActivity : AppCompatActivity() {
30      override fun onCreate(savedInstanceState: Bundle?) {
31          binding.btGuardar.setOnClickListener {
32              // Guardar transacción
33              binding.etGuardado.setClickListener {
34                  // Obtiene los valores ingresados por el usuario
35                  val tipo = binding.spinnerTipo.selectedItem.toString()
36                  val cantidadTexto = binding.etCantidad.text.toString()
37                  val categoria = binding.spinnerCategoria.selectedItem.toString()
38                  val fecha = binding.etFecha.text.toString()
39
40                  // Validaciones de campos
41                  if (cantidadTexto.isEmpty() || fecha.isEmpty()) {
42                      Toast.makeText(context, "Completa todos los campos", Toast.LENGTH_SHORT).show()
43                      return@setOnClickListener
44                  }
45
46                  // Validar que la fecha no sea posterior a la actual
47                  val formatoFecha = SimpleDateFormat(pattern: "dd/MM/yyyy", Locale.getDefault())
48                  val fechaSeleccionada: Date? = try {
49                      formatoFecha.parse(fecha)
50                  } catch (e: Exception) {
51                      null
52                  }
53
54                  val hoy = Calendar.getInstance().time
55                  if (fechaSeleccionada != null && fechaSeleccionada.after(hoy)) {
56                      Toast.makeText(context, "La fecha es incorrecta", Toast.LENGTH_SHORT).show()
57                      return@setOnClickListener
58                  }
59
60                  // Convertir cantidad a número
61                  val cantidad: Double = try {
62                      cantidadTexto.toDouble()
63                  } catch (e: NumberFormatException) {
64                      binding.etCantidad.error = "Cantidad inválida"
65                      return@setOnClickListener
66                  }
67
68              }
69          }
70      }
71  }
```

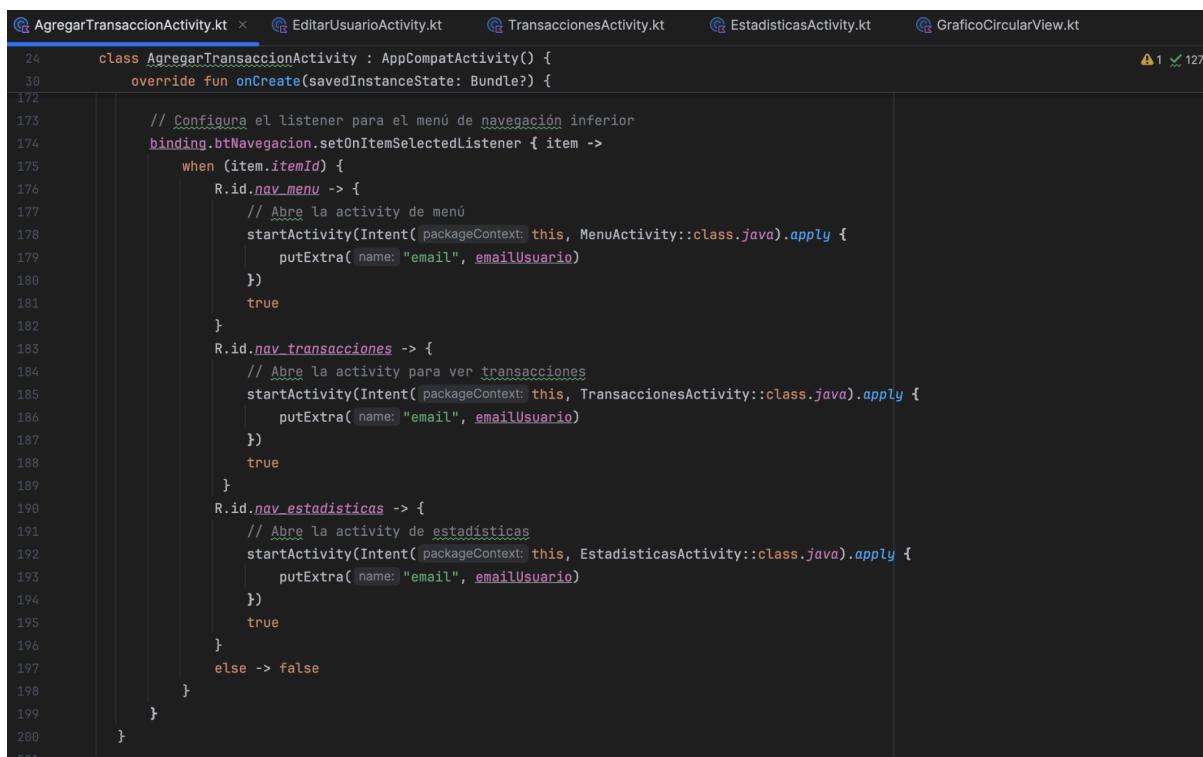


```
24  class AgregarTransaccionActivity : AppCompatActivity() {
30      override fun onCreate(savedInstanceState: Bundle?) {
31          binding.btGuardar.setOnClickListener {
32              // Buscar usuario por email en Firebase
33              database.orderByChild(path: "email").equalTo(emailUsuario)
34                  .addSingleValueEvent(object : ValueEventListener {
35                      override fun onDataChange(snapshot: DataSnapshot) {
36                          if (snapshot.exists()) {
37                              for (usuarioSnap in snapshot.children) {
38                                  // Calcula el saldo total del usuario
39                                  val transaccionesSnap = usuarioSnap.child(path: "transacciones")
40                                  var total = 0.0
41
42                                  for (transSnap in transaccionesSnap.children) {
43                                      val tipoTrans = transSnap.child(path: "tipo").getValue(String::class.java)
44                                      val cantidadTrans = transSnap.child(path: "cantidad").getValue(Double::class.java) ?: 0.0
45
46                                      if (tipoTrans == "Ingreso") {
47                                          total += cantidadTrans
48                                      } else if (tipoTrans == "Gasto") {
49                                          total -= cantidadTrans
50                                      }
51
52                                  }
53
54                                  // Si el gasto es mayor al saldo, muestra error
55                                  if (tipo == "Gasto" && cantidad > total) {
56                                      Toast.makeText(context, "El gasto supera tu saldo disponible", Toast.LENGTH_LONG).show()
57                                      return
58                                  }
59
60                              }
61
62                          }
63                      }
64                  }
65              }
66          }
67      }
68  }
```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma



```
24     class AgregarTransaccionActivity : AppCompatActivity() {
25         override fun onCreate(savedInstanceState: Bundle?) {
26             binding.btGuardar.setOnClickListener {
27                 .addListenerForSingleValueEvent(object : ValueEventListener {
28                     override fun onDataChange(snapshot: DataSnapshot) {
29                         // Crea y guarda la transacción en Firebase
30                         val transaccionId = usuarioSnap.child( path: "transacciones" ).ref.push().key
31                         val transaccion = Transaccion(transaccionId, tipo, cantidad, categoria, fecha)
32
33                         usuarioSnap.ref.child( pathString: "transacciones" ).child(transaccionId!!).setValue(transaccion)
34                         .addOnSuccessListener {
35                             Toast.makeText( context: this@AgregarTransaccionActivity,
36                                 text: "Transacción guardada", Toast.LENGTH_SHORT).show()
37                             binding.etCantidad.setText("")
38                             binding.etFecha.setText("")
39                             finish() // Vuelve a la pantalla anterior
40                         }
41                         .addOnFailureListener {
42                             Toast.makeText( context: this@AgregarTransaccionActivity,
43                                 text: "Error al guardar transacción", Toast.LENGTH_SHORT).show()
44                         }
45                     }
46                 } else {
47                     Toast.makeText( context: this@AgregarTransaccionActivity,
48                         text: "Usuario no encontrado", Toast.LENGTH_SHORT).show()
49                 }
50             }
51         }
52
53         override fun onCancelled(error: DatabaseError) {
54             Toast.makeText( context: this@AgregarTransaccionActivity,
55                 text: "Error: ${error.message}", Toast.LENGTH_SHORT).show()
56         }
57     }
58 }
```



```
24     class AgregarTransaccionActivity : AppCompatActivity() {
25         override fun onCreate(savedInstanceState: Bundle?) {
26
27             // Configura el listener para el menú de navegación inferior
28             binding.btNavegacion.setOnItemSelectedListener { item ->
29                 when (item.itemId) {
30                     R.id.nav_menu -> {
31                         // Abre la actividad de menú
32                         startActivity(Intent( packageContext: this, MenuActivity::class.java).apply {
33                             putExtra( name: "email", emailUsuario)
34                         })
35                         true
36                     }
37                     R.id.nav_transacciones -> {
38                         // Abre la actividad para ver transacciones
39                         startActivity(Intent( packageContext: this, TransaccionesActivity::class.java).apply {
40                             putExtra( name: "email", emailUsuario)
41                         })
42                         true
43                     }
44                     R.id.nav_estadisticas -> {
45                         // Abre la actividad de estadísticas
46                         startActivity(Intent( packageContext: this, EstadisticasActivity::class.java).apply {
47                             putExtra( name: "email", emailUsuario)
48                         })
49                         true
50                     }
51                     else -> false
52                 }
53             }
54         }
55     }
56 }
```

## 6.5. EditarUsuarioActivity (Modificar información del usuario)

```

1 package com.example.moneymate
2
3 > import ...
18 </> class EditarUsuarioActivity : AppCompatActivity() {
19     private lateinit var binding: ActivityEditarUsuarioBinding
20     private lateinit var database: DatabaseReference
21     // Variable para almacenar el email del usuario que inició sesión
22     private var emailUsuario: String? = null
23     // Variable para almacenar la clave del usuario en la base de datos
24     private var usuarioKey: String? = null
25     // Variable para almacenar el objeto del usuario original (para comparar cambios)
26     private var usuarioOriginal: Usuario? = null
27
28     override fun onCreate(savedInstanceState: Bundle?) {
29         super.onCreate(savedInstanceState)
30         enableEdgeToEdge()
31
32         // Inicializa el binding con el layout correspondiente
33         binding = ActivityEditarUsuarioBinding.inflate(layoutInflater)
34         setContentView(binding.root)
35
36         // Recuperar email del usuario desde el intent
37         emailUsuario = intent.getStringExtra(name: "email")
38
39         // Inicializa la referencia a la base de datos Firebase, apuntando a la rama "Usuarios"
40         database = FirebaseDatabase.getInstance(url: "https://moneymate-6790e-default.firebaseio.com/").getReference(path: "Usuarios")
41
42         // Carga los datos actuales del usuario en el formulario
43         cargarDatosUsuario()
44

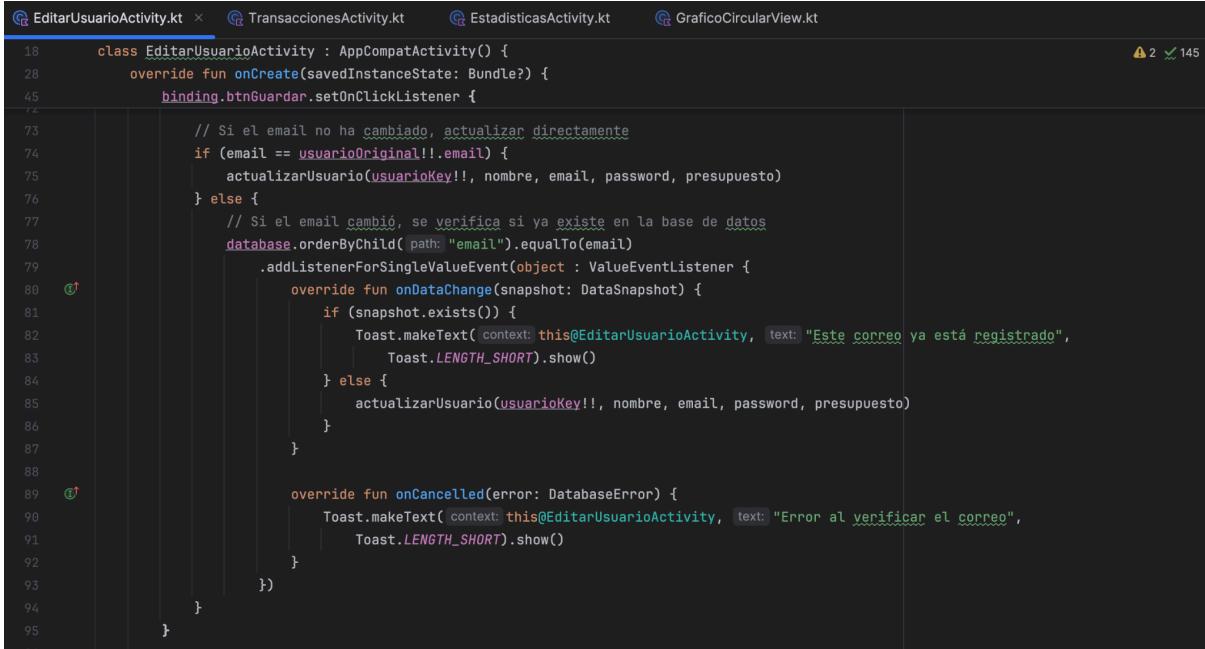
```

```

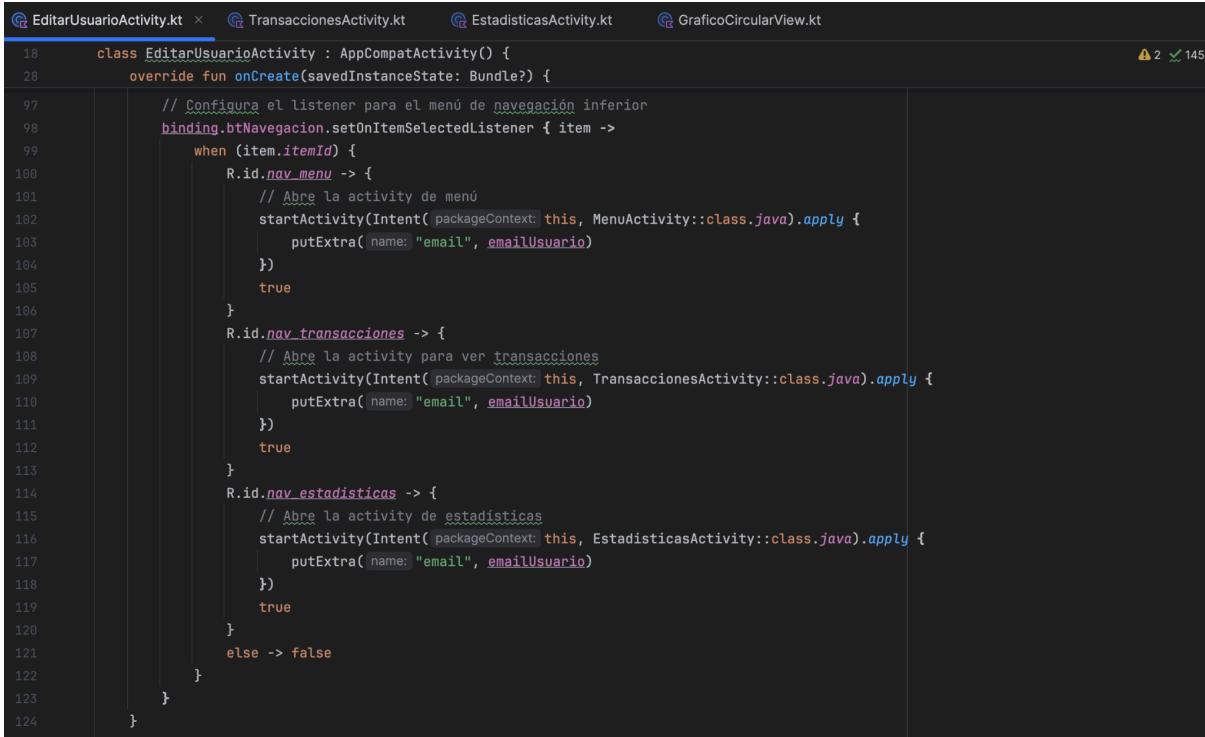
18     class EditarUsuarioActivity : AppCompatActivity() {
19         override fun onCreate(savedInstanceState: Bundle?) {
45             binding.btnSave.setOnClickListener {
46                 // Obtiene los datos del formulario
47                 val nombre = binding.etNombre.text.toString().trim()
48                 val email = binding.etEmail.text.toString().trim()
49                 val password = binding.etPassword.text.toString().trim()
50                 val presupuesto = binding.etPresupuesto.text.toString().toDoubleOrNull() ?: 0.0
51
52                 // Validaciones de campos
53                 if (nombre.isEmpty()) {
54                     binding.etNombre.error = "Nombre requerido"
55                     return@setOnClickListener
56                 }
57
58                 if (email.isEmpty()) {
59                     binding.etEmail.error = "Email requerido"
60                     return@setOnClickListener
61                 }
62
63                 if (password.isEmpty()) {
64                     binding.etPassword.error = "Contraseña requerida"
65                     return@setOnClickListener
66                 }
67
68                 if (!contraseñaValida(password)) {
69                     Toast.makeText(context: this, text: "La contraseña debe tener al menos 6 caracteres y un número", Toast.LENGTH_SHORT).show()
70                     return@setOnClickListener
71                 }
72             }
73         }
74     }

```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

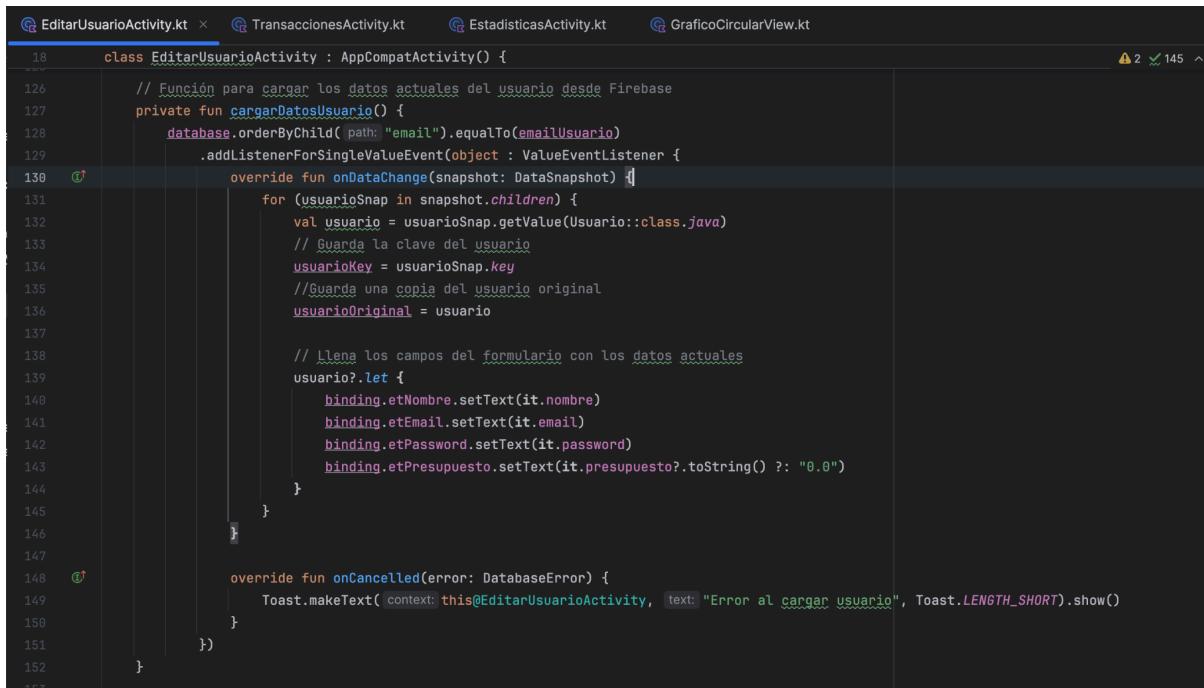


```
18     class EditarUsuarioActivity : AppCompatActivity() {
28         override fun onCreate(savedInstanceState: Bundle?) {
45             binding.btnSaveGuardar.setOnClickListener {
73                 // Si el email no ha cambiado, actualizar directamente
74                 if (email == usuarioOriginal!!.email) {
75                     actualizarUsuario(usuarioKey!!, nombre, email, password, presupuesto)
76                 } else {
77                     // Si el email cambió, se verifica si ya existe en la base de datos
78                     database.orderByChild("email").equalTo(email)
79                     .addListenerForSingleValueEvent(object : ValueEventListener {
80                         override fun onDataChange(snapshot: DataSnapshot) {
81                             if (snapshot.exists()) {
82                                 Toast.makeText(context, "Este correo ya está registrado",
83                                     Toast.LENGTH_SHORT).show()
84                             } else {
85                                 actualizarUsuario(usuarioKey!!, nombre, email, password, presupuesto)
86                             }
87                         }
88
89                         override fun onCancelled(error: DatabaseError) {
90                             Toast.makeText(context, "Error al verificar el correo",
91                                     Toast.LENGTH_SHORT).show()
92                         }
93                     })
94                 }
95             }
96         }
97     }
```

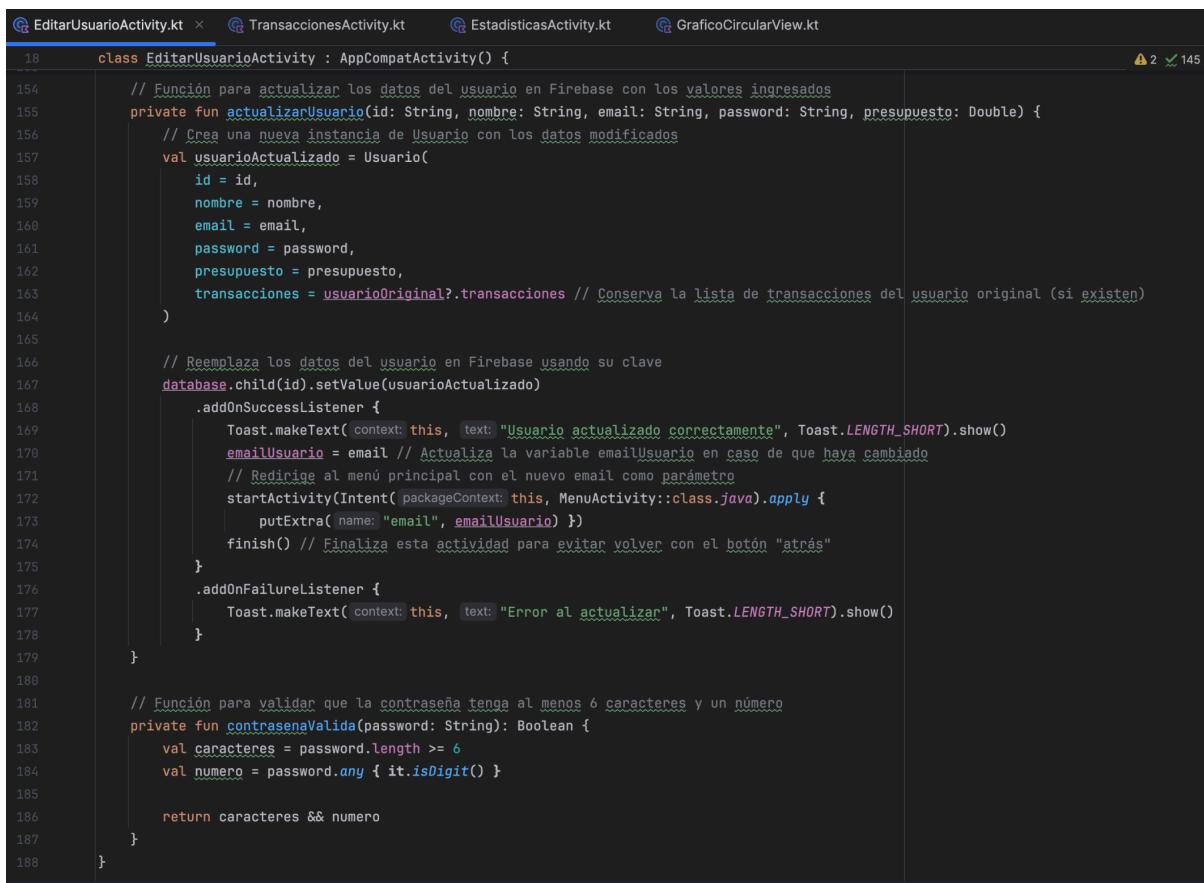


```
18     class EditarUsuarioActivity : AppCompatActivity() {
28         override fun onCreate(savedInstanceState: Bundle?) {
97             // Configura el listener para el menú de navegación inferior
98             binding.bnavegacion.setOnItemSelectedListener { item -
99                 when (item.itemId) {
100                     R.id.nav_menu -> {
101                         // Abre la actividad de menú
102                         startActivity(Intent(packageContext, MenuActivity::class.java).apply {
103                             putExtra("email", emailUsuario)
104                         })
105                         true
106                     }
107                     R.id.nav_transacciones -> {
108                         // Abre la actividad para ver transacciones
109                         startActivity(Intent(packageContext, TransaccionesActivity::class.java).apply {
110                             putExtra("email", emailUsuario)
111                         })
112                         true
113                     }
114                     R.id.nav_estadisticas -> {
115                         // Abre la actividad de estadísticas
116                         startActivity(Intent(packageContext, EstadisticasActivity::class.java).apply {
117                             putExtra("email", emailUsuario)
118                         })
119                         true
120                     }
121                     else -> false
122                 }
123             }
124         }
125     }
```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

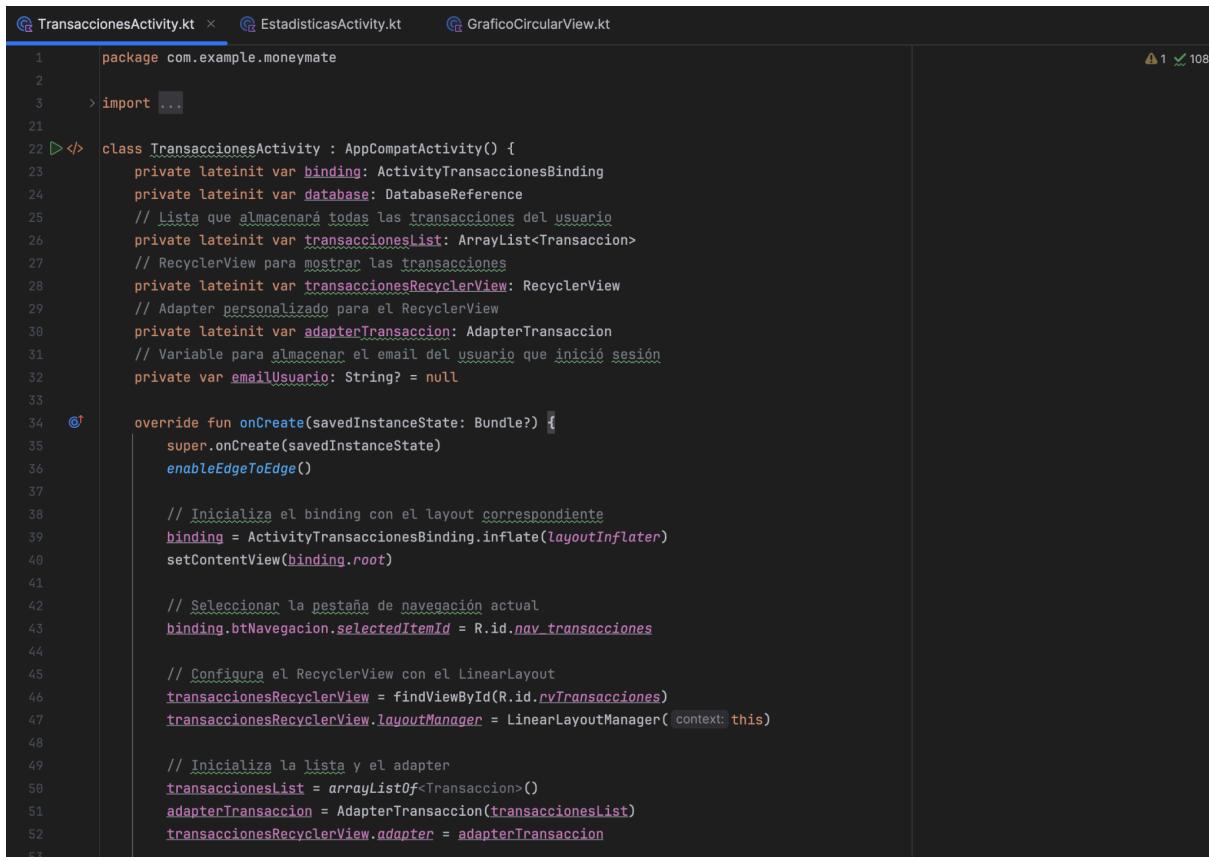


```
18  class EditarUsuarioActivity : AppCompatActivity() {
19
20      // Función para cargar los datos actuales del usuario desde Firebase
21      private fun cargarDatosUsuario() {
22          database.orderByChild("email").equalTo(emailUsuario)
23              .addValueEventListener(object : ValueEventListener {
24
25                  override fun onDataChange(snapshot: DataSnapshot) {
26                      for (usuarioSnap in snapshot.children) {
27                          val usuario = usuarioSnap.getValue(User::class.java)
28                          // Guarda la clave del usuario
29                          usuarioKey = usuarioSnap.key
30                          // Guarda una copia del usuario original
31                          usuarioOriginal = usuario
32
33                          // Llena los campos del formulario con los datos actuales
34                          usuario?.let {
35                              binding.etNombre.setText(it.nombre)
36                              binding.etEmail.setText(it.email)
37                              binding.etPassword.setText(it.password)
38                              binding.etPresupuesto.setText(it.presupuesto?.toString() ?: "0.0")
39                          }
40
41                  }
42
43                  override fun onCancelled(error: DatabaseError) {
44                      Toast.makeText(context, "Error al cargar usuario", Toast.LENGTH_SHORT).show()
45                  }
46
47              }
48
49      }
50
51  }
```



```
18  class EditarUsuarioActivity : AppCompatActivity() {
19
20      // Función para actualizar los datos del usuario en Firebase con los valores ingresados
21      private fun actualizarUsuario(id: String, nombre: String, email: String, password: String, presupuesto: Double) {
22          // Crea una nueva instancia de Usuario con los datos modificados
23          val usuarioActualizado = Usuario(
24              id = id,
25              nombre = nombre,
26              email = email,
27              password = password,
28              presupuesto = presupuesto,
29              transacciones = usuarioOriginal?.transacciones // Conserva la lista de transacciones del usuario original (si existen)
30          )
31
32          // Reemplaza los datos del usuario en Firebase usando su clave
33          database.child(id).setValue(usuarioActualizado)
34              .addOnSuccessListener {
35                  Toast.makeText(context, "Usuario actualizado correctamente", Toast.LENGTH_SHORT).show()
36                  emailUsuario = email // Actualiza la variable emailUsuario en caso de que haya cambiado
37                  // Redirige al menú principal con el nuevo email como parámetro
38                  startActivity(Intent(packageContext, MainActivity::class.java).apply {
39                      putExtra("email", emailUsuario)
40                  })
41                  finish() // Finaliza esta actividad para evitar volver con el botón "atrás"
42              }
43              .addOnFailureListener {
44                  Toast.makeText(context, "Error al actualizar", Toast.LENGTH_SHORT).show()
45              }
46
47      }
48
49      // Función para validar que la contraseña tenga al menos 6 caracteres y un número
50      private fun contraseñaValida(password: String): Boolean {
51          val caracteres = password.length >= 6
52          val numero = password.any { it.isDigit() }
53
54          return caracteres && numero
55      }
56
57  }
```

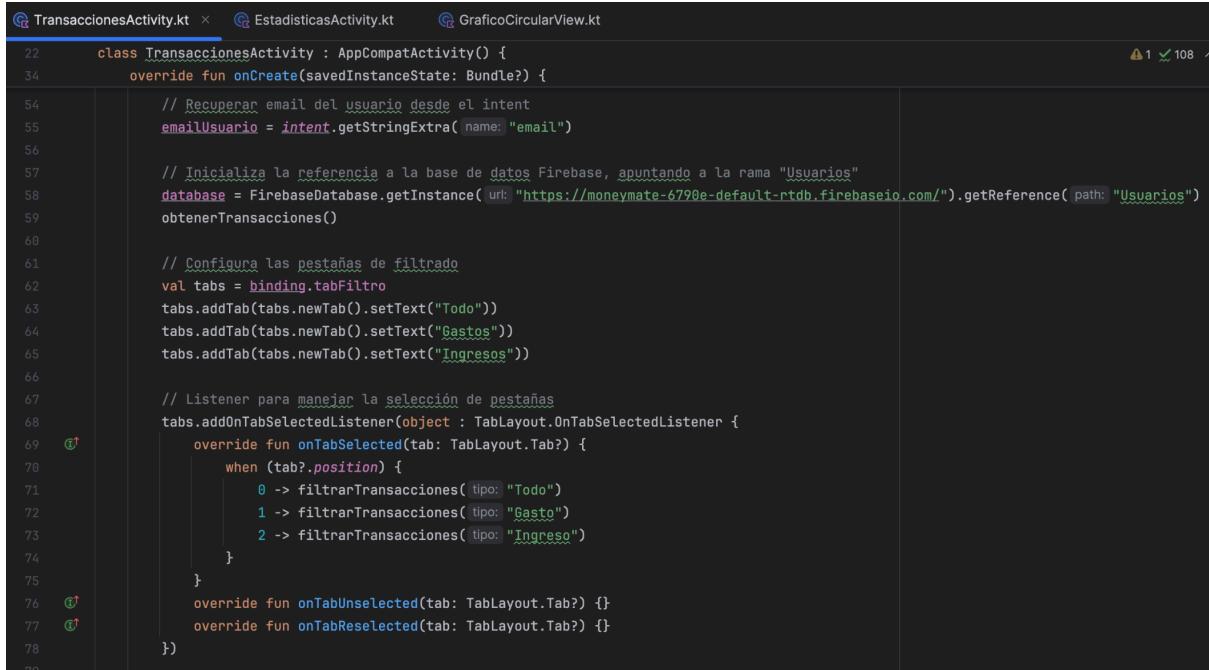
## 6.6. TransaccionesActivity (Lista de transacciones)



```

1 package com.example.moneymate
2
3 > import ...
21
22 </> class TransaccionesActivity : AppCompatActivity() {
23     private lateinit var binding: ActivityTransaccionesBinding
24     private lateinit var database: DatabaseReference
25     // Lista que almacenará todas las transacciones del usuario
26     private lateinit var transaccionesList: ArrayList<Transaccion>
27     // Recyclerview para mostrar las transacciones
28     private lateinit var transaccionesRecyclerView: RecyclerView
29     // Adapter personalizado para el RecyclerView
30     private lateinit var adapterTransaccion: AdapterTransaccion
31     // Variable para almacenar el email del usuario que inició sesión
32     private var emailUsuario: String? = null
33
34     override fun onCreate(savedInstanceState: Bundle?) {
35         super.onCreate(savedInstanceState)
36         enableEdgeToEdge()
37
38         // Inicializa el binding con el layout correspondiente
39         binding = ActivityTransaccionesBinding.inflate(layoutInflater)
40         setContentView(binding.root)
41
42         // Seleccionar la pestaña de navegación actual
43         binding.btNavegacion.selectedItemId = R.id.nav_transacciones
44
45         // Configura el RecyclerView con el LinearLayoutManager
46         transaccionesRecyclerView = findViewById(R.id.rvTransacciones)
47         transaccionesRecyclerView.layoutManager = LinearLayoutManager(context)
48
49         // Inicializa la lista y el adapter
50         transaccionesList = arrayListOf<Transaccion>()
51         adapterTransaccion = AdapterTransaccion(transaccionesList)
52         transaccionesRecyclerView.adapter = adapterTransaccion
53
54     }

```

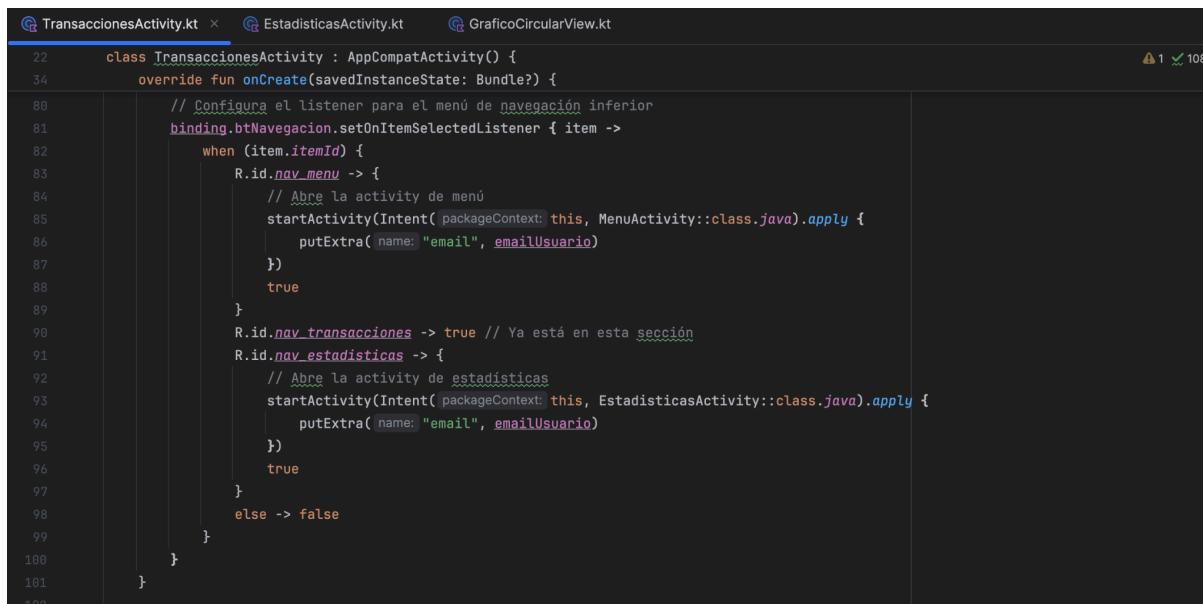


```

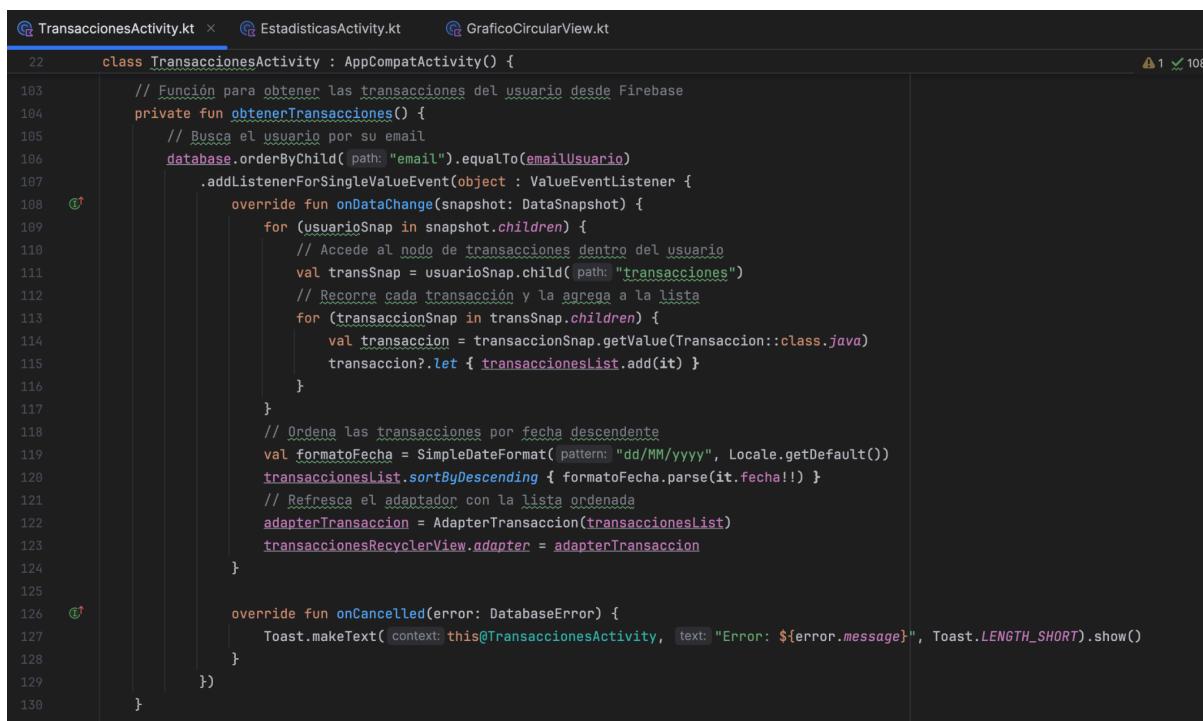
22     class TransaccionesActivity : AppCompatActivity() {
23         override fun onCreate(savedInstanceState: Bundle?) {
24
25             // Recuperar email del usuario desde el intent
26             emailUsuario = intent.getStringExtra(name: "email")
27
28             // Inicializa la referencia a la base de datos Firebase, apuntando a la rama "Usuarios"
29             database = FirebaseDatabase.getInstance(url: "https://moneymate-6790e-default.firebaseio.com/").getReference(path: "Usuarios")
30             obtenerTransacciones()
31
32             // Configura las pestañas de filtrado
33             val tabs = binding.tabFiltro
34             tabs.addTab(tabs.newTab().setText("Todo"))
35             tabs.addTab(tabs.newTab().setText("Gastos"))
36             tabs.addTab(tabs.newTab().setText("Ingresos"))
37
38             // Listener para manejar la selección de pestañas
39             tabs.addOnTabSelectedListener(object : TabLayout.OnTabSelectedListener {
40                 override fun onTabSelected(tab: TabLayout.Tab?) {
41                     when (tab?.position) {
42                         0 -> filtrarTransacciones(tipo: "Todo")
43                         1 -> filtrarTransacciones(tipo: "Gasto")
44                         2 -> filtrarTransacciones(tipo: "Ingreso")
45                     }
46                 }
47
48                 override fun onTabUnselected(tab: TabLayout.Tab?) {}
49                 override fun onTabReselected(tab: TabLayout.Tab?) {}
50             })
51
52         }
53
54     }

```

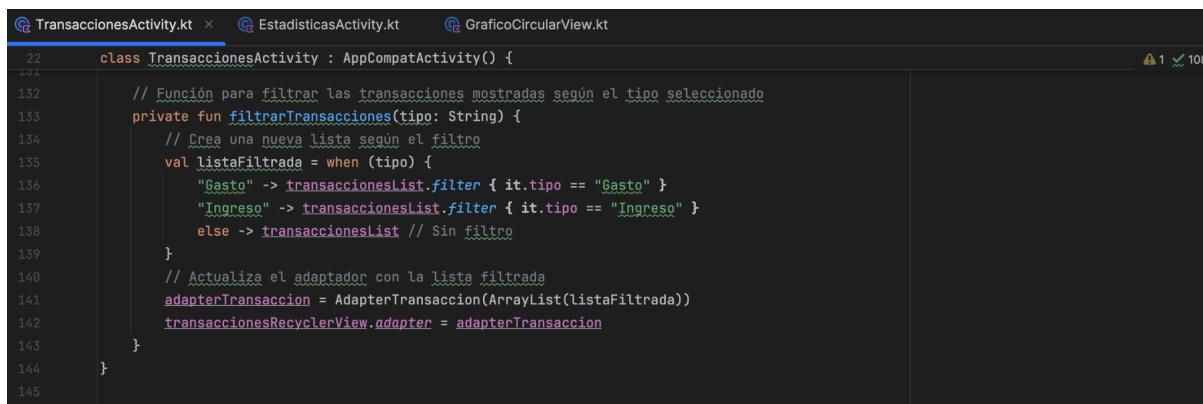
## Proyecto de Desarrollo de Aplicaciones Multiplataforma



```
22 class TransaccionesActivity : AppCompatActivity() {
23     override fun onCreate(savedInstanceState: Bundle?) {
24         // Configura el listener para el menú de navegación inferior
25         binding.btNavegacion.setOnItemSelectedListener { item -
26             when (item.itemId) {
27                 R.id.nav_menu -> {
28                     // Abre la actividad de menú
29                     startActivity(Intent(packageContext: this, MenuActivity::class.java).apply {
30                         putExtra(name: "email", emailUsuario)
31                     })
32                     true
33                 }
34                 R.id.nav_transacciones -> true // Ya está en esta sección
35                 R.id.nav_estadisticas -> {
36                     // Abre la actividad de estadísticas
37                     startActivity(Intent(packageContext: this, EstadisticasActivity::class.java).apply {
38                         putExtra(name: "email", emailUsuario)
39                     })
40                     true
41                 }
42                 else -> false
43             }
44         }
45     }
46 }
```



```
22 class TransaccionesActivity : AppCompatActivity() {
23     // Función para obtener las transacciones del usuario desde Firebase
24     private fun obtenerTransacciones() {
25         // Busca el usuario por su email
26         database.orderByChild(path: "email").equalTo(emailUsuario)
27             .addListenerForSingleValueEvent(object : ValueEventListener {
28                 override fun onDataChange(snapshot: DataSnapshot) {
29                     for (usuarioSnap in snapshot.children) {
30                         // Accede al nodo de transacciones dentro del usuario
31                         val transSnap = usuarioSnap.child(path: "transacciones")
32                         // Recorre cada transacción y la agrega a la lista
33                         for (transaccionSnap in transSnap.children) {
34                             val transaccion = transaccionSnap.getValue(Transaccion::class.java)
35                             transaccion?.let { transaccionesList.add(it) }
36                         }
37                     }
38                     // Ordena las transacciones por fecha descendente
39                     val formatoFecha = SimpleDateFormat(pattern: "dd/MM/yyyy", Locale.getDefault())
40                     transaccionesList.sortByDescending { formatoFecha.parse(it.fecha!) }
41                     // Refresca el adaptador con la lista ordenada
42                     adapterTransaccion = AdapterTransaccion(transaccionesList)
43                     transaccionesRecyclerView.adapter = adapterTransaccion
44                 }
45             }
46             override fun onCancelled(error: DatabaseError) {
47                 Toast.makeText(context: this@TransaccionesActivity, text: "Error: ${error.message}", Toast.LENGTH_SHORT).show()
48             }
49         }
50     }
51 }
```



```
22 class TransaccionesActivity : AppCompatActivity() {
23     // Función para filtrar las transacciones mostradas según el tipo seleccionado
24     private fun filtrarTransacciones(tipo: String) {
25         // Crea una nueva lista según el filtro
26         val listaFiltrada = when (tipo) {
27             "Gasto" -> transaccionesList.filter { it.tipo == "Gasto" }
28             "Ingreso" -> transaccionesList.filter { it.tipo == "Ingreso" }
29             else -> transaccionesList // Sin filtro
30         }
31         // Actualiza el adaptador con la lista filtrada
32         adapterTransaccion = AdapterTransaccion(ArrayList(listaFiltrada))
33         transaccionesRecyclerView.adapter = adapterTransaccion
34     }
35 }
```

### 6.6.1. AdapterTransacciones

```

1 package com.example.moneymate.Adapter
2
3 > import ...
4
5 // Adaptador para mostrar una lista de transacciones en un RecyclerView
6 class AdapterTransaccion(private val transacciones: ArrayList<Transaccion>) :
7     RecyclerView.Adapter<AdapterTransaccion.ViewHolder>() {
8
9     // ViewHolder que contiene las referencias a los elementos de cada ítem en la lista
10    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
11        val icono: ImageView = itemView.findViewById(R.id.icono) // Ícono que representa el tipo de transacción
12        val tvTipo: TextView = itemView.findViewById(R.id.tvTipo) // Texto que muestra si es ingreso o gasto
13        val tvCantidad: TextView = itemView.findViewById(R.id.tvCantidad) // Cantidad de la transacción
14        val tvCategoria: TextView = itemView.findViewById(R.id.tvCategoria) // Categoría de la transacción
15        val tvFecha: TextView = itemView.findViewById(R.id.tvFecha) // Fecha en formato legible
16        val cardView: CardView = itemView.findViewById(R.id.cardView) // CardView para cambiar el color de fondo
17
18    }
19
20    // Infla el layout de cada ítem de la lista y crea un ViewHolder asociado
21    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
22        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_transaccion, parent, attachToRoot: false)
23        return ViewHolder(view)
24    }
25
26
27
28
29    // ...
30
31
32
33

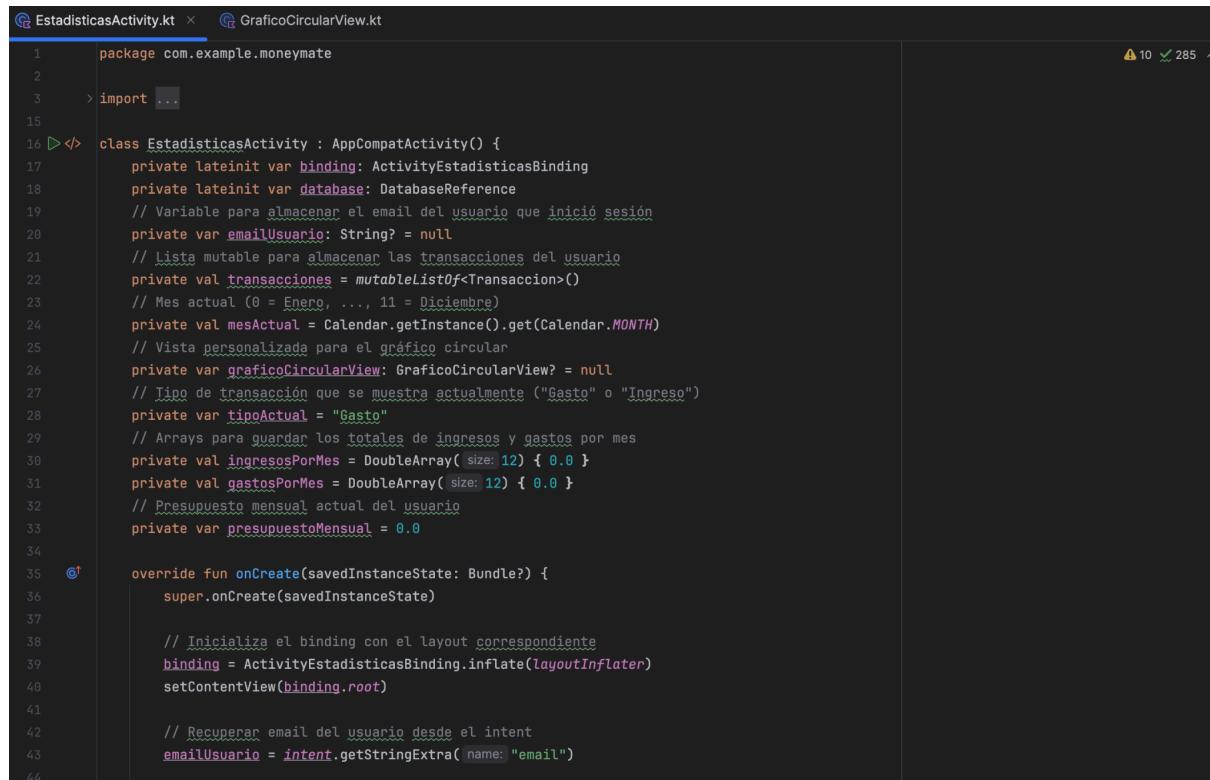
```

```

14 class AdapterTransaccion(private val transacciones: ArrayList<Transaccion>) :
15
16     // Vincula los datos de la transacción actual a los elementos del ViewHolder
17     override fun onBindViewHolder(holder: ViewHolder, position: Int) {
18
19         // Obtiene la transacción en la posición actual
20         val transaccion = transacciones[position]
21
22         // Asigna los textos correspondientes
23         holder.tvTipo.text = "${transaccion.tipo}"
24         holder.tvCantidad.text = "${transaccion.cantidad}"
25         holder.tvCategoria.text = "${transaccion.categoría}"
26         holder.tvFecha.text = transaccion.fecha
27
28
29         // Obtiene el contexto para acceder a recursos
30         val context = holder.itemView.context
31
32         // Asigna color de fondo del cardView según el tipo de transacción
33         val color = when (transaccion.tipo) {
34             "Gasto" -> context.getColor(R.color.azulClaro)
35             "Ingreso" -> context.getColor(R.color.azul)
36             else -> context.getColor(R.color.black)
37         }
38
39         holder.cardView.setCardBackgroundColor(color)
40
41         // Selecciona el ícono según el tipo de transacción
42         val iconoResult = when (transaccion.tipo) {
43             "Ingreso" -> R.drawable.icono_ingreso
44             "Gasto" -> R.drawable.icono_gasto
45             else -> R.drawable.icono_ingreso
46         }
47
48         holder.icono.setImageResource(iconoResult)
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

```

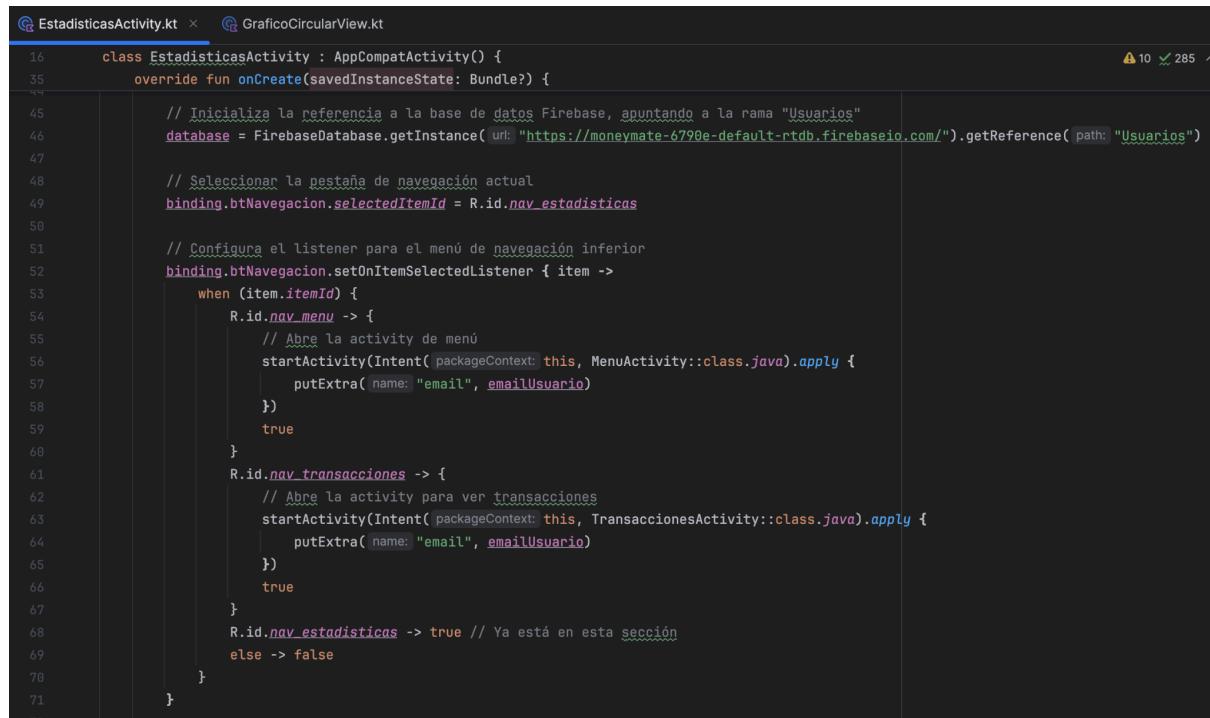
## 6.7. EstadísticasActivity (Gráficos y presupuesto mensual)



```

1 package com.example.moneymate
2
3 > import ...
15
16 class EstadisticasActivity : AppCompatActivity() {
17     private lateinit var binding: ActivityEstadisticasBinding
18     private lateinit var database: DatabaseReference
19     // Variable para almacenar el email del usuario que inició sesión
20     private var emailUsuario: String? = null
21     // Lista mutable para almacenar las transacciones del usuario
22     private val transacciones = mutableListOf<Transaccion>()
23     // Mes actual (0 = Enero, ..., 11 = Diciembre)
24     private val mesActual = Calendar.getInstance().get(Calendar.MONTH)
25     // Vista personalizada para el gráfico circular
26     private var graficoCircularView: GraficoCircularView? = null
27     // Tipo de transacción que se muestra actualmente ("Gasto" o "Ingreso")
28     private var tipoActual = "Gasto"
29     // Arrays para guardar los totales de ingresos y gastos por mes
30     private val ingresosPorMes = DoubleArray( size: 12 ) { 0.0 }
31     private val gastosPorMes = DoubleArray( size: 12 ) { 0.0 }
32     // Presupuesto mensual actual del usuario
33     private var presupuestoMensual = 0.0
34
35     override fun onCreate(savedInstanceState: Bundle?) {
36         super.onCreate(savedInstanceState)
37
38         // Inicializa el binding con el layout correspondiente
39         binding = ActivityEstadisticasBinding.inflate(layoutInflater)
40         setContentView(binding.root)
41
42         // Recuperar email del usuario desde el intent
43         emailUsuario = intent.getStringExtra( name: "email" )
44
45     }

```



```

16 class EstadisticasActivity : AppCompatActivity() {
35     override fun onCreate(savedInstanceState: Bundle?) {
44
45         // Inicializa la referencia a la base de datos Firebase, apuntando a la rama "Usuarios"
46         database = FirebaseDatabase.getInstance( url: "https://moneymate-6790e-default.firebaseio.com/" ).getReference( path: "Usuarios" )
47
48         // Seleccionar la gesta a de navegaci n actual
49         binding.btNavegacion.selectedItemId = R.id.nav_estadisticas
50
51         // Configura el listener para el men  de navegaci n inferior
52         binding.btNavegacion.setOnItemSelectedListener { item ->
53             when (item.itemId) {
54                 R.id.nav_menu -> {
55                     // Abre la actividad de men 
56                     startActivity(Intent( packageContext: this, MenuActivity::class.java ).apply {
57                         putExtra( name: "email", emailUsuario )
58                     })
59                     true
60                 }
61                 R.id.nav_transacciones -> {
62                     // Abre la actividad para ver transacciones
63                     startActivity(Intent( packageContext: this, TransaccionesActivity::class.java ).apply {
64                         putExtra( name: "email", emailUsuario )
65                     })
66                     true
67                 }
68                 R.id.nav_estadisticas -> true // Ya est  en esta secci n
69                 else -> false
70             }
71         }
72     }

```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

```
EstadisticasActivity.kt x GraficoCircularView.kt
16  class EstadisticasActivity : AppCompatActivity() {
35      override fun onCreate(savedInstanceState: Bundle?) {
73          // Listener para toggle que cambia entre mostrar gastos o ingresos
74          binding.toggleTipo.setOnCheckedChangeListener { _, isChecked ->
75              tipoActual = if (isChecked) "Ingreso" else "Gasto"
76              actualizarGraficoVerticalYGraficoCircular()
77          }
78
79          // Listener para editar el presupuesto mensual
80          binding.tvEditar.setOnClickListener {
81              actualizarPresupuestoMensual()
82          }
83
84          // Obtener transacciones desde Firebase
85          obtenerTransacciones()
86      }

```

```
EstadisticasActivity.kt x GraficoCircularView.kt
16  class EstadisticasActivity : AppCompatActivity() {
88      // Función para obtener transacciones del usuario desde Firebase
89      private fun obtenerTransacciones() {
90          if (emailUsuario == null) {
91              Toast.makeText(context, text: "Usuario no identificado", Toast.LENGTH_SHORT).show()
92              return
93          }
94
95          // Consulta para obtener usuario por email
96          database.orderByChild( path: "email").equalTo(emailUsuario)
97          .addListenerForSingleValueEvent(object : ValueEventListener {
98              override fun onDataChange(snapshot: DataSnapshot) {
99                  transacciones.clear() // Limpiar la lista de transacciones antes de cargar nuevas
100                 for (usuarioSnap in snapshot.children) {
101                     val usuario = usuarioSnap.getValue(Usuario::class.java)
102                     if (usuario != null) {
103                         // Obtener el presupuesto mensual del usuario
104                         presupuestoMensual = usuario.presupuesto ?: 0.0
105
106                         // Añadir todas las transacciones del usuario a la lista y procesarlas
107                         usuario.transacciones?.values?.forEach { trans ->
108                             transacciones.add(trans)
109                             procesarTransaccion(trans)
110                         }
111                     }
112                 }
113                 // Actualizar gráficos y mostrar presupuesto actualizado
114                 actualizarGraficoVerticalYGraficoCircular()
115                 actualizarPresupuestoMensual()
116             }
117
118             override fun onCancelled(error: DatabaseError) {
119                 Toast.makeText( context: this@EstadisticasActivity, text: "Error: ${error.message}", Toast.LENGTH_SHORT).show()
120             }
121         })
122     }

```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

```
④ EstadisticasActivity.kt × ④ GraficoCircularView.kt
16     class EstadisticasActivity : AppCompatActivity() {
124     // Función para actualizar la interfaz del presupuesto mensual
125     private fun actualizarPresupuestoMensual() {
126         val gastoMesActual = gastosPorMes[mesActual]
127
128         // Calcular porcentaje del gasto respecto al presupuesto, limitado al 100%
129         val porcentajeInicial = if (presupuestoMensual > 0) {
130             ((gastoMesActual / presupuestoMensual) * 100).toInt().coerceAtMost(maximumValue: 100)
131         } else {
132             0
133         }
134
135         // Mostrar texto con gasto y presupuesto actual
136         binding.tvPresupuesto.text = "${"%.2f".format(gastoMesActual)}€ de ${"%.2f".format(presupuestoMensual)}€"
137         // Actualizar barra de progreso con el porcentaje calculado
138         binding.progressBar.progress = porcentajeInicial
139
140         binding.tvEditar.setOnClickListener {
141             // Mostrar el EditText y el botón de guardar
142             binding.etNuevoPresupuesto.visibility = View.VISIBLE
143             binding.btnGuardar.visibility = View.VISIBLE
144             binding.btnCancelar.visibility = View.VISIBLE
145
146             // Configurar el botón de guardar
147             binding.btnGuardar.setOnClickListener {
148                 val nuevoPresupuestoTexto = binding.etNuevoPresupuesto.text.toString()
149                 if (nuevoPresupuestoTexto.isNotEmpty()) {
150                     val nuevoPresupuesto = nuevoPresupuestoTexto.toDoubleOrNull()
151                     if (nuevoPresupuesto != null && nuevoPresupuesto > 0) {
152                         presupuestoMensual = nuevoPresupuesto
153
154                         // Actualizar texto y barra de progreso con nuevo presupuesto
155                         val porcentaje = ((gastoMesActual / presupuestoMensual) * 100).toInt().coerceAtMost(maximumValue: 100)
156                         binding.tvPresupuesto.text = "${"%.2f".format(gastoMesActual)}€ de ${"%.2f".format(presupuestoMensual)}€"
157                         binding.progressBar.progress = porcentaje
158
159             
```

```
④ EstadisticasActivity.kt × ④ GraficoCircularView.kt
16     class EstadisticasActivity : AppCompatActivity() {
125     private fun actualizarPresupuestoMensual() {
126         binding.tvEditar.setOnClickListener {
127             binding.btnGuardar.setOnClickListener {
128                 // Guardar nuevo presupuesto en Firebase
129                 database.orderByChild(path: "email").equalTo(emailUsuario)
130                 .addListenerForSingleValueEvent(object : ValueEventListener {
131                     override fun onDataChange(snapshot: DataSnapshot) {
132                         for (usuarioSnap in snapshot.children) {
133                             val userId = usuarioSnap.key
134                             if (userId != null) {
135                                 database.child(userId).child(pathString: "presupuesto").setValue(nuevoPresupuesto)
136                             }
137                         }
138                     }
139
140                     override fun onCancelled(error: DatabaseError) {
141                         Toast.makeText(context: this@EstadisticasActivity, text: "Error al actualizar: ${error.message}", duration: Toast.LENGTH_SHORT).show()
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     }
```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

```
EstadisticasActivity.kt × GraficoCircularView.kt
16  class EstadisticasActivity : AppCompatActivity() {
125     private fun actualizarPresupuestoMensual() {
140         binding.tvEditar.setOnClickListener {
188             // Configurar el botón de cancelar
189             binding.btnCancel.setOnClickListener {
190                 // Ocultar el EditText y los botones
191                 binding.etNuevoPresupuesto.visibility = View.GONE
192                 binding.btnGuardar.visibility = View.GONE
193                 binding.btnCancel.visibility = View.GONE
194                 binding.etNuevoPresupuesto.setText("")
195             }
196         }
197     }
198
199     // Función para actualizar ambos gráficos (vertical y circular)
200     private fun actualizarGraficoVerticalYGraficoCircular() {
201         actualizarGraficoVertical()
202         generarGraficoCircular()
203     }
204
205     // Genera el gráfico circular mostrando la distribución de gastos o ingresos por categorías
206     private fun generarGraficoCircular() {
207         // Definir categorías según el tipo actual
208         val categorias = if (tipoActual == "Ingreso") {
209             listOf("Salario", "Ventas", "Inversiones", "Regalos", "Reembolsos", "Otros")
210         } else {
211             listOf("Alimentación", "Transporte", "Ocio", "Salud", "Hogar", "Otros")
212         }
213         val formato = SimpleDateFormat(pattern: "dd/MM/yyyy", Locale.getDefault())
214
215         // Obtener nombre del mes actual
216         val mesNombre = SimpleDateFormat(pattern: "MMMM", Locale(language: "es")).format(Calendar.getInstance().time)
217         binding.tvMes.text = "${mesNombre.replaceFirstChar { it.uppercase() }}"
218     }

```

```
EstadisticasActivity.kt × GraficoCircularView.kt
16  class EstadisticasActivity : AppCompatActivity() {
206     private fun generarGraficoCircular() {
210
219         // Calcular total por categoría para el mes actual y tipo seleccionado
220         val totalesPorCategoria = categorias.associateWith { cat ->
221             transacciones.filter { trans ->
222                 trans.tipo.equals(tipoActual, ignoreCase: true) &&
223                 trans.categoría == cat &&
224                 formato.parse(source: trans.fecha ?: "")?.let { fecha ->
225                     Calendar.getInstance().apply { time = fecha }.get(Calendar.MONTH) == mesActual
226                 } == true
227             }.sumOf { it.cantidad ?: 0.0 }
228         }
229         actualizarLeyenda(categorias) // Actualizar leyenda con los nombres de categorías
230
231         // Colores para las categorías (los mismos del layout)
232         val coloresCategorias = listOf(
233             getColor(R.color.azul1), // Categoría 1
234             getColor(R.color.azul2), // Categoría 2
235             getColor(R.color.azul3), // Categoría 3
236             getColor(R.color.azul4), // Categoría 4
237             getColor(R.color.azul5), // Categoría 5
238             getColor(R.color.azul6) // Categoría 6
239         )
240
241         // Solo se crea una vez el gráfico circular
242         if (graficoCircularView == null) {
243             graficoCircularView = GraficoCircularView(context: this)
244             graficoCircularView!!.layoutParams = binding.graficoCircular.layoutParams
245             val contenedor = binding.contenedorCircular as ViewGroup
246             contenedor.removeView(binding.graficoCircular) // Solo la primera vez
247             contenedor.addView(graficoCircularView)
248         }
249
250         // Asigna los datos calculados al gráfico circular
251         graficoCircularView!!.setDatos(totalesPorCategoria, coloresCategorias)
252     }

```

## Proyecto de Desarrollo de Aplicaciones Multiplataforma

```
④ EstadisticasActivity.kt × ④ GraficoCircularView.kt
16  class EstadisticasActivity : AppCompatActivity() {
254      private fun actualizarLeyenda(categorias: List<String>) {
255          // Asocia los TextView de la leyenda con las categorías
256          val textos = listOf(
257              binding.leyenda1.getChildAt( index: 1 ) as TextView,
258              binding.leyenda1.getChildAt( index: 3 ) as TextView,
259              binding.leyenda1.getChildAt( index: 5 ) as TextView,
260              binding.leyenda2.getChildAt( index: 1 ) as TextView,
261              binding.leyenda2.getChildAt( index: 3 ) as TextView,
262              binding.leyenda2.getChildAt( index: 5 ) as TextView
263          )
264          // Establece el texto correspondiente en cada posición
265          categorias.forEachIndexed { i, cat -> textos[i].text = "$cat" }
266      }
267
268      private fun procesarTransaccion(trans: Transaccion) {
269          try {
270              // Convierte la fecha a formato utilizable
271              val formato = SimpleDateFormat(pattern: "dd/MM/yyyy", Locale.getDefault())
272              val fecha = formato.parse(source: trans.fecha ?: "")
273              val calendario = Calendar.getInstance()
274              calendario.time = fecha ?: return
275              val mes = calendario.get(Calendar.MONTH)
276
277              // Acumula el valor en el array correspondiente al mes
278              when (trans.tipo) {
279                  "Ingreso" -> ingresosPorMes[mes] += trans.cantidad ?: 0.0
280                  "Gasto" -> gastosPorMes[mes] += trans.cantidad ?: 0.0
281              }
282          } catch (e: Exception) {
283              e.printStackTrace()
284          }
285      }
286  }
```

```
④ EstadisticasActivity.kt × ④ GraficoCircularView.kt
16  class EstadisticasActivity : AppCompatActivity() {
287      private fun actualizarGraficoVertical() {
288          val maxAltura = 140
289          val maxValor = maxOf( a: ingresosPorMes.maxOrNull() ?: 0.0, b: gastosPorMes.maxOrNull() ?: 0.0 )
290          val escala = if (maxValor > 0) maxAltura / maxValor else 1.0
291
292          // IDs de las barras de ingresos y gastos (por mes)
293          val idsIngresos = arrayOf(R.id.ingEne, R.id.ingFeb, R.id.ingMar, R.id.ingAbr, R.id.ingMay, R.id.ingJun,
294              R.id.ingJul, R.id.ingAgo, R.id.ingSep, R.id.ingOct, R.id.ingNov, R.id.ingDic)
295          val idsGastos = arrayOf(R.id.gasEne, R.id.gasFeb, R.id.gasMar, R.id.gasAbr, R.id.gasMay, R.id.gasJun,
296              R.id.gasJul, R.id.gasAgo, R.id.gasSep, R.id.gasOct, R.id.gasNov, R.id.gasDic)
297
298          // Asigna la altura adecuada a cada barra según la escala
299          for (i in 0 .. ≤ 11) {
300              actualizarBarra(binding.graficoVertical.findViewById(idsIngresos[i]), ingresosPorMes[i], escala)
301              actualizarBarra(binding.graficoVertical.findViewById(idsGastos[i]), gastosPorMes[i], escala)
302          }
303      }
304
305      // Los valores que se asignan a layoutParams.height están en pixels y no en dp.
306      // Función para ajustar dinámicamente la altura de las barras en el gráfico, según el valor y la escala
307      private fun actualizarBarra(view: View, valor: Double, escala: Double) {
308          // Hay que convertir los dp a pixels antes de asignar la altura
309          val alturaDp = (valor * escala).toInt()
310          val density = view.context.resources.displayMetrics.density
311          val alturaPx = (alturaDp * density).toInt()
312
313          // Aplica la nueva altura a la vista
314          val layoutParams = view.layoutParams
315          layoutParams.height = alturaPx
316          view.layoutParams = layoutParams
317      }
318  }
```

### **6.7.1. GraficoCircularView**

```
1 package com.example.moneymate
2
3 > import ...
4
5
6 // Clase personalizada que extiende View y dibuja un gráfico circular
7 class GraficoCircularView(context: Context) : View(context) {
8     // Mapa que contiene el total por cada categoría (por ejemplo: "Alimentación" -> 120.0)
9     var totalesPorCategoria: Map<String, Double> = mapOf()
10    // Lista de colores asociados a cada categoría
11    var coloresCategorías: List<Int> = listOf()
12
13    // Método sobreescrito que se llama automáticamente para dibujar la vista en la pantalla
14    override fun onDraw(canvas: Canvas) {
15        super.onDraw(canvas)
16        // Suma el total de todos los valores, para calcular los porcentajes
17        val total = totalesPorCategoria.values.sum()
18        if (total == 0.0) return // Si no hay datos, no dibuja nada
19
20        // Define el área en la que se va a dibujar el gráfico
21        val rectF = RectF(left: 0f, top: 0f, width.toFloat(), height.toFloat())
22        // El ángulo inicial se coloca en la parte superior del círculo (-90 grados)
23        var inicioAngulo = -90f
24        // Objeto Paint para definir el color y estilo de los sectores
25        val paint = Paint(Paint.ANTI_ALIAS_FLAG).apply { style = Paint.Style.FILL }
26
27        // Dibujar el gráfico circular
28        canvas.drawArc(rectF, inicioAngulo, 360f, true, paint)
29
30    }
31}
```

```
GraficoCircularView.kt x
11 class GraficoCircularView(context: Context) : View(context) {
12     override fun onDraw(canvas: Canvas) {
13
14         // Itera por cada categoría y su valor, junto con su índice para el color
15         totalesPorCategoria.entries.forEachIndexed { index, entry ->
16
17             val valor = entry.value
18             if (valor > 0) {
19
20                 // Calcula el ángulo proporcional que debe ocupar este sector
21                 val angulo = (valor / total * 360).toFloat()
22
23                 // Asigna el color correspondiente
24                 paint.color = coloresCategorias.getOrDefault(index) { Color.GRAY }
25
26                 // Dibuja el sector en el canvas
27                 canvas.drawArc(rectF, inicioAngulo, angulo, useCenter: true, paint)
28
29                 // Actualiza el ángulo de inicio para el siguiente sector
30                 inicioAngulo += angulo
31
32             }
33
34         }
35
36     }
37
38     // Método para asignar nuevos datos al gráfico y forzar su redibujo
39     fun setDatos(datos: Map<String, Double>, colores: List<Int>) {
40
41         totalesPorCategoria = datos
42         coloresCategorias = colores
43         invalidate()
44
45     }
46
47
48 }
49
50
51
52
53
54 }
```

## 7. Mayores dificultades encontradas

Durante el desarrollo de la aplicación me encontré con distintas dificultades tanto técnicas como de diseño. A continuación, detallo las más relevantes:

### 7.1. Integración con Firebase Realtime Database

Uno de los principales retos fue la correcta integración con **Firebase Realtime Database**, especialmente debido a su estructura jerárquica y no relacional. Acceder a información específica, como las transacciones de cada usuario, implicaba trabajar con nodos anidados y entender el uso de objetos **DataSnapshot**, así como los métodos **orderByChild** y **equalTo** para filtrar datos por campos concretos, como el email. Pero una vez que estructuré mejor el modelo de datos y entendí cómo navegar por los nodos, el trabajo con la base de datos fue mucho más fluido.

### 7.2. Procesamiento de fechas

Otro desafío fue trabajar con las **fechas**, ya que estaban almacenadas como cadenas de texto en formato "dd/MM/yyyy". Tuve que convertirlas en objetos **Date** y luego usar **Calendar** para saber a qué mes correspondía cada transacción. Esto generó algunos errores, sobre todo cuando las fechas venían mal formateadas o eran nulas. Para solucionarlo, añadí validaciones y controles que ayudaron a evitar fallos al calcular las estadísticas mensuales.

### 7.3. Implementación de gráficos personalizados

Diseñar mis propios gráficos fue otra tarea compleja. Para el **gráfico circular**, por ejemplo, tuve que calcular bien los ángulos proporcionales a cada categoría, aplicar los colores correctos y asegurarme de que el gráfico se redibujara cuando los datos cambiaban. También fue un reto ajustar la escala del **gráfico de barras vertical** y convertir correctamente las medidas para que los valores se representaran de forma visualmente clara y coherente.

### 7.4. Interactividad y cambios dinámicos

Incluir opciones interactivas, como el **interruptor** para alternar entre **ingresos** y **gastos**, me obligó a gestionar bien el estado interno de la aplicación. Tenía que asegurarme de que al cambiar el tipo de transacción, tanto los gráficos como las etiquetas se actualizaran al instante. Para evitar repetir código y facilitar el mantenimiento, agrupé todo el proceso en una única función que se encarga de actualizar ambos gráficos al mismo tiempo.

### 7.5. Adaptación del diseño a distintos dispositivos

Por último, una dificultad adicional fue conseguir que todos los elementos visuales de la aplicación se adaptaran correctamente a distintos **tamaños de pantalla** y **resoluciones**. Tuve que ajustar el diseño XML, revisar márgenes y tamaños de texto, y hacer varias pruebas en distintos dispositivos para asegurarme de que los gráficos y demás elementos visuales se mantuvieran legibles y equilibrados en todos los casos.