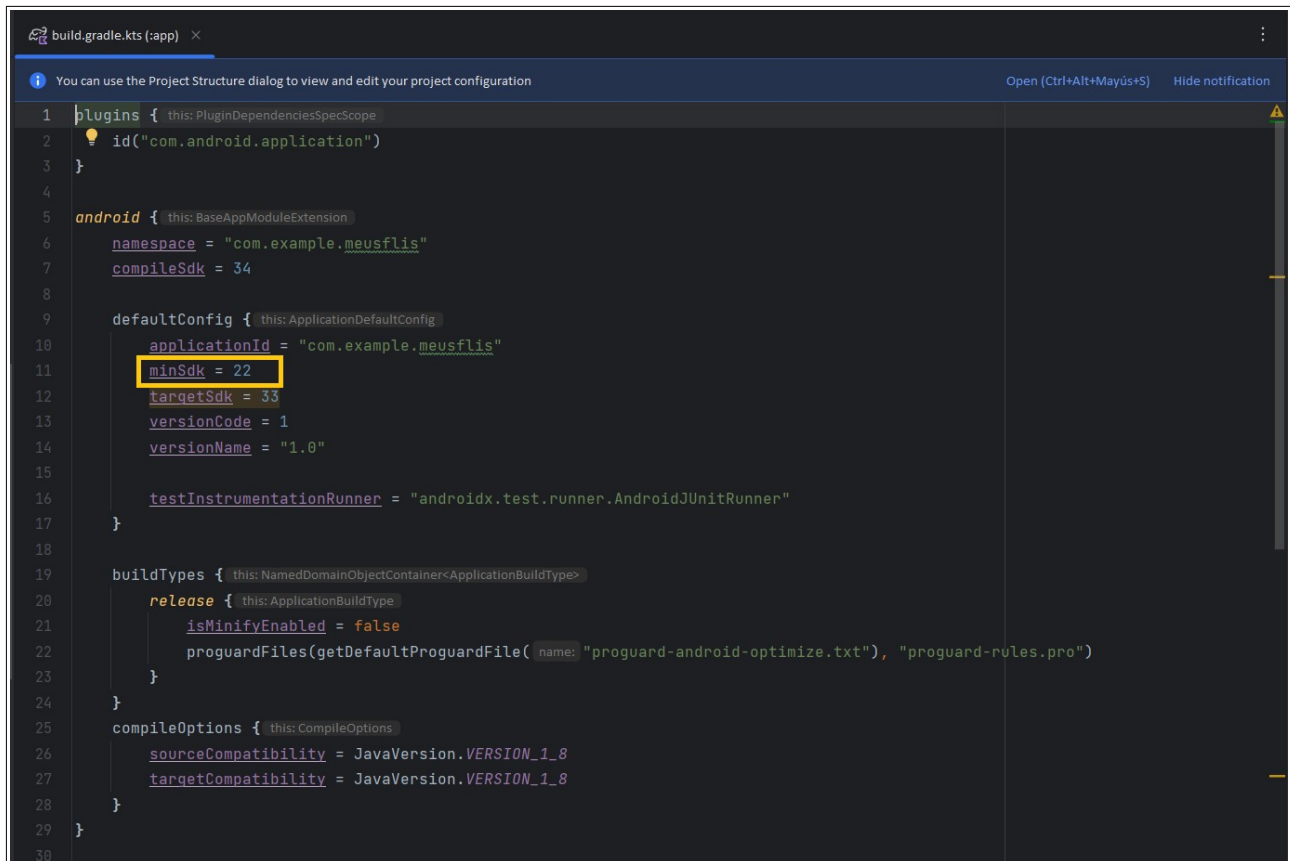


PMDM: ABRIL – JUNIO 2024 - APLICACIÓN RESUMEN

Se trata de realizar individualmente una aplicación con los siguientes requisitos como mínimo:

1. Debe poder ejecutarse en terminales con API 22 o superior.



```
1 plugins { this: PluginDependenciesSpecScope
2     id("com.android.application")
3 }
4
5 android { this: BaseAppModuleExtension
6     namespace = "com.example.meusflis"
7     compileSdk = 34
8
9     defaultConfig { this: ApplicationDefaultConfig
10         applicationId = "com.example.meusflis"
11         minSdk = 22
12         targetSdk = 33
13         versionCode = 1
14         versionName = "1.0"
15
16         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
17     }
18
19     buildTypes { this: NamedDomainObjectContainer<ApplicationBuildType>
20         release { this: ApplicationBuildType
21             isMinifyEnabled = false
22             proguardFiles(getDefaultProguardFile("name", "proguard-android-optimize.txt"), "proguard-rules.pro")
23         }
24     }
25
26     compileOptions { this: CompileOptions
27         sourceCompatibility = JavaVersion.VERSION_1_8
28         targetCompatibility = JavaVersion.VERSION_1_8
29     }
30 }
```

2. Debe constar de al menos dos *activities* que “dialoguen” entre sí; es decir, debe haber envío y/o retorno de información. Se empleará el mecanismo de *intents* explícitos.

```

LoginActivity.java  CatalogueActivity.java  ProfileActivity.java
168      * @param item El ítem del menú seleccionado.
169      * @return true si el ítem fue manejado, false de lo contrario.
170      */
171      @Override
172      public boolean onOptionsItemSelected(MenuItem item) {
173          Intent intent;
174          switch (item.getItemId()) {
175              case R.id.option_profile:
176                  intent = new Intent( packageContext: CatalogueActivity.this, ProfileActivity.class);
177                  intent.putExtra( name: "email", email);
178                  startActivity(intent);
179                  return true;
180          }

```

```

LoginActivity.java  CatalogueActivity.java  ProfileActivity.java
117
118      btnBackProfile.setOnClickListener(new View.OnClickListener() {
119          @Override
120          public void onClick(View v) {
121              Intent intent = new Intent( packageContext: ProfileActivity.this, CatalogueActivity.class);
122              intent.putExtra( name: "email", email);
123              startActivity(intent);
124          }
125      });

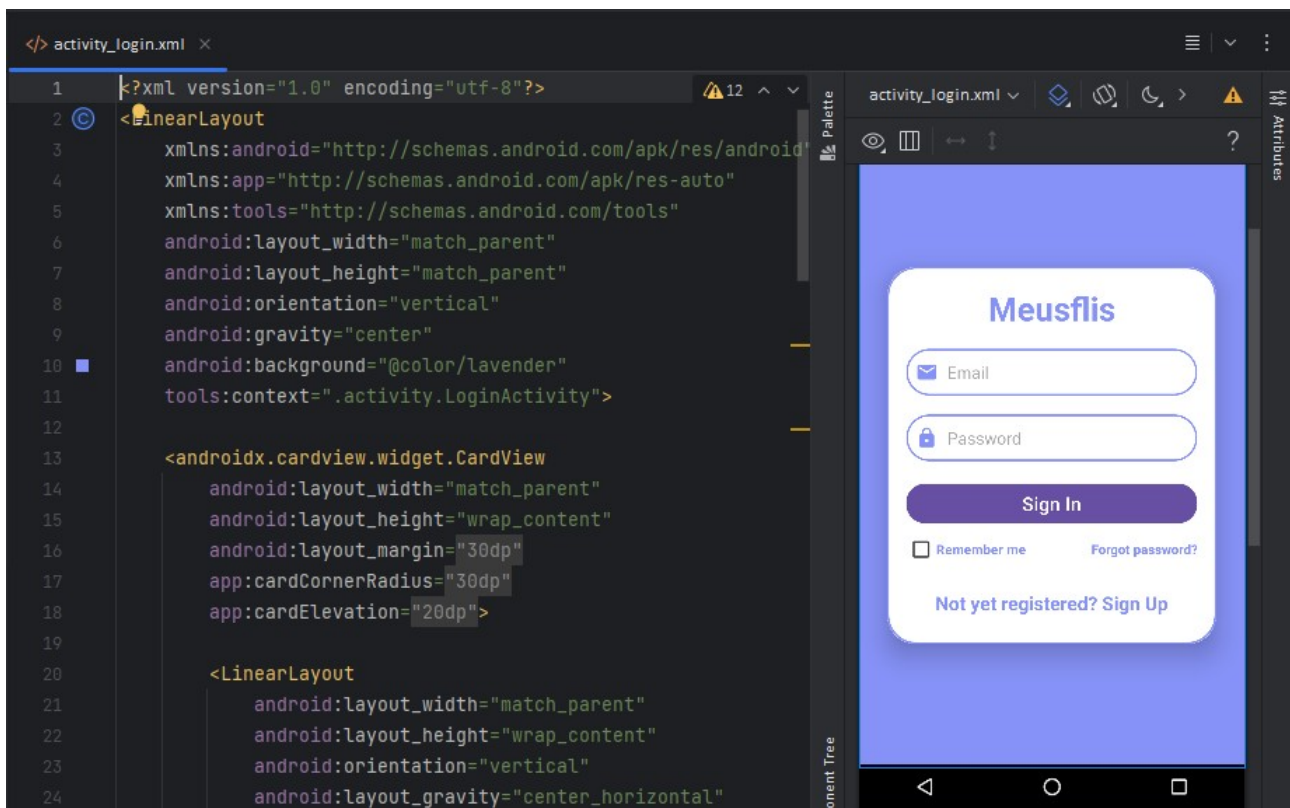
```

3. Empleo de dos llamadas a actividades mediante sendos intents implícitos.

```
LoginActivity.java CatalogueActivity.java ProfileActivity.java
212 /**
213  * Método para realizar una llamada telefónica.
214  */
215 private void makePhoneCall() {
216     String phoneNumber = getResources().getString(R.string.contactUsPhone);
217     if (ActivityCompat.checkSelfPermission(context, this, Manifest.permission.CALL_PHONE) == PackageManager.PERMISSION_GRANTED) {
218         Intent callIntent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:" + phoneNumber));
219         startActivity(callIntent);
220     }
221     else {
222         ActivityCompat.requestPermissions(activity, this, new String[]{Manifest.permission.CALL_PHONE});
223     }
224 }
```

```
LoginActivity.java CatalogueActivity.java ProfileActivity.java
304 break;
305
306 case R.id.ctx_anime:
307     openWebPage(selectedContent.getUrlAnime());
308     break;
309 }
310 return true;
311 }
312
313
314
315 /**
316  * Método para abrir una página web en un navegador.
317  * @param url La URL de la página web a abrir.
318  */
319 private void openWebPage(String url) {
320     Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
321     startActivity(intent);
322 }
```

4.Layouts con las vistas básicas: etiqueta/s de texto, caja/s de texto, botones, botones de radio, casilla/s de verificación, imagen/es...



5. Manejo de eventos asociados a las vistas que lo necesiten.

```
83 private void setListeners() {
84     btnSignIn.setOnClickListener(new View.OnClickListener() {
85         @Override
86         public void onClick(View view) {
87             if (!validateEmail() || !validatePassword()) {
88                 return;
89             }
90
91             String email = etLoginEmail.getText().toString();
92             String password = etLoginPassword.getText().toString();
93
94             boolean userExists = databaseHelper.checkUser(email, password);
95
96             if (userExists) {
97                 Toast.makeText(LoginActivity.this, "Sign Up Successful", Toast.LENGTH_SHORT).show();
98                 savePreferences(email);
99                 resetAttempts();
100                 Intent intent = new Intent(LoginActivity.this, CatalogueActivity.class);
101                 intent.putExtra("email", email);
102                 startActivity(intent);
103                 finish();
104             }
105             else {
106                 incrementAttempts();
107             }
108         }
109     });
110
111     tvForgotPassword.setOnClickListener(new View.OnClickListener() {
112         @Override
113         public void onClick(View view) { handleForgotPassword(); }
114     });
115
116     tvSignUpRedirect.setOnClickListener(new View.OnClickListener() {
117         @Override
118         public void onClick(View view) {
119             Intent intent = new Intent(LoginActivity.this, SignUpActivity.class);
120             startActivity(intent);
121         }
122     });
123 }
```

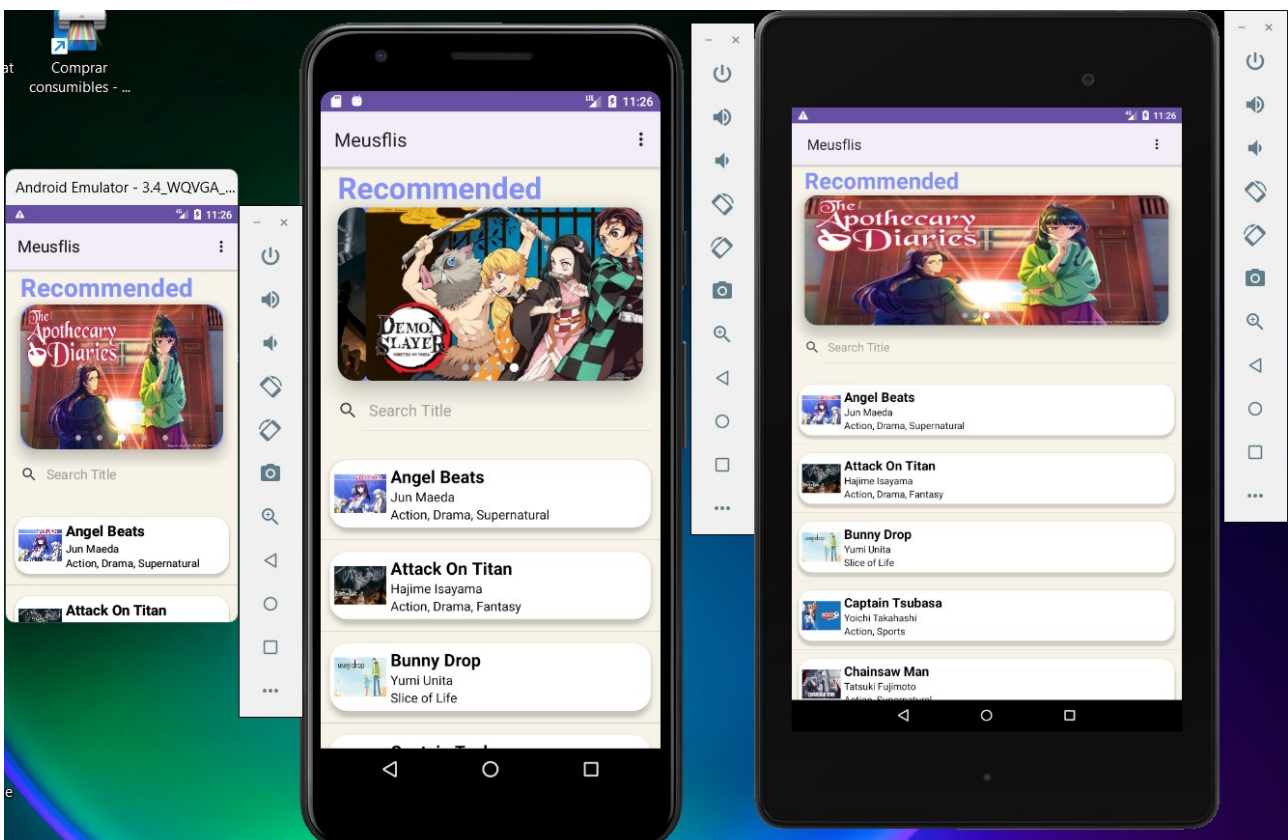
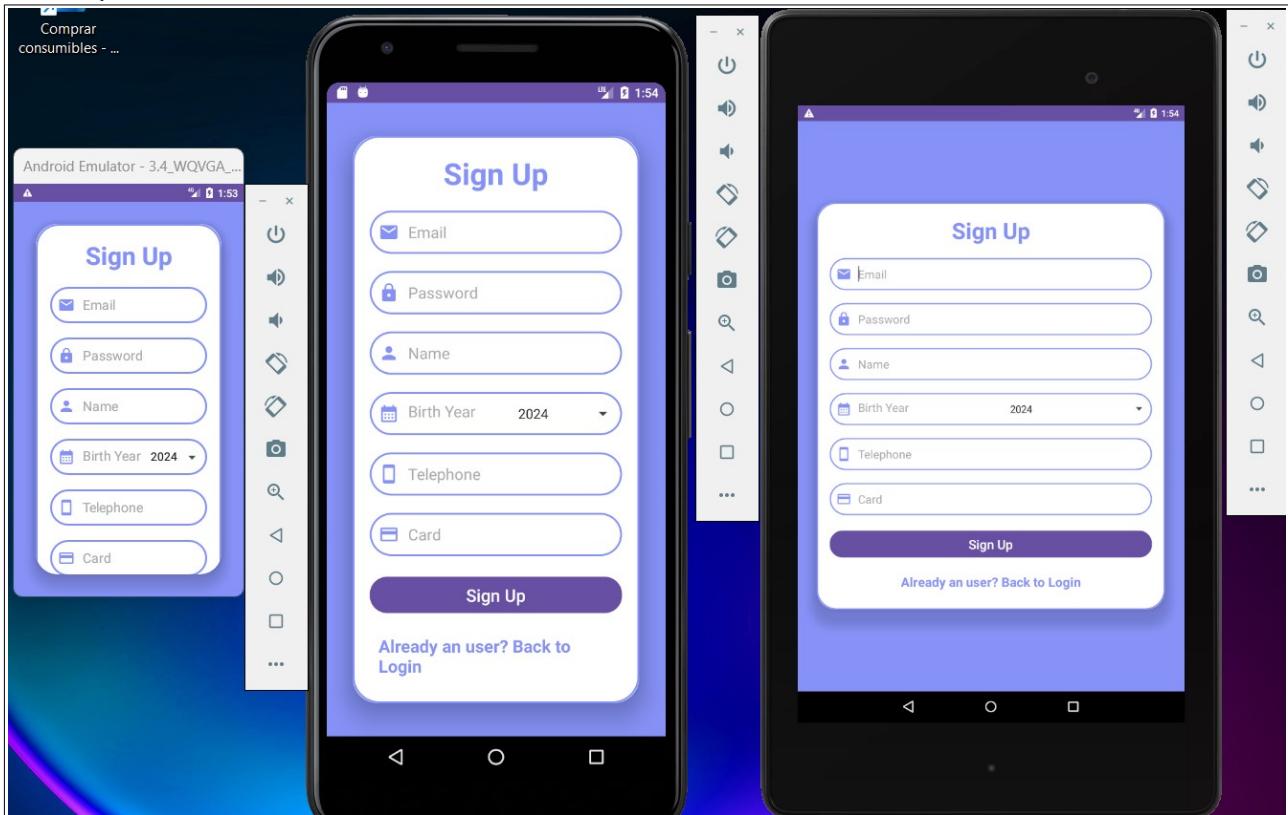
6.El aspecto de los layouts debe resultar razonablemente estético y visualizarse de forma similar en diferentes modelos de terminal.

Los dispositivos son:

3.4 WQVGA API 22

Pixel 3a API 23

Nexus 7 API 22



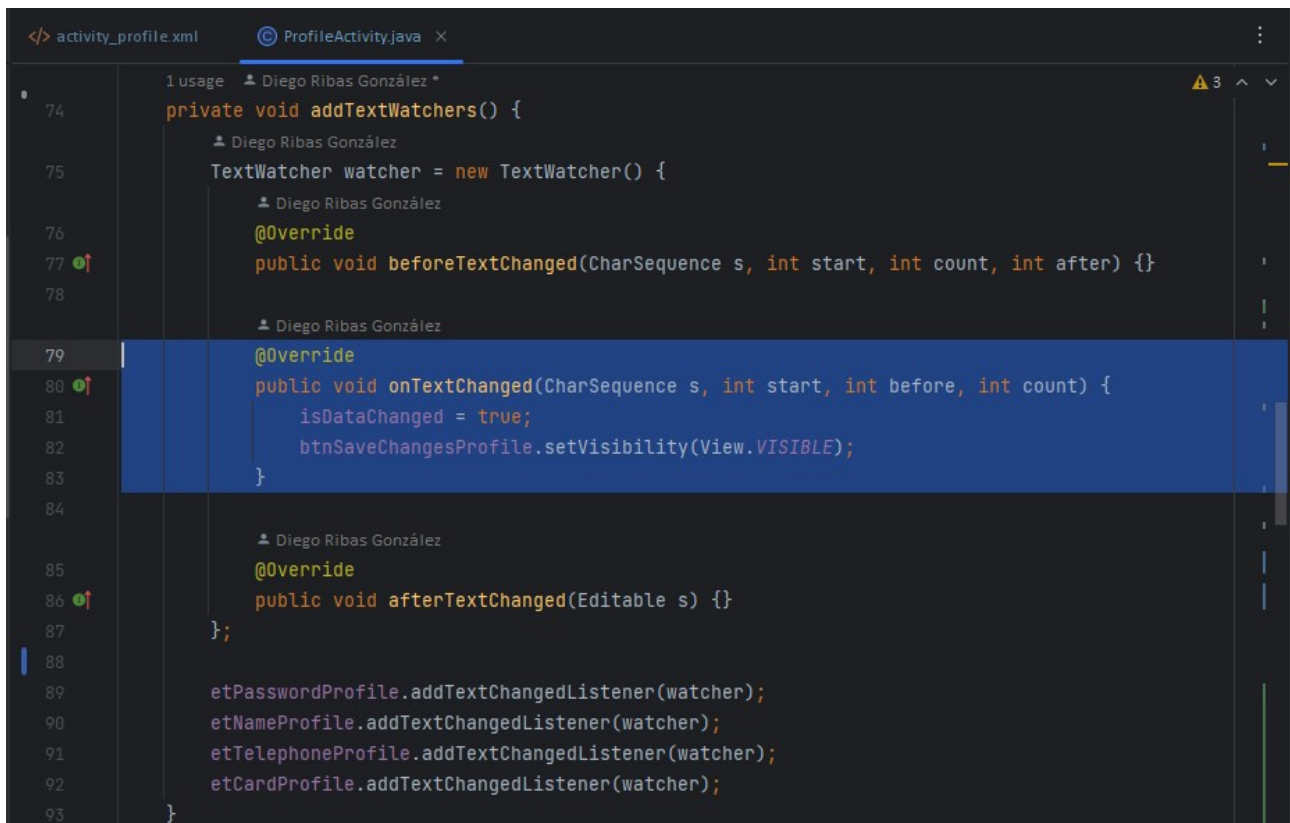
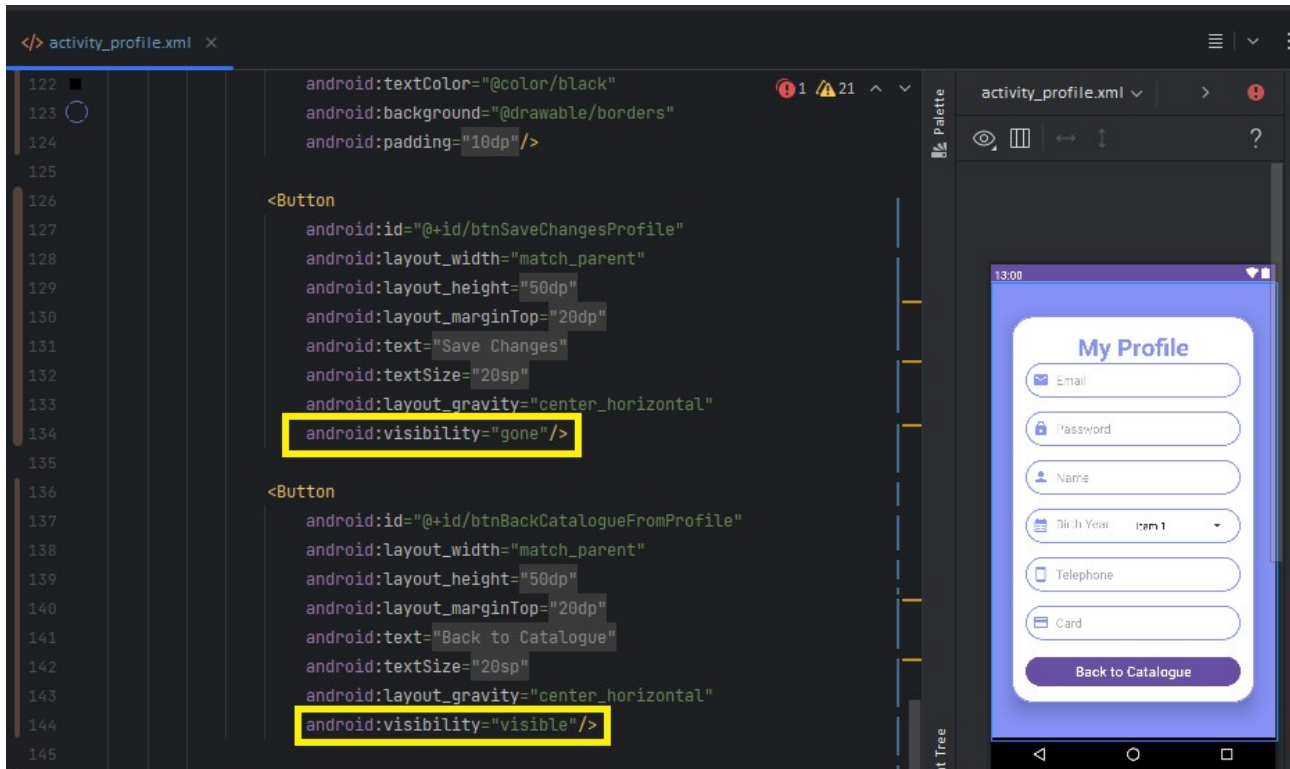
7. La entrada de datos en las/s caja/s de texto debe estar convenientemente filtrada (inputType...)

```
</> activity_login.xml    </> activity_signup.xml x
37
38      <EditText
39        android:id="@+id/etSignUpEmail"
40        android:layout_width="match_parent"
41        android:layout_height="50dp"
42        android:layout_marginTop="20dp"
43        android:drawableStart="@drawable/baseline_email_24"
44        android:drawablePadding="10dp"
45        android:hint="Email"
46        android:textColor="@color/black"
47        android:inputType="textEmailAddress"
48        android:background="@drawable/borders"
49        android:padding="10dp"/>
50
51      <EditText
52        android:id="@+id/etSignUpPassword"
53        android:layout_width="match_parent"
54        android:layout_height="50dp"
55        android:layout_marginTop="20dp"
56        android:drawableStart="@drawable/baseline_lock_24"
57        android:drawablePadding="10dp"
58        android:hint="Password"
59        android:textColor="@color/black"
60        android:inputType="textPassword"
61        android:background="@drawable/borders"
62        android:padding="10dp"/>
63
64      <EditText
65        android:id="@+id/etSignUpName"
66        android:layout_width="match_parent"
67        android:layout_height="50dp"
68        android:layout_marginTop="20dp"
69        android:drawableStart="@drawable/baseline_person_24"
70        android:drawablePadding="10dp"
71        android:hint="Name"
72        android:textColor="@color/black"
73        android:inputType="textCapSentences"
74      />
```

8.Si se trabaja con números decimales se usará un formato correcto (con un núm limitado de decimales, acorde al dato en cuestión).

La aplicación no trabaja con datos decimales.

9.Cambio de visibilidad en algún/os elementos.



10.Un menú de opciones (mínimo 2 opciones)

```
<?xml version="1.0" encoding="utf-8"?>
<menu
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto">

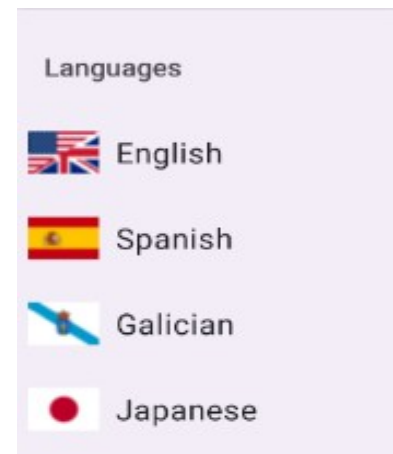
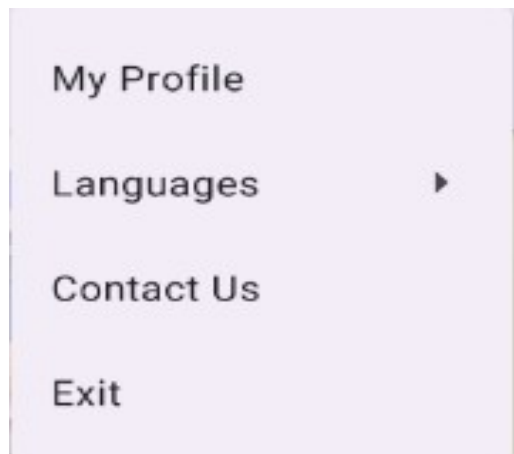
  <item android:id="@+id/option_profile" android:title="My Profile"/>

  <item android:id="@+id/option_languages" android:title="Languages" >
    <menu>
      <item android:id="@+id/option_english" android:title="English" android:icon="@drawable/english"
      <item android:id="@+id/option_spanish" android:title="Spanish" android:icon="@drawable/spanish"
      <item android:id="@+id/option_galician" android:title="Galician" android:icon="@drawable/galic
      <item android:id="@+id/option_japanese" android:title="Japanese" android:icon="@drawable/japan
    </menu>
  </item>

  <item android:id="@+id/option_contact" android:title="Contact Us"/>

  <item android:id="@+id/option_exit" android:title="Exit"/>

</menu>
```



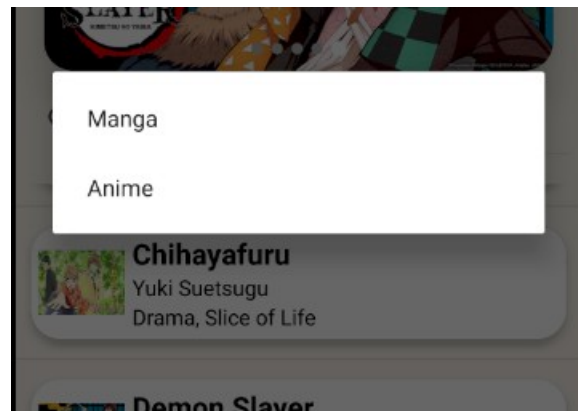
11.Un menú contextual (mínimo 2 opciones) aplicado a la vista/s que se considere conveniente.

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"

    <item android:id="@+id/ctx_manga" android:title="Manga"/>

    <item android:id="@+id/ctx_anime" android:title="Anime"/>

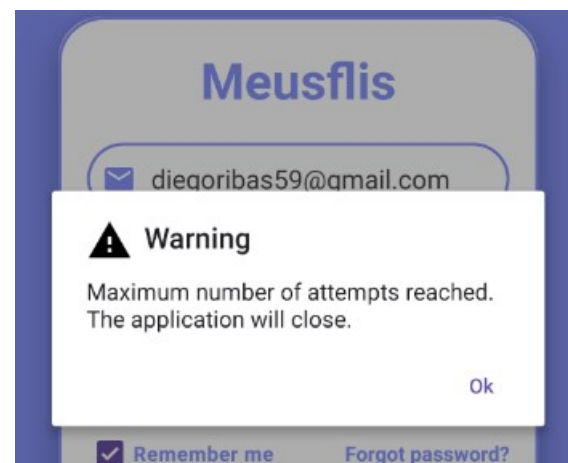
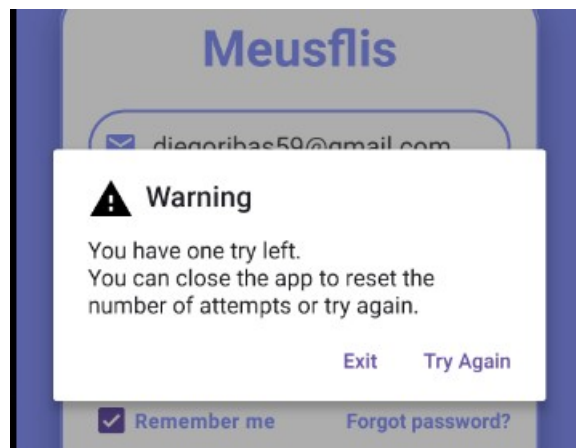
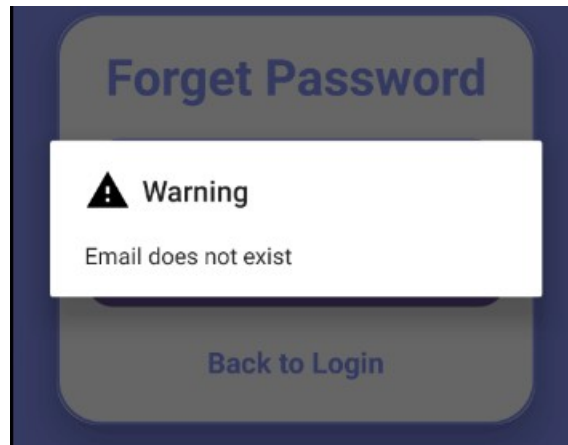
</menu>
```



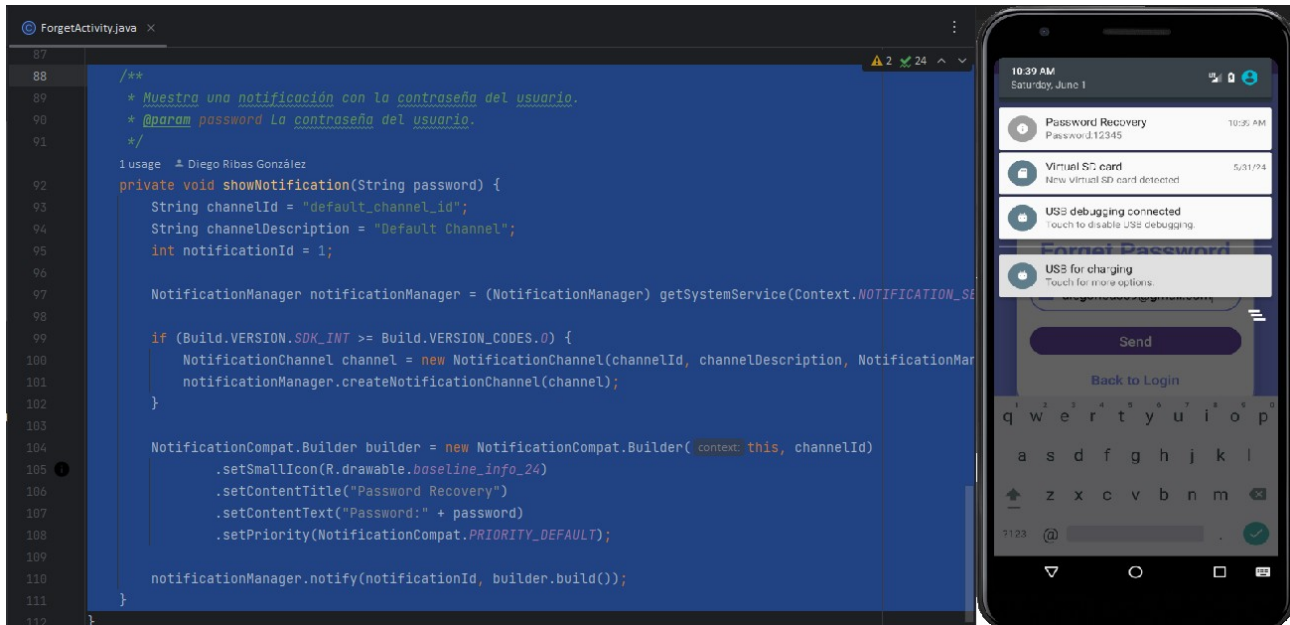
12. Notificaciones al usuario mediante Toast

```
        if (isUpdated) {  
            btnSaveChangesProfile.setVisibility(View.GONE);  
            Toast.makeText(context, ProfileActivity.this, "Profile updated successfully", Toast.LENGTH_SHORT).show();  
        }  
        else {  
            Toast.makeText(context, ProfileActivity.this, "Profile update failed", Toast.LENGTH_SHORT).show();  
        }  
    }  
}  
});
```

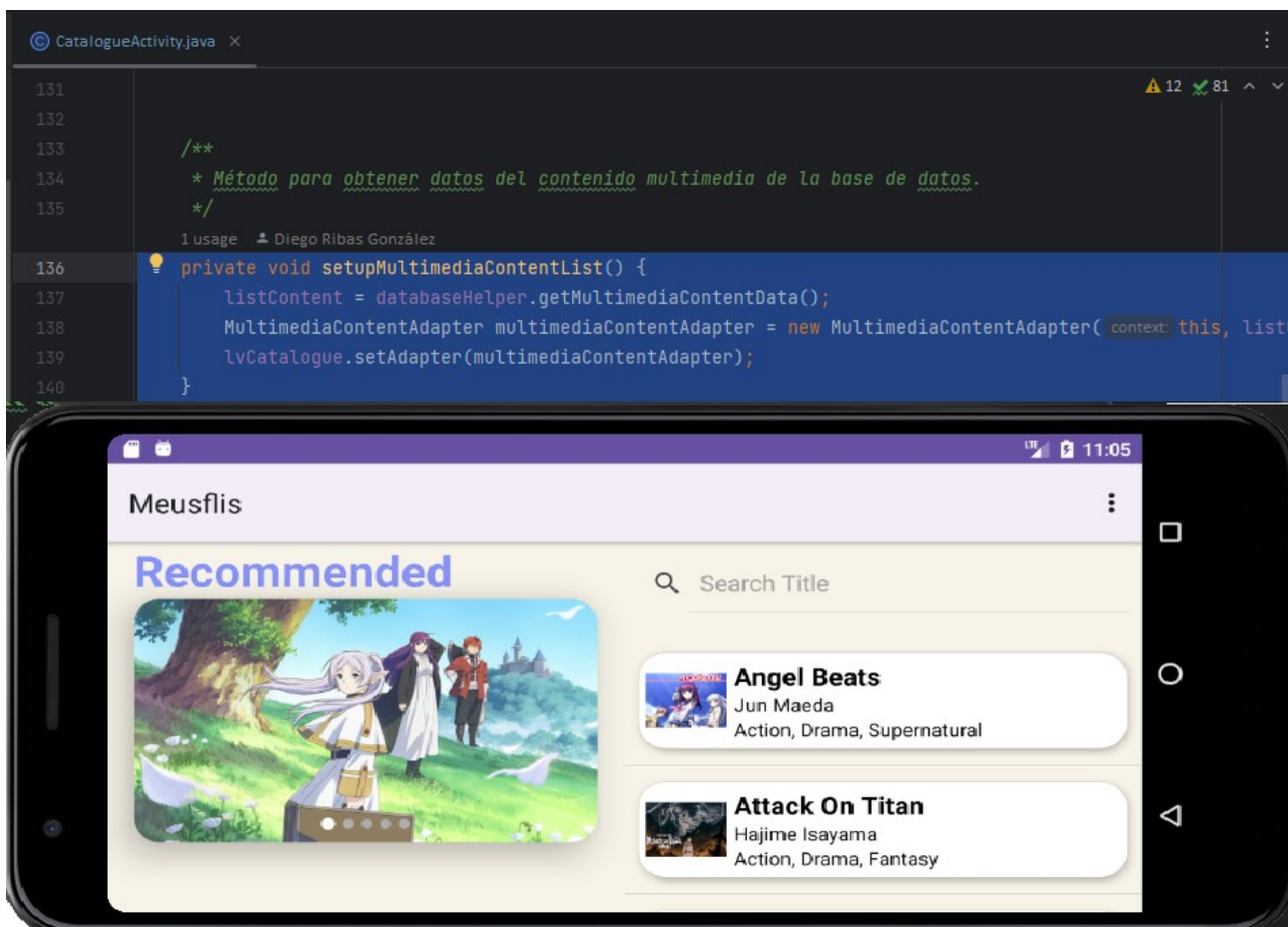
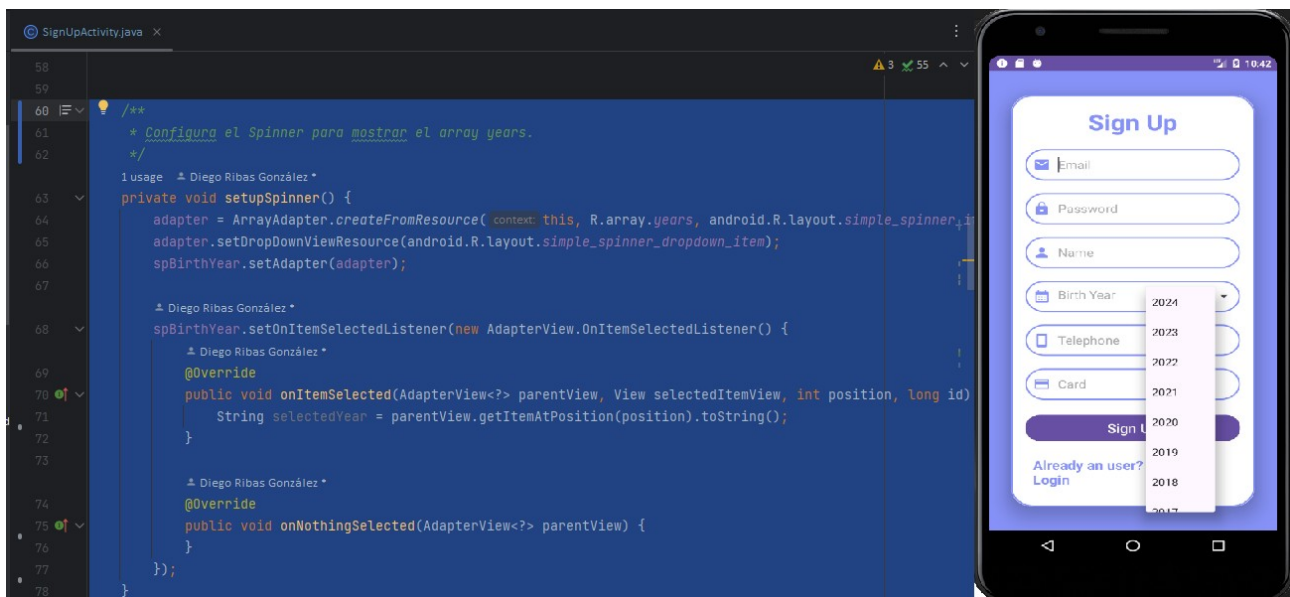
13. Notificaciones al usuario mediante ventanas de diálogo de los tres tipos básicos (mensaje, botón y varios botones). Se valorará la utilización de ventana de diálogo de selección múltiple.



14. Notificación al usuario en la barra de estado.



15. Dos listas de selección: un spinner y una listview personalizada (no puede tener una estructura idéntica a las listas que hemos trabajado en clase)



16. Manejo de información permanente mediante las dos formas vistas: preferencias compartidas y base de datos SQLite.

```
CatalogueActivity.java  LoginActivity.java x
63
64
65
66  /**
67   * Método para cargar las preferencias guardadas.
68   */
69   1 usage  ⚡ Diego Ribas González
70   private void loadPreferences() {
71       SharedPreferences preferences = getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
72       String savedEmail = preferences.getString(PREF_EMAIL, defValue: null);
73       if (savedEmail != null) {
74           etLoginEmail.setText(savedEmail);
75           chkRemember.setChecked(true);
76       }
77   }
```

```
DatabaseHelper.java  DatabaseConstants.java
1  package com.example.meusflis.database;
2
3  > import ...
19
15 usages  ⚡ Diego Ribas González *
20  public class DatabaseHelper extends SQLiteOpenHelper {
21      1 usage
22      private static final String DATABASE_NAME = "Database.db";
23      1 usage
24      private static final int DATABASE_VERSION = 1;
25
26      5 usages  ⚡ Diego Ribas González
27      public DatabaseHelper(Context context) {
28          super(context, DATABASE_NAME, factory: null, DATABASE_VERSION);
29      }
30
31      ⚡ Diego Ribas González
32      @Override
33      public synchronized SQLiteDatabase getReadableDatabase() { return super.getReadableDatabase(); }
34
35      ⚡ Diego Ribas González
36      @Override
37      public synchronized SQLiteDatabase getWritableDatabase() { return super.getWritableDatabase(); }
38  }
```


17. Las operaciones mínimas que se realizarán contra la base de datos serán inserción y consulta. Se valorará también la existencia de operaciones de borrado y/o modificación.

```
DatabaseHelper.java x
83
84 /**
85  * Método para insertar un nuevo usuario.
86  * @param email El correo electrónico del usuario.
87  * @param password La contraseña del usuario.
88  * @param name El nombre del usuario.
89  * @param telephone El número de teléfono del usuario.
90  * @param card La tarjeta del usuario.
91  * @return true si el usuario fue insertado exitosamente, false de lo contrario.
92  */
93 public boolean insertUser(String email, String password, String name, String birthyear, String telephone, String card) {
94     SQLiteDatabase db = this.getWritableDatabase();
95     ContentValues values = new ContentValues();
96
97     values.put(DatabaseConstants.COLUMN_EMAIL, email);
98     values.put(DatabaseConstants.COLUMN_PASSWORD, password);
99     values.put(DatabaseConstants.COLUMN_NAME, name);
100    values.put(DatabaseConstants.COLUMN_BIRTHYEAR, birthyear);
101    values.put(DatabaseConstants.COLUMN_TELEPHONE, telephone);
102    values.put(DatabaseConstants.COLUMN_CARD, card);
103
104    long result = db.insert(DatabaseConstants.TABLE_USER, null, values);
105    db.close();
106
107    return result != -1;
108 }
```

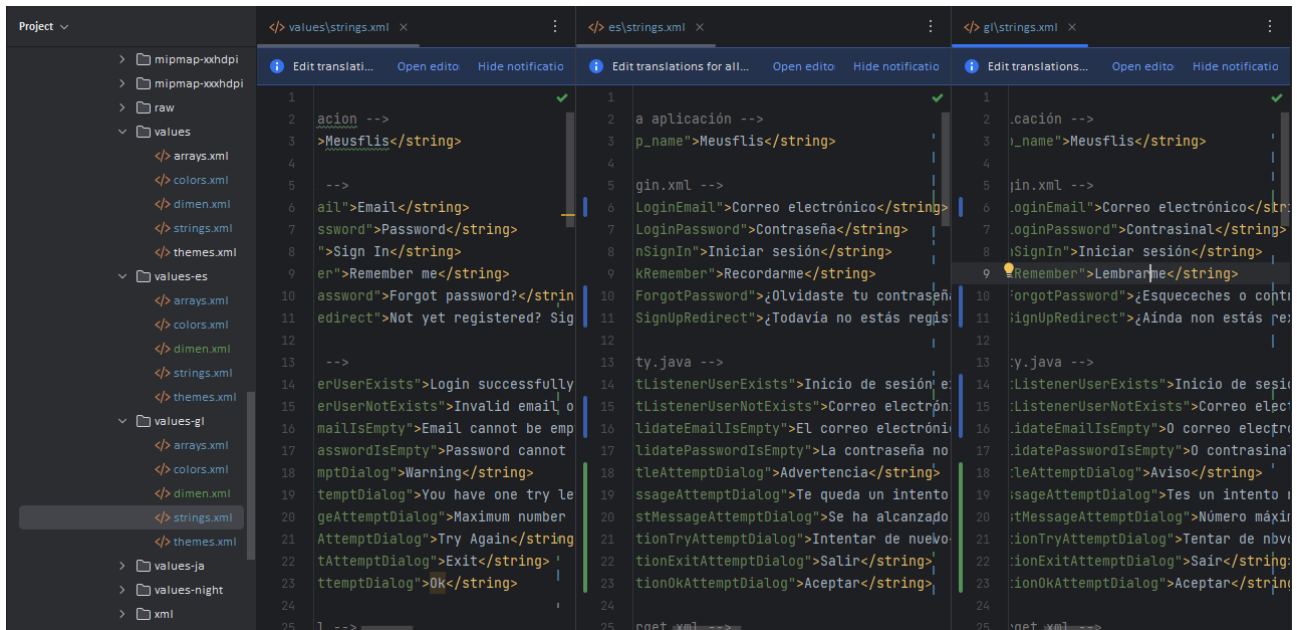
```
DatabaseHelper.java x
38
39
40 /**
41  * Método para verificar si el usuario existe.
42  * @param email El correo electrónico del usuario.
43  * @param password La contraseña del usuario.
44  * @return true si el usuario existe, false de lo contrario.
45  */
46 public boolean checkUser(String email, String password) {
47     SQLiteDatabase db = this.getReadableDatabase();
48     String[] columns = { DatabaseConstants.COLUMN_EMAIL };
49     String selection = DatabaseConstants.COLUMN_EMAIL + " = ? AND " + DatabaseConstants.COLUMN_PASSWORD + " = ?";
50     String[] selectionArgs = { email, password };
51     Cursor cursor = null;
52
53     try {
54         cursor = db.query(
55             DatabaseConstants.TABLE_USER,
56             columns, selection, selectionArgs,
57             null,
58             null,
59             null);
60
61         int cursorCount = cursor.getCount();
62         cursor.close();
63         db.close();
64
65         if (cursorCount > 0) {
```

18. Empleo de diferentes colores y dimensiones que se implementarán con sus correspondientes recursos.

```
</> colors.xml x </> dimen.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <color name="purple_200">#FFBB86FC</color>
4     <color name="purple_500">#FF6200EE</color>
5     <color name="purple_700">#FF3700B3</color>
6     <color name="teal_200">#FF03DAC5</color>
7     <color name="teal_700">#FF018786</color>
8     <color name="black">#FF000000</color>
9     <color name="white">#FFFFFFFF</color>
10    <color name="background">#f7f3e8</color>
11    <color name="lavender">#8692F7</color>
12    <color name="grey">#FF424242</color>
13 </resources>
```

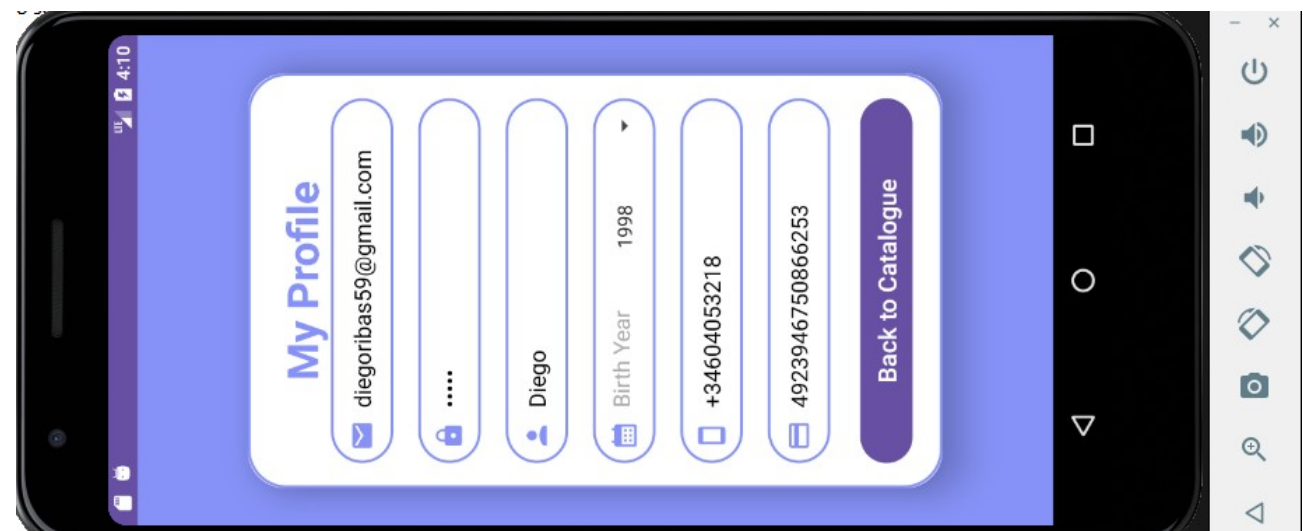
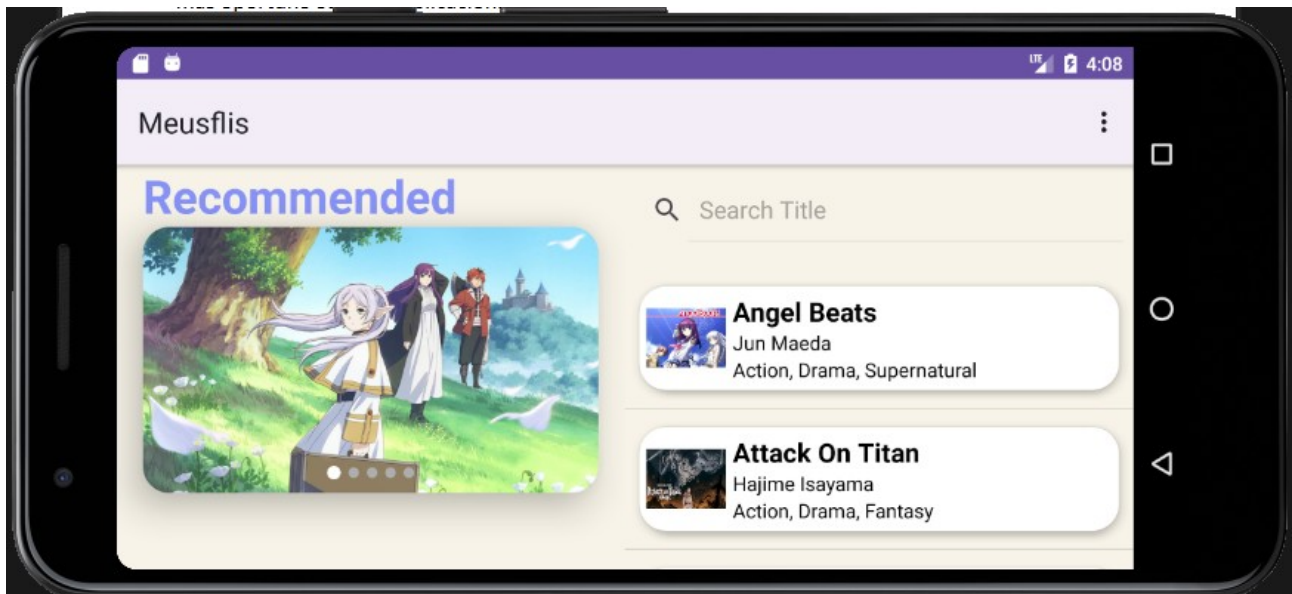
```
</> colors.xml </> dimen.xml x
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <!-- activity_login.xml -->
4     <dimen name="loginCardView_layoutMargin">30dp</dimen>
5     <dimen name="loginCardView_cardCornerRadius">30dp</dimen>
6     <dimen name="loginCardView_cardElevation">20dp</dimen>
7     <dimen name="loginPadding">20dp</dimen>
8     <dimen name="loginTvTitleSize">35sp</dimen>
9     <dimen name="loginEt_layoutHeight">50dp</dimen>
10    <dimen name="loginEt_layoutMarginTop">20dp</dimen>
11    <dimen name="loginEt_drawablePadding">10dp</dimen>
12    <dimen name="loginEt_padding">10dp</dimen>
13    <dimen name="loginBtn_layoutHeight">50dp</dimen>
14    <dimen name="loginBtn_layoutMarginTop">20dp</dimen>
15    <dimen name="loginBtn_cornerRadius">20dp</dimen>
16    <dimen name="loginBtn_textSize">20sp</dimen>
17    <dimen name="loginTvSignUp_layoutMarginTop">10dp</dimen>
18    <dimen name="loginTvSignUp_textSize">20sp</dimen>
19    <dimen name="loginTvSignUp_padding">10dp</dimen>
20
21    <!-- activity_catalogue.xml -->
22    <dimen name="menuPadding">10dp</dimen>
23
24    <!-- activity_catalogue.xml -->
25    <dimen name="catalogueTitle_layoutMarginStart">20dp</dimen>
26    <dimen name="catalogueTvTitleSize">35sp</dimen>
27    <dimen name="catalogueCardView_layoutHeight">200dp</dimen>
```

19. Internacionalización (la aplicación debe ejecutarse en español y gallego cuando éstos sean los idiomas del terminal. En inglés, por defecto.



20.El cambio de orientación de la pantalla debe estar correctamente gestionado (es decir, la app no puede romper su ejecución ni ejecutarse mal) o, en su caso, se limitará la ejecución al formato más oportuno según la aplicación.

CatalogueActivity es la único layout con versión land, el resto tienen declarado por AndroidManifest que solo pueden ser portrait.



21.La aplicación NO PUEDE ROMPER por errores de ejecución.

22.La temática será a elección de cada uno.

23.La entrega de la aplicación debe hacerse dentro del plazo indicado por la profesora.

24.Debe entregarse el código junto con un vídeo demostrativo del funcionamiento de la app.

25.Se podrán añadir, y serán valorados, otros elementos cualesquiera, vistos en clase o no, aunque no figuren entre los requisitos mínimos anteriores.

SearchView

```
/**
 * Método para configurar el SearchView y su listener.
 */
1 usage  👤 Diego Ribas González
private void setupSearchView() {
    1 usage  👤 Diego Ribas González
    svSearchMultimediaContent.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        1 usage  👤 Diego Ribas González
        @Override
        public boolean onQueryTextSubmit(String query) { return false; }

        1 usage  👤 Diego Ribas González
        @Override
        public boolean onQueryTextChange(String newText) {
            ArrayList<MultimediaContent> filteredList = databaseHelper.filterByTitle(newText);
            MultimediaContentAdapter multimediaContentAdapter = new MultimediaContentAdapter(context, Catal
            lvCatalogue.setAdapter(multimediaContentAdapter);
            return false;
        }
    });
}
```

ImageSlider

```
/**
 * Método para cargar las portadas de los contenidos más valorados en el ImageSlider.
 */
1 usage  👤 Diego Ribas González
private void loadTopContentCovers() {
    List<byte[]> topContentCovers = databaseHelper.getTopLikedContentCovers(limit: 5);

    if (topContentCovers == null) {
        topContentCovers = new ArrayList<>();
    }

    ArrayList<SlideModel> slideModels = new ArrayList<>();

    for (byte[] coverImage : topContentCovers) {
        Bitmap bitmap = BitmapFactory.decodeByteArray(coverImage, offset: 0, coverImage.length);
        Uri imageUri = saveImageToCache(bitmap);
        if (imageUri != null) {
            slideModels.add(new SlideModel(imageUri.toString(), ScaleTypes.FIT));
        }
    }

    thisTopMultimediaContent.setImageList(slideModels, ScaleTypes.FIT);
}
```

Cambio de Idiomas

```
/**
 * Método para cambiar la configuración de idioma de la aplicación.
 * @param lang El código del idioma a configurar.
 */
4 usages  👤 Diego Ribas González
protected void setLocale(String lang) {
    Locale locale = new Locale(lang);
    Locale.setDefault(locale);
    Resources res = getResources();
    Configuration config = new Configuration(res.getConfiguration());
    config.locale = locale;
    DisplayMetrics dm = res.getDisplayMetrics();
    res.updateConfiguration(config, dm);

    Intent refresh = new Intent(packageContext: this, CatalogueActivity.class);
    startActivity(refresh);
    finish();
}
```

LottieAnimation

```
10
11 </> public class SplashActivity extends AppCompatActivity {
12     private Handler handler = new Handler();
13     private Runnable runnable;
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_splash);
18
19         runnable = new Runnable() {
20             @Override
21             public void run() {
22                 Intent intent = new Intent(packageContext, LoginActivity.class);
23                 startActivity(intent);
24                 finish();
25             }
26         };
27         handler.postDelayed(runnable, delayMillis: 5000);
28     }
29
30     @Override
31     protected void onDestroy() {
32         super.onDestroy();
33         handler.removeCallbacks(runnable);
34     }
35 }
```