TEMA 3. API

Práctica 1 Consumir servicios Web

Objetivo específico	Aprender a como consumir un servicio web
Recursos	Android Studio XML Kotlin https://www.omdbapi.com/?apikey=d4a899e2&s=Harry%20Potter
Actividades	Prepararemos la aplicación para que recibamos la información de un servicio web
Producto(s)	Al final de la práctica deberá desplegar la información que se consumió en el servicio web

PASO 1. Abrir Android Studio.

Abrir Android Studio

PASO 2. Elegir la ProyectoFinal

Seleccionar ProyectoFinal

PASO 3. Configuración del proyecto

Abriremos build.gradle(module) y agregaremos las siguientes librerías para poder consumir servicios web y que sea más fácil manipularlo.

```
// Retrofit
implementation "com.squareup.retrofit2:retrofit:2.9.0"
implementation "com.squareup.retrofit2:converter-gson:2.9.0"
//Corrutinas
implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.6'
// Okhttp3 for the POST requests
```

```
implementation 'com.squareup.okhttp3:okhttp:4.9.0'
// Gson (To convert raw JSON to pretty JSON)
implementation 'com.google.code.gson:gson:2.8.6'
//Logger
implementation 'com.orhanobut:logger:2.2.0'
// ViewModel
implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1"
// LiveData
implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.3.1"
// Activity
implementation "androidx.activity:activity-ktx:1.2.2"
```

PASO 4. Crearemos las clases para consumir el servicio web

Primero modificaremos nuestro objeto Movie que ya habíamos creado previamente.

```
data class Movie(
    @SerializedName("Search") val data: MutableList<DataMovie>?,
    @SerializedName("totalResults") var totalResults: String,
    @SerializedName("Response") var Response: String
)

data class DataMovie(
    @SerializedName("Title") var Title: String,
    @SerializedName("Year") var Year: String,
    @SerializedName("imdbID") var imdbID: String,
    @SerializedName("Type") var Type: String,
    @SerializedName("Poster") var Poster: String
)
```

Crearemos un nuevo objeto llamado InfoMovie, este objeto lo usaremos más tarde.

```
data class InfoMovie(
    @SerializedName("Title") var Title: String,
    @SerializedName("Year") var Year: String,
    @SerializedName("Released") var Released: String,
    @SerializedName("Runtime") var Runtime: String,
    @SerializedName("Director") var Director: String,
    @SerializedName("Actors") var Actors: String,
    @SerializedName("Plot") var Plot: String,
    @SerializedName("Poster") var Poster: String,
    @SerializedName("Type") var Type: String,
    @SerializedName("Metascore") var Metascore: String,
    @SerializedName("Ratings") var Ratings: MutableList<Ratings>?
)

data class Ratings(
    @SerializedName("Source") var Source: String,
    @SerializedName("Value") var Value: String,
)
```

Crearemos un object con el nombre RetrofitHelper.

```
object RetrofitHelper {
    fun getRetrofit(): Retrofit {
        val urlBase = "https://www.omdbapi.com/"
        return Retrofit.Builder()
```

```
            .baseUrl(urlBase)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
    }
}
```

Después crearemos un interface llamado ApiClient.

```
interface ApiClient {

    @GET(".")
    suspend fun getMovies(@Query("apikey") apikey: String,
                          @Query("s") s: String): retrofit2.Response<Movie>

    @GET(".")
    suspend fun getInfoMovie(@Query("apikey") apikey: String,
                             @Query("i") i: String):
retrofit2.Response<InfoMovie>

}
```

Ahora crearemos una clase llamada ApiService.

```
class ApiService {

    private val retrofit = RetrofitHelper.getRetrofit()
    lateinit var message: String

    suspend fun getMovies(apikey: String, busqueda: String): Movie? {
        message = ""
        return withContext(Dispatchers.IO) {
            val response =
retrofit.create(ApiClient::class.java).getMovies(apikey, busqueda)

            val code = response.code()
            Log.d("movies", "$code")

            if (!response.isSuccessful) {
                try {
                    val jObjError =
JSONObject(response.errorBody()!!.string())
                    message = jObjError.getString("message")
                    Log.d("movies", "$message")
                } catch (e: Exception) {
                    message = e.message.toString()
                    Log.d("movies", "$message")
                }
            }
            response.body()
        }
    }

    suspend fun getInfoMovie(apikey: String, imdbID: String): InfoMovie? {
        message = ""
        return withContext(Dispatchers.IO) {
            val response =
retrofit.create(ApiClient::class.java).getInfoMovie(apikey, imdbID)
```

```
            val code = response.code()
            Log.d("movies", "$code")

            if (!response.isSuccessful) {
                try {
                    val jObjError =
JSONObject(response.errorBody()!!.string())
                    message = jObjError.getString("message")
                    Log.d("movies", "$message")
                } catch (e: Exception) {
                    message = e.message.toString()
                    Log.d("movies", "$message")
                }
            }
            response.body()
        }
    }
}
```

Crearemos otra clase llamada Provider.

```
class Provider {
    companion object {
        lateinit var getMovies: Movie
        lateinit var getInfoMovie: InfoMovie
    }
}
```

Ahora crearemos una clase más llamada ServiceRepository.

```
class ServiceRepository {

    private val api = ApiService()

    suspend fun getMovies(apikey: String, busqueda: String): Movie? {
        val response = api.getMovies(apikey, busqueda)
        val movies = response
        if (movies != null) {
            Provider.getMovies = movies
        }
        return response
    }

    suspend fun getInfoMovie(apikey: String, imdbID: String): InfoMovie? {
        val response = api.getInfoMovie(apikey, imdbID)
        val info = response
        if (info != null) {
            Provider.getInfoMovie = info
        }
        return response
    }
}
```

Ahora crearemos nuestro useCase llamada GetMovies.

```
class GetMovies {
    private val repository = ServiceRepository()
    suspend operator fun invoke(apikey: String, busqueda: String) : Movie? =
repository.getMovies(apikey, busqueda)
}
```

Crearemos un viewModel donde usaremos hilos para ayudarnos a esperar la respuesta de la petición que acabamos de realizar para poder recibir un resultado y poderlo pasar a nuestro activity y poderlo mostrar en pantalla.

```
class MainModel: ViewModel() {

    val responseMoviesList = MutableLiveData<MutableList<DataMovie>>()

    val useCases = GetMovies()

    fun getMovies(busqueda: String) {
        viewModelScope.launch {
            coroutineScope {
                var moviesList = emptyList<DataMovie>().toMutableList()
                val movieAsync = async { useCases("d4a899e2", busqueda) }
                val movieResponse = movieAsync.await()
                val movies = movieResponse

                if (movies != null) {
                    for (movie in movies.data!!) {
                        moviesList.add(movie)
                    }
                }
                responseMoviesList.postValue(moviesList)
            }
        }
    }
}
```

Ahora modificaremos nuestro MainActivity para poder desplegar el resultado que obtuvimos del servicio web.

```
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    private lateinit var searchView : SearchView
    private lateinit var recyclerView : RecyclerView

    var adapter : RecyclerAdapter = RecyclerAdapter()

    private var movies : MutableList<DataMovie> = mutableListOf()
    private var matchedMovie: MutableList<DataMovie> = mutableListOf()

    private val mainModel : MainModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
```

```
        searchView = binding.searchView
        recyclerView = binding.rvMovieList

        setUpRecyclerView()
    }

    fun setUpRecyclerView() {
        getMovies("Harry Potter")
        recyclerView.setHasFixedSize(true)
        recyclerView.layoutManager = LinearLayoutManager(this)
    }

    private fun search(text: String?) {
        matchedMovie = arrayListOf()

        text?.let {
            movies.forEach { movie ->
                if (movie.Title.contains(text, true)) {
                    matchedMovie.add(movie)
                }
            }
            updateRecyclerView()
            if (matchedMovie.isEmpty()) {
                Toast.makeText(this, "No match found!",
Toast.LENGTH_SHORT).show()
            }
            updateRecyclerView()
        }
    }

    private fun updateRecyclerView() {
        adapter.RecyclerAdapter(matchedMovie, this)
        recyclerView.adapter = adapter
        adapter.notifyDataSetChanged()
    }

    fun getMovies(busqueda: String) {
        mainModel.getMovies(busqueda)
        mainModel.responseMoviesList.observe(this, {
            Log.d("movies", "entra")
            Log.d("movies", "it = $it")
            if (it.isNotEmpty()) {
                for (movie in it) {
                    movies.add(movie)
                }
            }
            Log.d("movies", "movies = $movies")
            setUpAdapter(movies)
        })
    }

    private fun setUpAdapter(moviesList: List<DataMovie>) {
        adapter.RecyclerAdapter(moviesList, this)
        recyclerView.adapter = adapter

        searchView.setOnQueryTextListener(object :
SearchView.OnQueryTextListener{
```

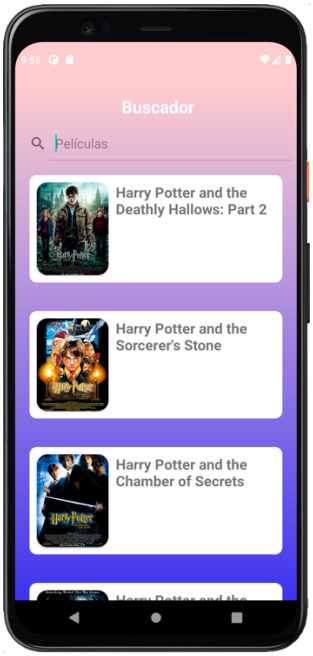
```
            override fun onQueryTextSubmit(query: String?): Boolean {
                searchView.clearFocus()
                search(query)
                return true
            }

            override fun onQueryTextChange(newText: String?): Boolean {
                search(newText)
                return true
            }
        })
    }
}
```

Actualizamos nuestro RecyclerAdapter

```
class RecyclerAdapter : RecyclerView.Adapter<RecyclerAdapter.ViewHolder>() {

    var movies: List<DataMovie>  = listOf()
    lateinit var context: Context

    fun RecyclerAdapter(movies : List<DataMovie>, context: Context){
        this.movies = movies
        this.context = context
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val item = movies.get(position)
        holder.bind(item, context)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ViewHolder {
        val layoutInflater = LayoutInflater.from(parent.context)
        return ViewHolder(layoutInflater.inflate(R.layout.item_movie, parent,
false))
    }

    override fun getItemCount(): Int {
        return movies.size
    }

    class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val title = view.findViewById(R.id.tvTitle) as TextView
        val poster = view.findViewById(R.id.ivPoster) as ImageView

        fun bind(movie:DataMovie, context: Context){
            title.text = movie.Title
            itemView.setOnClickListener(View.OnClickListener {
                val intent = Intent(context,
DetalleMovieActivity::class.java)
                intent.putExtra("imdbID", movie.imdbID)
                context.startActivity(intent)
            })
            Picasso.get().load(movie.Poster).into(poster)
        }
```

```
    }
}
```

Resultado



PASO 5. Modificaremos la segunda vista

Agregaremos un imageView, un textView para el título de la película, un apartado (cardView) para una rating y otro cardView para una breve descripción. Abriremos nuestro activity_detalle_movie.xml y reemplazamos el código por lo siguiente.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/gradient"
    tools:context=".DetalleMovieActivity">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/activity_detalle_tb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/transparente"
        app:layout_constraintTop_toTopOf="parent">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <ImageView
                android:id="@+id/button_back"
                android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:src="@drawable/ic_arrow_back_black_24dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:tint="@color/white"/>

        <TextView
            android:id="@+id/toolbar_detalle"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="Detalle"
            android:textColor="@color/white"
            android:textSize="24sp"
            android:textStyle="bold" />

    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.appcompat.widget.Toolbar>

<androidx.cardview.widget.CardView
    android:id="@+id/cv_poster"
    android:layout_width="100dp"
    android:layout_height="150dp"
    app:cardCornerRadius="15dp"
    android:layout_below="@id/activity_detalle_tb"
    android:layout_alignParentLeft="true"
    android:layout_marginLeft="25dp"
    android:layout_marginRight="15dp"
    android:layout_marginTop="35dp"
    android:layout_marginBottom="35dp">

    <ImageView
        android:id="@+id/ivPoster"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:adjustViewBounds="true"
        android:scaleType="centerCrop" />
</androidx.cardview.widget.CardView>


<TextView
    android:id="@+id/tvPelicula"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="@color/white"
    android:text="Nombre"
    android:textSize="18sp"
    android:textStyle="bold"
    android:layout_alignTop="@id/cv_poster"
    android:layout_toRightOf="@id/cv_poster"
    android:layout_alignParentRight="true"
    android:layout_marginRight="25dp"
    />

<androidx.cardview.widget.CardView
```

```
        android:id="@+id/cv_rating"
        android:layout_width="match_parent"
        android:layout_height="75dp"
        app:cardCornerRadius="15dp"
        android:layout_toRightOf="@id/cv_poster"
        android:layout_alignParentRight="true"
        android:layout_alignBottom="@id/cv_poster"
        android:layout_marginRight="25dp">

        <RatingBar
            android:id="@+id/ratingBar"
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:numStars="5"
            android:isIndicator="true"
            android:layout_gravity="top|center"
            android:rating="0.0"
            android:progressTint="@color/ratingColor"
            />

        <TextView
            android:id="@+id/tvRating"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="@color/black"
            android:textAlignment="center"
            android:text="Rating"
            android:textSize="18sp"
            android:layout_gravity="bottom"
            android:layout_marginBottom="10dp"
            />
    </androidx.cardview.widget.CardView>

    <androidx.cardview.widget.CardView
        android:id="@+id/cv_plot"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/cv_poster"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="25dp"
        android:layout_marginRight="25dp"
        app:cardCornerRadius="15dp">

        <TextView
            android:id="@+id/tvPlot"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_gravity="top"
            android:layout_marginRight="10dp"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="10dp"
            android:layout_marginBottom="30dp"
            android:text="Plot"
            android:textAlignment="center"
            android:textColor="@color/black"
            android:textSize="15sp" />
```

```
        <ImageView
            android:id="@+id/likeButton"
            android:layout_width="20dp"
            android:layout_height="20dp"
            android:src="@drawable/ic_heart"
            app:tint="@color/black"
            android:layout_gravity="bottom|end"
            android:layout_margin="10dp"/>
    </androidx.cardview.widget.CardView>

</RelativeLayout>
```

Nos marcará en rojo el color de nuestro Ratingbar, creen el color ratingColor con el color que ustedes deseen.

Podemos crear el corazón con código xml, en res/drawable creamos un xml nuevo llamado ic_heart y pegamos el siguiente código.

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="24dp"
    android:width="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24">
    <path android:fillColor="#fff"
android:pathData="M12,21.35L10.55,20.03C5.4,15.36 2,12.27 2,8.5C2,5.41 4.42,3
7.5,3C9.24,3 10.91,3.81 12,5.08C13.09,3.81 14.76,3 16.5,3C19.58,3 22,5.41
22,8.5C22,12.27 18.6,15.36 13.45,20.03L12,21.35Z" />
</vector>
```

Antes de continuar con el detalle de la segunda actividad debemos crear el viewModel para esta vista. Creamos una nueva clase llamada DetalleModel y pegamos lo siguiente.

```
class DetalleModel: ViewModel() {
    val responseInfoMovie = MutableLiveData<InfoMovie>()

    val useCases = GetInfoMovie()

    fun getInfoMovie(imdbID: String) {
        viewModelScope.launch {
            coroutineScope {

                val infoAsync = async { useCases("d4a899e2", imdbID) }
                val infoAsyncResponse = infoAsync.await()
                val info = infoAsyncResponse

                responseInfoMovie.postValue(info!!)
            }
        }
    }
}
```

Creamos nuestro caso de uso para hacer la petición al servidor, creamos una clase nueva kotlin llamada GetInfoMovie.

```
class GetInfoMovie {
    private val repository = ServiceRepository()
```

```
    suspend operator fun invoke(apikey: String, imdbID: String) : InfoMovie?
= repository.getInfoMovie(apikey, imdbID)
}
```

Ahora vamos a DetalleMovieActivity para pasarle la información que mostraremos en la vista.

```
class DetalleMovieActivity : AppCompatActivity() {

    private lateinit var binding: ActivityDetalleMovieBinding

    private lateinit var titleView : TextView
    private lateinit var ratingView : TextView
    private lateinit var plotView : TextView
    private lateinit var posterImage : ImageView
    private lateinit var backButton : ImageView
    private lateinit var likeButton : ImageView
    private lateinit var ratingBar : RatingBar

    private val detalleModel : DetalleModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityDetalleMovieBinding.inflate(layoutInflater)
        setContentView(binding.root)

        titleView = binding.tvPelicula
        ratingView = binding.tvRating
        plotView = binding.tvPlot
        posterImage = binding.ivPoster
        backButton = binding.buttonBack
        likeButton = binding.likeButton
        ratingBar = binding.ratingBar

        val imdbID = intent.getStringExtra("imdbID")

        if (imdbID != null) {
            detalleModel.getInfoMovie(imdbID)
            detalleModel.responseInfoMovie.observe(this, {
                Log.d("movies", "it = $it")

                val title = it.Title
                titleView.text = title
                ratingView.text = "-"
                var ratingFloat = 0.0f
                for (rating in it.Ratings!!) {
                    if (rating.Source.equals("Rotten Tomatoes")) {
                        ratingView.text = rating.Value
                        val split = rating.Value.split("%")
                        ratingFloat = (split[0].toFloat()*5)/100
                        ratingBar.rating = ratingFloat
                    }
                }
                plotView.text = it.Plot
                Picasso.get().load(it.Poster).into(posterImage)
                likeButton.setOnClickListener {
                    Toast.makeText(this, "Me encanta $title",
Toast.LENGTH_SHORT).show()
```

```
                }

            })
        }

        backButton.setOnClickListener {
            finish()
        }
    }
}
```

Aquí hacemos uso del objeto InfoMovie porque usamos otro servicio web en donde mandamos como parámetro el imdbID de la película que seleccionamos para traernos aún más información como por ejemplo una ligera descripción, el rating, los actores que participaron en la película, duración de la película, entre otros. Aquí podemos jugar con toda esta información y mostrar aún más detalle. Ademas tenemos una imagen que usaremos para saber si nos gusta esa película para en un futuro agregarla a una lista de películas favoritas.

PASO 6. Vista Favoritos

Primero crearemos una variable global, esto quiere decir que podemos acceder a ella desde cualquier clase en todo el proyecto. La crearemos en la clase MainActivity fuera de la clase.

```
lateinit var favoritosList: MutableList<InfoMovie>

class MainActivity : AppCompatActivity() {
```

Ahora inicializaremos la lista dentro de la función onCreate()

```
favoritosList = mutableListOf()
```

Volvemos a la clase DetalleMovieActivity y modificaremos setOnClickListener de nuestro likeButton

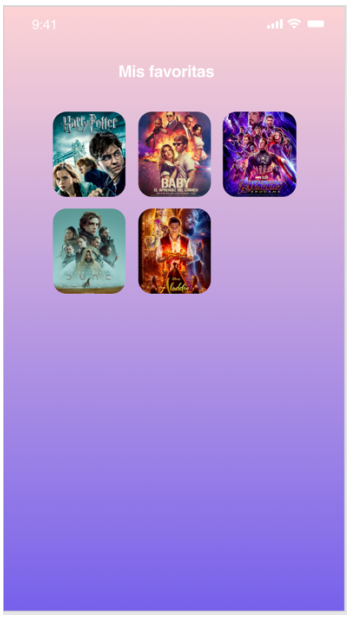
```
likeButton.setOnClickListener { v ->
    var existe = false
    if (favoritosList.size == 0) {
        existe = false
    } else {
        for (item in favoritosList) {
            if (item.Title == title) {
                existe = true
            }
        }
    }

    if (!existe) {
        favoritosList.add(InfoMovie(title, it.Year, it.Released, it.Runtime,
it.Director, it.Actors, it.Plot, it.Poster, it.Type, it.Metascore,
it.Ratings))
        Toast.makeText(this, "Me encanta $title", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(this, "$title ya se encuentra en la lista de
favoritos", Toast.LENGTH_SHORT).show()
    }
```

```
}
```

Como vemos buscamos en la lista de favoritos si ya existe la película que le damos a favoritos, si no existe lo agregamos a la lista, caso contrario avisamos que ya existe en la lista.

PASO 7. Crearemos la vista de peliculas favoritas

Ahora crearemos un Empty Activity en el que mostraremos la lista de películas favoritas que guardamos en nuestra lista en un recyclerView. Un ejemplo de como puede quedar esta vista.



PASO 8. Crearemos la vista de login

Ahora que ya sabemos algunas cosas crearemos la vista de login que será la primera vista que se mostrará al abrir nuestra aplicación. Para esto ahora crearemos un Empty Activity llamada LoginActivity junto con su layout llamada activity_login. Le agregaremos un color de background, puede ser el gradient que creamos o puede cambiar el color al que desee. Agregarmos un botón con el que vamos a ingresar a la aplicación. Para hacerlo más real podemos agregar dos cajas de texto (EditText).

Un EditText tiene como atributos inputType con el que textPassword hace que se oculte el text que vamos escribiendo. Otro atributo es hint que es el texto que aparece en el fondo del EditText , ejemplo escribirmos password en el hint nos aparecerá el texto e indicará al usuario que en ese contenedor se escribirá la contraseña.

Para que nuestra vista Login sea la primera en aparecer hay que ir al AndroidManifest y reemplazar el código por el siguiente.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="mx.iteso.proyectofinal">

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

    <application
        android:allowBackup="true"
```

```
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ProyectoFinal">
        <activity
            android:name=".LoginActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".DetalleMovieActivity"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="false">

        </activity>
    </application>

</manifest>
```

Ejemplo de como se debe ver la vista.