Datos del estudiante

**Nombre y apellidos** Nombre y apellidos del estudiante

**Fecha de entrega** 20/12/2024

Actividad 2: Conexión a una API-Rest

Objetivos de la actividad

En esta actividad debes programar la funcionalidad de una calculadora de salario neto. Aquí pondrás en práctica:

* Diseño de interfaces en Android
* Uso del lenguaje Kotlin
* Manejo de IDE Android Studio
* Conexión a RetroFit
* Uso de Corrutinas

Aunque la app es ficticia, la manera de enfocar esta actividad y lo que se requiere no difiere mucho a cómo funciona el mundo real.

Pautas de elaboración

Se debe de crear una App en Android Studio usando Kotlin como lenguaje de programación. El diseño de la App es libre, pero debe de tener una vista en la que se muestren los resultados. Usaremos una API pública de películas. La URL para obtener películas populares es algo como: <https://api.samplemovieapi.com/movies/popular>

Se deben de seguir los siguientes pasos para el desarrollo de la actividad:

1. Crea un nuevo proyecto en Android Studio.
2. Configura los permisos de internet en el archivo AndroidManifest.xml:

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

1. Agrega las dependencias en el archivo build.gradle:

implementation "com.squareup.retrofit2:retrofit:2.9.0"

implementation "com.squareup.retrofit2:converter-gson:2.9.0"

implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.6'

1. Crea la Data Class para los datos de la API:

Para que Retrofit pueda convertir los datos JSON a objetos Kotlin, se necesita tener una data class. El fichero JSON que devuelve la API tiene una estructura como esta:

{

"movies": [

{

"title": "The Matrix",

"poster": "https://link.to/poster.jpg",

"rating": 8.7

},

...

]

}

Se debe de crear en el directorio principal una clase llamada MovieResponse:

data class MovieResponse(

val movies: List<Movie>

)

data class Movie(

val title: String,

val poster: String,

val rating: Double

)

1. Crea la interfaz API para Retrofit MovieApiService:

interface MovieApiService {

@GET("movies/popular")

suspend fun getPopularMovies(): Response<MovieResponse>

}

Se deben de definir los métodos que se quieren implementar de la API. El uso de la palabra clave suspend permite que esta función se ejecute en segundo plano usando corrutinas.

1. Configura la instancia de Retrofit en Kotlin:

En el archivo MainActivity.kt se debe de configurar la instancia de Retrofit que se usará en la aplicación. Añade este código en el MainActivity:

private fun getRetrofit(): Retrofit {

return Retrofit.Builder()

.baseUrl("https://api.samplemovieapi.com/")

.addConverterFactory(GsonConverterFactory.create())

.build()

}

La función getRetrofit() devuelve una instancia de Retrofit configurada con la URL base de la API y un convertidor JSON.

1. Crea la función fetchMovies para hacer la solicitud de datos en el MainActivity:

private fun fetchMovies() {

CoroutineScope(Dispatchers.IO).launch {

val call = getRetrofit().create(MovieApiService::class.java).getPopularMovies()

val movies = call.body()

runOnUiThread {

if (call.isSuccessful) {

val movieList = movies?.movies ?: emptyList()

displayMovies(movieList)

} else {

showError()

}

}

}

}

Mediante esta función, se realiza la llamada a la API, en caso de que sea exitosa, se pasan los datos al método displayMovies() (se implementará más adelante) para mostrarlos en la interfaz de usuario. Si no ocurre esto, se muestra un mensaje de error.

1. Configura el RecyclerView y el Adapter

Se debe de usar un RecyclerView para mostrar los datos. Se agrega en el archivo activity\_main.xml un RecyclerView con el ID rvMovies. Posteriormente, se crea una clase MovieAdapter para enlazar los datos con las vistas:

class MovieAdapter(private val movies: List<Movie>) : RecyclerView.Adapter<MovieViewHolder>() {

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MovieViewHolder {

val layoutInflater = LayoutInflater.from(parent.context)

return MovieViewHolder(layoutInflater.inflate(R.layout.item\_movie, parent, false))

}

override fun getItemCount(): Int = movies.size

override fun onBindViewHolder(holder: MovieViewHolder, position: Int) {

val item = movies[position]

holder.bind(item)

}

}

1. Mostrar los datos mediante la función displayMovies

private fun displayMovies(movies: List<Movie>) {

val adapter = MovieAdapter(movies)

binding.rvMovies.layoutManager = LinearLayoutManager(this)

binding.rvMovies.adapter = adapter

}

1. Manejo de errores

Se define la función showError para mostrar un mensaje de error si la llamada a la API falla.

private fun showError() {

Toast.makeText(this, "Error al cargar películas", Toast.LENGTH\_SHORT).show()

}

Extensión y formato

Se debe entregar una carpeta de proyecto comprimido.

Además, se debe de entregar el enlace al repositorio público de github del proyecto.

Criterios de evaluación

Especifica los criterios con los que medir los resultados obtenidos por el estudiante con la herramienta de la rúbrica.

|  |  |  |
| --- | --- | --- |
| App RetroFit | Descripción | Peso  % |
| Criterio 1 | Creación del proyecto en Android Studio y configuración de los permisos y dependencias | 10% |
| Criterio 2 | Creación y diseño de la primera vista. | 20% |
| Criterio 3 | Creación de la Data Class | 10% |
| Criterio 4 | Creación de la interfaz API para Retrofit. | 10% |
| Criterio 5 | Configuración de la instancia de Retrofit. | 10% |
| Criterio 6 | Creación de la función para la solicitud de datos. | 10% |
| Criterio 7 | Configuración del RecyclerView y el Adapter | 20% |
| Criterio 8 | Mostrar datos y manejo de errores | 10% |
|  |  | **100 %** |