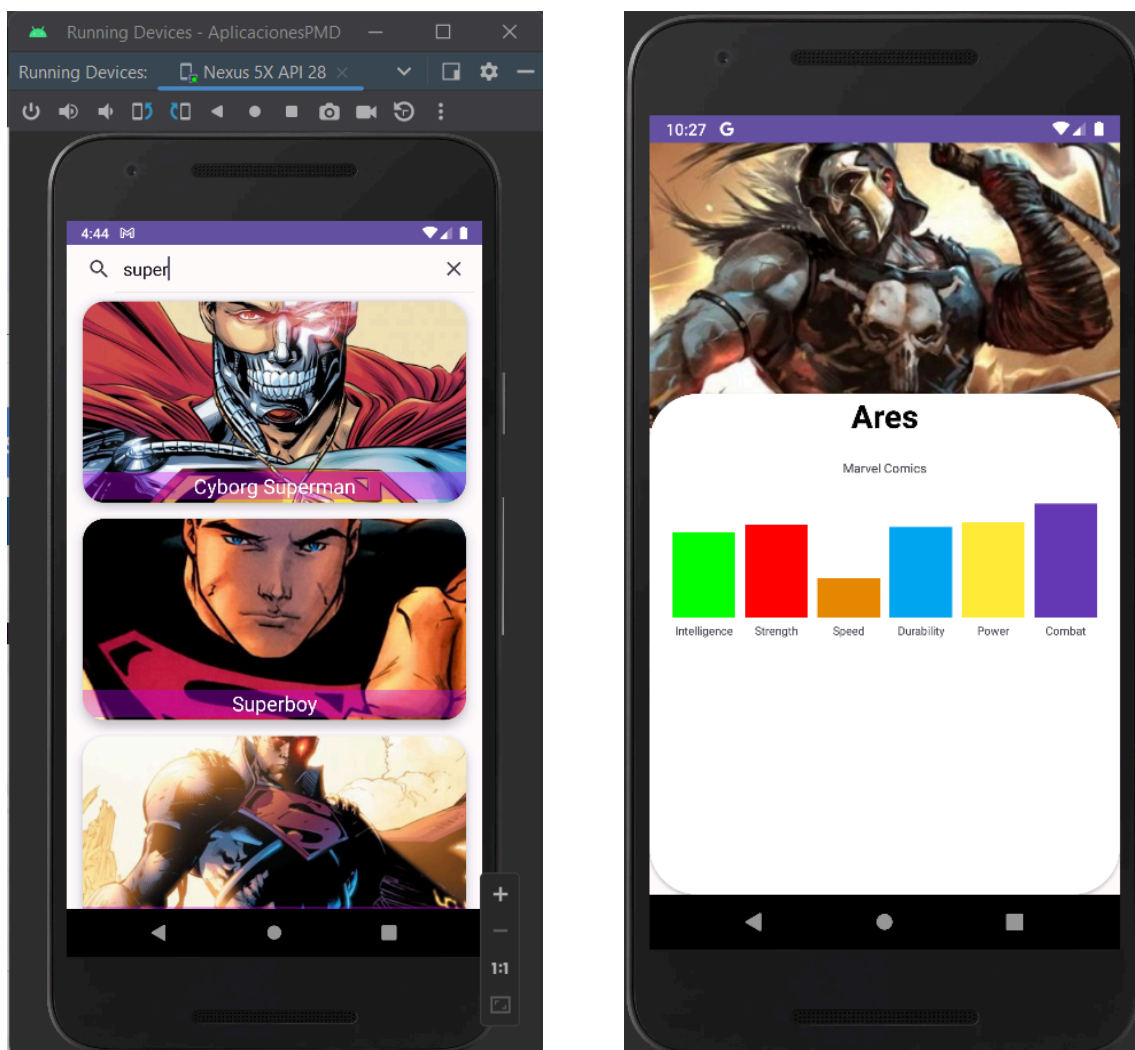



Modifica la aplicación creada para el estudio de consumo de APIs de tal manera que los datos obtenidos se almacenen en la BD Room.

1. Crea un nuevo proyecto en AndroidStudio y copia los archivos necesarios para que la aplicación funcione correctamente. Acuérdate de dar permiso de internet en el archivo *AndroidManifest.xml* y de instalar las dependencias de Retrofit, Picasso, Corrutinas y Room, además del plugin *id("kotlin-kapt")*..



2. Crea un directorio */database* y, dentro de éste, los directorios */entities* y */dao*. Tienes que crear dos tablas, una para la actividad principal donde se muestra el RecyclerView y otra para la pantalla de detalle de cada superhéroe. Los campos a almacenar en las tablas son los siguientes:
 - Tabla 1: *id* (clave primaria autogenerada), *name* (string) e *image* (string)
 - Tabla 2: *id* (clave primaria autogenerada), *intelligence*, *strength*, *speed*, *durability*, *power*, *combat*, *fullName* y *publisher* (todas tipo string)

	BD ROOM	PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES 2º DAM
---	----------------	--

- En la interfaz *ApiService.kt* modifica los métodos `@GET` para que ambos realicen una búsqueda de unos pocos datos. Por ejemplo:

```
interface ApiService {
    @GET("/api/10229233666327556/search/sp")
    suspend fun getSuperheroes() : Response<SuperHeroDataResponse>

    @GET("/api/10229233666327556/search/sp")
    suspend fun getSuperheroDetail() : Response<SuperHeroDetailResponse>
}
```

La clase *SuperheroDataResponse* puedes dejarla como está, será la que irá asociada a la primera tabla de la BD. Pero la clase *SuperheroDetailResponse* debes modificarla para que se parezca a la anterior, ya que ahora realizará una búsqueda de los *powerstats*, el nombre completo y la editorial de todos los superhéroes que se indiquen en el método `@GET` de la interfaz.


- Como no estamos usando inyección de dependencias, en cada actividad donde quieras hacer uso de la base de datos, deberás crear una instancia de la misma con el siguiente comando:

```
room = Room.databaseBuilder(this,
    SuperheroDatabase::class.java, "superheroes").build()
```

- En la actividad principal, en vez de pasarle los datos al Adapter desde la API, deberás hacerlo desde la BD. Para eso, has de crear un método previo que almacene los datos de la API en la BD al iniciar la aplicación. Los datos obtenidos del servidor habrá que transformarlos a datos manejables por la BD mediante un *mapping* implementando un método *toDatabase()* en la entidad correspondiente a la primera tabla. Algo parecido a esto:

```
fun SuperheroItemResponse.toDatabase() = ListEntity(name =
    name, image = superheroImage.url)
```

- Al ingresar los datos en Room, recuerda invocar al método *deleteAll()* de tu DAO antes de invocar al método *insertAll()* para no poblar la BD con registros repetidos cada vez que se ejecute la aplicación.
- Repite el mismo proceso para los datos consumidos de la API mediante la función *getSuperheroDetail()* de la interfaz, incluyendo este proceso en el método de almacenamiento anterior.
- Ahora, la función *searchByName()* de la actividad principal deberá tomar el texto ingresado por el usuario en el SearchView y aplicar un método de

 cpi fp Los Enlaces	BD ROOM	PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES 2º DAM
--	----------------	--

consulta en el DAO correspondiente para hacer la búsqueda por nombre sobre la BD.

9. Para la actividad que gestiona la pantalla con los detalles de un superhéroe determinado, en la función *getSuperheroInformation()*, se deberán realizar dos búsquedas sobre la base de datos: una en la primera tabla para obtener el nombre y la imagen del superhéroe y otra en la segunda para obtener los datos restantes.
10. Ten en cuenta que los objetos con los que van a trabajar todas las funciones ahora son de tipo *Entity*, de las clases creadas para cada una de las tablas. Al igual que la lista de elementos que se le pase al Adapter y al ViewHolder.

RÚBRICA DE CALIFICACIÓN

Clases para Room (3 puntos):

- Entidades bien definidas (1 punto)
- Objetos de acceso a datos con todos los métodos necesarios para realizar las consultas oportunas (1 punto)
- Clase para la declaración de la BD bien implementada (1 punto)

Adaptación del código (6 puntos):

- Modificación de interfaz de interacción con la API correcta (0.5 puntos)
- Clase *SuperheroDetailResponse* modificada correctamente (1 punto)
- Método de almacenamiento de datos en la BD desde la API bien implementado (1 punto)
- Mapeado de datos correcto (0.5 puntos)
- Función para realizar búsquedas por nombre desde la base de datos bien definida, con método de consulta en DAO correcto (1 punto)
- Función *getSuperheroInformation()* bien modificada, incluyendo métodos necesarios en los DAO correspondientes (1 punto)
- Modificación correcta de tipos de parámetros para todas las funciones y métodos afectados (0.5 puntos)
- Datos pasados a la interfaz de usuario correctamente (0.5 puntos)

Funcionamiento correcto (1 punto)