Retrofit2

Vamos a seguir utilizando la **App_03_01**.

PASOS PREVIOS

En primer lugar añadiremos la configuración al fichero build.gradle usando GSON

```
//Retrofit2
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

También añadiremos el permiso INTERNET en AndroidManifest.xml

IMPLEMENTACIÓN

Ahora vamos a obtener nuestros datos de animales a partir de un servicio web.
El servicio estará alojado en la siguiente URL:
https://aplicaciones.ivanlorenzo.es/pmdm/animales/animales.json

Devuelve directamente JSON

En primer lugar cambiaremos la data class Animal para que la propiedad imagen ahora sea un String (en el servicio web se obtendrá la ruta de la imagen).
Además, añadiremos los @SerializedName de cada propiedad

```
import com.google.gson.annotations.SerializedName
import java.io.Serializable

data class Animal(
    @SerializedName("nombre")
    val nombre:String,
    @SerializedName("imagenURL")
    val imagenURL: String,
    @SerializedName("descripcion")
    val descripcion: String = "",
    @SerializedName("votos")
    var votos: Int = 0
) : Serializable
```

Si abrimos ahora DatosAnimales veremos multitud de errores. Borraremos este fichero ya que los datos no los tendremos en la propia app.

Comenta las líneas que hagan referencia a algún imageView de la foto del animal en las clases AnimalAdapter y DetalleAnimalActivity

Y por último, consumiremos el servicio.

Para ello crea el fichero **AnimalApiService.kt** dentro de un paquete llamado **data.api**

Crearemos una constante con el valor de la URL del servicio

```
private const val URL_BASE =
    "https://aplicaciones.ivanlorenzo.es/pmdm/animales/"
```

Crearemos el objeto Retrofit

```
private val retrofit = Retrofit.Builder()
    .baseUrl(URL_BASE)
    .addConverterFactory(GsonConverterFactory.create())
    .build()
```

Crearemos una interface llamada AnimalApiService que tendrá un único método getAnimales. Este método hará una petición GET para obtener animales.json

```
interface AnimalApiService
{
    @GET("animales.json")
    suspend fun getAnimales() : List<Animal>
}
```

Por último, crearemos un objeto de la API que expone el servicio Retrofit

```
object AnimalesApi {
    val retrofitService: AnimalApiService by lazy { retrofit.create(AnimalApiService::class.java) }
}
```

Únicamente tendremos que cambiar el método get de la clase AnimalRepository para que llame al objeto AnimalesApi que acabamos de crear:

```
class AnimalRepository
{
    suspend fun get(): List<Animal> = AnimalesApi.retrofitService.getAnimales()
}
```

Mostrar imágenes a partir del nombre:

En la clase **AnimalAdapter** modificamos el código para obtener la imagen a partir del nombre de la imagen

```
val resID = itemView.context.resources.getIdentifier(animal.imagenURL, "drawable",
itemView.context.packageName)
binding.imageViewFotoAnimal.setImageResource(resID)
```

Hacemos lo mismo en la clase **DetalleAnimalActivity**:

```
val resID = resources.getIdentifier(animal.imagenURL, "drawable", packageName)
binding.imageViewAnimal.setImageResource(resID)
```

OBTENER IMÁGENES

En el paso anterior estamos cogiendo las imágenes del proyecto. Ahora vamos a borrar las imágenes del proyecto y vamos a descargarlas directamente de la web.

Para ello usaremos algunas de las múltiples librerías que podemos encontrar para realizar esta funcionalidad.
La que usaremos se llama Coil (https://coil-kt.github.io/coil/)

Añadiremos la dependencia en el fichero build.gradle:

```
//Coil
implementation("io.coil-kt:coil:1.2.2")
```

Cambiaremos el código para establecer la imagen en el imageViewAnimal de la clase DetalleAnimalActivity.
Simplemente llamando al método load y pasándole la URL de la imagen, él se encargará de descargarla en segundo plano:

```
binding.imageViewAnimal.load("https://aplicaciones.ivanlorenzo.es/pmdm/animales/$
{animal.imagenURL}.png")
```