# Guía de ejercicios - Consumo de API REST (III)

 ¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

## ¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

**¡Vamos con todo!**

## Tabla de contenidos


| | |
|---|---|
| Actividad guiada: Retrofit y Autenticación | 2 |
| ¡Manos a la obra! - User Glide para mostrar imágenes (remote) | 5 |
| **¡Manos a la obra! - Usar Picasso para mostrar imágenes (local)** | **6** |
| **Respuestas:** | **7** |
| Preguntas de proceso | 10 |
| Preguntas de cierre | 10 |
| Referencias bibliográficas | 10 |


**¡Comencemos!**

## Actividad guiada: Retrofit y Autenticación

Para usar AuthToken en Android con Retrofit en Kotlin, puedes seguir estos pasos:

Crea una interfaz que defina los extremos de la API que requieren autenticación. Por ejemplo, supongamos que tiene un punto de conexión/usuarios de API que requiere autenticación:

```
interface ApiService {
    @GET("/users")
    suspend fun getUsers(@Header("Authorization") authToken: String):
List<User>
}
```

En este ejemplo, hemos definido un único método getUsers() que devuelve una lista de objetos Usuario. Agregamos una anotación `@Header` al parámetro authToken para pasar el token de autenticación al extremo de la API.

Crea una instancia de Retrofit con un `OkHttpClient` que agregue el token de autenticación a cada solicitud. Aquí hay un ejemplo:

```
val authToken = "your-auth-token"

val client = OkHttpClient.Builder()
    .addInterceptor { chain ->
        val request = chain.request().newBuilder()
            .addHeader("Authorization", "Bearer $authToken")
            .build()
        chain.proceed(request)
    }
    .build()

val retrofit = Retrofit.Builder()
    .baseUrl("https://your-api-base-url.com")
    .client(client)
    .addConverterFactory(GsonConverterFactory.create())
    .build()
```

```
val apiService = retrofit.create(ApiService::class.java)
```

En este ejemplo, hemos creado un `OkHttpClient` que agrega el token de autenticación a cada solicitud mediante un Interceptor. Hemos agregado el encabezado de Autorización con el valor Bearer `$authToken`. Luego creamos una instancia de Retrofit con `OkHttpClient` y `GsonConverterFactory` para analizar la respuesta JSON.

Llame al extremo de la API con el parámetro authToken. Aquí hay un ejemplo:

```
val users = apiService.getUsers(authToken)
```

En este ejemplo, llamamos al método `getUsers()` en la instancia de `apiService` con el parámetro authToken.

Escribe pruebas unitarias para tu código. Aquí hay un ejemplo de cómo puede probar la interfaz `apiService`:

```
class ApiServiceTest {
    private lateinit var apiService: ApiService

    @Before
    fun setUp() {
        val client = OkHttpClient.Builder()
            .addInterceptor { chain ->
                val request = chain.request().newBuilder()
                    .addHeader("Authorization", "Bearer
your-auth-token")
                    .build()
                chain.proceed(request)
            }
            .build()

        val retrofit = Retrofit.Builder()
            .baseUrl("https://your-api-base-url.com")
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build()

        apiService = retrofit.create(ApiService::class.java)
    }
```

```
    @Test
    fun `getUsers() should add auth token to header`() = runBlocking {
        val authToken = "your-auth-token"
        val request = Request.Builder()
            .url("https://your-api-base-url.com/users")
            .addHeader("Authorization", "Bearer $authToken")
            .build()

        apiService.getUsers(authToken)

        verify(mockHttpClient).newCall(request).execute()
    }
}
```

En este ejemplo, hemos creado una prueba unitaria para el método `getUsers()` de la interfaz `ApiService`. Hemos creado una instancia simulada de HttpClient que intercepta la solicitud y verifica que el encabezado de Autorización se agregue con el valor correcto.

Luego llamamos al método `getUsers()` en la instancia de `ApiService` y verificamos que la solicitud se ejecuta con el encabezado correcto.
Una vez realizados estos pasos, el ejercicio habrá sido completado.

## ¡Manos a la obra! - User Glide para mostrar imágenes (remote)

En este ejercicio deberás utilizar Glide para mostrar imágenes almacenadas fuera del teléfono. Para esto puedes utilizar la siguiente:

https://isorepublic.com/wp-content/uploads/2023/02/iso-republic-leaf-macro-02-2048x1366.jpg

Recuerda validar que la imagen esté disponible en la URL dada, si deseas mostrar otra imagen, puedes elegir aquí: https://isorepublic.com/

1. Utiliza Glide para mostrar imágenes en servidores remotos.
2. Utiliza la misma imagen, esta vez agrega lo siguiente:
    a. Redimensiona la imagen a 500px x 500px
    b. Define un placeholder
    c. Define una imagen en caso de error
    d. Utiliza la propiedad Center Crop
3. Agrega los métodos necesarios para limpiar la memoria caché y la memoria caché en el disco.

## ¡Manos a la obra! - Usar Picasso para mostrar imágenes (local)

En el siguiente ejercicio, deberás mostrar una imagen que esté guardada en el teléfono, para esto, es necesario que agregues una nueva imagen a la carpeta “drawable”, puede ser la imagen que prefieras

1. Utiliza Picasso para mostrar imágenes locales (recursos).
2. Utiliza la misma imagen, esta vez agrega lo siguiente:
    a. Redimensiona la imagen a 500px x 500px
    b. Define un placeholder
    c. Define una imagen en caso de error
    d. Utiliza la propiedad Center Crop
3. Agrega los métodos necesarios para limpiar la memoria caché y la memoria caché en el disco.

## Respuestas:

1. Mostrar imágenes usando **Glide**:

**Primera parte:**

Agrega la dependencia Glide al archivo build.gradle de tu proyecto:

```
dependencies {
    implementation 'com.github.bumptech.glide:glide:4.12.0'
    kapt 'com.github.bumptech.glide:compiler:4.12.0'
}
```

Carga una imagen con Glide llamando al método `load()` con la URL de la imagen y luego llamando al método `into()` para mostrar la imagen en un ImageView:

```
val imageView = findViewById<ImageView>(R.id.imageView)

Glide.with(this)
    .load("https://example.com/image.jpg")
    .into(imageView)
```

**Segunda parte:**

Personaliza el comportamiento de carga de imágenes llamando a métodos adicionales en el objeto Glide.

```
val imageView = findViewById<ImageView>(R.id.imageView)

Glide.with(this)
    .load("https://example.com/image.jpg")
    .placeholder(R.drawable.placeholder_image)
    .error(R.drawable.error_image)
    .override(500, 500)
    .centerCrop()
    .into(imageView)
```

Borra la memoria caché de la imagen con Glide llamando a los métodos `clearDiskCache()` y `clearMemory()`:

```
Glide.get(context).clearDiskCache()
Glide.get(context).clearMemory()
```

2. Mostrar imágenes usando **Picasso**:

Agrega la dependencia de Picasso al archivo `build.gradle` de tu proyecto:

```
dependencies {
    implementation 'com.squareup.picasso:picasso:2.71828'
}
```

Cargue una imagen local con Picasso llamando al método `load()` con el archivo de imagen y luego llamando al método `into()` para mostrar la imagen en un ImageView:

```
val imageView = findViewById<ImageView>(R.id.imageView)

Picasso.get()
    .load(File("/path/to/image.jpg"))
    .into(imageView)
```

Personaliza el comportamiento de carga de la imagen llamando a métodos adicionales en el objeto Picasso.

```
val imageView = findViewById<ImageView>(R.id.imageView)

Picasso.get()
    .load(File("/path/to/image.jpg"))
    .placeholder(R.drawable.placeholder_image)
    .error(R.drawable.error_image)
    .resize(500, 500)
    .centerCrop()
    .into(imageView)
```

Borre la memoria caché de la imagen con Picasso llamando al método `clearCache()`:

```
Picasso.get().clearCache()
```

## Preguntas de proceso

**Reflexiona:**

- ¿Existe algún contenido que te haya resultado interesante o de utilidad hasta ahora? ¿Por qué?
- ¿Qué ejercicio se te hizo más difícil? ¿Puedes identificar por qué es más dificultoso para ti?
- ¿Cómo crees que podrían hacer más sencillo tu proceso de aprendizaje?

## Preguntas de cierre

- ¿Cuál es la forma más básica de usar Glide?
- ¿Cuál es la diferencia entre Glide, Picasso y Fresco?
- Nombra los pasos necesarios para poder realizar una autenticación con AuthToken

## Referencias bibliográficas


- Documentación de Glide: https://github.com/bumptech/glide
- Documentación de Picasso: https://square.github.io/picasso/
- Documentación de Fresco: https://frescolib.org/docs/