{desafío}
latam_

API REST y acceso a recursos remotos _

Parte II

Restful/Restless y REST Post

Introducción

Abordaremos las características que componen a un sistema REstful, diferenciando y aclarando otras variantes para tener conceptos más claros, respecto a estos términos utilizados en la industria, de esta manera desarrollaremos mejores aplicaciones basadas en arquitectura REST.

Recordando que REST incluye la forma en que los estados de los recursos se dirigen y transfieren datos y archivos a través de HTTP, profundizaremos en la utilización del método POST para continuar con la aplicación “RestApi” de los ejercicios y crear en esta ocasión nuevos post.

Adicionalmente, revisaremos los conceptos Restful y Restless de la programación de servicios informáticos REST.

Objetivos

- Conocer qué son los servicios RestFul
- Conocer qué son los servicios Restless
- Implementar Retrofit para consumir una API en una aplicación Android.

Restful

La palabra Restful hace referencia a la disponibilidad de servicios web u otros sistemas, que ofrecen recursos basados en el protocolo HTTP, por lo que para su acceso se usan los métodos GET, POST, etc., es decir, Restful se puede entender como un grupo de servicios web o programas basados en la arquitectura HTTP que convoca REST.

Estos servicios web o sistemas ofrecen recursos en diversos tipos de formatos, según se requiera, son accesibles mediante direcciones web (url) y no se necesitan frameworks para acceder a ellos, por eso se les consideran ligeros.

Restless

Restless es cualquier sistema que no sea Restful, es decir, basta con que el sistema no cumpla con alguna de las características de un sistema Restful.

Aunque parezca asombroso, la mayoría de los sistemas son RESTless dado que no cumplen con una característica elemental algo desconocida en su concepto, el Hypermedia as the Engine of Application State (HATEOAS), el cual es un componente de la arquitectura de aplicaciones REST que lo distingue de otras arquitecturas de red.

Características de un sistema Restful

- **Identificación inequívoca de todos los recursos**: todos los recursos han de poder identificarse con un URI (Unique Resource Identifier).

- **Interacción con los recursos por medio de representaciones**: si un cliente necesita un recurso, el servidor le envía una representación para que el cliente pueda modificar o borrar el recurso original.

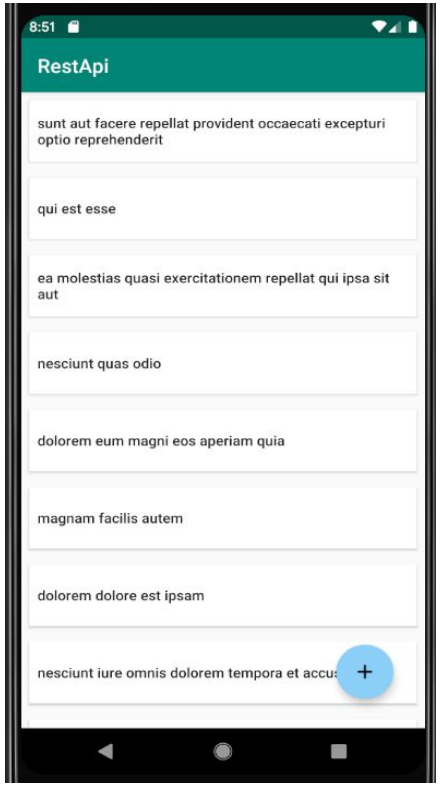
- **Mensajes explícitos**: cada mensaje intercambiado por el servidor y el cliente ha de contener todos los datos necesarios para entenderse.

- **HATEOAS**: esta estructura basada en hipermedia facilita a los clientes el acceso a la aplicación. No necesitan saber nada más de la interfaz para poder acceder y navegar.

Ejercicio 7

Implementar el método POST del api JsonPlaceHolder para crear nuevos posts desde nuestro sistema “RestApi”, dónde necesitaremos incluir un FloatingButton en la página de lista principal, cuya acción sea desplegar un Dialog o Modal que muestre tres campos para ingresar datos (los mismos que requiere el api) y un botón “ENVIAR” de acción.

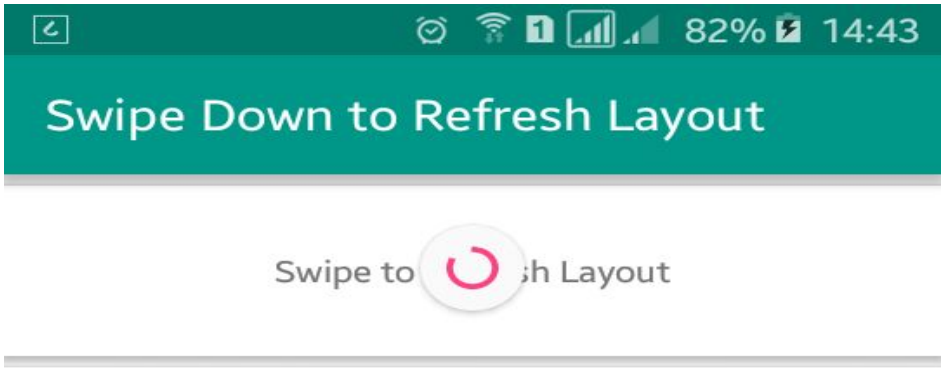
Ejercicio 7 - Solución



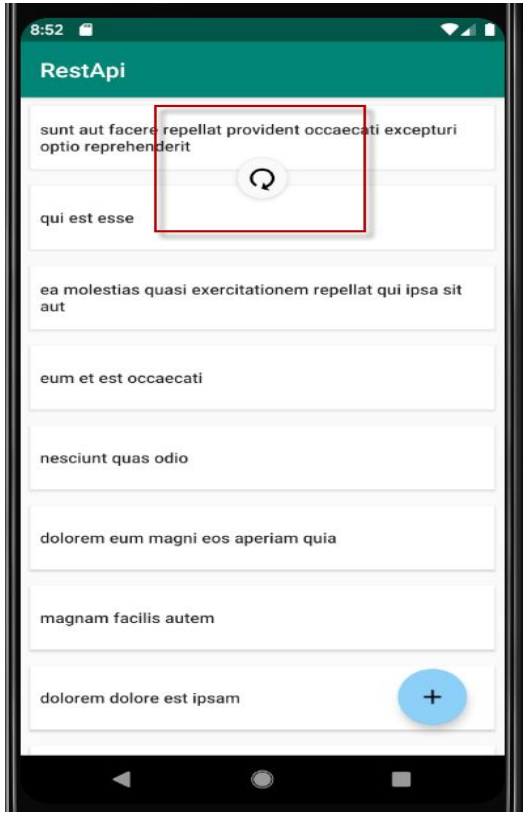
{desafío} latam_

Ejercicio 8

Ejercicio bonus, agregar la funcionalidad swipeToRefresh a la actividad principal para refrescar los datos cargados del api.



Ejercicio 8 - Solución



Imágenes con Picasso

Introducción

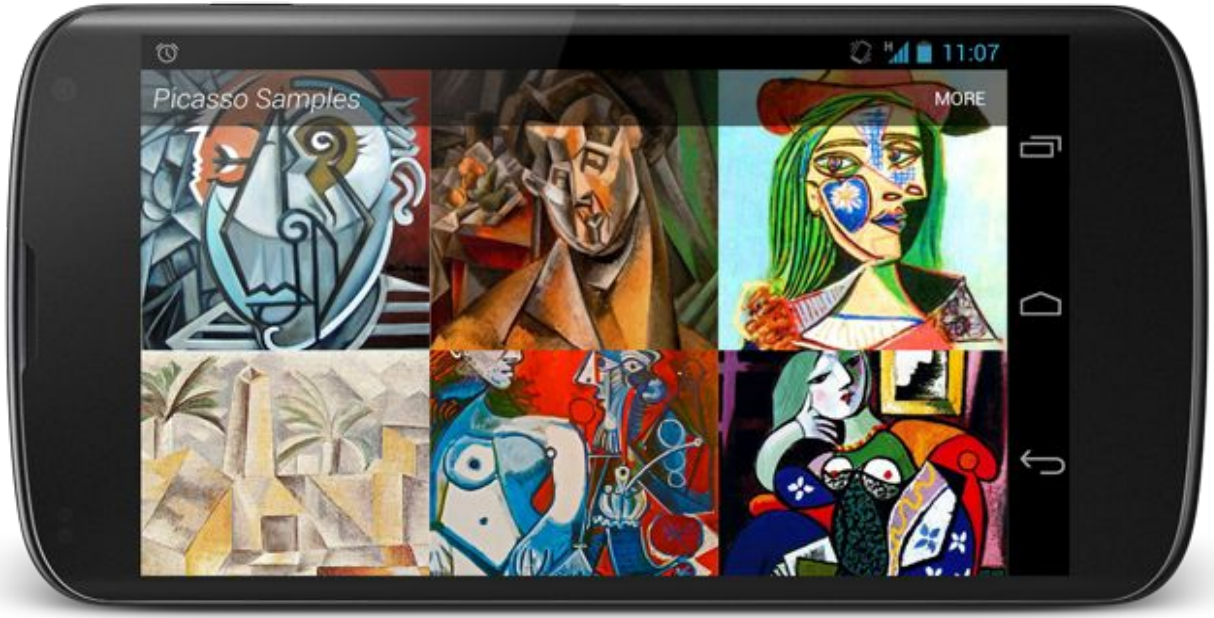
Estudiaremos cómo se realiza la carga y descarga de imágenes utilizando la librería más común y popular de android “Picasso”. El éxito de esta librería se debe a su gran eficiencia en el manejo de los recursos gráficos y en su rendimiento, siendo la de mejor manejo de la memoria del dispositivo.

Es esencial aprender a utilizar Picasso, ya que el trabajo con imágenes en las aplicaciones móviles son recurrentes, por ejemplo en listas, detalles, iconos, entre otras vistas, siendo imprescindible que sepamos no sobrecargar el sistema y practicar los mejores códigos para estas implementaciones.

Objetivos

- Conocer los conceptos asociados al uso de la librería Picasso
- Implementar Picasso en aplicaciones android
- Realizar caching de imágenes con Picasso
- Conocer librerías alternativas para el manejo de imágenes

Picasso



Picasso

Las imágenes agregan un contexto muy necesario y un toque visual a las aplicaciones de Android que hace que destaquen nuestros sistemas. Picasso permite la carga de imágenes sin problemas y de una manera muy sencilla, que en ocasiones, bastará con una línea de código.

Picasso fue creada con el propósito de mostrar imágenes de una manera super rápida y sencilla simplificando el proceso que en años anteriores se tenía que hacer; además Picasso se encarga de la solicitud HTTP, el almacenamiento en caché de imágenes, y hasta depurar, por lo tanto, nos hará la vida más fácil a la hora de trabajar con android.

Funcionalidades de Picasso

- Resource Loading

```
Picasso.get().load(R.drawable.landing_screen).into(imageView1);
Picasso.get().load("file:///android_asset/DvpvklR.png").into(imageView2);
Picasso.get().load(new File(...)).into(imageView3);
```

- Adapter Downloads

```
@Override public void getView(int position, View convertView, ViewGroup parent) {
  SquaredImageView view = (SquaredImageView) convertView;
  if (view == null) {
    view = new SquaredImageView(context);
  }
  String url = getItem(position);

  Picasso.get().load(url).into(view);
}
```

Funcionalidades de Picasso

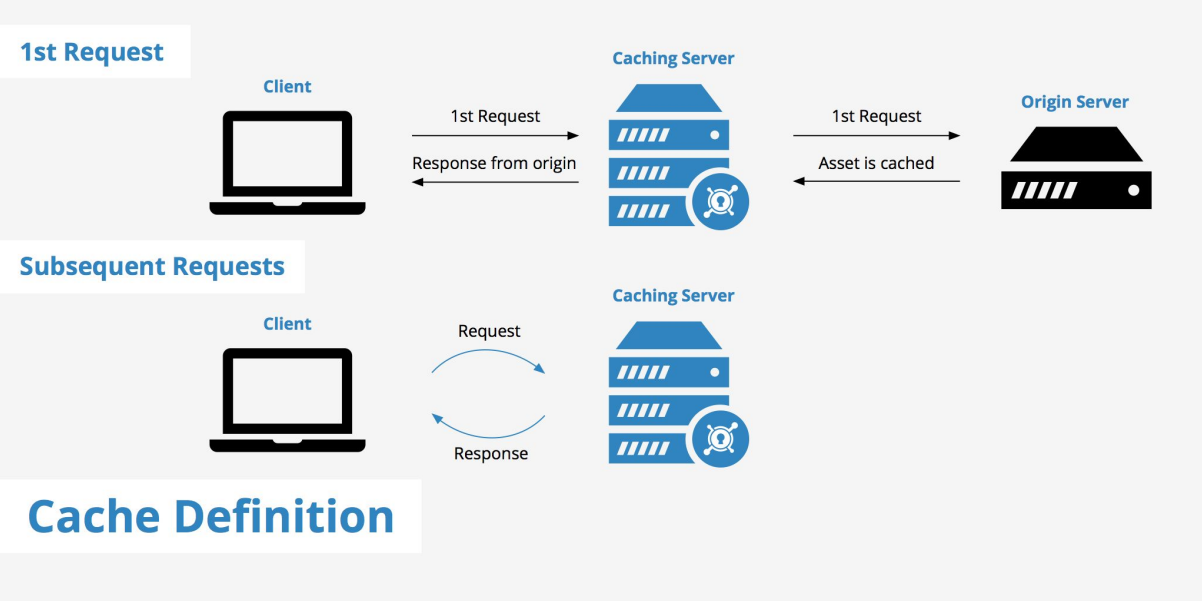
- Image Transformation

```
Picasso.get()
  .load(url)
  .resize(50, 50)
  .centerCrop()
  .into(imageView)
```

- Custom Image Transformations
- Place Holders

```
Picasso.get()
    .load(url)
    .placeholder(R.drawable.user_placeholder)
    .error(R.drawable.user_placeholder_error)
    .into(imageView);
```

Caché con Picasso



Caché con Picasso

Como hemos indicado, dependemos mucho de las imágenes en nuestras aplicaciones. En algunos casos las imágenes suelen repetirse, como por ejemplo una imagen de perfil, la cual sería ideal guardar de alguna forma para no tener que recargar dicha imagen cada vez que se ingresa a la aplicación usando el tráfico de datos, que termina siendo costoso en servidores y alojamientos. Ejemplos de este trabajo de caché de imágenes está en las redes sociales más populares como lo son Facebook e Instagram.

Básicamente de esto se trata cuando hablamos de caché de imágenes. Librerías como Picasso se encargan de almacenarlas y recuperarlas de la memoria volátil, memoria interna o descargarla nuevamente si lo anterior no funciona, también es posible eliminar las imágenes cuando no se utilizan más.

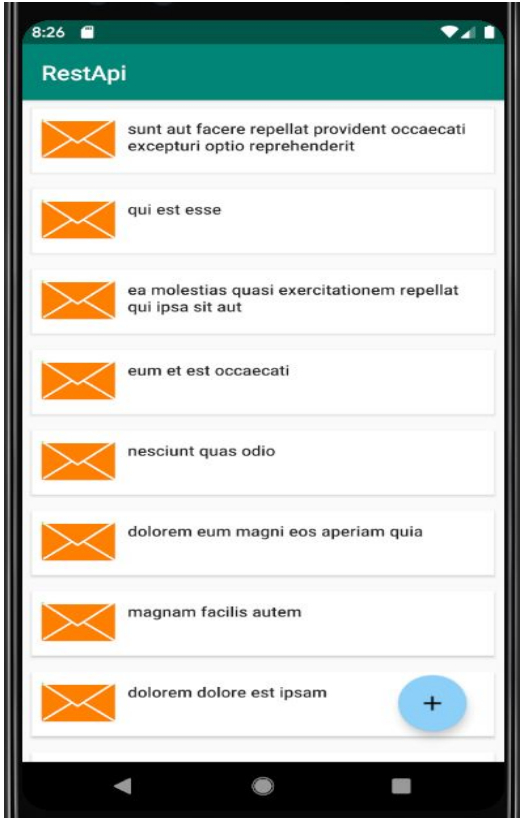
Ejercicio 8

En nuestra aplicación “RestApi”, agregar una imagen descriptiva única para cada post. en el listado del ejercicio anterior, utilizando además las técnicas de caché de imagen, todo con Picasso.

Url imagen: **https://cdn.pixabay.com/photo/2014/03/25/16/54/envelope-297570_960_720.png**

{desafío} latam_

Ejercicio 8 - Solución



Glide

Glide es un marco de carga de imágenes y gestión de medios de código abierto (open source) rápido y eficiente para Android, que ofrece una alternativa a Picasso. Glide envuelve la decodificación de medios, el almacenamiento en caché de memoria y disco, además de ofrecer la agrupación de recursos en una interfaz simple y fácil de usar.

El enfoque principal de Glide es hacer que el desplazamiento de cualquier tipo de lista de imágenes sea lo más suave y rápido posible, pero Glide también es efectivo para casi cualquier caso en el que necesite buscar, cambiar el tamaño y mostrar una imagen remota.

Implementar Glide

Para agregar las librerías de Glide, lo hacemos de la siguiente forma en nuestro archivo gradle de app:

```
dependencies {
  implementation 'com.github.bumptech.glide:glide:4.10.0'
  annotationProcessor 'com.github.bumptech.glide:compiler:4.10.0'
}
```

Por último para implementar glide en nuestras aplicaciones, podemos seguir el siguiente ejemplo en una actividad:

Implementar Glide

```
// For a simple view:
@Override public void onCreate(Bundle savedInstanceState) {
  ImageView imageView = (ImageView) findViewById(R.id.my_image_view);
  Glide.with(this).load("http://goo.gl/gEgYUd").into(imageView);
}
// For a simple image list:
@Override public View getView(int position, View recycled, ViewGroup container) {
  final ImageView myImageView;
  if (recycled == null) {
    myImageView = (ImageView) inflater.inflate(R.layout.my_image_view, container, false);
  } else {
    myImageView = (ImageView) recycled;
  }
  String url = myUrls.get(position);
  Glide
    .with(myFragment)
    .load(url)
    .centerCrop()
    .placeholder(R.drawable.loading_spinner)
    .into(myImageView);

  return myImageView;
}
```

Fresco

Fresco es una biblioteca poderosa para mostrar imágenes en Android. Descarga y almacena en caché imágenes remotas de manera eficiente en la memoria, utilizando una región especial de memoria no recolectada en Android llamada ashmem.

Al trabajar con Fresco, es útil estar familiarizado con los siguientes términos:

- ImagePipeLine
- Drawee

Implementando Fresco

```
dependencies {
 implementation 'com.facebook.fresco:fresco:1.13.0'
}
```

Inicializando Fresco:

```
Fresco.initialize(this)
```

SimpleDraweeView en reemplazo de ImageView en tu archivo layout:

```
<com.facebook.drawee.view.SimpleDraweeView
    android:id="@+id/posterImage"
    android:layout_width="match_parent"
    android:layout_height="500dp"
             --------------
    fresco:placeholderImage="@drawable/ic_broken_image" />
```

Implementando Fresco

Para cargar imágenes el código es el siguiente:

```
posterImage : SimpleDraweeView = findViewById(R.id.posterImage)
imgURI : Uri = Uri.parse("
https://hogaraldia.cl/fit/c/240/240/1*SF2VIRFshYt2etl6OhNm_Q.png")
posterImage.setImageURI(imgURI)
```

{desafío} latam_

Academia de talentos digitales

www.desafiolatam.com