Consumo de API REST

Reflejar objetos JSON en Kotlin data class

{desafío}
latam_

Construir una aplicación Android que consume un servicio REST actualizando la interfaz de usuario, acorde al lenguaje Kotlin y a la librería Retrofit

- Unidad 1:
  Acceso a datos en Android
- Unidad 2:
  Consumo de API REST
- Unidad 3:
  Testing
- Unidad 4:
  Distribución del aplicativo Android

{desafío} latam_

¿Qué aprenderás en esta sesión?

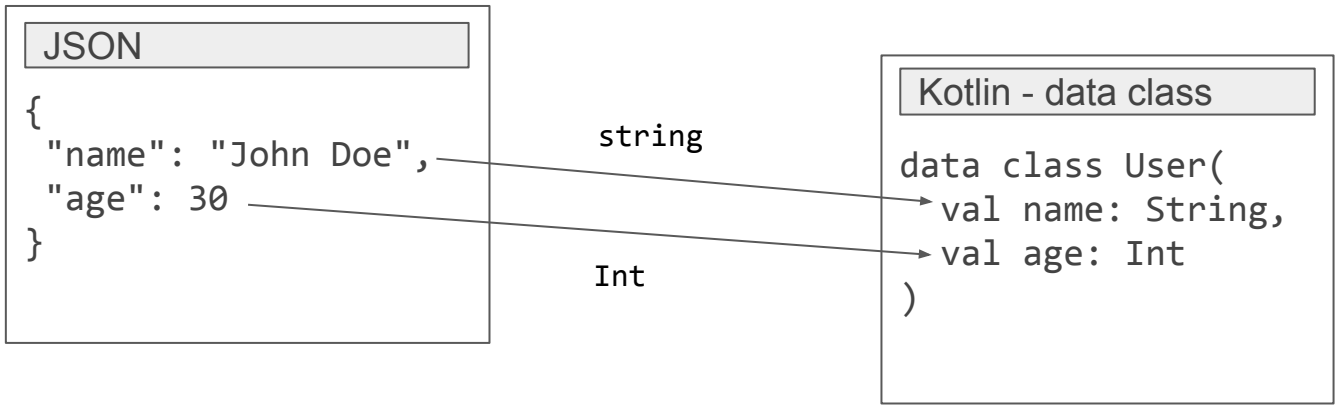
- *Reflejar objetos JSON en Kotlin data class.*
- *Reconocer el uso de DTO classes.*

¿Cuál es la ventaja de usar algo como NetworkResponse?

/* Reflejar objetos JSON en Kotlin data class */

Json a data class

La idea de convertir un objeto JSON a Kotlin, específicamente a data class, es bastante simple, basta con mapear los datos del objeto JSON y crear los equivalentes en un data class.



Json a data class - caso real

Sin embargo, esta tarea se vuelve compleja a medida que el objeto JSON se vuelve más complejo, veamos el siguiente ejemplo:

El siguiente fragmento corresponde al payload de Open Weather Map, puedes acceder a "https://openweathermap.org/current" y ver el json completo

```
{
  "coord": {
    "lon": 10.99,
    "lat": 44.34
  },
  "weather": [
    {
      "id": 501,
      "main": "Rain",
      "description": "moderate rain",
      "icon": "10d"
    }
  ],
...
```

Ejemplo:

A continuación vamos a analizar el objeto json por completo y crearemos la data class correspondiente:

Json:

```
"coord": {
    "lon": 10.99,
    "lat": 44.34
  },
```

Kotlin:

```
data class Coord(
    val lon: Double,
    val lat: Double
)
```

Ejemplo:

A pesar que el objeto weather en el json está definido como un arreglo, la data class en Kotlin no se define como lista.

Json:

```
"weather": [
   {
     "id": 501,
     "main": "Rain",
     "description": "moderate rain",
     "icon": "10d"
   }
 ],
```

Kotlin:

```
data class Weather(
    val id: Int,
    val main: String,
    val description: String,
    val icon: String
)
```

Ejemplo:

Es posible crear la data class solamente con mirar el json, sin embargo, se debe tener cuidado, ya que no sabes si alguno de esos datos pueden ser nulos.

Json:

```
"main": {
    "temp": 298.48,
    "feels_like": 298.74,
    "temp_min": 297.56,
    "temp_max": 300.05,
    "pressure": 1015,
    "humidity": 64,
    "sea_level": 1015,
    "grnd_level": 933
  },
```

Kotlin:

```
data class Main(
    val temp: Double,
    val feels_like: Double,
    val temp_min: Double,
    val temp_max: Double,
    val pressure: Int,
    val humidity: Int,
    val sea_level: Int,
    val grnd_level: Int
)
```

Ejemplo:

Si prestaste atencion, te daras cuenta que saltamos algunos valores, esos se verán a continuación ya que requieren un poco de explicación:

Json:

```
"wind": {
    "speed": 0.62,
    "deg": 349,
    "gust": 1.18
  },

"sys": {
    "type": 2,
    "id": 2075663,
    "country": "IT",
    "sunrise": 1661834187,
    "sunset": 1661882248
  },
```

Kotlin:

```
data class Wind(
    val speed: Double,
    val deg: Int,
    val gust: Double
)
data class Sys(
    val type: Int,
    val id: Int,
    val country: String,
    val sunrise: Int,
    val sunset: Int
)
```

Ejemplo:

Rain es un caso especial, para esto usamos la notación “`@Json`” con el parámetro de nombre para especificar que la clave JSON para la propiedad `oneHour` es "1h", que es un identificador válido en JSON pero no en Kotlin. La notación Json proviene de la biblioteca `com.squareup.moshi`, que se puede usar para analizar datos JSON en Kotlin.

Json:

```
"rain": {
    "1h": 3.16
  },
```

Kotlin:

```
data class Rain(
    @Json(name = "1h") val oneHour: Double
)
```

Ejemplo:

El resto de los valores en el objeto JSON se pueden agregar a la data class final, pon atención en la clase “weather” la cual es un arreglo.

WeatherData es la data class que se encarga de envolver todas las clases y valores del objeto json

Kotlin:

```kotlin
data class WeatherData(
    val coord: Coord,
    val weather: List<Weather>,
    val base: String,
    val main: Main,
    val visibility: Int,
    val wind: Wind,
    val rain: Rain,
    val clouds: Clouds,
    val dt: Int,
    val sys: Sys,
    val timezone: Int,
    val id: Int,
    val name: String,
    val cod: Int
)
```

/* Reconocer el uso de DTO classes */

DTO Classes

- En Android con Kotlin, una clase DTO (objeto de transferencia de datos) es una clase que se usa para representar una estructura de datos que se transferirá entre capas de una aplicación, generalmente a través de una red.

- Las clases DTO a menudo se usan en el desarrollo de Android para separar los datos de la lógica de una aplicación. Al definir la estructura de datos que se transferirá, puede asegurarse de que solo se transfieran los datos necesarios entre las diferentes capas de su aplicación, lo que puede ayudar a reducir el acoplamiento y mejorar el rendimiento.

- Una clase DTO normalmente contiene propiedades que representan los datos que se transfieren. Estas propiedades pueden ser tipos de datos simples, como String, Int, Float, etc., o pueden ser tipos de datos complejos, como otros DTO u objetos de dominio.

DTO Classes - Ejemplo

Aquí hay un ejemplo de una clase DTO en Android con Kotlin:

```kotlin
data class UserDTO(
    val id: Long,
    val name: String,
    val email: String,
    val age: Int
)
```

En este ejemplo, UserDTO es una clase de datos que representa un objeto de usuario que se transferirá entre diferentes capas de una aplicación. Contiene cuatro propiedades: id, nombre, correo electrónico y edad. Estas propiedades representan los datos que se transfieren y sus tipos de datos indican el tipo de datos que se transfieren.

DTO Classes - Beneficios

El uso de clases DTO puede proporcionar varios beneficios:

- **Evite el acoplamiento entre capas:** las clases DTO le permiten separar los datos de la lógica, lo que puede reducir el acoplamiento entre diferentes capas de su aplicación. Esto puede ayudar a que su código sea más modular, más fácil de probar y más fácil de mantener.

- **Facilite el mapeo de datos:** las clases DTO pueden facilitar el mapeo de datos entre diferentes capas de su aplicación. Por ejemplo, puede usar una clase DTO para asignar datos de una respuesta de red a un objeto de dominio o de un objeto de dominio a una entidad de base de datos.

DTO Classes - Beneficios

El uso de clases DTO puede proporcionar varios beneficios:

- **Rendimiento mejorado:** las clases de DTO pueden ayudar a mejorar el rendimiento de su aplicación al reducir la cantidad de datos transferidos a través de la red. Al definir la estructura de datos que se transferirá, puede asegurarse de que solo se transfieran los datos necesarios.

- **Compatibilidad con API externas:** las clases DTO se pueden usar para definir la estructura de datos esperada por las API externas. Mediante el uso de clases DTO, puede asegurarse de que su aplicación sea compatible con las API externas y se pueda adaptar fácilmente a los cambios en la API.

Demostración "Mapper"

Demo - Mapper

En la siguiente demostración usaremos la data class creada anteriormente y crearemos un par de funciones mapper que nos permitan esas data class a clases DTO (Data Transfer Object)

Demo - Mapper

Recordemos que una clase DTO, es una clase que está diseñada para representar un objeto simple o una estructura de datos que se usa para transferir datos entre diferentes partes de una aplicación.
El objetivo principal de una clase DTO es encapsular los datos y mantenerlos separados de la lógica de negocio y otras partes de la aplicación. Esto permite una clara separación de preocupaciones, lo que facilita el mantenimiento y la prueba de las diferentes partes de la aplicación.

En una clase DTO de Kotlin, los datos generalmente se representan mediante propiedades, que se definen mediante la sintaxis de propiedades de Kotlin.

Demo - Mapper

```
data class WeatherMainDto(
    val coord: Coord,
    val weather: List<Weather>,
    val base: String,
    val main: Main,
    val visibility: Long,
    val wind: Wind,
    val rain: Rain,
    val clouds: Clouds,
    val dt: Long,
    val sys: Sys,
    val timezone: Long,
    val id: Long,
    val name: String,
    val cod: Long
)
```

Ya viste lo que es una clase DTO y Mapper, ¿recuerdas lo que es una clase Entity en Room?

{desafío} latam_

Próxima sesión...

- *REST y autenticación*
- *Buenas prácticas de autenticación en requests HTTP*

{desafío}
latam_

Academia de
talentos digitales