Concurrencia y REST

En este último curso de AA hemos visto dos elementos muy importantes dentro de las aplicaciones móviles, la concurrencia la cual nos permite ejecutar procesos en paralelo liberando el hilo principal de la aplicación y por otro lado el consumo de una API que con el auge de la nube la persistencia de información está pasando a un modelo híbrido (las apps persisten tanto en el terminal como en la nube) como cloud donde la persistencia estará totalmente fuera del terminal. Para poder afianzar el conocimiento teórico hicimos una aplicación básica que consumía la API en background y con esta destreza finalizada hicimos la práctica final donde se hacía el consumo completo de una API con alguno de sus diferentes métodos (GET, PUT, POST, DELETE).

Anotaciones

En este tema vimos alguna de las anotaciones para lanzar procesos en background y posteriormente poder modificar la UI así como anotaciones de consumo API REST:

- `@Background` -> Permite lanzar una lógica en un hilo aparte del hilo de UI. Esto no implica que vaya a crear un nuevo hilo sino que puede generar un nuevo proceso dentro de un hilo ya usado por otros procesos (con esta técnica se hace una mejor gestión de los recursos). Ante los ojos del desarrollador veremos que los procesos se ejecutan en paralelo. Algunas de las opciones (parámetros de entrada) que tiene esta anotación:

  - Id -> permite poner un identificador al proceso de background.

  - Serial -> Permite que los procesos de background se ejecuten en orden secuencial

  - Delay -> Permite retrasar la ejecución del método.

- `@UiThread` -> La anotación "@Background" no permite la modificación de la UI (por diseño arquitectural de Android), esta anotación nos permite crear un método capaz de correr en el hilo de la UI. Parámetros de entrada:

  - Delay -> Permite retrasar la ejecución del método.

  - Propagation -> Nos permite activar la opción de "UiThread.Propagation.REUSE" por el cual nos permite reusar el hilo de UI en caso de estar activo, sino lo creará (por defecto, es decir sin ersta opción, siempre crea un nuevo hilo de UI).

  - Id -> permite poner un identificador al proceso de UI thread.

- `@Rest` -> Anotación anexa a una interfaz que nos permitirá generar de forma automática todo el código del cliente de la API. Al menos es bueno especificarle los siguientes parámetros de entrada:

- rootUrl: Endpoint del API REST

- converters: permite especificarle los converters que empleará el cliente.

- `@RestService` -> Instancia la interfaz que haya sido anotada con "@Rest".
- `@Get` -> El prototipo de la interfaz que esté anotado con esta anotación acabará generando de forma automática el código del método GET al path que se le especifíca como variable de entrada de la anotación.
- `@Post` -> El prototipo de la interfaz que esté anotado con esta anotación acabará generando de forma automática el código del método POST al path que se le especifíca como variable de entrada de la anotación.
- `@Put` -> El prototipo de la interfaz que esté anotado con esta anotación acabará generando de forma automática el código del método PUT al path que se le especifíca como variable de entrada de la anotación.
- `@Delete` -> El prototipo de la interfaz que esté anotado con esta anotación acabará generando de forma automática el código del método DELETE al path que se le especifíca como variable de entrada de la anotación.
- `@Path` -> Permite relacionar una variable que viene por URL con la variable de entrada del método.
- `@Body` -> Permite recuerpar el body de la petición.