Desafío - Weather Latam Consumo de API REST (II)

En este desafío validaremos nuestros conocimientos de Consumo de API, Retrofit y StateFlow. Para lograrlo, necesitarás aplicar durante el curso, especialmente lo relacionado a consumo de API, Retrofit, StateFlow, Room, Repository Pattern y MVVM, utilizando de apoyo el archivo: ***Apoyo Desafío evaluado - Consumo de API REST (II)***

Lee todo el documento antes de comenzar el desarrollo **individual**, para asegurarte de tener el máximo de puntaje y enfocar bien los esfuerzos.

Descripción

Este Desafío es la continuación del desafío y se recomienda terminarlo, ya que será ocupado más adelante.

Nuestro cliente, Weather Latam, está muy contento, pues ahora puede obtener los datos desde Openweathermap.com de forma online, sin embargo, nos ha pedido desarrollar una funcionalidad que permita consumir una API en Internet para así poder tener las últimas actualizaciones online y a su vez guardar estos datos en un base de datos local.

Antes de empezar, deberás completar los siguientes pasos, si ya completaste estos pasos en el desafío anterior, entonces, puedes saltarte esta sección e ir directamente a los requerimientos técnicos:

- Para obtener esta información usaremos el servicio de https://openweathermap.org/api
- Puedes seleccionar una de las siguientes key:
  - `fcd51b9342252e2bb7daa90b7f20c2e7`
  - `000477936bfcd7a4b6c887a3a149a0a5`
  - `5938e2f76c5c22ef4b2f9471841e7f0c`.
- El endpoint que usaremos es el siguiente: https://openweathermap.org/current
- Una vez tengas tu Key, reemplaza los siguientes valores en el endpoint: `https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}`, Reemplace los valores dentro los "{}", con los siguientes:
  - Para `lat y lon`, utiliza -33.43107, -70.64666 respectivamente
  - Para API Key, reemplaza con tu developer key

- Ejemplo: https://api.openweathermap.org/data/2.5/weather?lat=-33.43107&lon=-70.64666&appid=02735b3ab7109f96e982d9a42ae2a22e

Usando el archivo de ayuda:

1. Utiliza la interfaz que contiene el servicio para hacer el llamado al endpoint.
2. En la clase `RepositoryImp`, toma los datos que retorna el endpoint y guárdalos en Room, recuerda:
    a. Usa suspend fun para insertar, editar y borrar.
    b. Las queries tipo SELECT deben retornar tipos `Flow<TuObjecto>`
    c. Actualiza las funciones definidas en la interfaz Repository, no confundir con la clase `RepositoryImp`
    d. Crea un mappers para transformar los datos desde el endpoint a entity y desde entity a DTO.
3. Llamas las funciones definidas en `RepositoryImp` desde la clase ViewModel.
4. Llamas las funciones definidas en el ViewModel desde el Fragment que corresponda.
5. Usando StateFlow muestra los datos en pantalla.
6. Agrega Pull To Refresh en la vista principal (Home), para permitir al usuario actualizar los datos, esta funcionalidad debe funcionar de la siguiente forma:
    a. El evento `PullToRefresh` llama una función en el ViewModel.
    b. Esta función llama a la misma función encargada de consultar el endpoint.
    c. Luego el ciclo se repite, se guardan los datos en Room y se muestran los datos desde Room al usuario.
7. Recuerda controlar los posibles errores, por ejemplo 401, 404, 500,
8. Cuando el usuario cambie de Métrico a Imperial en `SettingsFragment`, al volver al Home los datos se deben actualizar automáticamente sin necesidad de ejecutar el Pull To Refresh.

Requerimientos

1. Actualizar la clase `RepositoryImp` para poder trabajar con el endpoint y guardar los datos en Room.
   (**4 Puntos**)

2. Implementar la funcionalidad Pull To Refresh, siguiendo lo que se explica en el punto 6 de los requerimientos técnicos.
   (**3 Puntos**)

3. Mostrar los datos usando StateFlow y usar el método `collect()` correctamente, es decir, cuando la base de datos tenga cambios, estos se deben mostrar en la vista automáticamente.
   (**2 Puntos**)

4. Permitir al usuario cambiar entre Métrico a Imperial y viceversa (tal y como se explica en el punto 8)
   (**1 Puntos**)

 ¡Mucho éxito!

Consideraciones y recomendaciones

- Pon mucha atención a las guías de ejercicio, serán de gran ayuda, especialmente las últimas Actividades Guiadas
- Recuerda usar los mappers para transformar datos desde endpoint a entities y a clases DTO, por ejemplo: no uses clase entites en los fragments.
- Lee los comentarios en el código fuente del proyecto para guiarte.
- Actualizar la clase `RepositoryImp` es la parte más importante del desafío, revisar las guías de ejercicios para apoyarte.