Rodrigo Andrés Mena Caballero 00078421

Preguntas Teóricas

1. ¿Qué es una coroutine en Kotlin y cómo se diferencia de un hilo tradicional?

En este caso el hilo tradicional siempre tiene que acabar una acción para poder continuar con otra, las coroutines permiten suspender y reanudar procesos, y también lleva un mejor orden para facilitar el entendimiento del código.

2. ¿Cuál es la importancia de la suspensión en las coroutines y cómo se implementa?

Es muy importante ya que no va a obstruir el Main Thread de la app y esto permite desarrollar código asíncrono y no bloqueante que pueda manejar actividades largas.
Para implementarse cuando se suspende una coroutine, el hilo que se estaba utilizando se libera para que otras coroutines puedan utilizarlo.

3. ¿Cuál es el propósito del Dispatcher en las coroutines y cómo se elige uno adecuado para cada tarea?

Su propósito es especificar en qué hilo (o hilos) se encuentra una coroutine y esto ayuda en las tareas de Input/Output de red o de archivos, que no pueden realizarse en el hilo principal.

- Dispatchers.Main: Este ejecuta coroutines en el hilo principal de la aplicación. Esto se puede utilizar para actualizar la interfaz de usuario o llevar a cabo otras acciones que requieren la participación de la interfaz de usuario.
- Dispatchers.IO: Este se usa para operaciones de entrada o salida de datos intensivos.
- Dispatchers.Default: Este se usará para tareas de computación intensivas.

Se pueden crear dispatchers personalizados para adaptarse a una necesidad que se tenga.

4. ¿Cuál es el propósito y el uso de la función async en las coroutines?

Es un constructor utilizado para lanzar coroutines de manera asíncrona y se obtiene el resultado del mismo utilizando el objeto **Deferred**. Luego el resultado se obtiene mediante la función **Await()**.

Principalmente será útil cuando se ejecutan tareas concurrentes dentro de la app y para combinar los resultados.