1- ViewBinding. Si tengo una clase que se llama Cliente_Main ¿cómo se llamará la clase generada automáticamente, que al inflar su layout tendremos disponibles todos los objetos?

Si tienes una clase llamada **Cliente_Main**, la clase generada automáticamente por **ViewBinding** se llamará **ClienteMainBinding**. Esta clase será utilizada para inflar su layout y tener acceso directo a todos los objetos definidos en el archivo XML asociado (por ejemplo, **activity_cliente_main.xml**).

2- ViewModel. Completa las frases.

a. ViewModel es un patrón recomendado en aplicaciones que siguen la arquitectura Model-View-ViewModel

b. Existe una separación de responsabilidades. El ____View_____ maneja la interfaz de usuario (UI) y el _____ViewModel_____ gestiona los datos y la lógica.

c. ____ViewModel____ es una clase que sirve para almacenar los datos de una interfaz de usuario de manera optimizada para los ciclos de vida. Proporciona datos dinámicos mediante la clase _____LiveData_____

d. _____LiveData_____es una clase de contenedor de datos observable. Solo actualiza observadores de componentes de apps que tienen un estado de ciclo de vida activo.

e. Un observador está representado por la clase ___Observer__

f. Un observador está en estado activo si el estado correspondiente al ciclo de vida es ____STARTED_____ o _____RESUMED___

3- Crear una Actividad con la plantilla ViewModel-NavigationDrawer. Qué pasos tienes que dar si quieres añadir otro elemento al menú lateral.

- Editar el XML del menú: Añade un nuevo <item> en el archivo del menú (res/menu/activity_main_drawer.xml).
- Actualizar el controlador: Gestiona la acción del nuevo elemento en onNavigationItemSelected().
- Agregar funcionalidad: Si es necesario, crea un nuevo fragmento o actividad para vincularlo al elemento.

4- Notificaciones.

a. ¿Qué son las notificaciones?

Son mensajes que las apps envían para informar al usuario, incluso cuando no están activas, mostrándose en la barra de notificaciones.

b. ¿Qué clase se utiliza para crear una notificación?

Se utiliza la clase **NotificationCompat.Builder**.

5- Receptor de Multidifusión.

a. ¿Qué es un receptor de multidifusión o Broadcast Receiver?

Un **Broadcast Receiver** es un componente de Android que permite a las aplicaciones escuchar eventos del sistema o de otras aplicaciones. Estos eventos pueden ser notificaciones globales como cambios de conectividad, batería baja, o mensajes personalizados enviados a través de difusiones.

b. Pon 3 o 4 ejemplos de Broadcast Receiver.

- android.intent.action.BOOT_COMPLETED: Se activa cuando el dispositivo termina de arrancar.
- android.net.conn.CONNECTIVITY_CHANGE: Detecta cambios en la conectividad de red.
- android.intent.action.BATTERY_LOW: Notifica que la batería está baja.
- android.intent.action.ACTION_POWER_CONNECTED: Indica que el dispositivo se ha conectado a una fuente de alimentación.

c. Describe los dos pasos que tienes que dar para poder recibir estas notificaciones del sistema.

1- Crear una subclase de la clase BroadcastReceiver e implementar el método onReceive() que recibirá un intent a través del cual se puede saber que notificación hemos recibido.

2- Registrar la subclase en el fichero de manifiesto de nuestra App como candidata a recibir ciertas notificaciones

Declara el receptor en el manifiesto utilizando la etiqueta <receiver> y especifica el filtro de intent correspondiente. Por ejemplo:

```
<receiver android:name=".ReceptorAvion">
  <intent-filter>
    <action android:name="android.intent.action.AIRPLANE_MODE" />
  </intent-filter>
</receiver>
```

Si se necesitan permisos específicos, también deben declararse en el manifiesto.