**Sesión 4-01 Clase del 6 de febrero**

En esta sesión vamos a realizar una programación básica del uso y comprobación de la conectividad de red del dispositivo. Se trata de comprobar si el dispositivo tiene acceso a Internet y si lo tiene a través de la red wifi y/o a través de la red de datos.

1.- Lo primero que tienes que hacer es crear un archivo **HomeView.kt** para la única pantalla que tendremos en la app y una clase ViewModel para proveer datos a **HomeView**. El código base para el **ViewModel** puede ser este:

class ConnectivityViewModel(application: Application) : AndroidViewModel(application) {  
 //contexto de la aplicación  
 private val context = application.*applicationContext  
}*

2.- En **MainActivity** debes instanciar el **ViewModel** y llamar a la pantalla **HomeView** pasándole el **ViewModel**:

val viewModel: ConnectivityViewModel by *viewModels*()  
*setContent* **{** Sesion4\_01\_ConectividadTheme **{** HomeView(viewModel)  
 **}  
}**

3.- El código para el diseño de la pantalla **HomeView** es muy básico, puedes usar éste:

Scaffold(  
 ) **{** padding **->** Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*padding*(padding),  
 verticalArrangement = Arrangement.Center,  
 horizontalAlignment = Alignment.CenterHorizontally  
 ) **{** TextField(  
 value = textUrlState,  
 onValueChange = **{** textUrlState= **it }**,  
 label = **{** Text("Ingrese la URL") **}**,  
 modifier = Modifier.*fillMaxWidth*().*padding*(16.*dp*)  
 )  
 Button(  
 onClick = **{ }**,  
 modifier = Modifier.*padding*(top = 16.*dp*)  
 ) **{** Text("Descargar")  
 **}** Text(  
 text = content,  
 modifier = Modifier.*padding*(16.*dp*),  
 textAlign = TextAlign.Center  
 )  
 **}  
}**

Añade las variables de estado **content** y **textUrlState** para que no haya errores.

4.- Como vamos a trabajar con comprobaciones de estado de conexión a redes, necesitamos agregar el permiso **ACCESS\_NETWORK\_STATE** al archivo de manifiesto.

5.- En el **ViewModel** vamos a añadir un LiveData para informar si el dispositivo tiene conectividad o no:

private val \_isConnected = MutableLiveData<Boolean>(false) // Conectividad  
val isConnected: LiveData<Boolean> = \_isConnected

Y añade un LiveData **\_content** para contener un texto del estado de conexión

En el **iniciador** del **ViewModel** programamos que se llame a un método interno para comprobar la conectividad:

init {  
 checkConnectivity()  
}

Y ahora, programamos la función **checkConnectivity().**

// Verifica conectividad (WiFi o datos móviles)  
private fun checkConnectivity() {  
 //obtener el servicio de conectividad  
 val connectivityManager =  
 context.getSystemService(Context.*CONNECTIVITY\_SERVICE*) as ConnectivityManager  
 //obtener las capacidades de red activas (WiFi, datos móviles, VPN, Ethernet)  
 val activeNetwork = connectivityManager.*activeNetwork* val networkCapabilities = connectivityManager.getNetworkCapabilities(activeNetwork)  
 //variable que comprueba si una de las redes activas es WIFI  
 val isWifiConnected =  
 networkCapabilities?.hasTransport(NetworkCapabilities.*TRANSPORT\_WIFI*) == true  
 if (isWifiConnected) {

\_content.postValue("Conectado a WiFi")

} else {  
 \_content.postValue("NO Conectado a WiFi")

}  
}

En **HomeView,** declara una variable de estado **isConnected** observadora del LiveData **isConnected** y modifica **content** para que sea observadora del LiveData **content.**

Prueba la ejecución y veras que en el componente **Text** se muestra el contenido de **content** y, por tanto, si hay conectividad wifi o no.

6.- Ahora queremos implementar que, en caso de no haber wifi, si hay conexión de datos se pregunte en un cuadro de diálogo si se quiere hacer uso de los datos.

6.1.- Dentro de la pantalla **Homeview,** una vez terminado el código del componente **Column**, debes añadir el siguiente código para que se muestre un cuadro de diálogo cuando una variable de su estado indique que debe mostrarse. En ese cuadro de diálogo se pregunta si se quieren usar la conexión de datos:

// Diálogo para datos móviles  
if (showMobileDataDialog) {  
 AlertDialog(  
 onDismissRequest = **{ }**,  
 title = **{** Text("Usar datos móviles") **}**,  
 text = **{** Text("No se detectó conexión WiFi. ¿Desea usar datos móviles?") **}**,  
 confirmButton = **{** TextButton(onClick = **{** viewModel.acceptUsingMobileData() **}**) **{** Text("Sí")  
 **}  
 }**,  
 dismissButton = **{** TextButton(onClick = **{** /\* No hacer nada \*/ **}**) **{** Text("No")  
 **}  
 }** )  
}

Para que este código funcione, habrá que declarar la variable de estado **showMobileDataDialog,** además de implementar en el **ViewModel** la función **acceptUsingMobileData.** Posteriormente se indicará como y cuando hacerlo.

6.2.- Volvemos al **ViewModel** y añadimos un LiveData para controlar si el cuadro de diálogo de **HomeView** debe mostrarse:

private val \_showMobileDataDialog = MutableLiveData<Boolean>(false) // Mostrar diálogo  
val showMobileDataDialog: LiveData<Boolean> = \_showMobileDataDialog

Y añadimos un **LiveData** para indicar si se van a usar los datos del dispositivo para la conexión:

private val \_isUsingMobileData = MutableLiveData<Boolean>(false) // Uso de datos móviles  
val isUsingMobileData: LiveData<Boolean> = \_isUsingMobileData

6.3.- En la función **checkConnectivity** añadimos una variable que contenga el estado de la conectividad de datos del dispositivo:

//variable que indica si hay conexión de datos móviles  
val isMobileDataConnected =  
 networkCapabilities?.hasTransport(NetworkCapabilities.*TRANSPORT\_CELLULAR*) == true

6.4.- Y modificamos código para que, encaso de no haber wifi y haber conexión de datos, se establezca que se muestre el cuadro de diálogo.

if (isWifiConnected) {  
 \_isConnected.postValue(true)  
 \_content.postValue("Conectado a WiFi")  
} else if (isMobileDataConnected) {  
 \_showMobileDataDialog.postValue(true) // Preguntar si usar datos móviles  
 \_content.postValue("consulta de uso datos móviles")  
} else {  
 \_isConnected.postValue(false)  
 \_content.postValue("No hay conexión de red.")  
}

6.5.- Añadimos en **ViewModel** la función **acceptUsingMobileData** referenciada en 6.1.

// Usuario aceptó usar datos móviles  
fun acceptUsingMobileData() {  
 //se usan datos móviles  
 \_isUsingMobileData.postValue(true)  
 //ya no hay que mostrar el cuadro de diálogo  
 \_showMobileDataDialog.postValue(false)  
}

6.6.- Sólo te falta declarar **HomeView** la variable **showMobileDataDialog** de estado observadora del correspondiente LiveData y probar la funcionalidad.

7.- Ahora vamos a hacer que cuando el dispositivo tenga wifi o se haya permitido conectividad por datos, se muestren en **HomeView** los componentes que hay dentro de **Column.** Para ello puedes usar este código dentro del contenido de Column:

if (isConnected || isUsingMobileData) {

// aquí van todos los componentes que había en column

}

Tendrás también que declarar la variable de estado observadora **isUsingMobileData.**

8.- Vamos a usar por primera vez un ejemplo de una notificación con un componente **SnackBar.** En caso de no haber conectividad, se escribirá un snackbar con un mensaje de error.

8.1.- En **HomeView** debes declarar una variable de estado para controlar si el SnackBar deberá mostrarse o no:

// Snackbar para mensajes de error  
val snackbarHostState = remember **{** SnackbarHostState() **}**

8.2.- Dentro del **Scaffold**, en una propiedad para el SnackBar de la pantalla asignamos un componente **SnackBar** que se muestra en función de la variable de estado anterior.

snackbarHost = **{** SnackbarHost(snackbarHostState) **}**

8.3.- En **ViewModel** añadimos un **LiveData** para los mensajes de error que se podrán notificar en el SnackBar:

private val \_errorMessage = MutableLiveData<String>("") // Mensaje de error  
val errorMessage: LiveData<String> = \_errorMessage

8.4.- Modificamos el código de **checkConnectivity** para usar ese estado del **LiveData** cuando no haya conexión. Dejamos de usar el LiveData **content** en esta parte de código, lo usaremos después para otra cosa.

if (isWifiConnected) {  
 \_isConnected.postValue(true)  
 } else if (isMobileDataConnected) {  
 \_showMobileDataDialog.postValue(true) // Preguntar si usar datos móviles  
} else {  
 \_isConnected.postValue(false)  
 \_errorMessage.postValue("No hay conexión de red.")  
}

8.5.- En **HomeView** declaramos una variable de estado observadora del LiveData **errorMessage**.

8.6.- Establecemos que al lanzarse **HomeView,** si no hay conexión se pueda mostrar la notificación en el **SnackBar:**

// Mostrar Snackbar si hay mensaje de error  
LaunchedEffect(errorMessage) **{** if (errorMessage.*isNotEmpty*()) {  
 snackbarHostState.showSnackbar(errorMessage)  
 }  
**}**

9.- Nos falta programar el botón **Descargar** de **HomeView.** Se trata de que, al clicarlo, se descargue el contenido de la url escrita en el **TextField.** El contenido descargado se tiene que pasar a **content** para así escribirlo en el componente **Text.**

9.1.- Establece en **onClick** del **Descargar** que se llame al método de descarga que se implementará en el **ViewModel** pasándole la url editada en el **TextView.**

viewModel.downloadContent(urlState.value)

9.2.- Añade el método de descarga al ViewModel.

Texto

Descripción generada automáticamente