**Sesión 1-01 Clase del 25 septiembre**

1.- Tras una instalación previa de Android Studio. Iniciamos el IDE por primera vez.

2.- Android nos pedirá instalar un software adicional y necesario para el IDE. Este software es personal para la cuenta de usuario que lo instala (se instala en su carpeta de perfil). El software que se instala en este proceso es:

* El conjunto de herramientas y librerías que permiten desarrollar para SO Android (SDK, Software Development Kit). Incluirá la versión más reciente del SDK de Android, la Versión U (API 34) y herramientas como **adb** para realizar acciones sobre dispositivos físicos o sobre emuladores.
* **Android Virtual Device (AVD) Manager**: Para gestionar dispositivos emulados que permitan probar aplicaciones. No es lo mismo **adb** que **AVD Manager**.
* Las **Build Tools,** herramientas que permiten compilar y construir aplicaciones. **Gradle**, por ejemplo, es una herramienta para construir aplicaciones y gestionar las dependencias.
* Herramientas de depuración y monitoreo como **debugger, LogCat** para ver los logs de depuración, y **Profiler**, para monitorear el rendimiento de las aplicaciones.
* **Repositorio de Google (Google Repository)** y **Repositorio de soporte de Android (Android Support Repository)** donde se puede acceder a muchas de las dependencias y bibliotecas específicas para Android.
* **HAXM**, es un motor de virtualización que hace que el emulador utilice la aceleración por hardware del procesador, lo que mejora significativamente la velocidad y el rendimiento. Es para procesadores **Intel.**

3.- Tras haber instalado, iniciamos un nuevo proyecto para **UI XML.**

Entre las plantillas de proyecto existente, seleccionamos las de **Phone and Tablet y,** dentro de ellas, la plantilla más básica, **Empty View Activity.**

4.- Establecemos unos datos iniciales de configuración del Proyecto:

* Nombre del proyecto
* Nombre del paquete principal del proyecto, usando convención de nombrado **reverse domain name.**
* Ruta o ubicación de la carpeta de proyecto.
* Versión mínima de Android en que se podrá ejecutar la aplicación que se genere con el proyecto.
* Lenguaje de construcción del proyecto (**Gradle**)

5.- Ver estructura el proyecto.

* Usar las vistas **Proyecto** (como es realmente la estructura de carpetas) y **Android** para ver los ficheros que contiene el proyecto.
* El archivo de manifiesto **AndroidManifest.xml.**
* La carpeta **res** de recursos:
  + **drawable** para contener algunos archivos de imagen que se usarán en algunas pantallas de la aplicación.
  + **mipmap** para contener los archivos de imagen de los iconos que permitirán lanzar a ejecución la aplicación en los dispositivos.
  + **layout** contiene los archivos **XML** que describen los elementos que componen cada pantalla y como se distribuyen. Vemos el archivo de la pantalla principal **activity\_main.xml.** Vemos este archivo en vista diseño (gráfica) y en vista código (XML).
  + **values/strings.xml**: Para definir las cadenas de texto de la aplicación y, en un futuro, adaptar la aplicación a varios idiomas.
  + **values/colors.xml**: Para definir una paleta de colores que se usrá en la aplicación.
  + **values/dimens.xml**: Para tamaños (dimensiones) como márgenes y padding.
  + **values/styles.xml**: Para definir estilos reutilizables para varios elementos de las pantallas (en Android Studio a las pantallas se las denomina **activities)**.
* **Ficheros buid.gradle** (de proyecto y de módulo). En el de proyecto se registran propiedades que afectan a todo el proyecto e indican versiones de pluggins, repositorios usados, configuraciones para el proceso de compilación, etc. En el de módulo se indican pluggins para el módulo, versión del SDK con la que se compila, mínimo SDK en el que la aplicación es ejecutable, versión SDK para la que se quiere que la aplicación esté optimizada (**targetSDK**), código y nombre para la versión de la aplicación y, lo más importante, las dependencias incluidas.
* La clase **MainActivity** donde se codifica en **kotlin** la funcionalidad de la pantalla principal. Cada pantalla de la aplicación deberá tener una clase **Activity** asociada y un fichero **XML** de descripción de la pantalla.

6.- Iniciamos ahora un nuevo proyecto para **UI Jetpack Compose** seleccionando la única plantilla para estos UI, la plantilla **Empty Activity.**

7.- Vemos la estructura del proyecto ya las diferencias con los de UI XML. Aquí realmente no hay ficheros XML de layout y, por tanto, no vemos el diseño de la pantalla hasta que ejecutamos la aplicación, sólo podemos ver una previa de algunos de sus componentes. Con JetPack Compose:

* No se usa la carpeta **/res/layout** y los ficheros **XML** de descripción de pantallas. Las pantallas de declaran y describen en lenguaje **kotlin** y dentro del mismo código en que se programa la funcionalidad de las pantallas (en el mismo fichero).
* Los temas y estilos se gestionan mediante **funcioComposable**. Se usan MaterialTheme y otros componentes similares para aplicar estilos y colores a la UI.
* Los colores y tipografías de los elementos se declaran para los componentes dentro del código que los carga.
* Hay un paquete **ui.theme** que contiene tres archivos kotlin:
  + **color.kt:** Donde se declaran las diferentes paletas de colores que se usarán en la aplicación.
  + **typography.kt**: Define los **tipos de letra (tipografía)** utilizados en la UI (Fuentes, pesos, tamaños).
  + **shape.kt**: Define las **formas** que se aplican a los elementos de la UI. Permite definir bordes redondeados o personalizados.
  + **theme.kt**: Es el archivo principal que integra todos los aspectos del tema: colores, tipografía y formas.

8.- Analizamos el código de la clase **MainActivity** que contiene la función **onCreate** (similar a una función **main).**

* Vemos lo que es una función **Composable.**
* Vemos lo que es una función **@Preview.** Vemos como funcionan las vistas **Preview.**
* Vemos para que sirve el método **setContent.**
* Vemos el uso de objetos y propiedades **modifier.**