Taller Móviles 2

Nombre YEISON ANDRES DIAZ VILLAMARIN

Móviles 2

Apreciado estudiante:

El siguiente taller está diseñado para reforzar los conocimientos teóricos y prácticos de desarrollo de aplicaciones móviles usando JAVA y Android Studio. Una vez culminado esté será depositado en el siguiente enlace de Share Point Entregas Moviles 2 - Martes, donde deberá crear una carpeta adjuntando el mismo, así como la aplicación que usted diseño y desarrollo como proyecto final.

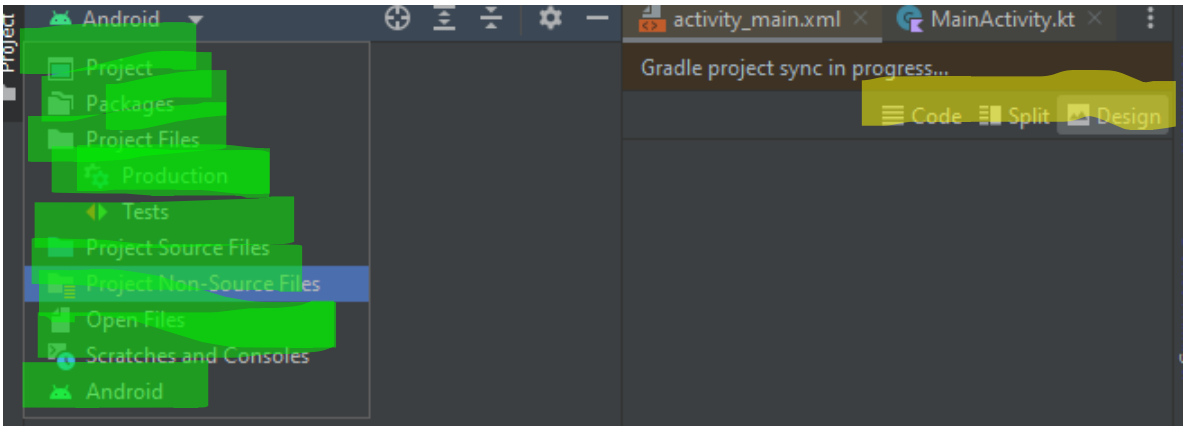
Responda las siguientes preguntas y resuelva los ejercicios propuestos, adjuntando el pantallazo del código.

1. ¿Qué es Android Studio?
2. Señale las principales herramientas de Android Studio e indique se funcionalidad dentro del IDE.
3. Indique los pasos para crear un nuevo proyecto en Android Studio.
4. ¿Qué son las API de Android?
5. ¿Cuáles son los criterios para seleccionar una API?
6. ¿Qué clases de archivos genera el proyecto al crearse?
7. ¿Qué es XML?
8. ¿Qué es una clase de Java?
9. ¿Qué son los componentes?
10. ¿En dónde están ubicados los componentes en IDE?
11. ¿Dé 5 ejemplos de componentes y describa su uso?
12. ¿Qué son los atributos?
13. Indique al menos 5 atributos para los componentes seleccionados.
14. Realice una aplicación que contenga al menos un Text View , un Text Plannig y un Buttom. Que imprima un saludo.
15. ¿Qué son los Layouts , cual es su utilidad y cuáles son los tipos que encontramos en el IDE?
16. ¿Qué es una Activity?
17. ¿Qué es un Intent? De un ejemplo
18. ¿Qué son las shared preferences? De un ejemplo
19. ¿Qué es el ciclo de vida de una app?
20. ¿Qué es el método OnCreate?
21. Indique la utilidad de los diferentes métodos del ciclo de vida de la App
22. ¿Para que sirve la clase adapter?
23. ¿Qué es el Recycler view?
24. ¿Que tipos de bases de datos podemos implementar en una aplicación móvil?
25. ¿Que es ROOM?
26. ¿Para qué sirve Firebase en un proyecto móvil?

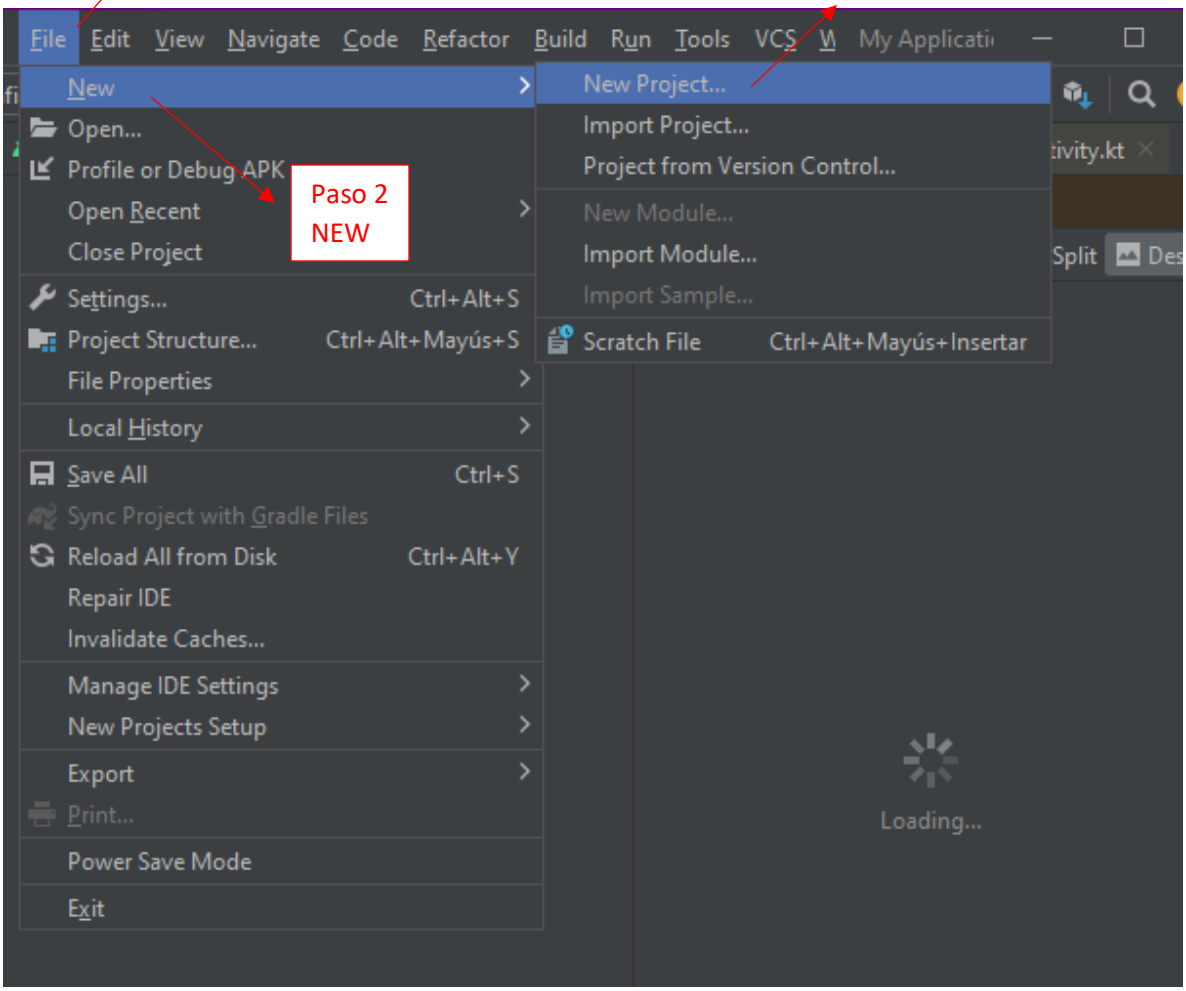
Respuestas

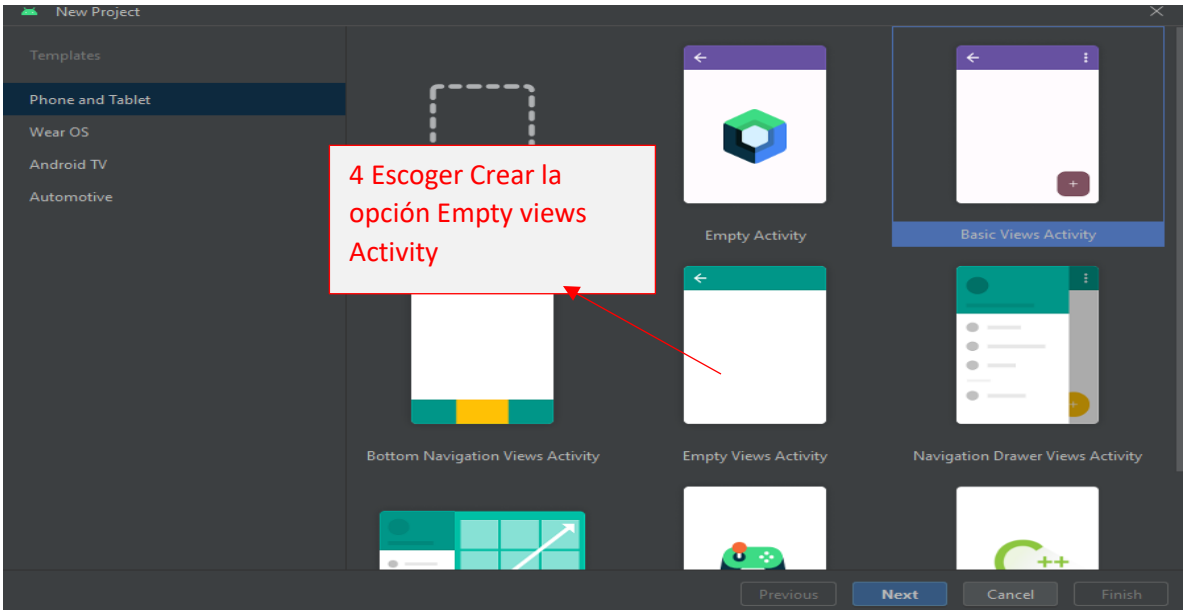
1 Android estudio es una plataforma que implementan los ide de desarrollo para entornos móviles además funciona en base a lenguajes como java y kotlin en la cual forman las estructura del Código en el desarrollo de APP móviles

2 las herramientas principales son las vistas y los modos de trabajo ejemplo vistas Codem,Split,Design que nos ayudan a modificar las características para ver en pantalla donde code nos deja implementar de cierta manera el codigo de la app y a diferencia de design que nos deja interpretar la interfaz también tenemos las herramientas de Ver codigo o interpretarlos en tipo



3:





5 Seleccionas Finish

4: API interfaz de programación de aplicaciones son las encargadas básicamente de conectar dos software mediante conjuntos de herramientas interpretadas en codigo entro otras funciones etc… permiten el desarrollo de software y mejorando su sistema ya que obtener ventajas ya que interactúan entre si

5: Criterios de API

- Fácil de usar y de aprender - Que no sean lentas - Que sean seguras

6: Carpeta src: Contiene todo el código fuente (java) de la aplicación, entre ellas nuestra actividad principal; que se almacenan en src/paquete/Actividad.java.

Carpeta bin: Directorio de salida; aquí se encuentra la .apk y otros recursos procedentes de la compilación el proyecto.

Carpeta libs: Contiene librerías privadas.

Carpeta res: Contiene los recursos (imágenes, vídeos, audio, etc) necesarios para generar una aplicación Android, también los diferentes tipos de recursos que deberán distribuir entre las siguientes carpetas:

res/drawable/: Guarda las imágenes y se divide en: drawable-ldpi, drawable-mdpi y drawable-hdpi, que dependerá de la resolución del dispositivo.

res/raw/: Contiene archivos de propósito general, la única diferencia con el directorio assets es que pueden ser referenciados usando el archivo R.

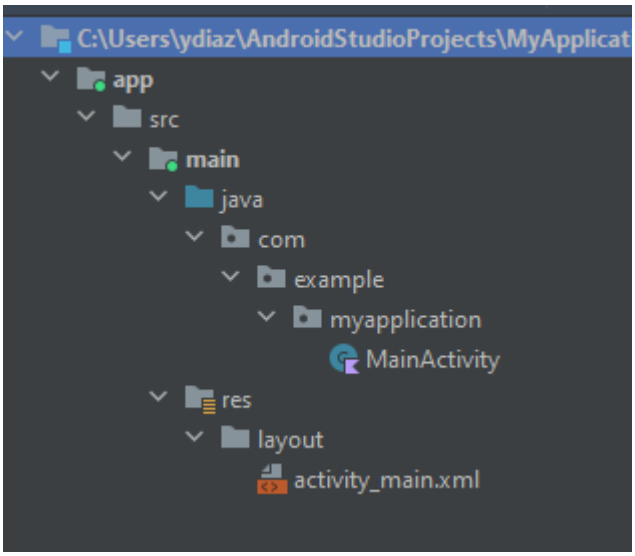
res/layout/: Incluye los archivos que definen el diseño de la interfaz gráfica, siempre en XML.

res/values/: Guarda los datos y tipos que utiliza la aplicación, tales como colores, cadenas de texto, estilos, dimensiones, etc.

Carpeta gen: Ésta carpeta guarda un conjunto de archivos (de código Java) creados automáticamente cuando se compila el proyecto, para poder dirigir los recursos de la aplicación. El archivo R ajusta automáticamente todas las referencias a archivos y valores de la aplicación (guardados en la carpeta res).

Carpeta assets: Guarda el resto de archivos necesarios para el correcto funcionamiento de la aplicación, como los archivos de datos o de configuración. La principal diferencia entre los recursos que almacena ésta carpeta y los que guarda la carpeta res, es que los recursos de ésta última genera un identificador por recurso, que se encargará de gestionar el archivo R y sólo se podrá acceder a ellos a través de determinados métodos de acceso, mientras que los recursos almacenados en la carpeta assets no generan identificador alguno y se accede a ellos a través de su ruta, como se hace con cualquier otro fichero.

Archivo AndroidManifest.xml:Este archivo es uno de los más importantes de cualquier aplicación Android. Se genera automáticamente al crear el proyecto, y en él se encuentra definida la configuración del proyecto en XML.



7: ¿Qué es XML?

XML significa "Extensible Markup Language" Lenguaje de marcado extensible y es un lenguaje de marcado que se utiliza para almacenar y transportar datos de manera estructurada. Se utiliza ampliamente en la web para intercambiar información entre diferentes aplicaciones y sistemas.

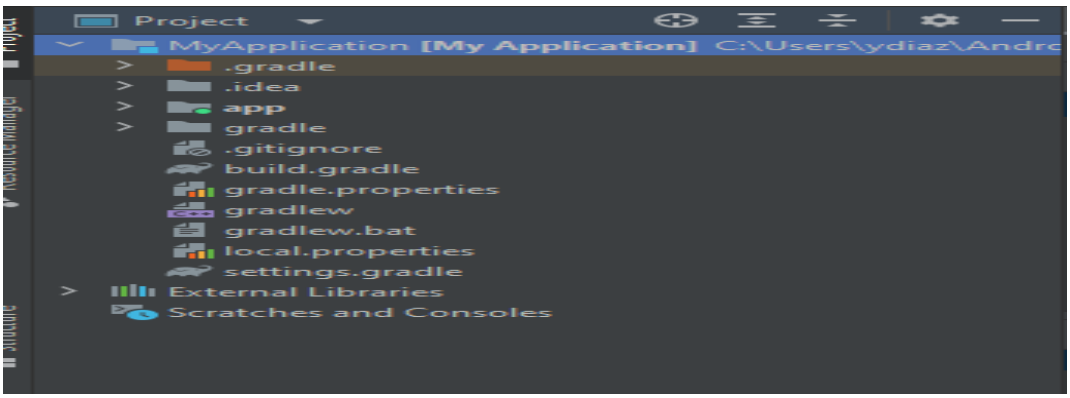
8: Las clases son uno de los subconjuntos de los objetos y son uno de los elementos más importantes para programar en este lenguaje. Estos elementos sirven para crear moldes o plantillas que pueden ser replicados para categorizar objetos con atributos similares.

Todos los elementos en Java están relacionados con una clase o con un objeto. Veamos cómo funciona esto a través de un ejemplo.

Ejemplo una receta es la clase ya que con eso podemos crear diferentes tipos de comidas

9: Los componentes en programación son las partes fundamentales que conforman un programa o sistema de software, como el código fuente, las bibliotecas, los módulos, las funciones, los objetos, entre otros.

10: Lado izquierdo



11: Activity

Las actividades (activity) representan el componente principal de la interfaz gráfica de una aplicación Android. Se puede pensar en una actividad como el elemento análogo a una ventana o pantalla en cualquier otro lenguaje visual.

View

Las vistas (view) son los componentes básicos con los que se construye la interfaz gráfica de la aplicación, análogo por ejemplo a los controles de Java o .NET. De inicio, Android pone a nuestra disposición una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, aunque también existe la posibilidad de extender la funcionalidad de estos controles básicos o crear nuestros propios controles personalizados.

Fragment

Los fragmentos (fragment) se pueden entender como secciones o partes (habitualmente reutilizables) de la interfaz de usuario de una aplicación. De esta forma, una actividad podría contener varios fragmentos para formar la interfaz completa de la aplicación, y adicionalmente estos fragmentos se podrían reutilizar en distintas actividades o partes de la aplicación. No es obligatorio utilizar fragmentos en una aplicación, pero sí nos serán de mucha ayuda en ciertas ocasiones, por ejemplo para adaptar la interfaz de nuestra aplicación a distintos dispositivos, tamaños de pantalla, orientación, etc.

Service

Los servicios (service) son componentes sin interfaz gráfica que se ejecutan en segundo plano. Conceptualmente, son similares a los servicios presentes en cualquier otro sistema operativo. Los servicios pueden realizar cualquier tipo de acción, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (p.ej. actividades) si se necesita en algún momento la interacción con del usuario.

Content Provider

Un proveedor de contenidos (content provider) es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones. Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación. De la misma forma, nuestra aplicación podrá acceder a los datos de otra a través de los content provider que ésta última haya definido.

Broadcast Receiver

Un broadcast receiver es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: “Batería baja”, “SMS recibido”, “Tarjeta SD insertada”, …) o por otras aplicaciones (cualquier aplicación puede generar mensajes (intents, en terminología Android) de tipo broadcast, es decir, no dirigidos a una aplicación concreta sino a cualquiera que quiera escucharlo)

Widget

Los widgets son elementos visuales, normalmente interactivos, que pueden mostrarse en la pantalla principal (home screen) del dispositivo Android y recibir actualizaciones periódicas. Permiten mostrar información de la aplicación al usuario directamente sobre la pantalla principal

Intent

Un intent es el elemento básico de comunicación entre los distintos componentes Android que hemos descrito anteriormente. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones. Mediante un intent se puede mostrar una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc.

12:
Los atributos en Android Studio son propiedades que se pueden agregar a un elemento de la interfaz de usuario para personalizar su apariencia o comportamiento.

13: Los atributos para los componentes seleccionados en Android Studio pueden variar dependiendo del componente en cuestión, pero algunos ejemplos comunes incluyen: tamaño, posición, color, texto, visibilidad, entre otros.

14:

15:
Los Layouts son elementos que se utilizan en Android Studio para organizar y estructurar la interfaz gráfica de una app. Su utilidad es permitir que los elementos visuales se distribuyan de manera ordenada y se adapten a diferentes tamaños de pantalla. Los tipos de Layouts que podemos encontrar en el IDE son: LinearLayout, RelativeLayout, ConstraintLayout, GridLayout, FrameLayout y TableLayout.

16: Una Activity es un componente de una aplicación de Android que representa una única pantalla con una interfaz de usuario.

17: Un Intent explícito se utiliza para iniciar un componente específico dentro de una aplicación, es decir, se especifica el nombre del componente que se desea iniciar. Por otro lado, un Intent implícito se utiliza para iniciar un componente que pueda manejar una acción específica, sin especificar el componente en sí.

Por ejemplo, si se desea abrir una actividad específica dentro de una aplicación, se utilizaría un Intent explícito. Si se desea abrir cualquier actividad que pueda manejar la visualización de un archivo PDF, se utilizaría un Intent implícito con la acción de visualización de PDF.

18: Las shared preferences son un mecanismo de almacenamiento de datos en Android que permite guardar información en pares clave-valor. Un ejemplo de uso de las shared preferences sería guardar la preferencia del usuario sobre el idioma de la aplicación para que la próxima vez que abra la app, se muestre en el idioma seleccionado.

19: El ciclo de vida de una app es el proceso completo que sigue una aplicación móvil desde su concepción hasta su retirada del mercado. Incluye etapas como el diseño, la programación, las pruebas, el lanzamiento, el mantenimiento y la actualización.
El ciclo de vida de una aplicación Android dentro del Código consta de los siguientes estados:
OnCreate
OnStart
OnResume
OnPause
OnStop

20: El método OnCreate es un método en la clase Activity de Android que se llama cuando se crea la actividad por primera vez.

21: Aquí hay un resumen de algunos de los métodos del ciclo de vida de la App:

- **onCreate()**: se llama cuando se crea la actividad por primera vez. Aquí se pueden inicializar los componentes de la interfaz de usuario, establecer los oyentes de eventos, obtener datos de otras actividades, etc[1].
- **onStart()**: se llama cuando la actividad está a punto de hacerse visible al usuario. Aquí se pueden realizar operaciones que preparan la actividad para mostrarse en la pantalla[1].
- **onResume()**: se llama cuando la actividad empieza a interactuar con el usuario. Aquí se pueden reanudar las tareas que se pausaron en el método onPause(), como reproducir música, actualizar la interfaz de usuario, etc[1].
- **onPause()**: se llama cuando el usuario deja de interactuar con la actividad. Aquí se pueden pausar las tareas que consumen recursos o que no son necesarias cuando la actividad no está en primer plano, como detener música, liberar la cámara, guardar datos, etc[1].
- **onStop()**: se llama cuando la actividad ya no es visible al usuario. Aquí se pueden liberar los recursos que no son necesarios cuando la actividad está oculta, como cancelar solicitudes de red, detener animaciones, etc[1].
- **onDestroy()**: se llama cuando la actividad está a punto de ser destruida por el sistema o por el usuario. Aquí se pueden realizar operaciones de limpieza, como cerrar conexiones de base de datos, eliminar archivos temporales, etc[1].

22: La clase adapter es un patrón de diseño que permite adaptar una interfaz para que pueda ser utilizada por una clase que de otro modo no podría ser utilizada. Por ejemplo, si tenemos una clase que opera con datos en formato XML y otra que opera con datos en formato JSON, podemos crear un adapter que convierta los datos de un formato a otro y permita la comunicación entre las clases. Un adapter tiene dos componentes esenciales:

Primero, necesitamos una clase que defina la interfaz que tiene que ser adaptada, el “adaptado”. Segundo, necesitamos la clase “adaptador” que defina la interfaz esperada y haga corresponder los métodos de la interfaz con aquellos definidos por el adaptado.

23: RecyclerView es un widget que es una versión más flexible y avanzada de GridView y ListView. Es un contenedor para mostrar grandes conjuntos de datos que se pueden desplazar de manera eficiente manteniendo un número limitado de vistas. RecyclerView optimiza el rendimiento al reciclar las vistas que ya no son visibles en lugar de crear nuevas cada vez.

24: Existen diferentes tipos de bases de datos que se pueden implementar en una aplicación móvil, dependiendo de las necesidades y características de la aplicación. Algunos de los tipos más comunes son:
- Bases de datos relacionales: son aquellas que almacenan los datos en tablas con filas y columnas, y que se basan en el modelo relacional. Permiten realizar consultas complejas y garantizar la integridad y consistencia de los datos. Algunos ejemplos de sistemas de gestión de bases de datos relacionales (SGBDR) son SQLite, MySQL, PostgreSQL, Oracle, etc.
- Bases de datos no relacionales: son aquellas que almacenan los datos en formatos distintos a las tablas, como documentos, clave-valor, grafos o columnas. Se basan en el modelo NoSQL (Not only SQL) y ofrecen mayor flexibilidad, escalabilidad y rendimiento que las bases de datos relacionales. Algunos ejemplos de sistemas de gestión de bases de datos no relacionales (SGBDNR) son MongoDB, Firebase, Neo4j, Cassandra, etc.

- Bases de datos distribuidas: son aquellas que almacenan los datos en varios nodos o servidores conectados entre sí, en lugar de en un único servidor centralizado. Permiten mejorar la disponibilidad, la tolerancia a fallos y el balanceo de carga de los datos. Algunos ejemplos de sistemas de gestión de bases de datos distribuidas (SGBDD) son CouchDB, DynamoDB, HBase, etc. Para elegir el tipo de base de datos más adecuado para una aplicación móvil, se deben tener en cuenta varios criterios, como la estructura, el tamaño, la seguridad, la flexibilidad, el soporte y la escalabilidad de los datos. También se debe considerar el tipo de aplicación, el público objetivo, el presupuesto y el tiempo disponible para el desarrollo.

25: Room es una biblioteca de Android que facilita el uso de bases de datos SQLite en las aplicaciones móviles. Room proporciona una capa de abstracción sobre SQLite que permite acceder y manipular los datos mediante objetos y anotaciones, sin tener que escribir consultas SQL complejas. Room también ofrece varias ventajas, como la verificación de consultas en tiempo de compilación, la migración automática de esquemas, el soporte para LiveData y RxJava, y la integración con otras bibliotecas de Android como WorkManager y Paging

26: Firebase es una plataforma de Google que ofrece diversos servicios y herramientas para el desarrollo de aplicaciones web y móviles. Firebase facilita el desarrollo de aplicaciones de calidad, rápidas y eficientes, al brindar soluciones para el backend, la base de datos, la autenticación, el almacenamiento, las notificaciones, las analíticas y la monetización. Firebase se integra con las principales plataformas móviles como Android, iOS y web, y se puede combinar y adaptar según las necesidades del proyecto.