Universidad Tecnológica de Huejotzingo

Actividad: Sumativa - Desarrollo de una aplicación wearable

Desarrollo para dispositivos inteligentes

Profesor: José Luna Hernández

Fernanda Alonso Apanco	3522110368
Maria Teresa Merales Valencia	3522110951
Irwin Miguel Arce Terrazas	3522111009
Jan Carlo Trujano Leon	3521110542

9° A

Universidad Tecnológica de HUEJOTZINGO

Contenido

1. Introducción....3
2. Configuración de Herramientas de Desarrollo....3
3. Configuración del Emulador Wear OS....4
4. Funcionalidad de la Aplicación....7
6. Principios de Diseño para Wearables Aplicados....9
8. Conclusión....10
9. Fuentes de Consulta....10

Universidad Tecnológica de HUEJOTZINGO

1. Introducción

El proyecto **DiabeTrackWear** es una aplicación diseñada para dispositivos con sistema operativo **Wear OS**, enfocada en mejorar la calidad de vida de personas que viven con diabetes. Esta aplicación permite a los usuarios registrar información vital como niveles de glucosa, hábitos alimenticios y otros indicadores importantes de salud, todo desde la comodidad de su smartwatch.

El desarrollo de DiabeTrackWear responde a una necesidad real en el contexto actual, donde la tecnología debe adaptarse a las exigencias de monitoreo constante y portabilidad. Además, se exploraron los principios del diseño de interfaces específicas para dispositivos con pantallas pequeñas y limitaciones de interacción táctil

2. Configuración de Herramientas de Desarrollo

Para el desarrollo de DiabeTrackWear se utilizó el entorno de desarrollo **Android Studio (Electric Eel | 2022.1.1 o superior)**, configurado con los siguientes elementos:

- **Lenguaje de Programación:** Kotlin
- **Diseño de Interfaz:** Jetpack Compose para Wear OS
- **Sistema de Construcción:** Gradle (versión 8.0+)
- **Dependencias clave:**
  - androidx.wear.compose para interfaces específicas
  - androidx.navigation para navegación entre pantallas
  - com.google.android.horologist para interacción con sensores y conectividad
- **SDK de Wear OS:** 30 y 33

Se crearon los módulos necesarios para mantener una arquitectura limpia y organizada, con separación entre vistas, lógica de negocio y comunicación entre dispositivos.

Dependencias necesarias para el proyecto

```
dependencies {
    implementation("androidx.compose.ui:ui-tooling-preview")
    implementation("androidx.compose.animation:animation")
    implementation("androidx.activity:activity-compose:1.9.0")
    implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.7.0")
    implementation("androidx.navigation:navigation-compose:2.6.0")
    implementation("androidx.compose.material3:material3:1.1.0")
    implementation("androidx.core:core-splashscreen:1.0.1")
    implementation("com.google.android.material:material:1.11.0")
    implementation("androidx.core:core-ktx:1.12.0")
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-play-services:1.7.3")

    // Compose para Wear OS
    implementation("androidx.wear.compose:compose-material:1.3.0")
    implementation("androidx.wear.compose:compose-foundation:1.3.0")
    implementation("androidx.wear.compose:compose-navigation:1.3.0")

    // Google Play Services para wearable
    implementation("com.google.android.gms:play-services-wearable:18.1.0")

    // Debug y test
    debugImplementation("androidx.compose.ui:ui-tooling")
    debugImplementation("androidx.compose.ui:ui-test-manifest")
    debugImplementation("androidx.wear:wear-tooling-preview:1.0.0")
    androidTestImplementation("androidx.compose.ui:ui-test-junit4")
}
```

3. Configuración del Emulador Wear OS

Para probar y validar la aplicación se configuró un emulador con las siguientes especificaciones:

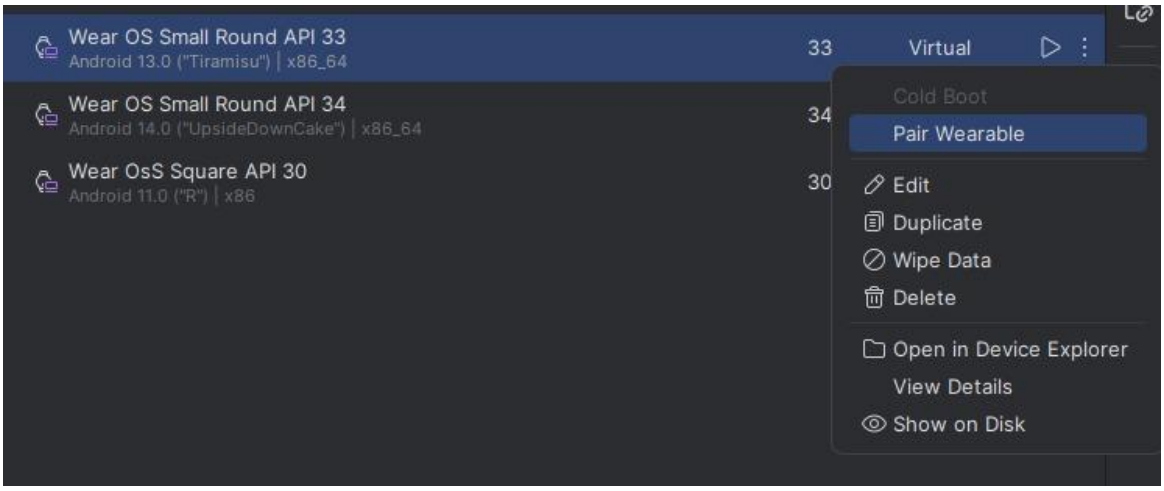
- **Nombre del AVD:** Wear_OS_Small_Round
- **Versión del sistema operativo:** Wear OS API 30 o 33
- **Tipo de dispositivo:** Small Round
- **Resolución:** 320x320 px
- **RAM:** 1.5 GB

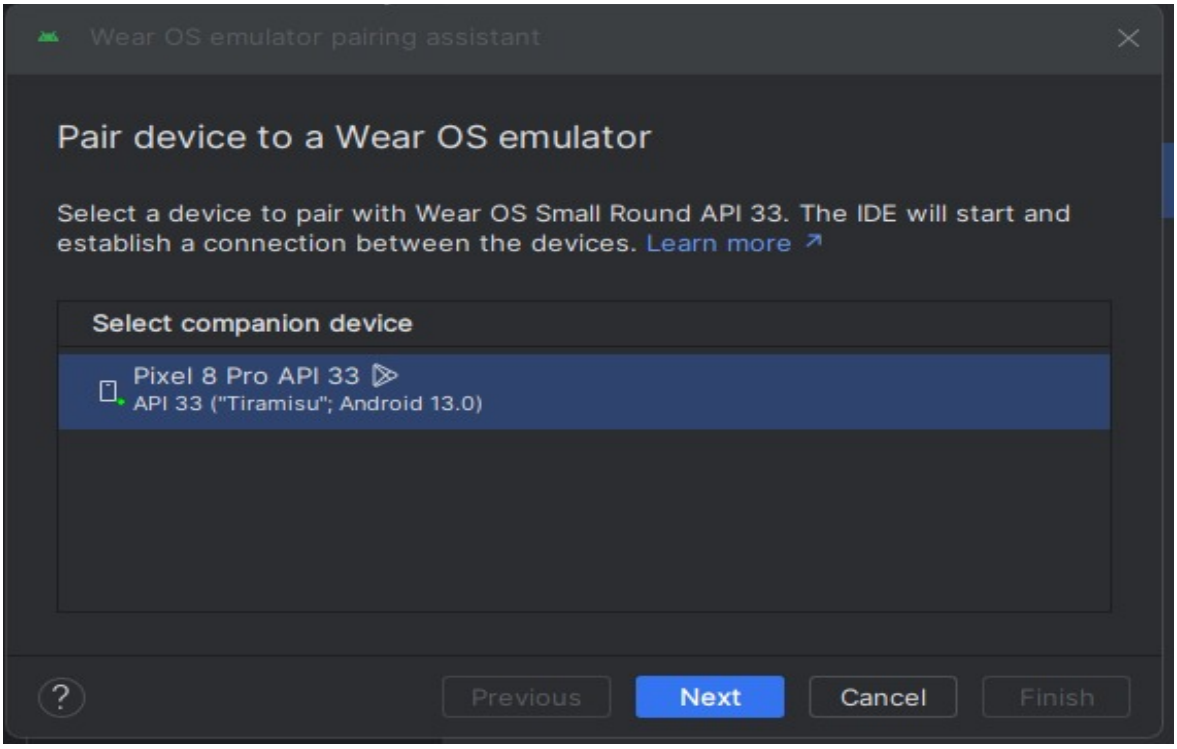
Pasos para la configuración:

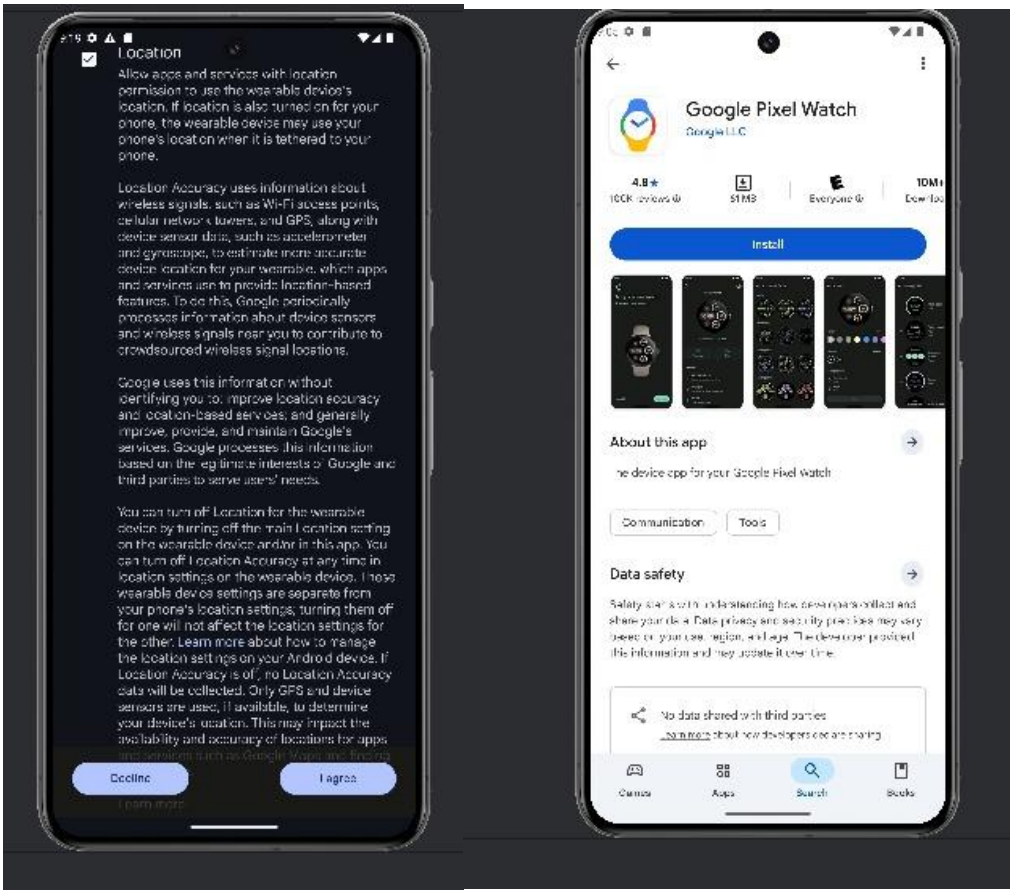
1. Ir a AVD Manager en Android Studio
2. Seleccionar "+ Create Virtual Device"
3. Elegir la categoría "Wear OS" > "Small Round"
4. Descargar imagen del sistema Wear OS (API 30 o 33)

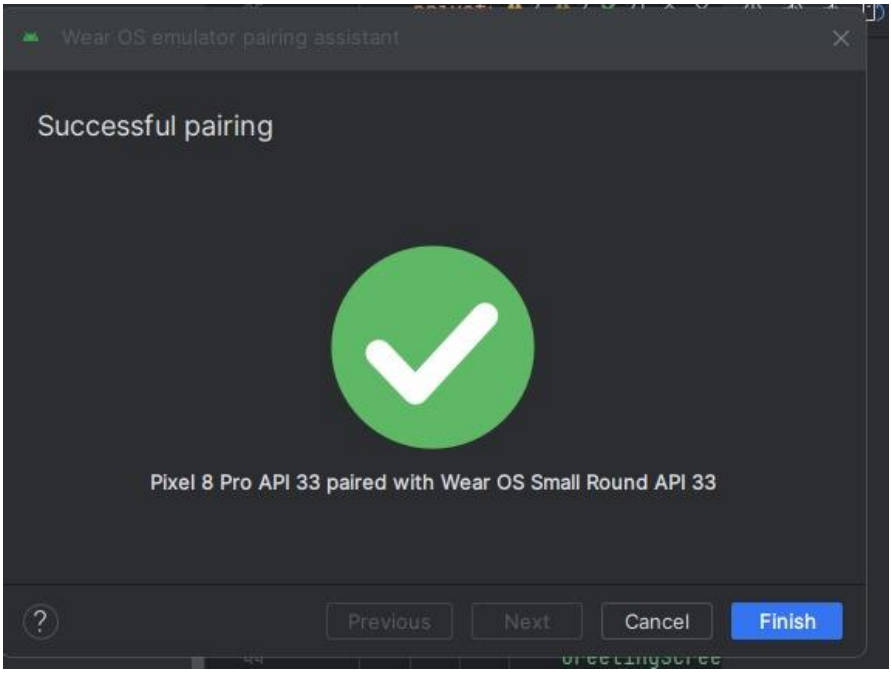
5. Asignar memoria RAM y almacenamiento interno (recomendado: 1.5 GB RAM, 2 GB almacenamiento)
6. Ejecutar el dispositivo para verificar funcionalidad

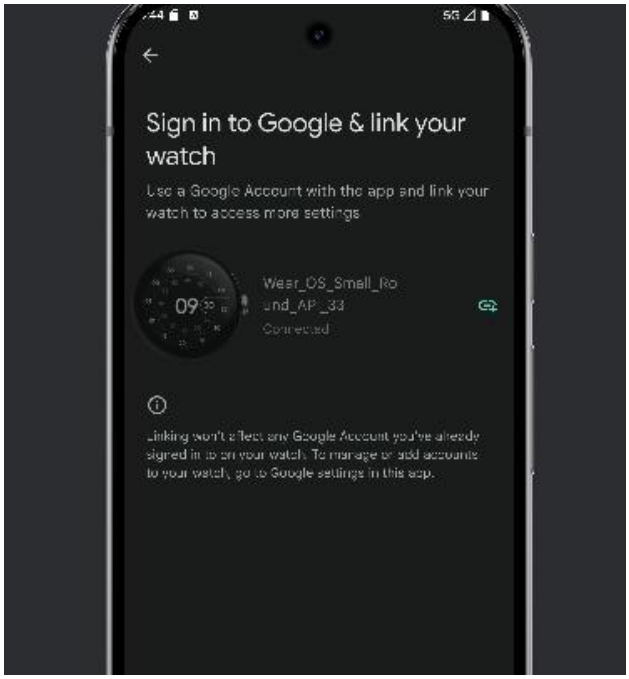
Se aseguró que el emulador soportara la conectividad para pruebas de mensajería entre dispositivos.






Location
Allow apps and services with location permission to use the wearable device's location. If location is also turned on for your phone, the wearable device may use your phone's location when it is tethered to your phone.
Location Accuracy uses information about wireless signals, such as Wi-Fi access points, cellular network towers, and GPS, along with device sensor data, such as accelerometer and gyroscope, to estimate more accurate device location for your wearable, which apps and services use to provide location-based features. To do this, Google periodically processes information about device sensors and wireless signals near you to contribute to crowdsourced wireless signal locations.
Google uses this information without identifying you to: improve location accuracy and location-based services; and generally improve, provide, and maintain Google's services. Google processes this information based on the legitimate interests of Google and third parties to serve users' needs.
You can turn off Location for the wearable device by turning off the main Location setting on the wearable device and/or in this app. You can turn off Location Accuracy at any time in location settings on the wearable device. These wearable device settings are separate from your phone's location settings; turning them off for one will not affect the location settings for the other. Learn more about how to manage the location settings on your Android device. If Location Accuracy is off, no Location Accuracy data will be collected. Only GPS and device sensors are used, if available, to determine your device's location. This may impact the availability and accuracy of locations for apps and services such as Google Maps and finding
Decline
I agree
Learn more
Google Pixel Watch
Google LLC
4.8★
51 MB
Everyone
Install
About this app
Communication
Tools
Data safety
No data shared with third parties
Learn more about how developers declare sharing
Games
Apps
Search
Books


Wear OS emulator pairing assistant
Successful pairing
Pixel 8 Pro API 33 paired with Wear OS Small Round API 33
Previous
Next
Cancel
Finish

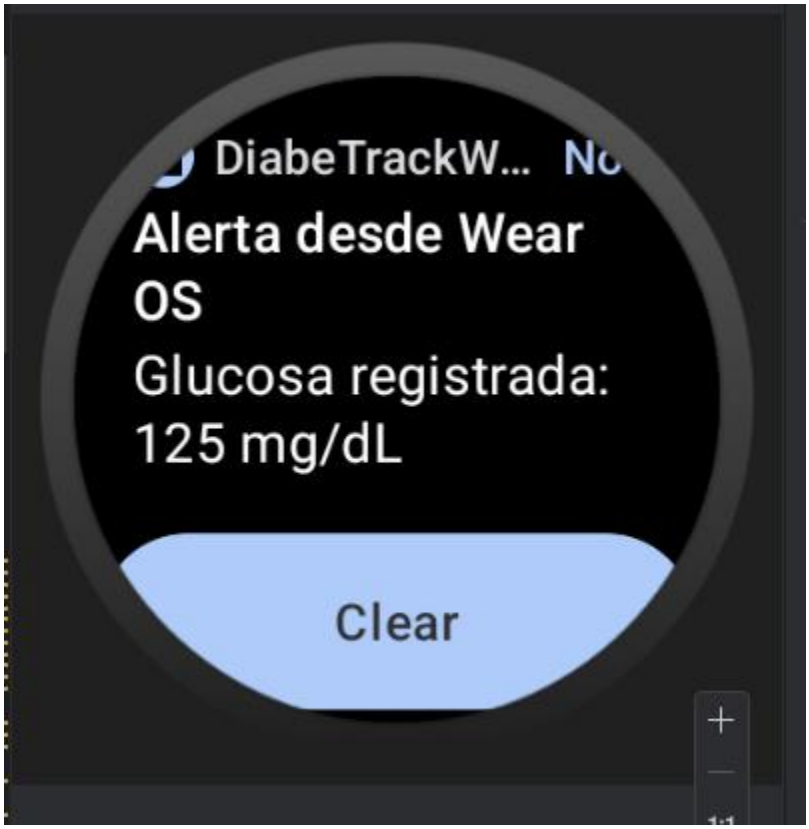


4. Funcionalidad de la Aplicación

La aplicación ofrece una experiencia intuitiva y centrada en el usuario. Sus funcionalidades principales incluyen:

- **Pantalla de bienvenida/inicio**: Presenta el nombre de la aplicación y opciones de navegación.
- **Menú principal**: Acceso a las diferentes secciones: monitoreo de glucosa, dieta, ejercicio, historial.
- **Pantalla de registro de glucosa**: Permite ingresar los niveles diarios.
- **Pantalla de registro de actividades o comidas**: Guarda hábitos que puedan afectar el control de la diabetes.
- **Historial**: Consulta de registros anteriores mediante listas o tarjetas.
- **Navegación fluida** entre pantallas utilizando NavController y composable.

Cada una de estas funcionalidades fue probada en el emulador y ajustada para una experiencia responsiva.



5. Comunicación entre Dispositivos Vinculados

Una de las características clave del proyecto fue implementar la capacidad de **enviar y recibir mensajes entre el reloj inteligente y un dispositivo Android vinculado**. Esto se logró mediante la integración de:

- **MessageClient de la API de Google Play Services**
- **CapabilityClient para detectar nodos disponibles**
- **WearableListenerService para recibir mensajes del otro dispositivo**

Funcionamiento:

1. El reloj detecta la conexión con el teléfono vinculado.
2. Cuando el usuario envía un registro desde el reloj, se empaqueta y se transmite usando la API de mensajería.
3. El teléfono responde o guarda la información en su base de datos.

Este sistema asegura sincronización continua y eficiencia en el traspaso de datos, clave para un seguimiento preciso en tiempo real.

6. Principios de Diseño para Wearables Aplicados

En el diseño de la aplicación se aplicaron los principios fundamentales de diseño para dispositivos Wear OS, enfocados en la simplicidad, legibilidad y accesibilidad:

- **Pantallas cortas y concisas**: Cada pantalla presenta la mínima cantidad de información necesaria.
- **Tipografía legible**: Se utilizó texto grande y botones accesibles para facilitar la interacción en una pantalla reducida.
- **Navegación circular**: Se aprovecha la navegación en carrusel o circular para adaptarse a la forma del dispositivo.
- **Modo oscuro y consumo eficiente de batería**: La interfaz predominante es oscura con texto claro para minimizar el uso de energía.
- **Interacción táctil optimizada**: Botones grandes, bien separados, con feedback visual inmediato.

7. Código Fuente y Estructura del Proyecto

El proyecto está estructurado en módulos bien definidos:

- **/screens**: Contiene todas las pantallas de la aplicación como HomeScreen, GlucoseScreen, HistoryScreen, etc.
- **/navigation**: Implementa el NavHostController para mover entre pantallas.
- **/services**: Implementa la lógica de conexión y mensajes entre dispositivos.

- **/theme**: Define colores, tipografía y estilos visuales.

Se hace uso de buenas prácticas como:

- Separación de responsabilidades (MVVM simplificado)
- Código modular y reutilizable
- Comentarios descriptivos
- Uso adecuado de remember y mutableStateOf para mantener estados

8. Conclusión

El desarrollo de DiabeTrackWear fue un ejercicio completo que integró habilidades de programación en Kotlin, diseño de interfaces para wearables y comunicación entre dispositivos. La aplicación cumple con todos los requerimientos establecidos y demuestra ser funcional, útil y bien estructurada.

Este proyecto no solo resuelve una necesidad real de monitoreo para personas con diabetes, sino que también demuestra un uso eficiente y moderno de las tecnologías de desarrollo móvil.

9. Fuentes de Consulta

- Documentación oficial de Android Developers: https://developer.android.com
- Guía de Jetpack Compose para Wear OS: https://developer.android.com/jetpack/compose/wear
- Google Codelabs para Wearables
- Stack Overflow (https://stackoverflow.com)
- GitHub para referencia de patrones de arquitectura
- Documentación oficial de Google Play Services para Wearables