# Preguntas Importantes Antes de Pedir Código

• ¿Qué quiero que haga mi aplicación? Registrar y monitorear los valores de presión arterial de los usuarios, mostrando reportes gráficos y alertas en caso de valores fuera del rango normal.

• ¿Cuál es la funcionalidad clave que necesito? Control del historial de presión arterial (sistólica, diastólica y pulso) y generación de reportes diarios, semanales y mensuales.

• ¿Qué sí debe resolver y qué no? Debe resolver el registro, almacenamiento y consulta de los datos de presión. No debe sustituir a un diagnóstico médico ni interpretar los datos clínicamente, solo mostrar registros y tendencias.

• ¿Quién usará esta funcionalidad y en qué escenario real? Personas que desean controlar su presión arterial desde el hogar, y médicos que podrían consultar el historial del paciente durante revisiones.

• ¿Qué datos necesito recibir y qué resultados espero devolver o mostrar? - Entradas: valores de presión sistólica, diastólica, pulso, fecha y hora. - Salidas: reporte en tabla o gráfica, alertas en caso de valores fuera de lo normal, y un historial navegable.

• ¿Debe hacerse en Java (Android Studio)? ¿Requiere conexión a internet, base de datos, API externa? Sí, puede hacerse en Android Studio con Java o Kotlin. No requiere conexión a internet obligatoria: puede funcionar con base de datos local (SQLite o Room). Opcional: conexión a internet para sincronizar los registros con la nube o exportar a PDF/Excel.

• ¿Cómo debería mostrarse en pantalla (UI/UX)? Pantalla principal con el último registro y botones para: - Registrar nueva medida - Ver historial - Ver reportes en gráfico - Exportar o compartir reporte

• ¿Qué validaciones y manejo de errores necesito? - Validar que los valores ingresados estén en un rango realista (ejemplo: sistólica entre 70 y 200 mmHg). - Mostrar mensajes de error si se dejan campos vacíos o se introducen valores inválidos.

• ¿Será algo pequeño o crecerá a futuro? Inicialmente será una app personal para registrar y visualizar datos. A futuro podría crecer con funciones como sincronización con dispositivos Bluetooth (tensiómetros digitales), exportación a la nube o integración con sistemas médicos.

# # PROMPT ESTRUCTURADO PARA GITHUB COPILOT AGENT

## 1. CONTEXTO Y OBJETIVO **Contexto:** Una aplicación de salud enfocada en el control de la presión arterial. **Objetivo:** Registrar y consultar valores de presión arterial y generar reportes visuales para el usuario.

## 2. REQUISITOS FUNCIONALES **Entradas:** Valores de presión sistólica, diastólica, pulso, fecha y hora. **Procesos:** Guardar los registros en una base de datos, calcular estadísticas y generar gráficas de tendencia. **Salidas:** Reporte en pantalla con historial y gráficas, mensajes de alerta en caso de valores anormales.

## 3. REQUISITOS TÉCNICOS **Plataforma:** Android (Android Studio). **Lenguaje:** Java o Kotlin. **APIs/Librerías:** Room/SQLite, MPAndroidChart (para gráficas). **Permisos:** Almacenamiento (si se exportan archivos), internet (si hay sincronización).

## 4. ESTRUCTURA DE LA SOLUCIÓN **Arquitectura:** MVVM. **Componentes principales:** - MainActivity (menú principal). - Activity/Fragment para ingreso de datos. - Activity/Fragment para historial y reportes. - DatabaseService (SQLite/Room). **Patrones de diseño:** Singleton (para acceso a la BD).

## 5. DETALLES DE IMPLEMENTACIÓN **Validaciones:** Rango válido de presión arterial. **Manejo de errores:** Mostrar mensajes si un campo está vacío o los valores son ilógicos. **Estado de la app:** Guardar último registro y mostrarlo al abrir. **Persistencia:** Base de datos local y opción de exportar.

## 6. CONSIDERACIONES ESPECÍFICAS **Compatibilidad:** Android 8.0 en adelante. **Rendimiento:** Optimización de consultas y gráficos. **Seguridad:** Protección de los datos médicos con encriptación básica/local. **Testing:** Pruebas con valores reales y simulados para garantizar el correcto manejo de datos.

NUEVO: Nueva opción para generar archivos PDF  con la información, análisis y recomendaciones sugeridas por Gemini para que el usuario las almacene en su dispositivo y pueda tenerlas guardadas por si las llegara a necesitar sin perderlas al cerrar la aplicación.