PROYECTO PRESIÓN ARTERIAL

Preguntas Importantes Antes de Pedir Código

- **¿Qué quiero que haga mi aplicación?** Que permita registrar y llevar un registro de la presión arterial del paciente, incluyendo valores sistólicos y diastólicos, fecha y hora de toma, brazo utilizado, medicación, enfermedades (actual, clínica, personales, familiares, psicosocial), y datos de identificación del paciente. Debe impedir tomar otra lectura antes de 15 minutos.
- **¿Cuál es la funcionalidad clave que necesito?** Registrar la presión arterial con todos sus datos asociados y almacenarlos en una base de datos local. Verificar el tiempo entre mediciones para evitar tomas prematuras.
- **¿Qué sí debe resolver y qué no?** Debe resolver el registro completo de la presión arterial con todos los campos especificados y la validación de tiempo mínimo entre tomas. No necesita conexión a internet ni API externas.
- **¿Quién usará esta funcionalidad y en qué escenario real**? Será usada por pacientes o profesionales de salud para monitorear la presión arterial en casa o en clínicas.
- **¿Qué datos necesito recibir y qué resultados espero devolver o mostrar? Recibe:** Datos de identificación del paciente, valores de presión arterial, brazo, medicación, enfermedades. **Muestra:** Listado de todas las mediciones registradas.
- **¿Debe hacerse en Java (Android Studio)?** Sí, en Java con Android Studio.

- **¿Requiere conexión a internet, base de datos, API externa?** Requiere base de datos local (Room) para almacenar los registros.
- **¿Cómo debería mostrarse en pantalla (UI/UX)? UI:** Formulario claro para ingresar datos, lista de mediciones anteriores, notificación si se intenta tomar presión antes de 15 minutos.
- **¿Qué validaciones y manejo de errores necesito?** Validar que no se tome presión antes de 15 minutos, que los campos obligatorios no estén vacíos, que los valores de presión sean numéricos válidos. Manejo de errores con mensajes como "Espere al menos 15 minutos" o "Campo requerido".
- **¿Será algo pequeño o crecerá a futuro?** De inicio será pequeño, pero puede crecer añadiendo gráficas, alertas, o exportación de datos.

Estructura de un Prompt Correcto

- **Contexto del proyecto:** Estoy desarrollando una app Android en Java, en Android Studio, para registrar y monitorear la presión arterial.
- **Objetivo claro:** Necesito una app que permita registrar la presión arterial con todos los datos solicitados, verifique el tiempo entre mediciones y muestre un historial de lecturas.
- **Detalles técnicos:** Debe usar Room para la base de datos local, validaciones de tiempo y campos, y mostrar una lista de mediciones.
- **Entradas y salidas:** El usuario ingresa los datos de la presión arterial y se guardan en la base de datos; se muestra un listado de todas las mediciones.

- **Formato esperado:** Dame la clase PresionArterial, el DAO, el adaptador para la lista, y el código para verificar el tiempo mínimo entre mediciones.
- **Extras opcionales:** Manejo de errores claros, notificaciones, buenas prácticas de código.
-

PROMPT ESTRUCTURADO PARA GITHUB COPILOT AGENT

1. CONTEXTO Y OBJETIVO

- Contexto: Estoy desarrollando una aplicación Android en Java, usando Android Studio. La app verifica la nacionalidad de una persona mediante el escaneo de su huella dactilar, excluyendo a personas de Guatemala y Estados Unidos. La app rellena automáticamente un formulario con datos personales mientras cronometra el tiempo del proceso.
- Objetivo: Registrar el tiempo que toma el análisis de la huella dactilar para detectar la nacionalidad del usuario, excluyendo ciertas nacionalidades, y mostrar este tiempo junto con los datos obtenidos.

2. REQUISITOS FUNCIONALES

- Entradas: Interacción del usuario para iniciar el escaneo de huella, autenticación biométrica del sistema Android.
- Procesos: Solicitar autenticación biométrica, simular la obtención de datos del usuario basado en la huella (excluyendo Guatemaltecos y Estadounidenses), iniciar y detener un cronómetro durante el proceso, validar los datos del formulario simulado.

- Salidas: Mostrar un formulario con los datos personales simulados del usuario, su nacionalidad (si es permitida), el tiempo total transcurrido del proceso de autenticación y verificación, y mensajes de estado o error.

3. REQUISITOS TÉCNICAS

- Plataforma: Android Studio (versión utilizada en desarrollo: compatible con API 24-36, usando Java).
- Lenguaje: Java
- APIs/Librerías: androidx.biometric:biometric (API 24+), SystemClock/Handler para cronómetro. (Nota: La conexión a base de datos y API externa real no se implementó en la versión desarrollada).
- Permisos: USE_BIOMETRIC, USE_FINGERPRINT (ya incluidos en AndroidManifest.xml).

4. ESTRUCTURA DE LA SOLUCIÓN

- Arquitectura: (Basado en la implementación actual) Estructura procedural centrada en MainActivity para manejar la UI, lógica de biometría, cronómetro y simulación de datos. (La estructura MVVM fue solicitada pero no implementada en la versión básica).
- Componentes principales: MainActivity (lógica principal), activity_main.xml (UI), build.gradle (dependencias).
- Patrones de diseño: (No explícitamente implementados en la versión básica desarrollada).

5. DETALLES DE IMPLEMENTACIÓN

- Validaciones:

    - Validar que los campos del formulario simulado no estén vacíos después de la "autenticación".
    - Validar que la edad sea un número dentro de un rango válido.
    - Validar que la nacionalidad simulada no sea de Guatemala o EE.UU.
    - Validar el estado del escáner biométrico (huella reconocida, no reconocida, error).
- Manejo de errores:
    - Mostrar mensajes claros si falla el lector de huella (p. ej., "Huella no reconocida", "Error de biometría").
    - Manejar el caso donde el dispositivo no tenga huellas registradas.
    - (Solicitado pero no implementado: manejo de excepciones en base de datos SQLite).
- Estado de la app: Estados identificables: Esperando acción, Escaneando huella (cronómetro en marcha), Autenticación exitosa/ fallida (cronómetro detenido), Mostrando resultados.
- Persistencia: (Solicitado pero no implementado: Guardar registros en SQLite y posibilidad de exportar).

6. CONSIDERACIONES ESPECÍFICAS

- Compatibilidad: Compatible con Android 7.0 (API 24) en adelante.
- Rendimiento: El proceso de autenticación biométrica depende del sistema Android; el cronómetro muestra el tiempo transcurrido con precisión.

- Seguridad: Uso de la API biométrica del sistema para la autenticación. (Solicitado pero no implementado: encriptación de datos locales, bloqueo de pantalla).
- Testing: (Solicitado pero no implementado: Pruebas unitarias, funcionales, de compatibilidad y seguridad). La app fue probada manualmente en un dispositivo físico con huella registrada.