Universidad Mariano Gálvez Guatemala, Campus Jutiapa Facultad de Ingeniería en Sistemas

Curso: Programación 2

Catedrático: Ing. Ruldin Ayala

Nombres del estudiante:

Lester David Payes Méndez 0905-24-22750

Nota: Como las actividad del turismo en clase era en dúo pues se siguió mejorando cada promt en conjunto con Mario David Tereta Sapalun 0905-15-297

.

App BIOpass o Proyecto 5BIOSpass

Contexto
Desarrolla una aplicación Android en **Java** usando **Android Studio**, con soporte mínimo Android 8.0 (API 26), que permita:

- Registrar turistas mediante **escaneo de huellas digitales** o registro **manual**.
- Guardar la **nacionalidad**, **estado del registro** (Éxito, Fallo, Manual), **motivo del registro manual** y **tiempo de escaneo** en **Room/SQLite**.
- Mostrar **historial de registros** y calcular métricas (total, exitosos, fallidos, manuales, promedio de tiempo, % de éxito) filtradas por nacionalidad.
- Todo en **modo oscuro** (noche) con diseño minimalista.

Requisitos Funcionales

Entradas:

- Huella digital (sensor del teléfono Android)
- Nacionalidad del turista
- Motivo del registro manual (si aplica)
- Estado del registro: Éxito, Fallo, Manual
- Tiempo de escaneo en milisegundos

Procesos:

1. Inicializar **contador de tiempo** al iniciar el escaneo.
2. Escaneo biométrico de huella digital usando **Android Biometric API**.
3. Validación de huella:
   - Si falla o timeout, permitir registro manual solicitando nacionalidad y motivo.
4. Guardar registro en **base de datos Room/SQLite**.
5. Mostrar animación de huella llenándose en tiempo real mientras se escanea.
6. Calcular métricas:
   - Total de registros
   - Exitosos
   - Fallidos
   - Manuales
   - Promedio de tiempo
   - % de éxito

7. Permitir filtrado de registros y métricas por nacionalidad.

Salidas:

- Confirmación de registro (éxito, fallo o manual)
- Mensajes de error claros
- Pantalla con historial de registros y métricas

- Actualización de métricas en tiempo real

Requisitos Técnicos

Lenguaje y plataforma:

- Java en Android Studio
- Arquitectura **MVVM** (MainActivity + Fragments + Repository + ViewModel)
- Base de datos local: **Room/SQLite**

APIs y librerías:

- Android Biometric API
- Room/SQLite
- RecyclerView, LiveData, ViewModel

Permisos en Manifest:

- `USE_BIOMETRIC`
- `INTERNET`
- `READ_EXTERNAL_STORAGE` (si se usa lector externo)

Recursos y UI:

- **Colores**: `background_dark`, `primary`, `secondary`, `error`
- **Drawables**: `progress_bar_dark`, íconos de botones y fondos
- **Layouts**:
  - MainActivity
  - FingerprintFragment
  - ManualFragment
  - Item de RecyclerView para registros
- **Strings**: todos los textos de botones, labels y mensajes
- **Estilos**:
  - Tema oscuro obligatorio
  - Indicadores de estado: Idle, Scanning, Success, Error
  - Botones grandes y legibles
  - Animación mínima pero clara de huella llenándose

Manifest completo con actividades y permisos declarados.
- **Nombre de la app en la pantalla**: colocar "BIOpass" en la **parte inferior derecha** de cada pantalla principal

---

Detalles de Implementación

Validaciones:

- Nacionalidad válida (no vacía)
- Huella obligatoria salvo nacionales excluidos o exentos (Guatemala y EE. UU.)
- Verificación de identificación si se registra manualmente

Estados de la app:

- **Idle**: esperando registro
- **Scanning**: escaneo en proceso
- **Success**: registro exitoso
- **Error**: fallo en el proceso

Persistencia y métricas:

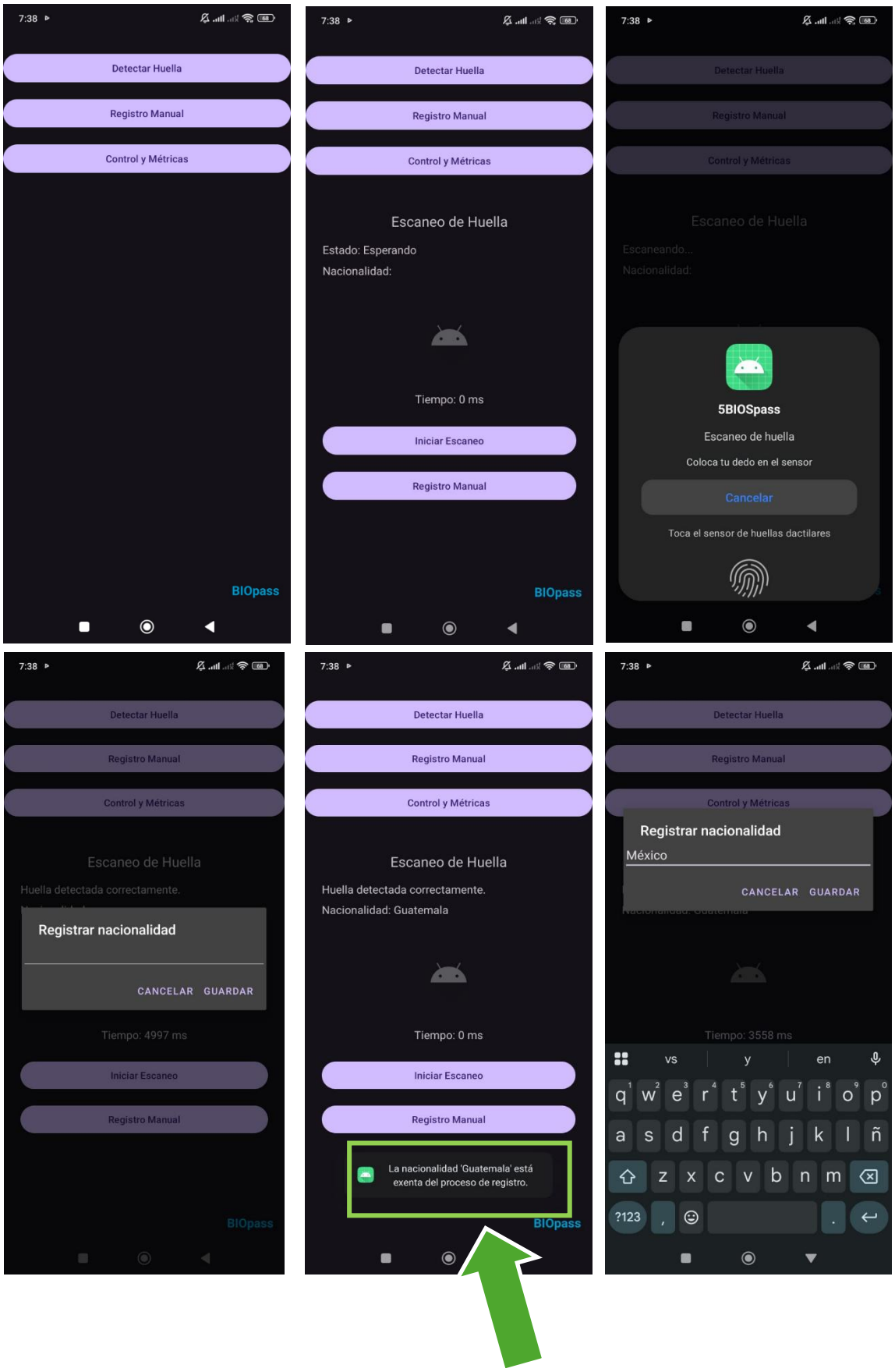
- Cada registro guarda: nacionalidad, estado, motivo manual (si aplica), tiempo de escaneo
- Tablas organizadas por nacionalidad
- Métricas calculadas y actualizadas en tiempo real
- Filtro de registros por nacionalidad
- Promedio de tiempo acumulado: sumar todos los tiempos de registros exitosos antes de salir del escaneo

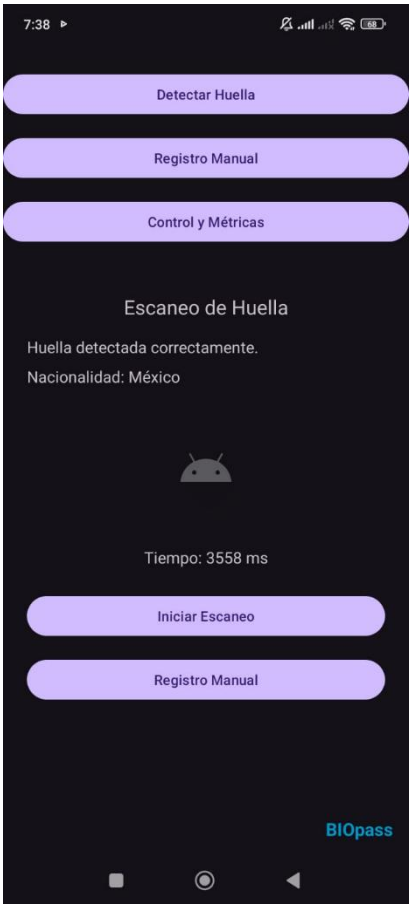
Control de errores:

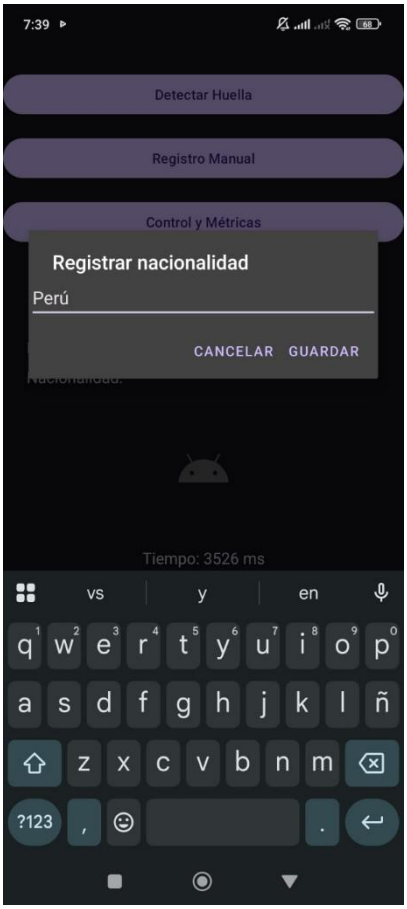
- Timeout en escaneo de huella
- Reintentos automáticos hasta éxito o registro manual
- Mensajes claros de error y confirmación

Instrucciones al agente IA

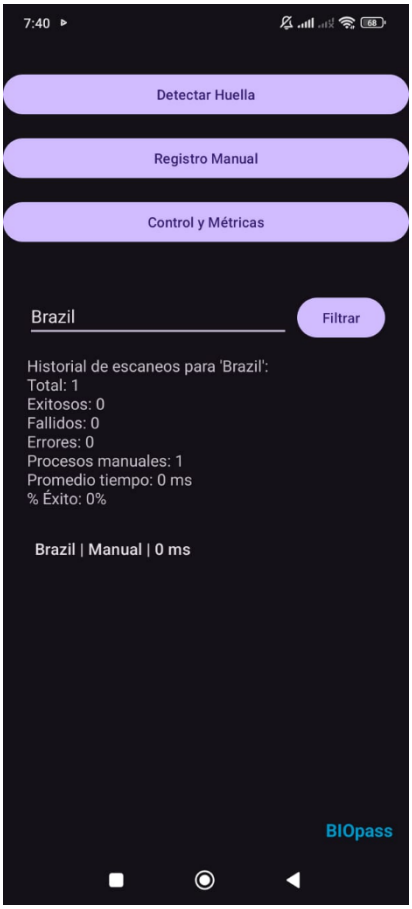
1. Genera **todos los archivos necesarios** para que el proyecto compile sin errores de AAPT:
   - `colors.xml`, `drawable/progress_bar_dark.xml`, `styles.xml`, `themes.xml`, `strings.xml`
   - Layouts de todas las pantallas y elementos
   - Manifest completo con permisos y actividades
2. Genera **código Java completo** para MainActivity, Fragments, Repository, ViewModel y DAO de Room
3. Configura **RecyclerView** para mostrar historial de registros
4. Implementa animación de huella y cálculo de tiempo
5. Cumple la arquitectura MVVM
6. Todo en **modo oscuro** obligatorio
7. Asegúrate de que **todos los nombres de recursos coincidan** en todos los archivos.

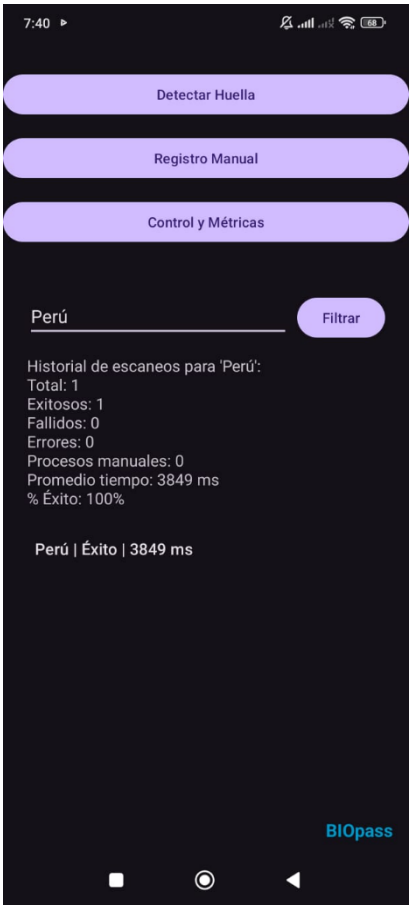

7:38
Detectar Huella
Registro Manual
Control y Métricas
BIOpass
Escaneo de Huella
Estado: Esperando
Nacionalidad:
Tiempo: 0 ms
Iniciar Escaneo
Registro Manual
Escaneando...
5BIOSpass
Escaneo de huella
Coloca tu dedo en el sensor
Cancelar
Toca el sensor de huellas dactilares
Huella detectada correctamente.
Registrar nacionalidad
CANCELAR GUARDAR
Tiempo: 4997 ms
Nacionalidad: Guatemala
Tiempo: 0 ms
La nacionalidad 'Guatemala' está exenta del proceso de registro.
México
Tiempo: 3558 ms

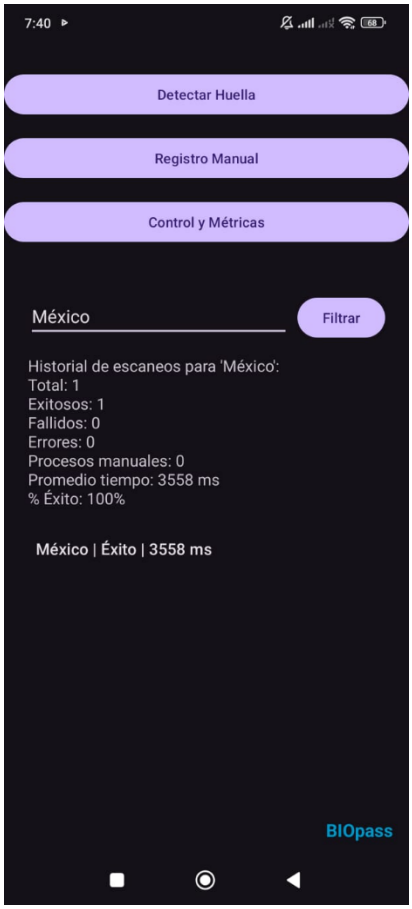

7:38
Detectar Huella
Registro Manual
Control y Métricas
Escaneo de Huella
Huella detectada correctamente.
Nacionalidad: México
Tiempo: 3558 ms
Iniciar Escaneo
Registro Manual
BIOpass


7:39
Detectar Huella
Registro Manual
Control y Métricas
Registrar nacionalidad
Perú
CANCELAR
GUARDAR
Tiempo: 3526 ms


7:40
Detectar Huella
Registro Manual
Control y Métricas
Registro Manual
Brazil
A persona de la tercera edad no le detecta su huella
Registrar
Registro manual exitoso. Nacionalidad: Brazil. Motivo: A persona de la tercera edad no le detecta su huella
BIOpass


7:40
Detectar Huella
Registro Manual
Control y Métricas
Brazil
Filtrar
Historial de escaneos para 'Brazil':
Total: 1
Exitosos: 0
Fallidos: 0
Errores: 0
Procesos manuales: 1
Promedio tiempo: 0 ms
% Éxito: 0%
Brazil | Manual | 0 ms
BIOpass


7:40
Detectar Huella
Registro Manual
Control y Métricas
Perú
Filtrar
Historial de escaneos para 'Perú':
Total: 1
Exitosos: 1
Fallidos: 0
Errores: 0
Procesos manuales: 0
Promedio tiempo: 3849 ms
% Éxito: 100%
Perú | Éxito | 3849 ms
BIOpass


7:40
Detectar Huella
Registro Manual
Control y Métricas
México
Filtrar
Historial de escaneos para 'México':
Total: 1
Exitosos: 1
Fallidos: 0
Errores: 0
Procesos manuales: 0
Promedio tiempo: 3558 ms
% Éxito: 100%
México | Éxito | 3558 ms
BIOpass


7:40
Detectar Huella
Registro Manual
Control y Métricas
Guatemala
Filtrar
El país 'Guatemala' está exento del proceso de control.
BIOpass

Lista de Compras

Prompt para Generar App Android "Lista de Compras" en Java

Estoy desarrollando una aplicación Android en Java llamada **"Lista de Compras"**, completamente offline, que use **SQLite** como base de datos local.

Objetivo de la App
La app debe permitir registrar productos con:

- Nombre (`String`)
- Estado (`PENDIENTE` / `COMPRADO`)
- Fecha (`yyyy-MM-dd`)
- **Cantidad (`int quantity`)**

Funcionalidades principales:

- Agregar, editar y eliminar productos
- Eliminar también desde la pantalla de edición (botón `Eliminar` visible solo en modo edición)
- Filtrar por estado, rango de fechas, nombre, categoría y rango de cantidad
- Generar un reporte con conteo total y por estado según los filtros
- Swipe-to-delete con `Snackbar` para deshacer
- Mensajes `Snackbar` / `Toast` en acciones (agregar, actualizar, eliminar, deshacer)

Requisitos Técnicos

- Lenguaje: Java (Android Studio)
- Base de datos: SQLiteOpenHelper
- MinSdkVersion: 21 (Android 5.0)
- Librerías: RecyclerView, MaterialComponents, ConstraintLayout
- Todas las operaciones de BD deben ejecutarse en background usando **ExecutorService** y actualizar la UI en el hilo principal
- La aplicación debe usar un **estilo de modo oscuro/noche** definido en `styles.xml` y `styles-night.xml` aplicado a toda la app
- Ajustes en tema:
  - `<item name="windowActionBar">false</item>`
  - `<item name="windowNoTitle">true</item>`
  - Uso de `Theme.MaterialComponents.DayNight` (sin `.DarkActionBar`) como tema base
- El proyecto usará **Gradle con Kotlin DSL (`build.gradle.kts`)**, por lo que las propiedades deben declararse con la sintaxis correcta (ejemplo: `isMinifyEnabled = true`)

Interfaz / UI/UX

- MainActivity con RecyclerView mostrando la lista de productos
- Toolbar incluida y configurada en las activities principales
- FloatingActionButton para agregar productos
- AddEditActivity con Toolbar, botón de navegación (flecha de regreso) y opción de **Eliminar producto**
- Filtros accesibles mediante un **FilterDialogFragment** o **FilterActivity**, no abriendo AddEditActivity
- Aplicación de filtros en MainActivity con métodos tipo `loadProductsFiltered(...)`
- ReportActivity mostrando conteo total y por estado según filtros
- SettingsActivity con toggle de tema
- Snackbar de confirmación al recibir resultados desde AddEditActivity o filtros
- Validaciones: nombre no vacío, fecha no nula, cantidad >=0, estado válido
- Manejo de errores de BD con try/catch y mostrar en Toast/Snackbar

---

Implementación de Filtros y Extras

- Por defecto, si un criterio está vacío, se ignora en la consulta
- Validar que `minQty <= maxQty` antes de aplicar filtros
- Mostrar `Snackbar` si no hay resultados
- Mantener operaciones de BD en background (`ExecutorService`) y actualizar UI en hilo principal
- Opcional: guardar últimos filtros en `SharedPreferences` y mostrar filtro activo en Toolbar

---

Entregables Requeridos

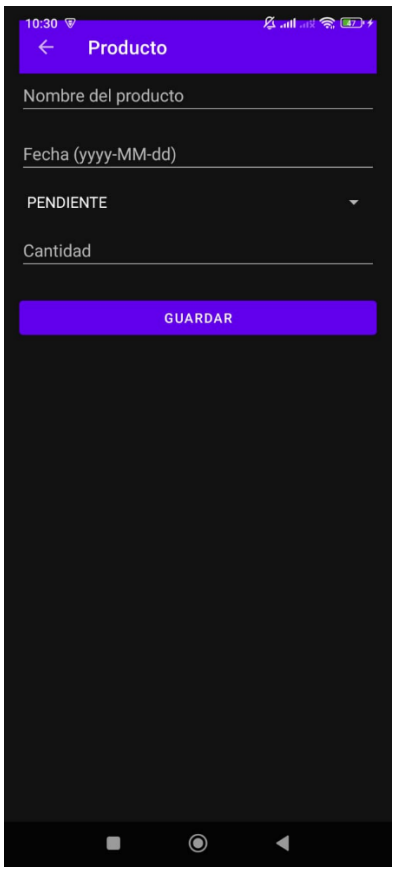
1. Clases Java completas:
   - Product (modelo) con campo `quantity`
   - DBHelper (SQLiteOpenHelper) con métodos CRUD adaptados para `quantity` y `getProductsFiltered(...)`
   - Adapter (RecyclerView) con soporte para swipe-to-delete y restauración de ítems
   - MainActivity
   - AddEditActivity (con opción eliminar)
   - ReportActivity
   - SettingsActivity
   - FilterDialogFragment.java o FilterActivity.java (para filtrar productos)

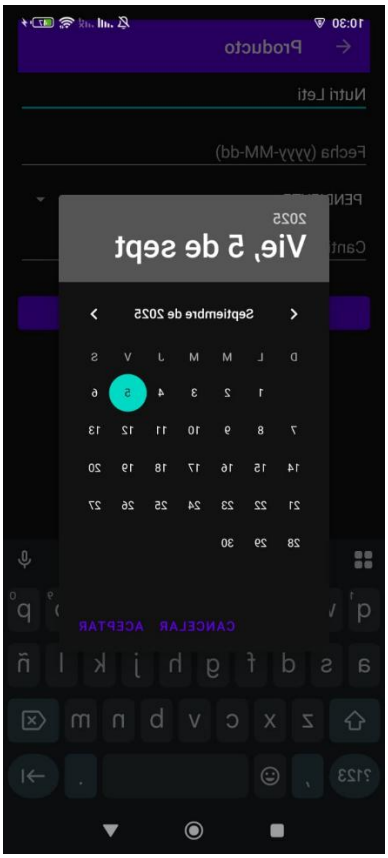
2. Todos los layouts XML necesarios (lista, ítem, formularios, reporte, ajustes) con IDs consistentes (ejemplo: toolbar en layouts principales)
3. Recursos en `res/values/` y `res/values-night/`:
   - `themes.xml` y `themes-night.xml` ajustados
   - `colors.xml` para esquema oscuro

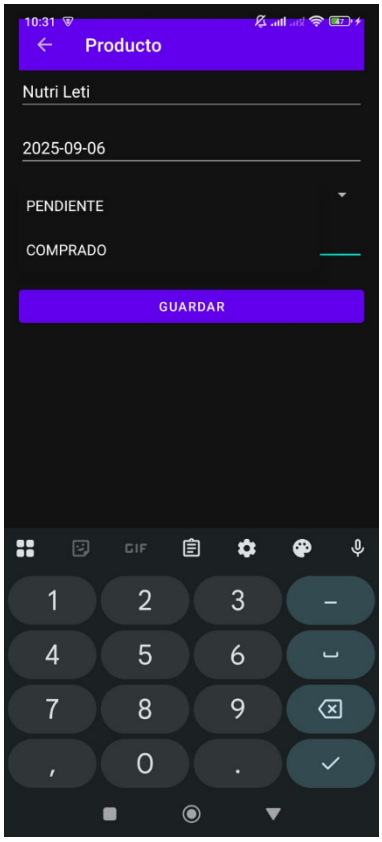
- `main_menu.xml` y demás menús necesarios
4. AndroidManifest.xml configurado y validado para compatibilidad con el nuevo tema y actividades
5. Contenido de `build.gradle.kts` (módulo) con todas las dependencias necesarias y configurado correctamente para Kotlin DSL (`isMinifyEnabled`)

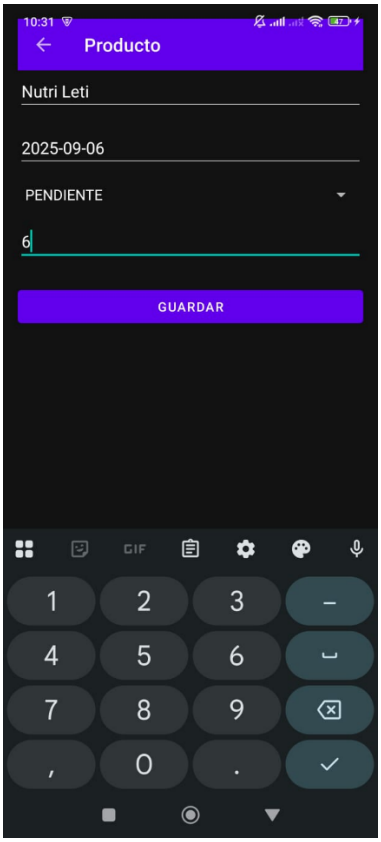
Formato de Respuesta Esperado

- Generar **cada archivo completo** (.java y .xml)
- Incluir **comentarios explicativos** y buenas prácticas de Android
- No omitir ningún detalle de la funcionalidad

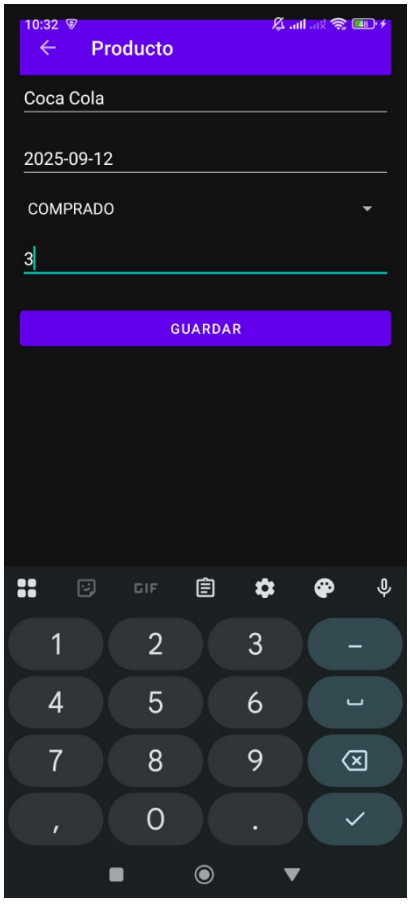


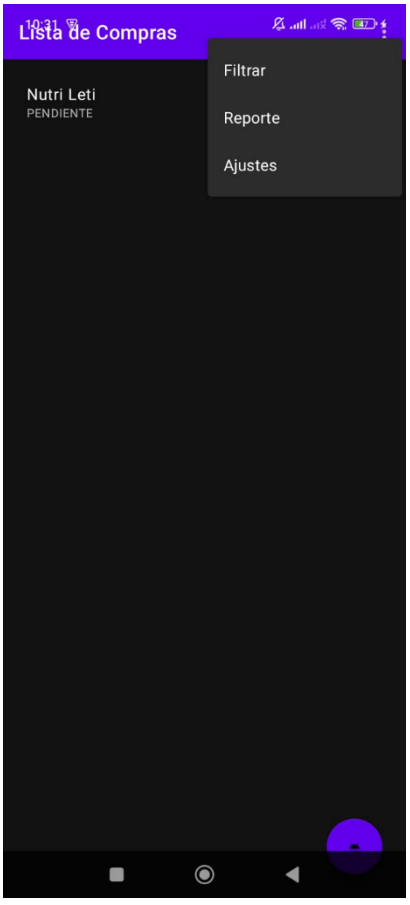


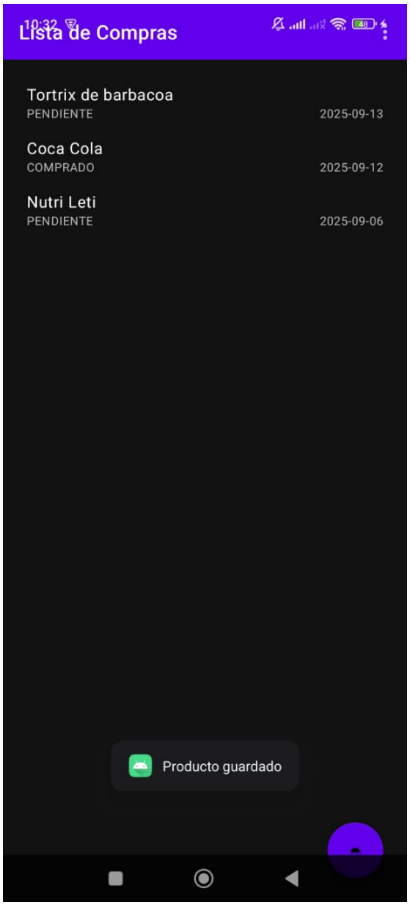

Producto
Nutri Leti
2025-09-06
PENDIENTE
COMPRADO
GUARDAR

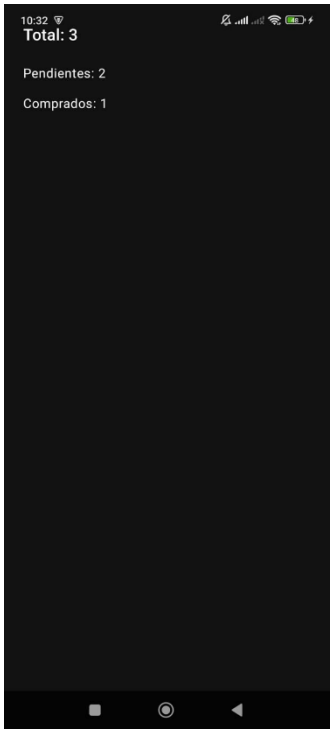

Producto
Nutri Leti
2025-09-06
PENDIENTE
6
GUARDAR

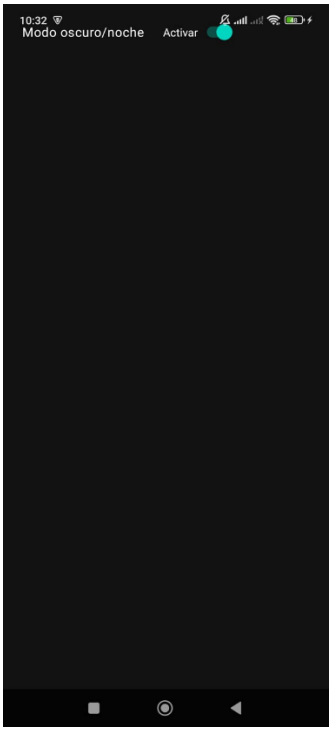

Lista de Compras
Nutri Leti
PENDIENTE
2025-09-06
Producto guardado

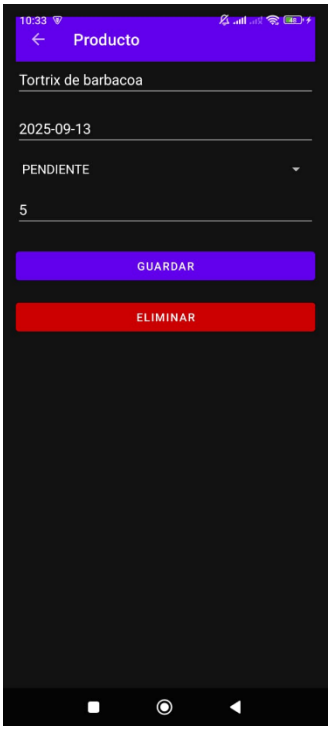

Producto
Coca Cola
2025-09-12
COMPRADO
3
GUARDAR

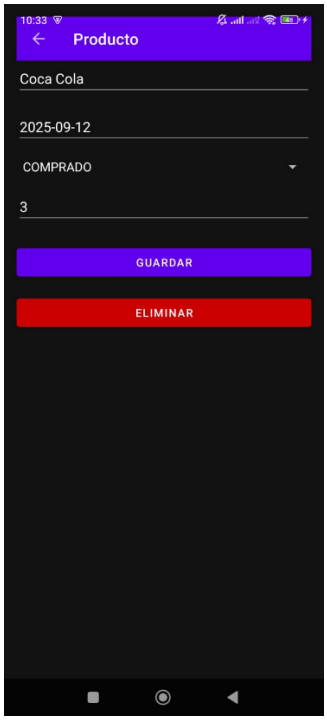

Lista de Compras
Nutri Leti
PENDIENTE
Filtrar
Reporte
Ajustes

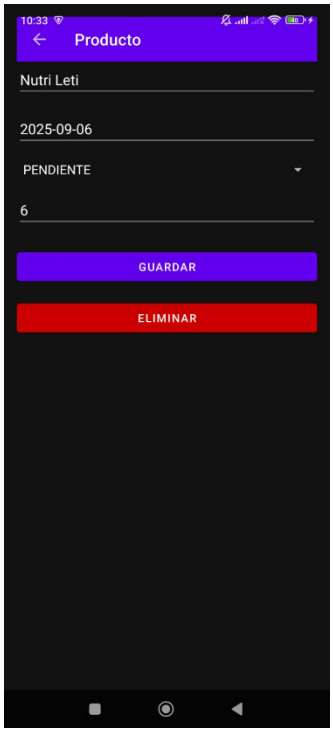

Lista de Compras
Tortrix de barbacoa
PENDIENTE
2025-09-13
Coca Cola
COMPRADO
2025-09-12
Nutri Leti
PENDIENTE
2025-09-06
Producto guardado


10:32
Total: 3
Pendientes: 2
Comprados: 1


10:32
Modo oscuro/noche
Activar


10:33
Producto
Tortrix de barbacoa
2025-09-13
PENDIENTE
5
GUARDAR
ELIMINAR


10:33
Producto
Coca Cola
2025-09-12
COMPRADO
3
GUARDAR
ELIMINAR


10:33
Producto
Nutri Leti
2025-09-06
PENDIENTE
6
GUARDAR
ELIMINAR

Incisos para realizar de la tarea

(están incluidos ambos proyectos en las respuestas)

a) Documento con pantallazos de las aplicaciones

- Toma capturas de pantalla de tus apps funcionando en el emulador o en tu teléfono.
- Asegúrate de mostrar:
    1. Pantalla de inicio.
    2. Escaneo de huella con animación.
    3. Registro manual completado.
    4. Historial de registros y métricas.
    5. Nombre de la app “Biopass” en la esquina inferior derecha.

Esto demuestra que la app funciona correctamente y que todos los elementos de UI y procesos están activos.

b) Experiencia y prompts utilizados

- Para desarrollar las apps utilicé prompts **muy detallados** en formato Markdown, especificando:
    1. Escaneo de huella digital con sensor del teléfono.
    2. Registro manual de nacionalidad y motivo.
    3. Persistencia con SQLite / Room.
    4. Indicadores de estado y métricas en tiempo real.
    5. UI en **modo oscuro** y con “Biopass” en la parte inferior derecha.

- **¿Me salió a la primera?**
    - No completamente; algunos ajustes fueron necesarios, como:
        - Agregar colores faltantes en colors.xml.
        - Crear drawables que faltaban (por ejemplo, progress_bar_dark.xml).
        - Revisar errores de redeclaración de clases y margin mal escrito en layouts.
- Corrigiendo estos detalles, la app quedó funcional y estable.

c) Qué aprendí en ambas aplicaciones de VibeCoding

- Cómo estructurar **prompts claros y completos** para que la IA genere código funcional.
- La importancia de la **persistencia local** con SQLite y Room para guardar datos de forma segura.
- Manejo de **errores y reintentos** en procesos biométricos.
- Diseño de **UI minimalista y funcional**, adaptando layouts y colores para modo oscuro.
- Cómo **integrar métricas en tiempo real** y control de estados (Idle, Scanning, Success, Error).
- Valor de planear bien la arquitectura (MVVM, Fragments, Repository) antes de programar.

Nota: como no somos perfectos y no salió a la primera, pero para no tener tantos promt, juntamos todos nuestros promt en un solo esto para cada proyecto.

Mi experiencia fue muy buena aprendí como a manejar la creación de promts profesionales.