Programa Lista de Compras de Mercado

Experiencia en la creación del Programa.

Aprender a usar agentes y creación por vibecoding fue algo gratificante, por la parte del programa fue realmente agradable ver la programación asistida a costa de un buen prompt que ayuda a la inteligencia poder realizar el código del programa mismo.

Prompts utilizados en el programa de lista de Mercado:

Prompt 1: Desarrolla una aplicación en Android con java y toma como base el proyecto que está abierto que haga lo siguiente:

- Cree una lista de compras con el nombre del producto y un estado, el estado puede ser (pendiente de compra y comprado) y fecha
- Los datos deben de ser guardados en sqLite .
- Incluye filtros por fecha o estado Tareas.
- Crea las activitys y views necesarias para la aplicación y su funcionalidad.
- Ponle un modo oscuro.
- Haz un reporte en base a los filtros. Si tienes alguna duda, pregunta antes de generar código.

Prompt 2: quiero que agregues a mi aplicación un modo oscuro (modo noche) que se puede activar y desactivar.

- El fondo principal debe de cambiar a colores oscuros
- El texto debe de ser claro
- Los botones, tarjetas y componentes deben mantener contrastes suficientes, usando tonos oscuros con bordes o sombras útiles
- Asegúrate que todos los estilos (fondos, textos, iconos y botones) se actualicen correctamente en ambos modos
- Usa buenas prácticas de accesibilidad
- Mantén el diseño consistente, moderno y agradable a la vista

¿Me salió a la primera?

En su mayoría todo me salió a la primera en la creación de código donde tuve gran parte de partida y solamente añadir una que otra cosa más necesaria para el correcto funcionamiento del programa.

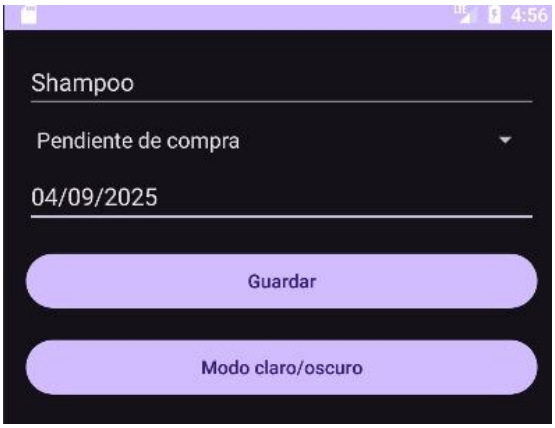
¿Tuve que corregir algo?

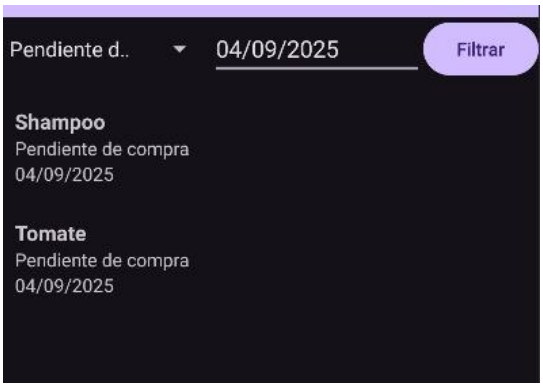
Si tuve que corregir unos cuantos errores, por ejemplos los errores típicos en los .xml con respecto a tamaños de los textview, buttons, edittext, etc. También en la parte de la creación del modo oscuro o nocturno, donde tuve que corregir bastante y crear un prompt especifico para la correcta creación e integración de este modo al programa.

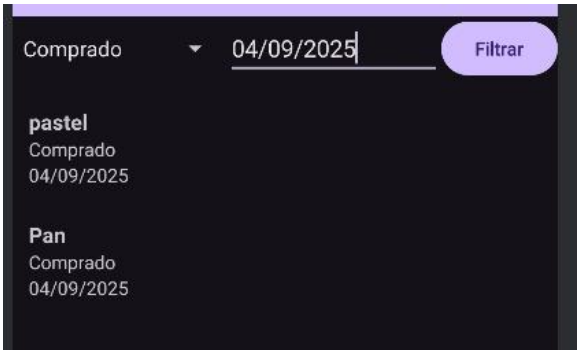
¿Qué aprendí en la aplicación desarrollada con vibecoding?

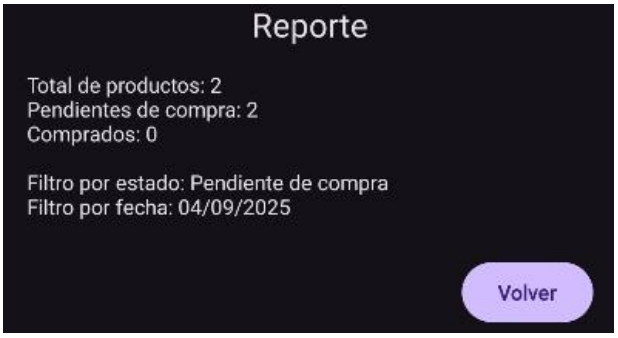
Principalmente conocer y poder hacer uso de este sistema de programación vibecoding; aprendí también más sobre la función de los botones, editext, viewtext, etc. Ya que el programa me demostró una forma más de como distribuir estos, en la creación de los activitys lo mismo. Aprendí sobre algo de mejor practica de programación y estructura de la creación de código del programa.

Capturas del funcionamiento.









Programa Lector de Huellas

Experiencia en la creación del Programa.

Aprender a utilizar agentes y aplicar el enfoque de vibecoding en la creación de un programa lector de huellas digitales fue una experiencia sumamente gratificante. Resultó interesante comprobar cómo, a través de la interacción con la inteligencia artificial, se podía orientar el desarrollo del proyecto sin necesidad de escribir cada línea de código de manera manual. Mediante la construcción de prompts adecuados fue posible guiar al modelo paso a paso, logrando que generara funciones clave para la captura, verificación y gestión de huellas. Este proceso no solo simplificó el trabajo técnico, sino que también permitió concentrarse más en el diseño y en la utilidad práctica del sistema. En definitiva, fue muy enriquecedor ver cómo la programación asistida potencia la creatividad y convierte una idea compleja, como un sistema biométrico, en un prototipo funcional.

Prompts utilizados en el programa de lector de huellas:

Prompt 1: Crea una aplicación Android en Java con las siguientes especificaciones:
La aplicación debe tener una sola Activity principal con un formulario que incluya los siguientes campos de entrada:

- Nombre
- Edad
- Nacionalidad

Al enviar el formulario: Si la nacionalidad ingresada es "Guatemalteca" o "Estadounidense" (sin importar mayúsculas o minúsculas), los datos deben guardarse directamente en una base de datos SQLite.
Para cualquier otra nacionalidad:

a. Implementa una clase independiente llamada BiometricHelper que gestione toda la lógica de autenticación mediante huella digital usando la API BiometricPrompt.
b. La clase BiometricHelper debe:
    - Iniciar el proceso de escaneo biométrico.
    - Manejar un temporizador que mida la duración del escaneo.
    - Notificar a la Activity principal mediante callbacks si la autenticación fue exitosa, cancelada o fallida.
c. La Activity principal debe comunicarse con esta clase para iniciar/detener el proceso biométrico y solo guardar los datos si la autenticación fue exitosa.

La base de datos SQLite debe organizarse creando una tabla distinta para cada nacionalidad registrada.
La interfaz debe mostrar mensajes claros y comprensibles para informar sobre estados, errores o resultados de la autenticación.
El código debe estar bien comentado, explicando la función de cada parte importante.
Incluye en el archivo AndroidManifest.xml:

- La declaración de la Activity.
- Los permisos necesarios para el uso de la autenticación biométrica.

El diseño debe ser modular, manteniendo la lógica biométrica en la clase BiometricHelper para que el código de la Activity principal se mantenga limpio y fácil de mantener.

Finalmente, entrega todo el código completo y funcional para que pueda ejecutarse correctamente en Android Studio sin errores.

¿Me salió a la primera?

En mayor parte se creo lo solicitado, tuve unas fallas donde tuve que cambiarle unas cosas y así poder mejorar el coding del programa.

¿Tuve que corregir algo?

Si tuve que corregir unos cuantos errores, como mantener la lógica en la clase BiometricHelper y cosas más del códing.

¿ Qué aprendí en la aplicación desarrollada con vibecoding?

Lo más valioso sobre este ejercicio fue la experiencia de estructurar una aplicación biométrica lectora de huellas, combinando programación tradicional con el apoyo de la IA mediante el enfoque de *vibecoding*. No se trata únicamente de lo técnico, como trabajar con Java, SQLite o la API BiometricPrompt, sino también de un cambio de mentalidad: usar la IA como asistente para concentrarme en la arquitectura y la lógica del sistema en lugar de solo escribir código. Al mismo tiempo, aprendí a manejar mejor los elementos de interfaz como botones, EditText y TextView, así como la organización de las Activities, lo que me permitió comprender buenas prácticas de programación y una forma más clara de estructurar el código del programa.

Capturas del funcionamiento.