Guión de la tercera práctica

Planificación en seis horas para desarrollar esta app en Android usando Java, con integración a la aplicación RESTful de comercio electrónico que ya. Esta planificación asume conocimientos previos en Android Studio y en RESTful APIs.

1. **Configuración del Proyecto y Conexión a la API (60 minutos)**
   - Configura el proyecto en Android Studio con el nombre y paquete adecuado.
   - Configura dependencias para el acceso a internet y JSON (usar Retrofit para consumir la API y Gson para la serialización).
   - Prueba la conexión inicial a la API: haz una solicitud básica para obtener productos y muestra los datos en un `RecyclerView` básico.
2. **Implementación de la Interfaz de Usuario para el Catálogo de Productos (60 minutos)**
   - Diseña la interfaz para mostrar la lista de productos en un `RecyclerView`.
   - Implementa los adaptadores y modelos para el catálogo.
   - Agrega detalles básicos de productos (nombre, precio, imagen).
   - Conecta la interfaz con la API usando Retrofit y muestra el catálogo en la app.
3. **Creación de la Funcionalidad de Carrito de Compras (60 minutos)**
   - Diseña la pantalla del carrito: incluye un `RecyclerView` para los productos seleccionados.
   - Implementa la lógica del carrito (agregar/quitar productos).
   - Persistencia temporal: utiliza `SharedPreferences` o un objeto en memoria para almacenar el carrito del usuario.
   - Prueba la funcionalidad de agregar/quitar productos y mostrar el carrito actualizado.
4. **Funcionalidad de Compra y Checkout (60 minutos)**
   - Crea la pantalla de checkout para confirmar la compra.
   - Implementa la lógica de compra: envía una solicitud a la API para procesar el pedido.
   - Manejo de errores y mensajes de confirmación al usuario.
5. **Integración con Google Maps (60 minutos)**
   - Configura Google Maps API en tu proyecto.
   - Muestra la ubicación del usuario y de los almacenes (puntos de venta) según la respuesta de la API.
   - Desarrolla la funcionalidad para que el usuario pueda ver la ubicación de los almacenes en el mapa y obtener indicaciones.
6. **Gestión de Productos-Sólo Personal Autorizado-(60 minutos)**
   - Crea una sección para la gestión de productos: incluir un formulario de autenticación para verificar al usuario.
   - Permite agregar y eliminar productos en el catálogo enviando las solicitudes correspondientes a la API.
   - Agrega validación de permisos y mensajes de error para usuarios no autorizados.
7. **Pruebas Finales**

Al finalizar cada hora, prueba la funcionalidad desarrollada en dispositivos/emuladores Android para asegurarte de que las partes integradas estén funcionando correctamente. Esto te ayudará a identificar y corregir errores sobre la marcha.

Entregables

1. **Código Fuente de la Aplicación Android**

- Todo el código fuente (carpetas src, res, y otros archivos de configuración de Android Studio).
- **Subcarpetas**:

  `src/main/java`: Código Java de la app, organizado en paquetes (`models`, `api`, `adapters`, `views`, etc.).

  `src/main/res`: Recursos (layouts XML, imágenes, etc.).

  src/main/AndroidManifest.xml: Archivo de configuración general de la app.

- **Proyecto**: Este archivo contendrá toda la estructura de tu app para que pueda ser abierta y ejecutada directamente en Android Studio.

2. **Archivo README**

- **Descripción**: Un archivo `README.md` que incluya:
  - Instrucciones de instalación y ejecución de la app.
  - Dependencias usadas (Retrofit, Gson, Google Maps API).
  - Descripción de la estructura de carpetas y organización del código.
  - Lista de endpoints API utilizados y su descripción.
- **Formato**: Puede incluir ejemplos de uso y capturas de pantalla, si es útil para los revisores.

3. **Manual de Usuario y Documentación Técnica**

- **Manual de Usuario:** Instrucciones detalladas sobre el uso de la aplicación, cómo buscar productos, usar el carrito, realizar compras y gestionar productos.
- **Documentación Técnica:** Explicación de la arquitectura de la app, diagramas de clases, flujo de datos, y cómo se realiza la autenticación y autorización para la gestión de productos.
- **Formato:** Puedes incluir estos documentos en formato PDF.

4. **Pruebas**

- Archivos de pruebas unitarias o funcionales si se desarrollaron (pueden ser archivos Java en el proyecto Android o un reporte de pruebas en PDF).

Notas:

`RecyclerView e`s un widget de Android que proporciona una forma eficiente y flexible de mostrar grandes conjuntos de datos en una lista o cuadrícula en la interfaz de usuario. Es una versión más avanzada y eficiente de ListView y permite reciclar las vistas para optimizar el uso de memoria y el rendimiento, especialmente en listas grandes.

`SharedPreferences` es una clase en Android que permite almacenar datos de tipo clave-valor en un archivo de preferencias. Es útil para guardar pequeñas cantidades de datos que deben persistir entre las sesiones de la aplicación, como configuraciones de usuario, tokens de autenticación, preferencias de la app, o datos del estado de la aplicación. `SharedPreferences` es ideal para almacenar datos simples y no sensibles, ya que no es una opción completamente segura para datos críticos (como contraseñas).

Organización en 2 Archivos ZIP para la entrega

Para reducir la cantidad de archivos y cumplir con la limitación de 2GB, puedes agrupar estos entregables en dos archivos ZIP:

1. **Archivo ZIP Principal** (ApellidosNombre_AppSource.zip):
- Incluye el código fuente completo de la aplicación (carpetas `src` y `res`, archivo `AndroidManifest.xml`).
- Tamaño esperado: Esto debería ser el archivo más grande, pero debería mantenerse dentro de la limitación < 2GB si no incluye archivos multimedia pesados.

2. **Archivo ZIP Secundario** (ApellidosNombre_DocsAndTests.zip):
- Incluye el README.md y la Documentación (manual de usuario, documentación técnica).
- Pruebas: Archivos de pruebas y reportes en PDF.
- Tamaño esperado: Dado que son documentos, este archivo debería ser considerablemente menor que el anterior.