
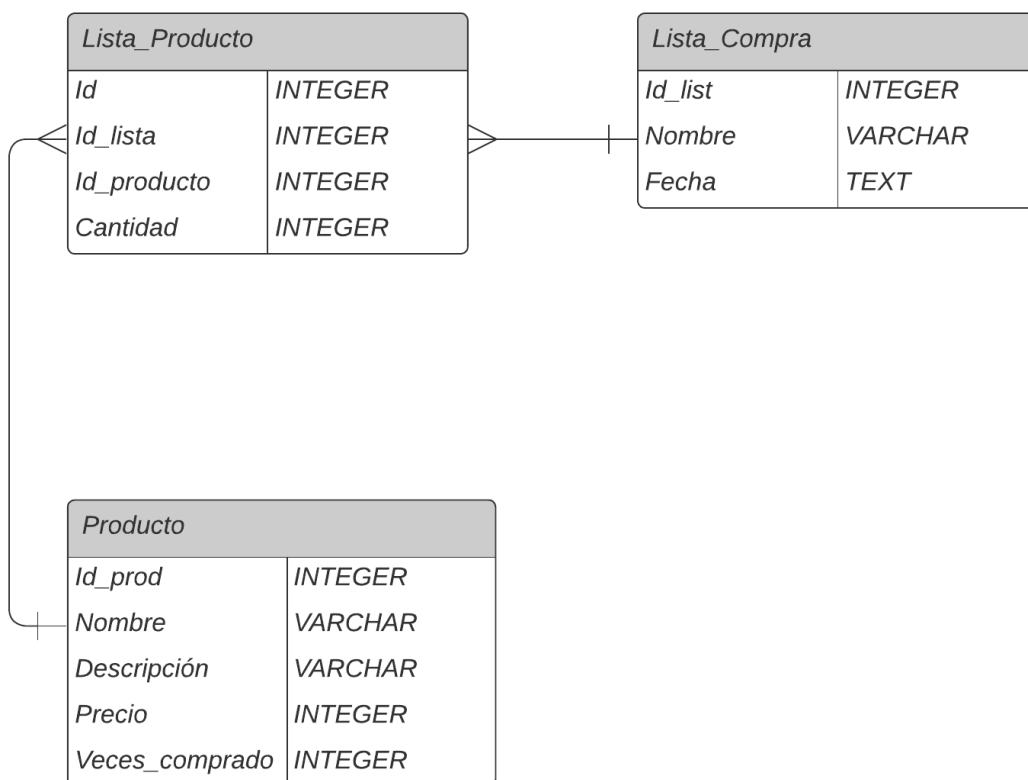


ACTIVIDAD 4.	
 <div> UNIDAD DE TRABAJO 4. COMUNICACIONES EXTERNAS </div>	
MÓDULO: PROGRAMACIÓN MULTIMEDIA Y DE DISPOSITIVOS MÓVILES	GRUPO: DAM2
FECHA: 20/01/2024	

ALUMNO/A (Nombre y Apellidos): David Ramos del Pino

Diagrama de ER (Entidad-Relación) de la BD



La BD cuenta con tres tablas, Producto, Lista_Compra y Lista_Producto.

Tabla Producto:

Esta tabla representa los productos que pueden estar en una lista de compras. Contiene la siguiente información: el id del producto, el nombre del producto, la descripción del producto, URL de la imagen, precio y la cantidad de veces que ha sido comprado.

Tabla Lista_Compra:

Esta tabla representa una lista de la compra. Contiene la siguiente información: el id de la lista, el nombre de la lista y la fecha de creación de la lista.

Tabla Lista_Producto:

Esta tabla es la tabla de relación, básicamente sirve para implementar una relación muchos a muchos entre las listas de compras y los productos. Se utiliza para registrar qué productos están asociados a cada lista de compras y la cantidad de cada producto en esa lista. Contiene la siguiente información: id propio de cada registro en la tabla, un id de lista, un id de producto y la cantidad

Explicación de Proyecto

¿Qué pasa al iniciar la Aplicación?

Creación de la BD

Se crea o abre la base de datos SQLite llamada "ListasCompras" y después se ejecutan consultas SQL para crear las tablas lista_compra, producto, y lista_producto en la base de datos, si no existen.

Verificación de Permisos

Verifica si la aplicación tiene permisos para leer contactos (READ_CONTACTS) y para enviar SMS (SEND_SMS). Si no tiene permisos, solicita permisos al usuario.

Concesión de Permisos

Se verifica si se han concedido los permisos en el método onRequestPermissionsResult, si se ha concedido el permiso contactos se llama al método leerContactos()

Leer Contactos

En este método utilizo el ContentResolver para acceder a la base de datos de contactos del dispositivo y recopilar la información de los contactos como pueden ser nombres y números de teléfono, después con esa información creo objetos Contacto y los agrego a un ArrayList estático de Contactos que está en la clase ListaContactos

Resumen de Funcionamiento de las Opciones

Crear Lista



Nombre de La Lista:

CREAR

Al darle click a la opción Crear Producto se abrirá el Fragmento **“HomeFragment”**, lo que hace este fragmento simplemente es que al darle al botón Crear coge la información de los campos y realiza un insert a la BD con el id de la lista, el nombre y la fecha, la fecha la obtiene del método **obtenerFechaActual()**, lo que hace ese método es realizar una consulta para obtener la fecha actual de la base de datos y la devuelve como una cadena. Después de haber hecho el insert creo un Bundle para pasar el ID de la lista recién creada como argumento al Fragment que me abrirá el RecyclerView de

Productos, utilizo el **Navigation.findNavController(v).navigate()** para navegar al **productoFragment** y le paso el Bundle con el ID de la lista.



Pimiento rojo Cantidad: _____



Pimiento rojo grande

Precio: **1**

AÑADIR

ELIMINAR

Pan Cantidad: _____



Barra de pan

Precio: **0**

AÑADIR

ELIMINAR

Yogures Cantidad: _____



Yogur natural de snatado



Una vez en el **ProductoFragment** obtengo el ID de la lista desde los argumentos del fragmento y lo guardo en una variable, después llamo al metodo **cargarProductosDesdeJSON()** de la clase **ProductoContent** y le paso como parámetro el contexto y el recyclerview de productos, una vez realizado todos los métodos de la clase **ProductoContent** utilizare el id de la lista guardado anteriormente pasándolo como parámetro al adapter del recyclerview.

¿Que pasa cuando llamamos al Método **cargarProductosDesdeJSON()** de la clase **ProductoContent**?

Recibe el contexto y el RecyclerView como parámetros. y crea una instancia de la clase interna **HttpAsyncTask** y ejecuta la tarea asíncrona para cargar productos desde el JSON, así obtengo un arraylist de Productos, con un método compruebo si ese producto está en la BD o no, si está llamo al método **actualizarVecesCompradoEnBD()** que como su propio nombre indica actualiza la columna veces_comprado de la tabla Productos, y si no está llamo al método **guardarProductoEnBD()** que guarda el producto en la BD.

En el Adaptador del recyclerview llamado **MyProductoRecyclerViewAdapter** Utiliza un **ExecutorService** para descargar las imágenes de los productos de una URL de forma asíncrona y no bloquear la interfaz de usuario. Y luego tengo configurados dos onClickListener para dos botones, eliminar y guardar, al pulsar cualquiera de los dos botones llama al método **insertarEnBD()**(pero antes voy a explicar como he pensado en los registros de una lista con el producto elegido, yo he pensado que solo debe haber un registro por cada producto de cada lista, por ejemplo si en la lista 1 tengo 1 pan cada vez que yo añada o elimine panes en la lista 1 solo modificare el registro que ya existe no creare uno nuevo) primero, obtiene el ID correspondiente al producto a través de su nombre, luego, verifica si ese producto ya está en la lista de la compra si existe, actualiza la cantidad, si no, inserta un nuevo registro además de eso, registra el número de veces que se ha comprado este producto llamando al método **actualizarVecesCompradoEnBD()** en la clase **ProductoContent**

Crear Productos

 Crear Producto 

Crear Producto

Nombre:

Descripcion:

Precio:

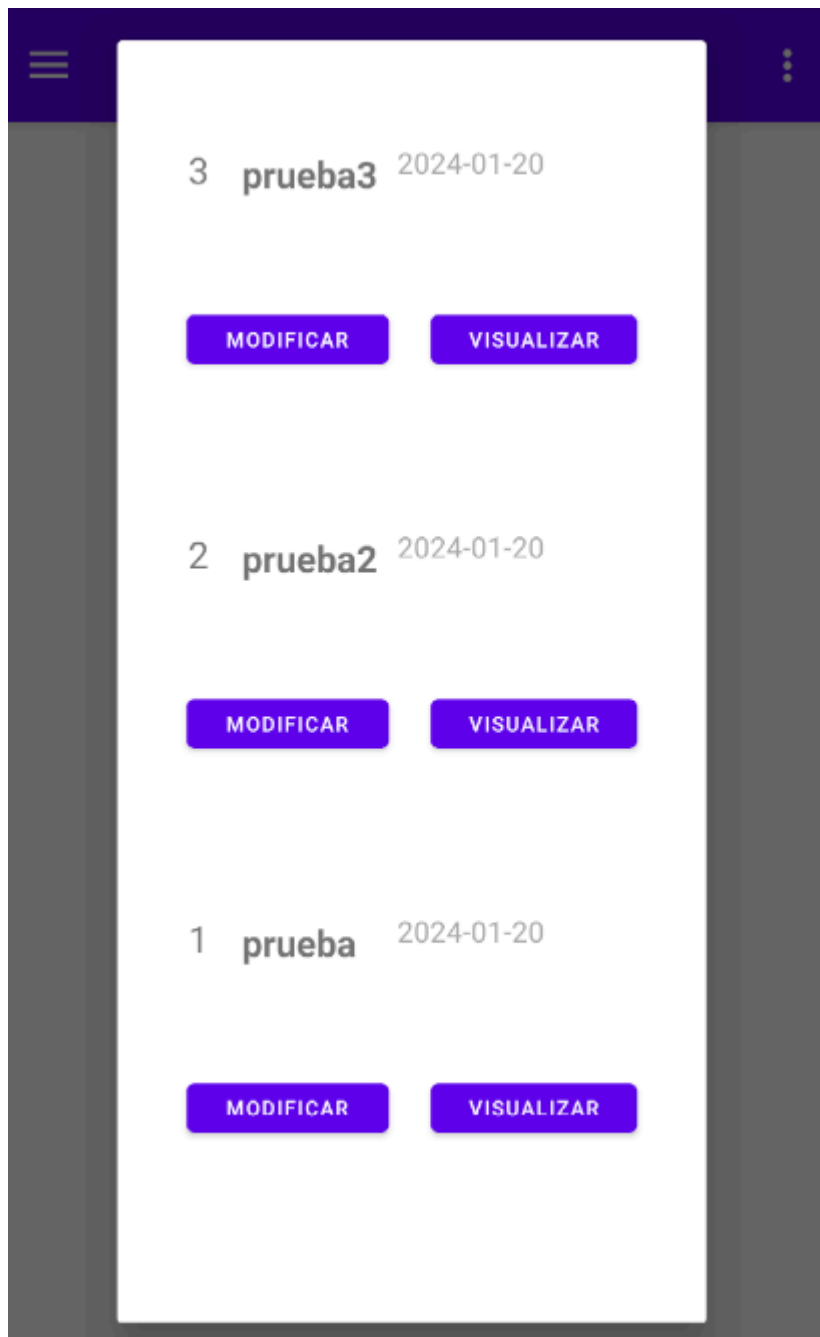
Nombre Imagen:

CREAR

Al darle click a la opción Crear Producto se abrirá el Fragmento **“SlideshowFragment”**, lo que hace este fragmento simplemente es que al darle al botón Crear coge la información de los campos y realiza un insert a la BD (Tienes que tener todos los campos con Información, excepto la foto, el campo foto es para que si subes una imagen nueva a la carpeta en la nube deberías escribir su nombre en este campo y la imagen se cargaría en este Producto en el RecyclerView)

Consultar Listas

Al darle click a la opción Consultar Listas se abrirá el Fragmento **“GalleryFragment”**, lo que hará este fragmento es una consulta a la tabla lista_compra en la que me sacará todas las listas ordenadas por la fecha de manera descendente y esas listas las guardaré en un arraylist de listas, después de eso creará un diálogo de la clase **DialogoListas** pasándole como parámetro el arraylist de listas y mostrará el diálogo, en la clase **DialogoListas** lo que haré será configurar el adaptador para el recyclerview llamado **MyListasRecyclerViewAdapter** y mostrare el recyclerview



en el adaptador he configurado dos botones, uno para modificar y otro para visualizar, el boton modificar lo que hará será llamar al método **abrirProductoFragment()** con el id de la lista como parámetro, lo que hará ese método es crear una instancia del fragmento **ProductoFragment** que se encargará de mostrar los productos y después reemplazar el fragmento actual, para que el fragmento se abra con la información correcta creo un Bundle para pasar el id de la lista al fragmento y después compruebo Si hay un diálogo de listas visible, si lo hay lo cierro. Para el botón visualizar la funcionalidad de reemplazar el fragmento es la misma, lo que cambia es el fragmento, es un nuevo fragmento llamado **VisualizarListaFragment** que llama a un recyclerview que solo muestra el nombre de producto, la imagen y la cantidad que ha comprado(esto lo hago con un método llamado

obtenerCantidadParaProducto()), para obtener cuales son los productos que tiene esa lista lo que hago es rellenar un arraylist de productos llamando al método **obtenerListaProductosParaListaSeleccionada()** y ese arraylist se lo paso al constructor del fragmento, la clase del adaptador se llama **MyVisualizarListaRecyclerViewAdapter**



Pimiento rojo

Cantidad: 4



Pan

Cantidad: 3

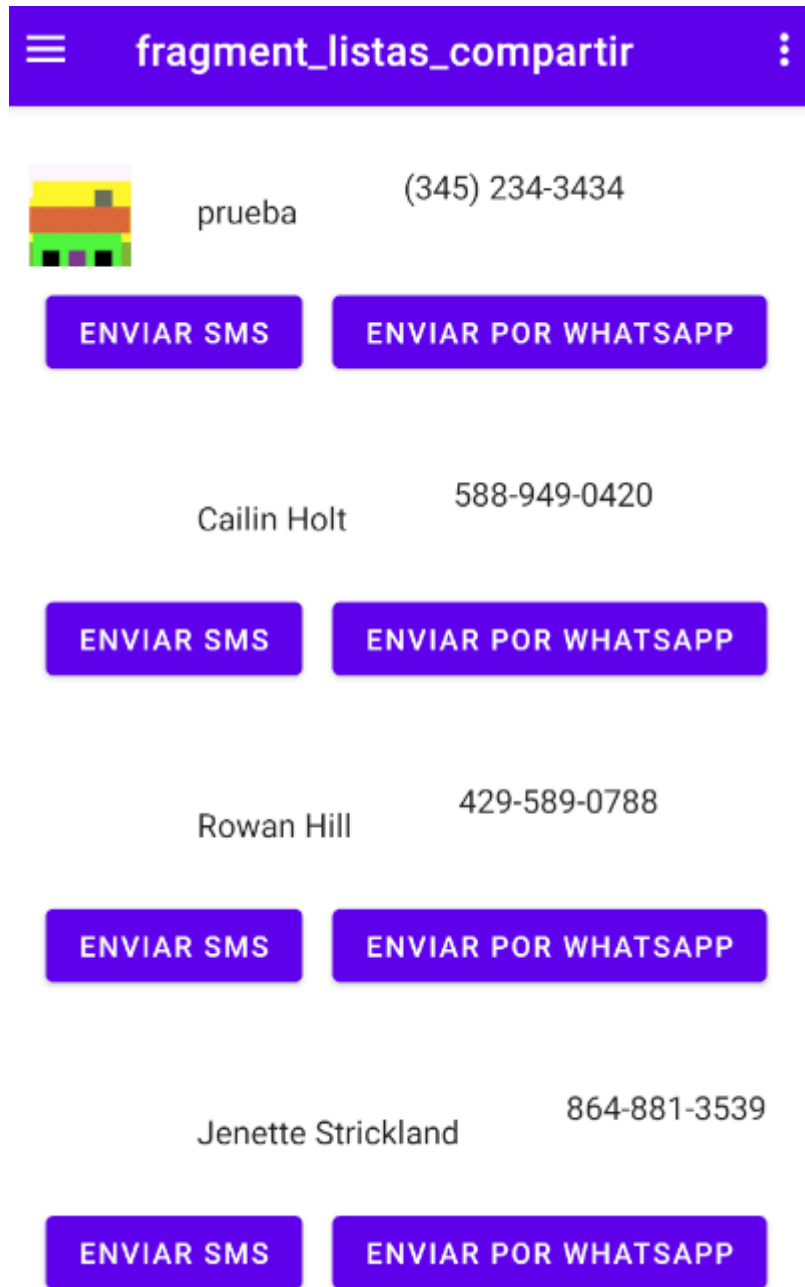
Compartir Listas

Al darle click a la opción Compartir Listas se abrirá el Fragmento **“ListasCompartir”**, lo que hará este fragmento es llenar un arraylist de listas con el método **obtenerListasOrdenadasPorFecha()** y creará un adaptador de la clase **MyListasCompartirRecyclerViewAdapter**, que mostrará un recyclerview con las listas ordenadas por fecha descendiente junto con un boton de enviar



ese botón lo que hará será crear una instancia de la clase **CompartirFragment** y con un Bundle le paso el id de la lista, después remplazo el fragmento actual por el fragmento de la clase **CompartirFragment** que acabo de crear

la clase **CompartirFragment** recibe el id de la lista y crea un adaptador de la clase **MyContactoRecyclerViewAdapter** y se lo pasa como parámetro y pone al recyclerview el adaptador, el recyclerview mostrará la imagen del contacto, el nombre, su número de teléfono y dos botones, uno para enviar un SMS con la lista y otro para enviar la lista por Whatsapp.



En el adaptador configurare la foto utilizando un **ExecutorService** para obtener la foto del contacto de manera asíncrona, y después se cargará en el **ImageView** usando un **Handler**, Después configurare los dos botones(para ambos la información del mensaje será obtenida por un método llamado **obtenerInformacionLista()**, que hará una consulta a la BD con el id de lista que le he pasado como parámetro al adapter y luego iré formateando el mensaje);

Enviar SMS:

crea un **Intent** con la acción **Intent.ACTION_VIEW**, le paso la **URI** con el formato para enviar un sms y con el **.putExtra()** le paso la información del mensaje obtenido anteriormente

Enviar por Whatsapp:

Realiza lo mismo que el enviar SMS, pero la **URI** en este caso el formato es para enviar un Whatsapp

Datos de Interés

En el **MainActivity** manejo el botón de retroceso, “el de la flecha que va para atrás”, para ello he tenido que implementar el método **onBackPressed()**, la función que realiza es que verifica si hay fragmentos en la pila y realiza la navegación hacia atrás. Si no hay fragmentos en la pila, realiza la acción predeterminada de retroceder. Además, oculta el fragmento actual si es necesario.

Implemente este método debido a que en el código del adapter **“MyListasRecyclerViewAdapter”** hay un **onClickListener** para el botón de modificar que llama al método **abrirProductoFragment()**, y uno para el botón de visualización que llama al método **abrirVisualizarListaFragment()**.

Lo que hacen ambos métodos es reemplazar el fragmento actual utilizando **FragmentManager** y agregan el fragmento nuevo a la pila de retroceso. Esto implica que al hacer clic en uno de estos botones, se crea un nuevo fragmento y se agrega a la pila de retroceso, lo que en mi caso provoca la superposición de fragmentos ya que el fragmento anterior no se oculta o elimina adecuadamente.

En el **“MyListasCompartirRecyclerViewAdapter”** tengo un método llamado **abrirCompartirFragment()** que hace lo mismo que los dos anteriores métodos, pero no da ese error, al investigarlo me di cuenta que podría ser debido a como estaban diseñados los XML, y me di cuenta de sí que es diferente ya que el **CompartirFragment** es un **FrameLayout** y los otros dos no, al intentar realizar diversos cambios para ambos XML al probar su funcionamiento no funcionaba el cambio, y me decía que la vista no existía, por lo que opte por simplemente poner **content** y de ahí que me de el error de superposición de fragmentos, es por eso que implemente este método, para solucionarlo