



Mahroz Jawad
2º DAM

Índice

1. Descripción del proyecto
2. Estructura del proyecto
3. Creación de la Base de datos
4. Servicio ApiRest
 - 4.1. Creando el servicio ApiRest desde Netbeans
 - 4.2. Utilizar el Servicio ApiRest desde el proyecto de Android Studio
5. Implementando la Aplicación
 - 5.01. Logueo
 - 5.02. Registro
 - 5.03. Página Principal
 - 5.06. Carrito
 - 5.07. Facturar
 - 5.08. Perfil
6. Funciones de Administrador
 - 6.01. Agregar Productos
 - 6.02. Eliminar Productos
 - 6.03. Editar Producto

1. Descripción del proyecto

Como el nombre de la aplicación dice que es una aplicación de moda. Efectivamente es eso. Es una aplicación en el que vamos a vender ropas. Tendrán dos modos de la aplicación uno para el cliente y el otro para el administrador. El administrador podrá añadir ropas. Para que el cliente pueda comprar las ropas.

El cliente puede ir bajando y agregando los productos en el carrito. Y al final que se vaya al carrito para Confirmar que ya se ha terminado. Y se genera una factura en la pantalla.

El administrador a su vez es un cliente también. Pero tiene algunas funcionalidades más que el cliente como, puede agregar producto, borrar productos o editar cualquier producto que ya esta para la venta.

2. Estructura del proyecto

La Estructura del proyecto he decidido muy simple.

Como ya sabemos es una aplicación del móvil, que se ha implementado con Android studio. Pues primero es una actividad, y luego se cambia los fragments en el mismo contexto. Pues he implementado que el login se hace en la actividad y luego se van cambiando los fragments. Uno a uno.

Vale, la primera pantalla que he hecho, es el de iniciar sesión, si no tienes el usuario creado, primero se crea el usuario. y después cuando se inicia la sesión, se entra en el fragment en el que tiene un recyclerView, que va mostrando los productos. Y tiene un toolbar encima. Que el toolbar tiene el nombre de la tienda, y un carrito, y una imagen con el que el cliente puede acceder a su perfil, cambiar la fotografía, el nombre, la contraseña etcétera. Y al empezar el carrito está vacío. Y si el cliente pincha a un producto, el carrito se tendrá esta ropa añadida. Y luego si el cliente quiere comprar, tiene que ir al carrito, y dar el botón para comprar. Pero si el cliente da al botón de borrar todo el carrito se borrará todo lo que tiene dentro del carrito.

El carrito tiene el recyclerView que muestra el producto. Y se puede sumar y restar la cantidad del producto o quitar el producto cuando haga falta, depende del cliente, si se ha equivocado o algo. Y igualmente con el fragment principal también se puede quitar el producto por si el cliente se equivoca.

Vale pues, cuando el cliente en el fragment de carrito de al botón de comprar se genera una factura (Diciendo que tanto dinero te has pagado, y muestra los productos que habías elegido para comprar (con el recyclerView) con la cantidad en el medio)

si el usuario es administrador, se inicia igual como los clientes, se compra igual como los clientes, pero le aparece un FAB (floating action button). Y con el botón se puede agregar los productos, para agregar se muestra para elegir la imagen, se rellena los campos, y se le da a aceptar para que se guarde todo en la base de datos.

Pero si no quiere añadir lo, lo cancela y se lo cancelará la agregación y se vuelve a su fragment principal. Pero si el administrador se lo da al botón de agregar, y se nota que se ha equivocado, no tendrá que preocupar, tiene la opción para borrar el producto en el fragment principal. Solo el administrador puede hacer el Longclick sobre el producto que quiere borrar. Aparece un dialogo para cancelar y borrar., vale y si el administrador no quiere borrar, solamente quiere editar, pues hace un click al producto y se llevará a otro fragment, en el que se podrá editar los datos del producto. Y si no quiere editar , lo cancela, y si quiere, lo modifica y guarda los cambios. Y se guardarán los cambios en la base de datos.

3. Creación de la Base de datos

En la base de datos he creado solo dos tablas, una tabla para los usuarios, y otra tabla para los productos de la tienda de la aplicación.

```
CREATE TABLE `productos` (  
  `id` int(11) NOT NULL,  
  `nombre` varchar(50) CHARACTER SET utf8 DEFAULT NULL,  
  `imagen` text CHARACTER SET utf8,  
  `descripcion` text CHARACTER SET utf8,  
  `stock` int(11) DEFAULT NULL,  
  `precio` double DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci
```

```
CREATE TABLE `usuarios` (  
  `id` int(11) NOT NULL,  
  `usuario` varchar(50) CHARACTER SET utf8 DEFAULT NULL,  
  `contrasena` varchar(20) CHARACTER SET utf8 DEFAULT NULL,  
  `nombre` varchar(50) CHARACTER SET utf8 DEFAULT NULL,  
  `apellidos` varchar(50) CHARACTER SET utf8 DEFAULT NULL,  
  `imagen` text CHARACTER SET utf8,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_spanish_ci
```

Tablas producto y usuarios que tiene la id como primary Key que no puede ser nulo.

Cada campo describe lo que se guarda en cada uno, creo que no hace falta explicarlo.

Y he insertado un usuario administrador que su id es "0" y en el proyecto el administrador inicia sesión con la id cero que nunca va a cambiar. Y las funcionalidades que puede hacer el administrador, podrá hacer con esta cuenta.

```
INSERT INTO usuarios (id, usuario, contraseña, nombre, apellidos, imagen) VALUES  
(0, "admin", "admin", "Mahroz", "Jawad", "");
```

4. Servicio ApiRest

Un servicio REST no es una arquitectura software, sino un conjunto de restricciones con las que podemos crear un estilo de arquitectura software, la cual podremos usar para crear aplicaciones web respetando HTTP.

4.1. Creando el servicio ApiRest desde Netbeans

Creamos un proyecto java web en Netbeans. Le agregamos un servicio que se llama en inglés (Restful web services from patterns) y nos crea una clase de configuración del ApiRest que vamos a implementar en Netbeans.

```
package MiModaBD;

import java.util.Set;
import javax.ws.rs.core.Application;

@javax.ws.rs.ApplicationPath("recursos")
public class ApplicationConfig extends Application {

    @Override
    public Set<Class<?>> getClasses() {
        Set<Class<?>> resources = new java.util.HashSet<>();
        addRestResourceClasses(resources);
        return resources;
    }

    /**
     * Do not modify addRestResourceClasses() method.
     * It is automatically populated with
     * all resources defined in the project.
     * If required, comment out calling this method in getClasses().
     */
    private void addRestResourceClasses(Set<Class<?>> resources) {
        resources.add(MiModaBD.ApiRest.class);
    }
}
```

Y La clase ApiRest que nos crea, hay tenomos que poner las ordenes que queremos para la api. Por Ejemplo POST, PUSH, GET, etc.

Antes de todos Creamos la clase usuarios, Productos con los mismos campos que tenemos en la base de datos para referenciarlos. Y creammos sus getters y setter, y lo que haga falta.

```
public class Usuario implements Serializable{  
    private int id;  
    private String usuario;  
    private String contrasena;  
    private String nombre;  
    private String apellidos;  
    private String imagen;  
  
    public Usuario() {  
    }  
}
```

```
public class Producto implements Serializable {  
    private int id;  
    private String nombre;  
    private String imagen;  
    private String descripcion;  
    private int stock;  
    private double precio;  
  
    public Producto() {  
    }  
}
```

Y empezamos con la clase ApiRest.

```
@Path("apiREST")  
public class ApiRest {  
  
    @Context  
    private UriInfo context;  
  
    /**  
     * Creates a new instance of ApiRest  
     */  
    public ApiRest() {  
    }  
}
```

la ruta de la clase configuración “recurso” y la ruta de la clase ApiRest “apiREST” juntamos con la ip del servidor, así obtendremos la ruta para llegar hasta la api (ip:puerto/recursos/apiREST), configurado en netbeans.

Vamos allá, Para crear ordenes de la Api, vamos a crear una clase normal con los métodos estáticos para llamar lo desde cualquier lugar, en este caso para llamarlo desde la clase ApiREST.

Voy a poner abajo todos a la vez y luego pongo los métodos que se ejecutan cada uno de los ordenes que implementamos.

```
@GET
@Path("/usuarios")
@Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
public ArrayList<Usuario> getUsuarios() {
    return DAOModa.getUsuarios();
}

@POST
@Path("/add_usuario")
@Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
public Response addUsuario(Usuario user) throws Exception{
    return DAOModa.addUsuario(user);
}

@PUT
@Path("/actualizar_usuario")
@Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
public Response ActualizarUsuario(Usuario u) {
    return DAOModa.ActualizarUsuario(u);
}
```

GET para obtener todos los usuarios, put para actualizar un usuario con el usuario, POST para agregar el usuario.

Igual vamos a hacer con los productos.

```
@GET
@Path("/productos")
@Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
public ArrayList<Producto> getProductos() {
    return DAOModa.getProductos();
}

@POST
@Path("/add_producto")
@Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
public Response addProducto(Producto p) throws Exception{
    return DAOModa.addProducto(p);
}

@PUT
@Path("/actualizar_producto/{id}")
@Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
public Response ActualizarProducto(@PathParam("id") int id, Producto p) {
    return DAOModa.ActualizarProducto(id, p);
}
```

GET, Obtener todos los productos, POST, agregar producto, PUT, para Actualizar con una id del producto.

Y para el producto he agregado el modo borrar para el administrador. Que si quiere borrar algún producto.

Pero para el usuario no lo creo porque si ya estas registrado, pues que estas registrado para siempre, si queremos podremos borrar el usuario con la base de datos manualmente.

```
@DELETE
@Path("/eliminar_producto/{id}")
@Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
public Response EliminarProducto(@PathParam("id") int id) {
    return DAOModa.EliminarProducto(id);
}
```

En las capturas anteriores, podemos ver que con la clase DAOModa sacamos lo que necesitamos, vamos a la clase DAOModa, Y implementamos lo que queremos que saque o devuelve para la aplicación que vamos a implementar en androidStudio.

Agregamos la librería “mysql conector java 5.1.48” para conectar con la base de datos que hemos creado

```
public class DAOModa {
    private static Connection con = null;

    static {
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(DAOModa.class.getName()).log(Level.SEVERE, null, ex);
        }
        String url = "jdbc:mysql://localhost:3306/mi_moda";
        url += "?autoReconnect=true&useSSL=false&zeroDateTimeBehavior=convertToNull";
        String usuario = "root";
        String password = "1234";
        try {
            con = (Connection) DriverManager.getConnection(url, usuario, password);
        } catch (SQLException ex) {
            Logger.getLogger(DAOModa.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

primero de todo Crearemos la conexión con nuestro base de datos, por eso necesitamos por static y entre las llaves la conexión con el usuario y contraseña a la base de datos. Utilizando la técnica JDBC.

Ese método estático se ejecuta, cuando le llamamos a la clase.

Vale vamos con nuestra lógica de base de datos que es realmente que nos importa para realizar cambios en la base de datos.

Ya que tenemos la conexión hecha, implementamos nuestros métodos que hemos visto en nuestra clase ApiRest.

```
public static ArrayList<Usuario> getUsuarios() {
    Usuario objeto = null;
    ArrayList<Usuario> usuarios = new ArrayList<>();
    try {
        // Crear Statement de la Consulta
        String sentenciaSQL = "SELECT * FROM usuarios";
        Statement statement = con.createStatement();
        // ResultSet
        ResultSet rs = statement.executeQuery(sentenciaSQL);
        while (rs.next()) {
            // Cargar valores
            objeto = new Usuario();
            objeto.setId(rs.getInt("id"));
            objeto.setNombre(rs.getString("nombre"));
            objeto.setContrasena(rs.getString("contrasena"));
            objeto.setApellidos(rs.getString("apellidos"));
            objeto.setUsuario(rs.getString("usuario"));
            objeto.setImagen(rs.getString("imagen"));

            usuarios.add(objeto);
        }
        rs.close();
        if (usuarios == null) {
            System.out.println("No existe ningún objeto");
        }
    } catch (SQLException ex) {
        Logger.getLogger(DAOModa.class.getName()).log(Level.SEVERE, null, ex);
    }
    return usuarios;
}
```

Rellenamos los usuarios con los valores que sacamos con la sentencia SQL. Y retornamos la lista de usuarios.

```

public static Response addUsuario(Usuario user) {
    ResponseEntity entity = new ResponseEntity();
    boolean encontrado = false;
    for (Usuario u : getUsuarios()) {
        if (u.getUsuario().equals(user.getUsuario())) {
            encontrado = true;
            break;
        }
    }
    if (!encontrado) {
        try {
            int size = getMaxIdTabla("usuarios") + 1;
            String insertTableSQL = "INSERT INTO usuarios"
                + "(id, nombre, contrasena, apellidos, usuario, imagen) VALUES"
                + "(" + size + ", '" + user.getNombre() + "', '" + user.getContrasena() + "', '" + user.getApellidos() + "', '" + user.getUsuario() + "', '" + user.ge
            PreparedStatement preparedStatement = con.prepareStatement(insertTableSQL);
            preparedStatement.executeUpdate();

        } catch (SQLException ex) {
            Logger.getLogger(DAOModa.class.getName()).log(Level.SEVERE, null, ex);
        }
        entity.setStatus(Response.Status.NO_CONTENT.getStatusCode());
        return Response.status(Response.Status.NO_CONTENT).entity(entity).build();
    } else {
        entity.setStatus(Response.Status.NOT_FOUND.getStatusCode());
        entity.setMensaje("El usuario ya existe");
        return Response.status(Response.Status.BAD_REQUEST).entity(entity).build();
    }
}

```

Para agregar el usuario, creamos una clase ResponseEntity, para responder al cliente que se ha agregado o no se ha agregado.

```

import java.io.Serializable;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

@XmlRootElement(name="response")
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(propOrder={"status", "mensaje"})
public class ResponseEntity implements Serializable {
    @XmlElement int status;
    @XmlElement String mensaje;

    public ResponseEntity() {
    }

    public ResponseEntity(int status, String mensaje) {
        this.status = status;
        this.mensaje = mensaje;
    }
}

```

Creamos los getters y setter. Y ya que podemos ver en addUsuarios que devolvemos el Response con un mensaje y el status que nos devuelve la Api.

Como podemos fijar hemos utilizado un método en addUsuario para sacar el tamaño de la tabla, pasando el nombre de la tabla.

```
public static int getMaxIdTabla(String nombreTabla) {  
  
    int maximo = 0;  
    try {  
  
        // Crear Statement de la Consulta  
        String sentenciaSQL = "SELECT max(id) AS 'idMax' FROM "+nombreTabla+";";  
        Statement statement = con.createStatement();  
  
        // ResultSet  
        ResultSet rs = statement.executeQuery(sentenciaSQL);  
        if (rs.next()) {  
            // Cargar valores  
            maximo = rs.getInt("idMax");  
        }  
        rs.close();  
  
    } catch (SQLException ex) {  
        Logger.getLogger(DAOModa.class.getName()).log(Level.SEVERE, null, ex);  
    }  
    return maximo;  
}
```

Pues ahora hacemos lo mismo con la tabla producto. Solo que la tabla producto tendrá un método más que es borrar para que el administrador pueda borrar el producto que quiere.

```
public static ArrayList<Producto> getProductos() {  
  
    Producto objeto = null;  
    ArrayList<Producto> productos = new ArrayList<>();  
    try {  
  
        // Crear Statement de la Consulta  
        String sentenciaSQL = "SELECT * FROM productos";  
        Statement statement = con.createStatement();  
  
        // ResultSet  
        ResultSet rs = statement.executeQuery(sentenciaSQL);  
        while (rs.next()) {  
            // Cargar valores  
            objeto = new Producto();  
            objeto.setId(rs.getInt("id"));  
            objeto.setNombre(rs.getString("nombre"));  
            objeto.setDescripcion(rs.getString("descripcion"));  
            objeto.setPrecio(rs.getDouble("precio"));  
            objeto.setStock(rs.getInt("stock"));  
            objeto.setImagen(rs.getString("imagen"));  
  
            productos.add(objeto);  
        }  
        rs.close();  
  
        if (productos == null) {  
            System.out.println("No existe ningún objeto");  
        }  
    } catch (SQLException ex) {
```

```
public static Response EliminarProducto(int idProducto) {
    ResponseEntity entity = new ResponseEntity();
    boolean encontrado = false;
    for (Producto p : getProductos()) {
        if (p.getId() == idProducto)
        {
            encontrado = true;
            break;
        }
    }
    if (encontrado) {
        try {
            String insertTableSQL = "DELETE FROM productos "
                + "WHERE id=" + idProducto;
            PreparedStatement preparedStatement = con.prepareStatement(insertTableSQL);
            preparedStatement.executeUpdate();

        } catch (SQLException ex) {
            Logger.getLogger(DAOModa.class.getName()).log(Level.SEVERE, null, ex);
        }
        entity.setStatus(Response.Status.NO_CONTENT.getStatusCode());
        return Response.status(Response.Status.NO_CONTENT).entity(entity).build();
    } else {
        entity.setStatus(Response.Status.NOT_FOUND.getStatusCode());
        entity.setMensaje("El Producto no existe");
        return Response.status(Response.Status.BAD_REQUEST).entity(entity).build();
    }
}
```

4.2. Utilizar el Servicio ApiRest desde el proyecto de Android Studio

primero de todos creamos un proyecto nuevo y empezamos crear nuestra api en android studio.

Creamos una interfaz ApiRest que nos va a dar el servicio ApiRest. En los que vamos a colocar nuestro métodos para obtener los datos de la ApiRest que hemos creado en Netbeans.


```
public interface ApiRestService {

    @GET("usuarios")
    Call<List<Usuario>> getUsuarios();

    @POST("add_usuario")
    @Headers({"Accept: application/json", "Content-Type: application/json"})
    Call<RespuestaJson> insertarUsuario(@Body Usuario user);

    @PUT("actualizar_usuario")
    @Headers({"Accept: application/json", "Content-Type: application/json"})
    Call<RespuestaJson> actualizarUsuario(@Body Usuario user);

    @GET("productos")
    Call<List<Producto>> getProductos();

    @POST("add_producto")
    @Headers({"Accept: application/json", "Content-Type: application/json"})
    Call<RespuestaJson> insertarProducto(@Body Producto producto);

    @PUT("actualizar_producto/{id}")
    @Headers({"Accept: application/json", "Content-Type: application/json"})
    Call<RespuestaJson> actualizarProducto(@Path("id") int id, @Body Producto producto);

    @DELETE("eliminar_producto/{id}")
    @Headers({"Accept: application/json", "Content-Type: application/json"})
    Call<RespuestaJson> eliminarProducto(@Path("id") int id);

}
```

Creamos las variables Globales en MainActivity.java

```
public ApiRestService apiRestService = null;
public String ipGlobal = "192.168.43.254";
```

Creamos los dos clases Usuario y Producto como hemos creado en Netbeans. Y creamos las listas en los que vamos a obtener los datos.

```
public ArrayList<Usuario> usuarios = new ArrayList<>();
public ArrayList<Producto> productos = new ArrayList<>();
```

Y creamos los métodos para Cada uno de los métodos que tenemos en Nuestro servicio ApiRest.

Como el nombre de cada método explica lo que hace, por eso no hace falta las explicaciones para decir cada método lo que hace.

Y los métodos crearemos en la misma activity principal.


```

public void getUsuarios() {
    usuarios.clear();
    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://" + ipGlobal + ":8081/recursos/apirest/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
    apiRestService = retrofit.create(ApiRestService.class);
    Call<List<Usuario>> call = apiRestService.getUsuarios();

    call.enqueue(new Callback<List<Usuario>>() {
        @Override
        public void onResponse(Call<List<Usuario>> call, Response<List<Usuario>> response) {
            for(Usuario u : response.body()) {
                usuarios.add(u);
            }
        }
        @Override
        public void onFailure(Call<List<Usuario>> call, Throwable t) {
            System.out.println("Fallado");
        }
    });
}

```

```

public void ActualizarUsuario(Usuario u, final Context context) {

    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://" + ipGlobal + ":8081/recursos/apirest/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
    apiRestService = retrofit.create(ApiRestService.class);

    apiRestService.actualizarUsuario(u).enqueue(new Callback<RespuestaJson>() {
        @Override
        public void onResponse(Call<RespuestaJson> call, Response<RespuestaJson> response) {
            if(response.isSuccessful()){
                Toast.makeText(context, text: "Se ha actualizado el usuario", Toast.LENGTH_SHORT).show();
            }
            else {
                Toast.makeText(context, text: "No se ha podido actualizar el usuario", Toast.LENGTH_SHORT).s
            }
        }

        @Override
        public void onFailure(Call<RespuestaJson> call, Throwable t) {

        }
    });
}

```

En getProductos he rellenado el arrayList como si el stock esta a "0" no lo agregue a la lista, pero si el usuario logeado es Administrador, que si que agregue para que el administrador pueda modificar los productos con la lógica de sus funciones.

```

public void getProductos() {
    productos.clear();
    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://"+ipGlobal+":8081/recursos/apirest/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
    apiRestService = retrofit.create(ApiRestService.class);
    Call<List<Producto>> call = apiRestService.getProductos();
    call.enqueue(new Callback<List<Producto>>() {
        @Override
        public void onResponse(Call<List<Producto>> call, Response<List<Producto>> response) {
            for(Producto producto : response.body()) {
                try {
                    charsetUTF8 = new String((producto.getNombre()).getBytes(charsetName: "UTF-8"), charsetName: "ISO-8859-1");
                    producto.setNombre(charsetUTF8);
                    charsetUTF8 = new String((producto.getDescripcion()).getBytes(charsetName: "UTF-8"), charsetName: "ISO-8859-1");
                    producto.setDescripcion(charsetUTF8);
                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();
                }
                if(producto.getStock() >= 1) {
                    productos.add(producto);
                } else {
                    if(usuarioLogeado.getId() == 0) {
                        productos.add(producto);
                    }
                }
            }
        }
        @Override
        public void onFailure(Call<List<Producto>> call, Throwable t) {
            System.out.println("Fallado");
        }
    });
}

```

```

public static void InsertarProducto(Producto p, final Context context) {
    ((MainActivity)context).apiRestService.insertarProducto(p).enqueue(new Callback<RespuestaJson>() {
        @Override
        public void onResponse(Call<RespuestaJson> call, Response<RespuestaJson> response) {
            if(response.isSuccessful()){
                Toast.makeText(context, text: "Se ha insertado el producto", Toast.LENGTH_SHORT).show();
            }
            else {
                Toast.makeText(context, text: "No se ha podido insertar el producto", Toast.LENGTH_SHORT).show();
            }
        }
        @Override
        public void onFailure(Call<RespuestaJson> call, Throwable t) {
        }
    });
}
}

```

```
public static void ActualizarProducto(Producto p, final Context context) {  
  
    ((MainActivity) context).apiRestService.actualizarProducto(p.getId(), p).enqueue(new Callback<RespuestaJson>() {  
        @Override  
        public void onResponse(Call<RespuestaJson> call, Response<RespuestaJson> response) {  
            if(response.isSuccessful()){  
                Toast.makeText(context, text: "Se ha actualizado el producto", Toast.LENGTH_SHORT).show();  
            }  
            else {  
                Toast.makeText(context, text: "No se ha podido actualizar el producto", Toast.LENGTH_SHORT).show();  
            }  
        }  
  
        @Override  
        public void onFailure(Call<RespuestaJson> call, Throwable t) {  
  
        }  
  
    });  
}
```

```
public static void EliminarProducto(int idProducto, final Context context) {  
  
    ((MainActivity) context).apiRestService.eliminarProducto(idProducto).enqueue(new Callback<RespuestaJson>() {  
        @Override  
        public void onResponse(Call<RespuestaJson> call, Response<RespuestaJson> response) {  
            if(response.isSuccessful()){  
                Toast.makeText(context, text: "Se ha eliminado el producto", Toast.LENGTH_SHORT).show();  
            }  
            else {  
                Toast.makeText(context, text: "No se ha podido eliminar el producto", Toast.LENGTH_SHORT).show();  
            }  
        }  
  
        @Override  
        public void onFailure(Call<RespuestaJson> call, Throwable t) {  
  
        }  
  
    });  
}
```

5.1. Logeo

En el Main Activity Creamos el Logeo del usuario. Primero de todos nos hace falta un diseño que crearemos con layout, y implementaremos la lógica para el logeo de una persona.



The screenshot shows a mobile application interface for 'Mi Moda'. At the top, there is a logo featuring a stylized yellow eye with a purple outline and the text 'Mi Moda' in purple. Below the logo, there are two input fields: 'Usuario' and 'Contraseña', both with red labels. A red error message 'Usuario' is visible above the first field. Below the input fields is a blue button with the text 'INICIAR'. At the bottom, there is a link that says 'No Tienes la cuenta aun? Create una!'. The entire interface is enclosed in a white border with a light gray background.

Así será más o menos, desde la vista.

El botón para logear y el texto para registrar, al darle al texto nos lleva a otro fragment de Registro que ahora vamos a hablar sobre el registro.

Pero primero hablamos de la implementación de MainActivity. Vamos a ver.

YA hemos creado algunas variable ya, pero al final quedaremos con estas variables en el MainActivity.

```
public ArrayList<Usuario> usuarios = new ArrayList<>();  
public ArrayList<Producto> productos = new ArrayList<>();  
  
public ArrayList<Producto> carrito = new ArrayList<>();  
public double precioTotalCarrito = 0;  
public Map<Integer, Integer> cantidadConLaIdCarrito = new HashMap<>();  
public Map<Integer, String> idProductoConValorRadioButton = new HashMap<>();  
  
public ApiRestService apiRestService = null;  
public String ipGlobal = "192.168.43.254";  
public Usuario usuarioLogeado;  
  
EditText etUsername;  
EditText etPassword;  
Button bLogin;  
TextView tvRegistrar;
```

(Creo que no podemos entender la funcion que tiene cada variable con los nombres que he puesto)

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    getSupportActionBar().hide();  
    getUsuarios();  
  
    etUsername = findViewById(R.id.username);  
    etPassword = findViewById(R.id.password);  
    bLogin = findViewById(R.id.buttonlogin);  
    tvRegistrar = findViewById(R.id.registrar);  
  
    bLogin.setOnClickListener((v) -> {  
        for(Usuario user : usuarios) {  
            if(user.getUsuario().equals(etUsername.getText().toString())) {  
                if(user.getContraseña().equals(etPassword.getText().toString())) {  
                    setContentView(R.layout.fragment_container_logeado);  
                    usuarioLogeado = user;  
                    getProductos();  
                    try {  
                        Thread.sleep( millis: 2000);  
                    } catch (InterruptedException e) {  
                        e.printStackTrace();  
                    }  
                    CreaFragmento(new MiModaPrincipal(user, context: MainActivity.this), R.id.fragment_container_logeado);  
                    break;  
                } else {  
                    Toast.makeText( context: MainActivity.this, text: "La contraseña no es correcta", Toast.LENGTH_SHORT).show();  
                }  
            } else {  
                Toast.makeText( context: MainActivity.this, text: "El Usuario no existe!", Toast.LENGTH_SHORT).show();  
            }  
        }  
    });  
}
```

obtengo los datos de los usuarios de la base de datos en una arrayList. Y aplicando la lógica logeando o mostrando el mensaje al usuario. Me hago un Sleep para obtener los productos desde la base de datos. Para que me rellene la lista de los productos primero y luego me cargue el fragmento.

Y para el registro tengo el fragmento y layout creado. Y se lanzará de esta manera.

```
tvRegistrar.setOnClickListener((v) -> {  
    setContentView(R.layout.fragment_container);  
    CreaFragmento(new RegistrarFragment( context: MainActivity.this), R.id.fragment_container);  
});
```

He creado en el MainActivity, algunas funciones que me pueden necesitar llamar para facilitar me utilizar desde el contexto principal.

```

public void CreaFragmento(Fragment fragmento, int id) {

    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

    fragmentTransaction.replace(id, fragmento);
    fragmentTransaction.commit();

}

```

Primero tengo CreaFragmento, para reemplazar el fragmento en el que estamos. Con la id del container que le queremos poner.

Porque tengo dos contenedores, uno para logeado y otro para no logeado, mejor dicho para registrar.

```

public void BoraFragmento(int id) {
    getSupportFragmentManager().beginTransaction().
        remove(getSupportFragmentManager().findFragmentById(id)).commit();
}

```

En cualquier lugar puedo necesitar Borrar el fragmento actual.

```

public static String BitmapAString(Bitmap imagen) {
    try {
        ByteArrayOutputStream stream = new ByteArrayOutputStream();
        imagen.compress(Bitmap.CompressFormat.PNG, 90, stream);
        byte[] byte_arr = stream.toByteArray();
        String image_str = Base64.encodeToString(byte_arr, Base64.DEFAULT);
        return image_str;
    } catch (Exception ex) {
        return "";
    }
}

public static Bitmap StringABitmap(String imagen) {
    byte[] decodedString = Base64.decode(imagen, Base64.DEFAULT);
    return BitmapFactory.decodeByteArray(decodedString, 0, decodedString.length);
}

```

Pasar las imagenes Bitmap a cadena, y cadena a bitmap.

```

public static RoundedBitmapDrawable CircularLaImagen(Bitmap imagenBitmap, Context context) {
    float ImageRadius = 2000.0f;
    RoundedBitmapDrawable RBD = RoundedBitmapDrawableFactory.create(((MainActivity) context).getResources(), imagenBitmap);
    RBD.setCornerRadius(ImageRadius);
    RBD.setCircular(true);
    //RBD.setAntiAlias(true);
    return RBD;
}

```

Este método sirve para circular la imagen bitmap.


```
public void PopBackFragment(View rootView, final Fragment fragmentAnterior) {  
    rootView.setFocusableInTouchMode(true);  
    rootView.requestFocus();  
    rootView.setOnKeyListener((v, keyCode, event) -> {  
        if( keyCode == KeyEvent.KEYCODE_BACK && event.getAction() == KeyEvent.ACTION_UP) {  
            //OnBackPressed  
            AbrirFragmentOActivityPrincipal(fragmentAnterior);  
            return true;  
        } else {  
            MainActivity.this.CreaFragmento(fragmentAnterior, R.id.fragment_container_logeado);  
        }  
        return false;  
    });  
}
```

Para quitar abrir otro fragmento. O la Actividad Principal, si se Cierra la sesión por Ejemplo.

```
public void AbrirFragmentOActivityPrincipal(Fragment fragmentAnterior) {  
    getFragmentManager().popBackStack( name: null, FragmentManager.POP_BACK_STACK_INCLUSIVE);  
    if(fragmentAnterior == null) {  
        Intent intent = new Intent( packageContext: MainActivity.this, MainActivity.class);  
        startActivity(intent);  
    } else {  
        MainActivity.this.CreaFragmento(fragmentAnterior, R.id.fragment_container_logeado);  
    }  
}
```

Para abrir el fragmento, y si el fragmento es null, se me cierra la sesión para entrar en MainActivity y empezar con un usuario nuevo.

```

public static void DialogoParaElegirImagen(Context context, final Fragment fragment) {
    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    builder.setMessage("Elige la opción")
        .setPositiveButton("Ir a la Galería", (dialog, id) -> {
            try{
                PerfilFragment perfilFragment = (PerfilFragment) fragment;
                perfilFragment.elegirImagen();
            } catch (Exception e){
                try{
                    AgregarProductoFragment agregarProductoFragment = (AgregarProductoFragment) fragment;
                    agregarProductoFragment.elegirImagen();
                } catch (Exception ex){
                    EditarProductoFragment editarProductoFragment = (EditarProductoFragment) fragment;
                    editarProductoFragment.elegirImagen();
                }
            }
        })
        .setNegativeButton("Tomar Foto", (dialog, id) -> {
            try{
                PerfilFragment perfilFragment = (PerfilFragment) fragment;
                perfilFragment.tomarFoto();
            } catch (Exception e){
                try{
                    AgregarProductoFragment agregarProductoFragment = (AgregarProductoFragment) fragment;
                    agregarProductoFragment.tomarFoto();
                } catch (Exception ex){
                    EditarProductoFragment editarProductoFragment = (EditarProductoFragment) fragment;
                    editarProductoFragment.tomarFoto();
                }
            }
        })
    builder.show();
}

```

Este Método es para sacar un dialogo para ir a la galería, o tomar Foto con la cámara.

```

@Override
public void onBackPressed() {
    finishAffinity();
}

```

Para cerrar la aplicación.

Vale, ya hemos visto varios métodos y ya sabemos que es lo que hace cada uno. Ahora voy a mostrar el registro. Y luego veremos donde vamos a utilizar cada uno de los métodos que hemos creado en la actividad Principal.

5.2. Registro

La vista va a ser como así.



Nombre

Apellidos

Usuario

Contraseña

REGISTRAR

Al registrar nos agrega el usuario en la base de datos. Y ya podremos iniciar la sesión con el usuario que registramos.

Y nos volvemos a la Actividad principal para logear la sesión.

Para la lógica tenemos que implementar lo todo eso, que funcione... vamos a crear el fragmento y aplicaremos la lógica para que funcione como queremos.


```
public class RegistrarFragment extends Fragment {

    private Context contexto;
    private ImageView logo;
    private EditText nombre;
    private EditText apellidos;
    private EditText usuario;
    private EditText contraseña;
    private Button registrar;
```

Pasaremos el contexto.

```
public RegistrarFragment(Context c) {
    this.contexto = c;
}
```

```
public View onCreateView(LayoutInflater inflater, final ViewGroup container, Bundle savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    View rootView = inflater.inflate(R.layout.activity_registrar, container, attachToRoot: false);

    logo = rootView.findViewById(R.id.imagenPerfil);
    nombre = rootView.findViewById(R.id.cantidad);
    apellidos = rootView.findViewById(R.id.apellidos);
    usuario = rootView.findViewById(R.id.username);
    contraseña = rootView.findViewById(R.id.password);
    registrar = rootView.findViewById(R.id.buttonRegistrar);
```

```
registrar.setOnClickListener(v -> {
    Usuario u = new Usuario();
    u.setImagen("");
    u.setNombre(nombre.getText().toString());
    u.setApellidos(apellidos.getText().toString());
    u.setUsuario(usuario.getText().toString());
    u.setContraseña(contraseña.getText().toString());

    if(u.getUsuario().equals("")) {
        Toast.makeText(getActivity(), "El usuario es obligatorio!", Toast.LENGTH_SHORT).show();
    } else {
        boolean puedoVolver = true;
        for (Usuario usuario : ((MainActivity)contexto).usuarios) {
            if(usuario.getUsuario().equals(u.getUsuario())) {
                puedoVolver = false;
                break;
            }
        }
        if(puedoVolver) {
            MainActivity.insertarUsuario(u, contexto);
            ((MainActivity)contexto).AbrirFragmentOActividadPrincipal(fragmentAnterior: null);
        } else {
            Toast.makeText(contexto, "Lo sentimos! El usuario ya esta elegido por alguien", Toast.LENGTH_SHORT).show();
        }
    }
});
```

Botón Registrar, puedo poner más obligaciones para el usuario. Pero he puesto lo fácil para los usuarios, que el usuario con el que va a loguear, tiene que ser distinto que todos los demás.

```
rootView.setFocusableInTouchMode(true);
rootView.requestFocus();
rootView.setOnKeyListener((v, keyCode, event) -> {
    if( keyCode == KeyEvent.KEYCODE_BACK && event.getAction() == KeyEvent.ACTION_UP) {
        //OnBackPressed
        ((MainActivity)contexto).AbrirFragmentOActivityPrincipal( fragmentAnterior: null);
        return true;
    }
    return false;
});
return rootView;
```

Retorno la vista.

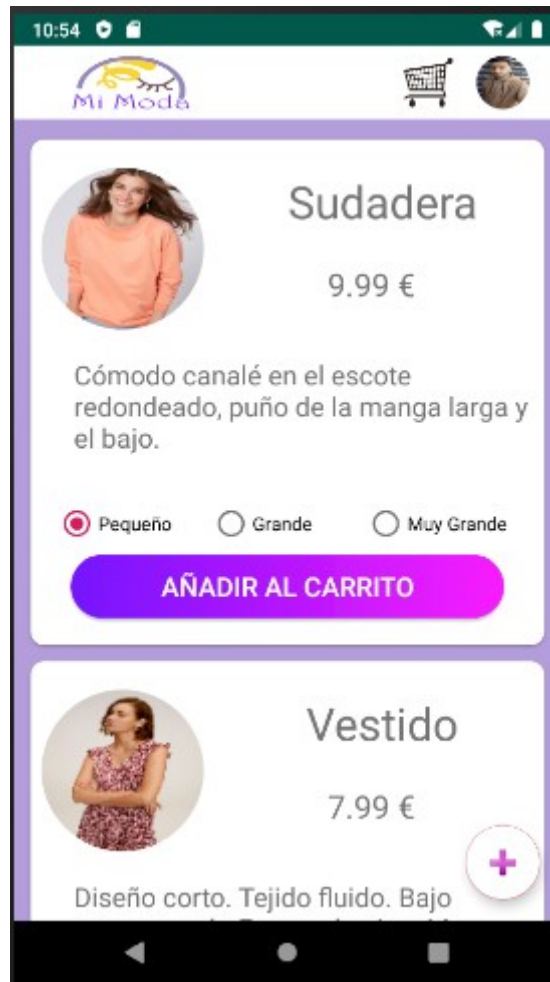
Y dentro creo para que si en el fragmento, el usuario le da al botón de atrás, que no se me cierre la aplicación, que se me abra la actividad de logueo para iniciar la sesión.

Si ya tiene la cuenta.

Nada Más, ya vamos por la vista principal que hemos creado. El grosor del proyecto.

5.3. Página Principal

Así es la vista que tendremos en la vista principal.



Vamos ella a implementarlo.

- Primero de todos necesitamos crear un layout que tenga el toolbar y el recycler view y FAB.
- Segundo tenemos que crear el layout del cardview.
- Tercero crear el adaptador y el holder para el recyclerView.
- Cuarto aplicar la lógica para que funcione como lo que queremos que funcione.

Vamos desde la primera parte.

Así se quedará la vista



Es un Toolbar en el que tiene 2 imagenes, una para perfil y otra para carrito, a los que hablaremos más adelante.

RecyclerView Normal, y FAB. Podemos cambiar los colores etc. y hacer lo que queramos para que tenga un buen aspecto.

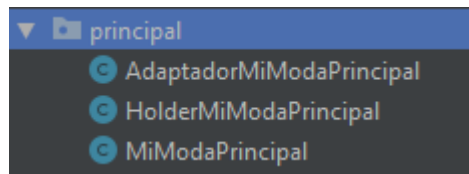
Segundo dije que crear el cardView

Crearemos otra layout para carView.



Imagen, text views, Radio Buttones y Button.

Y en el código java, crearemos el fragmento, y aplicaremos la lógica que queremos que funcione para este fragmento.



Tengo creado 3 clases para eso.

Vamos por partes...

- AdaptadorMiModaPrincipal:

```
public void setClickOnView(View.OnClickListener listener) {
    if(listener!=null) this.listener= listener;
}

@Override
public int getItemCount() { return ((MainActivity)c).productos.size(); }

@Override
public void onClick(View view) { if(listener!=null) listener.onClick(view); }

public void setOnLongClick(View.OnLongClickListener longClickListener) {
    if(longClickListener != null) {
        this.longClickListener = longClickListener;
    }
}

@Override
public boolean onLongClick(View view) {
    if(longClickListener!=null) longClickListener.onLongClick(view);
    return false;
}
}
```

```
public class AdaptadorMiModaPrincipal extends RecyclerView.Adapter implements View.OnClickListener, View.OnLongClickListener {

    Context c;
    HolderMiModaPrincipal h;
    View.OnClickListener listener, listenerImagen;
    View.OnLongClickListener longClickListener;

    public AdaptadorMiModaPrincipal(Context c) {
        this.c = c;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.principal_cardview, parent, attachToRoot false);
        view.setOnClickListener(this);
        view.setOnLongClickListener(this);
        h = new HolderMiModaPrincipal(view,c);
        return h;
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
        Producto p = ((MainActivity)c).productos.get(position);
        ((HolderMiModaPrincipal)holder).bind(p, position);
    }
}
```

Es un adaptador que me sirve para obtener los productos y meter los en cada uno de los productos de la lista en el recyclerView android Studio.

- HolderMiModaPrincipal

Mi Holder de Recycler para la lista:

```
public class HolderMiModaPrincipal extends RecyclerView.ViewHolder implements View.OnClickListener {

    Context contexto;
    View itemView;

    TextView precio, descripcion, nombre, stock;
    Producto producto = null;
    Button añadirAlCarrito;
    RadioButton pequeño, grande, muyGrande;
    ImageView imagen;
    View.OnClickListener listener;

    public HolderMiModaPrincipal(View itemView, final Context contexto) {
        super(itemView);
        this.contexto = contexto;
        this.itemView = itemView;
        precio = itemView.findViewById(R.id.precio);
        descripcion = itemView.findViewById(R.id.descripcion);
        nombre = itemView.findViewById(R.id.cantidad);
        stock = itemView.findViewById(R.id.stock);
        imagen = itemView.findViewById(R.id.imagenProductoCarrito);
        añadirAlCarrito = itemView.findViewById(R.id.ButtonAnadirAlCarrito);
        pequeño = itemView.findViewById(R.id.pequeño);
        grande = itemView.findViewById(R.id.grande);
        muyGrande = itemView.findViewById(R.id.muy_grande);
    }
}
```

```
añadirAlCarrito.setOnClickListener((v) -> {
    if(añadirAlCarrito.getText().toString().equalsIgnoreCase("Añadir al carrito")) {
        Toast.makeText(contexto, text "Añadido al carrito", Toast.LENGTH_SHORT).show();
        añadirAlCarrito.setText("Quitar del carrito");
        añadirAlCarrito.setBackgroundColor(Color.LTGRAY);

        ((MainActivity) contexto).carrito.add(producto);
        ((MainActivity) contexto).precioTotalCarrito += producto.getPrecio();
        ((MainActivity) contexto).cantidadConLaIdCarrito.put(producto.getId(), 1);
    } else {
        Toast.makeText(contexto, text "Quitado del carrito", Toast.LENGTH_SHORT).show();
        añadirAlCarrito.setText("Añadir al carrito");
        añadirAlCarrito.setBackground(ContextCompat.getDrawable(contexto, R.drawable.button_btn));

        ((MainActivity) contexto).carrito.remove(producto);
        int cantidad = ((MainActivity) contexto).cantidadConLaIdCarrito.get(producto.getId());
        ((MainActivity) contexto).precioTotalCarrito -= producto.getPrecio() * cantidad;
        ((MainActivity) contexto).cantidadConLaIdCarrito.remove(producto.getId());
    }
});
```

Botón añadir al carrito, se me añade al carrito, cambia el texto y el aspecto del botón. Y quitar del carrito se vuelve a la misma, y con el hasMap de saber la cantidad de los productos del carrito,

sabremos el precio y ponemos en una variable el precio total de la compra para mostrar en el carrito.

```
pequeño.setOnClickListener((v) -> {
    pequeño.setChecked(true);
    ((MainActivity) contexto).idProductoConValorRadioButton.put(producto.getId(), pequeño.getText().toString());
});
grande.setOnClickListener((v) -> {
    grande.setChecked(true);
    ((MainActivity) contexto).idProductoConValorRadioButton.put(producto.getId(), grande.getText().toString());
});
muyGrande.setOnClickListener((v) -> {
    muyGrande.setChecked(true);
    ((MainActivity) contexto).idProductoConValorRadioButton.put(producto.getId(), muyGrande.getText().toString());
});
```

Por ahora no utilizo los radio Botones, más adelante diré sobre ellos. Pero le guardo en una variable para luego si volvemos del carrito, se nos quede al mismo sitio donde nos dejamos.

```
public void bind(Producto producto, int pos){
    this.producto = producto;
    String p = String.format("%.2f", producto.getPrecio());
    precio.setText(p + " €");
    descripcion.setText(producto.getDescripcion());
    nombre.setText(producto.getNombre());
    if(!producto.getImagen().equals("")){
        Bitmap imagenBitmap = MainActivity.StringABitmap(producto.getImagen());
        imagen.setImageDrawable(MainActivity.CircularLaImagen(imagenBitmap, contexto));
    }
    //Comprobando los productos ya añadidos para cuando se vuelve a refrescar:
    for (Producto pro : ((MainActivity) contexto).carrito) {
        if (pro.getId() == producto.getId()) {
            añadirAlCarrito.setText("Quitar del carrito");
            añadirAlCarrito.setBackgroundColor(Color.LTGRAY);
        }
    }
    if(((MainActivity) contexto).idProductoConValorRadioButton.containsKey(producto.getId())) {
        if (((MainActivity) contexto).idProductoConValorRadioButton.get(producto.getId()) == pequeño.getText().toString()) {
            pequeño.setChecked(true);
        } else if (((MainActivity) contexto).idProductoConValorRadioButton.get(producto.getId()) == grande.getText().toString()) {
            grande.setChecked(true);
        } else {
            muyGrande.setChecked(true);
        }
    }
    if(producto.getStock() > 10) {
        stock.setText("");
    } else {
        stock.setText("Quedan solamente " + producto.getStock() + " en Stock");
    }
}
```

Cuando viene un producto desde Adaptador, de la lista de los productos para ponerlo en cardview. Aplicamos la lógica para poner las cosas que queremos que este como queremos. Podemos cambiar la lógica como queremos.

```
@Override
public void onClick(View view) { if (listener != null) listener.onClick(view); }
```

Al final tengo un click listener que si le damos el click al carview. Haga lo que debe de hacer, si esta implementada.

- MiModaPrincipal

```
public class MiModaPrincipal extends Fragment {

    private static long back_pressed;

    private Usuario usuarioLogeado;
    private Context contexto;
    public RecyclerView recyclerView;
    AdaptadorMiModaPrincipal adaptadorMiModaPrincipal;

    ImageView carrito;
    ImageView perfil;
    FloatingActionButton fabAñadir;

    public MiModaPrincipal(Usuario usuario, Context context) {
        usuarioLogeado = usuario;
        contexto = context;
    }

    public View onCreateView(LayoutInflater inflater, final ViewGroup container, Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View rootView = inflater.inflate(R.layout.activity_logeado, container, attachToRoot false);

        recyclerView = rootView.findViewById(R.id.recycler);
        recyclerView.setHasFixedSize(true);
        adaptadorMiModaPrincipal = new AdaptadorMiModaPrincipal(getContext());
        recyclerView.setAdapter(adaptadorMiModaPrincipal);
        recyclerView.setLayoutManager(new GridLayoutManager(getActivity(), spanCount 1));

        carrito = rootView.findViewById(R.id.carrito);
```

Creamos las variable, pasamos el contexto, y el usuario logeado actualmente. Ponemos la cosa del recyclerView para poder realizar la lista con el.


```
carrito = rootView.findViewById(R.id.carrito);
carrito.setOnClickListener((v) -> {
    ((MainActivity)contexto).CreaFragmento(new CarritoFragment(contexto, fragmentAnterior: MiModaPrincipal.this), R.id.fragment_container_logeado);
});

perfil = rootView.findViewById(R.id.perfil);
if(!((MainActivity)contexto).usuarioLogeado.getImagen().equals("")) {
    Bitmap bmp = ((MainActivity)contexto).StringABitmap(((MainActivity)contexto).usuarioLogeado.getImagen());
    perfil.setImageDrawable(MainActivity.CircularLaImagen(bmp, contexto));
}
perfil.setOnClickListener((v) -> {
    ((MainActivity)contexto).CreaFragmento(new PerfilFragment(contexto, usuarioLogeado, fragmentAnterior: MiModaPrincipal.this), R.id.fragment_container_logeado);
});
```

Utilizamos el método desde contexto que nos hace la vida más fácil. Y creamos fragmentos, de los otros nuevos fragmentos, al dar al carrito o al perfil. Estos fragmentos ahora lo explico después de terminar con el principal.

```
fabAñadir = rootView.findViewById(R.id.fabAñadir);
if(usuarioLogeado.getId() == 0) {
    fabAñadir.setOnClickListener((v) -> {
        ((MainActivity)contexto).CreaFragmento(new AgregarProductoFragment(contexto, fragmentAnterior: MiModaPrincipal.this), R.id.fragment_container_logeado);
    });
    adaptadorMiModaPrincipal.setOnLongClick((v) -> {
        final int pos = recyclerView.getChildAdapterPosition(v);
        final Producto p = ((MainActivity)contexto).productos.get(pos);

        AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
        builder.setMessage("¿Seguro que quieres eliminar a " + p.getNombre() + "?");
        builder.setPositiveButton( text "ELIMINAR", (dialog, which) -> {
            MainActivity.EliminarProducto(p.getId(), contexto, p);
            try {
                Thread.sleep( millis: 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            recyclerView.setAdapter(adaptadorMiModaPrincipal);
        });
        builder.setNegativeButton( text "CANCELAR", (dialog, which) -> {
            dialog.cancel();
        });
        builder.create().show();

        return false;
    });
    adaptadorMiModaPrincipal.setClickOnView((v) -> {
        int child = recyclerView.getChildAdapterPosition(v);
        Producto producto = ((MainActivity)contexto).productos.get(child);
        ((MainActivity)contexto).CreaFragmento(new EditarProductoFragment(contexto, fragmentAnterior: MiModaPrincipal.this, producto, child), R.id.fragment_container_logeado);
    });
}
```

Si el usuario tiene la **id “0”** significa que es administrador. Que el puede hacer el click al Cardview, logClick y le aparece el FAB para añadirlo. Si no es, no lo aparece el FAB ni el usuario puede hacer el click sobre cardview y Tampoco LongClick.

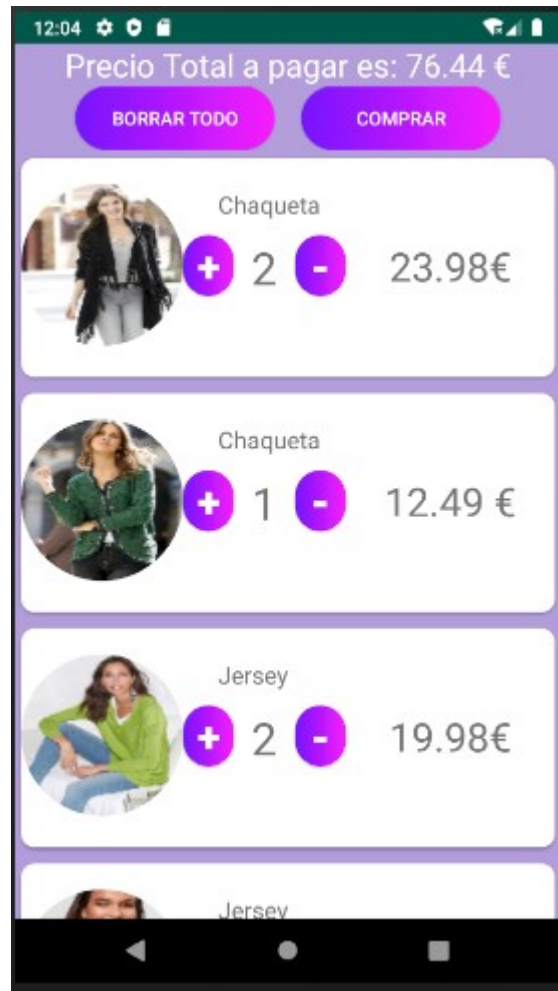
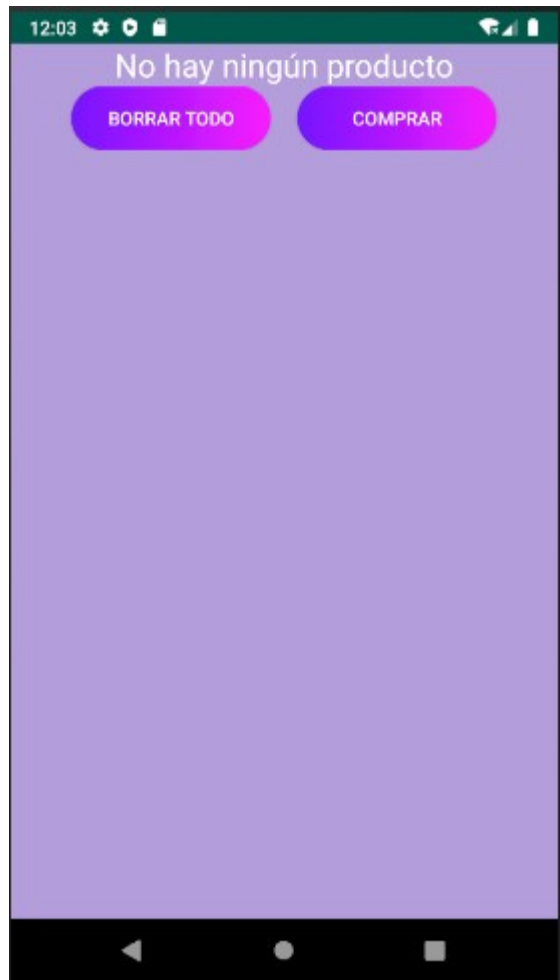
```
});  
} else {  
    fabAñadir.hide();  
}  
  
rootView.setFocusableInTouchMode(true);  
rootView.requestFocus();  
rootView.setOnKeyListener((v, keyCode, event) -> {  
    if( keyCode == KeyEvent.KEYCODE_BACK && event.getAction() == KeyEvent.ACTION_UP) {  
        getFragmentManager().popBackStack( name: null, FragmentManager.POP_BACK_STACK_INCLUSIVE);  
  
        //OnBackPressed  
        if (back_pressed + 1000 > System.currentTimeMillis())  
            (MainActivity)contexto.onBackPressed();  
        else {  
        }  
        back_pressed = System.currentTimeMillis();  
  
        return true;  
    }  
    return false;  
});  
  
return rootView;  
}
```

Oculto fab si el usuario no es administrador.

Y por último si el usuario de el botón de atrás que hay 1 segundo de tiempo para volver a dar para salir de la aplicación, quiere decir habrá que dar dos veces para salir de la aplicación.

5.06. Carrito

Esto es la estructura que tiene el Carrito:



También tiene tres clases.

- Su Adaptador
- Su Holder
- Su Fragmento (Clase principal para el carrito)
- **Adaptador**

```

public class AdaptadorCarrito extends RecyclerView.Adapter implements View.OnClickListener, View.OnLongClickListener {

    Context c;
    Fragment fragmentAnterior;
    HolderCarrito h;
    View.OnClickListener listener, listenerImagen;
    View.OnLongClickListener longClickListener;

    public AdaptadorCarrito(Context c, Fragment fragmentAnterior) {
        this.c = c;
        this.fragmentAnterior = fragmentAnterior;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.carrito_cardview, parent, attachToRoot: false);
        view.setOnClickListener(this);
        view.setOnLongClickListener(this);
        h = new HolderCarrito(view, c);
        return h;
    }
}

```

Pasamos el contexto y el fragmento anterior para Saber de cual fragmento viene porque luego utilizamos la misma clase para ahorrar el código utilizamos con la clase factura. Con el mismo holder ya en esta misma veremos como vamos arreglando con la factura en el holder del carrito.

```

@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
    Producto p = ((MainActivity)c).carrito.get(position);
    String d = String.format("%.2f", p.getPrecio());
    d = d.replace( oldChar: ',', newChar: '.');
    double precio = Double.valueOf(d);
    p.setPrecio(precio);
    ((HolderCarrito)holder).bind(p, fragmentAnterior);
}

public void setClickImage(View.OnClickListener listener) {
    if(listener!=null) listenerImagen= listener;
}

public void setClickOnView(View.OnClickListener listener) {
    if(listener!=null) this.listener= listener;
}

@Override
public int getItemCount() { return ((MainActivity)c).carrito.size(); }

@Override
public void onClick(View view) { if(listener!=null) listener.onClick(view); }

public void SetOnLongClick(View.OnLongClickListener longClickListener) {
    if(longClickListener != null) {
        this.longClickListener = longClickListener;
    }
}

@Override
public boolean onLongClick(View view) {
    if(longClickListener!=null) longClickListener.onLongClick(view);
    return false;
}
}

```

Aplicamos la lógica, sobre el precio que queremos mostrar. Y los click que vamos a realizar. Y vamos al Holder:

- Holder

```
public class HolderCarrito extends RecyclerView.ViewHolder implements View.OnClickListener {

    public static final String PRECIO_TOTAL_A_PAGAR_ES = "Precio Total a pagar es: ";
    Context contexto;
    Fragment fragmentAnterior;

    View itemView;

    ImageView imagenProductoCarrito;
    TextView precio, cantidad, nombreProducto, InformeProducto;
    Button plus, minus;
    Producto producto = null;
    int cantidadProducto = 1;
    double d;

    View.OnClickListener listener;

    public HolderCarrito(View itemView, final Context contexto) {
        super(itemView);
        this.contexto = contexto;
        this.itemView = itemView;
        imagenProductoCarrito = itemView.findViewById(R.id.imagenProductoCarrito);
        precio = itemView.findViewById(R.id.precio);
        cantidad = itemView.findViewById(R.id.cantidad);
        nombreProducto = itemView.findViewById(R.id.nombreProducto);
        InformeProducto = itemView.findViewById(R.id.InformeProducto);
        plus = itemView.findViewById(R.id.plus);
        minus = itemView.findViewById(R.id.minus);
    }
}
```



```
plus.setOnClickListener((v) -> {  
    if(cantidadProducto < producto.getStock()) {  
        cantidadProducto++;  
        ((MainActivity) contexto).cantidadConLaIdCarrito.put(producto.getId(), cantidadProducto);  
  
        CambiarPrecio();  
        ((MainActivity) contexto).precioTotalCarrito += producto.getPrecio();  
        ((CarritoFragment) fragmentAnterior).setTextoPrecio(PRECIO_TOTAL_A_PAGAR_ES + String.format("%.2f", ((MainActivity) contexto).precioTotalCarrito) + " €");  
    } else {  
        Toast.makeText(contexto, text: "No hay más productos en el Stock", Toast.LENGTH_SHORT).show();  
    }  
});
```

si damos click a plus, nos suma un producto del mismo hasta el ultimo número del stock que haya. Y aplicando la lógica cambiando los precios, y arreglando si se necesita cambiar algo del contexto, porque vamos a meter un producto más en el carrito. O mostrando el mensaje, esta lógica se puede cambiar si queremos con otra lógica.

```
minus.setOnClickListener((v) -> {  
    if(cantidadProducto > 1) {  
        cantidadProducto--;  
        ((MainActivity) contexto).cantidadConLaIdCarrito.put(producto.getId(), cantidadProducto);  
  
        ((MainActivity) contexto).precioTotalCarrito -= producto.getPrecio();  
        ((CarritoFragment) fragmentAnterior).setTextoPrecio(PRECIO_TOTAL_A_PAGAR_ES + String.format("%.2f", ((MainActivity) contexto).precioTotalCarrito) + " €");  
  
        CambiarPrecio();  
    } else {  
    }
```

Para restar el producto, tenemos el botón restar. Con el que podamos restar hasta “1” el producto.

```
    } else {  
        final AlertDialog.Builder builder = new AlertDialog.Builder(contexto);  
        builder.setMessage("Estás seguro que quieres quitar el producto!");  
        builder.setCancelable(false);  
        builder.setPositiveButton("Quitar", (dialog, id) -> {  
            for (Producto p : ((MainActivity) contexto).carrito) {  
                if (p.getId() == producto.getId()) {  
                    ((MainActivity) contexto).cantidadConLaIdCarrito.remove(producto.getId());  
                    ((MainActivity) contexto).precioTotalCarrito -= producto.getPrecio();  
                    ((CarritoFragment) fragmentAnterior).QuitarProducto();  
                    ((MainActivity) contexto).carrito.remove(p);  
  
                    if (((MainActivity) contexto).carrito.size() == 0)  
                        ((CarritoFragment) fragmentAnterior).setTextoPrecio("No hay ningún producto");  
                    break;  
                }  
            }  
        });  
        builder.setNegativeButton("Cancelar", (dialog, id) -> {  
            dialog.cancel();  
        });  
        AlertDialog alert = builder.create();  
        alert.show();  
    }  
};  
}
```

Si el producto es 1 y damos al botón de restar, se no aparece el dialogo preguntando si nos queremos eliminar o no. Siempre aplicamos la lógica que tenemos por los precios, y la cantidad de los productos que haya elegido.

```

public void bind(Producto producto, Fragment fragmentAnterior){
    this.producto = producto;
    if(!producto.getImagen().equals("")){
        Bitmap imagenBitmap = MainActivity.StringABitmap(producto.getImagen());
        imagenProductoCarrito.setImageDrawable(MainActivity.CircularLaImagen(imagenBitmap, contexto));
    }
    String p = String.format("%.2f", producto.getPrecio() * ((MainActivity)contexto).cantidadConLaIdCarrito.get(producto.getId()));
    precio.setText(p + " €");

    if(((MainActivity) contexto).cantidadConLaIdCarrito.containsKey(producto.getId())) {
        cantidadProducto = ((MainActivity) contexto).cantidadConLaIdCarrito.get(producto.getId()).intValue();
    }
    cantidad.setText(String.valueOf(cantidadProducto));

    nombreProducto.setText(producto.getNombre());
    InformeProducto.setTextColor(Color.RED);
    if(producto.getStock() > 10) {
        InformeProducto.setText("");
    } else {
        InformeProducto.setText("Quedan " + producto.getStock() + " productos en Stock...");
    }

    this.fragmentAnterior = fragmentAnterior;
}

```

Cuando entramos co cada Producto desde adaptador, hay que rellenar los datos para mostrarlo en el cardview del carrito o de la factura, según el fragmento en el que estemos.

```

try {
    if(((CarritoFragment) fragmentAnterior).carrito) {
    }
}
catch (Exception e) {
    InformeProducto.setVisibility(View.INVISIBLE);
    plus.setVisibility(View.INVISIBLE);
    minus.setVisibility(View.INVISIBLE);
    precio.setText(String.format("%.2f", cantidadProducto * producto.getPrecio()) + "€");
}
}

```

No encontrado otra forma más normal de hacerlo, con “try catch” se me lanza un excepción si el fragmento en el que estemos no es del carrito, y si no es del carrito ponemos algunas funciones invisibles, porque en la factura no nos vamos a necesitar.

```
@Override
public void onClick(View view) { if (listener != null) listener.onClick(view); }

private void CambiarPrecio() {
    cantidad.setText(String.valueOf(cantidadProducto));
    d = (producto.getPrecio()*cantidadProducto);
    String p = String.format("%.2f", d);
    precio.setText(p + "€");
}
```

Y unas funciones que nos necesitamos para cumplir el código escrito.

Tener en cuenta en todos lugares aplicar la lógica sobre los precios y la cantidad de los productos. Porque si se cambia algún punto, se pierde la lógica y no se cuadra bien con los precios.

- Fragmento

```
public class CarritoFragment extends Fragment {

    public static final String PRECIO_TOTAL_A_PAGAR_ES = "Precio Total a pagar es: ";
    private String textoPorDefecto = "No hay ningún producto";
    MiModaPrincipal fragmentAnterior;
    Context contexto;

    TextView precioTotal;
    Button borrarTodo, comprar;
    boolean carrito;

    private RecyclerView recyclerView = null;
    AdaptadorCarrito adaptadorCarrito = null;

    public CarritoFragment() {

    }

    public CarritoFragment(Context c, MiModaPrincipal fragmentAnterior) {
        this.contexto = c;
        this.fragmentAnterior = fragmentAnterior;
        carrito = true;
    }
}
```

```
public View onCreateView(LayoutInflater inflater, final ViewGroup container, Bundle savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    final View rootView = inflater.inflate(R.layout.activity_carrito, container, attachToRoot: false);

    precioTotal = rootView.findViewById(R.id.precioTotal);
    borrarTodo = rootView.findViewById(R.id.borrarTodo);
    comprar = rootView.findViewById(R.id.comprar);
}
```

```
if (((MainActivity)contexto).carrito.size() == 0) {
    precioTotal.setText(textoPorDefecto);
    Toast.makeText(contexto, text: "Todavía no hay ningún producto seleccionado", Toast.LENGTH_SHORT).show();
} else {
    precioTotal.setText(PRECIO_TOTAL_A_PAGAR_ES + String.format("%.2f", ((MainActivity)contexto).precioTotalCarrito) + "€");

    borrarTodo.setOnClickListener((v) -> {
        ((MainActivity)contexto).cantidadConLaIdCarrito.clear();
        ((MainActivity)contexto).carrito.clear();
        ((MainActivity)contexto).precioTotalCarrito = 0;
        precioTotal.setText(textoPorDefecto);
        recyclerView.setAdapter(adaptadorCarrito);
    });

    comprar.setOnClickListener((v) -> {
        if (((MainActivity)contexto).carrito.size() > 0) {
            for (Producto pCarrito : ((MainActivity)contexto).carrito) {
                Integer productoActualComprado = ((MainActivity)contexto).cantidadConLaIdCarrito.get(pCarrito.getId());
                pCarrito.setStock(pCarrito.getStock() - productoActualComprado);
                MainActivity.ActualizarProducto(pCarrito, contexto, index: -1);
                if (pCarrito.getStock() <= 0) {
                    ((MainActivity)contexto).productos.remove(pCarrito);
                }
            }
            ((MainActivity)contexto).CreaFragmento(new FacturaFragment(contexto, fragmentAnterior), R.id.fragment_container_logeado);
        } else {
            Toast.makeText(contexto, text: "No hay ningún producto para comprar", Toast.LENGTH_SHORT).show();
        }
    });
}
```

Siempre estaremos aplicando la lógica, por si queremos cambiar el texto del precio, al dar al botón de borrar, borraremos todo lo que tenemos en el carrito, y todo relacionado con el carrito. Y al darle al botón de comprar, vamos a la factura Fragmento, siempre aplicando la lógica que tenemos con el carrito. No hay que perder la lógica de los productos del carrito. Y si no hay productos simplemente nos muestra un mensaje de que no hay productos.


```
recyclerView = rootView.findViewById(R.id.recycler);
recyclerView.setHasFixedSize(true);
adaptadorCarrito = new AdaptadorCarrito(getContext(), fragmentAnterior, this);
recyclerView.setAdapter(adaptadorCarrito);
recyclerView.setLayoutManager(new GridLayoutManager(getActivity(), 1));
adaptadorCarrito.setOnClickListener((v) -> {
    //Toast.makeText(contexto, "Click", Toast.LENGTH_SHORT).show();
});

}

(MainActivity)contexto).PopBackFragment(rootView, fragmentAnterior);

return rootView;
}

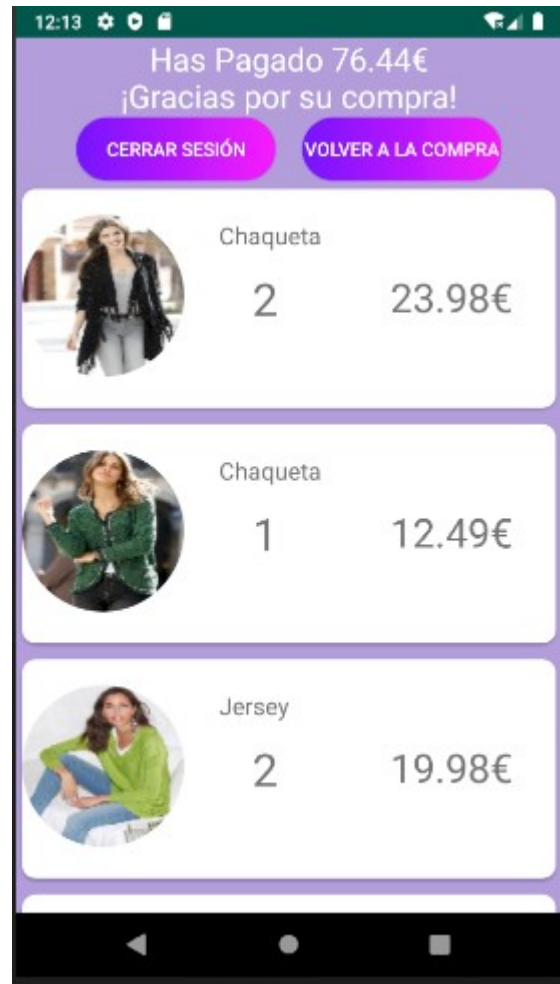
public void QuitarProducto() {
    if ((MainActivity)contexto).carrito.size() == 0) {
        borrarTodo.callOnClick();
    } else {
        precioTotal.setText(PRECIO_TOTAL_A PAGAR ES + String.format("%.2f", ((MainActivity)contexto).precioTotalCarrito) + "€");
    }
    recyclerView.setAdapter(adaptadorCarrito);
}

public void setTextoPrecio(String precio) { precioTotal.setText(precio); }
```

Implementando RecyclerView, porque sin este no va a funcionar nada. Y al final el método popBack que dije en el MainActivity para quitar el fragmento actual para volver al principal. Y los métodos que no pueden interesar para no acumular el código, o duplicar o tener acceso con más facilidad a las variables privados.

5.07. Facturar

y así me aparece la factura, claro no he puesto para comprar el producto, compra con la tarjeta o algo profesional, porque todo está imaginado y no hay nada en nuestro stock no podemos vender, lo mejor dicho esto podemos decir como un demo. Una factura, que ya has comprado y punto. Y así se muestra lo que has comprado:



Dando Click a volver a comprar, se vuelve con todo vacío a la página principal y empezaremos como si vamos a comprar más. O directamente podemos cerrar la sesión.

```
public class FacturaFragment extends Fragment {
    public static final String PRECIO_TOTAL_A_PAGAR_ES = "Precio Total a pagar es: ";
    private String textoPorDefecto = "No hay ningún producto";
    MiModaPrincipal fragmentAnterior;
    Context contexto;

    TextView precioTotal;
    Button salir, volverAComprar;
    boolean factura;

    private RecyclerView recyclerView = null;
    AdaptadorCarrito adaptadorCarrito = null;

    public FacturaFragment() {

    }

    public FacturaFragment(Context c, MiModaPrincipal fragmentAnterior) {
        this.contexto = c;
        this.fragmentAnterior = fragmentAnterior;
        factura = true;
    }

    public View onCreateView(LayoutInflater inflater, final ViewGroup container, Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        final View rootView = inflater.inflate(R.layout.activity_carrito, container, attachToRoot: false);

        precioTotal = rootView.findViewById(R.id.precioTotal);
        salir = rootView.findViewById(R.id.borrarTodo);
        salir.setText("Cerrar Sesión");
        volverAComprar = rootView.findViewById(R.id.comprar);
    }
}
```

El fragmento de Factura va a ser muy similar a la del carrito. Es lo mismo para ahorrar el código, solo que tiene algunas cosas ocultas.

```
volverAComprar.setText("Volver a la compra");

precioTotal.setText("Has Pagado " + String.format("%.2f", ((MainActivity) contexto).precioTotalCarrito)
"\n¡Gracias por su compra!");

salir.setOnClickListener((v) -> {
    ((MainActivity) contexto).BoraFragmento(R.id.fragment_container);

    getActivity().finish();
    Intent intent = new Intent(getActivity(), MainActivity.class);
    startActivity(intent);
});

volverAComprar.setOnClickListener((v) -> {
    ((MainActivity) contexto).carrito.clear();
    ((MainActivity) contexto).precioTotalCarrito = 0;
    ((MainActivity) contexto).cantidadConLaldCarrito.clear();
    ((MainActivity) contexto).CreaFragmento(fragmentAnterior, R.id.fragment_container_logeado);
});

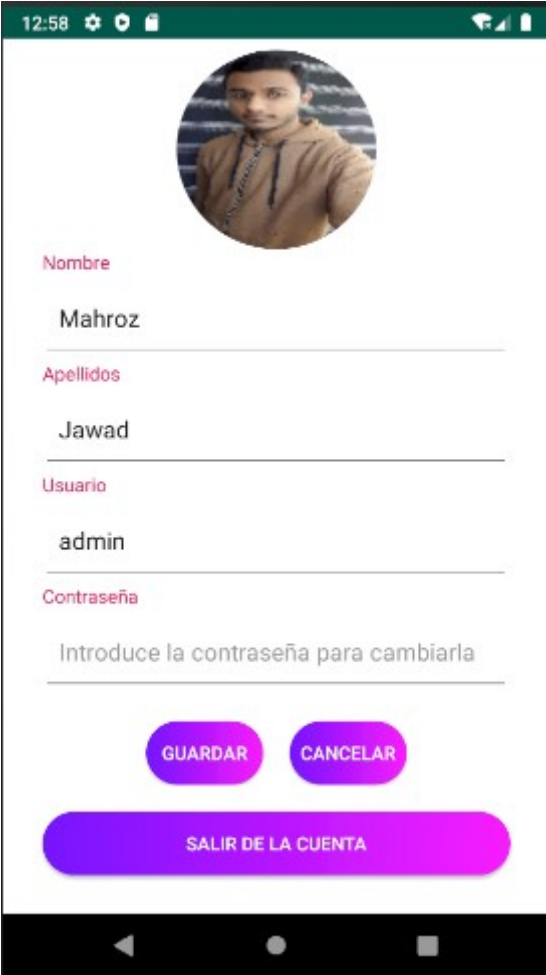
recyclerView = rootView.findViewById(R.id.recycler);
recyclerView.setHasFixedSize(true);
adaptadorCarrito = new AdaptadorCarrito(getContext(), fragmentAnterior, this);
recyclerView.setAdapter(adaptadorCarrito);
recyclerView.setLayoutManager(new GridLayoutManager(getActivity(), spanCount: 1));
adaptadorCarrito.setClickOnView((v) -> {
    //Toast.makeText(contexto, "Click", Toast.LENGTH_SHORT).show();
});

((MainActivity) contexto).PopBackFragment(rootView, fragmentAnterior);
```

Implementamos el recyclerView, Aplicamos la lógica con el precio, y implementamos para cada botón que queremos que haga. Y al final PopBackFragment para quitar el fragmento actual y abrir la principal con el usuario que tenemos logeado.

5.08. Perfil

Así será el aspecto del fragmento perfil:

A screenshot of a mobile application's profile page. At the top, there is a circular profile picture of a man with a beard wearing a brown jacket. Below the picture, the form fields are labeled in red: 'Nombre' (Name) with the value 'Mahroz', 'Apellidos' (Surnames) with the value 'Jawad', 'Usuario' (Username) with the value 'admin', and 'Contraseña' (Password) with the placeholder text 'Introduce la contraseña para cambiarla'. At the bottom of the form, there are three buttons: 'GUARDAR' (Save) and 'CANCELAR' (Cancel) in red rounded rectangles, and a larger blue rounded rectangle button labeled 'SALIR DE LA CUENTA' (Log out). The top status bar shows the time 12:58 and various icons. The bottom navigation bar is black with standard Android icons.

Si pinchamos a la imagen nos muestra el dialogo que tenemos el método en el mainActivity, con lo que traemos la imagen para guardar.
y el código que se implementa para que funcione, como queremos nosotros.

```

public class PerfilFragment extends Fragment {

    private final int COD_ELEGIR_IMAGEN = 1;
    private final int COD_TOMAR_FOTO = 2;

    Context contexto;
    Usuario usuarioLogeado;
    MiModaPrincipal fragmentAnterior;

    private ImageView imagenPerfil;
    private EditText nombre;
    private EditText apellidos;
    private EditText usuario;
    private EditText contraseña;
    private Button guardar;
    private Button cancelar;
    private Button salir;
    String imagenString;

    public PerfilFragment(Context c, Usuario usuarioLogeado, MiModaPrincipal fragmentAnterior) {
        this.contexto = c;
        this.usuarioLogeado = usuarioLogeado;
        this.fragmentAnterior = fragmentAnterior;
    }

    public View onCreateView(LayoutInflater inflater, final ViewGroup container, Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        final View rootView = inflater.inflate(R.layout.activity_perfil, container, attachToRoot: false);
    }

```

```

    imagenPerfil = rootView.findViewById(R.id.imagenPerfil);

    imagenPerfil.setOnClickListener((v) -> {
        MainActivity.DialogoParaElegirImagen(contexto, fragment: PerfilFragment.this);
    });

    nombre = rootView.findViewById(R.id.cantidad);
    apellidos = rootView.findViewById(R.id.apellidos);
    usuario = rootView.findViewById(R.id.username);
    contraseña = rootView.findViewById(R.id.password);

    guardar = rootView.findViewById(R.id.guardar);
    cancelar = rootView.findViewById(R.id.cancelar);
    salir = rootView.findViewById(R.id.salir);

    nombre.setText(usuarioLogeado.getNombre());
    apellidos.setText(usuarioLogeado.getApellidos());

    if(!usuarioLogeado.getImagen().equals("")){
        imagenPerfil.setImageDrawable(MainActivity.CircularLaImagen(((MainActivity)contexto).StringABitmap(usuarioLogeado.getImagen()), contexto));
    }
    else {
        imagenPerfil.setImageResource(R.drawable.por_defecto);
    }

    usuario.setText(usuarioLogeado.getUsuario());

```

Obtendremos los datos del usuario y le mostramos, y si quiere guardar, puede cambiar los datos y cuando se lo de al botón del guardar, se le guarda los datos nuevos insertados.

```

guardar.setOnClickListener((v) -> {
    boolean existe = false;
    for(Usuario u : ((MainActivity)contexto).usuarios) {
        if(u.getUsuario() == usuario.getText().toString()) {
            existe = true;
            break;
        }
    }
    if(!existe) {
        usuarioLogeado.setImagen(imagenString);
        usuarioLogeado.setNombre(nombre.getText().toString());
        usuarioLogeado.setUsuario(usuario.getText().toString());
        usuarioLogeado.setApellidos(apellidos.getText().toString());

        if(!contraseña.getText().toString().equals("")){
            usuarioLogeado.setContraseña(contraseña.getText().toString());
        }

        ((MainActivity)contexto).ActualizarUsuario(usuarioLogeado, contexto);
        ((MainActivity)contexto).CreaFragmento(fragmentAnterior, R.id.fragment_container_logeado);
    } else {
        Toast.makeText(contexto, "El usuario ya existe", Toast.LENGTH_SHORT).show();
    }
});

cancelar.setOnClickListener((v) -> {
    ((MainActivity)contexto).CreaFragmento(fragmentAnterior, R.id.fragment_container_logeado);
});

```

O cancelar se lo vuelve al principal.

```

salir.setOnClickListener((v) -> {
    getActivity().finish();
    Intent intent = new Intent(getActivity(), MainActivity.class);
    startActivity(intent);
});

((MainActivity)contexto).PopBackFragment(rootView, fragmentAnterior);
return rootView;
}

```

O salir del fragmento.

```

public void elegirImagen() {
    Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
    intent.setType("image/*");
    if (intent.resolveActivity(getActivity().getPackageManager()) != null) {
        startActivityForResult(intent, COD_ELEGIR_IMAGEN);
    }
}

public void tomarFoto() {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (intent.resolveActivity(getActivity().getPackageManager()) != null) {
        startActivityForResult(intent, COD_TOMAR_FOTO);
    }
}
}

```


No hace falta describir los métodos porque el nombre lo describe.

```
@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    Bitmap imagenBitmap = null;

    if (requestCode == COD_ELEGIR_IMAGEN && resultCode == RESULT_OK && data != null) {
        Uri rutaImagen = data.getData();

        try {
            imagenBitmap = MediaStore.Images.Media.getBitmap(getActivity().getContentResolver(), rutaImagen);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    if (requestCode == COD_TOMAR_FOTO && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        imagenBitmap = (Bitmap) extras.get("data");
    }

    if (imagenBitmap != null) {
        imagenBitmap = Bitmap.createScaledBitmap(imagenBitmap, dstWidth: 100, dstHeight: 100, filter: false);
    }

    imagenString = ((MainActivity)contexto).BitmapAString(imagenBitmap);

    if (imagenString != "") {
        imagenPerfil.setImageDrawable(MainActivity.CircularLaImagen(imagenBitmap, contexto));
    }
}
```

Traemos la imagen, y cambiamos la escala, porque no me dejaba insertar las imágenes de tamaño muy grandes.

6.01. Agregar Productos

Es muy similar a la del perfil, Se ve el aspecto como Así:



1:14

Nombre

Stock

Precio

Descripción

CANCELAR AGREGAR

Cuando pulsamos el signo del “plus” (suma), se nos lleva para elegir de la galería o tomar la foto. Y traemos la imagen y rellenamos los campos y si damos a agregar se agregará el producto en la base de datos. Y ya la siguiente vez que el usuario iniciará la sesión se le mostrará el producto agregado. O si le damos a cancelar volvemos a la página principal como si no hemos hecho nada.

```
public class AgregarProductoFragment extends Fragment {

    private final int COD_ELEGIR_IMAGEN = 1;
    private final int COD_TOMAR_FOTO = 2;

    private Context contexto;
    private Fragment fragmentAnterior;

    private Producto producto;
    private EditText nombre;
    private EditText stock;
    private EditText precio;
    private EditText descripcion;
    private ImageView imagenProducto;
    private Button buttonAgregarProducto;
    private Button cancelarAgregarProducto;
    private boolean imagenElegido;
    private String imagenString = "";

    public AgregarProductoFragment(Context contexto, Fragment fragmentAnterior) {
        this.contexto = contexto;
        this.fragmentAnterior = fragmentAnterior;
        imagenElegido = false;
    }

    public View onCreateView(LayoutInflater inflater, final ViewGroup container, Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        final View rootView = inflater.inflate(R.layout.activity_agregar_producto, container, attachToRoot: false);
```

6.02. Eliminar Productos

Como ya lo tengo puesto el código en el 5.03, es con el OnLongClick, se me aparece el dialogo pidiendo me que si queremos eliminar el producto. Si le damos que elimine, pues que lo elimina desde la base de datos, y de la aplicación con el arrayList.

6.03. Editar Producto

Así se quedará el aspecto en la vista:



2:09

Nombre

Jersey

Stock

6

Precio

19.99

Descripción

tricotado con anchas rayas, en combinación de puntos, con detalle de hilos.

CANCELAR GUARDAR

y podemos modificarlo o cancelarlo, depende de lo que queramos.

Como ya lo tengo puesto el código en el 5.03, es con el `OnClick`, si le damos el click, se nos lleva a un fragmento donde traeremos los datos en los `editText`, y si queremos cambiar algo del producto, cambiamos y guardamos y salimos, como habíamos hecho en el de editar usuario utilizando el perfil fragmento. Pues así ahora con el producto:

```

public EditarProductoFragment(Context contexto, Fragment fragmentAnterior, Producto producto, int index) {
    this.contexto = contexto;
    this.fragmentAnterior = fragmentAnterior;
    this.producto = producto;
    this.index = index;
}

public View onCreateView(LayoutInflater inflater, final ViewGroup container, Bundle savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    final View rootView = inflater.inflate(R.layout.activity_editar_producto, container, attachToRoot: false);

    nombre = rootView.findViewById(R.id.nombre);
    stock = rootView.findViewById(R.id.stock);
    precio = rootView.findViewById(R.id.precio);
    descripcion = rootView.findViewById(R.id.descripcion);
    imagenProducto = rootView.findViewById(R.id.imagenProducto);
    imagenProducto.setOnClickListener((v) -> {
        MainActivity.DialogoParaElegirImagen(contexto, fragment: EditarProductoFragment.this);
    });

    nombre.setText(producto.getNombre());
    stock.setText(String.valueOf(producto.getStock()));
    precio.setText(String.valueOf(producto.getPrecio()));
    descripcion.setText(producto.getDescripcion());

    imagenString = producto.getImagen();
    Bitmap imagenBitmap = MainActivity.StringABitmap(producto.getImagen());
    imagenProducto.setImageDrawable(MainActivity.CircularLaImagen(imagenBitmap, contexto));

```

```

buttonEditarProducto = rootView.findViewById(R.id.buttonEditarProducto);
cancelarEditarProducto = rootView.findViewById(R.id.cancelarEditarProducto);

buttonEditarProducto.setOnClickListener((v) -> {
    if( nombre.getText().toString().equals("") ||
        stock.getText().toString().equals("") ||
        precio.getText().toString().equals("") ||
        descripcion.getText().toString().equals("") ||
        imagenString.equals("")) {
        Toast.makeText(contexto, text: "Falta algún campo Obligatorio!", Toast.LENGTH_SHORT).show();
    } else {
        producto.setNombre(nombre.getText().toString());
        producto.setDescripcion(descripcion.getText().toString());
        producto.setPrecio(Double.valueOf(precio.getText().toString()));
        producto.setStock(Integer.valueOf(stock.getText().toString()));

        //Se actualiza el campo imagenString en onActivityResult
        producto.setImagen(imagenString);

        MainActivity.ActualizarProducto(producto, contexto, index);
        cancelarEditarProducto.callOnClick();
    }
});

((MainActivity) contexto).PopBackStackFragment(rootView, fragmentAnterior);

cancelarEditarProducto.setOnClickListener((v) -> {
    ((MainActivity) contexto).AbrirFragmentOActividadPrincipal(fragmentAnterior);
});

return rootView;

```

```
public void elegirImagen() {
    Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
    intent.setType("image/*");
    if (intent.resolveActivity(getActivity().getPackageManager()) != null) {
        startActivityForResult(intent, COD_ELEGIR_IMAGEN);
    }
}

public void tomarFoto() {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (intent.resolveActivity(getActivity().getPackageManager()) != null) {
        startActivityForResult(intent, COD_TOMAR_FOTO);
    }
}
```

Elegiendo la imagen con la forma que queremos la imagen.

```
@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    Bitmap imagenBitmap = null;

    if (requestCode == COD_ELEGIR_IMAGEN && resultCode == RESULT_OK && data != null) {
        Uri rutaImagen = data.getData();

        try {
            imagenBitmap = MediaStore.Images.Media.getBitmap(getActivity().getContentResolver(), rutaImagen);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    if (requestCode == COD_TOMAR_FOTO && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        imagenBitmap = (Bitmap) extras.get("data");
    }

    imagenBitmap = Bitmap.createScaledBitmap(imagenBitmap, (dstWidth: 100, (dstHeight: 100, filter: false);
    imagenString = MainActivity.BitmapAString(imagenBitmap);

    if(imagenString != "") {
        imagenProducto.setImageDrawable(MainActivity.CircularLaImagen(imagenBitmap, contexto));
    }
}
}
```

Traemos la imagen, cambiamos la escala para que la imagen se pueda guardar en la base de datos, se ve muy escalada pero, yo no he meter una imagen del tamaño grande en la base dedos. Porque son máximo como 65536 caracteres solo.