

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS CARRERA DE SOFTWARE

APLICACIONES MÓVILES

NOMBRE: Edison Geovanny Guaichico Piñan

TEMA: Desarrollo de un sistema de pedidos en Android Studio

FECHA: 24/11/2022

Tabla de Contenidos

1. Estructura del proyecto.	2
2. Estructura de las carpetas con las clases java.	2
3. Estructura de carpetas de las vistas.	3
4. Código fuente.	3
5. Funcionamiento.....	48

- **Introducción**

Se ha desarrollado una aplicación que permite gestionar los pedidos de compra de un cliente, facturación de ventas, así como la gestión de productos (CRUD) y la gestión de clientes (CRUD) en Android Studio.

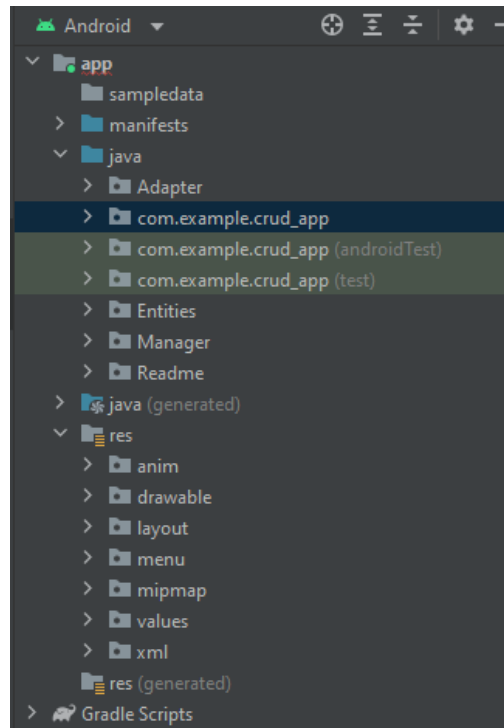
La aplicación está enfocada en un almacén de venta de equipos tecnológicos.

Esta aplicación está dividida en dos grupos, la primera parte es la del cliente, quien podrá realizar la solicitud de pedido a la aplicación., la segunda parte es la del vendedor quien podrá ver los pedidos de los clientes, realizar la gestión de productos, realizar facturas de venta y ver la lista de pedidos y facturas.

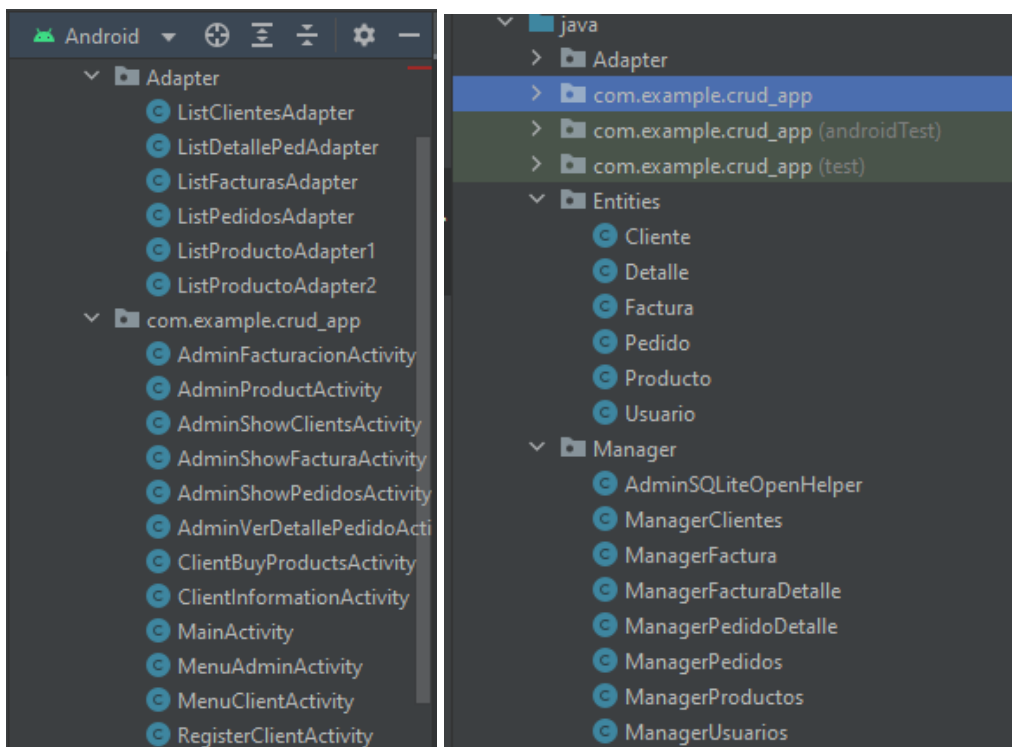
Para un mejor manejo del sistema se ha creado un login, este permite enviar a dos cuentas diferentes, la primera cuenta es para el “Vendedor”, quien puede crear los productos, ver los pedidos realizados por los clientes y realizar facturas. En cambio, la otra cuenta es para los clientes quienes realizan el pedido.

- Desarrollo

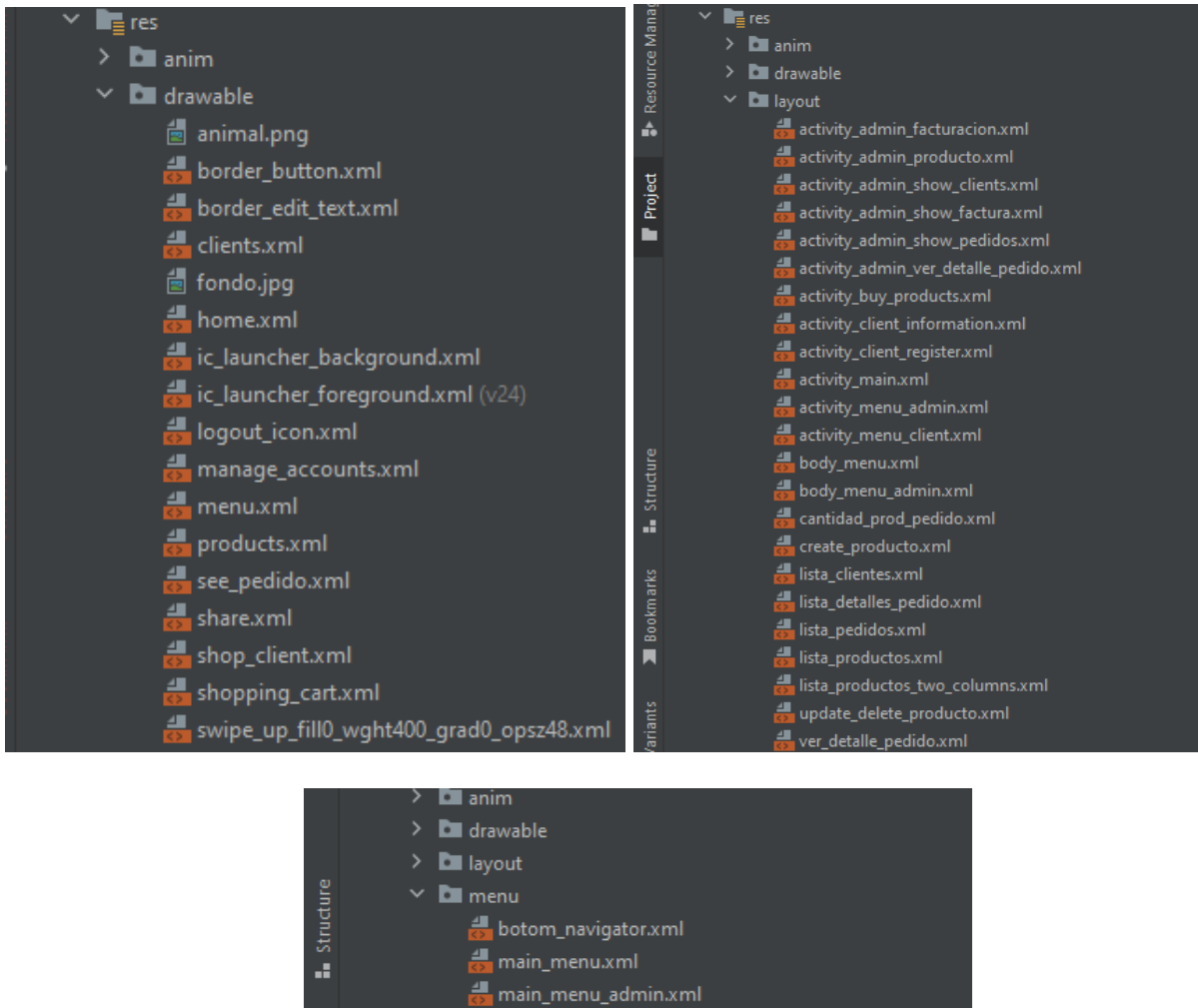
1. Estructura del proyecto.



2. Estructura de las carpetas con las clases java.



3. Estructura de carpetas de las vistas.



4. Código fuente.

- **AdminSQLiteOpenHelper**

```
public class AdminSQLiteOpenHelper extends SQLiteOpenHelper {

    public AdminSQLiteOpenHelper(@Nullable Context context, @Nullable String name,
    @Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL("CREATE TABLE usuario(id_usuario INTEGER PRIMARY KEY
        AUTOINCREMENT,cedula text(11), nombre text(20),apellido text(20),correo
        text(20),usuario text(20), clave text(20),estado text(5))");
        sqLiteDatabase.execSQL("CREATE TABLE cliente(id_cliente INTEGER PRIMARY KEY
        AUTOINCREMENT,cedula text(11), nombre text(20),apellido text(20),correo
        text(20),usuario text(20), clave text(20),estado text(5))");
    }
}
```

```

        sqLiteDatabase.execSQL("CREATE TABLE producto(id_producto INTEGER PRIMARY KEY AUTOINCREMENT, descripcion text(20), stock text(4) , status text (13), estado text (5), price text(10))");

        sqLiteDatabase.execSQL("CREATE TABLE pedido(id_pedido INTEGER PRIMARY KEY AUTOINCREMENT, " +
            " id_cliente int, " +
            "total text, "+
            "iva text(10), "+
            "fecha text(10), " +
            "facturar text (2), "+
            "estado text(5), " +
            "foreign key(id_cliente) references cliente(id_cliente))");
        sqLiteDatabase.execSQL("CREATE TABLE pedido_det(id INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "id_pedido int, " +
            "id_producto int, "+
            "cantidad int, "+
            "subtotal text(10), "+
            "estado text(5), " +
            "foreign key(id_pedido) references pedido(id_pedido), " +
            "foreign key(id_producto) references producto(id_producto))");
        sqLiteDatabase.execSQL("CREATE TABLE factura(id_factura INTEGER PRIMARY KEY AUTOINCREMENT, " +
            " id_cliente int, " +
            "total text, "+
            "iva text(10), "+
            "fecha text(10), " +
            "despachado text (2), "+
            "estado text(5), " +
            "foreign key(id_cliente) references cliente(id_cliente))");
        sqLiteDatabase.execSQL("CREATE TABLE factura_det(id INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "id_factura int, " +
            "id_producto int, "+
            "cantidad int, "+
            "subtotal text(10), "+
            "estado text(5), " +
            "foreign key(id_factura) references pedido(id_factura), " +
            "foreign key(id_producto) references producto(id_producto))");
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }
}

```

- **Entidades**

```
public class Cliente {
    private int id_cliente;
    private String cedula;
    private String nombre;
    private String apellido;
    private String correo;
    private String usuario;
    private String clave;
    private String estado;
}
```

```
public class Usuario {
    private int id_usuario;
    private String cedula;
    private String nombre;
    private String apellido;
    private String correo;
    private String usuario;
    private String clave;
    private String estado;
}
```

```
public class Producto implements Serializable {
    private int id_producto;
    private String descripcion;
    private String stock;
    private String status;
    private String price;
}
```

```
public class Pedido {
    private int id;
    private int idCliente;
    private String cedulaCliente;
    private String nombreCliente;
    private String apellidoCliente;
    private String total;
    private String iva;
    private String fecha;
    private String facturar;
}
```

```
public class Factura {
    private int id;
    private int idCliente;
    private String cedulaCliente;
    private String nombreCliente;
    private String apellidoCliente;
    private String total;
    private String iva;
    private String fecha;
    private String despachado;
}
```

```
public class Detalle implements Serializable {
    private int id;
    private int id_pedido;
    private String descripcionProducto;
    private int idProducto;
    private int cantidad;
    private String subtotal;
}
```

- **Manager**
ManagerClientes

```
public class ManagerClientes {
    private AdminSQLiteOpenHelper adsql;
    private SQLiteDatabase db;

    public ManagerClientes(Context context, String dbname, int version){
        adsql =new AdminSQLiteOpenHelper(context,dbname,null,version);
    }

    public Cliente[] allClientes(){
        db=adsql.getReadableDatabase();
        String[] param = new String[1];
        param[0]="true";
        Cursor cursor=db.rawQuery("SELECT * FROM cliente WHERE estado=?",param);
        Cliente [] ps;
        Cliente p;
        int i=0;
        if(cursor.getCount()<=0) {
            return null;
        }else{
            ps=new Cliente[cursor.getCount()];
            while (cursor.moveToNext()){
                p=new Cliente();
                p.setId_cliente(cursor.getInt(0));
                p.setCedula(cursor.getString(1));
                p.setNombre(cursor.getString(2));
                p.setApellido(cursor.getString(3));
                p.setCorreo(cursor.getString(4));
                p.setUsuario(cursor.getString(5));
                p.setClave(cursor.getString(6));
                p.setEstado(cursor.getString(7));
                ps[i++]=p;
            }
            return ps;
        }
    }

    public boolean insertCliente(String cedula,String nombre, String apellido,
String correo,String usuario,String clave){
        db=adsql.getWritableDatabase();

        try {
            ContentValues data=new ContentValues();
            data.put("cedula",cedula);
            data.put("nombre",nombre);
            data.put("apellido",apellido);
            data.put("correo",correo);
            data.put("usuario",usuario);
            data.put("clave",clave);
            data.put("estado","true");
            db.insert("cliente",null,data);
            db.close();
            return true;
        }catch (Exception e){
            return false;
        }
    }

    public boolean updateCliente(int id, String nombre, String apellido, String
correo,String usuario,String clave){
        db=adsql.getWritableDatabase();
    }
```

```

        ContentValues data=new ContentValues();
        data.put("id_cliente",id);
        data.put("nombre",nombre);
        data.put("apellido",apellido);
        data.put("correo",correo);
        data.put("usuario",usuario);
        data.put("clave",clave);
        db.update("cliente",data,"id_cliente="+id,null);
        db.close();
        return true;
    }

    public boolean deleteCliente(int id){
        db=adsql.getWritableDatabase();
        String[] param = new String[2];
        param[0]="false";
        param[1]=String.valueOf(id);
        db.execSQL("UPDATE cliente SET estado=? WHERE id_cliente=?",param);
        db.close();
        return true;
    }

    public Cliente getCliente(int id){
        db=adsql.getReadableDatabase();
        Cursor cursor=db.rawQuery("SELECT * FROM cliente WHERE id_cliente="+id,null);
        Cliente p;
        if(cursor.getCount()<=0) {
            return null;
        }else{
            p=new Cliente();
            cursor.moveToNext();
            p.setId_cliente(cursor.getInt(0));
            p.setCedula(cursor.getString(1));
            p.setNombre(cursor.getString(2));
            p.setApellido(cursor.getString(3));
            p.setCorreo(cursor.getString(4));
            p.setUsuario(cursor.getString(5));
            p.setClave(cursor.getString(6));
            p.setEstado(cursor.getString(7));
        }
        return p;
    }

    public Cliente login(String usuario,String clave){
        db=adsql.getReadableDatabase();
        String[] param = new String[3];
        param[0]= usuario;
        param[1]= clave;
        param[2]= "true";
        Cursor cursor=db.rawQuery("SELECT * FROM cliente cli WHERE cli.usuario=? AND cli.clave=? AND cli.estado=?",param);

        if(cursor.getCount()<=0) {
            return null;
        }else{
            Cliente p=new Cliente();
            cursor.moveToFirst();
            p.setId_cliente(cursor.getInt(0));
            p.setCedula(cursor.getString(1));
            p.setNombre(cursor.getString(2));
            p.setApellido(cursor.getString(3));
            p.setCorreo(cursor.getString(4));
            p.setUsuario(cursor.getString(5));
            p.setClave(cursor.getString(6));

```

```

        return p;
    }
}

```

ManagerUsuario

```

public class ManagerUsuarios {
    private AdminSQLiteOpenHelper adsql;
    private SQLiteDatabase db;

    public ManagerUsuarios(Context context, String dbname, int version){
        adsql =new AdminSQLiteOpenHelper(context,dbname,null,version);
    }

    public Usuario[] allUsers(){
        db=adsql.getReadableDatabase();
        String[] param = new String[1];
        param[0]="true";
        Cursor cursor=db.rawQuery("SELECT * FROM usuario WHERE estado = ?",param);
        Usuario [] ps;
        Usuario p;
        int i=0;
        if(cursor.getCount()<=0) {
            return null;
        }else{
            ps=new Usuario[cursor.getCount()];
            while (cursor.moveToNext()){
                p=new Usuario();
                p.setIdUsuario(cursor.getInt(0));
                p.setCedula(cursor.getString(1));
                p.setNombre(cursor.getString(2));
                p.setApellido(cursor.getString(3));
                p.setCorreo(cursor.getString(4));
                p.setUsuario(cursor.getString(5));
                p.setClave(cursor.getString(6));
                p.setEstado(cursor.getString(7));
                ps[i++]=p;
            }
            return ps;
        }
    }

    public boolean insertUser(String cedula,String nombre, String apellido, String correo,String usuario,String clave){
        db=adsql.getWritableDatabase();

        try {
            ContentValues data=new ContentValues();
            data.put("cedula",cedula);
            data.put("nombre",nombre);
            data.put("apellido",apellido);
            data.put("correo",correo);
            data.put("usuario",usuario);
            data.put("clave",clave);
            data.put("estado","true");
            db.insert("usuario",null,data);
            db.close();
            return true;
        }catch (Exception e){
            return false;
        }
    }

    public boolean updateUser(int id, String nombre, String apellido, String correo,String usuario,String clave){

```



```

        db=adsql.getWritableDatabase();
        ContentValues data=new ContentValues();
        data.put("id_usuario",id);
        data.put("nombre",nombre);
        data.put("apellido",apellido);
        data.put("correo",correo);
        data.put("usuario",usuario);
        data.put("clave",clave);
        db.update("usuario",data,"id_usuario="+id,null);
        db.close();
        return true;
    }

    public boolean deleteUser(int id){
        db=adsql.getWritableDatabase();
        String[] param = new String[2];
        param[0]="false";
        param[1]=String.valueOf(id);
        db.execSQL("UPDATE usuario SET estado=? WHERE id_usuario=?",param);
        db.close();
        return true;
    }

    public Usuario getUser(int id){
        db=adsql.getReadableDatabase();
        Cursor cursor=db.rawQuery("SELECT * FROM usuario WHERE id_usuario="+id,null);
        Usuario p;
        if(cursor.getCount()<=0) {
            return null;
        }else{
            p=new Usuario();
            cursor.moveToNext();
            p.setIdUsuario(cursor.getInt(0));
            p.setCedula(cursor.getString(1));
            p.setNombre(cursor.getString(2));
            p.setApellido(cursor.getString(3));
            p.setCorreo(cursor.getString(4));
            p.setUsuario(cursor.getString(5));
            p.setClave(cursor.getString(6));
            p.setEstado(cursor.getString(7));
        }
        return p;
    }

    public int countUsuario (){
        db=adsql.getReadableDatabase();
        Cursor cursor=db.rawQuery("SELECT * FROM usuario",null);
        int count=cursor.getCount();
        db.close();
        return count;
    }

    public Usuario login(String usuario,String clave){
        db=adsql.getReadableDatabase();
        String[] param = new String[3];
        param[0]= usuario;
        param[1]= clave;
        param[2]= "true";
        Cursor cursor=db.rawQuery("SELECT * FROM usuario user WHERE user.usuario=? AND user.clave=? AND user.estado=?",param);
    }

```

```

        if(cursor.getCount()<=0) {
            return null;
        }else{
            Usuario p=new Usuario();
            cursor.moveToFirst();
            p.setIdUsuario(cursor.getInt(0));
            p.setCedula(cursor.getString(1));
            p.setNombre(cursor.getString(2));
            p.setApellido(cursor.getString(3));
            p.setCorreo(cursor.getString(4));
            p.setUsuario(cursor.getString(5));
            p.setClave(cursor.getString(6));
            return p;
        }
    }
}

```

ManagerProducto

```

public class ManagerProductos {
    private AdminSQLiteOpenHelper adsql;
    private SQLiteDatabase db;

    public ManagerProductos(Context context, String dbname, int version){
        adsql =new AdminSQLiteOpenHelper(context,dbname,null,version);
    }

    public Producto[] allProductos(){
        db=adsql.getReadableDatabase();
        String[] param = new String[1];
        param[0]="true";
        Cursor cursor=db.rawQuery("SELECT * FROM producto WHERE estado=? ORDER BY
descripcion",param);
        Producto [] ps;
        Producto p;
        int i=0;
        if(cursor.getCount()<=0) {
            return null;
        }else{
            ps=new Producto[cursor.getCount()];
            while (cursor.moveToNext()){
                p=new Producto();
                p.setId(cursor.getInt(0));
                p.setDescripcion(cursor.getString(1));
                p.setStock(cursor.getString(2));
                p.setStatus(cursor.getString(3));
                p.setPrice(cursor.getString(5));
                ps[i++]=p;
            }
            return ps;
        }
    }

    public boolean insertProducto(String descripcion,int stock,String
disponible,String price){
        db=adsql.getWritableDatabase();
        ContentValues data=new ContentValues();
        data.put("descripcion",descripcion);
        data.put("stock",String.valueOf(stock));
        data.put("status",disponible);
        data.put("estado","true");
        data.put("price",price);
        Log.i("precio",""+price);
        db.insert("producto",null,data);
        db.close();
        return true;
    }
}

```

```

    }

    public boolean updateProducto(Producto prod) {
        db=adsql.getWritableDatabase();
        ContentValues data=new ContentValues();
        data.put("id_producto",prod.getId());
        data.put("descripcion",prod.getDescripcion());
        data.put("stock",prod.getStock());
        data.put("status",prod.getStatus());
        data.put("price",prod.getPrice());
        db.update("producto",data,"id_producto="+prod.getId(),null);
        db.close();
        return true;
    }

    public boolean deleteProducto(int id) {
        db=adsql.getWritableDatabase();
        String[] param = new String[2];
        param[0]="false";
        param[1]=String.valueOf(id);
        db.execSQL("UPDATE producto SET estado=? WHERE id_producto=?",param);
        db.close();
        return true;
    }

    public boolean updateStockProducto(List<Detalle> det) {
        db=adsql.getWritableDatabase();

        for (int i = 0; i < det.size(); i++) {
            int cantidad=det.get(i).getCantidad();
            Producto prod=getProducto(det.get(i).getIdProducto());
            int stockActual=Integer.parseInt(prod.getStock());
            String[] param = new String[2];
            param[0]=String.valueOf(stockActual-cantidad);
            param[1]=String.valueOf(det.get(i).getIdProducto());
            db.execSQL("UPDATE producto SET stock=? WHERE id_producto=?",param);
        }
        db.close();
        return true;
    }

    public Producto getProducto(int id) {
        db=adsql.getReadableDatabase();
        Cursor cursor=db.rawQuery("SELECT * FROM producto WHERE
id_producto="+id,null);

        if(cursor.getCount()<=0) {
            return null;
        }else{
            Producto p=new Producto();
            cursor.moveToFirst();
            p.setId(cursor.getInt(0));
            p.setDescripcion(cursor.getString(1));
            p.setStock(cursor.getString(2));
            p.setStatus(cursor.getString(3));
            p.setPrice(cursor.getString(4));
            return p;
        }
    }
}

```

ManagerPedido

```

public class ManagerPedidos {
    private AdminSQLiteOpenHelper adsql;
    private SQLiteDatabase db;
}

```

```

public ManagerPedidos(Context context, String dbname, int version){
    adsq1 =new AdminSQLiteOpenHelper(context,dbname,null,version);
}

public Pedido[] allPedidos(){
    db=adsq1.getReadableDatabase();
    Cursor cursor=db.rawQuery("SELECT
ped.id_pedido,cli.cedula,cli.nombre,cli.apellido,ped.total,ped.iva,ped.fecha,ped.facturar FROM pedido ped INNER JOIN cliente cli ON
cli.id_cliente=ped.id_cliente",null);
    Pedido [] ps;
    Pedido p;
    int i=0;
    if(cursor.getCount()<=0) {
        return null;
    }else{
        ps=new Pedido[cursor.getCount()];
        while (cursor.moveToNext()){
            p=new Pedido();
            p.setId(cursor.getInt(0));
            p.setCedulaCliente(cursor.getString(1));
            p.setNombreCliente(cursor.getString(2));
            p.setApellidoCliente(cursor.getString(3));
            p.setTotal(cursor.getString(4));
            p.setIva(cursor.getString(5));
            p.setFecha(cursor.getString(6));
            p.setFacturar(cursor.getString(7));
            ps[i++]=p;
        }
        return ps;
    }
}

public boolean insertPedido(int idCliente, String total, String iva, String fecha){
    db=adsq1.getWritableDatabase();

    ContentValues data=new ContentValues();
    data.put("id_cliente",idCliente);
    data.put("total",total);
    data.put("iva",iva);
    data.put("fecha",fecha);
    data.put("facturar","no");
    data.put("estado","true");
    db.insert("pedido",null,data);
    db.close();
    return true;
}

public Pedido getPedido(int id){
    db=adsq1.getReadableDatabase();
    String[] param = new String[1];
    param[0]=String.valueOf(id);

    Cursor cursor=db.rawQuery("SELECT
ped.id_pedido,cli.id_cliente,cli.cedula,cli.nombre,cli.apellido,ped.total,ped.iva,ped.fecha,ped.facturar FROM pedido ped INNER JOIN cliente cli ON
cli.id_cliente=ped.id_cliente WHERE ped.id_pedido=?",param);

```

```

        if(cursor.getCount()<=0) {
            return null;
        }else{
            Pedido p=new Pedido();
            cursor.moveToFirst();
            p.setId(cursor.getInt(0));
            p.setIdCliente(cursor.getInt(1));
            p.setCedulaCliente(cursor.getString(2));
            p.setNombreCliente(cursor.getString(3));
            p.setApellidoCliente(cursor.getString(4));
            p.setTotal(cursor.getString(5));
            p.setIva(cursor.getString(6));
            p.setFecha(cursor.getString(7));
            p.setFacturar(cursor.getString(8));
            return p;
        }
    }

    public int countPedido (){
        db=adsql.getReadableDatabase();
        Cursor cursor=db.rawQuery("SELECT * FROM pedido",null);
        int count=cursor.getCount();
        db.close();
        return count;
    }

    public boolean updatePedidoFacturado(int id, String facturar){
        db=adsql.getWritableDatabase();
        String[] param = new String[2];
        param[0]= facturar;
        param[1]= String.valueOf(id);
        db.execSQL("UPDATE pedido SET facturar=? WHERE id_pedido=?",param);
        db.close();
        return true;
    }
}

```

ManagerFactura

```

public class ManagerFactura {
    private AdminSQLiteOpenHelper adsql;
    private SQLiteDatabase db;

    public ManagerFactura(Context context, String dbname, int version){
        adsql =new AdminSQLiteOpenHelper(context,dbname,null,version);
    }

    public Factura[] allFacturas(){
        db=adsql.getReadableDatabase();
        Cursor cursor=db.rawQuery("SELECT
fac.id_factura,cli.cedula,cli.nombre,cli.apellido,fac.total,fac.iva,fac.fecha,fac.d
espachado FROM factura fac INNER JOIN cliente cli ON
cli.id_cliente=fac.id_cliente",null);
        Factura [] ps;
        Factura p;
        int i=0;
        if(cursor.getCount()<=0) {
            return null;
        }else{
            ps=new Factura[cursor.getCount()];
            while (cursor.moveToNext()){
                p=new Factura();
                p.setId(cursor.getInt(0));
                p.setCedulaCliente(cursor.getString(1));
                p.setNombreCliente(cursor.getString(2));
                p.setApellidoCliente(cursor.getString(3));
            }
        }
    }
}

```

```

        p.setTotal(cursor.getString(4));
        p.setIva(cursor.getString(5));
        p.setFecha(cursor.getString(6));
        p.setDespachado(cursor.getString(7));
        ps[i++]=p;
    }
    return ps;
}

}

public boolean insertFactura(int idCliente, String total, String iva, String
fecha){
    db=adsql.getWritableDatabase();

    ContentValues data=new ContentValues();
    data.put("id_cliente",idCliente);
    data.put("total",total);
    data.put("iva",iva);
    data.put("fecha",fecha);
    data.put("despachado","no");
    data.put("estado","true");
    db.insert("factura",null,data);
    db.close();
    return true;
}

public Factura getFactura(int id){
    db=adsql.getReadableDatabase();
    String[] param = new String[1];
    param[0]=String.valueOf(id);

    Cursor cursor=db.rawQuery("SELECT
fac.id_factura,cli.id_cliente,cli.cedula,cli.nombre,cli.apellido,fac.total,fac.iva,
fac.fecha,fac.despachado FROM factura fac INNER JOIN cliente cli ON
cli.id_cliente=fac.id_cliente WHERE fac.id_factura=?",param);

    if(cursor.getCount()<=0) {
        return null;
    }else{
        Factura p=new Factura();
        cursor.moveToFirst();
        p.setId(cursor.getInt(0));
        p.setIdCliente(cursor.getInt(1));
        p.setCedulaCliente(cursor.getString(2));
        p.setNombreCliente(cursor.getString(3));
        p.setApellidoCliente(cursor.getString(4));
        p.setTotal(cursor.getString(5));
        p.setIva(cursor.getString(6));
        p.setFecha(cursor.getString(7));
        p.setDespachado(cursor.getString(8));
        return p;
    }
}

public int countFactura (){
    db=adsql.getReadableDatabase();
    Cursor cursor=db.rawQuery("SELECT * FROM factura",null);
    int count=cursor.getCount();
    db.close();
    return count;
}

public boolean updateFacturaDespachado(int id, String despachado){
    db=adsql.getWritableDatabase();

```

```

        String[] param = new String[2];
        param[0]= despachado;
        param[1]= String.valueOf(id);
        db.execSQL("UPDATE factura SET despachado=? WHERE id_factura=?",param);
        db.close();
        return true;
    }
}

```

ManagerPedidoDetalle

```

public class ManagerDetalle {
    private AdminSQLiteOpenHelper adsql;
    private SQLiteDatabase db;

    public ManagerDetalle(Context context, String dbname, int version) {
        adsql = new AdminSQLiteOpenHelper(context, dbname, null, version);
    }

    public boolean insertPedidoDetalle(List<Detalle> det, int idPedido) {
        db = adsql.getWritableDatabase();

        for (int i = 0; i < det.size(); i++) {
            ContentValues data = new ContentValues();
            data.put("id_pedido", idPedido);
            data.put("id_producto", det.get(i).getIdProducto());
            data.put("cantidad", det.get(i).getCantidad());
            data.put("subtotal", det.get(i).getSubtotal());
            data.put("estado", "true");
            db.insert("pedido_det", null, data);
        }
        db.close();
        return true;
    }

    public Detalle[] getPedidoDetalle(int id) {
        db = adsql.getReadableDatabase();
        String[] param = new String[1];
        param[0]= String.valueOf(id);
        Cursor cursor = db.rawQuery("SELECT
ped.id_pedido,prod.id_producto,prod.descripcion, det.cantidad,det.subtotal FROM
pedido_det det INNER JOIN producto prod ON det.id_producto=prod.id_producto INNER
JOIN pedido ped ON ped.id_pedido=det.id_pedido WHERE ped.id_pedido=?", param);
        Detalle[] ps;
        Detalle p;
        int i=0;
        if (cursor.getCount() <= 0) {
            return null;
        } else {
            ps=new Detalle[cursor.getCount()];
            while (cursor.moveToNext()) {
                p = new Detalle();
                p.setId_pedido(cursor.getInt(0));
                p.setIdProducto(cursor.getInt(1));
                p.setDescripcionProducto(cursor.getString(2));
                p.setCantidad(cursor.getInt(3));
                p.setSubtotal(cursor.getString(4));
                ps[i++] = p;
            }
        }
        return ps;
    }
}

```

ManagerFacturaDetalle

```
public class ManagerFacturaDetalle {
    private AdminSQLiteOpenHelper adsql;
    private SQLiteDatabase db;

    public ManagerFacturaDetalle(Context context, String dbname, int version) {
        adsql = new AdminSQLiteOpenHelper(context, dbname, null, version);
    }

    public boolean insertFacturaDetalle(List<Detalle> det, int idFactura) {
        db = adsql.getWritableDatabase();
        for (int i = 0; i < det.size(); i++) {
            ContentValues data = new ContentValues();
            data.put("id_factura", idFactura);
            data.put("id_producto", det.get(i).getIdProducto());
            data.put("cantidad", det.get(i).getCantidad());
            data.put("subtotal", det.get(i).getSubtotal());
            data.put("estado", "true");
            db.insert("factura_det", null, data);
        }
        db.close();
        return true;
    }

    public Detalle[] getFacturaDetalle(int id) {
        db = adsql.getReadableDatabase();
        String[] param = new String[1];
        param[0] = String.valueOf(id);
        Cursor cursor = db.rawQuery("SELECT fac.id_factura,prod.descripcion,
det.cantidad,det.subtotal FROM factura_det det INNER JOIN producto prod ON
det.id_producto=prod.id_producto INNER JOIN factura fac ON
fac.id_factura=det.id_factura WHERE fac.id_factura=?", param);
        Detalle[] ps;
        Detalle p;
        int i=0;
        if (cursor.getCount() <= 0) {
            return null;
        } else {
            ps=new Detalle[cursor.getCount()];
            while (cursor.moveToNext()) {
                p = new Detalle();
                p.setId_pedido(cursor.getInt(0));
                p.setDescripcionProducto(cursor.getString(1));
                p.setCantidad(cursor.getInt(2));
                p.setSubtotal(cursor.getString(3));
                ps[i++] = p;
            }
        }
        return ps;
    }
}
```

- *Activities*

Main Activity

```
public class MainActivity extends AppCompatActivity {

    EditText txtUser,txtPassword;
    Button btnLogin;
    ManagerClientes crud;
    ManagerUsuarios gestion;
    TextView btnCreateCount;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        txtUser=(EditText) findViewById(R.id.txtUser);
        txtPassword=(EditText) findViewById(R.id.txtPassword);
        btnLogin=(Button) findViewById(R.id.btnLogin);
        btnCreateCount=(TextView) findViewById(R.id.btnCrearCuenta);
        SpannableString line = new SpannableString("Crear Cuenta Nueva");
        line.setSpan(new UnderlineSpan(), 0, line.length(), 0);
        btnCreateCount.setText(line);

        crud=new ManagerClientes(this, "compras",1);
        gestion=new ManagerUsuarios(this, "compras",1);

        if(gestion.countUsuario()==0) {
            gestion.insertUser("1006598657","Laura","Campos","laura@gmail.com","vendedor","admin");
        }

        btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String user=txtUser.getText().toString();
                String password=txtPassword.getText().toString();

                if((user.equals("vendedor") && password.equals("admin"))){
                    //LOGIN DE VENDEDOR
                    if (gestion.login(user,password)==null) {
                        Toast.makeText(getApplicationContext(), "No existe el usuario",Toast.LENGTH_SHORT).show();
                    }else{
                        Usuario usuario=gestion.login(user,password);
                        if(usuario.getUsuario().equals("vendedor") && usuario.getClave().equals("admin")){
                            Intent intento =new
                            Intent(MainActivity.this,MenuAdminActivity.class);
                            startActivity(intento);
                        }
                    }
                }else {
                    //LOGIN DE CLIENTE
                    if (crud.login(user, password) == null) {
                        Toast.makeText(getApplicationContext(), "No existe el cliente",
                        Toast.LENGTH_SHORT).show();
                    } else {
                        Cliente cli = crud.login(user, password);

                        SharedPreferences pref = getSharedPreferences("datos",
                        Context.MODE_PRIVATE);
                        SharedPreferences.Editor editor = pref.edit();//Se va ha
                        guardar las preferencias
                        editor.putString("idCli",
                        String.valueOf(cli.getId_cliente()));
                        editor.putString("cedulaCli", cli.getCedula());
                    }
                }
            }
        });
    }
}
```

Inicialización de la aplicación

```

        editor.putString("nombreCli", cli.getNombre());
        editor.putString("apellidoCli", cli.getApellido());
        editor.putString("correoCli", cli.getCorreo());
        editor.putString("usuarioCli", cli.getUsuario());
        editor.putString("claveCli", cli.getClave());
        editor.commit();
        Intent intento = new Intent(MainActivity.this,
MenuClientActivity.class);
        startActivity(intento);
    }
}
txtUser.setText("");
txtPassword.setText("");
});

btnCreateCount.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intento =new
Intent(MainActivity.this,RegisterClientActivity.class);
        startActivity(intento);
    }
});
}
}

```

RegisterClientActivity

```

public class RegisterClientActivity extends AppCompatActivity {
    EditText txtCedula,txtNombre,txtApellido,txtCorreo,txtUsuario,txtClave;
    Button btnGuardar;
    ManagerClientes crud;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_client_register);

        txtCedula=(EditText)findViewById(R.id.txtCedula);
        txtNombre=(EditText)findViewById(R.id.txtNombre);
        txtApellido=(EditText)findViewById(R.id.txtApellido);
        txtCorreo=(EditText)findViewById(R.id.txtCorreo);
        txtUsuario=(EditText)findViewById(R.id.txtUsuario);
        txtClave=(EditText)findViewById(R.id.txtClave);
        btnGuardar=(Button)findViewById(R.id.btnGuardarCliente);
        crud=new ManagerClientes(this, "compras",1);

        btnGuardar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String cedula=txtCedula.getText().toString();
                String nombre=txtNombre.getText().toString();
                String apellido=txtApellido.getText().toString();
                String correo=txtCorreo.getText().toString();
                String usuario=txtUsuario.getText().toString();
                String clave=txtClave.getText().toString();

                if(crud.insertCliente(cedula,nombre,apellido,correo,usuario,clave))
{
                    Toast.makeText(getApplicationContext(), "Cliente ingresado
correctamente", Toast.LENGTH_SHORT).show();
                    Intent intento = new Intent(RegisterClientActivity.this,
MainActivity.class);
                    startActivity(intento);

```

```

    }
    });
}
}

```

MenuClientActivity

```

public class MenuClientActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {

    DrawerLayout drawerLayout;
    NavigationView navigationView;
    Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu_client);

        drawerLayout= (DrawerLayout) findViewById(R.id.drawer_layout);
        navigationView =(NavigationView) findViewById(R.id.nav_view);
        toolbar=(Toolbar) findViewById(R.id.toolbar);
        toolbar.setBackgroundColor(Color.parseColor("#FFFFFF"));

        setSupportActionBar(toolbar);
        navigationView.bringToFront();
        ActionBarDrawerToggle toggle=new
ActionBarDrawerToggle(this,drawerLayout,toolbar,R.string.navigation_drawer_open,R.s
tring.navigation_drawer_close);
        drawerLayout.addDrawerListener(toggle);
        toggle.syncState();
        navigationView.setNavigationItemSelectedListener(this);
        navigationView.setCheckedItem(R.id.nav_home);
    }

    @Override
    public void onBackPressed(){ //Para cuando presiona sobre el menu
        if (drawerLayout.isDrawerOpen(GravityCompat.START)){
            drawerLayout.closeDrawer(GravityCompat.START);
        }else{
            super.onBackPressed();
        }
    }

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        switch (item.getItemId()){
            case R.id.nav_home:
                break;
            case R.id.shop_client:
                Intent intent1 =new Intent(MenuClientActivity.this,
ClientBuyProductsActivity.class);
                intent1.putExtra("base", "menu");
                startActivity(intent1);
                break;

            case R.id.client_information:
                Intent intent2 =new Intent(MenuClientActivity.this,
ClientInformationActivity.class);
                startActivity(intent2);
                break;
            case R.id.share:
                Toast.makeText(getApplicationContext(), "Share your ideas",
Toast.LENGTH_SHORT).show();
                break;
        }
    }
}

```

```

        case R.id.logout:

            SharedPreferences.Editor editor = getSharedPreferences("datos",
MODE_PRIVATE).edit();
            editor.clear().apply(); //Elimina las preferencias del dispositivo

            Intent intento3 =new
Intent(MenuClientActivity.this,MainActivity.class);
            startActivity(intento3);
            break;
        }
        return true;
    }
}

```

MenuAdminActivity

```

public class MenuAdminActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {
    DrawerLayout drawerLayout;
    NavigationView navigationView;
    Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu_admin);

        drawerLayout= (DrawerLayout) findViewById(R.id.drawer_layoutA);
        navigationView =(NavigationView) findViewById(R.id.nav_viewA);
        toolbar=(Toolbar) findViewById(R.id.toolbarA);

        setSupportActionBar(toolbar);
        navigationView.bringToFront();
        ActionBarDrawerToggle toggle=new
ActionBarDrawerToggle(this,drawerLayout,toolbar,R.string.navigation_drawer_open,R.s
tring.navigation_drawer_close);
        drawerLayout.addDrawerListener(toggle);
        toggle.syncState();
        navigationView.setNavigationItemSelectedListener(this);
        navigationView.setCheckedItem(R.id.nav_homeA);
    }
    @Override
    public void onBackPressed(){ //Para cuando presiona sobre el menu
        if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
            drawerLayout.closeDrawer(GravityCompat.START);
        }else{
            super.onBackPressed();
        }
    }
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {

        switch (item.getItemId()){
            case R.id.nav_homeA:
                break;

            case R.id.show_pedidosA:
                Intent intento1 =new Intent(MenuAdminActivity.this,
AdminShowPedidosActivity.class);
                startActivity(intento1);
                break;

            case R.id.invoiceA:
                Intent intento2 =new Intent(MenuAdminActivity.this,
AdminFacturacionActivity.class);
                intento2.putExtra("base","menu");

```

```

        startActivity(intento2);
        break;
    case R.id.show_invoiceA:
        Intent intento3 =new Intent(MenuAdminActivity.this,
AdminShowFacturaActivity.class);
        startActivity(intento3);
        break;

    case R.id.gestion_productosA:
        Intent intento4 =new Intent(MenuAdminActivity.this,
AdminProductActivity.class);
        startActivity(intento4);
        break;
    case R.id.show_clientesA:
        Intent intento5 =new Intent(MenuAdminActivity.this,
AdminShowClientsActivity.class);
        startActivity(intento5);
        break;
    case R.id.logoutA:
        Intent intento6 =new
Intent(MenuAdminActivity.this,MainActivity.class);
        startActivity(intento6);
        break;
    }
    return true;
}
}

```

ClientInformationActivity

```

public class ClientInformationActivity extends AppCompatActivity {
    EditText
txtEditCliNombre,txtEditCliApellido,txtEditCliCorreo,txtEditCliUsuario,txtEditCliCl
ave;
    Button btnActualizarCli, btnEliminarCli;
    ManagerClientes crud;
    String idCliente;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_client_information);

        txtEditCliNombre=(EditText) findViewById(R.id.txtEditCliNombre);
        txtEditCliApellido=(EditText) findViewById(R.id.txtEditCliApellido);
        txtEditCliCorreo=(EditText) findViewById(R.id.txtEditCliCorreo);
        txtEditCliUsuario=(EditText) findViewById(R.id.txtEditCliUsuario);
        txtEditCliClave=(EditText) findViewById(R.id.txtEditCliClave);
        btnActualizarCli=(Button) findViewById(R.id.btnActualizarCli);
        btnEliminarCli=(Button) findViewById(R.id.btnEliminarCli);

        //Obtener los parametros de SharedPreferences
        SharedPreferences pref= getSharedPreferences("datos",
Context.MODE_PRIVATE);
        SharedPreferences.Editor editor=pref.edit();//Se tiene acceso de modo
edicion
        idCliente=pref.getString("idCli","");
        txtEditCliNombre.setText(pref.getString("nombreCli",""));
        txtEditCliApellido.setText(pref.getString("apellidoCli",""));
        txtEditCliCorreo.setText(pref.getString("correoCli",""));
        txtEditCliUsuario.setText(pref.getString("usuarioCli",""));
        txtEditCliClave.setText(pref.getString("claveCli",""));

        crud=new ManagerClientes(this, "compras",1);
    }
}

```

```

        btnActualizarCli.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String nombre=txtEditCliNombre.getText().toString();
                String apellido=txtEditCliApellido.getText().toString();
                String correo=txtEditCliCorreo.getText().toString();
                String usuario=txtEditCliUsuario.getText().toString();
                String clave=txtEditCliClave.getText().toString();

                if
(crud.updateCliente(Integer.parseInt(idCliente), nombre, apellido, correo, usuario, clav
e)) {
                    Toast.makeText(getBaseContext(), "Cliente actualizado
correctamente", Toast.LENGTH_SHORT).show();
                }
            }
        });
        btnEliminarCli.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (crud.deleteCliente(Integer.parseInt(idCliente))) {
                    Intent intentol =new Intent(ClientInformationActivity.this,
MainActivity.class);
                    startActivity(intentol);
                    Toast.makeText(getBaseContext(), "Cliente eliminado
correctamente", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}

```

AdminProductsActivity

```

public class AdminProductActivity extends AppCompatActivity {
    ManagerProductos crud;
    List<Producto> listaProd;
    FloatingActionButton floatBtn;
    SearchView searchView;
    ListProductoAdapter1 listAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_producto);

        floatBtn=(FloatingActionButton)findViewById(R.id.floatBtn);
        crud=new ManagerProductos(this, "compras",1);
        searchView= (SearchView)findViewById(R.id.search_viewProd1);
        searchView.clearFocus();
        searchView.setQueryHint("search products ..");
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {

            @Override
            public boolean onQueryTextSubmit(String s) {
                return false;
            }

            @Override
            public boolean onQueryTextChange(String s) {
                filterList(s);
                return false;
            }
        });
        init(); //Llama a método listar productos
    }
}

```

```

        floatBtn.setOnClickListener(new View.OnClickListener() { // Abre el modal
para Insertar Producto
            @Override
            public void onClick(View v) {
                AlertDialog.Builder e=new
AlertDialog.Builder(AdminProductActivity.this);
                LayoutInflater inflater=getLayoutInflater();
                View view=inflater.inflate(R.layout.create_producto,null);
                e.setView(view);

                EditText txtDescripcionProd,txtStockProd,txtPrecioProd;
                RadioButton rdDispo, rdNoDispo;
                Button btnGuardarProducto;

                txtDescripcionProd=(EditText)
view.findViewById(R.id.txtDescripcionProducto);
                txtStockProd=(EditText) view.findViewById(R.id.txtStockProducto);
                txtPrecioProd=(EditText) view.findViewById(R.id.txtPrecioProducto);
                rdDispo=(RadioButton) view.findViewById(R.id.rdDisponible);
                rdNoDispo=(RadioButton) view.findViewById(R.id.rdNoDisponible);
                btnGuardarProducto=(Button) view.findViewById(R.id.btnGuardarProd);

                AlertDialog ad=e.create();
                ad.show();
                btnGuardarProducto.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        String descripcion =
txtDescripcionProd.getText().toString();
                        int stock =
Integer.parseInt(txtStockProd.getText().toString());
                        String precio=txtPrecioProd.getText().toString();

                        if (rdDispo.isChecked() == true) {
                            if (crud.insertProducto(descripcion, stock,
"disponible",precio)) {
                                Toast.makeText(getApplicationContext(), "Producto
insertado correctamente", Toast.LENGTH_SHORT).show();
                            }
                        }

                        if (rdNoDispo.isChecked() == true) {
                            if (crud.insertProducto(descripcion, stock, "no
disponible",precio)) {
                                Toast.makeText(getApplicationContext(), "Producto
insertado correctamente", Toast.LENGTH_SHORT).show();
                            }
                        }
                        ad.dismiss();
                    }
                });
                init();
            }
        });
    }

    public void init(){
        Producto[] productos = crud.allProductos();
        listaProd= new ArrayList<>();

        if(productos!=null) {
            listaProd= Arrays.asList(productos);
            listAdapter = new ListProductoAdapter1(listaProd, this, new
ListProductoAdapter1.OnItemClickListener() {

```

```

        @Override
        public void OnItemClick(Producto prod) {
            productSelected(prod);
        }
    });
    RecyclerView recyclerView = findViewById(R.id.listRecyclerView);
    recyclerView.setHasFixedSize(true);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    recyclerView.setAdapter(listAdapter);
}

}

public void productSelected(Producto prod){ //Modal para actualizar y eliminar
producto
    AlertDialog.Builder e=new AlertDialog.Builder(AdminProductActivity.this);
    LayoutInflater inflater=getLayoutInflater();
    View view=inflater.inflate(R.layout.update_delete_producto,null);
    e.setView(view);

    TextView idProd;
    EditText txtDescripcionProdUpdate,txtStockProdUpdate,txtPrecioProdUpdate;
    RadioButton rdDispoUpdate, rdNoDispoUpdate;
    Button btnActualizarProducto,btnEliminarProducto;

    idProd=(TextView)view.findViewById(R.id.txtIdProdSeleccionado);
    txtDescripcionProdUpdate=(EditText)
view.findViewById(R.id.txtEditDescripcion);
    txtStockProdUpdate=(EditText) view.findViewById(R.id.txtEditStock);
    txtPrecioProdUpdate=(EditText) view.findViewById(R.id.txtEditPrecio);
    rdDispoUpdate=(RadioButton) view.findViewById(R.id.rdDisponibleUpdate);
    rdNoDispoUpdate=(RadioButton) view.findViewById(R.id.rdNoDisponibleUpdate);
    btnActualizarProducto=(Button) view.findViewById(R.id.btnActualizarProd);
    btnEliminarProducto=(Button) view.findViewById(R.id.btnEliminarProd);

    idProd.setText("Producto: "+prod.getId());
    txtDescripcionProdUpdate.setText(prod.getDescripcion());
    txtPrecioProdUpdate.setText(prod.getPrice()+"");
    txtStockProdUpdate.setText(String.valueOf(prod.getStock()));
    rdDispoUpdate.setChecked(true);
    if(prod.getStatus().equals("disponible")){
        rdDispoUpdate.setChecked(true);
    }else{
        rdNoDispoUpdate.setChecked(true);
    }
    AlertDialog ad=e.create();
    ad.show();

    btnActualizarProducto.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String descripcion = txtDescripcionProdUpdate.getText().toString();
            String stock = txtStockProdUpdate.getText().toString();
            String precio=txtPrecioProdUpdate.getText().toString();

            if (rdDispoUpdate.isChecked() == true) {
                if (crud.updateProducto(new Producto(prod.getId(),descripcion,
stock, "disponible",precio))) {
                    Toast.makeText(getBaseContext(), "Producto actualizado
correctamente", Toast.LENGTH_SHORT).show();
                }
            }

            if (rdNoDispoUpdate.isChecked() == true) {
                if (crud.updateProducto(new Producto(prod.getId(),descripcion,

```



```

stock, "no disponible", precio))) {
            Toast.makeText(getBaseContext(), "Producto actualizado
correctamente", Toast.LENGTH_SHORT).show();
        }
    }
    init();
    ad.dismiss();
}

});

btnEliminarProducto.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (crud.deleteProducto(prod.getId())) {
            Toast.makeText(getBaseContext(), "Producto eliminado
correctamente", Toast.LENGTH_SHORT).show();
        }
        init();
        ad.dismiss();
    }
});
init();
}

public void filterList(String query) {
    Log.i("s", query);
    List<Producto> filterList= new ArrayList<>();
    for (int i = 0; i < listaProd.size() ; i++) {
        if (listaProd.get(i).getDescripcion().toLowerCase().contains(query.toLowerCase())) {
            filterList.add(listaProd.get(i));
        }
    }

    if (filterList.isEmpty()) {
        Toast.makeText(getBaseContext(), "No data found",
Toast.LENGTH_SHORT).show();
    } else {
        listAdapter.setFilterList(filterList);

        for (int i = 0; i < filterList.size() ; i++) {
            Log.i("des", filterList.get(i).getDescripcion());
        }
    }
}
}
}
}

```

AdminShowClientsActivity

```

public class AdminShowClientsActivity extends AppCompatActivity {
    ManagerClientes crud;
    List<Cliente> listClientes;
    SearchView searchView;
    ListClientesAdapter listAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_show_clients);

        crud = new ManagerClientes(this, "compras", 1);
        searchView = (SearchView) findViewById(R.id.search_viewCli);
        searchView.clearFocus();
        searchView.setQueryHint("search clients ..");
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {

```

```

        @Override
        public boolean onQueryTextSubmit(String s) {
            return false;
        }

        @Override
        public boolean onQueryTextChange(String s) {
            filterList(s);
            return false;
        }
    });
    init(); //Llama a método listar clientes
}

public void init() {
    Cliente[] clientes = crud.allClientes();
    listClientes = new ArrayList<>();

    if (clientes != null) {
        listClientes = Arrays.asList(clientes);
        if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.N) {
            listAdapter = new ListClientesAdapter(listClientes, this, new
ListClientesAdapter.OnItemClickListener() {
                @Override
                public void OnItemClick(Cliente cli) {
                    clienteSelected(cli);
                }
            });
        }
        RecyclerView recyclerView = findViewById(R.id.listRecyclerViewCli);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setAdapter(listAdapter);
    }
}

public void clienteSelected(Cliente cli){};

public void filterList(String query){
    List<Cliente> filterList= new ArrayList<>();
    for (int i = 0; i <listClientes.size() ; i++) {
        if(listClientes.get(i).getApellido().toLowerCase().contains(query.toLowerCase())){
            filterList.add(listClientes.get(i));
        }
    }

    if(filterList.isEmpty()){
        Toast.makeText(getApplicationContext(), "No data found",
Toast.LENGTH_SHORT).show();
    }else{
        for (int i = 0; i <filterList.size() ; i++) {
            Log.i("des",filterList.get(i).getApellido());
        }
        listAdapter.setFilterList( filterList);
    }
}
}

```

ClientBuyProductsActivity

```

public class ClientBuyProductsActivity extends AppCompatActivity implements
Serializable{
    ManagerProductos mProducto;
}

```

```

ManagerPedidos mPedidos;
ManagerPedidoDetalle mPedDetalle;
List<Producto> listaProd;
List<Detalle> productosPedidoDetalle;
ListProductoAdapter2 listAdapter;
SearchView searchView;
Button btnGuardarPedido, btnCancelarPedido, btnCarritoPedido;
TextView txtSubtotalValue, txtIvaValue, txtTotalValue;
double totalPedido;
double totalIva;
String idCliente;
int cant, idProducto;
boolean productosSeleccionadoParaPedido;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_buy_products);

    //Inicialización de variables
    txtSubtotalValue =(TextView) findViewById(R.id.txtSubtotalValue);
    txtIvaValue =(TextView) findViewById(R.id.txtIvaValue);
    txtTotalValue =(TextView) findViewById(R.id.txtTotalValue);
    productosPedidoDetalle=new ArrayList<>();
    productosSeleccionadoParaPedido=false;
    btnGuardarPedido=(Button) findViewById(R.id.btnGuardarPedido);
    btnCancelarPedido=(Button) findViewById(R.id.btnCancelarPedido);
    btnCarritoPedido=(Button) findViewById(R.id.btnVerCarritoPedido);
    mPedidos=new ManagerPedidos(this, "compras",1);
    mPedDetalle=new ManagerDetalle(this, "compras",1);
    mProducto=new ManagerProductos(this, "compras",1);
    searchView= (SearchView)findViewById(R.id.search_viewProd2);
    searchView.clearFocus();
    searchView.setQueryHint("search products ..");
    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String s) {
            return false;
        }

        @Override
        public boolean onQueryTextChange(String s) {
            filterList(s);
            return false;
        }
    });
    init(); //Llama a método listar productos

    //Obtener los parametros de SharedPreferences.
    SharedPreferences pref= getSharedPreferences("datos",
Context.MODE_PRIVATE);
    SharedPreferences.Editor editor=pref.edit();//Se tiene acceso de modo
edicion
    idCliente=pref.getString("idCli","");

    //Recibe los parámetros desde el Detalle
    Bundle bundle = this.getIntent().getExtras();
    if(bundle.getString("base").equals("detalle")){
        bundle = getIntent().getBundleExtra("lista");
        ArrayList<Detalle> list= (ArrayList<Detalle>)
bundle.getSerializable("detalle");
        productosPedidoDetalle=list;
        productosSeleccionadoParaPedido=true;
        mostrarCamposCalculados(productosPedidoDetalle);
    }
}

```

```

    }

    btnGuardarPedido.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(productosSeleccionadoParaPedido) {
                //Obtener la fecha actual del sistema
                String fechaCadena = new SimpleDateFormat("dd-MM-
yyyy").format(new Date());
                //Guardar Cabecera
                mPedidos.insertPedido(Integer.parseInt(idCliente),
String.valueOf(totalPedido), String.valueOf(totalIva), fechaCadena);
                //Guardar Detalle
                mPedDetalle.insertPedidoDetalle(productosPedidoDetalle,
mPedidos.countPedido());
                //Reducir Stock
                mProducto.updateStockProducto(productosPedidoDetalle);
                //Limpieza de datos
                LimpiarCampos();
                productosSeleccionadoParaPedido=false;
                Toast.makeText(getBaseContext(), "Pedido guardado",
Toast.LENGTH_SHORT).show();
            }else{
                Toast.makeText(getBaseContext(), "Debes seleccionar algún
producto", Toast.LENGTH_SHORT).show();
            }
        }
    });

    btnCancelarPedido.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            LimpiarCampos();
            Toast.makeText(getBaseContext(), "Pedido cancelado",
Toast.LENGTH_SHORT).show();
        }
    });

    btnCarritoPedido.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intento =new Intent(ClientBuyProductsActivity.this,
AdminVerDetallePedidoActivity.class);
            intento.putExtra("opcion", "1");
            SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
            intento.putExtra("fecha", sdf.format(new Date()));
            intento.putExtra("total", String.valueOf(totalPedido));
            Bundle bundle = new Bundle();
            ArrayList<Detalle> lista = new ArrayList<>();
            for (int i = 0; i < productosPedidoDetalle.size(); i++) {
                lista.add(productosPedidoDetalle.get(i));
            }
            bundle.putSerializable("detalle", lista);
            intento.putExtra("lista", bundle);
            startActivity(intento);
        }
    });
}

public void init(){
    Producto[] productos = mProducto.allProductos();
    listaProd= new ArrayList<>();

    if(productos!=null) {
        listaProd= Arrays.asList(productos);
    }
}

```

```

        listAdapter = new ListProductoAdapter2(listaProd, this, new
ListProductoAdapter2.OnItemClickListener() {
            @Override
            public void onItemClick(Producto prod) {
                productSelected(prod);
            }
        });
        RecyclerView recyclerView =
findViewById(R.id.listRecyclerViewTwoColum);
        recyclerView.setLayoutManager(new
GridLayoutManager(this, 2, LinearLayoutManager.HORIZONTAL, false));
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setAdapter(listAdapter);
    }
    public void productSelected (Producto prod) {
        AlertDialog.Builder e=new
AlertDialog.Builder(ClientBuyProductsActivity.this);
        LayoutInflater inflater=getLayoutInflater();
        View view=inflater.inflate(R.layout.cantidad_prod_pedido,null);
        e.setView(view);

        AlertDialog ad=e.create();
        ad.show();

        Producto p=prod;
        idProducto=prod.getId();
        EditText cantidad=(EditText) view.findViewById(R.id.txtCantidadPedido);
        Button btnSeleccionarProducto=(Button)
view.findViewById(R.id.btnSeleccionarProducto);

        cantidad.addTextChangedListener(new TextWatcher() {
            @Override
            public void onTextChanged(CharSequence s, int start, int before, int
count) {
                if(cantidad.getText().hashCode() == s.hashCode()){
                    if(!(cantidad.getText().toString().matches("[+-
]?\\d*(\\.\\d+)?"))){ //Verifica si es dígito
                        cantidad.setError(null);
                        cantidad.setError("Ingresar un número");
                    }
                }
            }
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int
after) {}
            @Override
            public void afterTextChanged(Editable s) {}
        });

        btnSeleccionarProducto.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                cant=Integer.parseInt(cantidad.getText().toString());
                double subtotal=Math.round((cant*Double.parseDouble(p.getPrice())) *
100.0) / 100.0;
                productosPedidoDetalle.add(new
Detalle(idProducto,p.getDescripcion(),cant,String.valueOf(subtotal)));
                productosSeleccionadoParaPedido=true;
                ad.dismiss();
                mostrarCamposCalculados(productosPedidoDetalle);
                Toast.makeText(getBaseContext(), "Producto seleccionado",
Toast.LENGTH_SHORT).show();
            }
        });
    }

```

Validación de
campo

```

    });
}
public void filterList(String query){
    List<Producto> filterList= new ArrayList<>();
    for (int i = 0; i <listaProd.size() ; i++) {
if(listaProd.get(i).getDescripcion().toLowerCase().contains(query.toLowerCase())){
        filterList.add(listaProd.get(i));
    }
    }

    if(filterList.isEmpty()){
        Toast.makeText(getBaseContext(), "No data found",
Toast.LENGTH_SHORT).show();
    }else{
        listAdapter.setFilterList(filterList);
    }
}
public void mostrarCamposCalculados(List<Detalle>lista){
    double subTotal=0;
    double iva=0;
    for (Detalle dt: lista) {
        subTotal+= Double.parseDouble(dt.getSubtotal());
        iva+= Double.parseDouble((dt.getSubtotal()))*0.12;
    }
    totalIva=Math.round(((iva)) * 100.0) / 100.0;
    totalPedido=Math.round(((subTotal+iva)) * 100.0) / 100.0;
    txtSubtotalValue.setText(subTotal+"");
    txtIvaValue.setText(iva+"");
    txtTotalValue.setText(totalPedido+"");
}
public void LimpiarCampos(){
    productosPedidoDetalle=new ArrayList<>();
    totalPedido = 0;
    totalIva = 0;
    txtSubtotalValue.setText("");
    txtIvaValue.setText("");
    txtTotalValue.setText("");
}

@Override
public void onBackPressed() {
    Intent intento =new Intent(ClientBuyProductsActivity.this,
MenuClientActivity.class);
    startActivity(intento);
}
}
}

```

AdminShowPedidosActivity

```

public class AdminShowPedidosActivity extends AppCompatActivity {
    ManagerPedidos gestion;
    List<Pedido> listPedidos;
    SearchView searchView;
    ListPedidosAdapter listAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_show_pedidos);

        gestion = new ManagerPedidos(this, "compras", 1);
        searchView = (SearchView) findViewById(R.id.search_viewPed);
    }
}

```

```

searchView.clearFocus();
searchView.setQueryHint("search orders ..");
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String s) {
        return false;
    }

    @Override
    public boolean onQueryTextChange(String s) {
        filterList(s);
        return false;
    }
});
init();
}

public void init() {
    Pedido[] pedidos = gestion.allPedidos();
    listPedidos = new ArrayList<>();

    if (pedidos != null) {
        listPedidos = Arrays.asList(pedidos);
        listAdapter = new ListPedidosAdapter(listPedidos, this, new
ListPedidosAdapter.OnItemClickListener() {
            @Override
            public void onItemClick(Pedido ped) {pedidoSelected(ped);
            }
        });
        RecyclerView recyclerView = findViewById(R.id.listRecyclerViewPed);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setAdapter(listAdapter);
    }
}

public void pedidoSelected(Pedido ped){
    Intent intento =new Intent(AdminShowPedidosActivity.this,
AdminVerDetallePedidoActivity.class);
    intento.putExtra("opcion","0");
    intento.putExtra("id",String.valueOf(ped.getId()));
    intento.putExtra("fecha",ped.getFecha());
    intento.putExtra("total",ped.getTotal());
    startActivity(intento);
}

public void filterList(String query){
    List<Pedido> filterList= new ArrayList<>();
    for (int i = 0; i <listPedidos.size() ; i++) {
        if(listPedidos.get(i).getApellidoCliente().toLowerCase().contains(query.toLowerCase
())){
            filterList.add(listPedidos.get(i));
        }
    }

    if(filterList.isEmpty()){
        Toast.makeText(getBaseContext(), "No data found",
Toast.LENGTH_SHORT).show();
    }else{
        listAdapter.setFilterList(filterList);
    }
}
@Override

```

```

    public void onBackPressed() {
        Intent intento =new Intent (AdminShowPedidosActivity.this,
MenuAdminActivity.class);
        startActivity(intento);
    }
}

```

AdminFacturacionActivity

```

public class AdminFacturacionActivity extends AppCompatActivity {
    ManagerProductos mProducto;
    ManagerFactura mFactura;
    ManagerFacturaDetalle mFacDetalle;
    ManagerClientes mCliente;
    List<Producto> listaProd;
    List<Detalle> productosFacturaDetalle;
    ListProductoAdapter2 listAdapter;
    SearchView searchView;
    Spinner spinner;
    Button btnGuardarFactura,btnCancelarFactura,btnDetalleFactura;
    TextView txtSubtotalValue,txtIvaValue,txtTotalValue;
    double totalFactura;
    double totalIva;
    int cant,idProducto;
    boolean productosSeleccionadoParaFactura;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_facturacion);

        //Inicialización de variables
        txtSubtotalValue =(TextView)
findViewById(R.id.txtSubtotalValueFacturacion);
        txtIvaValue =(TextView) findViewById(R.id.txtIvaValueFacturacion);
        txtTotalValue =(TextView) findViewById(R.id.txtTotalValueFacturacion);
        productosSeleccionadoParaFactura=false;
        btnGuardarFactura=(Button)findViewById(R.id.btnGuardarFactura);
        btnCancelarFactura=(Button)findViewById(R.id.btnCancelarFacturacion);
        btnDetalleFactura=(Button)findViewById(R.id.btnVerDetalleFacturacion);
        mProducto=new ManagerProductos(this, "compras",1);
        mCliente=new ManagerClientes(this, "compras",1);
        mFactura=new ManagerFactura(this, "compras",1);
        mFacDetalle=new ManagerFacturaDetalle(this, "compras",1);
        productosFacturaDetalle=new ArrayList<>();
        spinner=(Spinner)findViewById(R.id.spinner);
        searchView= (SearchView)findViewById(R.id.search_viewProdFacturacion);
        searchView.clearFocus();
        searchView.setQueryHint("search products ..");
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String s) {
                return false;
            }

            @Override
            public boolean onQueryTextChange(String s) {
                filterList(s);
                return false;
            }
        });
        init(); //Llama a método listar productos
        initClient(); //Llama a método listar clientes

        //Recibe los parámetros de regreso desde el Detalle
        Bundle bundle = this.getIntent().getExtras();
    }
}

```



```

        if(bundle.getString("base").equals("detalle")){
            bundle = getIntent().getBundleExtra("lista");
            ArrayList<Detalle> list= (ArrayList<Detalle>)
bundle.getSerializable("detalle");
            productosFacturaDetalle=list;
            productosSeleccionadoParaFactura=true;
            mostrarCamposCalculados(productosFacturaDetalle);
        }

        btnGuardarFactura.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(productosSeleccionadoParaFactura) {
                    //Obtener la fecha actual del sistema
                    String fechaCadena = new SimpleDateFormat("dd-MM-yyyy").format(
new Date());

                    //Guardar Cabecera
                    Cliente cli= (Cliente) spinner.getSelectedItem();

mFactura.insertFactura(Integer.parseInt(cli.getId_cliente()+""),
String.valueOf(totalFactura), String.valueOf(totalIva), fechaCadena);
                    //Guardar Detalle
                    mFacDetalle.insertFacturaDetalle(productosFacturaDetalle,
mFactura.countFactura());
                    //Reducir Stock
                    mProducto.updateStockProducto(productosFacturaDetalle);
                    //Limpieza de datos
                    LimpiarCampos();
                    productosSeleccionadoParaFactura=false;
                    Toast.makeText(getBaseContext(), "Factura guardado",
Toast.LENGTH_SHORT).show();
                }else{
                    Toast.makeText(getBaseContext(), "Debes seleccionar algún
producto", Toast.LENGTH_SHORT).show();
                }
            }
        });

        btnCancelarFactura.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                LimpiarCampos();
                Toast.makeText(getBaseContext(), "Factura cancelado",
Toast.LENGTH_SHORT).show();
            }
        });

        btnDetalleFactura.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intento =new Intent(AdminFacturacionActivity.this,
AdminVerDetallePedidoActivity.class);
                intento.putExtra("opcion","2");
                SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
                intento.putExtra("fecha", sdf.format(new Date()));
                intento.putExtra("total",String.valueOf(totalFactura));
                Bundle bundle = new Bundle();
                ArrayList<Detalle> lista = new ArrayList<>();
                for (int i = 0; i < productosFacturaDetalle.size(); i++) {
                    lista.add(productosFacturaDetalle.get(i));
                }
                bundle.putSerializable("detalle", lista);
                intento.putExtra("lista",bundle);
                startActivity(intento);
            }
        });
    }
}

```

```

    }

    public void init() {
        Producto[] productos = mProducto.allProductos();
        listaProd = new ArrayList<>();

        if (productos != null) {
            listaProd = Arrays.asList(productos);
            listAdapter = new ListProductoAdapter2(listaProd, this, new
ListProductoAdapter2.OnItemClickListener() {
                @Override
                public void onItemClick(Producto prod) {
                    productSelected(prod);
                }
            });
            RecyclerView recyclerView =
findViewById(R.id.listRecyclerViewTwoColum);
            recyclerView.setLayoutManager(new GridLayoutManager(this, 2,
LinearLayoutManager.HORIZONTAL, false));
            recyclerView.setHasFixedSize(true);
            recyclerView.setLayoutManager(new LinearLayoutManager(this));
            recyclerView.setAdapter(listAdapter);
        }
    }

    public void initClient() {
        Cliente[] clientes = mCliente.allClientes();
        List<Cliente> lista = new ArrayList<>();

        if (clientes != null) {
            lista = Arrays.asList(clientes);
        }
        ArrayAdapter<Cliente> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, lista);
        spinner.setAdapter(adapter);
    }

    public void productSelected (Producto prod) {
        AlertDialog.Builder e = new
AlertDialog.Builder(AdminFacturacionActivity.this);
        LayoutInflater inflater = getLayoutInflater();
        View view = inflater.inflate(R.layout.cantidad_prod_pedido, null);
        e.setView(view);

        AlertDialog ad = e.create();
        ad.show();

        Producto p = prod;
        idProducto = prod.getId();
        EditText cantidad = (EditText) view.findViewById(R.id.txtCantidadPedido);
        Button btnSeleccionarProducto = (Button)
view.findViewById(R.id.btnSeleccionarProducto);

        cantidad.addTextChangedListener(new TextWatcher() {
            @Override
            public void onTextChanged(CharSequence s, int start, int before, int
count) {
                if (cantidad.getText().hashCode() == s.hashCode()) {
                    if (! (cantidad.getText().toString().matches("[+-
]?\\d*(\\.\\d+)?")) { //Verifica si es dígito
                        cantidad.setError(null);
                        cantidad.setError("Ingresar un número");
                    }
                }
            }
        })
    }
    @Override

```

```

        public void beforeTextChanged(CharSequence s, int start, int count, int
after) {}

        @Override
        public void afterTextChanged(Editable s) {}
    });

    btnSeleccionarProducto.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            cant=Integer.parseInt(cantidad.getText().toString());
            double subtotal=Math.round((cant*Double.parseDouble(p.getPrice()))
* 100.0) / 100.0;
            productosFacturaDetalle.add(new
Detalle(idProducto,p.getDescripcion(),cant,String.valueOf(subtotal)));
            productosSeleccionadoParaFactura=true;
            ad.dismiss();
            mostrarCamposCalculados(productosFacturaDetalle);
            Toast.makeText(getBaseContext(), "Producto seleccionado",
Toast.LENGTH_SHORT).show();
        }
    });
}

public void filterList(String query){
    List<Producto> filterList= new ArrayList<>();
    for (int i = 0; i <listaProd.size() ; i++) {

if(listaProd.get(i).getDescripcion().toLowerCase().contains(query.toLowerCase())){
            filterList.add(listaProd.get(i));
        }
    }

    if(filterList.isEmpty()){
        Toast.makeText(getBaseContext(), "No data found",
Toast.LENGTH_SHORT).show();
    }else{
        listAdapter.setFilterList(filterList);
    }
}

public void mostrarCamposCalculados(List<Detalle>lista){
    double subTotal=0;
    double iva=0;
    for (Detalle dt: lista) {
        subTotal+= Double.parseDouble(dt.getSubtotal());
        iva+= Double.parseDouble((dt.getSubtotal()))*0.12;
    }
    totalIva=Math.round(((iva)) * 100.0) / 100.0;
    totalFactura=Math.round(((subTotal+iva)) * 100.0) / 100.0;
    txtSubtotalValue.setText(subTotal+"");
    txtIvaValue.setText(iva+"");
    txtTotalValue.setText(totalFactura+"");
}

public void LimpiarCampos() {
    productosFacturaDetalle=new ArrayList<>();
    totalFactura = 0;
    totalIva = 0;
    txtSubtotalValue.setText("");
    txtIvaValue.setText("");
    txtTotalValue.setText("");
}

@Override
public void onBackPressed() {
    Intent intento =new Intent(AdminFacturacionActivity.this,
MenuAdminActivity.class);
}

```

```

        startActivity(intento);
    }
}

```

AdminShowFacturaActivity

```

public class AdminShowFacturaActivity extends AppCompatActivity implements
Serializable {

    ManagerFactura gestion;
    List<Factura> listFactura;
    SearchView searchView;
    ListFacturasAdapter listAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_show_factura);

        gestion = new ManagerFactura(this, "compras", 1);
        searchView = (SearchView) findViewById(R.id.search_viewFac);
        searchView.clearFocus();
        searchView.setQueryHint("search orders ..");
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String s) {
                return false;
            }

            @Override
            public boolean onQueryTextChange(String s) {
                filterList(s);
                return false;
            }
        });
        init(); //Llama a método listar facturas
    }

    public void init() {
        Factura[] facturas = gestion.allFacturas();
        listFactura = new ArrayList<>();

        if (facturas != null) {
            listFactura = Arrays.asList(facturas);
            listAdapter = new ListFacturasAdapter(listFactura, this, new
ListFacturasAdapter.OnItemClickListener() {
                @Override
                public void OnItemClick(Factura fac) {facturaSelected(fac);
                }
            });
            RecyclerView recyclerView = findViewById(R.id.listRecyclerViewFac);
            recyclerView.setHasFixedSize(true);
            recyclerView.setLayoutManager(new LinearLayoutManager(this));
            recyclerView.setAdapter(listAdapter);
        }
    }

    public void facturaSelected(Factura fac){
        Intent intento =new Intent(AdminShowFacturaActivity.this,
AdminVerDetallePedidoActivity.class);
        intento.putExtra("opcion","3");
        intento.putExtra("id",String.valueOf(fac.getId()));
        intento.putExtra("fecha",fac.getFecha());
        intento.putExtra("total",fac.getTotal());
        Log.i("id factura en show",String.valueOf(fac.getId()));
        startActivity(intento);
    }
}

```

```

    }
    public void filterList(String query){
        List<Factura> filterList= new ArrayList<>();
        for (int i = 0; i <listFactura.size() ; i++) {

if(listFactura.get(i).getApellidoCliente().toLowerCase().contains(query.toLowerCase())){

            filterList.add(listFactura.get(i));

        }

    }

    if(filterList.isEmpty()){
        Toast.makeText(getBaseContext(), "No data found",
Toast.LENGTH_SHORT).show();
    }else{
        listAdapter.setFilterList(filterList);
    }
}
@Override
public void onBackPressed() {
    Intent intento =new Intent(AdminShowFacturaActivity.this,
MenuAdminActivity.class);
    startActivity(intento);
}
}

```

AdminVerDetalleActivity

```

public class AdminVerDetalleActivity extends AppCompatActivity implements
Serializable,BottomNavigationView.OnNavigationItemSelectedListener{
    TextView
txtNumPedido,txtFechaPedido,txtCliente,txtCedulaCliente,txtSubtotal,txtIva,txtTotal
,titleCabecera;

    ManagerPedidoDetalle mDetPedido;
    ManagerPedidos mPedido;
    ManagerFacturaDetalle mDetFactura;
    ManagerFactura mFactura;
    ManagerProductos mProducto;
    ManagerClientes mCliente;
    List<Detalle> listDetalle; //Variable para listar detalles
    ArrayList<Detalle> detallePedido; //Detalle utilizado para pedido
    ArrayList<Detalle> detalleFactura; //Detalle utilizado para factura
    ListDetallePedAdapter listAdapter;
    Button btnDespacharPedido,btnFacturarPedido;
    String opcion; //Hay la posibilidad de venir desde 4 partes al detalle
    BottomNavigationView BottomNavigationView;

    @SuppressWarnings("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_ver_detalle_pedido);

        btnDespacharPedido=(Button) findViewById(R.id.btnDespacharPedido);
        btnFacturarPedido=(Button) findViewById(R.id.btnFacturarPedido);
        titleCabecera=(TextView) findViewById(R.id.textView5);
        txtNumPedido=(TextView) findViewById(R.id.txtNumPedido);
        txtFechaPedido=(TextView) findViewById(R.id.txtFechaPedido);
        txtCliente=(TextView) findViewById(R.id.txtClientePedido);
        txtCedulaCliente=(TextView) findViewById(R.id.txtCedulaCliPedido);
        txtSubtotal=(TextView) findViewById(R.id.txtSubtotalPedido);
        txtIva=(TextView) findViewById(R.id.txtIvaPedido);
        txtTotal=(TextView) findViewById(R.id.txtTotalPedido);
        mPedido=new ManagerPedidos(this, "compras",1);
        mDetPedido=new ManagerDetalle(this, "compras",1);
    }
}

```

```

        mFactura=new ManagerFactura(this, "compras",1);
        mDetFactura=new ManagerFacturaDetalle(this, "compras",1);
        mProducto=new ManagerProductos(this, "compras",1);
        mCliente=new ManagerClientes(this, "compras",1);
        BottomNavigationView=(BottomNavigationView)
findViewById(R.id.bottom_navDetalle);
        BottomNavigationView.setOnNavigationItemSelectedListener(this);

        //Obtener los parametros del activity anterior.
        Bundle bb=this getIntent().getExtras();
        opcion=bb.getString("opcion");
        txtFechaPedido.setText(bb.getString("fecha"));
        txtTotal.setText("Total: "+bb.getString("total"));

        if(opcion.equals("0")){ // Pedido ya realizados, ver detalle desde
administrador
            int idPed=Integer.parseInt(bb.getString("id"));
            Pedido pedido=mPedido.getPedido(idPed);
            Cliente cliente=mCliente.getCliente(pedido.getIdCliente());
            mostrarDatosClientes((cliente.getApellido()+"
"+cliente.getNombre()),cliente.getCedula());
            btnDespacharPedido.setVisibility(View.GONE);
            btnFacturarPedido.setVisibility((pedido.getFacturar().equals("si")) ?
View.GONE : View.VISIBLE );
            titleCabecera.setText("Num Pedido:");
            txtNumPedido.setText(idPed+"");
            initDetallePedido(idPed);
            btnFacturarPedido.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    //Obtener la fecha actual del sistema
                    String fechaCadena = new SimpleDateFormat("dd-MM-
yyyy").format(new Date());
                    //Guardar Cabecera
                    mFactura.insertFactura(Integer.parseInt(pedido.getIdCliente()+""),
String.valueOf(pedido.getTotal()), String.valueOf(pedido.getIva()), fechaCadena);
                    //Guardar Detalle
                    mDetFactura.insertFacturaDetalle(listDetalle,
mFactura.countFactura());
                    //Reducir Stock
                    mProducto.updateStockProducto(listDetalle);

                    if(mPedido.updatePedidoFacturado(idPed,"si")) {
                        Toast.makeText(getBaseContext(), "Pedido facturado",
Toast.LENGTH_SHORT).show();
                        Intent intento =new
Intent(AdminVerDetallePedidoActivity.this,AdminShowPedidosActivity.class);
                        startActivity(intento);
                    }
                }
            });

        }else if(opcion.equals("1")){ // Pedido en ejecución, ver detalle desde
cliente
            //Obtener los parametros del cliente de SharedPreferences
            SharedPreferences pref= getSharedPreferences("datos",
Context.MODE_PRIVATE);
            SharedPreferences.Editor editor=pref.edit();
            mostrarDatosClientes((pref.getString("apellidoCli","")+
"+pref.getString("nombreCli",""),pref.getString("cedulaCli",""));

            btnDespacharPedido.setVisibility(View.GONE);
            btnFacturarPedido.setVisibility(View.GONE);
            titleCabecera.setText("Num Pedido:");

```

```

        txtNumPedido.setText("");
        Bundle bundle = new Bundle();
        bundle = getIntent().getBundleExtra("lista");
        detallePedido= (ArrayList<Detalle>) bundle.getSerializable("detalle");
//Obtener el detalle de pedido.
        mostrarValoresCalculados(detallePedido); //Calcular iva y subtotal
        initCarritoCompra();

    }else if (opcion.equals("2")){ // Facturas en ejecucion, ver detalle desde
administrador
        btnDespacharPedido.setVisibility(View.GONE);
        btnFacturarPedido.setVisibility(View.GONE);
        titleCabecera.setText("Num Factura:");
        txtNumPedido.setText("");
        mostrarDatosClientes("", "");
        Bundle bundle = new Bundle();
        bundle = getIntent().getBundleExtra("lista");
        detalleFactura= (ArrayList<Detalle>) bundle.getSerializable("detalle");
//Obtener el detalle de factura.
        mostrarValoresCalculados(detalleFactura); //Calcular iva y subtotal
        initDetalleFacturaEjecucion();

    }else{ // Facturas ya realizados, ver detalle desde
administrador
        btnDespacharPedido.setVisibility(View.VISIBLE);
        titleCabecera.setText("Num Factura:");
        int idFactura=Integer.parseInt(bb.getString("id"));
        Factura factura=mFactura.getFactura(idFactura);
        Cliente cliente=mCliente.getCliente(factura.getIdCliente());
        mostrarDatosClientes((cliente.getApellido()+"
"+cliente.getNombre()),cliente.getCedula());
        btnFacturarPedido.setVisibility(View.GONE);
        btnDespacharPedido.setVisibility((factura.getDespachado().equals("si"))
? View.GONE : View.VISIBLE );
        txtNumPedido.setText(String.valueOf(idFactura));
        initDetalleFactura(idFactura);
        btnDespacharPedido.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (mFactura.updateFacturaDespachado(idFactura,"si")) {
                    Toast.makeText(getApplicationContext(), "Factura despachado",
Toast.LENGTH_SHORT).show();
                    Intent intento =new
Intent(AdminVerDetallePedidoActivity.this,AdminShowFacturaActivity.class);
                    startActivity(intento);
                }
            }
        });
    }
}

public void initDetallePedido(int id){
    Detalle[] detalle = mDetPedido.getPedidoDetalle(id);
    listDetalle= new ArrayList<>();

    if(detalle!=null) {
        listDetalle= Arrays.asList(detalle);
        listAdapter = new ListDetallePedAdapter(0,listDetalle, this, new
ListDetallePedAdapter.OnItemClickListener() {

            @Override
            public void OnItemClick(Detalle cli, int position) {}
        });
    }
}

```

```

        RecyclerView recyclerView =
findViewById(R.id.listRecyclerViewDetallePed);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setAdapter(listAdapter);

        //Calcular iva y subtotal
        mostrarValoresCalculados(listDetalle);
    }
}

public void initCarritoCompra(){
    if( detallePedido!=null) {
        listDetalle= detallePedido;
        listAdapter = new ListDetallePedAdapter(1,listDetalle, this, new
ListDetallePedAdapter.OnItemClickListener() {
            @Override
            public void OnItemClick(Detalle det, int pos)
{eliminarDetalle(det,pos);}
        });
        RecyclerView recyclerView =
findViewById(R.id.listRecyclerViewDetallePed);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setAdapter(listAdapter);
    }
}

public void initDetalleFacturaEjecucion(){
    if( detalleFactura!=null) {
        listDetalle= detalleFactura;
        listAdapter = new ListDetallePedAdapter(1,listDetalle, this, new
ListDetallePedAdapter.OnItemClickListener() {
            @Override
            public void OnItemClick(Detalle det, int pos)
{eliminarDetalle(det,pos);}
        });
        RecyclerView recyclerView =
findViewById(R.id.listRecyclerViewDetallePed);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setAdapter(listAdapter);
    }
}

public void initDetalleFactura(int id){
    Detalle[] detalle = mDetFactura.getFacturaDetalle(id);
    listDetalle= new ArrayList<>();

    if(detalle!=null) {
        listDetalle= Arrays.asList(detalle);
        listAdapter = new ListDetallePedAdapter(0,listDetalle, this, new
ListDetallePedAdapter.OnItemClickListener() {

            @Override
            public void OnItemClick(Detalle cli, int position) {}
        });
        RecyclerView recyclerView =
findViewById(R.id.listRecyclerViewDetallePed);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setAdapter(listAdapter);

        //Calcular iva y subtotal
        mostrarValoresCalculados(listDetalle);
    }
}

```



```

}

public void eliminarDetalle(Detalle ped, int posicion){
    if(opcion.equals("1")){ //Elimina el detalle de pedido
        detallePedido.remove(posicion);
    }else{ //Elimina el detalle de factura
        detalleFactura.remove(posicion);
    }

    listAdapter.notifyDataSetChanged();
    List<Detalle> lista=listAdapter.getPedidoDetalles();
    mostrarValoresCalculados(lista);
}

public void mostrarValoresCalculados( List<Detalle> list){
    //Calcular iva y subtotal
    double subTotal=0;
    double iva=0;

    for (Detalle dt: list) {
        subTotal+= Double.parseDouble(dt.getSubtotal());
        iva+= Double.parseDouble((dt.getSubtotal()))*0.12;
    }
    txtIva.setText("Iva: "+Math.round((iva) * 100.0) / 100.0);
    txtSubtotal.setText("Subtotal: "+Math.round((subTotal) * 100.0) / 100.0);
    txtTotal.setText("Total: "+Math.round((iva+subTotal) * 100.0) / 100.0);
}

public void mostrarDatosClientes(String cliente,String cedula){
    txtCliente.setText(cliente);
    txtCedulaCliente.setText(cedula);
}

@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {
    if(opcion.equals("0")){ //Debe regresar a pedidos en la cuenta de
administrador
        Intent intento = new Intent(AdminVerDetallePedidoActivity.this,
AdminShowPedidosActivity.class);
        startActivity(intento);

    }else if(opcion.equals("1")){ //Debe regresar a realizar compra en la
cuenta de cliente
        List<Detalle> ped = listAdapter.getPedidoDetalles();
        Intent intento = new Intent(AdminVerDetallePedidoActivity.this,
ClientBuyProductsActivity.class);
        Bundle bundle = new Bundle();
        ArrayList<Detalle> lista = new ArrayList<>();
        for (int i = 0; i < ped.size(); i++) {
            lista.add(ped.get(i));
        }
        intento.putExtra("base", "detalle");
        bundle.putSerializable("detalle", lista);
        intento.putExtra("lista", bundle);
        startActivity(intento);

    }else if (opcion.equals("2")){ //Debe regresar a realizar facturación en la
cuenta de administrador
        List<Detalle> ped = listAdapter.getPedidoDetalles();
        Intent intento = new Intent(AdminVerDetallePedidoActivity.this,
AdminFacturacionActivity.class);
        Bundle bundle = new Bundle();
        ArrayList<Detalle> lista = new ArrayList<>();
        for (int i = 0; i < ped.size(); i++) {
            lista.add(ped.get(i));
        }
        intento.putExtra("base", "detalle");

```

```

        bundle.putSerializable("detalle", lista);
        intento.putExtra("lista", bundle);
        startActivity(intento);

    }else{ //Debe regresar a facturas en la cuenta de administrador
        Intent intento = new Intent(AdminVerDetallePedidoActivity.this,
AdminShowFacturaActivity.class);
        startActivity(intento);
    }
    return false;
}
}

```

- **Adaptador**

AdaptadorClientes

```

public class ListClientesAdapter extends
RecyclerView.Adapter<ListClientesAdapter.ViewHolder> {
    private List<Cliente> clientes;
    private LayoutInflater mInflater;
    private Context contex;
    final ListClientesAdapter.OnItemClickListener listener;

    public interface OnItemClickListener{
        void onItemClick(Cliente cli);
    }

    public ListClientesAdapter(List<Cliente> clientes, Context contex,
ListClientesAdapter.OnItemClickListener listener) {
        this.clientes = clientes;
        this.mInflater = LayoutInflater.from(contex);
        this.contex = contex;
        this.listener=listener;
    }
    public void setFilterList(List<Cliente> filterList){
        this.clientes=filterList;
        notifyDataSetChanged();
    }

    @Override
    public ListClientesAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        View view =
mInflater.from(parent.getContext()).inflate(R.layout.lista_clientes, parent,
false); //Relacion con la vista de lista
        return new ListClientesAdapter.ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(final ListClientesAdapter.ViewHolder holder, int
position) {
        holder.cv.setAnimation(AnimationUtils.loadAnimation(contex,R.anim.fade_transition))
;
        holder.bindData(clientes.get(position));
    }

    @Override
    public int getItemCount() {
        return clientes.size();
    }
}

```

```

public void setItems(List<Cliente>items){clientes=items;}

public class ViewHolder extends RecyclerView.ViewHolder{
    ImageView iconImage;
    TextView client,correo,estado;
    CardView cv;

    ViewHolder(View itemView){
        super(itemView);

        iconImage=itemView.findViewById(R.id.iconImageViewCli);
        client=itemView.findViewById(R.id.clientTextViewCli);
        correo=itemView.findViewById(R.id.correoTextViewCli);
        estado=itemView.findViewById(R.id.estadoTextViewCli);
        cv=itemView.findViewById(R.id.cvCli);
    }

    void bindData(final Cliente items){
        iconImage.setColorFilter(Color.rgb(25,200,100),
PorterDuff.Mode.SRC_IN);
        client.setText(items.getApellido()+" "+items.getNombre());
        correo.setText(items.getCorreo());
        estado.setText((items.getEstado().equals("true") ? "activo" :
"inactivo"));
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                listener.OnItemClick(items);
            }
        });
    }
}
}

```

AdaptadorProductos

```

public class ListProductoAdapter1 extends
RecyclerView.Adapter<ListProductoAdapter1.ViewHolder> {
    private List<Producto> productos;
    private LayoutInflater mInflater;
    private Context contex;
    final ListProductoAdapter1.OnItemClickListener listener;

    public interface OnItemClickListener{
        void OnItemClick(Producto prod);
    }

    public ListProductoAdapter1(List<Producto> productos, Context contex,
ListProductoAdapter1.OnItemClickListener listener) {

        this.productos = productos;
        this.mInflater = LayoutInflater.from(contex);
        this.contex = contex;
        this.listener=listener;
    }

    public void setFilterList(List<Producto> filterList){
        this.productos=filterList;
        notifyDataSetChanged();
    }

    @Override
    public ListProductoAdapter1.ViewHolder onCreateViewHolder(ViewGroup parent, int

```

```

viewType) {
    View view =
mInflater.from(parent.getContext()).inflate(R.layout.lista_productos, parent,
false); //Relacion con la vista de lista
    return new ListProductoAdapter1.ViewHolder(view);
}

@Override
public void onBindViewHolder(final ListProductoAdapter1.ViewHolder holder, int
position) {

holder.cv.setAnimation(AnimationUtils.loadAnimation(context,R.anim.fade_transition))
;
    holder.bindData(productos.get(position));
}

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

public void setItems(List<Producto>items){productos=items;}

public class ViewHolder extends RecyclerView.ViewHolder{
    ImageView iconImage;
    TextView description,stock,status,price;
    CardView cv;

    ViewHolder(View itemView){
        super(itemView);

        iconImage=itemView.findViewById(R.id.iconImageView);
        description=itemView.findViewById(R.id.nameTextView);
        stock=itemView.findViewById(R.id.stockTextView);
        status=itemView.findViewById(R.id.statusTextView);
        cv=itemView.findViewById(R.id.cv);
    }

    void bindData(final Producto items){
        iconImage.setColorFilter(Color.rgb(25,200,100),
PorterDuff.Mode.SRC_IN);
        description.setText(items.getDescripcion());
        stock.setText("Stock: "+String.valueOf(items.getStock()));
        status.setText(items.getStatus());
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                listener.OnItemClick(items);
            }
        });
    }
}
}

```

AdaptadorPedidos

```

public class ListPedidosAdapter extends
RecyclerView.Adapter<ListPedidosAdapter.ViewHolder> {
    private List<Pedido> pedidos;
    private LayoutInflater mInflater;
    private Context context;
    final ListPedidosAdapter.OnItemClickListener listener;

    public interface OnItemClickListener{
        void OnItemClick(Pedido prod);
    }
}

```

```

    public ListPedidosAdapter(List<Pedido> pedidos, Context contex,
ListPedidosAdapter.OnItemClickListener listener) {
        this.pedidos = pedidos;
        this.mInflater = LayoutInflater.from(contex);
        this.contex = contex;
        this.listener=listener;
    }
    public void setFilterList(List<Pedido> filterList){
        this.pedidos=filterList;
        notifyDataSetChanged();
    }

    @Override
    public ListPedidosAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        View view =
mInflater.from(parent.getContext()).inflate(R.layout.lista_pedidos, parent, false);
//Relacion con la vista de lista
        return new ListPedidosAdapter.ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(final ListPedidosAdapter.ViewHolder holder, int
position) {

holder.cv.setAnimation(AnimationUtils.loadAnimation(contex,R.anim.fade_transition))
;
        holder.bindData(pedidos.get(position));
    }

    @Override
    public int getItemCount() {
        return pedidos.size();
    }

    public void setItems(List<Pedido>items){pedidos=items;}

    public class ViewHolder extends RecyclerView.ViewHolder{
        TextView client,total,fecha, facturar;
        CardView cv;

        ViewHolder(View itemView){
            super(itemView);

            client=itemView.findViewById(R.id.clientTextViewPed);
            total=itemView.findViewById(R.id.totalTextViewPed);
            fecha=itemView.findViewById(R.id.dateTextViewPed);
            facturar=itemView.findViewById(R.id.estadoFacturado);
            cv=itemView.findViewById(R.id.cvPed);
        }

        void bindData(final Pedido items){
            client.setText(items.getApellidoCliente()+"
"+items.getNombreCliente());
            total.setText("Total $: "+items.getTotal());
            fecha.setText(items.getFecha());
            facturar.setText((items.getFacturar().equals("si")) ? "facturado": "sin
facturar");
            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    listener.OnItemClick(items);
                }
            });
        }
    }

```

```

    }
    });
}
}
}

```

AdaptadorFacturas

```

public class ListFacturasAdapter extends
RecyclerView.Adapter<ListFacturasAdapter.ViewHolder> {
    private List<Factura> facturas;
    private LayoutInflater mInflater;
    private Context contex;
    final ListFacturasAdapter.OnItemClickListener listener;

    public interface OnItemClickListener{
        void onItemClick(Factura prod);
    }

    public ListFacturasAdapter(List<Factura> facturas, Context contex,
ListFacturasAdapter.OnItemClickListener listener) {
        this.facturas = facturas;
        this.mInflater = LayoutInflater.from(contex);
        this.contex = contex;
        this.listener=listener;
    }
    public void setFilterList(List<Factura> filterList){
        this.facturas=filterList;
        notifyDataSetChanged();
    }

    @Override
    public ListFacturasAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        View view =
mInflater.from(parent.getContext()).inflate(R.layout.lista_pedidos, parent, false);
//Relacion con la vista de lista
        return new ListFacturasAdapter.ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(final ListFacturasAdapter.ViewHolder holder, int
position) {
holder.cv.setAnimation(AnimationUtils.loadAnimation(contex,R.anim.fade_transition))
;
        holder.bindData(facturas.get(position));
    }

    @Override
    public int getItemCount() {
        return facturas.size();
    }

    public void setItems(List<Factura>items){facturas=items;}

    public class ViewHolder extends RecyclerView.ViewHolder{
        TextView client,total,fecha, despachado;
        CardView cv;
        ViewHolder(View itemView){
            super(itemView);

            client=itemView.findViewById(R.id.clientTextViewPed);
            total=itemView.findViewById(R.id.totalTextViewPed);
            fecha=itemView.findViewById(R.id.dateTextViewPed);

```

```

        despachado=itemView.findViewById(R.id.estadoFacturado);
        cv=itemView.findViewById(R.id.cvPed);
    }

    void bindData(final Factura items){
        client.setText(items.getApellidoCliente()+"
"+items.getNombreCliente());
        total.setText("Total $: "+items.getTotal());
        fecha.setText(items.getFecha());
        despachado.setText((items.getDespachado().equals("si")) ? "despachado":
"sin despachar");
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                listener.OnItemClick(items);
            }
        });
    }
}
}
}

```

AdaptadorDetalle

```

public class ListDetallePedAdapter extends
RecyclerView.Adapter<ListDetallePedAdapter.ViewHolder> {
    private List<Detalle> pedidoDetalles;
    private LayoutInflater mInflater;
    private Context contex;
    int opcion;
    ListDetallePedAdapter.OnItemClickListener listener;

    public interface OnItemClickListener{
        void OnItemClick(Detalle cli, int position);
    }

    public ListDetallePedAdapter(int opcion, List<Detalle> PedidoDetalles, Context
contex, ListDetallePedAdapter.OnItemClickListener listener) {
        this.pedidoDetalles = PedidoDetalles;
        this.mInflater = LayoutInflater.from(contex);
        this.contex = contex;
        this.listener=listener;
        this.opcion=opcion;
    }

    @Override
    public ListDetallePedAdapter.ViewHolder onCreateViewHolder(ViewGroup parent,
int viewType) {
        View view =
mInflater.from(parent.getContext()).inflate(R.layout.lista_detalle_pedido, parent,
false); //Relacion con la vista de lista
        return new ListDetallePedAdapter.ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(final ListDetallePedAdapter.ViewHolder holder, int
position) {
        holder.cv.setAnimation(AnimationUtils.loadAnimation(contex,R.anim.fade_transition))
;
        holder.bindData(pedidoDetalles.get(position),position);
    }
}

```

```

@Override
public int getItemCount() {
    return pedidoDetalles.size();
}

public List<Detalle> getPedidoDetalles() {
    return pedidoDetalles;
}

public void setItems(List<Detalle>items){pedidoDetalles=items;}

public class ViewHolder extends RecyclerView.ViewHolder{
    TextView descripcion,cantidad,subtotal,eliminar;
    CardView cv;

    ViewHolder(View itemView){
        super(itemView);
        descripcion=itemView.findViewById(R.id.descripcionDetalle);
        cantidad=itemView.findViewById(R.id.cantidadDetalle);
        subtotal=itemView.findViewById(R.id.subTotalDetalle);
        eliminar=itemView.findViewById(R.id.btnEliminarDetPedido);
        cv=itemView.findViewById(R.id.cvDet);
    }

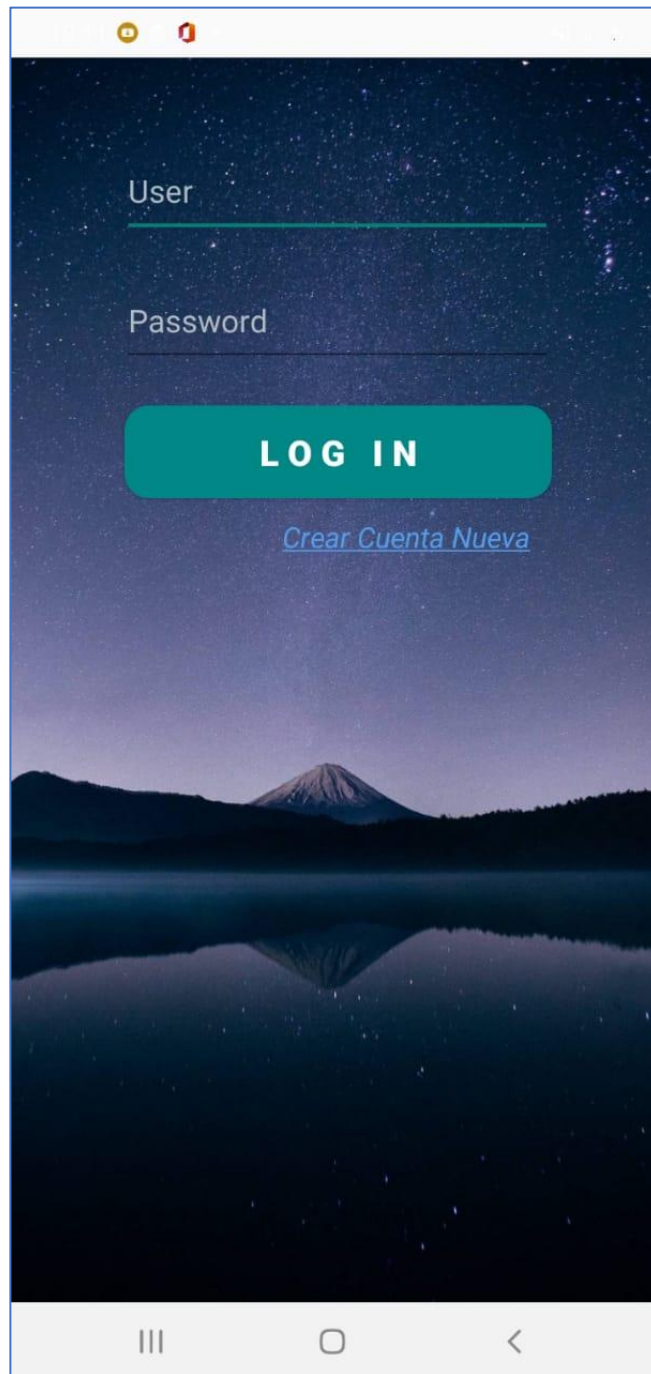
    void bindData(final Detalle items, int position){
        eliminar.setVisibility((opcion==0) ? View.GONE : View.VISIBLE);
        descripcion.setText(items.getDescripcionProducto());
        cantidad.setText(items.getCantidad()+"");
        subtotal.setText(items.getSubtotal());

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                listener.OnItemClick(items,position);
            }
        });
    }
}
}

```

5. Funcionamiento.

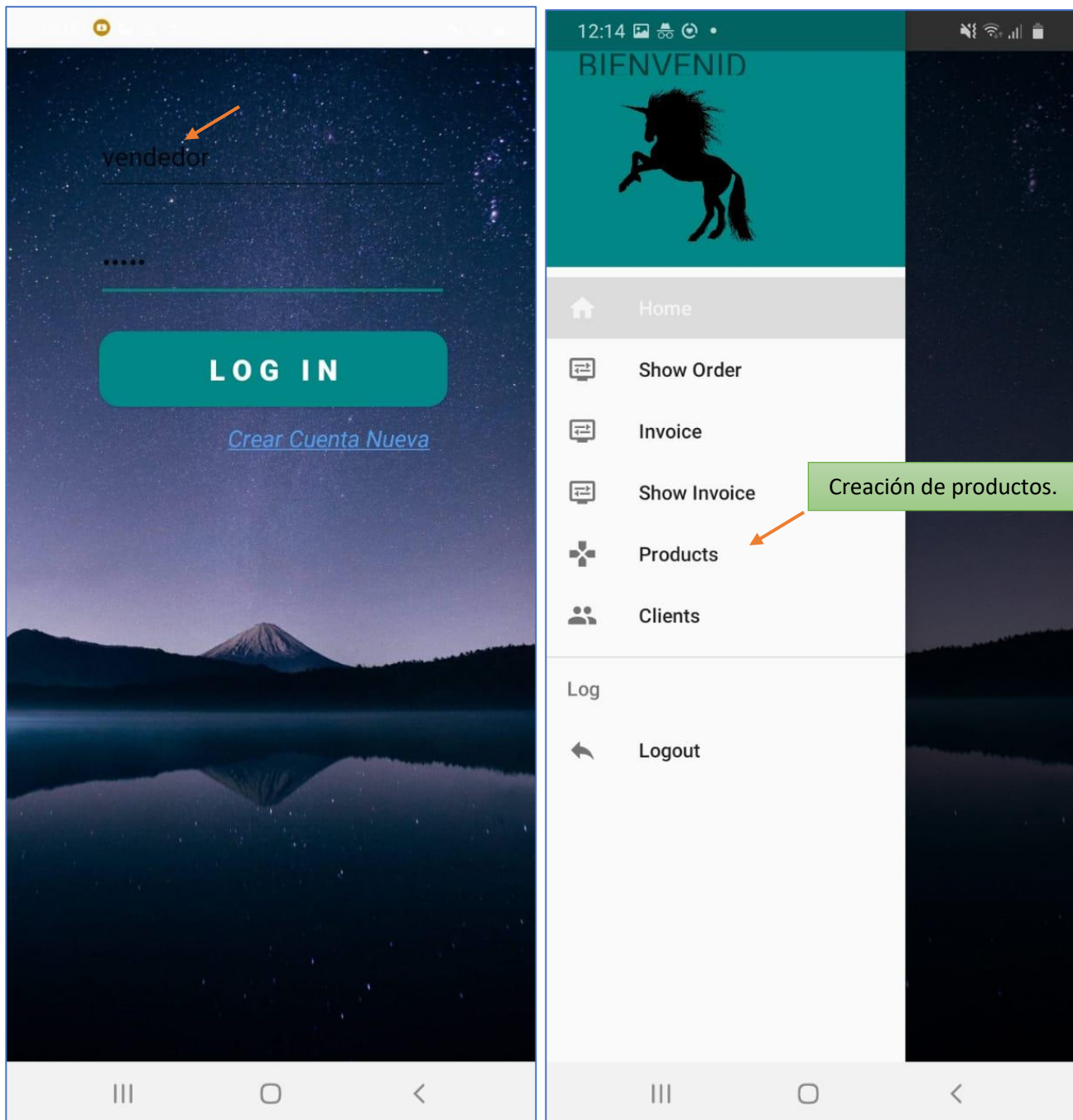
- Vista Login.



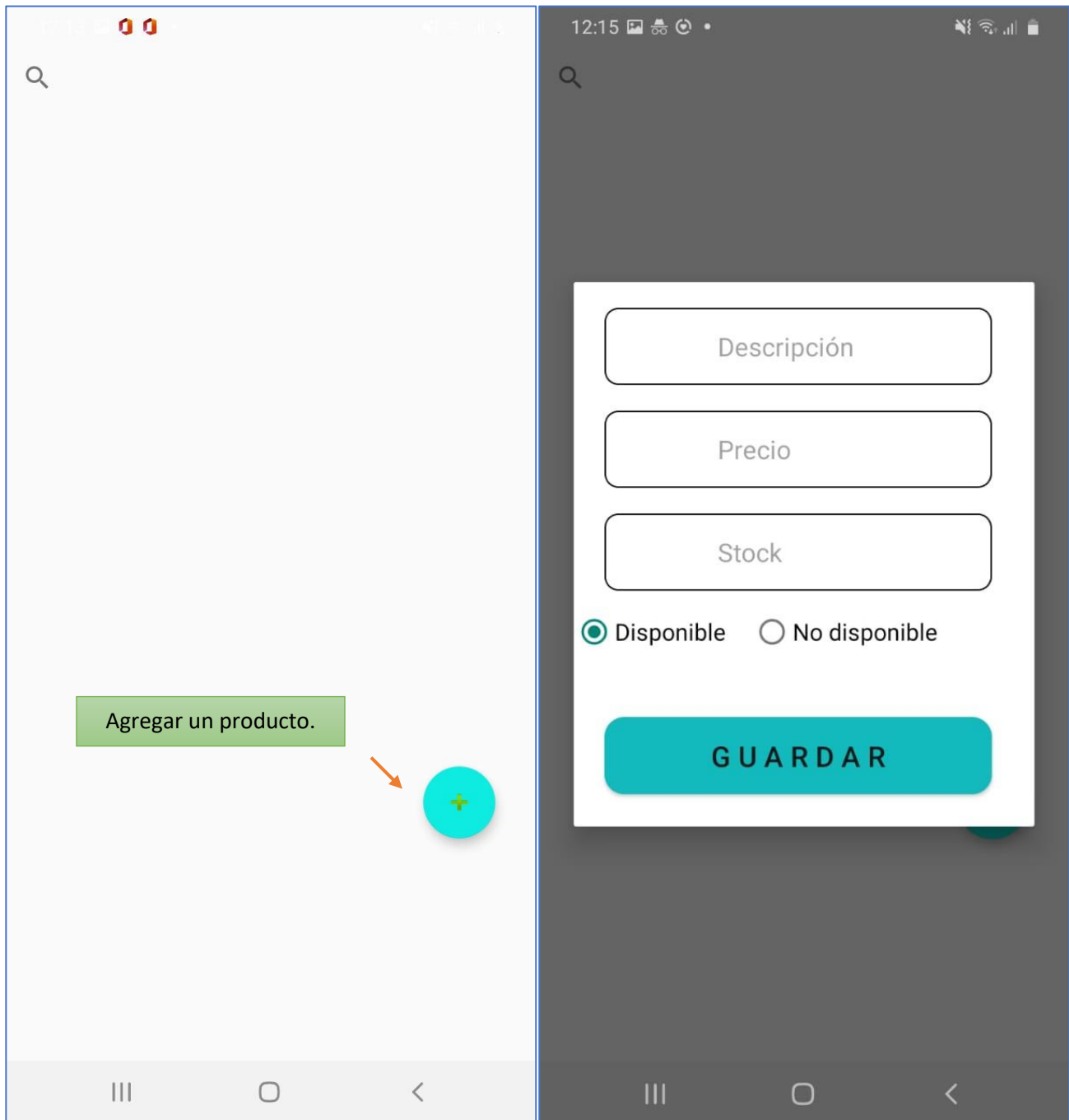
- Inicio de sesión con la cuenta del “Vendedor”.

Nota: El Vendedor ya tiene una cuenta por defecto.

Nota: El Vendedor puede crear productos.



- En la cuenta del Vendedor se procede a la creación de un producto.



12:15

Mouse

8

50

☒ Disponible ☐ No disponible

GUARDAR

días minutos

1 2 3 4 5 6 7 8 9 0

q w e r t y u i o p

a s d f g h j k l ñ

↑ z x c v b n m

!#1 ! Español (US) . Done

Se ha agregado el producto.

Mouse
Stock: 50 disponible

Parlantes
Stock: 50 disponible

Teclado
Stock: 50 disponible

Usb
Stock: 50 disponible

+

- Actualización de un producto.

Nota: Para ello se debe pulsar sobre el producto que se desea modificar.

12:17

Se cargan los datos.

Producto: 1

Mouse

10

55

☒ disponible ☐ No disponible

ACTUALIZAR ELIMINAR

Se ha actualizado el stock del producto.

Mouse Stock: 55 disponible

Parlantes Stock: 50 disponible

Teclado Stock: 50 disponible

Usb Stock: 50 disponible

Producto actualizado correctamente

- Eliminación de un producto.

Nota: De igual forma se debe pulsar sobre algún producto y se tiene la opción de eliminar.

11:42

Producto: 5

Usb

50

50

☒ disponible ☐ No disponible

ACTUALIZAR ELIMINAR

Mouse Stock: 55 disponible

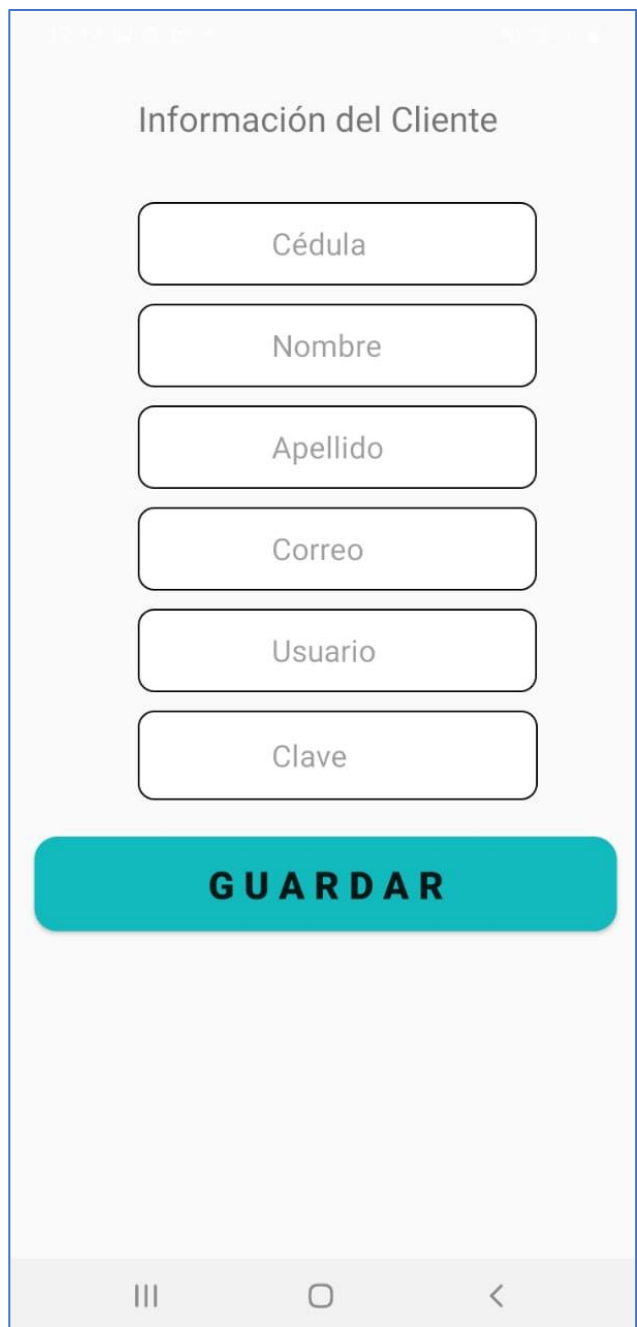
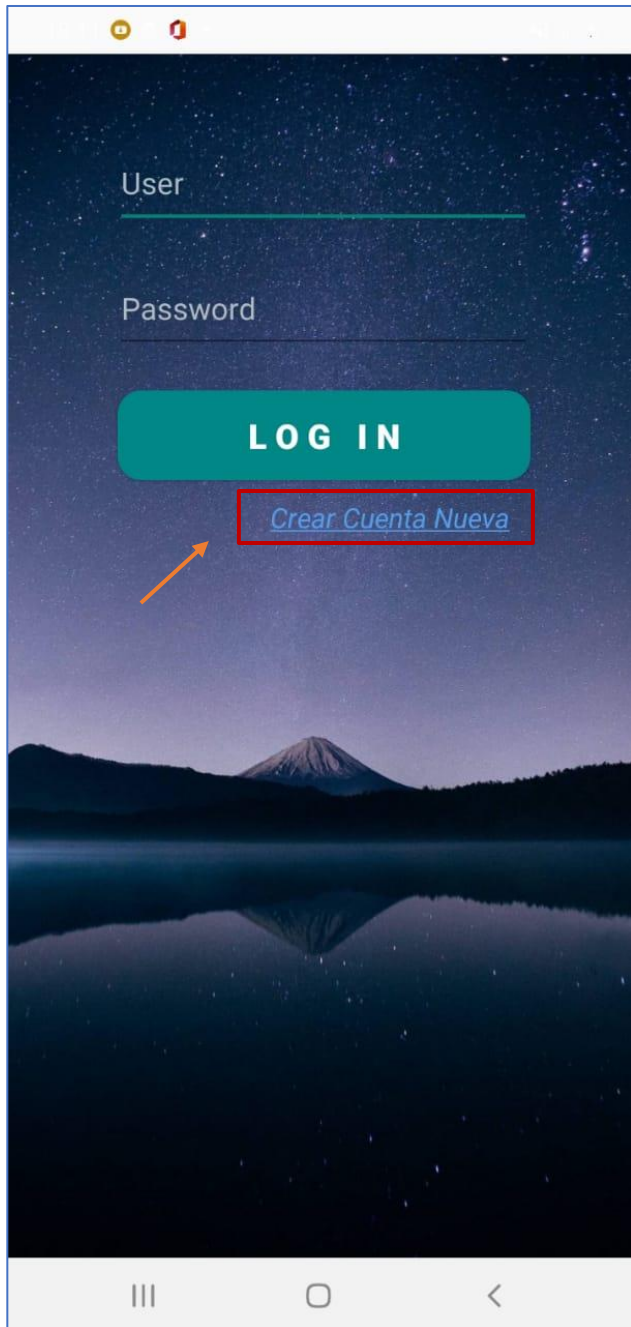
Parlantes Stock: 50 disponible

Teclado Stock: 50 disponible

Se ha eliminado el producto Usb.

Producto eliminado correctamente

- Creación de una cuenta de cliente.



Información del Cliente

1004660138

Luis

Lopez

luis@gmail.com

luis

luis

GUARDAR

Iniciar sesión con la
cuenta del cliente.

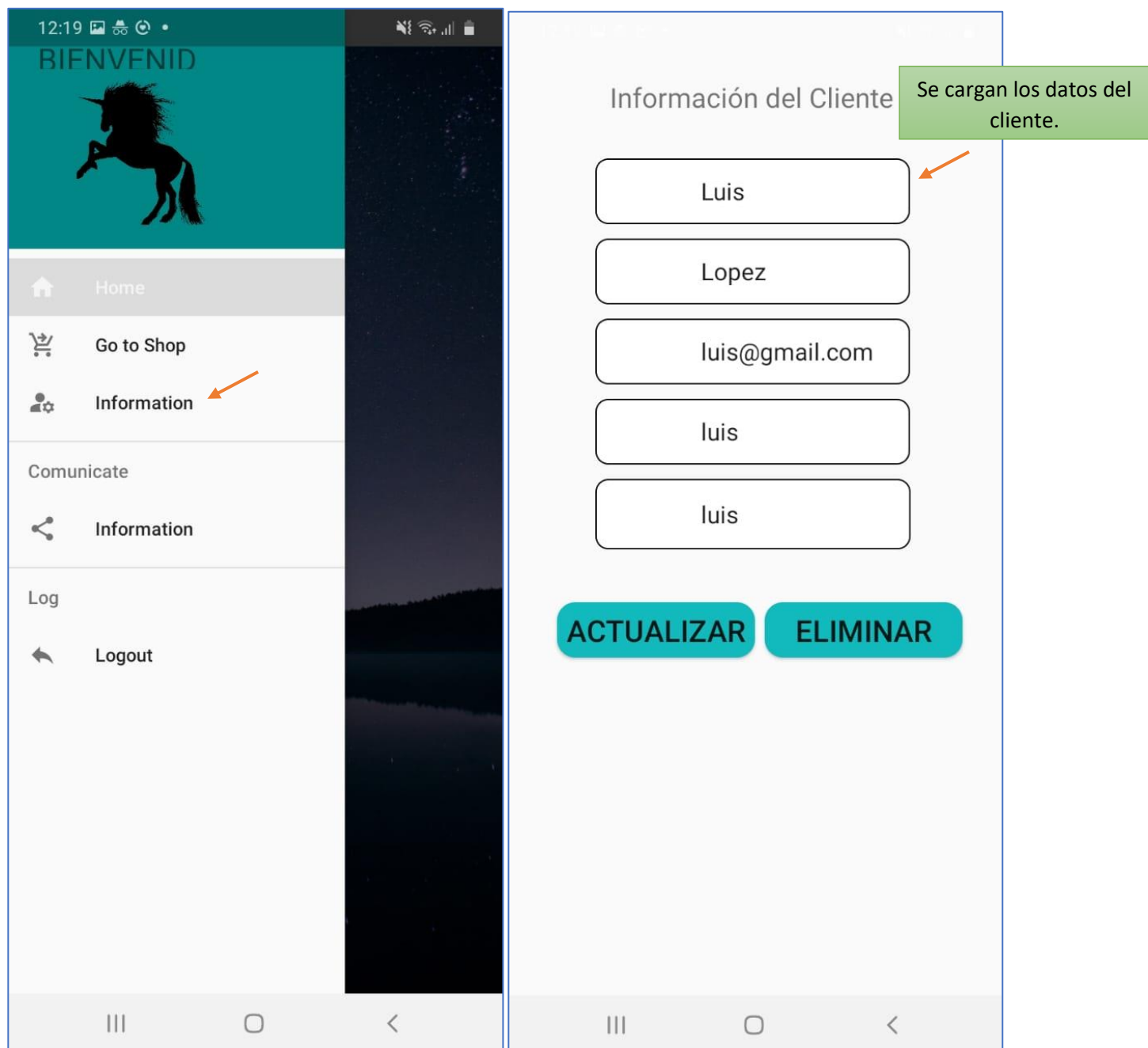
Luis

....

LOG IN

[Crear Cuenta Nueva](#)

- Dentro de la cuenta de cliente se puede actualizar los datos de este.



Información del Cliente

Luis

Yepez

luis@gmail.com

luis

luis

ACTUALIZAR

ELIMINAR

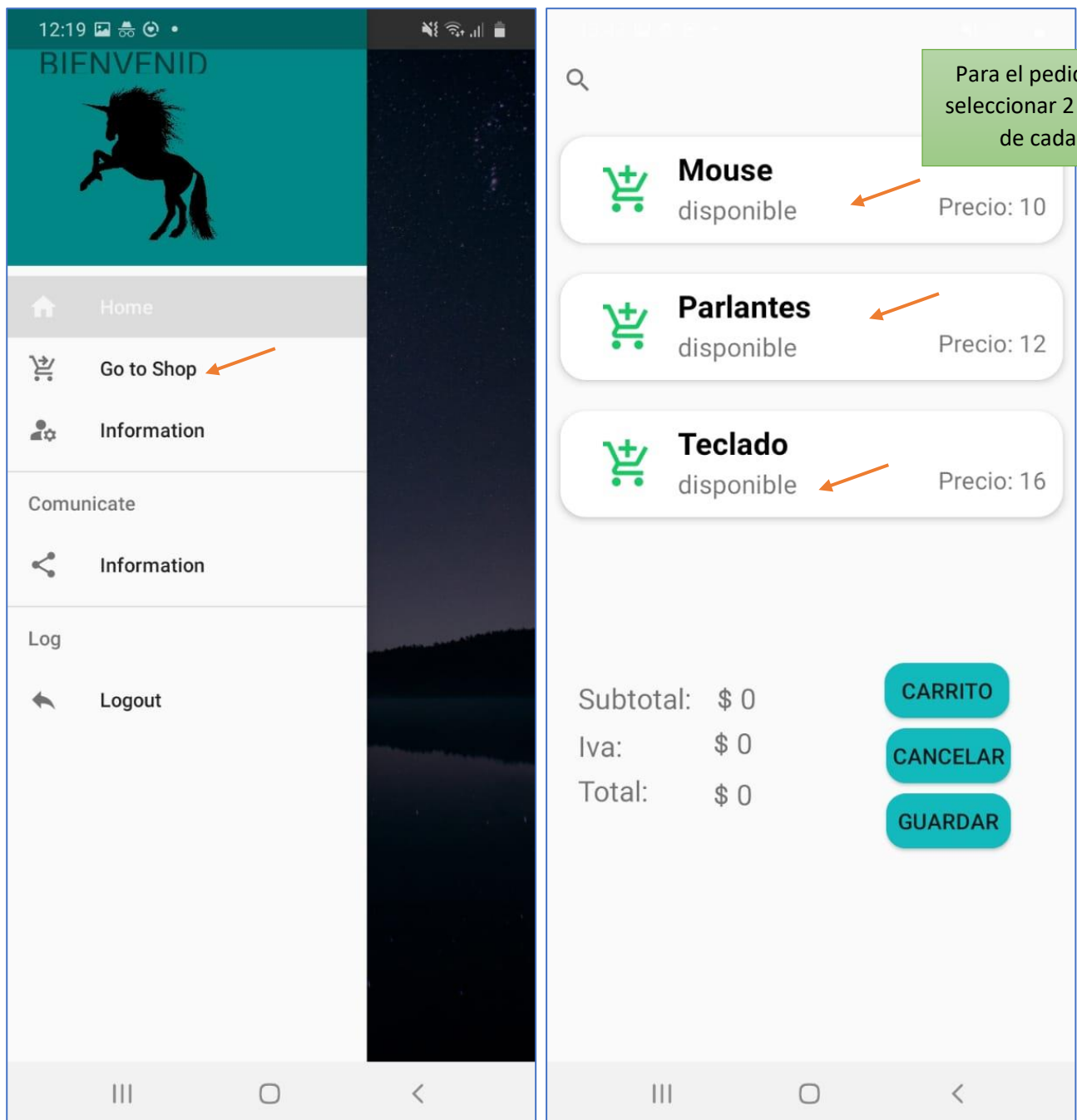
Cliente actualizado correctamente

Se modificará el apellido del cliente.

Nota: Más adelante se verá el cliente actualizado.

- Dentro de la cuenta del cliente se procede a realizar el pedido de compra.

Nota: Seleccionar los productos deseados.



Nota: Se desea realizar un pedido de 2 Mouse, 2 Parlantes y 2 Teclados.



Nota: Vemos el detalle del pedido, aquí, se puede eliminar un producto del carrito si se desea.

Detalle

Num Pedido:

Fecha: 20-11-2022

Cliente: Yepez Luis

Cédula: 1004660138

descripción	cantidad	subtotal	
Mouse	2	20.0	X
Parlantes	2	24.0	X
Teclado	2	32.0	X

Subtotal: 76.0

Iva: 9.12

Total: 85.12

Eliminar detalle de pedido.

Regresar

Mouse

disponible

Precio: 10

Parlantes

disponible

Precio: 12

Teclado

disponible

Precio: 16

Subtotal:

Iva:

Total:

CARRITO

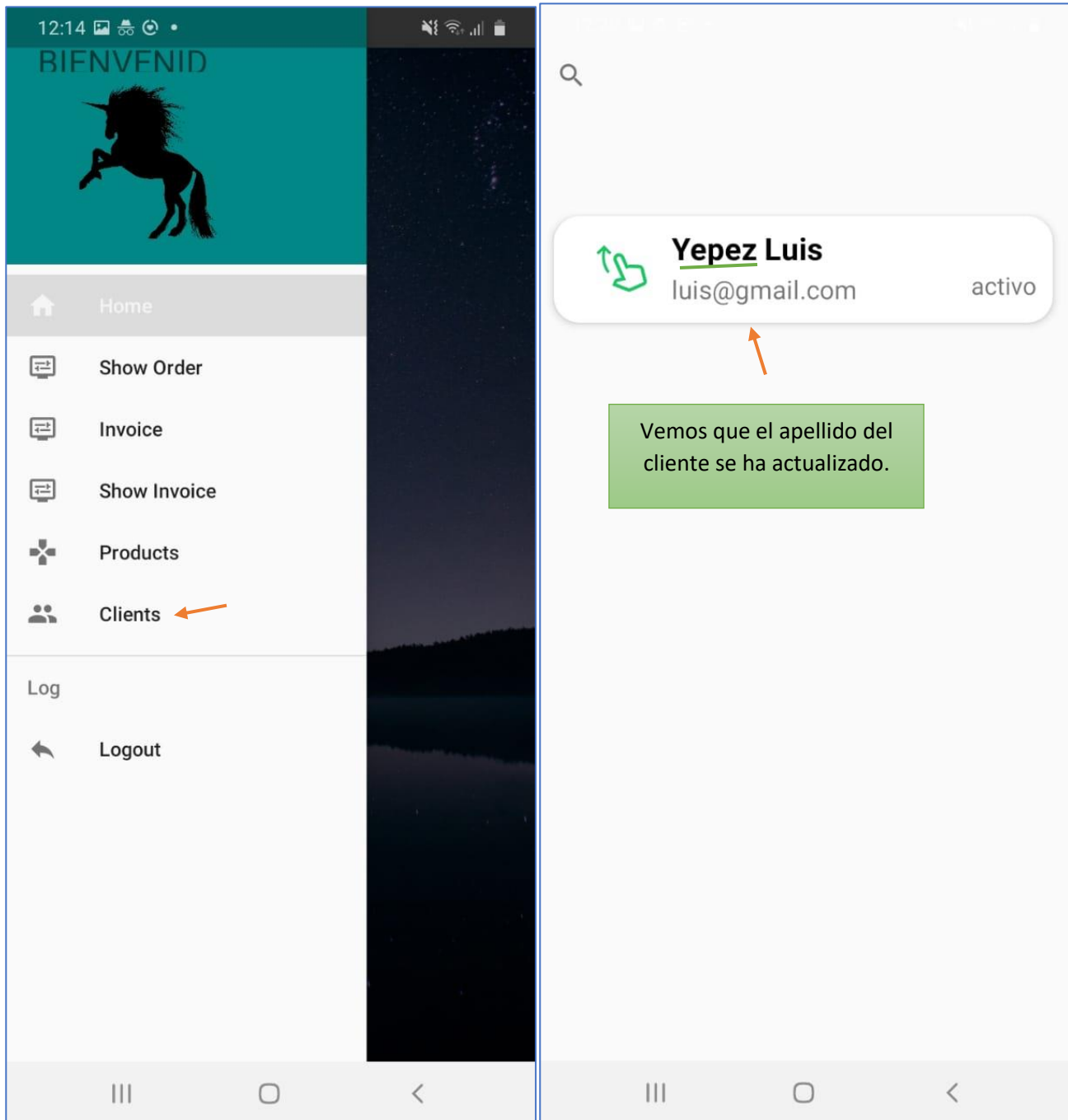
CANCELAR

GUARDAR

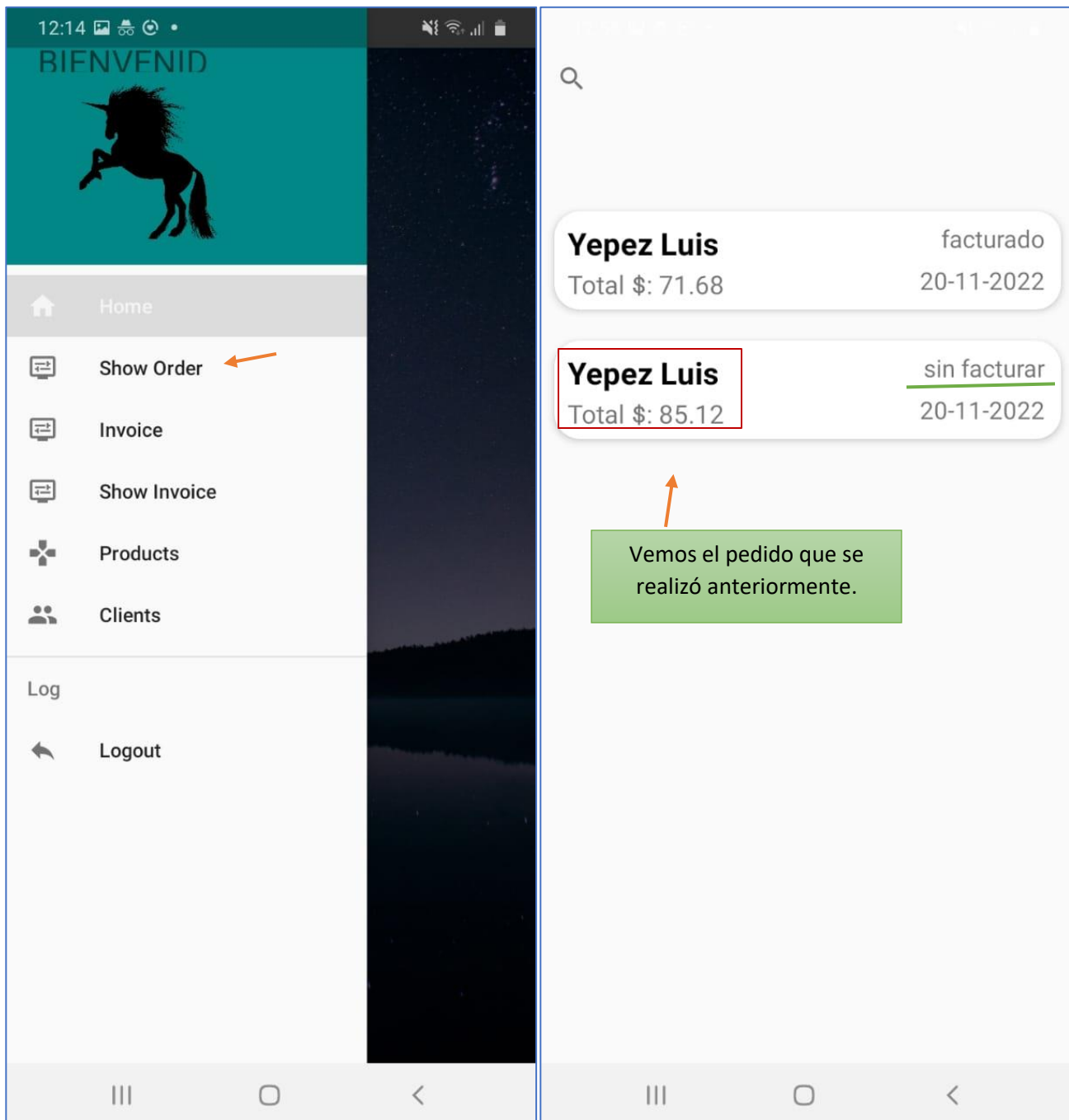
Pedido guardado

Finalmente se guarda el pedido.

- Después nuevamente se inicia sesión con la cuenta de “Vendedor”, allí se puede ver todos los clientes del sistema.

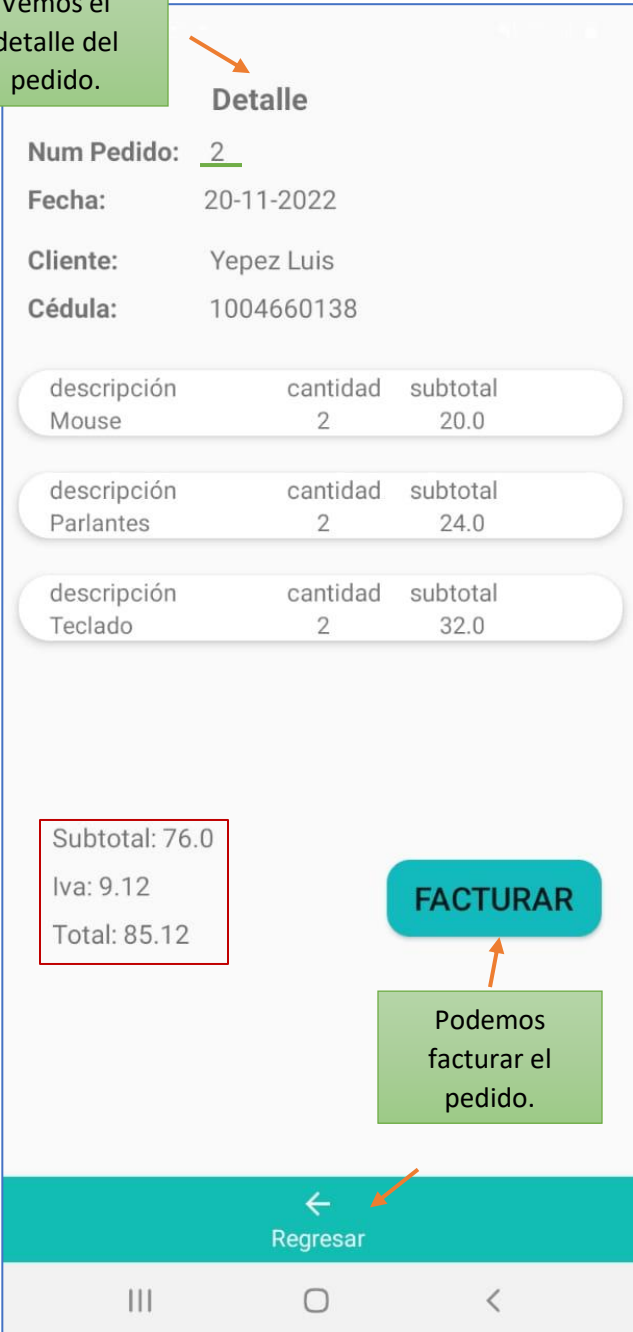


- Dentro de la cuenta del Vendedor se puede ver los pedidos realizados por los clientes.



Si seleccionamos ese pedido, vemos el detalle. También la existe la opción de seleccionar **facturar**. “Es decir que el pedido se va a facturar”.

Vemos el detalle del pedido.



Detalle

Num Pedido: 2

Fecha: 20-11-2022

Cliente: Yepez Luis

Cédula: 1004660138

descripción	cantidad	subtotal
Mouse	2	20.0
Parlantes	2	24.0
Teclado	2	32.0

Subtotal: 76.0


Iva: 9.12

Total: 85.12

FACTURAR

Podemos facturar el pedido.

← Regresar



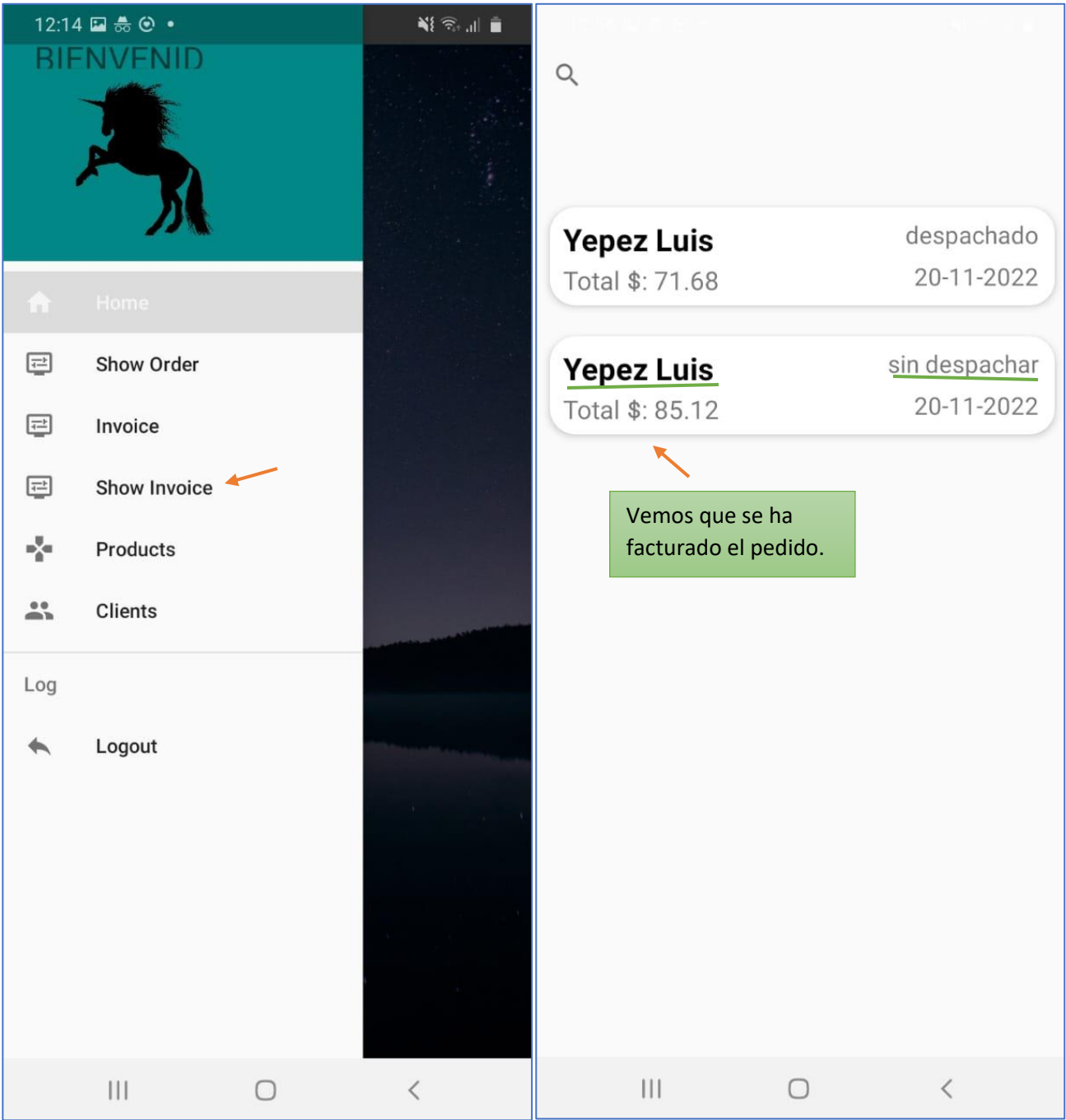
Yepez Luis facturado
Total \$: 71.68 20-11-2022

Yepez Luis facturado
Total \$: 85.12 20-11-2022

El pedido ya ha sido facturado y se ha reducido el stock de los productos.

Pedido facturado

Si vamos a ver las facturas, ese pedido ya estará ahí.



Si seleccionamos esa factura se puede ver el detalle de este. El sistema tiene la posibilidad de **despachar** la factura. “Es decir que los productos ya fueron entregados”.

Factura # 2

Detalle

Num Factura: 2

Fecha: 20-11-2022

Cliente: Yepez Luis

Cédula: 1004660138

descripción	cantidad	subtotal
Mouse	2	20.0
Parlantes	2	24.0
Teclado	2	32.0

Subtotal: 76.0
Iva: 9.12
Total: 85.12

DESPACHAR

←
Regresar

Factura # 2

Yepez Luis despachado

Total \$: 71.68 20-11-2022

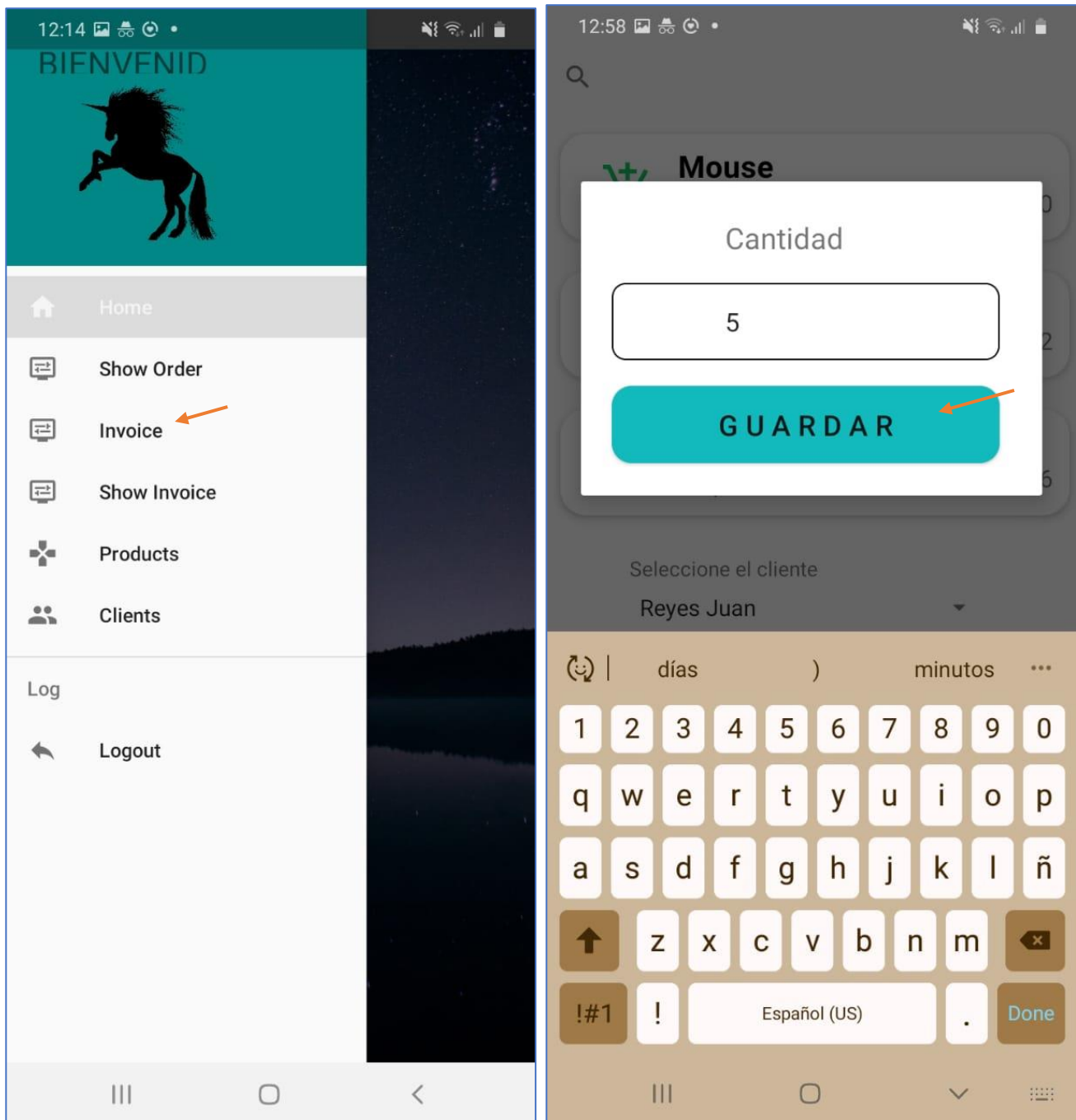
Yepez Luis despachado

Total \$: 85.12 20-11-2022


Factura despachado

El sistema también tiene la opción de facturación de productos.


Nota: Se seleccionarán 5 Parlantes.




🔍

**Mouse**
disponible

Precio: 10

**Parlantes**
disponible

Precio: 12

**Teclado**
disponible

Precio: 16

Seleccione el cliente

Reyes Juan

▼

Subtotal: 60.0

Iva: 7.19999

Total: 67.2

DETALLE

CANCELAR

GUARDAR

Detalle

Num Factura:

Fecha: 20-11-2022

Cliente:

Cédula:

descripción	cantidad	subtotal	
Parlantes	5	60.0	X

Subtotal: 60.0

Iva: 7.2

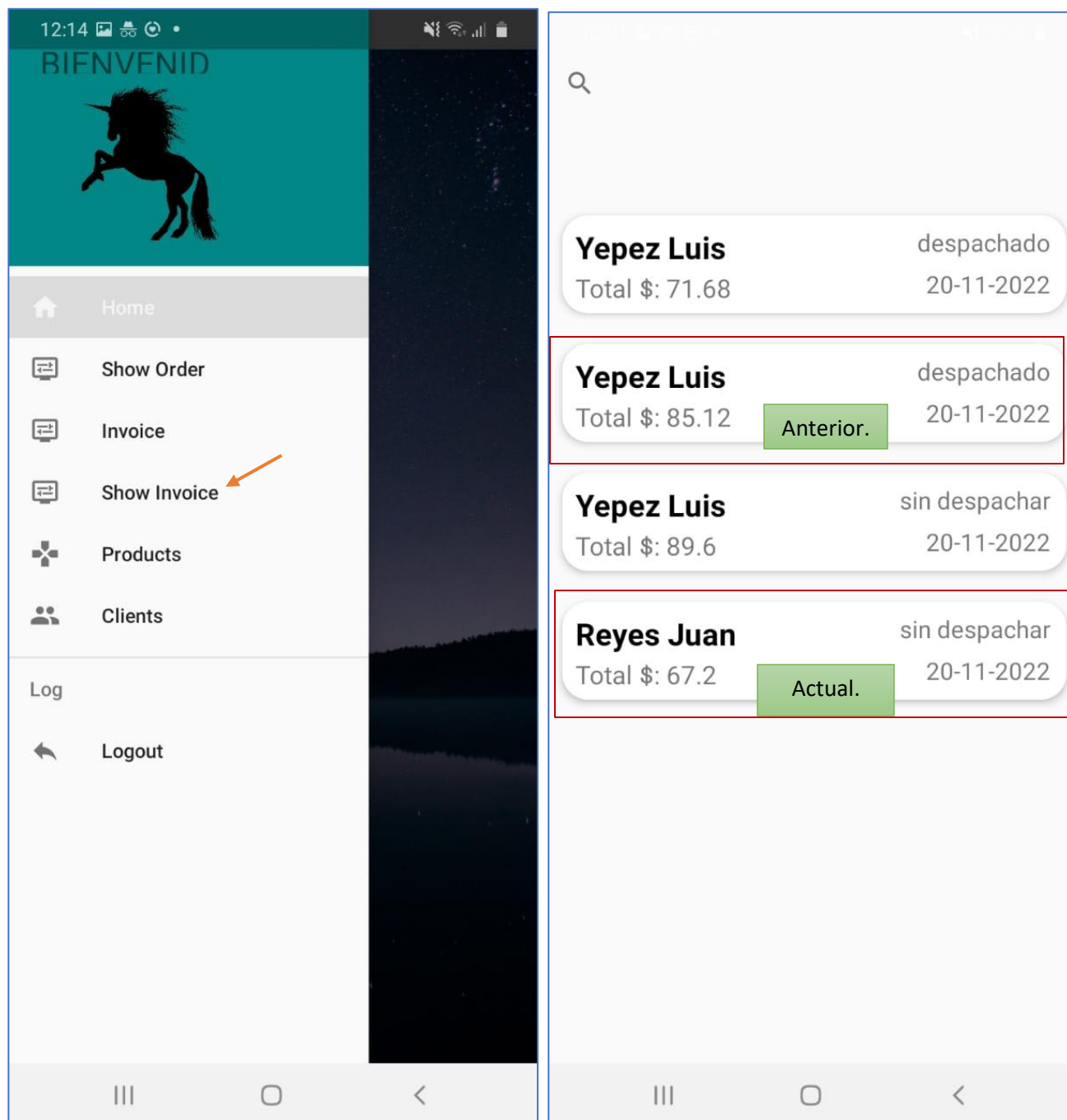
Total: 67.2

←

Regresar

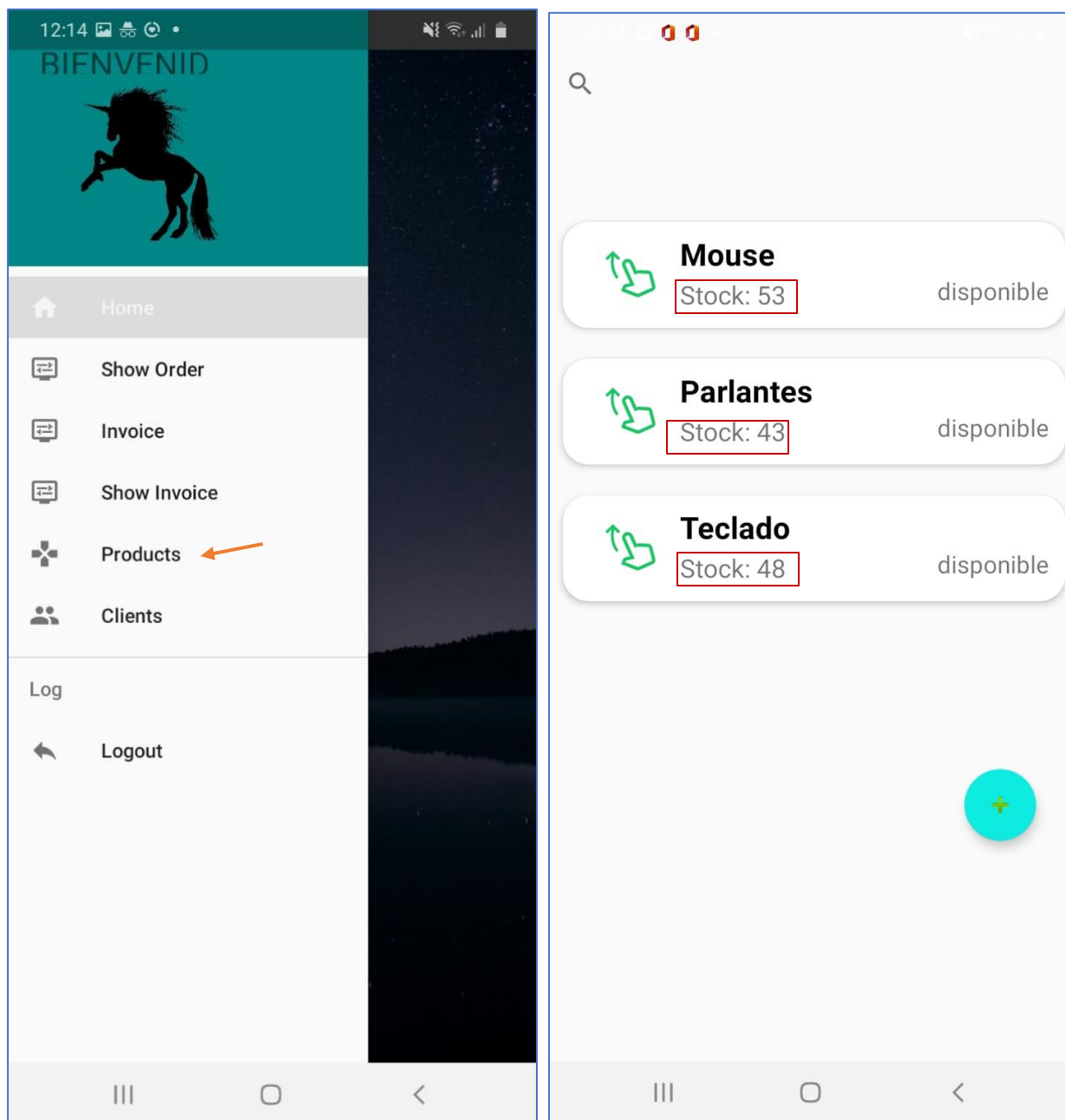
Nota: Finalmente guardamos la factura.

Luego, si vamos a la lista de las facturas, podemos ver el que se acaba de crear.



Nota: Se puede ver el detalle de la factura, y el despacho del mismo.

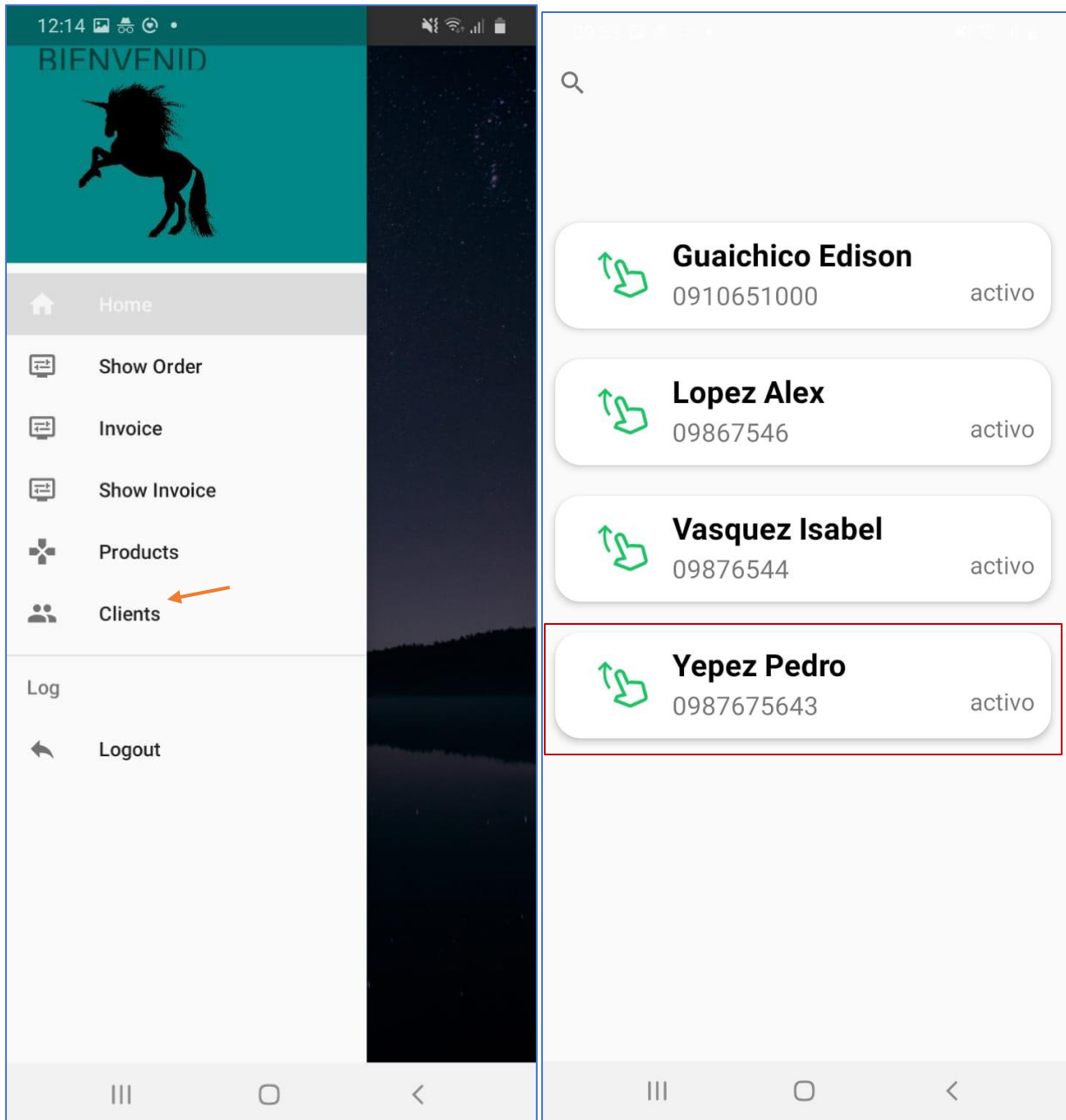
- Reducción de stock de los productos comprados.



Otras funcionalidades del sistema

- Eliminación de un cliente.

Clientes disponibles, en este caso se eliminará el **cliente PEDRO YEPEZ**.



Para esto se debe iniciar sesión con la cuenta del cliente a eliminar. En la opción **Information**, existe la posibilidad y Actualizar y **Eliminar** Cliente.

The image displays two mobile application screens side-by-side.

Left Screen: Información del Cliente

This screen is titled "Información del Cliente". It contains five input fields for client data, stacked vertically:

- First field: "Pedro" (highlighted with a red border)
- Second field: "Yepez"
- Third field: "0987675643"
- Fourth field: "pedro"
- Fifth field: "pedro"

At the bottom of the form are two buttons:

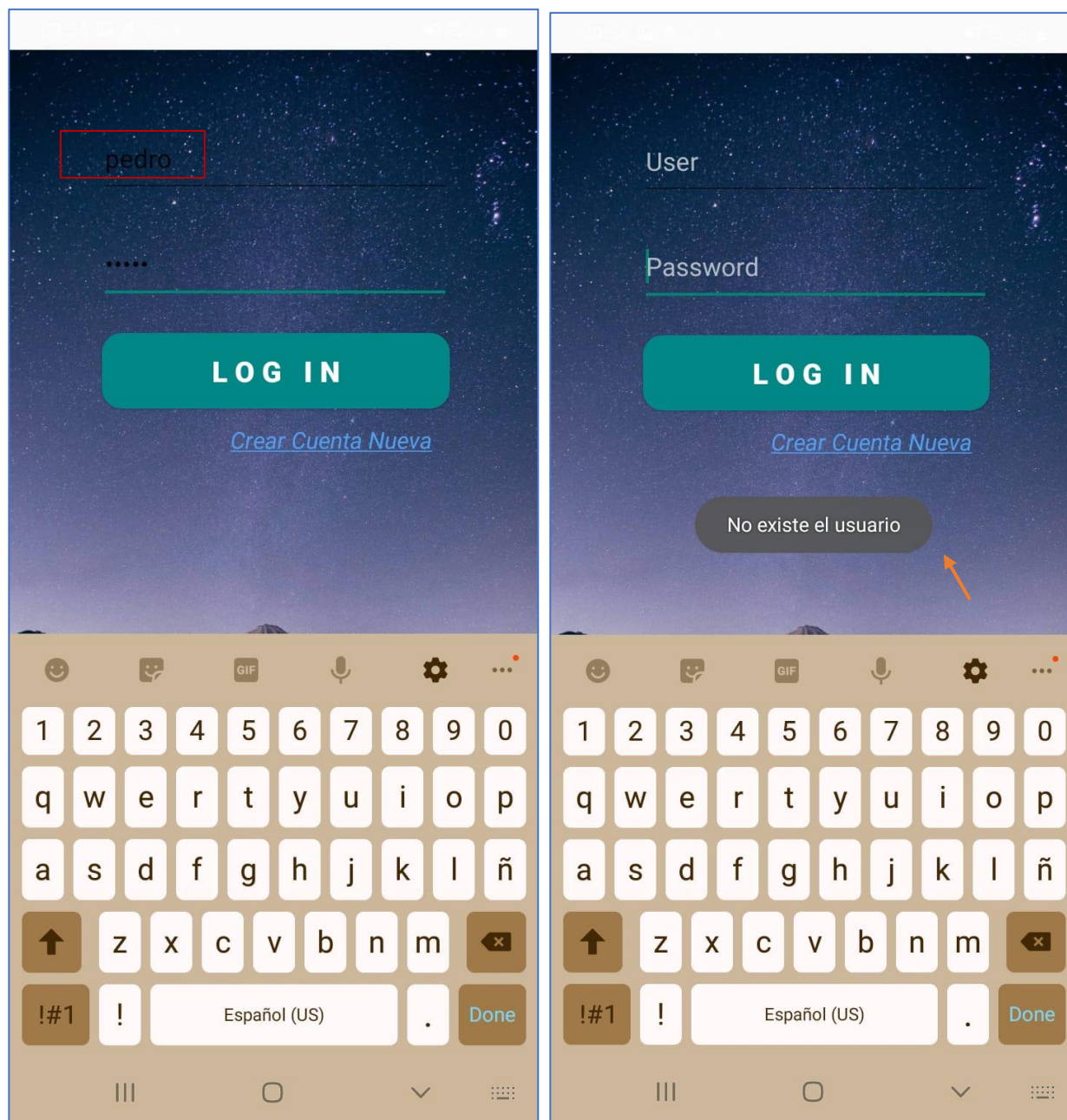
- "ACTUALIZAR" (teal button)
- "ELIMINAR" (teal button, highlighted with a red border)

Right Screen: Login and Success

This screen features a login form with a dark, starry background and a mountain landscape at the bottom. It includes:

- A header bar with a logo and a dropdown arrow.
- "User" and "Password" input fields.
- A large teal "LOG IN" button.
- A link labeled "Crear Cuenta Nueva" in blue text.
- A success message in a grey box: "Cliente eliminado correctamente".

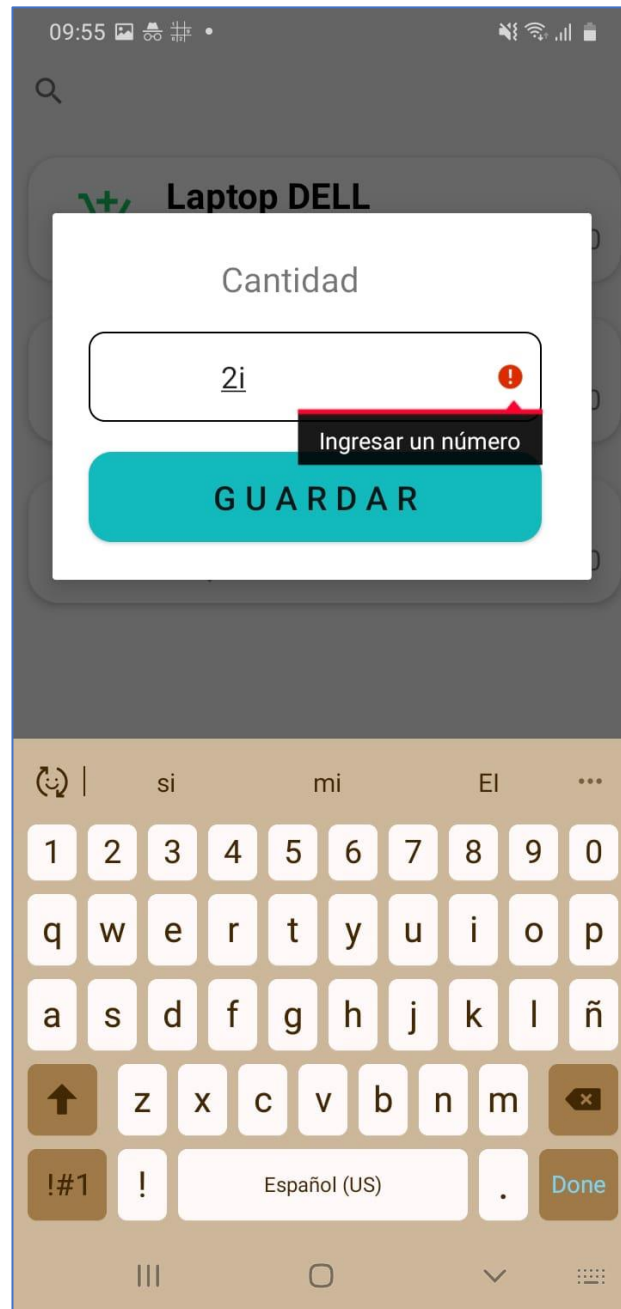
Después, intentar iniciar sesión con la cuenta del cliente que se ha eliminado.



Nota: El cliente ya no existe, es decir que se ha eliminado correctamente.

- Validación de campos.

En la ventana donde se crea el pedido o se crea la factura, se necesita un valor de entrada tipo numérico, así que validamos esta situación.



• Conclusión

El desarrollo de esta aplicación ha permitido conocer más a detalle la estructura y el flujo de navegación de los proyectos dentro de Android Studio. Por otra parte, diseñar la vista en este software personalmente es mucho más complejo, pues se utiliza xml para ese fin. Pese a esta situación se ha realizado una vista bastante amigable que permite manejar la aplicación de forma muy sencilla, satisfaciendo las expectativas del cliente.