Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones

Primer Parcial - 3 capas

PROYECTO PARA EMPRENDEDOR "Megapet"

CARRERA: INGENIERIA EN SISTEMAS

MATERIA: INF-552 ARQUITECTURA DE SOFTWARE

DOCENTE: ING. VEIZAGA GONZALES JOSUE OBED

ESTUDIANTE: :

PERICENA CHOQUE LUISHIÑO

GESTION: 2-2024

SANTA CRUZ – BOLIVIA

Tabla de contenido

1 Descripcion Del Problema

Muchos emprendedores enfrentan desafíos diarios que dificultan el crecimiento y la eficiencia de sus negocios, especialmente en la gestión de pedidos. Sin un sistema automatizado, se ven forzados a realizar tareas clave de manera manual, lo que no solo consume tiempo, sino que también aumenta el margen de error y la frustración. Estos problemas incluyen:

Gestión de Pedidos Sin Herramientas: La falta de una plataforma centralizada para manejar los pedidos genera confusión, retrasos y posibles errores. Los emprendedores tienen dificultades para mantener el control de sus productos, inventarios y clientes, lo que impacta directamente en la experiencia del cliente y en la imagen de su negocio.

Envío Manual de Información al Repartidor: Cada pedido y ubicación de entrega deben ser enviados manualmente a los repartidores, lo que no solo es un proceso lento, sino que también aumenta las posibilidades de errores en la entrega, generando insatisfacción tanto en los clientes como en los repartidores.

Generación y Envío de Documentos: Las proformas y listas de entrega se crean manualmente y son enviadas como PDF a los clientes, lo que enlentece el ciclo de ventas y desorganiza el seguimiento de los pedidos. Este proceso no solo consume tiempo valioso, sino que también limita la capacidad del emprendedor para agilizar y expandir su negocio.

En un mundo cada vez más digital, estos obstáculos impiden a los emprendedores competir de manera eficiente, impidiendo que su negocio crezca al ritmo que podrían lograr con un sistema más optimizado.

.

2 2. Objetivos

2.1 Objetivo General

Desarrollar una aplicación móvil con una base de datos local que permita a los emprendedores gestionar sus pedidos de manera más eficiente. La aplicación facilitará la creación de catálogos de productos, el envío de pedidos y la coordinación de entregas a través de WhatsApp, todo desde una plataforma centralizada.

Además, automatizará la generación de documentos como proformas y listas de entrega en formato PDF, permitiendo su envío ágil por WhatsApp. Esto mejorará la organización, el seguimiento y la agilidad de los procesos del negocio.

2.2 Objetivos Específicos

- Desarrollar una base de datos local que almacene de manera segura la información de los clientes y productos, incluyendo detalles de contacto, historial de compras y preferencias, permitiendo un acceso rápido y eficiente.
- Implementar un sistema para la creación y gestión de pedidos y catálogos de productos, permitiendo a los emprendedores organizar y actualizar su inventario de manera sencilla, categorizando productos según su disponibilidad y demanda.
- Automatizar la generación y envío de archivos PDF con detalles de pedidos y proformas, permitiendo su distribución rápida a través de plataformas de mensajería como WhatsApp, mejorando la comunicación con los clientes y agilizando el proceso de ventas.

- Habilitar el envío de ubicaciones GPS a través de WhatsApp para facilitar la coordinación de entregas, permitiendo que los emprendedores compartan fácilmente los puntos de entrega con los repartidores.
- Diseñar una interfaz intuitiva y fácil de usar para emprendedores y repartidores, asegurando que la plataforma les permita gestionar de manera eficiente sus productos, pedidos y entregas sin necesidad de conocimientos técnicos avanzados.

3 Captura de Requisitos

3.1 Identificar casos de uso

CU1: Gestionar Cliente

CU2: Gestionar Repartidor

CU3: gestionar Entrega

CU4: Gestionar Producto

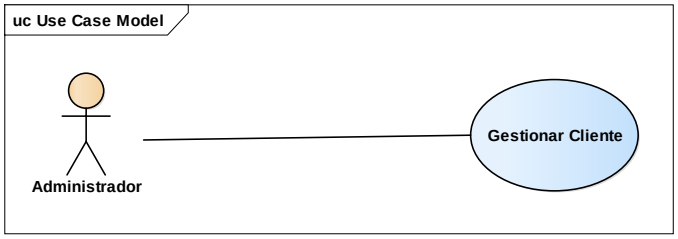
CU5: Registrar Catalogo

CU6: Gestionar Proforma

CU7: Gestionar DetalleCatalogo

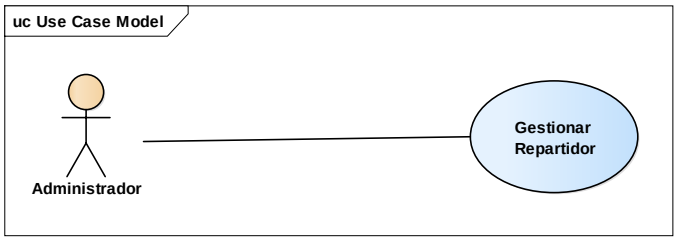
3.2 Detalle de caso de uso

3.2.1 CU1: Gestionar Cliente



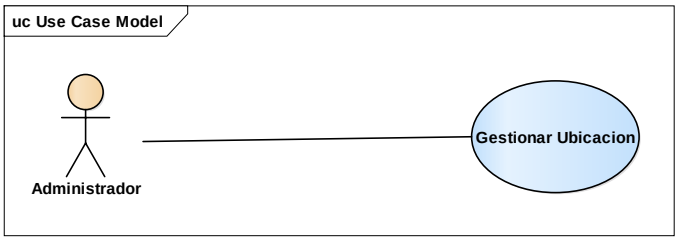
CASO DE USO	CU1: Gestionar Cliente
Propósito	Administrar cliente
Descripción	Permite gestionar a los clientes de la aplicación
Actores	Administrador
Actor Iniciador	Administrador
Precondición	Ninguna
Proceso	1. Agregar Cliente 1.1. insertar datos 1.2. guardar datos 2. Modificar Cliente 2.1. Seleccionar Cliente 2.2. Modificar datos 2.3. Guardar cambios 3. Eliminar Cliente 3.1. Seleccionar Cliente 3.2. Modificar datos 3.3. Guardar cambios
Postcondición	Ninguna
Excepciones	1. No inserto ningún dato 2. No se seleccionó ningún usuario
Prototipo	

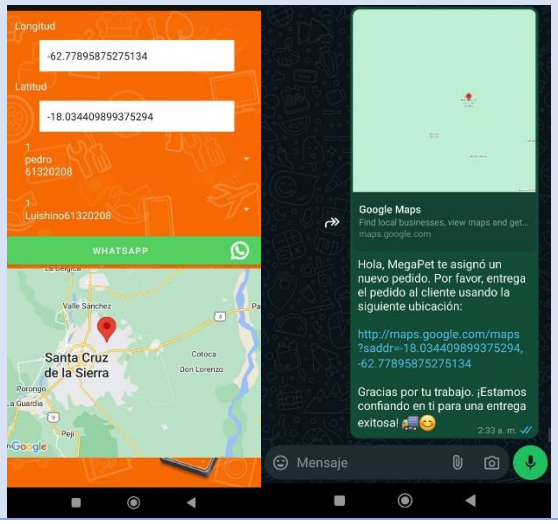
3.2.2 CU2: Gestionar Repartidor



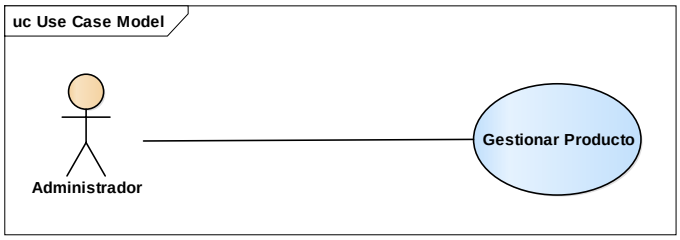
CASO DE USO	CU1: Gestionar Repartidor
Propósito	Administrar repartidor
Descripción	Permite gestionar a los repartidores
Actores	Administrador
Actor Iniciador	Administrador
Precondición	Ninguna
Proceso	1. Agregar Repartidor 1.1. insertar datos 1.2. guardar datos 2. Modificar Repartidor 2.1. Seleccionar Repartidor 2.2. Modificar datos 2.3. Guardar cambios 3. Eliminar Repartidor 3.1. Seleccionar Repartidor 3.2. Modificar datos 3.3. Guardar cambios
Postcondición	Ninguna
Excepciones	3. No inserto ningún dato 4. No se seleccionó ningún usuario
Prototipo	

3.2.3 CU3: Gestionar Entrega



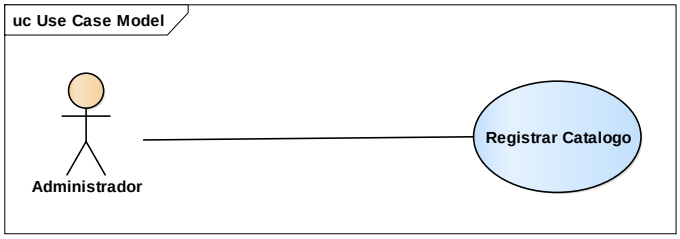
CASO DE USO	CU1: Gestionar Ubicación
Propósito	Administrar ubicación
Descripción	Permite registrar la ubicación global de cada cliente y enviar ubicación al repartidor.
Actores	Administrador
Actor Iniciador	Administrador
Precondición	Ninguna
Proceso	1. Agregar Ubicación 1.1. insertar datos 1.2. guardar datos 2. Modificar Ubicación 2.1. Seleccionar Ubicación 2.2. Modificar datos 2.3. Guardar cambios 3. Eliminar Ubicación 3.1. Seleccionar Ubicación 3.2. Modificar datos 3.3. Guardar cambios
Postcondición	Gestionar Cliente
Excepciones	5. No inserto ningún dato 6. No se seleccionó ninguna ubicación
Prototipo	

3.2.4 CU4: Gestionar Producto



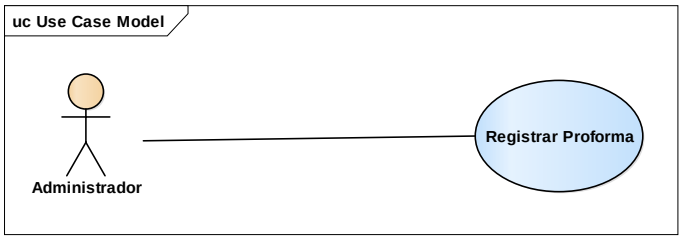
CASO DE USO	CU1: Gestionar Producto
Propósito	Administrar productos
Descripción	Permite registrar los productos pertenecientes a un catalogo
Actores	Administrador
Actor Iniciador	Administrador
Precondición	Ninguna
Proceso	1. Agregar Producto 1.1. insertar datos 1.2. guardar datos 2. Modificar Producto 2.1. Seleccionar Producto 2.2. Modificar datos 2.3. Guardar cambios 3. Eliminar Producto 3.1. Seleccionar Producto 3.2. Modificar datos 3.3. Guardar cambios
Postcondición	Ninguna
Excepciones	7. No inserto ningún dato 8. No se seleccionó ningún producto
Prototipo	

3.2.5 CU5: Registrar Catalogo



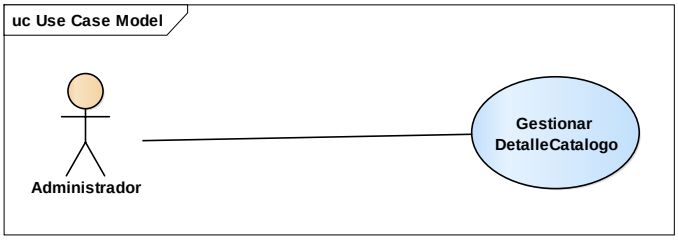
CASO DE USO	CU1: Gestionar Catalogo
Propósito	Administrar catalogo
Descripción	Permite gestionar un catálogo con distintos productos
Actores	Administrador
Actor Iniciador	Administrador
Precondición	Ninguna
Proceso	1. Agregar Catalogo 1.1. Seleccionar productos 1.2. guardar datos 2. Modificar Catalogo 2.1. Seleccionar Producto 2.2. Cambiar Producto 2.3. Guardar cambios 3. Eliminar Catalogo 3.1. Seleccionar Catalogo 3.2. Confirmar eliminación 3.3. Guardar cambios
Postcondición	Gestionar Producto
Excepciones	9. No inserto ningún dato 10. No se seleccionó ningún producto

3.2.6 CU6: Gestionar Proforma



CASO DE USO	CU1: Gestionar Proforma
Propósito	Administrar proforma
Descripción	Permite gestionar la cotización de productos pertenecientes a un catalogo
Actores	Administrador
Actor Iniciador	Administrador
Precondición	Ninguna
Proceso	1. Agregar Proforma 1.1. Seleccionar productos 1.2. guardar datos 2. Modificar Proforma 2.1. Seleccionar Producto 2.2. Cambiar Producto 2.3. Guardar cambios 3. Eliminar Proforma 3.1. Seleccionar Catalogo 3.2. Confirmar eliminación 3.3. Guardar cambios
Postcondición	Gestionar Producto
Excepciones	11. No inserto ningún dato 12. No se seleccionó ningún producto
Prototipo	

3.2.7 CU7: Gestionar DetalleCatalogo

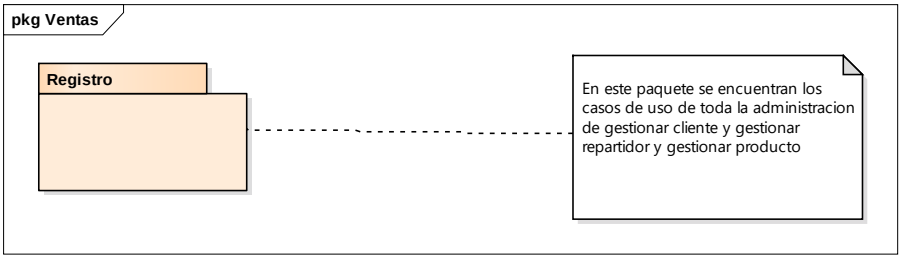


CASO DE USO	CU1: Gestionar Detalle Catalogo
Propósito	Administrar detalle catalogo
Descripción	Permite gestionar los detalles generados a partir de un catalogo
Actores	Administrador
Actor Iniciador	Administrador
Precondición	Ninguna
Proceso	1. Agregar Proforma 1.1. Seleccionar productos 1.2. guardar datos 2. Modificar Proforma 2.1. Seleccionar Producto 2.2. Cambiar Producto 2.3. Guardar cambios 3. Eliminar Proforma 3.1. Seleccionar Catalogo 3.2. Confirmar eliminación 3.3. Guardar cambios 4. PDF 4.1. Crear Catalogo PDF
Postcondición	Gestionar Producto
Excepciones	13. No inserto ningún dato 14. No se seleccionó ningún producto
Prototipo	

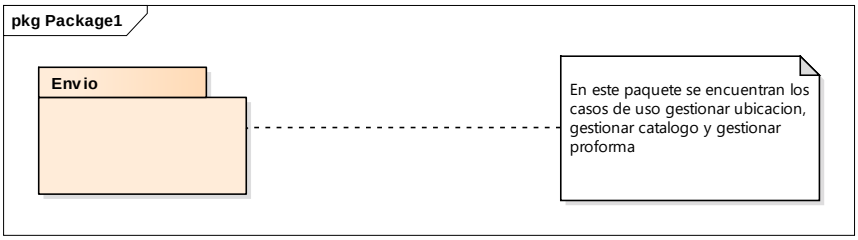
4 Análisis

4.1 Identificación de módulos

4.1.1 Paquete 1: Registro

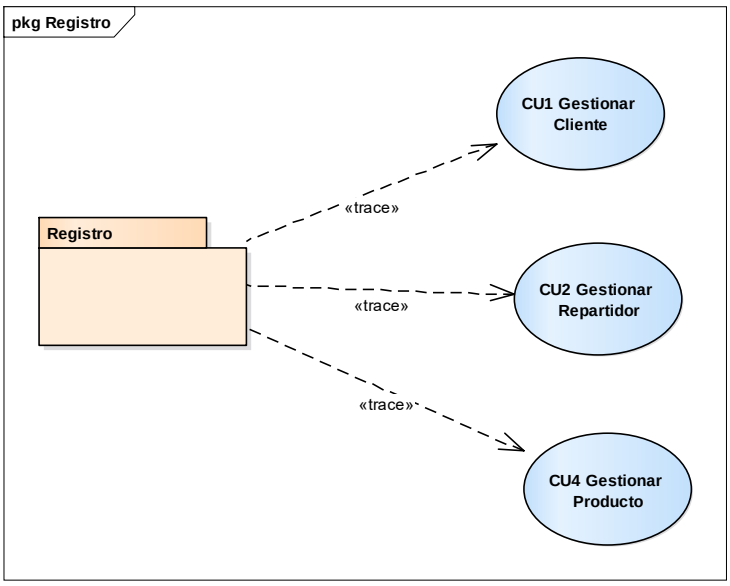


4.1.2 Paquete 2: Envió

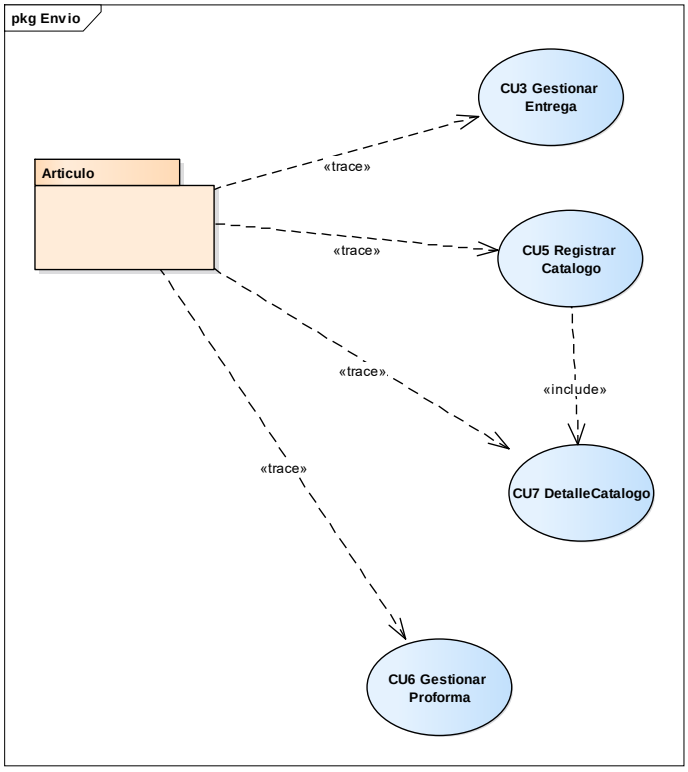


4.2 Vista de módulos

4.2.1 Paquete 1: Registro

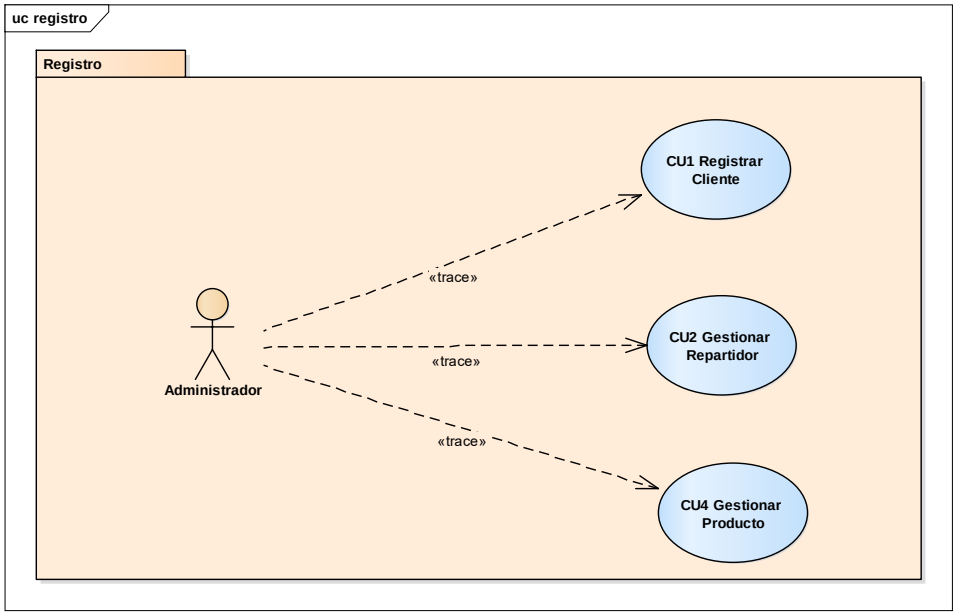


4.2.2 Paquete 2: Articulo

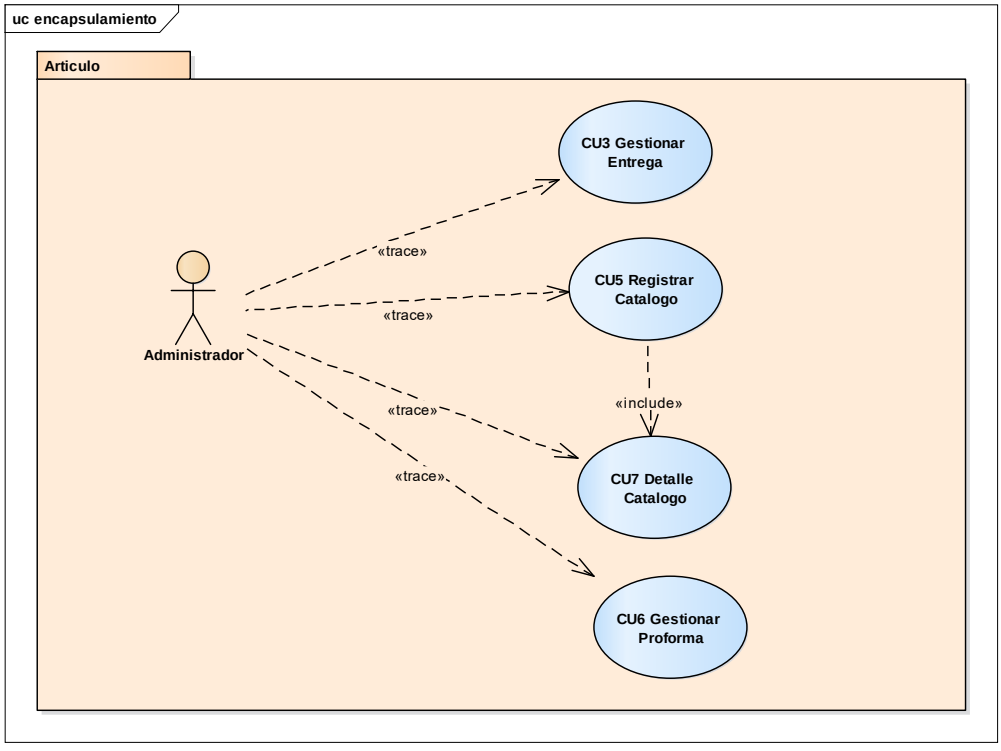


4.3 Encapsulamiento

4.4 Registro



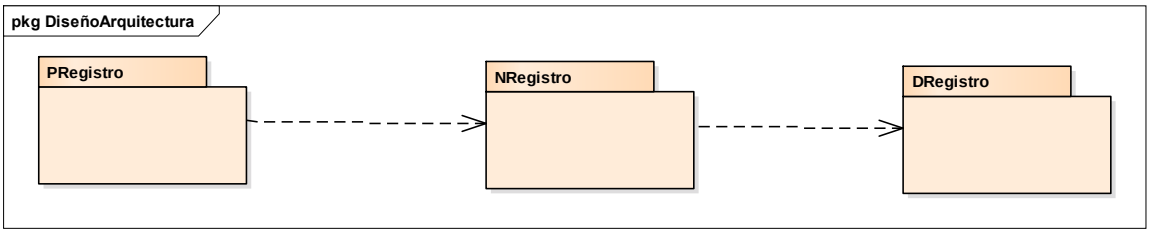
4.5 Articulo



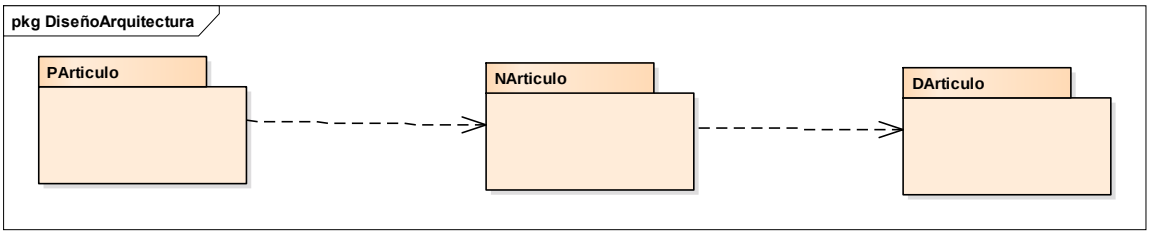
5 Diseño

5.1 Diseño de arquitectura

5.1.1 Paquete 1: Registro

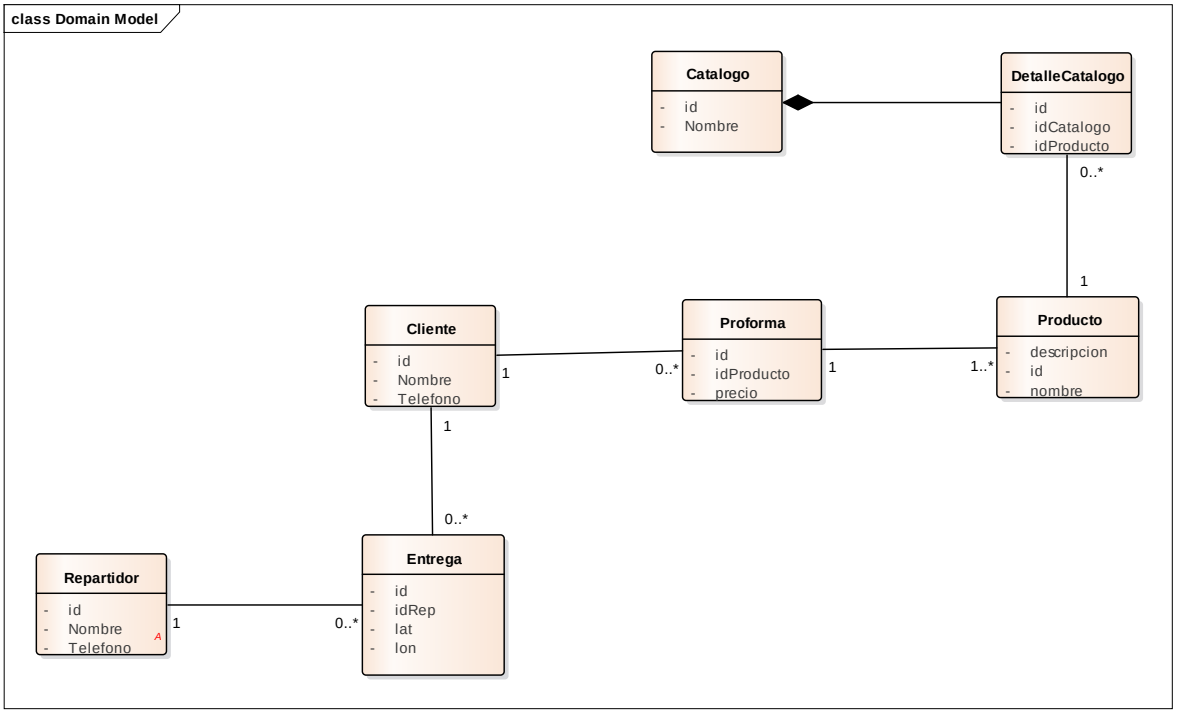


5.1.2 Paquete 2: Articulo



5.2 Diseño de la base de datos

5.2.1 Diseño conceptual



5.2.2 Diseño lógico

Cliente

PK

ID	NOMBRE	TELEFONO

Repartidor

PK

ID	NOMBRE	TELEFONO

Entrega

PK **FK**

ID	LON	LAT	ID_REPARTIDOR

Catalogo

PK

ID	NOMBRE_CATALOGO

Producto

PK

ID	NOMBRE	Descripcion

Proforma

PK **FK**

ID	PRECIO	Id_Producto

DetalleCatalogo

PK **PK**

FK

idCatalogo	id	idProducto

5.2.3 Diseño físico

5.2.3.1 Script base de datos SQLite

```
-- Habilitar claves foráneas
PRAGMA foreign_keys = ON;

-- Creación de la tabla Cliente
CREATE TABLE IF NOT EXISTS Cliente (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   nombre TEXT NOT NULL,
   telefono TEXT NOT NULL
);

-- Creación de la tabla Repartidor
CREATE TABLE IF NOT EXISTS Repartidor (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   nombre TEXT NOT NULL,
   telefono TEXT NOT NULL
);

-- Creación de la tabla Producto
CREATE TABLE IF NOT EXISTS Producto (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   nombre TEXT NOT NULL,
   imagen BLOB,
   descripcion TEXT
);

-- Creación de la tabla Catálogo
CREATE TABLE IF NOT EXISTS Catalogo (
```

```
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nombre TEXT NOT NULL
);

-- Creación de la tabla Detalle Catálogo
CREATE TABLE IF NOT EXISTS DetalleCatalogo (
    idcatalogo INTEGER NOT NULL,
    idproducto INTEGER NOT NULL,
    FOREIGN KEY (idcatalogo) REFERENCES Catalogo(id),
    FOREIGN KEY (idproducto) REFERENCES Producto(id)
);

-- Creación de la tabla Proforma
CREATE TABLE IF NOT EXISTS Proforma (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    idproducto INTEGER NOT NULL,
    idcliente INTEGER NOT NULL,
    precio REAL NOT NULL,
    FOREIGN KEY (idproducto) REFERENCES Producto(id),
    FOREIGN KEY (idcliente) REFERENCES Cliente(id)
);

-- Creación de la tabla Entrega
CREATE TABLE IF NOT EXISTS Entrega (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    idrep INTEGER NOT NULL,
    idcliente INTEGER NOT NULL,
    latit TEXT,
    longit TEXT,
    FOREIGN KEY (idrep) REFERENCES Repartidor(id),
```

```
    FOREIGN KEY (idcliente) REFERENCES Cliente(id)
);
```

6 Diseño de la interfaz

6.1 Menu principal

6.2 CU1: Gestionar Cliente


Nombre :
Ingresar Nombre
Telefono :
Ingresar Telefono
AGREGAR
EDITAR
ELIMINAR
Item 1
Sub Item 1
Item 2
Sub Item 2
Item 3
Sub Item 3
Item 4
Sub Item 4
Item 5
Sub Item 5

6.3 CU2: Gestionar Repartidor

6.4 CU3: gestionar Entrega

6.5 CU4: Gestionar Producto

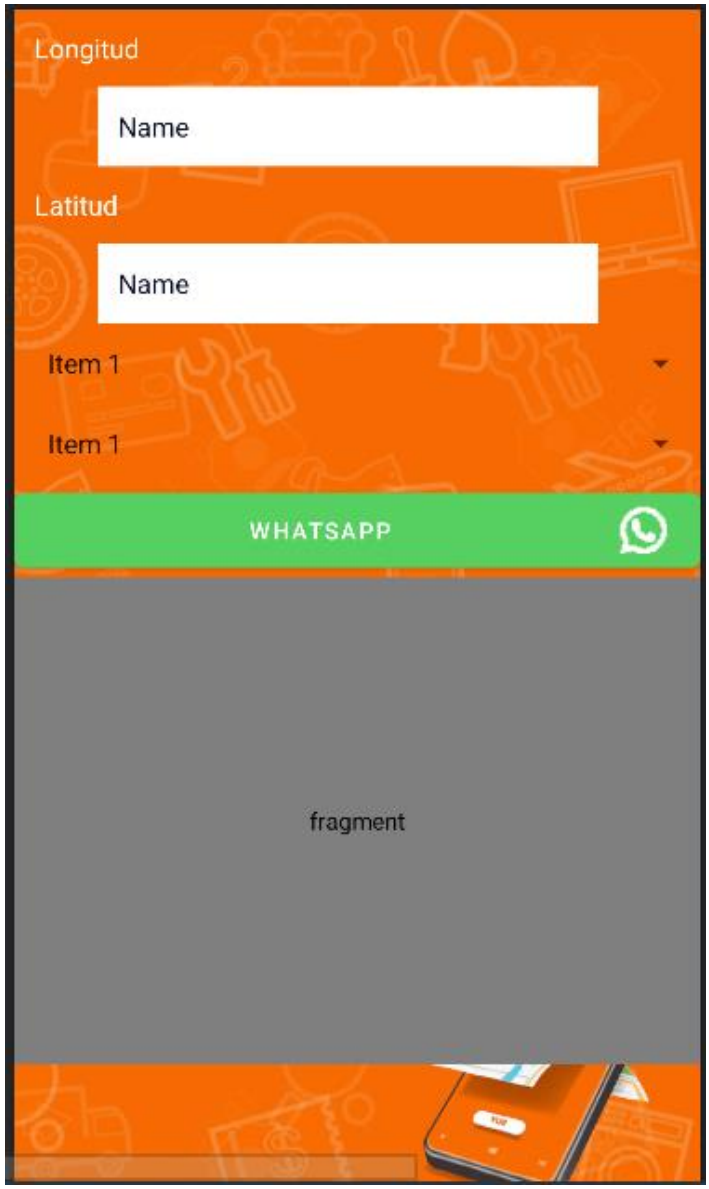
6.6 CU5: Registrar Catalogo

6.7 CU6: Gestionar Proforma

6.8 CU7: Gestionar DetalleCatalogo


idCatalogo
Ingresar ID
Productos
Item 1
AGREGAR
ELIMINAR
PDF
WHATSSAP
Item 1
Sub Item 1
Item 2
Sub Item 2
Item 3
Sub Item 3
Item 4
Sub Item 4
Item 5
Sub Item 5
Item 6
Sub Item 6
Item 7
Sub Item 7
Item 8

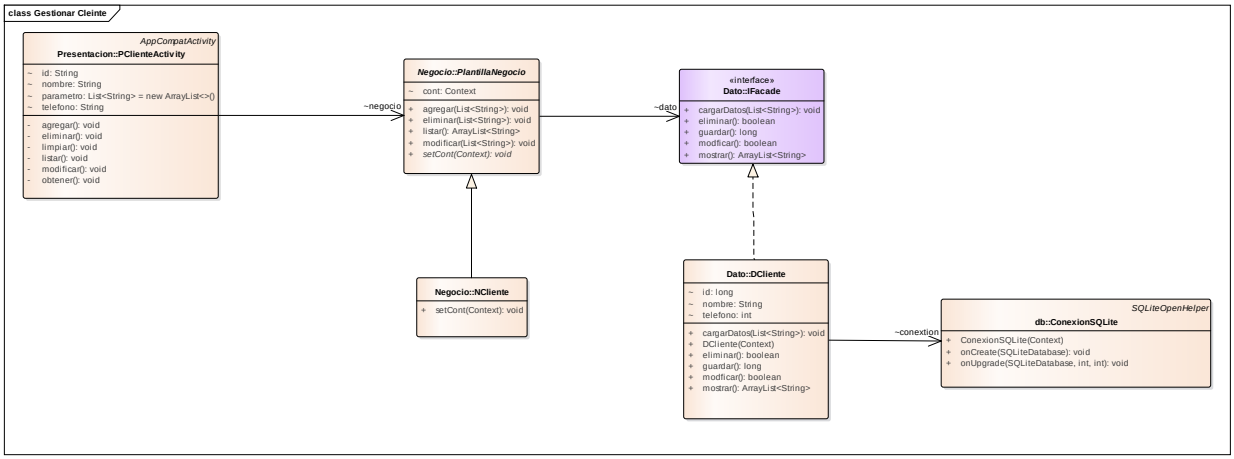
6.9 Gestionar Entrega - Ubicación



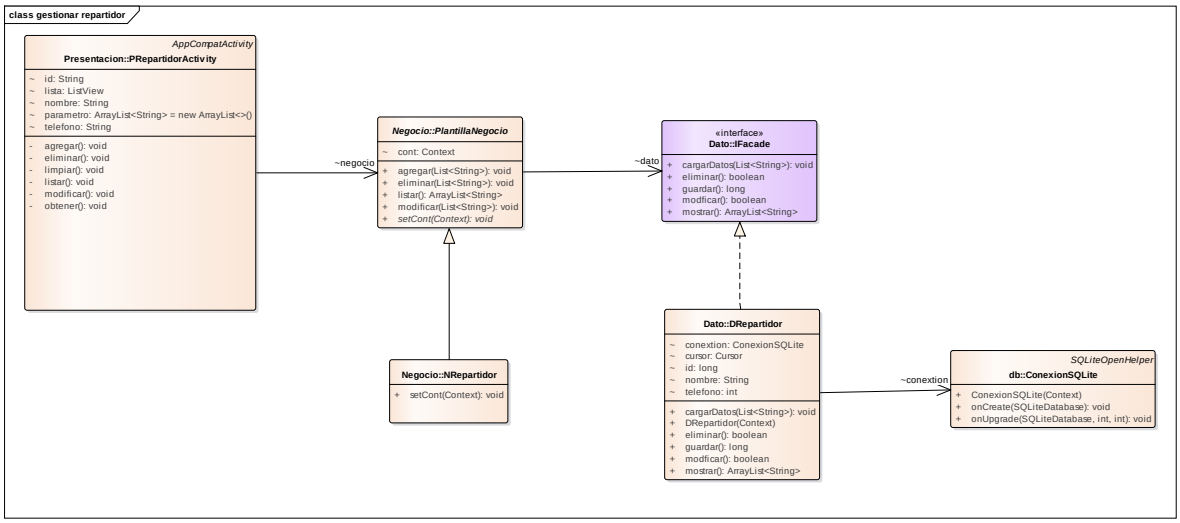
7 Diseño procedimental

7.1 Diseño de clases dinámico

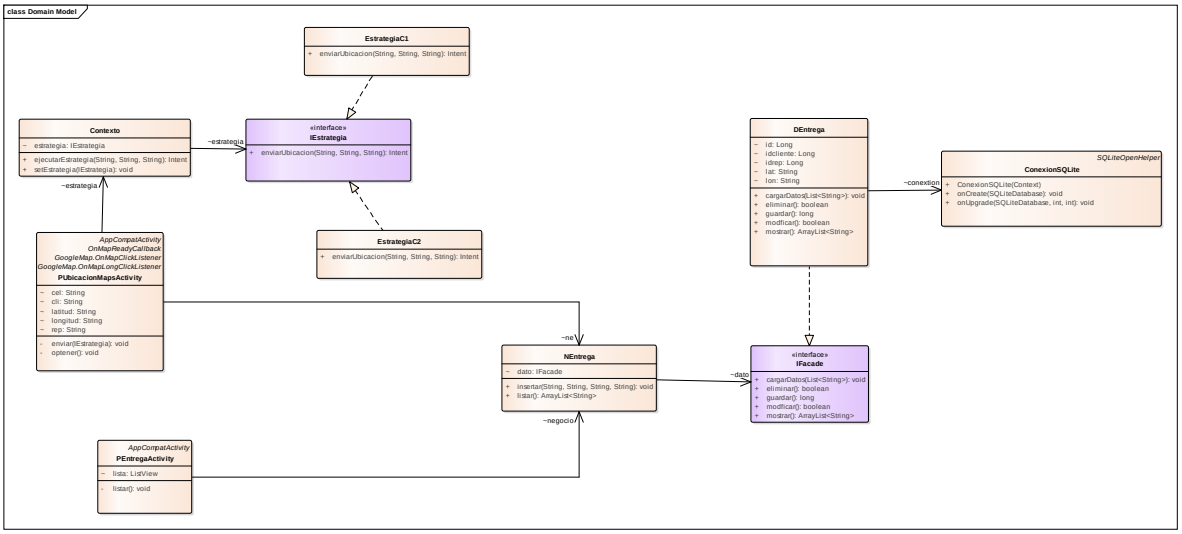
7.1.1 CU1: Gestionar Cliente



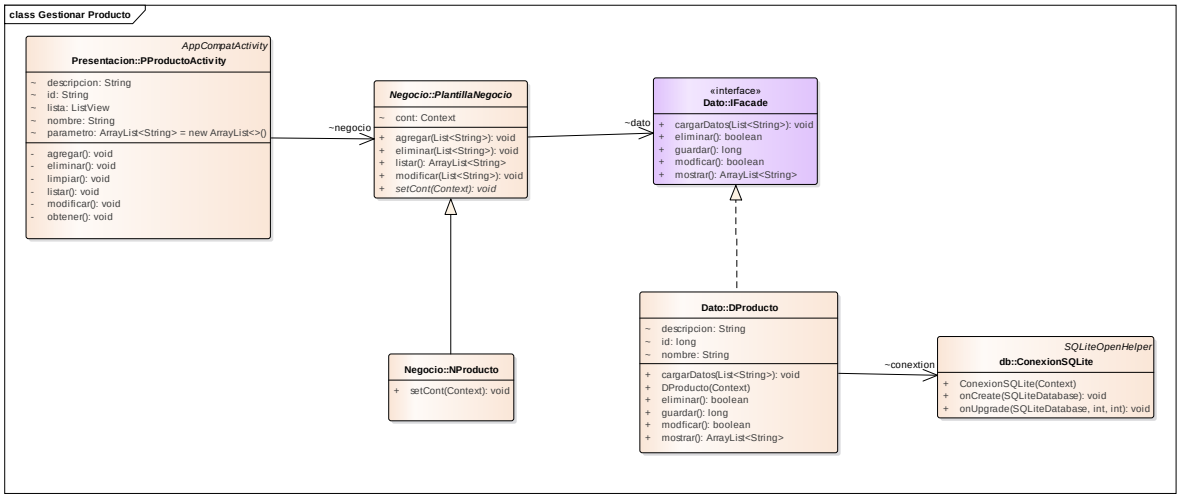
7.1.2 CU2: Gestionar Repartidor



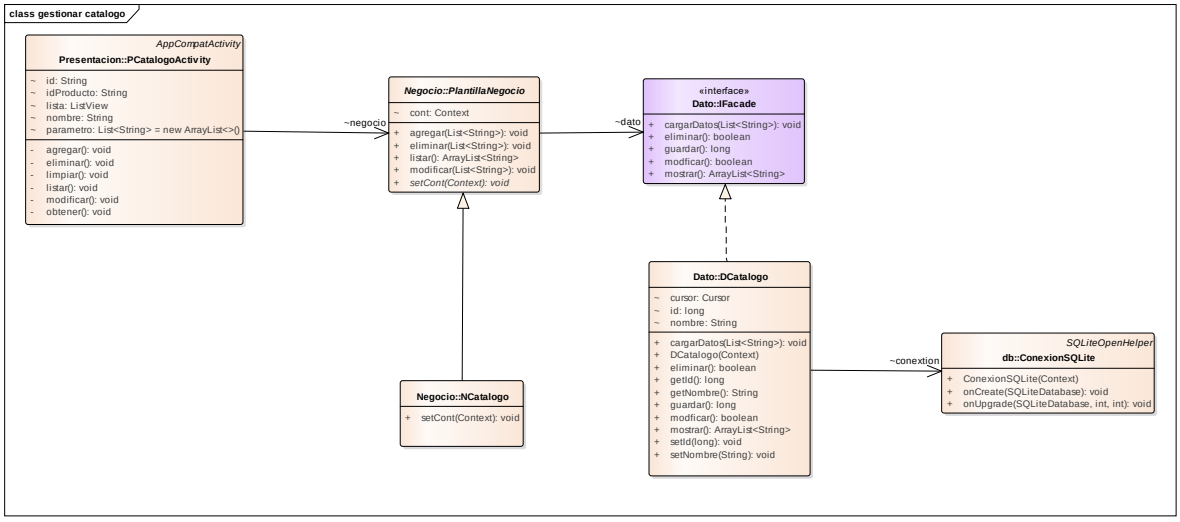
7.1.3 CU3: gestionar Entrega (Ubicación)



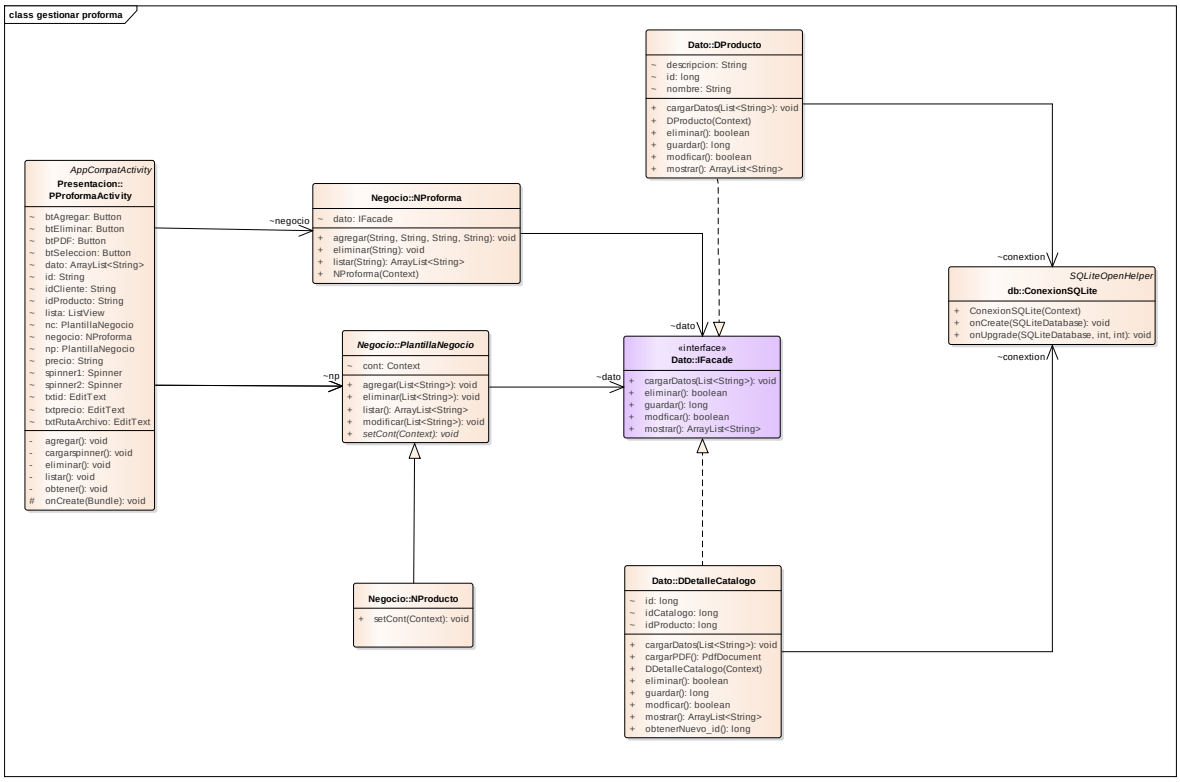
7.1.4 CU4: Gestionar Producto



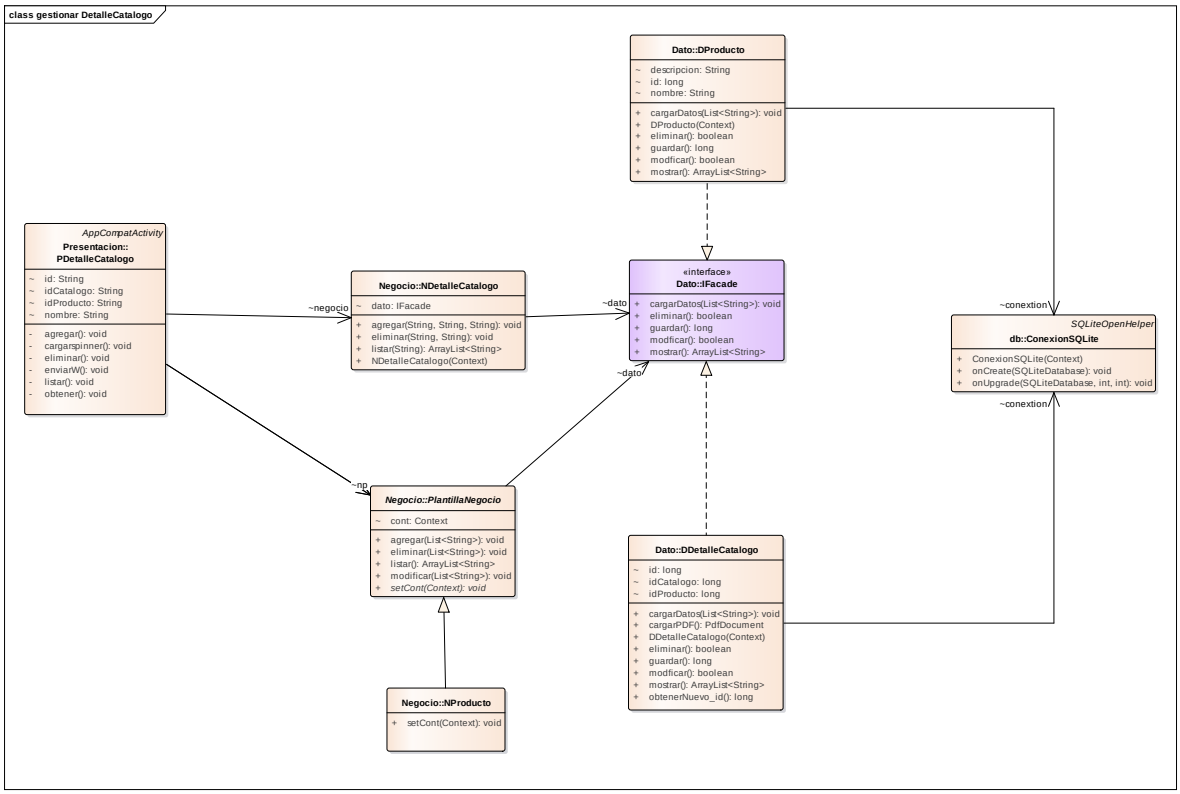
7.1.5 CU5: Registrar Catalogo



7.1.6 CU6: Gestionar Proforma

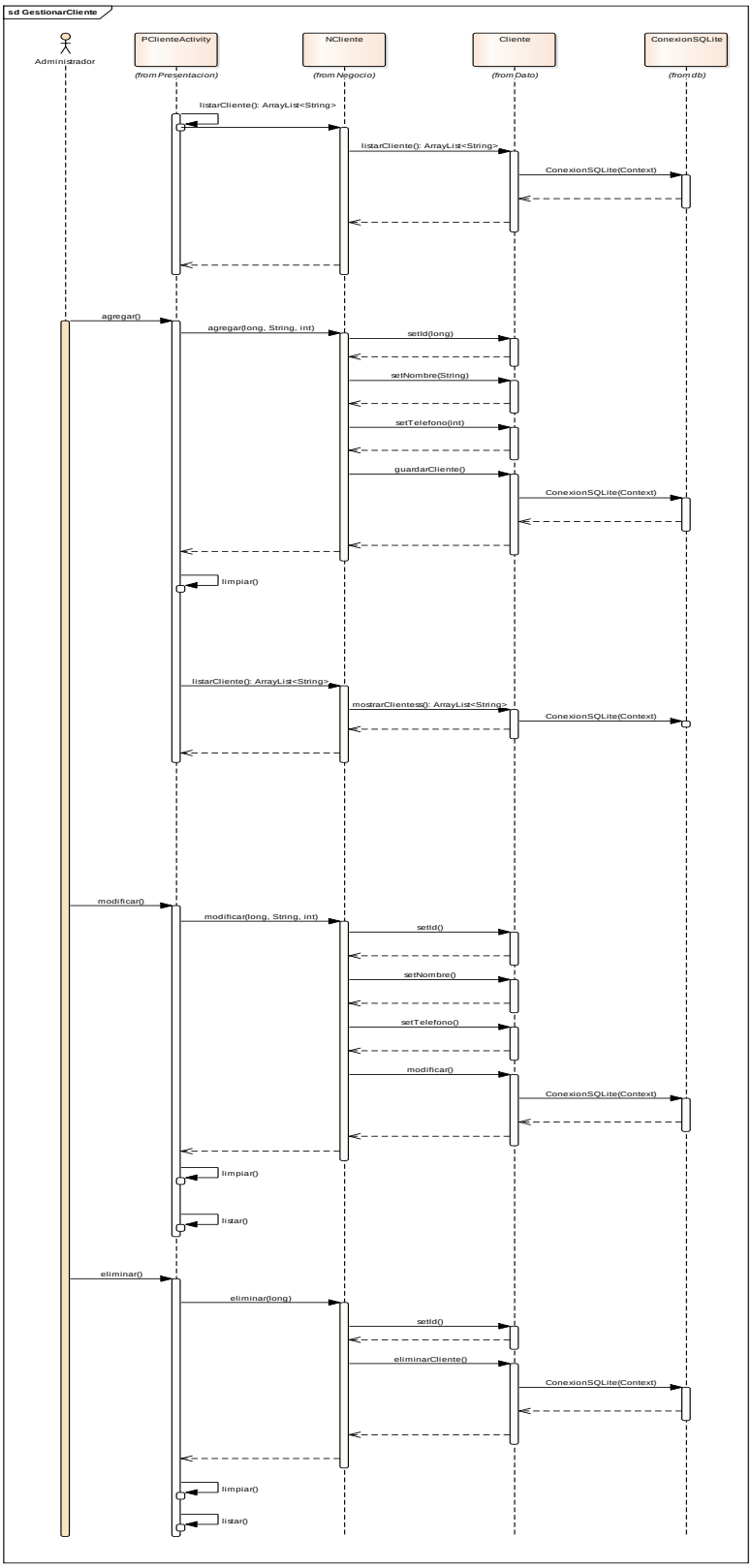


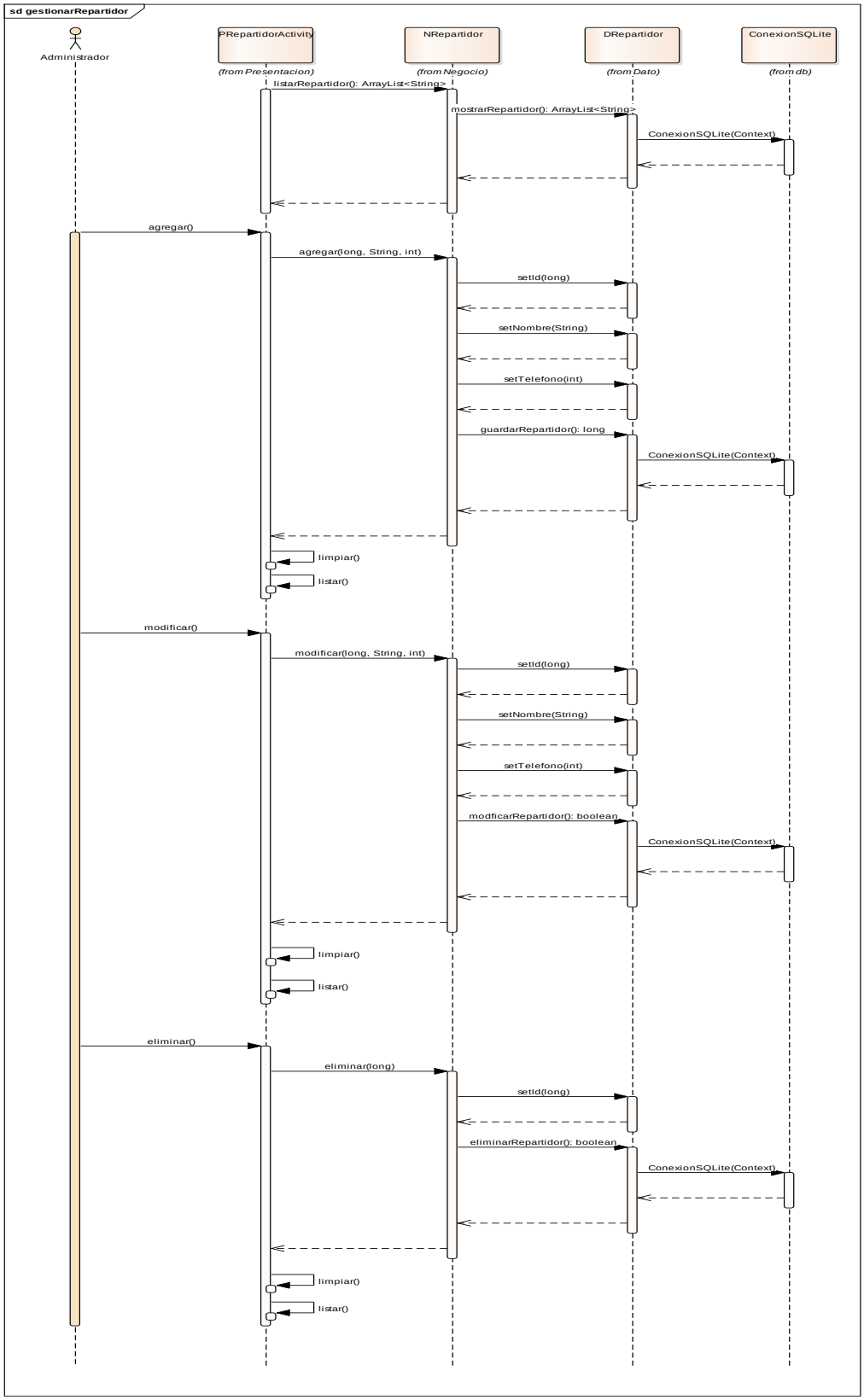
7.1.7 CU7: Gestionar DetalleCatalogo



7.2 Diagrama de secuencia

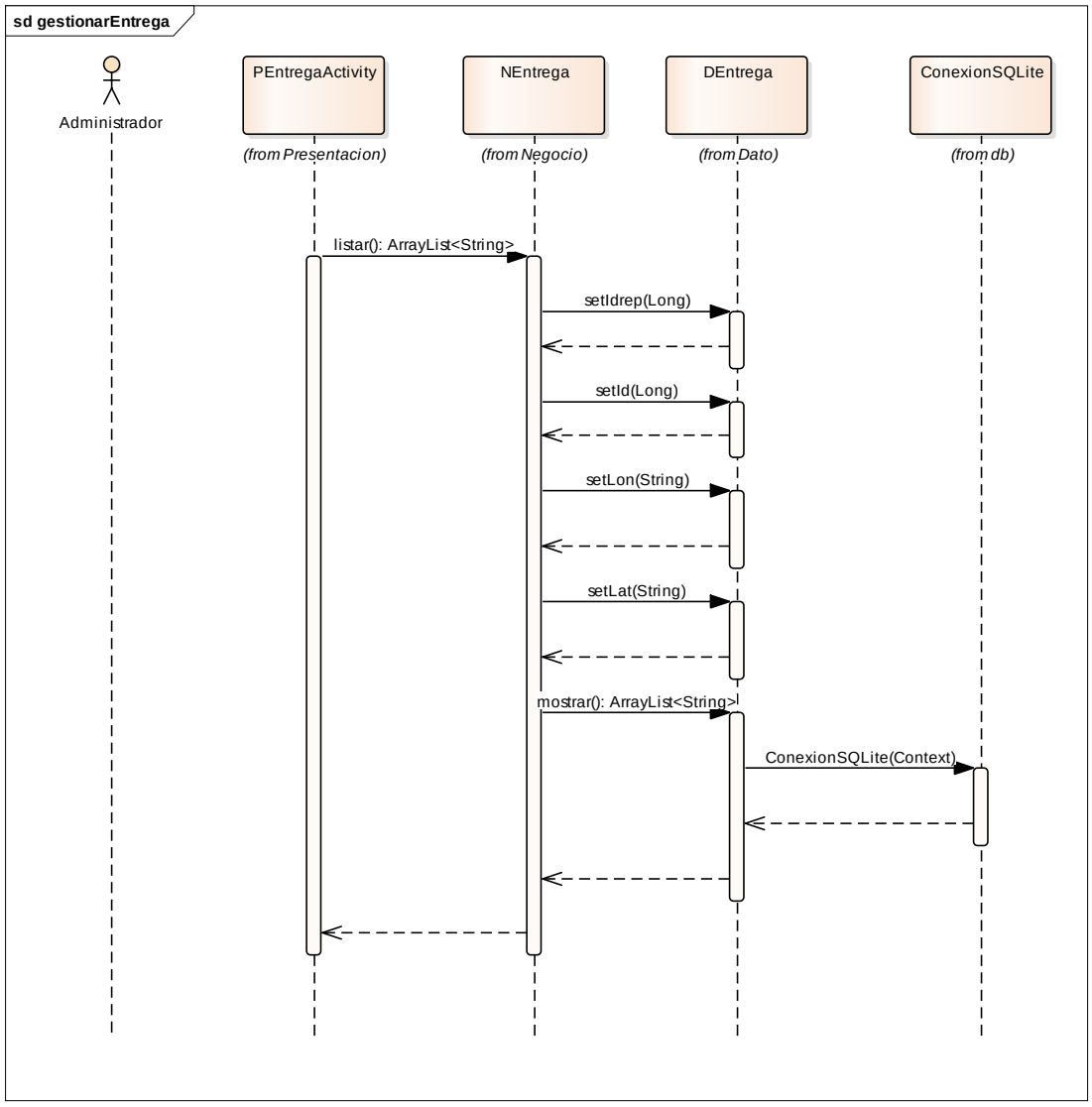
7.2.1 CU1: Gestionar Cliente



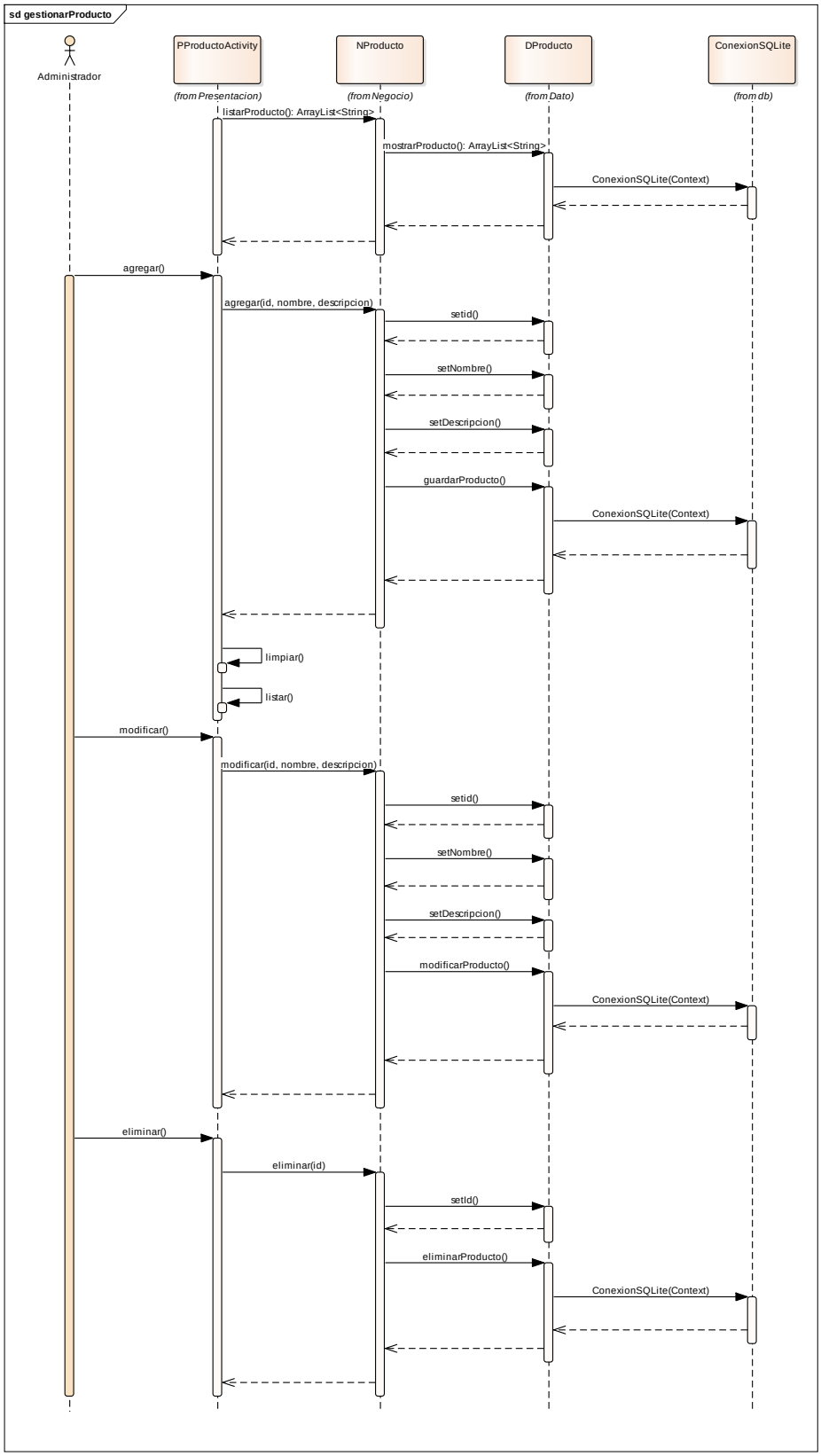


7.2.2 CU2: Gestionar Repartidor

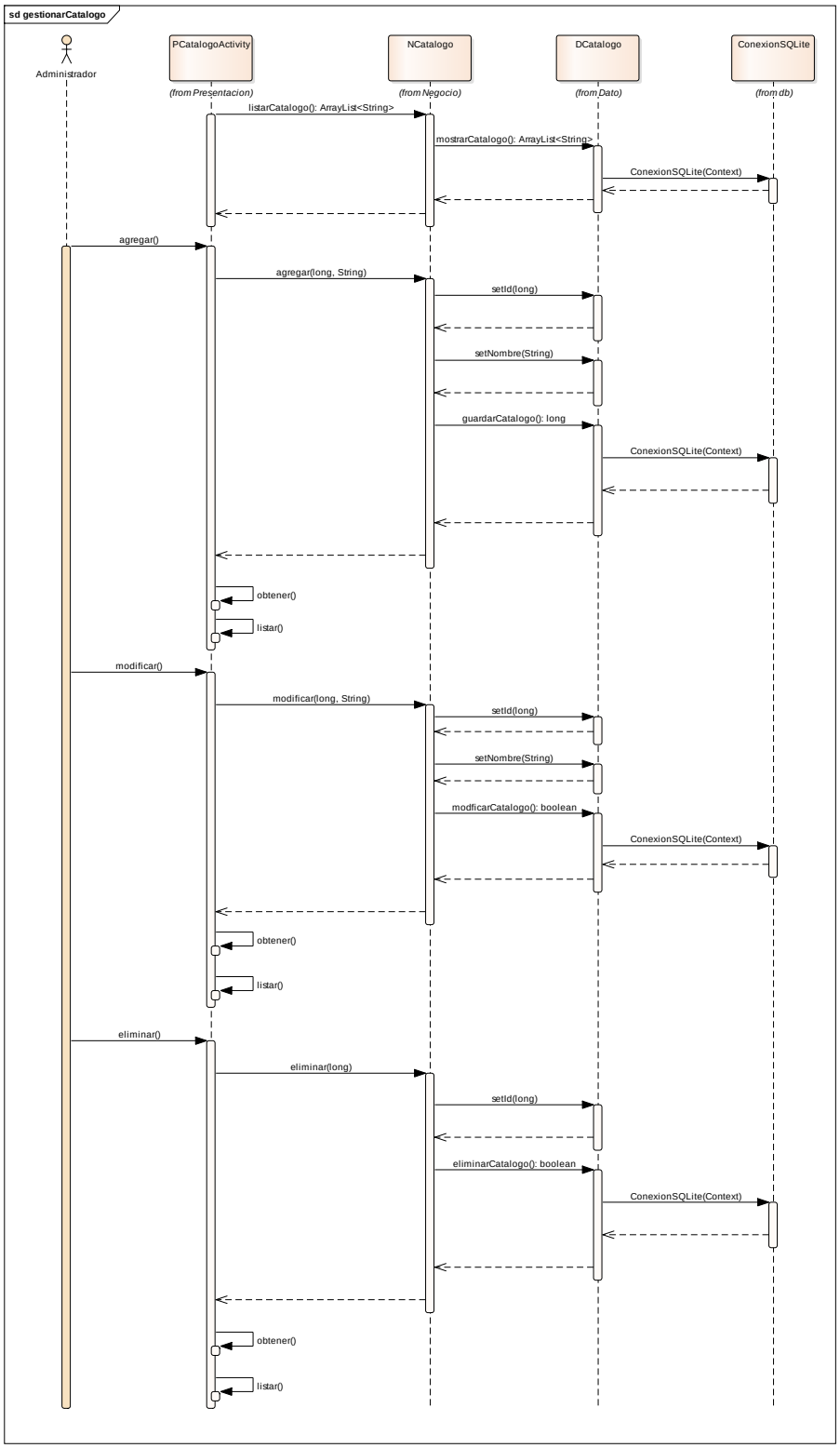
7.2.3 CU3: gestionar Entrega (ubicación)



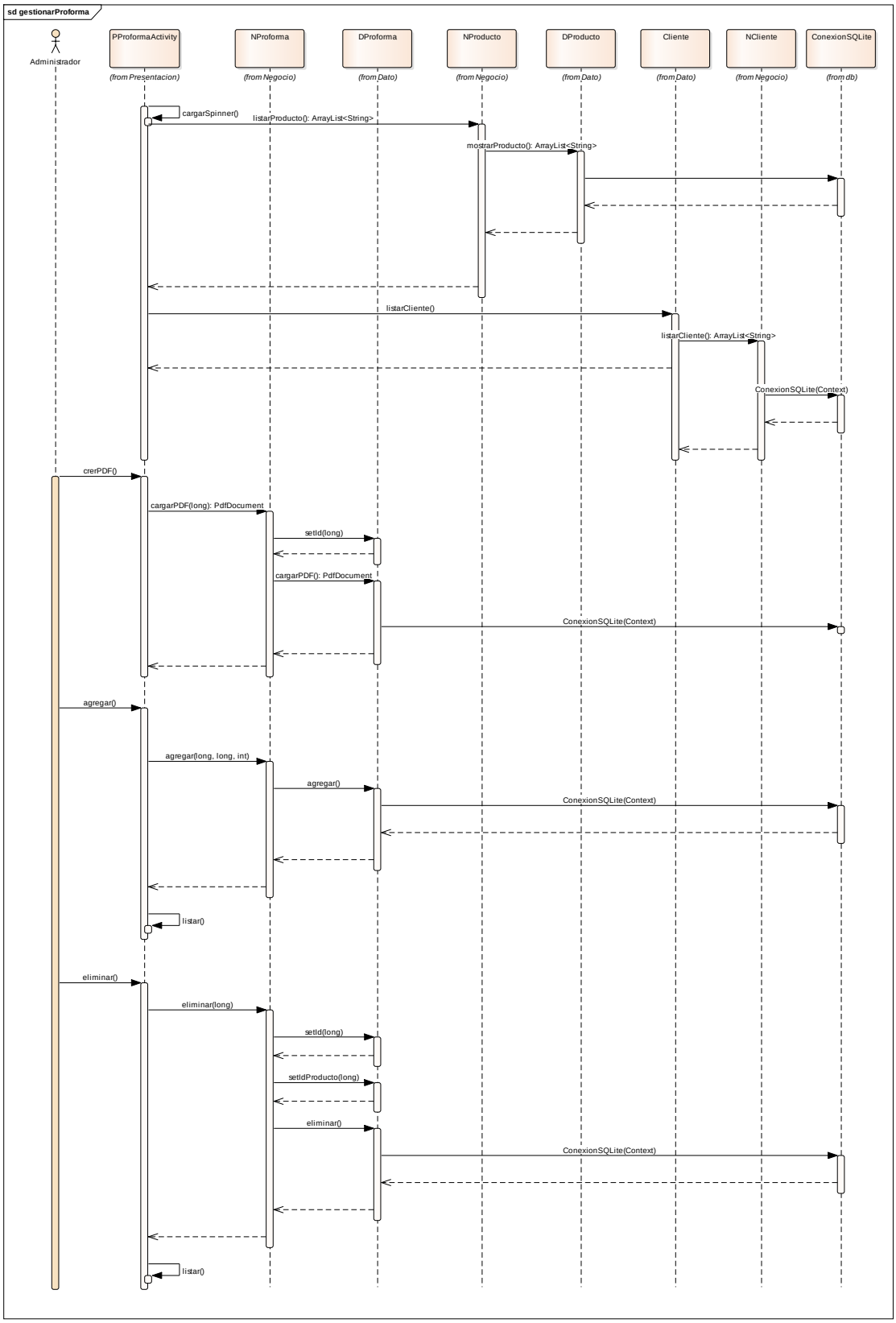
7.2.4 CU4: Gestionar Producto



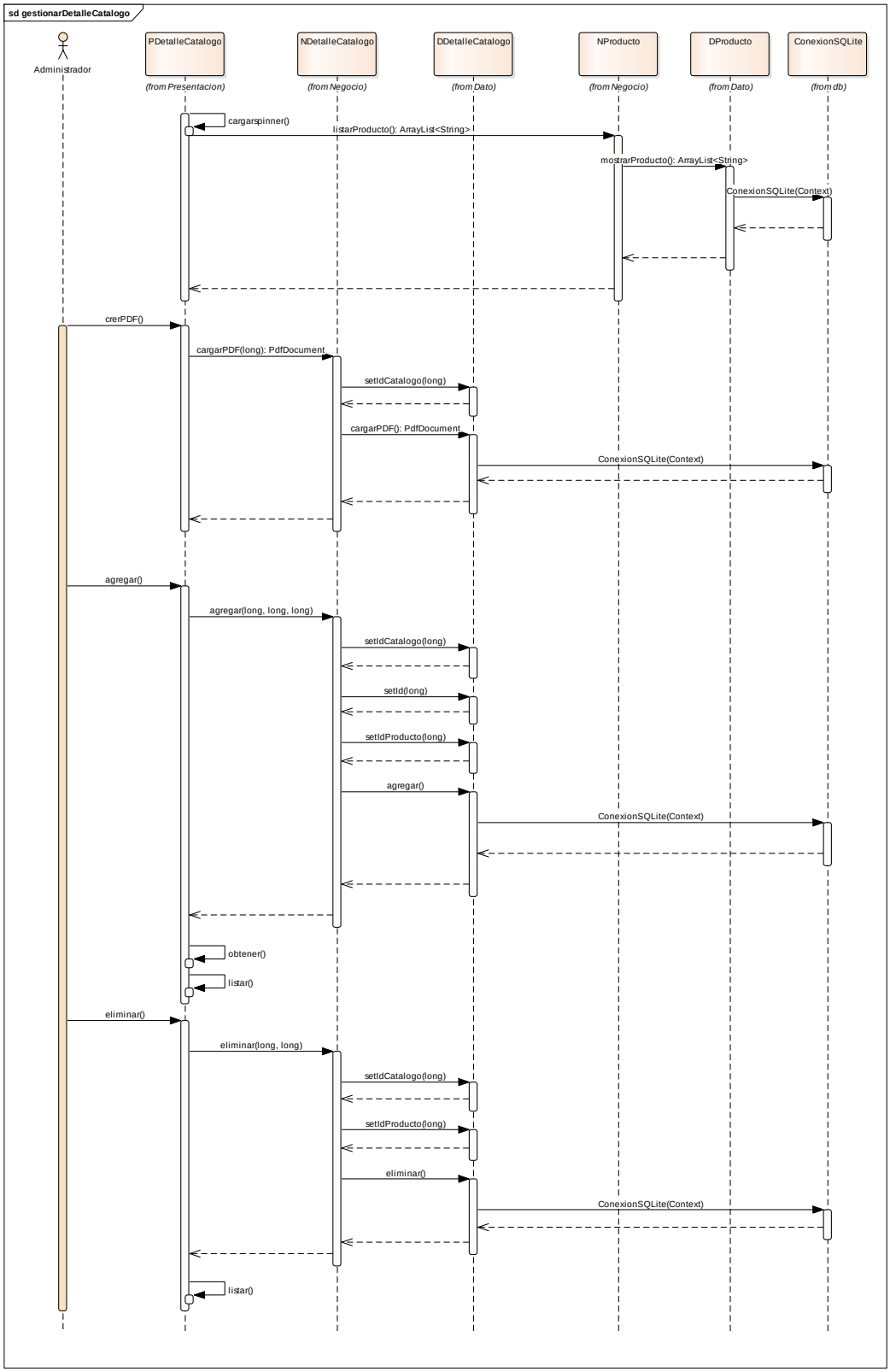
7.2.5 CU5: Registrar Catalogo



7.2.6 CU6: Gestionar Proforma



7.2.7 CU7: Gestionar Detalle Catalogo



8 Codificación

8.1 db

8.1.1 ConexionSQLite

```
public class ConexionSQLite extends SQLiteOpenHelper {
    ConexionSQLite conexion;
    private static final int data_base_v =1;
    private static final String database_Nombre ="emprende.db";
    public static final String table_cliente ="cliente";
    public static final String table_producto ="producto";
    public static final String table_repartidor ="repartidor";
    public static final String table_catalogo ="catalogo";
    public static final String table_detalle_catalogo ="detallecatalogo";
    public static final String table_proforma ="proforma";
    public static final String table_Entrega ="entrega";

    // Constructor de la clase que recibe el contexto
    public ConexionSQLite(@Nullable Context context) {
        super(context, database_Nombre, null, data_base_v);
    }

    // Método para crear las tablas en la base de datos
    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        // Habilitar claves foráneas
        sqLiteDatabase.execSQL("PRAGMA foreign_keys = ON;");

        // Creación de la tabla Cliente
        sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS " + table_cliente + "
(" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "nombre TEXT NOT NULL, " +
                "telefono TEXT NOT NULL)");

        // Creación de la tabla Repartidor
        sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS " + table_repartidor
+ " (" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "nombre TEXT NOT NULL, " +
                "telefono TEXT NOT NULL)");

        // Creación de la tabla Producto
        sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS " + table_producto +
" (" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "nombre TEXT NOT NULL, " +
                "imagen BLOB, " +
                "descripcion TEXT)");

        // Creación de la tabla Catálogo
```

```
        sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS " + table_catalogo +
" (" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "nombre TEXT NOT NULL)");

        // Creación de la tabla Detalle Catálogo
        sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS " +
table_detalle_catalogo + " (" +
                "idcatalogo INTEGER NOT NULL, " +
                "idproducto INTEGER NOT NULL, " +
                "FOREIGN KEY (idcatalogo) REFERENCES " + table_catalogo + "(id),
" +
                "FOREIGN KEY (idproducto) REFERENCES " + table_producto +
"(id))");

        // Creación de la tabla Proforma
        sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS " + table_proforma +
" (" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "idproducto INTEGER NOT NULL, " +
                "idcliente INTEGER NOT NULL, " +
                "precio REAL NOT NULL, " +
                "FOREIGN KEY (idproducto) REFERENCES " + table_producto + "(id),
" +
                "FOREIGN KEY (idcliente) REFERENCES " + table_cliente +
"(id))");

        // Creación de la tabla Entrega
        sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS " + table_Entrega + "
(" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "idrep INTEGER NOT NULL, " +
                "idcliente INTEGER NOT NULL, " +
                "latit TEXT, " +
                "longit TEXT, " +
                "FOREIGN KEY (idrep) REFERENCES " + table_repartidor + "(id), "
+
                "FOREIGN KEY (idcliente) REFERENCES " + table_cliente +
"(id))");
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS "+table_repartidor);
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS "+table_cliente);
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS "+table_producto);
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS "+table_catalogo);
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS "+table_detalle_catalogo);
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS "+table_proforma);
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS "+table_Entrega);
```

```
        onCreate(sqLiteDatabase);
    }
}
```

8.2 Dato

8.2.1 Dcliente

```
public class DCliente{
    ConexionSQLite conextion;
    Cursor cursor;
    long id;
    String nombre;
    int telefono;

    public DCliente(@Nullable Context context) {
        conextion= new ConexionSQLite(context);
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
```

```
}

public void setNombre(String nombre) {
   this.nombre = nombre;
}

public int getTelefono() {
   return telefono;
}

public void setTelefono(int telefono) {
   this.telefono = telefono;
}

public long guardarCliente() {
   long id=0;
   try{
      SQLiteDatabase db = conextion.getWritableDatabase();
      ContentValues registro = new ContentValues();
      registro.put("nombre",nombre);
      registro.put("telefono",telefono);
      id = db.insert(conextion.table_cliente,null,registro);
   }catch (Exception ex){
      ex.toString();
   }
   return id;
}

public ArrayList<String> mostrarClientess() {
```

```
        SQLiteDatabase db = conextion.getWritableDatabase();

        ArrayList<String> listaCliente = new ArrayList<>();
        String cliente = "";
        cursor = null;

        cursor = db.rawQuery("select * from cliente", null);
        if (cursor.moveToFirst()) {
            do {
                cliente=  String.valueOf(cursor.getInt(0)) +"\n";
                cliente+=  (cursor.getString(1));
                cliente+= String.valueOf(cursor.getInt(2));
                listaCliente.add(cliente);
            } while (cursor.moveToNext());
        }
        cursor.close();
        return listaCliente;
    }

    public boolean modficarCliente() {
        boolean correcto = false;

        SQLiteDatabase db = conextion.getWritableDatabase();
        try {
            db.execSQL("UPDATE "+ conextion.table_cliente + " SET nombre= '"+nombre+"', telefono=
'"+telefono+"' WHERE id= '"+id+"'");
            correcto = true;
        }catch (Exception ex){
            ex.toString();
```

```
            correcto = false;
        } finally {
            db.close();
        }
        return correcto;
    }

    public boolean eliminarCliente() {
        boolean correcto = false;
        SQLiteDatabase db = conextion.getWritableDatabase();
        try {
            int filasAfectadas = db.delete(conextion.table_cliente, "id = ?", new
String[]{String.valueOf(id)});

            if (filasAfectadas > 0) {
                correcto = true;
            } else {
                System.out.println("No se encontró ningún cliente con el id proporcionado.");
            }
        } catch (Exception ex) {
            System.out.println("Error al eliminar el cliente: " + ex.getMessage());
            correcto = false;
        } finally {
            db.close();
        }
        return correcto;
    }
}
```

8.2.2 DRepartidor

```
public class DRepartidor {
    ConexionSQLite conextion;
    Cursor cursor;

   long id;
    String nombre;
    int telefono;

   public DRepartidor(@Nullable Context context) {

        conextion= new ConexionSQLite(context);
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }
```

```
public void setNombre(String nombre) {
    this.nombre = nombre;
}

public int getTelefono() {
    return telefono;
}

public void setTelefono(int telefono) {
    this.telefono = telefono;
}

public long guardarRepartidor() {
    long id=0;
    try{
        SQLiteDatabase db = conextion.getWritableDatabase();
        ContentValues registro = new ContentValues();
        registro.put("nombre",nombre);
        registro.put("telefono",telefono);
        id = db.insert(conextion.table_repartidor,null,registro);

    }catch (Exception ex){
        ex.toString();

    }
    return id;
```

```
}
public ArrayList<String> mostrarRepartidor() {
    SQLiteDatabase db = conextion.getWritableDatabase();

    ArrayList<String> listaRepartidor = new ArrayList<>();
    String repartidor = "";
    cursor = null;

    cursor = db.rawQuery("select * from repartidor", null);
    if (cursor.moveToFirst()) {
        do {
            repartidor=  String.valueOf(cursor.getInt(0)) +"\n";
            repartidor+=  (cursor.getString(1))+"\n";
            repartidor+= String.valueOf(cursor.getInt(2));
            listaRepartidor.add(repartidor);
        } while (cursor.moveToNext());
    }
    cursor.close();

    return listaRepartidor;
}


public boolean modficarRepartidor() {
    boolean correcto = false;
```

```
        SQLiteDatabase db = conextion.getWritableDatabase();
        try {
            db.execSQL("UPDATE "+ conextion.table_repartidor + " SET
nombre= '"+nombre+"', telefono= '"+telefono+"' WHERE id= '"+id+"'");
            correcto = true;
        }catch (Exception ex){
            ex.toString();
            correcto = false;
        } finally {
            db.close();
        }

        return correcto;
    }

    public boolean eliminarRepartidor() {
        boolean correcto = false;

        SQLiteDatabase db = conextion.getWritableDatabase();
        try {
            //db.execSQL("DELETE FROM"  + conextion.table_repartidor + "'
WHERE id= '"+id+"'");
            db.execSQL("DELETE FROM " + conextion.table_repartidor + "
WHERE id = '" + id + "'");
            correcto = true;
        }catch (Exception ex){
            ex.toString();
```

```
            correcto = false;
        } finally {
            db.close();
        }
        return correcto;
    }
```

8.2.3 DEntrega

```
public class DEntrega {
    String lat,lon;
    Long id,idrep,idcliente;

    ConexionSQLite conextion;
    Cursor cursor;
    public DEntrega(@Nullable Context context) {

        conextion= new ConexionSQLite(context);
    }

    public void setLat(String lat) {
        this.lat = lat;
    }

    public void setLon(String lon) {
        this.lon = lon;
    }

    public void setId(Long id) {
        this.id = id;
```

```
}

public void setIdrep(Long idrep) {
   this.idrep = idrep;
}

public void setIdcliente(Long idcliente) {
   this.idcliente = idcliente;
}

public void agregar() {
   try{
      SQLiteDatabase db = conextion.getWritableDatabase();
      ContentValues registro = new ContentValues();
      registro.put("idrep",idrep);
      registro.put("idcliente",idcliente);
      registro.put("latit",lat);
      registro.put("longit",lon);
      id = db.insert(conextion.table_Entrega,null,registro);
   }catch (Exception ex){
      ex.toString();
   }
}

public ArrayList<String> mostrar() {
   SQLiteDatabase db = conextion.getWritableDatabase();

   ArrayList<String> listaEntrega = new ArrayList<>();
   String entrega = "Nro\t Repartidor \t latitud \t longitud";
```

```
        cursor = null;
        listaEntrega.add(entrega);

//      cursor = db.rawQuery("SELECT * FROM entrega ", null);
        cursor = db.rawQuery("SELECT e.id, r.nombre,c.nombre, e.latit, e.longit FROM entrega as e,
repartidor as r, cliente as c WHERE r.id=e.idrep and e.idcliente=c.id", null);
        if (cursor.moveToFirst()) {
            do {
                entrega=  String.valueOf(cursor.getInt(0)) +"\t\t\t";
                entrega+= cursor.getString(1)+"\t\t\t";
                entrega+= cursor.getString(2)+"\t\t\t\t\t\t\t\t";
                entrega+= cursor.getString(3)+"\t\t";
                entrega+= cursor.getString(4);
                listaEntrega.add(entrega);
            } while (cursor.moveToNext());
        }
        cursor.close();

        return listaEntrega;
    }
}
```

8.2.4 DProducto

```
public class DProducto {
    ConexionSQLite conextion;
    Cursor cursor;
    long id;
    String nombre;
    String descripcion;
    public DProducto(@Nullable Context context) {
```

```
    conextion= new ConexionSQLite(context);
}
public long getId() {
    return id;
}
public void setId(long id) {
    this.id = id;
}
public String getNombre() {
    return nombre;
}
public void setNombre(String nombre) {
    this.nombre = nombre;
}
public String getDescripcion() {
    return descripcion;
}
public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}
public long guardarProducto() {
    long id=0;
    try{
        SQLiteDatabase db = conextion.getWritableDatabase();
        ContentValues registro = new ContentValues();
        registro.put("nombre",nombre);
        registro.put("descripcion",descripcion);
        id = db.insert(conextion.table_producto,null,registro);

    }catch (Exception ex){
        ex.toString();
    }
    return id;
```

```
    }
    public ArrayList<String> mostrarProducto() {
        SQLiteDatabase db = conextion.getWritableDatabase();

        ArrayList<String> listaProducto = new ArrayList<>();
        String producto = "";
        cursor = null;

        cursor = db.rawQuery("select * from producto", null);
        if (cursor.moveToFirst()) {
            do {
                producto=  String.valueOf(cursor.getInt(0)) +"\n";
                producto+=  (cursor.getString(1));
                producto+= (cursor.getString(2));
                listaProducto.add(producto);
            } while (cursor.moveToNext());
        }
        cursor.close();

        return listaProducto;
    }




    public boolean modficarProducto() {
        boolean correcto = false;

        SQLiteDatabase db = conextion.getWritableDatabase();
        try {
            db.execSQL("UPDATE "+ conextion.table_producto + " SET nombre= '"+nombre+"', descripcion=
'"+descripcion+"' WHERE id= '"+id+"'");
```

```
            correcto = true;
        }catch (Exception ex){
            ex.toString();
            correcto = false;
        } finally {
            db.close();
        }

        return correcto;
    }

    public boolean eliminarProducto() {
        boolean correcto = false;

        SQLiteDatabase db = conextion.getWritableDatabase();
        try {
            db.execSQL("DELETE FROM "+ conextion.table_producto + " WHERE id= '"+id+"'");
            correcto = true;
        }catch (Exception ex){
            ex.toString();
            correcto = false;
        } finally {
            db.close();
        }

        return correcto;
    }
}
```

8.2.5 DCatalogo

```
public class DCatalogo {
```

```
ConexionSQLite conextion;
Cursor cursor;



long id;
String nombre;



public DCatalogo(@Nullable Context context) {

   conextion = new ConexionSQLite(context);
}

public long getId() {
   return id;
}

public void setId(long id) {
   this.id = id;
}

public String getNombre() {
   return nombre;
}

public void setNombre(String nombre) {
   this.nombre = nombre;
}
```

```
public long guardarCatalogo() {
    long id = 0;
    try {
        SQLiteDatabase db = conextion.getWritableDatabase();
        ContentValues registro = new ContentValues();
        registro.put("nombre", nombre);
        id = db.insert(conextion.table_catalogo, null, registro);
    } catch (Exception ex) {
        ex.toString();
    }
    return id;
}

public ArrayList<String> mostrarCatalogo() {
    SQLiteDatabase db = conextion.getWritableDatabase();
    ArrayList<String> listaRepartidor = new ArrayList<>();
    String repartidor = "";
    cursor = null;
    cursor = db.rawQuery("select * from catalogo", null);
    if (cursor.moveToFirst()) {
        do {
            repartidor = String.valueOf(cursor.getInt(0)) + "\n";
            repartidor += (cursor.getString(1)) + "\n";
            repartidor += String.valueOf(cursor.getInt(2));
            listaRepartidor.add(repartidor);
        } while (cursor.moveToNext());
    }
    cursor.close();
```

```
        return listaRepartidor;
    }



    public boolean modficarCatalogo() {
        boolean correcto = false;
        SQLiteDatabase db = conextion.getWritableDatabase();
        try {
            db.execSQL("UPDATE " + conextion.table_catalogo + " SET nombre= '" + nombre +  "' 
WHERE id= '" + id + "'");
            correcto = true;
        } catch (Exception ex) {
            ex.toString();
            correcto = false;
        } finally {
            db.close();
        }

        return correcto;
    }

    public boolean eliminarCatalogo() {
        boolean correcto = false;
        SQLiteDatabase db = conextion.getWritableDatabase();
        try {
            //db.execSQL("DELETE FROM"  + conextion.table_repartidor + "' WHERE id= '"+id+"'");
            db.execSQL("DELETE FROM " + conextion.table_detalle_catalogo + " WHERE idcatalogo = '" 
+ id + "'");
            db.execSQL("DELETE FROM " + conextion.table_catalogo + " WHERE id = '" + id + "'");
            correcto = true;
```

```
        } catch (Exception ex) {
            ex.toString();
            correcto = false;
        } finally {
            db.close();
        }
        return correcto;
    }
}
```

8.2.6 DProforma

```
package com.example.emprende.emprende.Dato;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.graphics.Paint;

import android.graphics.pdf.PdfDocument;


import androidx.annotation.Nullable;


import com.example.emprende.emprende.db.ConexionSQLite;


import java.util.ArrayList;

public class DProforma {



    ConexionSQLite conextion;

    Cursor cursor;
```

```
    long id,idProducto,idCliente;
    int precio;

    public DProforma(@Nullable Context context) {
        conextion= new ConexionSQLite(context);
    }
/*
    public long obtenerNuevo_id() {
        long i=1;
        SQLiteDatabase db = conextion.getWritableDatabase();
        cursor = null;

        cursor = db.rawQuery("select count(*) from detallecatalogo where
idcatalogo='"+idCatalogo+"'", null);
        cursor.moveToFirst();
        long x=cursor.getInt(0);

        if (x!=0){
            cursor = db.rawQuery("select max(id) from detallecatalogo where
idcatalogo='"+idCatalogo+"'", null);
            cursor.moveToFirst();
            i=cursor.getInt(0)+1;
        }

        cursor.close();
        return i;
    }*/

    public long getIdProducto() {
```

```
    return idProducto;
}

public void setIdProducto(long idProducto) {
    this.idProducto = idProducto;
}

public long getId() {
    return id;
}

public void setId(long id) {
    this.id = id;
}

public int getPrecio() {
    return precio;
}

public void setIdCliente(long idCliente) {
    this.idCliente = idCliente;
}

public void setPrecio(int precio) {
    this.precio = precio;
}
```

```
public void agregar() {
    try {
        SQLiteDatabase db = conextion.getWritableDatabase();
        ContentValues registro = new ContentValues();
        registro.put("id", id);
        registro.put("idproducto", idProducto);
        registro.put("idcliente", idCliente);
        registro.put("precio", precio);
        id = db.insert(conextion.table_proforma, null, registro);

    } catch (Exception ex) {
        ex.toString();

    }
}

public ArrayList<String> listar() {
    SQLiteDatabase db = conextion.getWritableDatabase();

    ArrayList<String> listaDProforma = new ArrayList<>();
    String pProforma = "";
    cursor = null;

    cursor = db.rawQuery("select * from proforma",null);
    if (cursor.moveToFirst()) {
        do {
            pProforma = String.valueOf(cursor.getInt(0)) + "\n";
            pProforma += String.valueOf(cursor.getInt(1))+ "\n";
```

```
            pProforma += String.valueOf(cursor.getInt(2));
            listaDProforma.add(pProforma);
        } while (cursor.moveToNext());
    }
    cursor.close();

    return listaDProforma;
}

public void eliminar() {

    SQLiteDatabase db = conextion.getWritableDatabase();
    try {
        //db.execSQL("DELETE FROM"  + conextion.table_detalle_catalogo + "' WHERE idcatalogo=
'"+idCatalogo+"'");
        db.execSQL("DELETE FROM " + conextion.table_proforma + " WHERE  idproducto =
'"+idProducto+"'");
    } catch (Exception ex) {
        ex.toString();
    } finally {
        db.close();
    }

}

public PdfDocument cargarPDF() {
    SQLiteDatabase db = conextion.getWritableDatabase();
    ArrayList<String> listaDProforma = new ArrayList<>();
    String pProforma = "";
    cursor = null;
```

```
    int y=25;
    int i=1;
    PdfDocument pdfDocument = new PdfDocument();
    PdfDocument.PageInfo pageInfo = new PdfDocument.PageInfo.Builder(300, 600,1).create();
    PdfDocument.Page page = pdfDocument.startPage(pageInfo);
    cursor = db.rawQuery("select pf.id, p.nombre,pf.precio from proforma as pf , producto as p "+
"WHERE pf.idproducto = p.id  and pf.id = '"+ id +"'" , null);
    if (cursor.moveToFirst()) {
      do {
        pProforma = String.valueOf(cursor.getInt(0)) + "\t";
        pProforma += String.valueOf(cursor.getString(1))+ "\t";
        pProforma += String.valueOf(cursor.getInt(2));
        i++;
        //page.getCanvas().drawText(pCatalogo,10, 25, new Paint());
        page.getCanvas().drawText(pProforma,10, y*i, new Paint());
      } while (cursor.moveToNext());
    }
    cursor.close();
    pdfDocument.finishPage(page);
    return pdfDocument;
  }



  public PdfDocument cargarPDF(long id) {
    SQLiteDatabase db = conextion.getWritableDatabase();
    PdfDocument pdfDocument = new PdfDocument();
    Cursor cursor = null;
```

```
    try {
        PdfDocument.PageInfo pageInfo = new PdfDocument.PageInfo.Builder(300, 600,
1).create();
        PdfDocument.Page page = pdfDocument.startPage(pageInfo);
        Paint paint = new Paint();
        int y = 25;
        int i = 1;

        String query = "SELECT pf.id, p.nombre, pf.precio FROM proforma AS pf JOIN producto AS p
ON pf.idproducto = p.id WHERE pf.id = ?";
        cursor = db.rawQuery(query, new String[]{String.valueOf(id)});

        if (cursor.moveToFirst()) {
          do {
            String pProforma = cursor.getInt(0) + "\t" + cursor.getString(1) + "\t" + cursor.getInt(2);
            page.getCanvas().drawText(pProforma, 10, y * i++, paint);
          } while (cursor.moveToNext());
        }

        pdfDocument.finishPage(page);
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (cursor != null) {
            cursor.close();
        }
    }
    return pdfDocument;
  }
```

```
}
```

8.2.7 DDetalleCatalogo

```
public class DProforma {

   ConexionSQLite conextion;

   Cursor cursor;

   long id,idProducto,idCliente;

   int precio;


   public DProforma(@Nullable Context context) {

      conextion= new ConexionSQLite(context);

   }

/*

   public long obtenerNuevo_id() {

      long i=1;

      SQLiteDatabase db = conextion.getWritableDatabase();

      cursor = null;

      cursor = db.rawQuery("select count(*) from detallecatalogo where
idcatalogo='"+idCatalogo+"'", null);

      cursor.moveToFirst();

      long x=cursor.getInt(0);

      if (x!=0){

         cursor = db.rawQuery("select max(id) from detallecatalogo where
idcatalogo='"+idCatalogo+"'", null);

         cursor.moveToFirst();

         i=cursor.getInt(0)+1;

      }

       cursor.close();

      return i;

   }*/
```

```
public long getIdProducto() {
   return idProducto;
}
public void setIdProducto(long idProducto) {
   this.idProducto = idProducto;
}
public long getId() {
   return id;
}

public void setId(long id) {
   this.id = id;
}

public int getPrecio() {
   return precio;
}
public void setIdCliente(long idCliente) {
   this.idCliente = idCliente;
}
public void setPrecio(int precio) {
   this.precio = precio;
}
 public void agregar() {
   try {
      SQLiteDatabase db = conextion.getWritableDatabase();
      ContentValues registro = new ContentValues();
      registro.put("id", id);
      registro.put("idproducto", idProducto);
```

```
        registro.put("idcliente", idCliente);
        registro.put("precio", precio);
        id = db.insert(conextion.table_proforma, null, registro);

    } catch (Exception ex) {
        ex.toString();

    }
}

public ArrayList<String> listar() {
    SQLiteDatabase db = conextion.getWritableDatabase();
    ArrayList<String> listaDProforma = new ArrayList<>();
    String pProforma = "";
    cursor = null;
    cursor = db.rawQuery("select * from proforma",null);
    if (cursor.moveToFirst()) {
        do {
            pProforma = String.valueOf(cursor.getInt(0)) + "\n";
            pProforma += String.valueOf(cursor.getInt(1))+ "\n";
            pProforma += String.valueOf(cursor.getInt(2));
            listaDProforma.add(pProforma);
        } while (cursor.moveToNext());
    }
    cursor.close();

    return listaDProforma;
}
public void eliminar() {
```

```
        SQLiteDatabase db = conextion.getWritableDatabase();

        try {

            //db.execSQL("DELETE FROM"  + conextion.table_detalle_catalogo + "' WHERE idcatalogo=
'"+idCatalogo+"'");

            db.execSQL("DELETE FROM " + conextion.table_proforma + " WHERE  idproducto =
'"+idProducto+"'");

        } catch (Exception ex) {

            ex.toString();

        } finally {

            db.close();

        }

    }

    public PdfDocument cargarPDF() {

        SQLiteDatabase db = conextion.getWritableDatabase();

        ArrayList<String> listaDProforma = new ArrayList<>();

        String pProforma = "";

        cursor = null;

        int y=25;

        int i=1;

        PdfDocument pdfDocument = new PdfDocument();

        PdfDocument.PageInfo pageInfo = new PdfDocument.PageInfo.Builder(300, 600,1).create();

        PdfDocument.Page page = pdfDocument.startPage(pageInfo);

        cursor = db.rawQuery("select pf.id, p.nombre,pf.precio from proforma as pf , producto as p "+
"WHERE pf.idproducto = p.id  and pf.id = '"+ id +"'" , null);

        if (cursor.moveToFirst()) {

            do {

                pProforma = String.valueOf(cursor.getInt(0)) + "\t";

                pProforma += String.valueOf(cursor.getString(1))+ "\t";

                pProforma += String.valueOf(cursor.getInt(2));
```

```
            i++;
            //page.getCanvas().drawText(pCatalogo,10, 25, new Paint());
            page.getCanvas().drawText(pProforma,10, y*i, new Paint());
        } while (cursor.moveToNext());
    }
    cursor.close();
    pdfDocument.finishPage(page);
    return pdfDocument;
  }

  public PdfDocument cargarPDF(long id) {
    SQLiteDatabase db = conextion.getWritableDatabase();
    PdfDocument pdfDocument = new PdfDocument();
    Cursor cursor = null;
    try {
        PdfDocument.PageInfo pageInfo = new PdfDocument.PageInfo.Builder(300, 600,
1).create();
        PdfDocument.Page page = pdfDocument.startPage(pageInfo);
        Paint paint = new Paint();
        int y = 25;
        int i = 1;

        String query = "SELECT pf.id, p.nombre, pf.precio FROM proforma AS pf JOIN producto AS p
ON pf.idproducto = p.id WHERE pf.id = ?";
        cursor = db.rawQuery(query, new String[]{String.valueOf(id)});

        if (cursor.moveToFirst()) {
            do {
                String pProforma = cursor.getInt(0) + "\t" + cursor.getString(1) + "\t" + cursor.getInt(2);
                page.getCanvas().drawText(pProforma, 10, y * i++, paint);
```

```
            } while (cursor.moveToNext());
        }
        pdfDocument.finishPage(page);
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (cursor != null) {
            cursor.close();
        }
    }
    return pdfDocument;
  }
}
```

8.3 Negocio

8.3.1 NCliente

```
public class NCliente {
  DCliente dc;
  public NCliente(@Nullable Context context) {
    this.dc = new DCliente(context);
  }
  public void agregar(long id, String nombre, int telefono) {
    dc.setId(id);
    dc.setNombre(nombre);
    dc.setTelefono(telefono);
    dc.guardarCliente();
  }
```

```
    public void modificar(long id, String nombre, int telefono) {
        dc.setId(id);
        dc.setNombre(nombre);
        dc.setTelefono(telefono);
        dc.modficarCliente();
    }

    public boolean eliminar(long id) {
        if (id > 0) {  // Validar que el id sea un valor válido
            dc.setId(id);
            return dc.eliminarCliente();
        } else {
            System.out.println("Error: El id proporcionado no es válido.");
            return false;
        }
    }

    public ArrayList<String> listarCliente() {
        return dc.mostrarClientess();

    }
}
```

8.3.2 NRepartidor

```
public class NRepartidor {
    DRepartidor dr;
    public NRepartidor(@Nullable Context context) {
        this.dr = new DRepartidor(context);
    }
```

```
    public void agregar(long id, String nombre, int telefono) {
        dr.setId(id);
        dr.setNombre(nombre);
        dr.setTelefono(telefono);
        dr.guardarRepartidor();
    }
    public void modificar(long id, String nombre, int telefono) {
        dr.setId(id);
        dr.setNombre(nombre);
        dr.setTelefono(telefono);
        dr.modficarRepartidor();
    }

    public void eliminar(long id) {
        dr.setId(id);
        dr.eliminarRepartidor();
    }
    public ArrayList<String> listarRepartidor() {
        return dr.mostrarRepartidor();
    }
}
```

8.3.3 NEntrega

```
public class NEntrega {

    DEntrega d;
    public NEntrega(@Nullable Context context) {
        this.d = new DEntrega(context);
    }
```

```
    public void insertar(Long rep,Long idcliente, String latitud, String longitud) {
        d.setIdrep(rep);
        d.setIdcliente(idcliente);
        d.setLat(latitud);
        d.setLon(longitud);
        d.agregar();
    }


    public ArrayList<String> listar() {
        return d.mostrar();
    }
}
```

8.3.4 NProducto

```
public class NProducto {
    DProducto dp;
    public NProducto(@Nullable Context context) {
        this.dp = new DProducto(context);
    }

    public void agregar(long id, String nombre, String descripcion) {
        dp.setId(id);
        dp.setNombre(nombre);
        dp.setDescripcion(descripcion);
        dp.guardarProducto();
    }
```

```
    public void modificar(long id, String nombre, String descripcion) {
        dp.setId(id);
        dp.setNombre(nombre);
        dp.setDescripcion(descripcion);
        dp.modficarProducto();
    }

    public void eliminar(long id) {
        dp.setId(id);

        dp.eliminarProducto();
    }

    public ArrayList<String> listarProducto() {
        return dp.mostrarProducto();

    }
}
```

8.3.5 NCatalogo

```
public class NCatalogo {

    DCatalogo dr;



    public NCatalogo(@Nullable Context context) {
        this.dr = new DCatalogo(context);
    }
```

```
    public void agregar(long id, String nombre) {
        dr.setId(id);
        dr.setNombre(nombre);
        dr.guardarCatalogo();
    }

    public void modificar(long id, String nombre) {
        dr.setId(id);
        dr.setNombre(nombre);
        dr.modficarCatalogo();
    }

    public void eliminar(long id) {
        dr.setId(id);
        dr.eliminarCatalogo();
    }

    public ArrayList<String> listarCatalogo() {
        return dr.mostrarCatalogo();

    }
}
```

8.3.6 NProforma

```
public class NProforma {
    DProforma dato;

    public NProforma(@Nullable Context context) { this.dato = new DProforma(context);
    }
    //public long obtenerNuevo_id(long idCatalogo) {
```

```
//  return dato.obtenerNuevo_id();
//}

public void agregar(long id,long idProducto,long idCliente,int precio) {

   dato.setId(id);
   dato.setIdProducto(idProducto);
   dato.setIdCliente(idCliente);
   dato.setPrecio(precio);
   dato.agregar();
}

public ArrayList<String> listar(long idProducto) {
   dato.setIdProducto(idProducto);
   return dato.listar();
}

public void eliminar( long idProducto) {

   dato.setIdProducto(idProducto);
   dato.eliminar();
}

public PdfDocument cargarPDF(long id) {
   try {
      dato.setId(id);
      return dato.cargarPDF(id);
   } catch (Exception e) {
      e.printStackTrace();
```

```
            return null;
        }
    }
}
```

8.3.7 NDetalleCatalogo

```
public class NDetalleCatalogo {
    DDetalleCatalogo dato;

    public NDetalleCatalogo(@Nullable Context context) { this.dato = new
DDetalleCatalogo(context);
    }
    public long obtenerNuevo_id(long idCatalogo) {
        return dato.obtenerNuevo_id();
    }

    public void agregar(long idCatalogo, long id, long idProducto) {
        dato.setIdCatalogo(idCatalogo);
        dato.setId(id);
        dato.setIdProducto(idProducto);
        dato.agregar();
    }

    public ArrayList<String> listar(long idCatalogo) {
        dato.setIdCatalogo(idCatalogo);
        return dato.listar();
    }

    public void eliminar(long idCatalogo, long idProducto) {
        dato.setIdCatalogo(idCatalogo);
```

```
        dato.setIdProducto(idProducto);
        dato.eliminar();
    }

    public PdfDocument cargarPDF(long idCatalogo) {
        dato.setIdCatalogo(idCatalogo);
        return dato.cargarPDF();
    }
}
```

8.4 Presentacion

8.4.1 PCliente

```
public class PClienteActivity extends AppCompatActivity {

    NCliente negocio;

    Button btAgregar,btModificar,btEliminar;
    EditText txtId,txtNombre,txtTelefono;
    ListView lista;

    long id;
    String nombre;
    int telefono;

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

```
negocio= new NCliente(PClienteActivity.this);

btAgregar=findViewById(R.id.btAgregar);
btModificar=findViewById(R.id.btModificar);
btEliminar=findViewById(R.id.btEliminar);



txtId=findViewById(R.id.txtid);
txtNombre=findViewById(R.id.txtnombre);
txtTelefono=findViewById(R.id.txtTelefono);

lista=findViewById(R.id.lista);
listar();

btAgregar.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      obtener();
      agregar();
   }
});
btModificar.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      obtener();
      modificar();
   }
});
btEliminar.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View view) {
            obtener();
            eliminar();
        }
    });
}

private void obtener() {
    id = Integer.parseInt(txtId.getText().toString());
    nombre= txtNombre.getText().toString();
    telefono= Integer.valueOf(txtTelefono.getText().toString());
}

private void agregar()  {
    negocio.agregar(id, nombre, telefono);
    this.limpiar();
    listar();
}

private void listar() {
    ArrayList<String> dato=negocio.listarCliente();
    ArrayAdapter<String> adapter = new
ArrayAdapter<>(this,android.R.layout.simple_list_item_1,dato);
    lista.setAdapter(adapter);
}

private void limpiar() {
    try {
```

```
        // Validar que los campos no sean nulos antes de limpiar
        if (txtId != null && txtTelefono != null && txtNombre != null) {
            txtId.setText("");
            txtTelefono.setText("");
            txtNombre.setText("");
        } else {
            // Si algún campo es nulo, lanza una advertencia en el log o la interfaz
            System.out.println("Advertencia: Uno o más campos están nulos.");
        }
    } catch (Exception e) {
        // Manejar cualquier excepción inesperada
        System.out.println("Error al limpiar los campos: " + e.getMessage());
    }
}



private void modificar() {
    negocio.modificar(id,nombre,telefono);
    this.limpiar();
    listar();
}

private void eliminar() {
    if (id > 0) {  // Validar que el id sea un valor positivo
        boolean resultado = negocio.eliminar(id);
        if (resultado) {
            this.limpiar();
            listar();
            System.out.println("El cliente ha sido eliminado correctamente.");
```

```
            } else {
                System.out.println("Error al eliminar el cliente. Verifique si el id es correcto.");
            }
        } else {
            System.out.println("El id no es válido. Ingrese un id correcto.");
        }
    }



}
```

8.4.2 PRepartidor

```
public class PRepartidorActivity extends AppCompatActivity {

    NRepartidor negocio;

    Button btAgregar,btModificar,btEliminar;
    EditText txtId,txtNombre,txtTelefono;
    ListView lista;

    long id;
    String nombre;
    int telefono;

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

```
negocio= new NRepartidor(PRepartidorActivity.this);



btAgregar=findViewById(R.id.btAgregar);
btModificar=findViewById(R.id.btModificar);
btEliminar=findViewById(R.id.btEliminar);



txtId=findViewById(R.id.txtid);
txtNombre=findViewById(R.id.txtnombre);
txtTelefono=findViewById(R.id.txtTelefono);

lista=findViewById(R.id.lista);
listar();

btAgregar.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      obtener();
      agregar();
   }
});
btModificar.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      obtener();
      modificar();
   }
});
```

```
        btEliminar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                obtener();
                eliminar();
            }
        });
    }

    private void obtener() {
        id = Integer.parseInt(txtId.getText().toString());
        nombre= txtNombre.getText().toString();
        telefono= Integer.valueOf(txtTelefono.getText().toString());
    }

    private void agregar()  {

        negocio.agregar(id, nombre, telefono);

        this.limpiar();
        listar();
    }

    private void listar() {
        ArrayList<String> dato=negocio.listarRepartidor();
        ArrayAdapter<String> adapter = new
ArrayAdapter<>(this,android.R.layout.simple_list_item_1,dato);
        lista.setAdapter(adapter);
    }
```

```
private void limpiar() {
    try {
        // Validar que los campos no sean nulos antes de limpiar
        if (txtId != null && txtTelefono != null && txtNombre != null) {
            txtId.setText("");
            txtTelefono.setText("");
            txtNombre.setText("");
        } else {
            // Si algún campo es nulo, lanza una advertencia en el log o la interfaz
            System.out.println("Advertencia: Uno o más campos están nulos.");
        }
    } catch (Exception e) {
        // Manejar cualquier excepción inesperada
        System.out.println("Error al limpiar los campos: " + e.getMessage());
    }
}



private void modificar() {
    negocio.modificar(id,nombre,telefono);
    this.limpiar();
    listar();
}

private void eliminar() {
    negocio.eliminar(id);
    this.limpiar();
```

```
        listar();
    }
}
```

8.4.3 Pentrega

```
public class PEntregaActivity  extends AppCompatActivity {

    ListView lista;

    NEntrega negocio;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pentrega);
        negocio= new NEntrega(PEntregaActivity.this);
        lista=findViewById(R.id.lista);
        listar();

    }

    private void listar() {
        ArrayList<String> dato=negocio.listar();
        ArrayAdapter<String> adapter = new
ArrayAdapter<>(this,android.R.layout.simple_list_item_1,dato);
        lista.setAdapter(adapter);
    }

}
```

8.4.4 PProducto

```
public class PProductoActivity extends AppCompatActivity {
    private static final int REQUEST_SELECT_IMAGE = 1 ;
    NProducto negocio;

    Button btAgregar,btModificar,btEliminar,btimagen;
    EditText txtId,txtNombre,txtdescripcion;
    ListView lista;
    ImageView imageView;
    long id;
    String nombre;
    String descripcion;


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


        negocio= new NProducto(PProductoActivity.this);


        btAgregar=findViewById(R.id.btAgregar);
        btModificar=findViewById(R.id.btModificar);
        btEliminar=findViewById(R.id.btEliminar);
        btimagen=findViewById(R.id.btimagen);
        imageView= findViewById(R.id.imageView);
```

```
txtId=findViewById(R.id.txtid);
txtNombre=findViewById(R.id.txtnombre);
txtdescripcion=findViewById(R.id.txtdescripcion);

lista=findViewById(R.id.lista);
listar();

btAgregar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        obtener();
        agregar();
    }
});
btModificar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        obtener();
        modificar();
    }
});
btEliminar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        obtener();
        eliminar();
    }
});
btimagen.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            // Crea un intent para abrir la galería
            Intent intent = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);

            startActivityForResult(intent, REQUEST_SELECT_IMAGE);
        }
    });
  }

  private void obtener() {
    id = Integer.parseInt(txtId.getText().toString());
    nombre= txtNombre.getText().toString();
    descripcion= txtdescripcion.getText().toString();
  }

  private void agregar()  {

    negocio.agregar(id, nombre, descripcion);

    this.limpiar();
    listar();
  }

  private void listar() {
    ArrayList<String> dato=negocio.listarProducto();
    ArrayAdapter<String> adapter = new
ArrayAdapter<>(this,android.R.layout.simple_list_item_1,dato);
    lista.setAdapter(adapter);
```

```
    }


    private void limpiar() {
        if (txtId.getText().length() > 0 || txtdescripcion.getText().length() > 0 ||
txtNombre.getText().length() > 0) {
            txtId.setText("");
            txtdescripcion.setText("");
            txtNombre.setText("");
            System.out.println("Campos limpiados correctamente");
        } else {
            System.out.println("Los campos ya están vacíos");
        }
    }



    private void modificar() {
        negocio.modificar(id,nombre,descripcion);
        this.limpiar();
        listar();
    }

    private void eliminar() {
        negocio.eliminar(id);
        this.limpiar();
        listar();
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
```

```
        if (requestCode == REQUEST_SELECT_IMAGE && resultCode == RESULT_OK && data != null) {
            // Obtiene la URI de la imagen seleccionada
            Uri imageUri = data.getData();

            try {
                // Convierte la URI a un Bitmap (puedes omitir esto si no necesitas mostrar la imagen)
                Bitmap bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), imageUri);

                // Muestra el Bitmap en un ImageView (puedes omitir esto si no necesitas mostrar la imagen)
                imageView.setImageBitmap(bitmap);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

8.4.5 PCatalogo

```
public class PDetalleCatalogo extends AppCompatActivity {

    NDetalleCatalogo negocio;

    NProducto np;
    TextView textViewid, textViewnombre;
    ListView lista;
    Button btAgregar, btEliminar, btPDF, btW,btimagen;
    Spinner spinner1;
```

```
long id, idProducto, idCatalogo;
String nombre;
ImageView imageView;
ArrayList<String> dato;

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

    negocio = new NDetalleCatalogo(PDetalleCatalogo.this);
    np = new NProducto(PDetalleCatalogo.this);



    btAgregar = findViewById(R.id.btAgregar);
    btEliminar = findViewById(R.id.btEliminar);
    btPDF = findViewById(R.id.btPDF);
    btW = findViewById(R.id.btW);
    spinner1 = findViewById(R.id.spinner1);


    String catalogo = getIntent().getExtras().getString("catalogo");

    this.idCatalogo = Integer.valueOf(catalogo.split("\n")[0]);
    nombre = catalogo.split("\n")[1];
    cargarspinner();
```

```
        textViewid = findViewById(R.id.textViewid);
        textViewid.setText("Identificador N: " + idCatalogo);
        textViewnombre = findViewById(R.id.textViewnombre);
        textViewnombre.setText(nombre);

        lista = findViewById(R.id.lista);
        listar();
        ActivityCompat.requestPermissions(this, new String[]{READ_EXTERNAL_STORAGE,
WRITE_EXTERNAL_STORAGE}, PackageManager.PERMISSION_GRANTED);

        btAgregar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                obtener();
                agregar();
            }
        });

        btEliminar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                obtener();
                eliminar();
            }
        });

        btPDF.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
```

```
            crerPDF();
        }
    });

    btW.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            enviarW();
        }
    });



}


  private void enviarW() {
    String filePath =
Environment.getExternalStorageDirectory().getPath()+"/Download/"+nombre+".pdf";
    //String filePath = getFilesDir().getAbsolutePath() + "/Download/"+nombre+".pdf";
// Crea un URI a partir del archivo
    Uri fileUri = Uri.parse("file://" + filePath);
    //File outputFile = new File(Environment.getExternalStoragePublicDirectory
(Environment.DIRECTORY_DOWNLOADS), nombre+".pdf");
    File outputFile = new File(String.valueOf(filePath));
    Uri uri = Uri.fromFile(outputFile);
   /* Intent share = new Intent();
    share.setAction(Intent.ACTION_SEND);
    share.setType("application/pdf");
    share.putExtra(Intent.EXTRA_STREAM, uri);
    share.setPackage("com.whatsapp");
```

```
        startActivity(share);
*/
// Crea un intent con la acción de enviar
        /*Intent intent = new Intent(Intent.ACTION_SEND);
        intent.setType("text/plain");
        intent.putExtra(Intent.EXTRA_STREAM, fileUri);
        intent.setPackage("com.whatsapp"); // Especifica que se abrirá WhatsApp*/

// Comprueba si WhatsApp está instalado en el dispositivo
       /* if (intent.resolveActivity(getPackageManager()) != null) {
            startActivity(intent);
        } else {
            // WhatsApp no está instalado
            Toast.makeText(getApplicationContext(), "WhatsApp no está instalado",
Toast.LENGTH_SHORT).show();
        }*/

        /*String filePath = Environment.getExternalStorageDirectory().getPath() + "/Download/" +
nombre + ".pdf";
        File file = new File(filePath);

        Uri fileUri = FileProvider.getUriForFile(this, "com.example.emprende.emprende", file);*/

        Intent share = new Intent(Intent.ACTION_SEND);
        share.setType("application/pdf");
        share.putExtra(Intent.EXTRA_STREAM, fileUri);
        share.setPackage("com.whatsapp");

        share.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
```

```
        if (share.resolveActivity(getPackageManager()) != null) {
            startActivity(share);
        } else {
            Toast.makeText(getApplicationContext(), "WhatsApp no está instalado",
Toast.LENGTH_SHORT).show();
        }
    }



    private void crerPDF() {

        PdfDocument pdfDocument = negocio.cargarPDF(idCatalogo);
        String filePath =
Environment.getExternalStorageDirectory().getPath()+"/Download/"+nombre+".pdf";
        File file = new File(filePath);
        try {
            pdfDocument.writeTo(new FileOutputStream(file));
        } catch (IOException e) {
            e.printStackTrace();
        }
        pdfDocument.close();
    }


    private void cargarspinner() {
        dato = np.listarProducto();
        ArrayAdapter<String> adapter = new
ArrayAdapter<>(this,android.R.layout.simple_list_item_1,dato);
```

```
        spinner1.setAdapter(adapter);
    }
    private void obtener() {
        id =negocio.obtenerNuevo_id(idCatalogo) ;
        idProducto= Integer.valueOf(this.spinner1.getSelectedItem().toString().split("\n")[0]);
    }
    private void agregar()  {
        negocio.agregar(idCatalogo, id, idProducto);
        listar();
    }
    private void listar() {
        dato=negocio.listar(idCatalogo);
        ArrayAdapter<String> adapter = new
ArrayAdapter<>(this,android.R.layout.simple_list_item_1,dato);
        lista.setAdapter(adapter);
    }
    private void eliminar() {
        negocio.eliminar(idCatalogo,idProducto);
        listar();
    }
}
```

9 Bibliografía

1. Documentación oficial de Android.

https://developer.android.com/

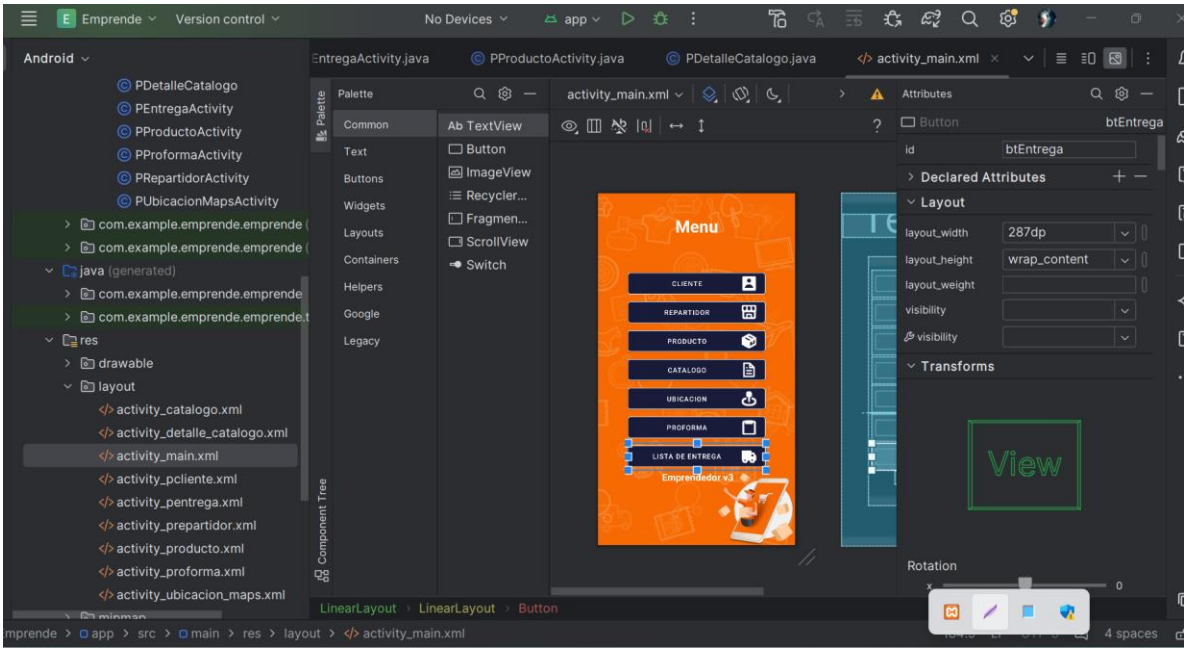
2. Tutorial de SQLite en Android.

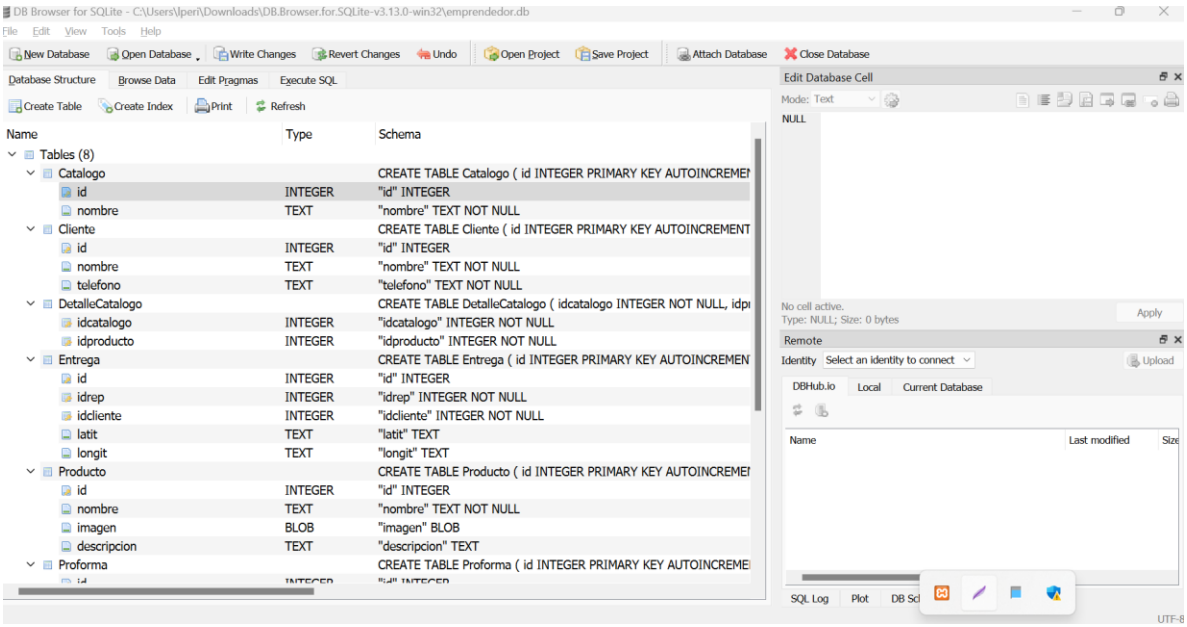
https://www.tutorialspoint.com/android/android_sqlite_database.htm

3. Android Code Labs.

https://codelabs.developers.google.com/?cat=Android

10 Anexo






Menu
CLIENTE
REPARTIDOR
PRODUCTO
CATALOGO
UBICACION
PROFORMA
LISTA DE ENTREGA
Emprendedor v3