

Cátedra: Lenguaje de programación JAVA

Ingeniería en Sistemas de Información

Trabajo Práctico Integrador

Gestión de ventas farmacia



Integrantes:

Legajo	Nombre y apellido	Email
47772	Ramiro Di Giacinti	dia.digiacinti.ramiro@gmail.com
46095	Facundo Braida	facundobraida98@gmail.com
47773	Bruno Mollo	dia.mollo.bruno@gmail.com

Docentes:

- Ricardo Tabacman
- Adrian Meca

CUU - Vender Medicamentos por Obra Social a un cliente

Objetivo: Realizar venta de productos

Actor: Principal: Vendedor

Iniciador: Vendedor

Camino Basico:

1. El vendedor busca en el sistema los datos del cliente ingresando su dni. El sistema valida que el cliente exista.
2. El vendedor busca el medicamento y la cantidad del mismo. El sistema controla la validez del medicamento y agrega a la venta el medicamento.
3. El vendedor cierra la venta ingresando el número de receta y el sistema imprime el comprobante de venta.

Nota: el paso 2 se repite por cada producto.

Camino Alternativo:

1. A. El sistema no encuentra el dni del cliente, se da de alta el cliente.
2. A. El sistema no encuentra el medicamento buscado, el sistema informa.

CUR - Emitir listado para las obras sociales

Objetivo: Facturar a las obras sociales

Actor: Principal: Administrador

Iniciador: Vendedor

Camino Basico:

1.El vendedor registra una venta <CUU: Vender Medicamentos por Obra Social a un cliente>

2.A fin de mes, el administrador indica al sistema que debe enviar los mails a las obras sociales con las ventas a reintegrar <CUU: Enviar mails a obras Sociales>

Nota: el paso 1 se repite por cada cliente.

Capturas de pantalla del sistema:



Ingresar cliente para venta

Dni del cliente:

Buscar Cliente

Nuevo Cliente

Volver al menu

This screenshot shows a search interface for a client. It has a text input field for the DNI number (43000000) and a 'Buscar Cliente' button. There are also 'Nuevo Cliente' and 'Volver al menu' buttons. The background is the same medical-themed pattern as the dashboard.

Nueva Venta

Nombre Medicamento
Eut
Euthrox

Cantidad
1

Medicamento Cantidad Precio con descuento Subtotal con descuento

Medicamento	Cantidad	Precio con descuento	Subtotal con descuento
Ibumax	2	\$840,00	\$1680,00

Total: \$1680,00

7899783215

Realizar Venta

Volver al menu

This screenshot shows a form for entering a new sale. It includes fields for the name of the medicine (Eut, Euthrox), quantity (1), and a table for adding items. The table has columns for medicine name, quantity, discounted price, and subtotal. A total amount of \$1680,00 is displayed at the bottom, along with a tracking number (7899783215). There are 'Realizar Venta' and 'Volver al menu' buttons. The background is the same medical-themed pattern.

Imprimir

Total: 1 hoja de papel



Impresora

Adobe PDF



Copias

1

Disposición

Vertical

Horizontal

Páginas

Todos

Por ejemplo, 1-5, 8, 11-13

Color

Color



Más opciones de configuración

Imprimir usando el diálogo de sistema... (Ctrl+Shift+P)

Fecha y Hora: 21-03-2023 10:34:03

Cliente: DiGia Ramiro

DNI: 43000000

Obra Social: Medife

Descuento: 30.0%

Medicamento | Precio Unidad | Cantidad | Subtotal Cliente

Ibumax	840,00	2	1680,00
--------	--------	---	---------

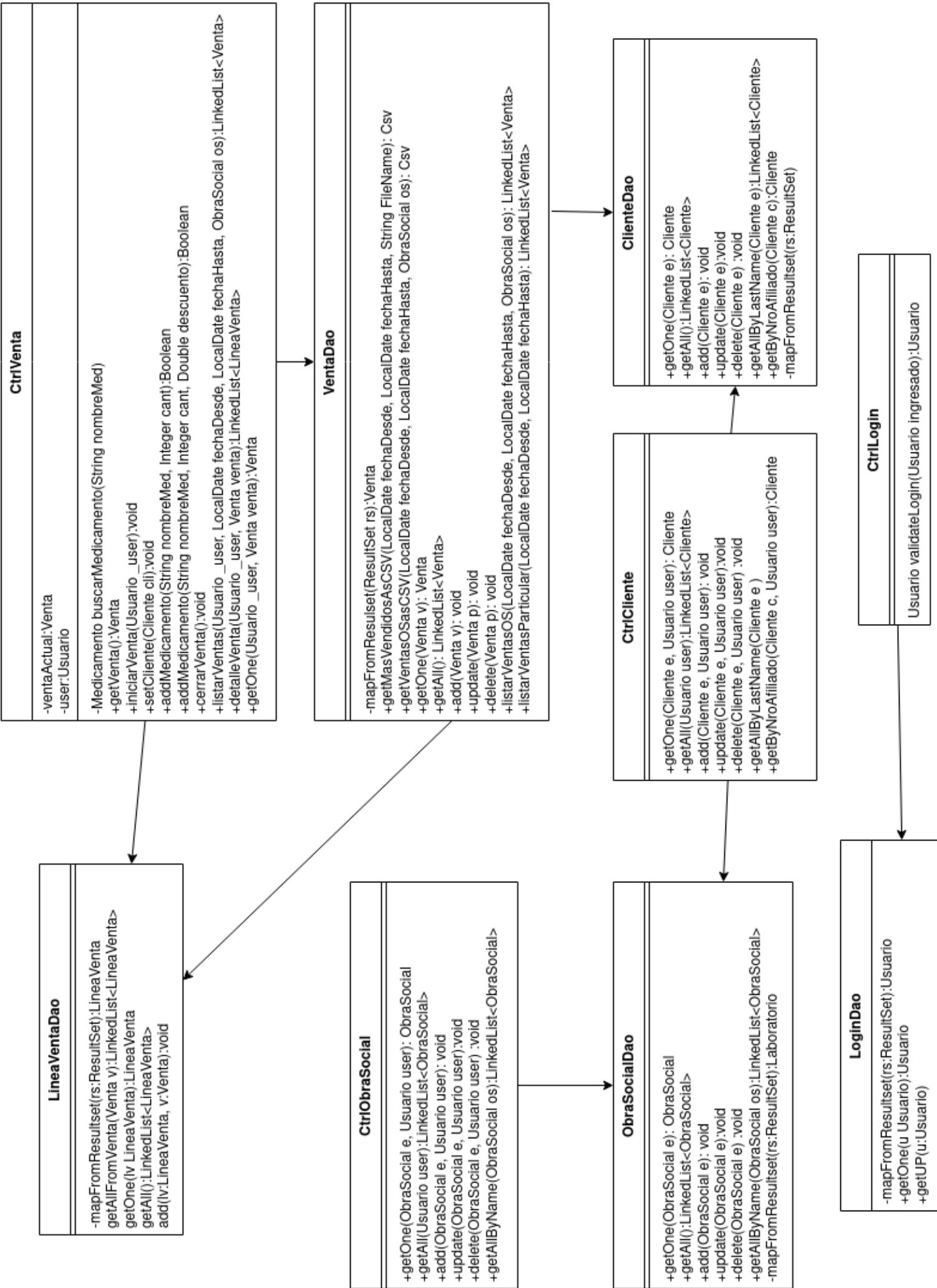
Total: 1680,00 Cajero/a: Pepito

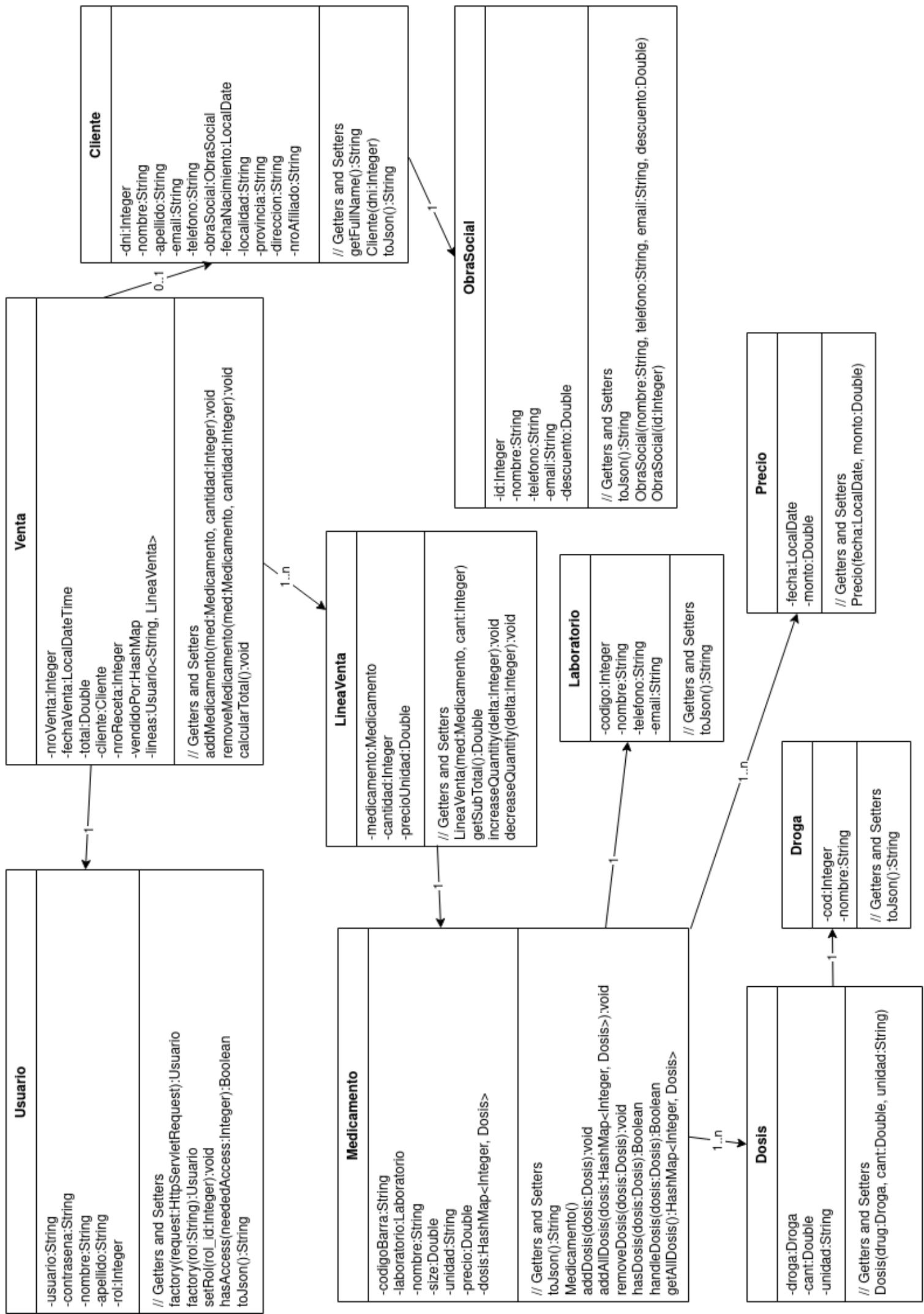
Gracias por su compra!

Imprimir

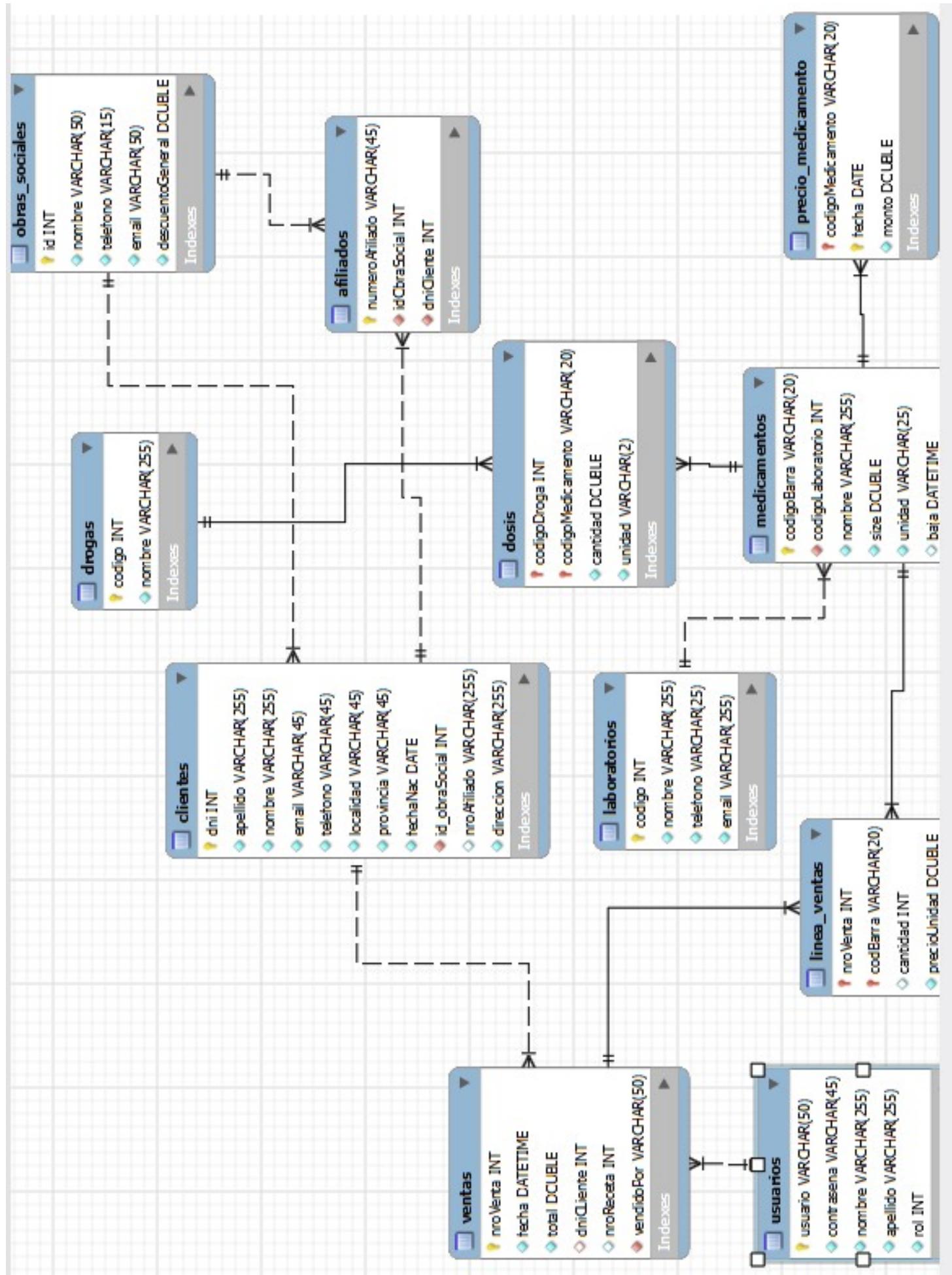
Cancelar







Modelo de Datos



VentaServlet

```
public class VentaServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    Usuario user=Usuario.factory(request);

    CtrlVenta con= (CtrlVenta) request.getSession().getAttribute("CtrlVenta");
    String query= request.getPathInfo();
    try {
        switch (query) {
            case "/iniciarVentaOS": {
                request.getSession().setAttribute("cliente", null);
                con = new CtrlVenta();
                con.iniciarVenta(user);
                request.getSession().setAttribute("CtrlVenta", con);
                request.getRequestDispatcher("/WEB-INF/ui-venta/buscarCliente.html").forward(request,
response);
                break;
            }
            default:
                throw new AppException("no hay",404);
        }
    } catch (Exception e) {
        ExceptionDispacher.manage(e, response);
    }
}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    CtrlVenta con= (CtrlVenta) request.getSession().getAttribute("CtrlVenta");
    CtrlCliente ccli = new CtrlCliente();
    CtrlObraSocial ctrlOS= new CtrlObraSocial();
    Usuario user=Usuario.factory(request);

    try {
        switch (request.getPathInfo()) {
            case "/addMedicamentoOS": {
                Integer cantidad= Integer.parseInt(request.getParameter("cantidad"));
                String nombreMed = request.getParameter("name_med");
                Double dcto = con.getVenta().getCliente().getObraSocial().getDescuento();
                Boolean medEncontrado=con.addMedicamento(nombreMed, cantidad,dcto);
                request.setAttribute("medEncontrado", medEncontrado);
            }
        }
    }
}
```

```

request.getRequestDispatcher("/WEB-INF/ui-venta/agregarMedicamentosOS.jsp").forward(request, response);
        break;
    }
    case "/buscarCliente": {
        Integer dni= Integer.parseInt(request.getParameter("dniCliente"));
        Cliente c = ccli.getOne(new Cliente(dni), user);
        if(c!=null) {
            request.getSession().setAttribute("cliente", c);
            con.setCliente(c);

request.getRequestDispatcher("/WEB-INF/ui-venta/agregarMedicamentosOS.jsp").forward(request, response);
        } else {
            request.getRequestDispatcher("/WEB-INF/ui-venta/buscarCliente.html").forward(request,
response);
        }
        break;
    }
    case "/cerrarVenta": {
        String strNroRec = request.getParameter("nroReceta");
        Integer nroRec = strNroRec==null?null:Integer.parseInt(request.getParameter("nroReceta"));
        Integer cantProd=con.getVenta().getLineas().size();

        if (cantProd == 0) {
            String action = strNroRec == null ? "agregarMedicamentos.jsp" :
"agregarMedicamentosOS.jsp";
            request.getRequestDispatcher("/WEB-INF/ui-venta/" + action).forward(request,
response);
        } else {
            con.getVenta().setNroReceta(nroRec);
            con.cerrarVenta();
        }

request.getRequestDispatcher("/WEB-INF/ui-venta/imprimirVenta.jsp").forward(request, response);
    }      break;
}
default:
    throw new AppException("no hay",404);
}catch (Exception e) {ExceptionDispacher.manage(e, response); }}}
```

CtrlVenta

```
public class CtrlVenta {  
  
    private VentaDao vDao= new VentaDao();  
    private MedicamentoDao mDao=new MedicamentoDao();  
    private Venta ventaActual;  
    private Usuario user;  
  
    private Medicamento buscarMedicamento(String nombreMed) throws AppException {  
        Medicamento med= new Medicamento();  
        med.setNombre(nombreMed);  
        return mDao.getByName(med);  
    }  
    public Venta getVenta() {  
        return ventaActual;  
    }  
    public void iniciarVenta(Usuario _user) throws AppException {  
        user=_user;  
        if(!_user.hasAccess(Usuario.VENDEDOR)) {throw new AppException("Debe ser vendedor", 401);}  
        ventaActual=new Venta();  
        ventaActual.setVendidoPor(user);  
    }  
    public void setCliente(Cliente cli) throws AppException {  
        if(!_user.hasAccess(Usuario.VENDEDOR)) {throw new AppException("Debe ser vendedor", 401);}  
        ventaActual.setCliente(cli);  
    }  
    public Boolean addMedicamento(String nombreMed, Integer cant, Double descuento) throws AppException  
{  
        if(!_user.hasAccess(Usuario.VENDEDOR)) {throw new AppException("Debe ser vendedor", 401);}  
        Medicamento med = buscarMedicamento(nombreMed);  
        if(med==null) {return false; }  
        med.setPrecio(med.getPrecio()*(1-(descuento/100)));  
        ventaActual.addMedicamento(med, cant);  
        return true;  
    }  
    public void cerrarVenta() throws AppException {  
        if(!_user.hasAccess(Usuario.VENDEDOR)) {throw new AppException("Debe ser vendedor", 401);}  
        ventaActual.setFechaVenta(LocalDateTime.now());  
        ventaActual.calcularTotal();  
        vDao.add(ventaActual);  
    }  
}
```

VentaDao

```
public class VentaDao extends Dao<Venta>{

    private ClienteDao cliDao= new ClienteDao();
    private LineaVentaDao lvDao= new LineaVentaDao();

    @Override
    public void add(Venta v) throws AppException {
        StatementWrapper tablaVenta= new StatementWrapper("INSERT INTO ventas"
                + "( fecha, total, dniCliente, nroReceta, vendidoPor) VALUES"
                + "(?,?,?,?,?)"
                .push(v.getFechaVenta())
                .push(v.getTotal())
                .push((v.getCliente()!=null)? v.getCliente().getDni() : null)
                .push((v.getNroReceta()!=null)? v.getNroReceta() : null)
                .push(v.getVendidoPor().getUsuario());

        doAddWithGeneratedKeys(tablaVenta, v);

        for(LineaVenta lv : v.getLineas()) {
            lvDao.add(lv, v);
        }
    }
}
```

LineaVentaDao

```
public class LineaVentaDao extends Dao<LineaVenta>{

    MedicamentoDao medDao= new MedicamentoDao();

    public void add(LineaVenta lv, Venta v) throws AppException {
        doModification(new StatementWrapper("INSERT INTO linea_ventas(nroVenta, codBarra, cantidad,
precioUnidad) VALUES (?,?,?,?,?)"
                .push(v.getNroVenta())
                .push(lv.getMedicamento().getCodigoBarra())
                .push(lv.getCantidad())
                .push(lv.getPrecioUnidad())
            );
    }
}
```

MedicamentoDao

```
public class MedicamentoDao extends Dao<Medicamento>{

    LaboratorioDao ldao = new LaboratorioDao();
    DosisDao ddao= new DosisDao();
    PrecioDao pDao=new PrecioDao();

    @Override
    protected Medicamento mapFromResulset(ResultSet rs) throws SQLException, AppException {
        Medicamento med=new Medicamento();
        Laboratorio lab = new Laboratorio();
        lab.setCodigo(rs.getInt("codigoLaboratorio"));
        med.setCodigoBarra(rs.getString("codigoBarra"));
        med.setLaboratorio(ldao.getOne(lab));
        med.setNombre(rs.getString("nombre"));
        med.setPrecio(pDao.getLatestPrice(med));

        med.setSize(rs.getDouble("size"));
        med.setUnidad(rs.getString("unidad"));
        med.addAllDosis(ddao.getDosisOfMedicamento(med));
        return med;
    }

    public Medicamento getByName(Medicamento m) throws AppException{
        Medicamento med=doGetOne(
            new StatementWrapper( "select * from medicamentos where nombre like ? and baja is
null")
            .push(m.getNombre()+"%")
        );
        if(med!=null) {
            med.setPrecio(pDao.getLatestPrice(med));
        }
        return med;
    }

    public LinkedList<Medicamento> getAllByPartialName(Medicamento obj) throws AppException {
        Connection con =DbConnector.getInstancia().getConn();
        PreparedStatement st=null;
        ResultSet rs=null;
        try {
            st= con.prepareStatement("select nombre from medicamentos where nombre like ? and baja is
null");
            st.setString(1, obj.getNombre()+"%");
            rs= st.executeQuery();
        }
        catch (SQLException e) {
            throw new AppException("Error en la consulta de medicamentos: "+e.getMessage());
        }
        finally {
            try {
                if(rs!=null)
                    rs.close();
                if(st!=null)
                    st.close();
                if(con!=null)
                    con.close();
            }
            catch (SQLException e) {
                throw new AppException("Error al cerrar las conexiones: "+e.getMessage());
            }
        }
        return new LinkedList<Medicamento>(rs);
    }
}
```

```
LinkedList<Medicamento> arr= new LinkedList<>();
while(rs.next()) {
    Medicamento m= new Medicamento();
    m.setNombre(rs.getString("nombre"));
    arr.push(m);
}

return arr;

}

catch (SQLException e) {
    e.printStackTrace();
    throw new AppException("Internal Database error", 500);
}

finally {
    try {
        if(st!=null)st.close();
        if(rs!=null)rs.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

}

}

}
```

PrecioDao

```
public class PrecioDao extends Dao<Precio>{

    @Override
    protected Precio mapFromResulset(ResultSet rs) throws SQLException {
        Precio precio=new Precio();
        precio.setFecha(rs.getDate("fecha").toLocalDate());
        precio.setMonto(rs.getDouble("monto"));
        return precio;
    }

    public Double getLatestPrice(Medicamento med) throws AppException {
        StatementWrapper stw=new StatementWrapper("select * from precio_medicamento "
                + "where codigoMedicamento=? order by fecha desc limit 1 ");
        stw.push(med.getCodigoBarra());
        return doGetOne(stw).getMonto();
    }

    public void add(Medicamento med, Precio p) throws AppException {
        StatementWrapper stw=new StatementWrapper("insert into precio_medicamento "
                + "(codigoMedicamento, fecha, monto) values (?,?,?)");
        stw.push(med.getCodigoBarra());
        stw.push(p.getFecha());
        stw.push(p.getMonto());
        doModification(stw);
    }

    public LinkedList<Precio> getPrices(Medicamento med) throws AppException {
        return doFindAll(new StatementWrapper("select * from precio_medicamento where
        codigoMedicamento=? "
                + "order by fecha desc").push(med.getCodigoBarra()));
    }

}
```

Venta

```
public class Venta implements HasAutogeneratedKey{  
  
    @AutoGenerated  
    private Integer nroVenta;  
    final static String dateFormat="yyyy-MM-dd";  
    public final static DateTimeFormatter dFormat= DateTimeFormatter.ofPattern(dateFormat);  
    private LocalDateTime fechaVenta;  
    private Double total;  
    private Cliente cliente;  
    private Integer nroReceta;  
    private HashMap<String, LineaVenta> lineas=new HashMap<String, LineaVenta>();    private Usuario  
vendidoPor;  
  
  
    public void addMedicamento(Medicamento med, Integer cantidad) {  
        String cod=med.getCodigoBarra();  
  
        if(lineas.containsKey(cod)) {  
            lineas.get(cod).increaseQuantity(cantidad);  
        }  
        else {  
            lineas.put(cod, new LineaVenta(med, cantidad));  
        }  
    }  
  
  
    public void removeMedicamento(Medicamento med, Integer cantidad) {  
        String cod=med.getCodigoBarra();  
        if(lineas.containsKey(cod)) {  
            lineas.get(cod).decreaseQuantity(cantidad);  
  
            if(lineas.get(cod).getCantidad()<=0) {  
                lineas.remove(cod);  
            }  
        }  
    }  
  
  
    public void calcularTotal() {  
        this.total=0.;  
        for(LineaVenta lv : lineas.values()) {  
            total+=lv.getSubTotal();  
        }  
    }  
}
```

```
public LinkedList<LineaVenta> getLineas(){
    return new LinkedList<LineaVenta>(lineas.values());
}

public Integer getNroVenta() {
    return nroVenta;
}
public void setNroVenta(Integer nroVenta) {
    this.nroVenta = nroVenta;
}
public LocalDateTime getFechaVenta() {
    return fechaVenta;
}
public void setFechaVenta(LocalDateTime fechaVenta) {
    this.fechaVenta = fechaVenta;
}
public Double getTotal() {
    calcularTotal();
    return total;
}
public Double getTotalDB() {
    return this.total;
}
public void setTotal(Double total) {
    this.total = total;
}
public Cliente getCliente() {
    return this.cliente;
}
public void setCliente(Cliente cliente) {
    this.cliente = cliente;
}
public Integer getNroReceta() {
    return nroReceta;
}
public void setNroReceta(Integer nroReceta) {
    this.nroReceta = nroReceta;
}

public Usuario getVendidoPor() {
    return vendidoPor;
}
public void setVendidoPor(Usuario vendidoPor) {
    this.vendidoPor = vendidoPor;
}}
```

Usuario

```
public class Usuario implements Jsonable{

    public static final int ANYONE=1789;
    public static final int ADMIN=0;
    public static final int VENDEDOR=1;

    public static Usuario factory(HttpServletRequest request) {
        Usuario u=(Usuario) request.getSession().getAttribute("user");
        if(u==null) {
            return new Usuario();
        }
        else {
            return u;
        }
    }

    public static Usuario factory(String rol) {
        Usuario u= new Usuario();
        u.setRol(Integer.parseInt(rol));
        return u;
    }

    private String usuario;
    private String contrasena;
    private String nombre;
    private String apellido;
    private Integer rol;

    public void setRol(Integer rol_id) {
        if(rol_id==0) {
            this.rol=ADMIN;
        }
        else if(rol_id==1) {
            this.rol=VENDEDOR;
        }
    }

    public Boolean hasAccess(Integer NeededAccess) throws AppException {
        if(NeededAccess==Usuario.ANYONE){ return true; }
        if(this.rol==null) { return false; }

        switch (NeededAccess) {
        case Usuario.ADMIN:
            if(this.rol==Usuario.ADMIN) { return true; }
            else {return false; }
        case Usuario.VENDEDOR:
            if(this.rol==Usuario.ADMIN || this.rol==Usuario.VENDEDOR) { return true; }
        }
    }
}
```

```

        else { return false; }

    default:
        throw new AppException("Unexpected value: " + NeededAccess, 500);
    }
}

@Override
public String toJson() {
    JsonMaker jm= new JsonMaker();
    jm.set("user", usuario);
    jm.set("rol", rol);
    return jm.getJSONObject();
}
}

```

LineaVenta

```

public class LineaVenta {

    private Medicamento medicamento;
    private Integer cantidad;
    private Double precioUnidad;

    public LineaVenta(){}
    public LineaVenta(Medicamento med, Integer cant) {
        this.medicamento=med;
        precioUnidad=med.getPrecio();

        this.cantidad=cant;
    }

    public Double getSubTotal() {
        return cantidad*precioUnidad;
    }

    public void increaseQuantity(Integer delta) {
        this.cantidad+=delta;
    }

    public void decreaseQuantity(Integer delta) {
        this.cantidad-=delta;
    }
}

```

Venta

```
public class Venta implements HasAutogeneratedKey{  
  
    @AutoGenerated  
    private Integer nroVenta;  
    final static String dateFormat="yyyy-MM-dd";  
    public final static DateTimeFormatter dFormat= DateTimeFormatter.ofPattern(dateFormat);  
    private LocalDateTime fechaVenta;  
    private Double total;  
    private Cliente cliente;  
    private Integer nroReceta;  
    private HashMap<String, LineaVenta> lineas=new HashMap<String, LineaVenta>();  
    private Usuario vendidoPor;  
  
    public void addMedicamento(Medicamento med, Integer cantidad) {  
        String cod=med.getCodigoBarra();  
  
        if(lineas.containsKey(cod)) {  
            lineas.get(cod).increaseQuantity(cantidad);  
        }  
        else {  
            lineas.put(cod, new LineaVenta(med, cantidad));  
        }  
    }  
    public void removeMedicamento(Medicamento med, Integer cantidad) {  
        String cod=med.getCodigoBarra();  
        if(lineas.containsKey(cod)) {  
            lineas.get(cod).decreaseQuantity(cantidad);  
  
            if(lineas.get(cod).getCantidad()<=0) {  
                lineas.remove(cod);  
            }  
        }  
    }  
    public void calcularTotal() {  
        this.total=0.;  
        for(LineaVenta lv : lineas.values()) {  
            total+=lv.getSubTotal();  
        }  
    }  
    public LinkedList<LineaVenta> getLineas(){  
        return new LinkedList<LineaVenta>(lineas.values());  
    }  
    public Double getTotal() {  
        calcularTotal();  
        return total;  
    }  
}
```

Repositorio con el código



<https://github.com/BrunoMollo/Proyecto-Java>