ANEXO C: MANUAL TÉCNICO

C.1 INTRODUCCIÓN

La finalidad del manual técnico es proporcionar la lógica con que se ha desarrollado la aplicación.

C.2 Objetivo

Proporcionar una guía de clases y código de programación utilizado para la integración de las tecnologías utilizadas para conseguir los objetivos, para que se pueda en cualquier momento corregir errores o seguir implementando el Sistema.

C.3 Clases, Funciones y código de programación utilizados para el desarrollo de la Aplicación

Para entender los procesos internos que realiza el sistema debemos entender cada uno de los objetos generados dentro de la estructura de la aplicación **SanAntonioStore**. A continuación se describe los archivos de configuración de la aplicación web.

Archivo de configuración **persistence.xml**: Permite realizar la conexión con la base de datos y generar el mapeo de las tablas de la base de datos en clases POJOs java.

Este archivo se encuentra en el directorio de la aplicación **SanAntonioStore\src\conf\persistence.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
   <persistence-unit name="SanAntonioStorePU" transaction-type="JTA">
      <jta-data-source>java:/jboss/datasources/SanAntonioStoreDS</jta-data-source>
            <exclude-unlisted-classes>false</exclude-unlisted-classes>
      <properties>
         <property name="javax.persistence.target-database" value="PostgreSQL" />
         <property name="javax.persistence.logging.level" value="INFO" />
         <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver" />
         <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://localhost:5432/SanAntonioStore" />
         <property name="javax.persistence.jdbc.user" value="administrador" />
         <property name="javax.persistence.jdbc.password" value="admin" />
      </properties>
   </persistence-unit>
</persistence>
```

Archivo de configuración **web.xml**: Permite parametrizar valores iniciales para arrancar la aplicación. Este archivo se encuentra en el directorio de la aplicación **SanAntonioStore\web\WEB-INF\web.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
  <context-param><param-name>javax.faces.PROJECT_STAGE</param-name><param-value>Production</param-value>
  </context-param>
  <servlet>
        <servlet-name>Faces Servlet</servlet-name>servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup></servlet>
  <servlet-mapping><servlet-name>Faces Servlet</servlet-name><url-pattern>*.xhtml</url-pattern> </servlet-mapping>
  <session-config><session-timeout>120</session-timeout></session-config>
  <filter><filter-name>AuthAdminFilter</filter-name><filter-class>com.utn.security.AuthAdminFilter</filter-class> </filter>
  <filter-mapping><filter-name>AuthAdminFilter</filter-name><url-pattern>/administration/*</url-pattern></filter-mapping>
  <welcome-file-list><welcome-file>index.xhtml</welcome-file></welcome-file-list>
  <filter>
        <filter-name>PrimeFaces FileUpload Filter</filter-name>
    <filter-class>org.primefaces.webapp.filter.FileUploadFilter</filter-class>
    <init-param> <param-name>thresholdSize</param-name><param-value>512000</param-value></init-param>
    <init-param><param-name>uploadDirectory</param-name><param-value>C:\etc</param-value></init-param>
  </filter>
  <filter-mapping>
        <filter-name>PrimeFaces FileUpload Filter</filter-name><servlet-name>Faces Servlet</servlet-name>
  </filter-mapping>
</web-app>
```

Archivo de configuración changeset-sas.xml: Permite conectar con el repositorio de datos para administrar las reglas Drools desde Guvnor. Este archivo se encuentra en el directorio de la aplicación **SanAntonioStore\src\java\com\utn\drools\rules\changeset-sas.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<change-set xmlns='http://drools.org/drools-5.0/change-set'
   xmlns:xs='http://www.w3.org/2001/XMLSchema-instance'
    xs:schemaLocation='http://drools.org/drools-5.0/change-set
http://anonsvn.jboss.org/repos/labs/labs/jbossrules/trunk/drools-api/src/main/resources/change-set-1.0.0.xsd' >
   <add>
      <resource source='http://localhost:8080/jboss-
brms/org.drools.guvnor.Guvnor/package/com.utn.jpa.model.entities/SanAntonioStoreDrools' type='PKG'
        basicAuthentication="enabled"  username="admin" password="admin" />
   </add>
</change-set>
```

Archivo de reglas de negocio **CarroCompras.drl**: Almacena todas las reglas de negocio de la aplicación para ser ejecutados por el motor de reglas de negocio Drools almacenado en el repositorio de datos. Este archivo se encuentra en el directorio de la aplicación **SanAntonioStore\src\java\com\utn\drools\rules\CarroCompras.drl**

```
rule "R1: Ingreso de Artesanias al Carrito de Compras"  salience 30 lock-on-active true
when
	$cliente    : Cliente (estadoCliente == 'S')
	$pedido     : Pedido(idCliente.id == $cliente.id , $idPedido : id )
	$detalle    : DetallePedido(idPedido.id == $idPedido, idArtesania.getExistencias() > 0 )
   then
	modify( $detalle ){  setImporteDetalle(($detalle.getCantidad() * $detalle.getPrecio() ) )  };
	modify( $pedido ){  setImporte(($detalle.cantidad * $detalle.idArtesania.costo) + $pedido.getImporte() )   };
end
rule "R2: Descuento del 10% del Total de la Compra, cuando el Total es Mayor o Igual que 50 dÃ³lares"   salience 27
when
	$cliente : Cliente (estadoCliente == 'S')
	$pedido  : Pedido(idCliente.id == $cliente.id , importe >= 50 , $idPedido : id )
	$total : Number() from accumulate( DetallePedido( idArtesania.getExistencias() > 0, idPedido.id == $idPedido, $importe :
importeDetalle ), sum($importe ) )
then
	$pedido.setImporte($total - ($total * 0.10));
end
```

```
rule "R3: Total de la Compra"  salience 28 lock-on-active true
when
	$cliente : Cliente (estadoCliente == 'S')
	$pedido  : Pedido(idCliente.id == $cliente.id , $idPedido : id)
	$total : Number() from accumulate( DetallePedido( idArtesania.getExistencias() > 0, idPedido.id == $idPedido, $importe :
importeDetalle ), sum($importe ) )
then
	$pedido.setImporte($total);
end
rule "R4: Descuento del 5% del Total de la Compra si la Cantidad es Mayor de 2 ArtesanÃ-as Diferentes" salience -15
	lock-on-active true
when
	$cliente : Cliente (estadoCliente == 'S')
	$pedido  : Pedido(idCliente.id == $cliente.id , $idPedido : id)
	$detalleList : ArrayList(size >= 2) from collect(DetallePedido(idArtesania.getExistencias() > 0,  idPedido.id == $idPedido))
then
	$pedido.setImporte($pedido.getImporte() - ($pedido.getImporte() * 0.05));
 end
```

```
rule "R5: Descuento del 2% si el Tipo de Cliente es Minorista"          salience -19 lock-on-active true
when
      $cliente : Cliente (categoria == 'Minorista' , estadoCliente == 'S');
      $pedido  : Pedido(idCliente == $cliente );
then
      $pedido.setImporte(  $pedido.getImporte() - ($pedido.getImporte() * 0.02));
end
rule "R6: Descuento del 3% si la Artesania esta en Oferta" salience 29   lock-on-active true
when
      $cliente : Cliente ( estadoCliente == 'S')
      $pedido  : Pedido(idCliente == $cliente )
      $detalle : DetallePedido(idArtesania.getExistencias() > 0, idArtesania.oferta == 'S' )
   then
      modify(  $detalle ){  setDescuento(($detalle.getImporteDetalle() * 0.03)),
         setImporteDetalle($detalle.getImporteDetalle() - $detalle.getDescuento() )
      };
end
```

Integración del motor de reglas de producción Drools 5.5 en la arquitectura MVC a través de un componente java EJB.

En la capa del Modelo del patrón de diseño MVC se realizó la integración con la base de datos Postgres a través de componentes EJB de Entidad y componentes DAO (DAO: Objeto de Acceso a Datos Java) esto por cada tabla de la base de datos.

```
package com.utn.jpa.model.entities;
@Entity
@Table(name = "artesania")
public class Artesania implements Serializable {


}
```

Dichos EJBs de Entidad presentan la información en forma de objetos q son administrados por los componentes DAO y mediante herencia de una clase y una interface genérica podrán ser consumidos.

```
package com.utn.ejb.dao.business;
public interface GenericDAOInterface<T> {
    T crear(T entidad);
    T actualizar(T entidad);
    void eliminar(T entidad);
    T buscarPorId(Object id);
}
```

```
package com.utn.ejb.dao.business;
public class GenericDAO<T> implements GenericDAOInterface<T> {
    @PersistenceContext(unitName = "SanAntonioStorePU")
    protected EntityManager em;
    @Override
    public T crear(T entidad) {
        em.persist(entidad);
        return entidad;
    }
    @Override
    public T actualizar(T entidad) {
        return em.merge(entidad);
    }
@Override
    public void eliminar(T entidad) {
        em.remove(em.merge(entidad));
    }
    @Override
    public T buscarPorId(Object id) {
        Class<T> claseEntidad;
        claseEntidad = (Class<T>) ((ParameterizedType) getClass().getGenericSuperclass()).getActualTypeArguments()[0];
        return em.find(claseEntidad, id);
    }
}
```

Aquí se muestra un ejemplo de un componente DAO **ArtesaniaDAO** que encapsula a la entidad **Artesania**

```
package com.utn.ejb.dao.business;
import javax.ejb.Stateless;
import com.utn.jpa.model.entities.Artesania;
import java.util.List;
import javax.persistence.Query;
@Stateless
public class ArtesaniaDAO extends GenericDAO<Artesania> implements ArtesaniaDAOLocal {
    @Override
    public List<Artesania> buscarPorNombre(String nombre) {
        Query q = em.createQuery("SELECT object(a) FROM Artesania AS a "    + "  WHERE (a.nombre LIKE :nombre)");
        q.setParameter("nombre", "%" + nombre + "%");
        return q.getResultList();
    }
    @Override
    public List<Artesania> buscarPorTipoArtesania(String tipoArtesania) {
        Query q = em.createQuery("SELECT object(a) FROM Artesania AS a " + "  WHERE (a.tipoArtesania LIKE :tipoArtesania)");
        q.setParameter("tipoArtesania", "%" + tipoArtesania + "%");
        return q.getResultList();
    }
    @Override
    public List<Artesania> buscarPorCategoria(Integer idCategoria) {
        Query q = em.createQuery("SELECT object(a) FROM Artesania AS a " + "  WHERE (a.idCategoria.id = :idCategoria)");
        q.setParameter("idCategoria", idCategoria);
        return q.getResultList();
    }
    @Override
    public List<Artesania> buscarTodos() {
        Query q = em.createQuery("SELECT object(a) FROM Artesania AS a");
        return q.getResultList();
    }
    @Override
    public List<Artesania> buscarTodosRango(int inicio, int tamano) {
        Query q = em.createQuery("SELECT object(a) FROM Artesania AS a");
        q.setFirstResult(inicio);
        q.setMaxResults(tamano);
        return q.getResultList();
    }
    /**
     *
     * @return
     */
    @Override
    public int contador() {
        Query q = em.createQuery("SELECT count(a) FROM Artesania AS a");
        return q.getFirstResult();
    }
    /**
     *
     * @param IdArtesania
     * @return
     */
    @Override
    public List<Artesania> buscarPorId(Integer IdArtesania) {
        Query q = em.createNamedQuery("Artesania.findById");
        q.setParameter("id", IdArtesania);
        return q.getResultList();
    }

}
```

Interface DAO **ArtesaniaDAOLocal** para acceder desde la capa de Controlador

```
package com.utn.ejb.dao.business;
import java.util.List;
import javax.ejb.Local;
import com.utn.jpa.model.entities.Artesania;
@Local
public interface ArtesaniaDAOLocal extends GenericDAOInterface<Artesania>{
    List<Artesania> buscarPorNombre(String nombre);
    List<Artesania> buscarTodos();
    List<Artesania> buscarPorTipoArtesania(String tipoArtesania);
    List<Artesania> buscarPorCategoria(Integer idCategoria);
    List<Artesania> buscarTodosRango(int inicio, int tamano);
    List<Artesania> buscarPorId(Integer IdArtesania) ;
    int contador();
}
```

Una vez que los datos de la base de datos se cargan en la capa del modelo ya se tiene los objetos para manipularlos en la capa de Controlador que interactúa con la capa del motor de reglas de producción a través de un EJB de Sesión **CompraService** en el cual se crea una instancia del motor Drools conjuntamente creando instancias para todos los objetos que estén referenciados dentro de las reglas de negocio para ser consumidos por el motor de reglas de producción cuando se crea una sesión.

```
package com.utn.ejb.business;
import javax.ejb.EJB;
import javax.ejb.Stateful;
import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import org.drools.KnowledgeBase;
import org.drools.KnowledgeBaseFactory;
import org.drools

@Stateful
public class CompraService implements CompraServiceLocal {
    private static final String PERSISTENCE_UNIT_NAME = "SanAntonioStorePU";
    private EntityManagerFactory entityManagerFactory;
    private EntityManager entityManager;
    private JpaEntityManager jpa;
    private ServerSession serverSession;
    UnitOfWork uow;
    @EJB
    ArtesaniaDAOLocal artesaniaFacade;
    @EJB
    ClienteDAOLocal clienteFacade;
    @EJB
    DetallePedidoFacade detallePedidoFacade;
    @EJB
    PedidoFacade pedidoFacade;
    private List<ArtesaniaCompra> carroCompra = null;
    private Cliente cliente = null;
    private Cliente clienteActual = null;
    private boolean confirmado = false;
    private Pedido pedido = null;
    private List<DetallePedido> listCarroCompra = null;
    private Pago pago = null;
    @Inject
    private Catalog catalog;
    public Catalog getCatalog() {
        return catalog;
    }
    public void setCatalog(Catalog catalog) {
        this.catalog = catalog;
    }
    @Inject
    private UsuariosController usuariosController;
    public UsuariosController getUsuariosController() {
        return usuariosController;
    }
    public void setUsuariosController(UsuariosController usuariosController) {
        this.usuariosController = usuariosController;
    }
    @Inject
    private PedidoCompraController pedidoCompraController;
    public PedidoCompraController getPedidoCompraController() {
        return pedidoCompraController;
    }
    public void setPedidoCompraController(PedidoCompraController pedidoCompraController) {
        this.pedidoCompraController = pedidoCompraController;
    }
…
```

El siguiente método **ingresarHechosAMemoriaDeTrabajo**() que forma parte de la clase **CompraService** crea una sesión para la base de conocimiento del motor de reglas de producción en la cual se registran los hechos a la memoria de trabajo y estos hechos se afectarán mediante las reglas de negocio con las que serán emparejadas. Cuando el cliente registra una artesanía en el carrito de compras se dispara el método para calcular los valores de acuerdo a las reglas de negocio

```
@Override
    public void ingresarHechosAMemoriaDeTrabajo() {
        try {
            KnowledgeBase kbase = CompraService.readKnowledgeBase();
            StatefulKnowledgeSession ksession = kbase.newStatefulKnowledgeSession();
            KnowledgeRuntimeLogger logger = KnowledgeRuntimeLoggerFactory.newFileLogger(ksession, "test");
            this.cliente = this.usuariosController.getClienteActual();
            if (this.cliente == null) {
                Usuario u = new Usuario();
                this.cliente = new Cliente();
                List<Cliente> listCliente = new ArrayList<Cliente>();
                listCliente.add(this.cliente);
                u.setClienteList(listCliente);
            }
            ksession.insert(this.cliente);
            this.pedido = new Pedido();
            int count = pedidoFacade.count();
            this.pedido.setId(count + 1);
            this.pedido.setIdCliente(this.cliente);
            ksession.insert(this.pedido);

            int countDetalle = detallePedidoFacade.count();
            for (ArtesaniaCompra item : this.carroCompra) {
                DetallePedido $detalle = new DetallePedido();
                $detalle.setCantidad(new BigDecimal(item.getCantidad()));
                $detalle.setDescuento(new BigDecimal(0));
                $detalle.setEstadoDetallePedido("S");
                $detalle.setId(++countDetalle);
                $detalle.setIdArtesania(item.getArtesania());
                $detalle.setIdPedido(this.pedido);
                BigDecimal d1 = new BigDecimal(item.getCantidad());
                $detalle.setImporteDetalle(d1.multiply(item.getArtesania().getCosto()));
                $detalle.setPrecio(item.getArtesania().getCosto());
                $detalle.setReferencia("Buena");//revisar
                ksession.insert($detalle);
            }
            ksession.fireAllRules();
            Collection<?> factHandles = ksession.getFactHandles();
            List<FactHandle> f = new ArrayList<FactHandle>();
            f.addAll((Collection<? extends FactHandle>) factHandles);

            this.carroCompra = new ArrayList<ArtesaniaCompra>();
            this.listCarroCompra = new ArrayList<DetallePedido>();
            countDetalle = detallePedidoFacade.count();
            for (FactHandle fh : f) {
                Object o = (Object) ksession.getObject(fh);
                String tipo = o.getClass().getName();
                int indice = tipo.lastIndexOf(".");
                tipo = tipo.substring(indice + 1);
                if (tipo.equals("DetallePedido")) {
                    DetallePedido dp = (DetallePedido) o;
                    dp.setId(++countDetalle);
                    this.listCarroCompra.add(dp);
                    this.carroCompra.add(new ArtesaniaCompra(dp.getIdArtesania(),
                            dp.getCantidad().longValue(), dp.getPrecio(), dp.getDescuento()));
                }
                if (tipo.equals("Pedido")) {                this.pedido = (Pedido) o;               }
                if (tipo.equals("Cliente")) {               this.cliente = (Cliente) o;              }
            }
            this.pedido.setCiudadPedido(pedidoCompraController.getCiudadPedido());
            this.pedido.setDireccionPedido(pedidoCompraController.getDireccionPedido());
            this.pedido.setNombreEmbarque(pedidoCompraController.getNombreEmbarque());
            this.pedido.setProvinciaPedido(pedidoCompraController.getProvinciaPedido());
            this.pedido.setPaisPedido(pedidoCompraController.getPaisPedido());
            this.pedido.setFechaPedido(new Date());
            this.pedido.setEstadoPedido("S");
            this.pedido.setDetallePedidoList(this.listCarroCompra);
            this.pago = new Pago();
            this.pago.setCodigoSeguridadTarjeta(pedidoCompraController.getCodigoSeguridadTarjeta());
            this.pago.setCodigoTarjeta(pedidoCompraController.getCodigoTarjeta());
            this.pago.setCuentaBancaria(pedidoCompraController.getCuentaBancaria());
            this.pago.setDescripcion(pedidoCompraController.getDescripcion());
            this.pago.setEntidadBancarea(pedidoCompraController.getEntidadBancarea());
            this.pago.setIdPedido(this.pedido);
            this.pago.setReembolso(pedidoCompraController.getReembolso());
            this.pago.setId(count + 1);//Buscar el secuencial
            List<Pago> listPago = new ArrayList<Pago>();
            listPago.add(this.pago);
            this.pedido.setPagoList(listPago);
            ksession.dispose();
        } catch (Throwable t) {     this.setMensajeMotor("El motor de reglas no esta funcionando");
t.printStackTrace();
        }
    }
```

El método **inicializarUOW()**que forma parte de la clase **CompraService** crea una sesión para administrar la persistencia de los objetos que interviene en el carro de compras.

```
public void inicializarUOW() {
        entityManagerFactory = Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
        entityManager = entityManagerFactory.createEntityManager();
        jpa = (JpaEntityManager) entityManager.getDelegate();
        serverSession = jpa.getServerSession();
        uow = serverSession.acquireUnitOfWork();
}
```

El método **generarPedido()** permite generar el pedido y persistir los objetos modificados por el motor de reglas de producción en la base de datos.

```
@Override
   public Pedido generarPedido(PedidoCompraController pedidoCompraController) {

      if (this.cliente != null) {
         this.pedido.setDetallePedidoList(this.listCarroCompra);
         uow.registerObject(this.pedido);
         System.out.println("Artículo registrado en el contexto de JPA: " + this.pedido.getId());
         for (DetallePedido dp : this.listCarroCompra) {
            Artesania a = dp.getIdArtesania();
            Artesania artesaniaClone = (Artesania) uow.registerObject(a);
            artesaniaClone.setExistencias(artesaniaFacade.buscarPorId(a.getId()).get(0).getExistencias() -
dp.getCantidad().intValue());
            System.out.println("Artículo registrado en el contexto de JPA: " + a.getDescripcion());
            uow.registerObject(dp);
            System.out.println("Artículo registrado en el contexto de JPA: " + dp.getIdArtesania().getDescripcion());
         }
         uow.registerObject(this.pago);
         System.out.println("Artículo registrado en el contexto de JPA: " + this.pago.toString());
         uow.commit();
      }
      return this.pedido;
   }
```

El metodo **readKnowledgeBase()**que forma parte de la clase **CompraService** crea una sesión a través de un agente de conocimiento a través del archivo de configuración **changeset-sas.xml**

```
private static KnowledgeBase readKnowledgeBase() throws Exception {
        final KnowledgeAgent kagent = KnowledgeAgentFactory.newKnowledgeAgent("kagent");
        ResourceFactory.getResourceChangeNotifierService().start();
        ResourceFactory.getResourceChangeScannerService().start();
        kagent.setSystemEventListener(new PrintStreamSystemEventListener());
        kagent.applyChangeSet(ResourceFactory.newClassPathResource("com/utn/drools/rules/changeset-sas.xml"));
        KnowledgeBase kbase = kagent.getKnowledgeBase();
        return kbase;
    }
```

Configuración del repositorio de datos donde se almacenan las reglas de negocio por versiones y se gestione desde Guvnor.

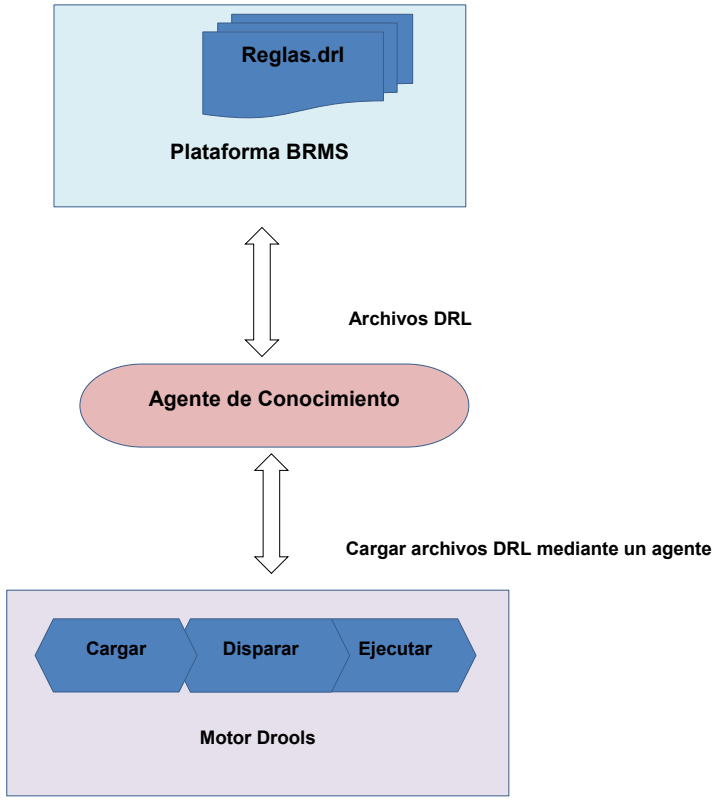

Figura 0.14: Esquema entre el motor de reglas y el BRMS
Fuente: Propia

El repositorio datos se crea en eclipse para almacenar las reglas de negocio por versiones para la tienda virtual. El repositorio se lo configura como muestra la imagen.

Figura 0.15: Configuración del repositorio
Fuente: Propia

Luego en Guvnor se crea el paquete de conocimiento **com.utn.jpa.model.entities** donde se carga las clases del modelo de hechos y las reglas de negocio, a continuación se crea el paquete que engloba a las reglas y hechos

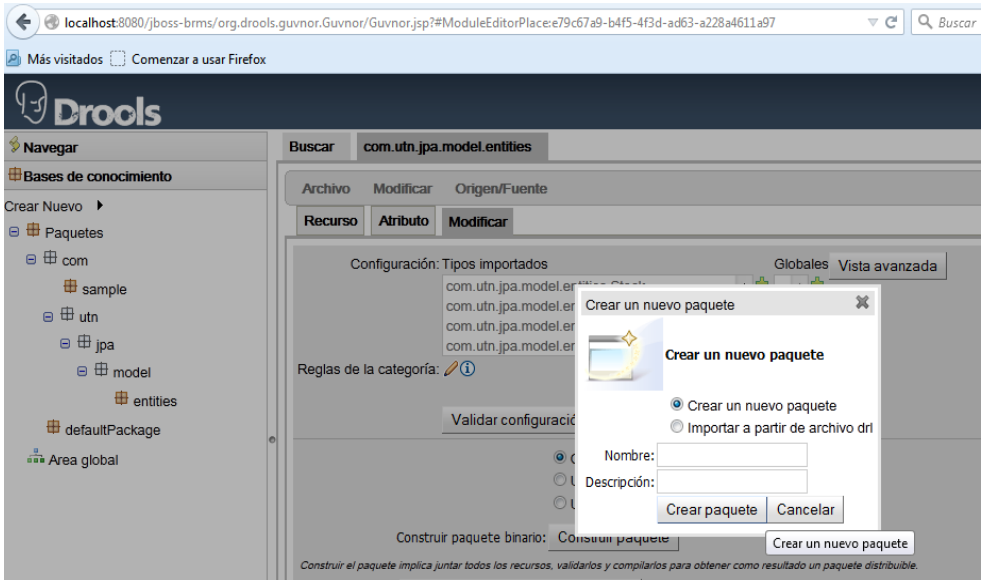

Figura 0.16: Creación de un paquete en GUVNOR
Fuente: Propia

Luego se crea el modelo de clase **ModelSanAntonioStore** como muestra.

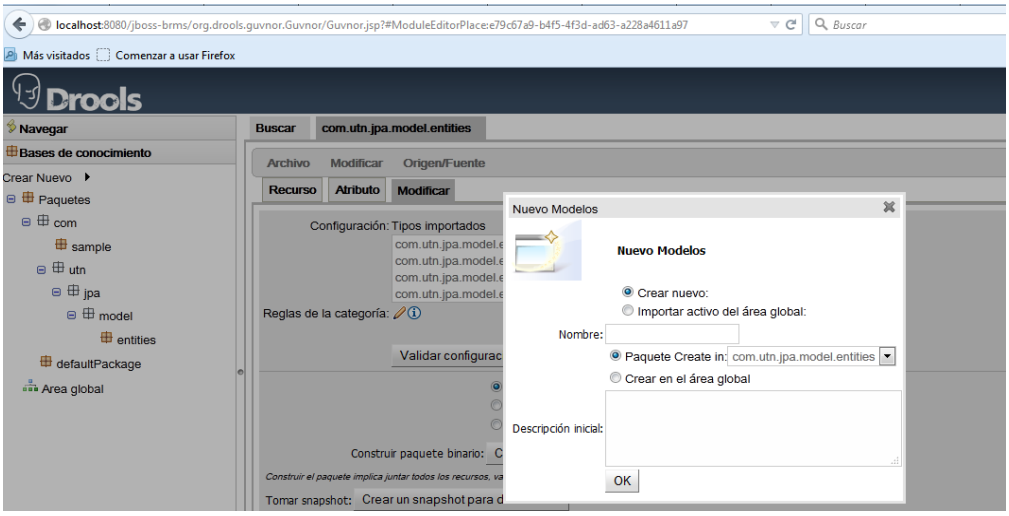

Figura 0.17: Creación de un modelo de clase en GUVNOR
Fuente: Propia

Luego se importar el modelo ModelSanAntonioStore.jar que es un archivo *.jar de las clases del modelo de la base de datos en forma de clases generadas

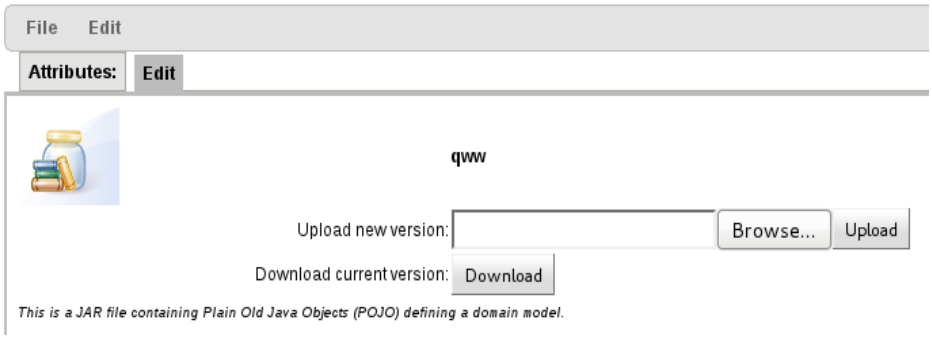

Figura 0.18: Importación del modelo de clases en GUVNOR
Fuente: Propia

Luego se crea el archivo con formato *.drl en el que se almacenara las reglas de negocio.

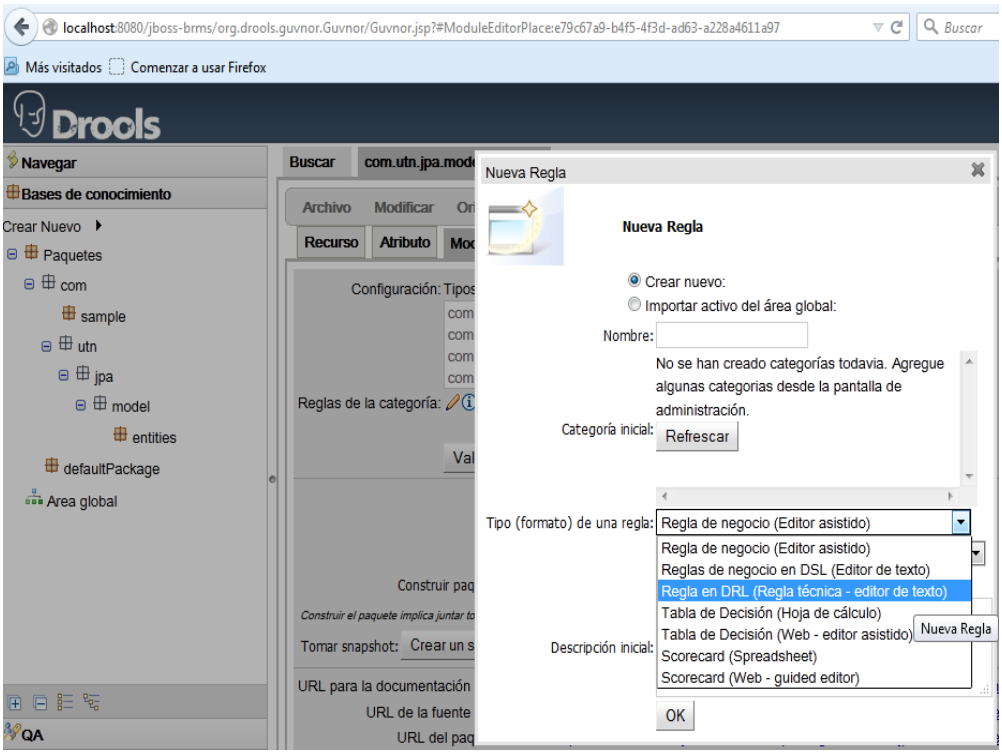

Figura 0.19: Creación de un archivo de reglas en GUVNOR
Fuente: Propia