UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS COMPUTACIONALES

TEMA: "ESTUDIO DE LA METODOLOGÍA DE REGLAS DE NEGOCIO ORIENTADO A LA UTILIZACIÓN DE UN MOTOR DE REGLAS DE PRODUCCIÓN"

AUTOR: FRANKLIN GERMÁN CHAMORRO CHUQUÍN

DIRECTOR: ING. JOSÉ LUÍS RODRÍGUEZ

IBARRA-ECUADOR

2015

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA
AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La UNIVERSIDAD TÉCNICA DEL NORTE dentro del proyecto Repositorio Digital institucional determina la necesidad de disponer los textos completos de forma digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual ponemos a disposición la siguiente investigación:

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD	1002492567
APELLIDOS Y NOMBRES	CHAMORRO CHUQUÍN FRANKLIN GERMÁN
DIRECCIÓN	San Antonio de Ibarra, Calle Luís Enrique Cevallos y Monseñor Leonidas Proaño
EMAIL	frank_tp76@hotmail.com
TELÉFONO FIJO	062551382
TELÉFONO MOVIL	0981981103

DATOS DE LA OBRA	
TÍTULO	"ESTUDIO DE LA METODOLOGÍA DE REGLAS DE NEGOCIO ORIENTADO A LA UTILIZACÍON DE UN MOTOR DE REGLAS DE PRODUCCIÓN"
AUTOR	CHAMORRO CHUQUÍN FRANKLIN GERMÁN
FECHA	
PROGRAMA	PREGRADO
TÍTULO POR EL QUE	INGENIERO EN SISTEMAS COMPUTACIONALES
DIRECTOR	ING. JOSÉ LUÍS RODRÍGUEZ

Firma: ……..
Nombre: Franklin Germán Chamorro Chuquín

Cédula: 100249256 -7

Ibarra a los 31 días del mes de Octubre del 2014

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, CHAMORRO CHUQUÍN FRANKLIN GERMÁN, con cédula de identidad Nro. 100249256-7, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y el uso del archivo digital en la biblioteca de la universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión, en concordancia con la Ley de Educación Superior Artículo 143.

……..

Firma

Nombre: CHAMORRO CHUQUÍN FRANKLIN GERMÁN

Cédula: 100249256-7

Ibarra a los 31 días de Octubre del 2014

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

CONSTANCIA

El autor manifiesta que la obra objeto de la presente autorización es original y se desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

……..

Firma:

Nombre: CHAMORRO CHUQUÍN FRANKLIN GERMÁN

Cédula: 0401219308

Ibarra a los 31 días de Octubre del 2014

UNIVERSIDAD TÉCNICA DEL NORTE

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE INVESTIGACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Franklin Germán Chamorro Chuquín, con cédula de identidad Nro. 1002492567 manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de propiedad intelectual del Ecuador, artículos 4,5 y 6 en calidad de autor del trabajo de grado denominado: "ESTUDIO DE LA METODOLOGÍA DE REGLAS DE NEGOCIO ORIENTADO A LA UTILIZACIÓN DE UN MOTOR DE REGLAS DE PRODUCCIÓN", que ha sido desarrollado para optar por el título de: **Ingeniero en Sistemas Computacionales**, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes citada.

En concordancia suscribo este documento en el momento en el que hago la entrega del trabajo final en formato impreso y digital a la biblioteca de la Universidad Técnica del Norte.

Firma:

Nombre: Franklin Germán Chamorro Chuquín

Cédula: 1002492567

Ibarra a los 31 días de Octubre del 2014

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

CERTIFICACIÓN DEL ASESOR

Certifico que la Tesis **"ESTUDIO DE LA METODOLOGÍA DE REGLAS DE NEGOCIO ORIENTADO A LA UTILIZACIÓN DE UN MOTOR DE REGLAS DE PRODUCCIÓN"**, previo a la obtención del título de Ingeniero en Sistemas Computacionales, ha sido realizado en su totalidad por el señor Chamorro Chuquín Franklin Germán portador de la cédula de identidad número: 100249256-7.

Ibarra a los 31 días de Octubre del 2014

.....

Ing. José Luís Rodríguez

Director de Proyecto

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

DEDICATOR*IA*

A mis padres, mi hermana, mi hermano y mi sobrina,

Porque nunca han dejado de creer en mí, porque siempre estuvieron

presentes en mis buenos y malos momentos dándome su apoyo

incondicional e impulsándome a salir adelante.

Mi logro es su logro.

A mí querida sobrina Alisson,

Por llenar de alegría mi vida, por ser mi fuente

de inspiración para salir adelante.

Siempre serás la luz que ilumina mi alma.

Franklin.

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

AGRADECIMIENTO

A Dios, por brindarme la oportunidad de vivir, por colmar siempre de bendiciones mi vida, por haber puesto en mi camino a mi familia y darme la oportunidad de cumplir con mis sueños.

A mis padres, mi hermana, mi hermano, y mi sobrina por la confianza, cariño y apoyo que siempre me han dado, porque han contribuido positivamente para llevar a cabo esta jornada.

Un sincero agradecimiento al Ing. José Luis Rodríguez por su tiempo y ayuda.

Franklin.

CERTIFICACIÓN

Ibarra, 31 de Octubre del 2014

Señores

UNIVERSIDAD TÉCNICA DEL NORTE

Presente

De mis consideraciones.-

Siendo auspiciantes del proyecto de tesis del egresado CHAMORRO CHUQUÍN FRANKLIN GERMÁN con CI: 100249256-7 quien desarrolló su trabajo con el tema "ESTUDIO DE LA METODOLOGÍA DE REGLAS DE NEGOCIO ORIENTADO A LA UTILIZACIÓN DE UN MOTOR DE REGLAS DE PRODUCCIÓN", con el aplicativo: "IMPLEMENTACIÓN DE UNA TIENDA VIRTUAL DE ARTESANÍAS PARA LA EMPRESA ENCHAPES ESPECIALES PRODUCTORES DE ACCESORIOS PARA DECORACIÓN DE MUEBLES" me es grato informar que se han superado con satisfacción las pruebas técnicas y la revisión de cumplimiento de los requerimientos funcionales, por lo que se recibe el proyecto como culminado y realizado por parte del egresado CHAMORRO CHUQUÍN FRANKLIN GERMÁN. Una vez que hemos recibido la capacitación y documentación respectiva, nos comprometemos a continuar utilizando el mencionado aplicativo en beneficio de nuestra empresa.

El egresado CHAMORRO CHUQUÍN FRANKLIN GERMÁN, puede hacer uso de este documento para los fines pertinentes en la Universidad Técnica del Norte.

Lic. Carlos Chamorro Flores
Gerente Propietario, Empresa Enchapes Especiales Productores de Accesorios para la Decoración de Muebles.

Ibarra a los 31 días del mes de Octubre del 2014

TABLA DE CONTENIDOS

ÍNDICE DE GRAFICOS

ÍNDICE DE TABLAS

RESUMEN

En el desarrollo de aplicaciones de software los arquitectos, ingenieros y analistas de software se enfrentan a la ardua tarea de obtener los requerimientos de usuario que son las reglas de negocio de una situación del mundo real a modelar, que luego serán traducidas en código fuente en la etapa de desarrollo de dichas aplicaciones, por tal motivo esta investigación pretende dar una pauta inicial a futuras investigaciones que conlleven a realizar mejoras en el desarrollo de software en la capa de Lógica de Negocio de la arquitectura de una aplicación de software.

La necesidad de este aspecto surge debido a que cuando las reglas de negocio que rigen a la empresa que utiliza una aplicación de software, tienen que ser modificadas por razones estratégicas de la propia empresa o por las leyes del entorno que se desenvuelve; se tiene que detener la aplicación y navegar dentro del código fuente de la aplicación para realizar los cambios requeridos.

Este trabajo se enfoca en utilizar una metodología para obtener reglas de negocio válidas para utilizarlas en un motor de reglas de producción para tratar de llegar a una solución de calidad, logrando el desacople de la aplicación en la capa de Lógica de Negocio de una arquitectura de software, en la cual no sea necesario ocultar las reglas de negocio dentro de código fuente sino tenerlas expuestas.

ABSTRACT

In the development of applications of software the architects, engineers and analysts of software confront with the arduous task of obtaining user's requisites, these are the rules of business of a situation to model, that next they will be translated in hard code in the stage of development of these applications. For such motive this investigation attempts giving an initial guideline to future investigations that bear to accomplish improvements in the development of software in Logica's layer of Business of the architecture of an application.

The need of this aspect happens owed to than when the rules of business that govern the company that uses an application of software, have to be modified by strategic reasons of the own company or for the laws of the surroundings that is unraveled; It is had to stop application and to navigate inside the hard-code of the application to accomplish the required changes.

This work focuses on using a methodology to get out rules of business been worth to use them in a motor of rules of production to try from coming to a solution of quality, achieving itself the uncouple of the application at Logica's layer of Business of an architecture of software, in which not be necessary concealing the rules of business inside hard code but to have them exposed.

CAPÍTULO I

1.1 INTRODUCCIÓN

En los inicios del desarrollo de software un aplicativo consistía en una especie de “caja negra” en la que se mantenía oculto su implementación y se mezclaban los diferentes componentes para ponerlo en funcionamiento. Conforme avanzó el tiempo empezaron a aparecer metodologías conjuntamente con herramientas para poder aplicarlas con el propósito de poder obtener un producto de mejor calidad. Se empezó a desarrollar metodologías y herramientas en las que muchas veces las personas encargadas de dicho desarrollo estaban obligadas a adaptase a estás para lograr sus metas. Y la mayoría de las veces se enfocaban en la aplicación de software en si misma pero se olvidaban del motivo por el cual se elaboraba el aplicativo de software. Las diferentes metodologías y herramientas de software empezaron a abarcar diferentes etapas del desarrollo de software; cada una de las cuales con sus respectivas normativas. Esto provocaba que los procesos y reglas de negocio con las que se regía la empresa queden escondidos en el código fuente de los diferentes lenguajes de programación que se utilizaban para el desarrollo de los sistemas de software.

Actualmente existe una variedad de metodologías y herramientas que facilitan el desarrollo de software. Metodologías que separan la información, y las aplicaciones que las manejan. Existen herramientas desarrolladas a través de motores de inferencia y sistemas de administración de reglas de producción que permiten utilizar estas reglas y tener una separación mucho más clara de dichas reglas y el sistema que las maneja.

Desatender el mayor bien que tiene una empresa como son sus reglas de negocio hace que se pierdan recursos tanto humanos como tecnológicos ya que tarde o temprano es necesario navegar en una gran cantidad de líneas de código fuente de las aplicaciones para poder modificar la lógica de la empresa.

1.2 PROBLEMA

Conocedores del problema de los desarrolladores de software que enfrentan la necesidad de modificar las reglas de negocio de una aplicación de software ocultas en líneas de código fuente cada vez que el entorno cambia donde la aplicación es utilizada. Situación que afecta a todas las aplicaciones de software que su lógica de negocio cambia con demasiada frecuencia ya sea por razones estratégicas de la propia empresa o por las leyes del entorno que se desenvuelve.

El impacto de ello es cuando se hace necesario modificar las reglas de negocio de la empresa en el aplicativo, los recursos materiales y humanos para la implementación del nuevo requerimiento no satisfacen las diferentes necesidades de los involucrados.
Una solución exitosa debería permitir modificar las reglas de negocio sin necesidad de sacar de producción al aplicativo de software.

Por lo que es necesario lograr la separación de las reglas de negocio empresariales del resto de sistema para que puedan ser rehusadas; Para poder rastrear las reglas de negocio y saber el motivo de su creación y el cambio que se ha producido en ellas.
Por lo tanto las reglas de negocio serán exteriorizadas para los actores dentro del contexto empresarial. Y esto permitirá ubicarlas para que puedan ser cambiadas en cualquier momento.

1.3 OBJETIVOS

1.3.1 Objetivo General

Implementar una Tienda Virtual de artesanías para la empresa ENCHAPES ESPECIALES productores de accesorios para decoración de muebles.

1.3.2 Objetivos Específicos

1. Documentar la metodología de reglas de negocio.
2. Investigar el funcionamiento del motor de reglas de producción DROOLS.
3. Utilizar la metodología RUP para el desarrollo del aplicativo web.
4. Desarrollar una Tienda Virtual de artesanías en tecnología JSF, para integrar el motor de reglas de producción DROOLS.
5. Validar el aplicativo web.

1.4 JUSTIFICACIÓN

En la actualidad, las empresas eligen optimizar sus recursos por medio de la implementación de sistemas que automaticen sus procesos de gestión empresarial, ya que dichos sistemas son efectivos y aportan grandes beneficios para la empresa, ya sea en tiempo, dinero y esfuerzo. Al mismo tiempo, el constante adelanto tecnológico y científico exige a las empresas adoptar nuevas formas de procesar su información y por medios más modernos que hacen que la información se traslade en el menor tiempo posible, hacia su destino.

Impulsar la utilización de motores de inferencia o sistemas de reglas de producción que manejan las reglas de negocio en su lógica empresarial, es una buena forma de disminuir la brecha tecnológica. Ya que en una empresa está el personal que domina las reglas de negocio y el personal que domina la tecnología de la información. Aplicaciones con este tipo de tecnología lo que permitirá es que las reglas de negocio tengan más significancia en el manejo de la empresa y se disminuirá la dependencia de personal capacitado tecnológicamente.

Esto permitirá que las reglas de negocio estén al descubierto y listas para ser utilizadas y modificadas sin el mayor esfuerzo más que el de dominar las reglas de negocio. Con la tecnología Web disponible en herramientas de desarrollo de software con licencia libre, se puede hacer que los sistemas sean robustos, escalables y seguros. Al utilizar herramientas con licencia libre como DROOLS Y JSF basado en el patrón de diseño de software MVC (Modelo, Vista, Controlador), se podrá realizar el aplicativo por medio de capas, lo cual permitirá realizar la integración.

La implementación del aplicativo Web, se llevará a cabo con el uso de herramientas Open Source.

Los beneficios de poner en producción una Tienda Virtual de artesanías para la empresa ENCHAPES ESPECIALES productores de accesorios para decoración de muebles están dirigidos a vender y promocionar las artesanías que se fabrica; y de la misma manera promocionar la parroquia San Antonio de Ibarra.

1.5 IMPACTOS

Impacto económico. Este trabajo, aporta grandes beneficios económicos como;

- Ahorro de Recursos.- automatización de procesos, optimizando tiempo y recursos.
- Ahorro en los Costos.- el mantenimiento de datos pueden ser realizados a costos más bajos por unidad de datos tratados.
- Aumento considerable en los recursos percibidos.- por la promoción y venta de artesanías ya que el nuevo sistema estará expuesto a un número mayor de potenciales clientes.

Impacto Tecnológico. La empresa ENCHAPES ESPECIALES productores de accesorios para decoración de muebles se beneficia adoptando una Tienda Virtual aprovechando las nuevas tecnologías con software libre usadas actualmente para aplicaciones web, con una base de datos sólida y confiable que permite gestionar la información de forma rápida y precisa, esto hará que el proceso de venta sea más rápido y confiable.

Al utilizar herramientas con licencia libre basadas en la arquitectura de diseño de software Modelo Vista Controlador (MVC), se puede realizar aplicaciones por medio de capas, lo cual permitirá que el diseño y desarrollo se en módulos integrables.

Con la tecnología Web disponibles en herramientas de desarrollo de software con licencia libre, se tendrá un sistema robusto, escalable y seguro, con las siguientes ventajas tecnológicas:

- **Parametrizable** ya que en su funcionamiento se puede personalizar todas las variables, parámetros y funciones de tal forma que se ajusten a las normativas que rigen al interior de una empresa.
- **Flexible** ya que se enfoca en la capa de negocio, las reglas comerciales que rigen en la empresa auspiciante podrán ser administradas a través de un sistema de administración de reglas.
- **Integrado a un Sistema de Administración de Reglas de Negocio** a través del repositorio de reglas de negocio Guvnor.

El sistema al estar disponible en la Web es una herramienta útil para promocionar y vender artesanías.

Impacto Educativo y Social. Este proyecto aporta un estudio introductorio a la metodología de reglas de negocio, motor de reglas de producción Drools-Expert, Guvnor BRMS (Sistemas de Administración de Reglas de Negocio).

Para la sociedad y la empresa auspiciante el sistema colaborará en la administración, promoción y venta de artesanías así como también dará a conocer la cultura de la parroquia de San Antonio de Ibarra.

1.6 BENEFICIOS

A los Artesanos.

- Promoción de las artesanías elaboradas.
- Ventas de las artesanías elaboradas.

A los Clientes

- Comprar artesanías sin necesidad de trasladarse físicamente a una tienda
- Acceder a las promociones y descuentos por su compra

A la Administración

- Gestión de Artesanías
- Gestión de Artesanos
- Gestión Pedidos
- Crear una imagen corporativa.

1.7 ALCANCE

El documento Visión contiene los módulos que forman parte de la Implementación de la Tienda Virtual de artesanías para la empresa ENCHAPES ESPECIALES productores de accesorios para decoración de muebles, así como los servicios que estarán disponibles para los clientes y usuarios.

El aplicativo Web tendrá las siguientes funcionalidades:

- Usuario y clave para el ingreso al sistema.
- Gestión de Clientes.
- Gestión de Artesanías.
- Gestión de Artesanos.
- Gestión del Carrito de Compras.
- Gestión de Pedidos.
- Gestión de las reglas de negocio a través del sistema de administración de reglas de negocio BRMS DROOLS.

Módulos del Proyecto

En este proyecto se van a desarrollar los siguientes Módulos:

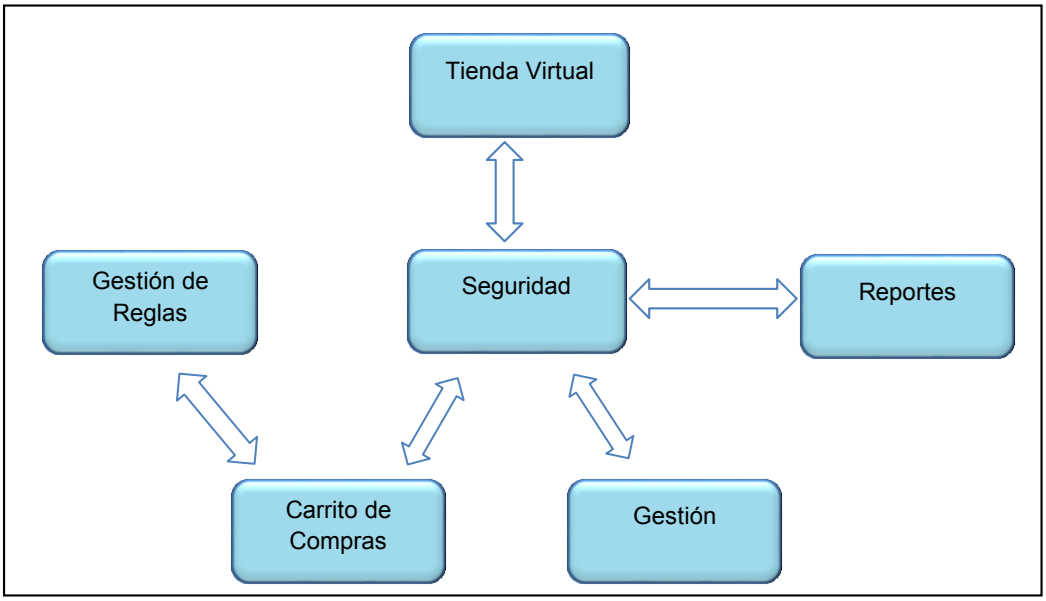


Figura 1.1: Módulos del Proyecto
Fuente: Propia

Módulo de Seguridad del Sistema

Realiza la autenticación del usuario tanto para el módulo de gestión como para el módulo de carrito de compras.

Módulo de Gestión

- Gestión de Artesanos.- En esta opción se realizan la creación, actualización, eliminación de un artesano.
- Gestión de Artesanías.- En esta opción se realizan la creación, actualización, eliminación de una artesanía.
- Gestión de Pedidos.- En esta opción se actualiza el estado del pedido

Módulo de Carrito de Compras

Permite al usuario registrado realizar compras de artesanías.

Reportes

- Reporte de Artesanos
- Reportes Artesanías
- Artesanías más compradas

1.8 HERRAMIENTAS DE DESARROLLO

La implementación del aplicativo se llevó a cabo con las siguientes herramientas:

	Herramientas	¿Por qué?
Base de Datos	PostgreSQL 9.1	Alta concurrencia, Amplia variedad de tipos nativos, Seguridad en términos generales, Integridad en BD. De libre licencia
Lenguaje de Desarrollo	JSF 2.0	Es una tecnología y framework para aplicaciones Java basadas en la web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE (Fundación Wikimedia, 2014). Permite aplicar técnicas de programación orientado a objetos De licencia libre.
Técnica de desarrollo Web	PrimeFaces 3.5	Es una librería de componentes para Java Server Faces (JSF 2) de código abierto que cuenta con un conjunto de componentes enriquecidos que facilitan la creación de las aplicaciones web.

		Está bajo la licencia de Apache License V2 (Fundación Wikimedia, 2014) De licencia libre.
Módulo de administración de reglas de negocio	Guvnor 5.5	Es un administrador de reglas de negocio para manejar reglas en un ambiente multiusuario que permite gestionarlas con una interface web. (The JBoss Drools team, Guvnor User Guide, 2014). De licencia libre.
Servidor de Aplicaciones	JBoss-AS-7.1.1	Es un servidor de aplicaciones Java EE de código abierto implementado en Java puro. (Fundación Wikimedia, 2014)
Entorno de Desarrollo Integrado	Eclipse Juno, Netbeans 7.4	Entornos de Desarrollo Integrado

Tabla 1.1: Herramientas de desarrollo.

Fuente: Propia

CAPÍTULO II

2.1 MARCO TEÓRICO

2.1.1 REGLAS DE NEGOCIO.

En el mundo de los negocios, las reglas se relacionan directamente con el comportamiento de la gente en el contexto del negocio, por lo tanto las reglas de negocio son una guía de que existe una obligación en lo referente a la administración, acción, práctica o procedimiento dentro de una actividad particular o escenario del negocio.

Una regla de negocio es cualquier conocimiento que puede ser expresado en el siguiente formato:

Cuando “algo” es verdad, Entonces haga “esto”.

Cualquier organización involucrada en los negocios tiene reglas que están detalladas en reglamentos, procedimientos, en el conocimiento de un experto del negocio o incluso en código fuente de aplicaciones de software existentes.

Para que un negocio realice decisiones válidas, esté deposita su confianza en las reglas y hechos de calidad que están disponibles para las personas que toman las decisiones y para los sistemas que se alimentan con dicha información. Estas reglas y estos hechos son conocidos como reglas comerciales.

2.1.2 METODOLOGÍA DE REGLAS DE NEGOCIO.

En una metodología comercial de reglas, las reglas son el activo más valioso para una organización comercial. De hecho, una metodología de reglas de negocio para el desarrollo de sistemas eleva la importancia de las reglas comerciales para el negocio y conlleva esa importancia en función de la organización y metodología del desarrollo de sistemas.

No hay una definición estándar de la industria para las reglas de negocio. Como guía se traduce la siguiente definición según (Barbara Von Halle, 2002, pág. 32) “Una regla de negocio es una declaración que define u obliga algún aspecto del negocio. Está dirigido a afirmar estructuras de negocio, controlar o influenciar el comportamiento del negocio.”

Una metodología de reglas de negocio es una forma formal de administrar y automatizar las reglas comerciales de una organización a fin de que el negocio se

comporte y evolucione como sus líderes pretenden, para alcanzar los objetivos del negocio según (Barbara Von Halle, 2002, pág. 12).

Una metodología de reglas de negocio aplicada en el desarrollo de sistemas computacionales permite al negocio automatizar su lógica de negocio inteligentemente, así como también implantar cambios desde adentro de sí mismo y aprender mejor y más rápido como alcanzar las metas propuestas.

En un negocio cualquiera sea su objetivo necesita sistemas de software en los cuales sus reglas sean separadas de otros componentes para que sean conocidas por todos los integrantes en el contexto del negocio, y sean fácilmente rastreables desde su creación y su implementación; por lo tanto ser fácilmente localizables para el cambio así es que todo el mundo pueda mejorarlas.

Éstos son los cuatro principios de la metodología de reglas de negocio según (Barbara Von Halle, 2002, pág. 13)

- Separar las reglas.
- Rastrear las reglas.
- Exteriorizar las reglas.
- Situar o localizar las reglas para el cambio

Diez lecciones recopilas mirando un juego preescolar de T-Ball (béisbol para niños), tomado y traducido de (Barbara Von Halle, 2002, pág. 15) para aplicarlos a una organización:

1. Las reglas de negocio son la base para el comportamiento ordenado entre todos los jugadores.
2. Las reglas de negocio influencian no sólo el comportamiento de jugadores sino que de los asistentes.
3. Las reglas de negocio enseñan e inspiran confianza; Conducen a la mayor productividad en la toma de decisiones.
4. Las reglas de negocio pueden aliviar estrés porque explican resultados.
5. Las reglas de negocio guían la libertad de elección de los jugadores.
6. Las reglas comerciales guían el comportamiento asía como los objetivos comunes por lo que se tiene mejor probabilidad de ser encontrados.
7. Las reglas de negocio son el más conveniente soporte cuando participan en una manera coherente entre todos los jugadores pertinentes.

8. Las reglas de negocio pueden motivar o desmotivar a los jugadores.
9. Las reglas de negocio determinan la probabilidad de lograr metas comunes.
10. Las reglas de negocio son mecanismos por los cuales una organización se cambia a sí mismo.

Por lo tanto la pregunta que surge es: ¿Qué Es Una Metodología de Reglas de Negocio?

Según (Barbara Von Halle, 2002, pág. 15) se toma y traduce. "Una metodología de reglas de negocio es una metodología y posiblemente una tecnología especial por el cual se captura, cuestiona, publica, automatiza, y cambia reglas de una perspectiva comercial estratégica. El resultado es un sistema de reglas de negocio, un sistema automatizado en el cual las reglas son separadas, lógicamente y quizá físicamente, de otros aspectos del sistema y compartidos a través de almacenes de datos, interfaces del usuario, y quizá las aplicaciones".

Por lo que una metodología comercial de reglas es un conjunto de fases, pasos, técnicas y un conjunto de directrices para proporcionar sistemas de reglas de negocio.

2.1.3 MOTIVACIÓN PARA LAS REGLAS DE NEGOCIO.

En un negocio se encuentran todo tipo de reglas de negocio que satisfacen requerimientos a los cuales se les debe dar cumplimiento y son emitidas por Instituciones reguladoras, unidades estratégicas de negocio (UEN: Es una unidad operativa, que agrupa productos o servicios diferenciados, vendidos a un conjunto definido de clientes y que al mismo tiempo enfrentan un grupo determinado de competidores.) (Hax Arnoldo C, 2005), compradores, competidores y en general las condiciones de mercado, todos generan reglas de negocio en un cambio constante.

El conocimiento y la experiencia de un negocio cualesquiera sean sus objetivos se convierten en reglas de negocio esto implica depender de la presencia del experto para efectuar las operaciones del negocio; lo que provoca procesos costosos para realizar estas operaciones; operaciones que pueden automatizarse y así el experto disponer de más tiempo para estudiar mejoras en los procesos para el negocio.

Los negocios están expuestos a dificultades y presiones que en muchas ocasiones son insuperables y causan la quiebra del mismo; por lo que el negocio requiere cambios en la forma que el negocio funciona a través de sus sistemas automatizados. Aún más

crítico es la relación con los clientes, los competidores, o los requisitos legislativos que imponen muchas de estas presiones.

Hay muchos panoramas en que el negocio puede ser mejorado aplicando una metodología de reglas de negocio:

Uno de tantos panoramas son los sistemas automatizados existentes en el negocio que son el principal obstáculo para cambiar dicho negocio ya que no existe documentación correcta y la lógica del negocio está sepultada en el código fuente lo que hace muy costoso su mejoramiento.

Otro panorama implica las nuevas reglas, ordenanzas o los mandatos legislativos que requieren ser incorporados rápidamente al contexto comercial, por lo que se abre las puertas para nuevas oportunidades comerciales y necesariamente se requiere cambios en los sistemas existentes. Las nuevas oportunidades comerciales exigen cambios en los sistemas existentes o promueve la construcción de sistemas nuevos.

Por otro lado el nuevo mercado que promueve la Internet hace que surjan nuevos productos y servicios.

Entonces la competencia entre negocios con productos y servicios similares surge amenazadora cuando la presencia es a través de la web o solo en un lugar físico.

Y sobre todo cuando un negocio adquiere rentabilidad se produce la necesidad para consolidar bases de información, de clientes, compras, provocando la evaluación de políticas existentes, prácticas, procesos, y las reglas.

Según (Barbara Von Halle, 2002, pág. 16) se toma y traduce. "La reingeniería de procesos de negocio es continúa. Algunos de estos esfuerzos algunas veces tienen como meta crear una perspectiva global coherente para un proceso dado".

Desde el contexto del negocio hay deficiencias en la mayoría de metodologías y tecnologías con respecto a cómo se tratan a las reglas de negocio ya que al final dichas reglas quedan sepultadas dentro del código fuente de la aplicación.

Los beneficios de una metodología de reglas de negocio en el desarrollo de sistemas informáticos según (Barbara Von Halle, 2002, págs. 20-21) se toma y traduce como la metodología comercial de reglas es expresada en cuatro principios que servirán de base para este estudio:

"(S) Separar. Esto quiere decir que se separa las reglas de todos los demás aspectos de los requisitos y en el sistema mismo. Se hace esto primordialmente para rehusar reglas.

(T) Rastrear. Esto quiere decir que se mantiene una conexión de cada regla en dos direcciones. La primera dirección está hacia sus orígenes. Una regla tiene orígenes en los aspectos de la motivación del negocio, como misiones comerciales, las metas, los

objetivos, las estrategias, los métodos, y las políticas. La segunda dirección a rastrear es la implementación de la regla. Se puede evaluar el impacto de los cambios de la regla.

(E) Exteriorizar. Esto quiere decir que se expresa una regla en un formato comprensible para las audiencias poco técnicas de negocio, y que se hace disponible la regla a estas audiencias.

(P) Posicionar. Esto quiere decir que siempre se sitúa una regla para el cambio precisamente porque se espera que las reglas cambien como un curso normal de negociar. Se hace esto entonces para producir cambios en las reglas fácil y rápidamente. Posicionar una regla para el cambio puede significar implementarlo en una tecnología que tiene previsto cambios fáciles, como un producto comercial de reglas".

A partir de estos principios se puede ver que uno de los problemas principales que se presenta en el desarrollo de sistemas con metodologías tradicionales es que la lógica del negocio o sea sus reglas quedan sumergidas en el código fuente del sistema, lo que provoca un consumo de tiempo y dinero en modificar las reglas para que se adapten a nuevos desafíos.

Al tener separadas las reglas comerciales de los demás componentes del sistema se puede aplicar técnicas para procesar reglas eficientemente.

De esta manera el negocio requiere poder rastrear la efectividad de las reglas desde su origen y determinar con el paso del tiempo si la regla es correcta y realizar su aplicación en la organización.

Al poder ser expresadas las reglas de negocio en un formato entendible para los expertos del negocio, permite que sean accesibles a las personar para que puedan ser entendidas, optimizadas, cuestionadas, inspeccionadas, medidas de acuerdo a lo que persigue el negocio.

Aun sin usar un producto comercial de reglas, saber dónde se encuentra una regla para analizar el impacto de un cambio de dicha regla en los eventos comerciales ahorra tiempo en la toma de decisiones para la organización lo que le da una oportunidad más frente a sus competidores que tienen su lógica de negocio oculta en el código de sus aplicaciones.

La diferencia más significativa en una metodología de regla de negocio frente a otra metodología está en los cuatro principios según (Barbara Von Halle, 2002, págs. 20-21). Como un recordatorio, las diferencias son que una metodología comercial de regla simplemente apunta a separar, rastrear, exteriorizar, y ubicar o localizar las reglas de los demás componentes del sistema.

2.1.4 CLASIFICACIÓN DE LAS REGLAS DE NEGOCIO.

Hay un sinnúmero de propuestas sobre la clasificación de las reglas de negocio, pero desafortunadamente, no hay esquema comercial universal de clasificación de regla.

En la tabla siguiente según (Barbara Von Halle, 2002, págs. 33-35) se toma y traduce para poder ver una clasificación de reglas de negocio según el esquema comercial.

RECURSO	Esquema comercial de clasificación de regla
Business Rules Group (2000)	Derivación: Una declaración de conocimiento que se derivó de otro conocimiento en el negocio. Computo matemático. Inferencia. La afirmación estructural: Un concepto definido o una declaración de un hecho que expresa algún aspecto de la estructura de la empresa. Esto abarca ambos términos y hechos ensamblados de estos términos. Términos. Hechos. La afirmación de acción: Una declaración de una restricción o una condición que limita o controla las acciones de la empresa. Autorización. Condición. Restricción de integridad.
Ross (2001)	• Hechos. • Términos. • Reglas. • Restricciones. • Derivaciones. • Inferencias. • Oportunidad del momento. • Secuencia. • Heurísticas.

General Data Analysis Rule Types	Atributo de reglas. • Unicidad. • Verificación de valor. Cálculos. Inferencias. Restricciones de atributo de multi-entidad. Reglas de relación. • Cardinalidad. • Integridad de referencia. • Cuentas de cardinalidad.
C. J. Date (2000)	• Restricción. • Restricción de estado. • Restricción de transición. • Estímulo/respuesta. • Derivación. • Cómputo. • Inferencia.
C. J. Date (2000)	Chris Date además propone otro esquema para las restricciones que se basa en la estructura de los datos. • Restricciones de dominio. • Restricciones de columna. • Restricciones de tabla (restricciones dentro de una tabla). • Restricciones de la base de datos (restricciones entre dos o más tablas).
Versata Inc.	• Reglas de integridad referencial • Derivaciones (cómputos de atributo) • Validación(valores de atributos obligatorio/ opcionales, min, max) • Restricción (¿El atributo para las restricciones de atributo dentro de una entidad?). • Acción/evento. • Reglas de presentación.

USoft Inc. (Mallens [1997])	• Reglas de restricción: Las restricciones comerciales en la información a ser almacenadas, que no es permitido. • Reglas de comportamiento: Cómo debe el sistema comportarse en situaciones dadas, Lo que el sistema debería hacer automáticamente. • Reglas deductivas: Cómo debería estar la información derivada o calculada. • Reglas de presentación: Cómo el sistema se presenta al usuario, cómo debe el trabajo y las tareas ser organizado. • Reglas de instrucción: Cómo el usuario debe operar el sistema en ciertas situaciones.

Tabla 2.1: Clasificación de Reglas de Negocio.
Fuente: Traducción de **(Barbara Von Halle, 2002, págs. 33-35)**

La clasificación que se usará en esta documentación se enfoca específicamente para el descubrimiento de reglas y su validación en el contexto comercial, lo que implica utilizar los 4 principios mencionados anteriormente.

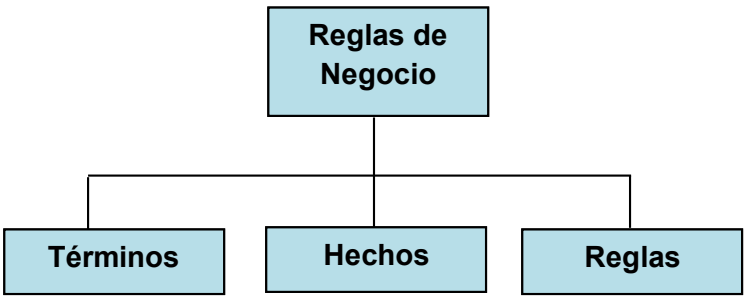


Figura 2.1: Un esquema comercial de clasificación de regla de alto nivel.
Fuente: Traducción de **(Barbara Von Halle, 2002, pág. 36)**

2.1.4.1 Los términos

Un término es un sustantivo o una frase sustantiva (llamado también nombre que es un tipo de palabra que al combinarse con otras, forma frases sustantivas. Es decir es un conjunto de palabras que se agrupan alrededor de un sustantivo o de una palabra que equivale a un sustantivo) (Karen Coral Rodríguez, 2004) con una descripción convenida. Un término puede definir cualesquiera de lo siguiente:

- Un concepto, como por ejemplo: vendedor, cliente, factura.
- Una propiedad de un concepto, como código de evaluación de crédito de cliente.
- Un valor, como Femenino/Masculino.
- Una colección de valores, como días de la semana, o meses del año.

2.1.4.2 Los hechos

Un hecho es una declaración que asocia términos, a través de preposiciones (Partícula invariable que une dos palabras estableciendo una relación de dependencia entre ellas. (LAROUSSE, 2001)), Frases verbales (se construyen combinando un verbo con diferentes tipos de frases preposicionales modificadores) (Karen Coral Rodríguez, Manual de Gramática del Castellano, 2004), en observaciones apreciables, pertinentes al negocio. Los ejemplos de hechos son:

- El cliente puede colocar órdenes.
- La orden es por línea ítem.
- La línea ítem es por producto.
- El cliente califica por el código de evaluación de crédito de cliente.

Los términos y los hechos son la semántica detrás de las reglas. También se convertirán en la base para un modelo lógico de datos y la base de datos física y quizá un modelo comercial de objetos.

2.1.4.3 Las reglas

Las reglas, es la base de una metodología de reglas de negocio. Una regla es una afirmación declarativa que aplica la lógica o cómputo a los valores de información. Una regla resulta ya sea en el descubrimiento de información nueva o una decisión acerca de tomar acciones. Los ejemplos de reglas son:

- Un cliente nuevo no puede hacer un pedido que la orden excede $1000.
- El embarque de una orden para un cliente existente que no ha pagado su última factura, será retrasado hasta que ese pago sea recibido.

Entonces, una regla resultará ser lógica ejecutable que usa información como la introducción de datos y crea como salida, ya sea información o acción.

A medida que un evento comercial o un caso de uso sucede, una persona empresarial puede inyectar reglas para controlar su ejecución y al menos hay cuatro formas posibles que una regla puede guiar un evento comercial. Específicamente, una regla puede hacer uno de los siguientes puntos en el contexto del evento comercial:

- Presenta información acerca del evento comercial.
- Restringe información creada por el evento comercial.

- Da inicio a una acción fuera del límite del sistema de objetivo o el evento comercial.
- Crea información nueva a partir de información existente.

Estas clasificaciones indican lo que la regla intenta hacer desde el punto de vista de una persona comercial. La clasificación de regla que se seguirá en esta documentación se describe a continuación. Tanto los cálculos o cómputo y las inferencias crean conocimiento mientras las restricciones restringen o guían comportamiento dando mayor significado comercial. Tanto los cálculos, las inferencias, y las restricciones obligatorias son las clases de reglas más frecuentemente soportadas en productos comerciales de reglas.

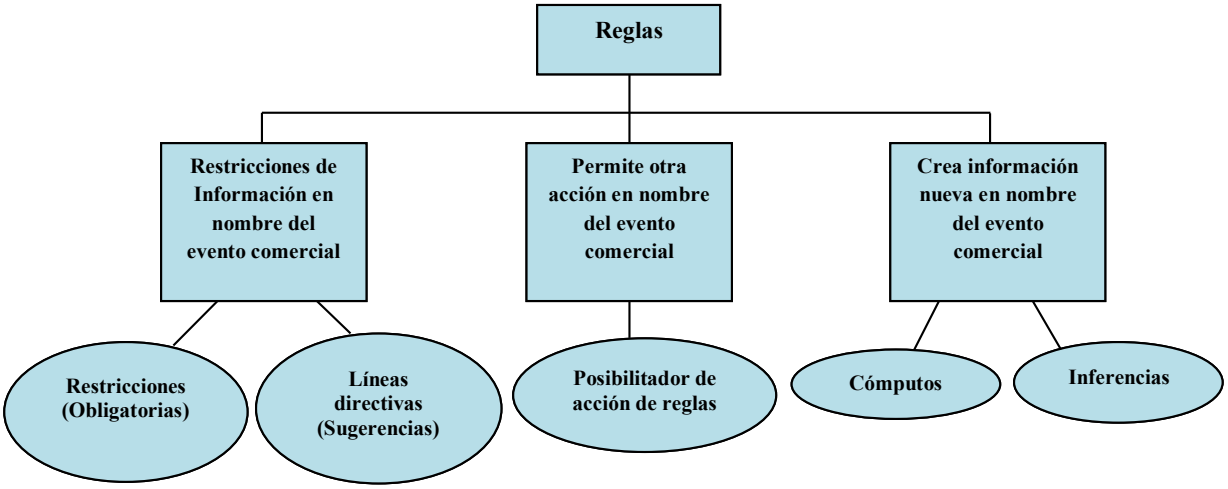


Figura 2.2: Esquema de Clasificación de reglas.
Fuente: Traducción de **(Barbara Von Halle, 2002, pág. 36)**

2.1.4.4 Restricciones

Una restricción puede ser una restricción obligatoria o una restricción propuesta en el comportamiento del evento comercial. Una restricción obligatoria es una declaración completa que expresa una circunstancia incondicional que debe ser verdadera o no verdadera para el evento comercial para completar con la integridad. Los ejemplos de restricciones obligatorias son:

- Un cliente no debe tener más que 10 órdenes abiertos a la vez.
- La cantidad total de dólares de una orden del cliente no debe ser mayor que la cantidad individual del límite de crédito de la orden del cliente.

2.1.4.5 Las líneas directivas

Una línea directiva es una declaración completa que expresa una advertencia acerca de una circunstancia que debería ser verdadera o no verdadera. Una línea directiva no fuerza la circunstancia a ser verdadera o no verdadera, pero meramente advierte acerca de ella, dejando al humano tomar la decisión. Porque una línea directiva sólo da aviso y no deniega, provee una libertad de elección. Un ejemplo de una línea directiva es:

- Un cliente no debería tener más que 10 órdenes abiertos a la vez.
- Los Posibilitadores de Acción.

Un posibilitador de acción es una declaración completa que prueba condiciones y al encontrarlos verdadero, inicia otro evento comercial, mensaje, u otra actividad. Es decir, un posibilitador de acción inicia una nueva acción externa al alcance del sistema o el incremento bajo el estudio u otra actividad.

Los ejemplos de Posibilitadores de acción son:

- Si una orden del cliente es válida, luego inicie el proceso de entrega de la Orden.
- Si un cliente es de alto riesgo, entonces notifique al administrador de servicios de pos-venta.

Las reglas de posibilitador de acción pueden ser usadas en algunos productos comerciales de reglas para crear una secuencia orientada en eventos de pasos de flujo de trabajo. Puede ser de ayuda para pensar acerca de restricciones obligatorias y poner en marcha a los posibilitadores como los opuestos. Las restricciones obligatorias detienen un evento a completar. Los posibilitadores de acción inician un evento.

2.1.4.6 Cálculos o cómputo

Un cálculo es una declaración completa que provee de un algoritmo para lograr el valor de un término donde tales algoritmos pueden incluir suma, diferencia, producto, cociente, conteo, mínimos, máximos, promedios.

Un ejemplo de una regla de cálculo es:

- El total de importe a pagar por una orden es la suma de la cantidad línea-Ítem (s) para la orden y el impuesto.
- El resultado de ejecutar una regla de cálculo es crear un trozo de información nuevo.

El resultado de ejecutar una regla de cómputo es crear un trozo de información nuevo. Se usa el conocimiento de término para querer decir una información creada por una regla. Es decir, la información no es simplemente conocida mirándolo o leyéndolo, necesita ser creada según una regla de algún tipo. En este caso, el trozo de información

nuevo es un valor nuevo para un atributo, como cantidad de dólar de orden de cliente total. Más tarde, el modelo lógico de datos, quizá el diseño de la base de datos y el modelo del objeto, pueden reflejar estos trozos nuevos de información.

2.1.4.7 Las inferencias

Una inferencia es una declaración completa que prueba condiciones y al encontrarlos verdadero, establece la verdad de un hecho nuevo.

Los ejemplos de inferencias son:

- Si un cliente no tiene facturas pendientes de pago, entonces el cliente es de estatus preferido.
- Si un cliente es de estatus preferido, entonces la orden del cliente califica para un 20 por ciento de descuento.

El resultado de ejecutar una regla de inferencia es crear un trozo de información nuevo, por consiguiente conocimiento.

Usando la definición de reglas comerciales, todas las reglas comerciales se tratan de datos.

Es decir, los términos definen conceptos de datos y circunstancias, los hechos definen relaciones entre datos, las restricciones y las líneas directivas prueban valores de datos, los cómputos llegan a un valor de datos, las inferencias llegan a una conclusión de datos, y los posibilitadores de acción evalúan valores de datos antes de iniciar acción.

2.1.5 VISIÓN GENERAL DE LA METODOLOGÍA COMERCIAL DE REGLAS

A continuación se presenta un resumen y traducción en forma de pasos de la metodología de reglas de negocio según (Barbara Von Halle, 2002, págs. 64-67).

2.1.5.1 Paso 1: Alcance

El siguiente es un conjunto de pasos para la fase del alcance. Esto indica que no hay una diferencia significativa en el alcance de una metodología de reglas de negocio con respecto a otras metodologías de desarrollo de software. Sin embargo en el paso 3 la metodología de reglas de negocio hace énfasis en descubrir el contexto completo comercial en el cual se revelará y medirá el valor de las reglas:

1. Hacer investigación inicial del negocio o empresa.
2. Desarrollar declaraciones iniciales del alcance.
3. Investigar el contexto comercial completo.
4. Identificar eventos comerciales.

5. Identificar a los stakeholders.
6. Identificar situaciones.
7. Identificar procesos de respuesta de eventos.
8. Identificar métricas comerciales de funcionamiento.
9. Identificar sujetos de datos.
10. Identificar requisitos adicionales.
11. Identificar restricciones comerciales.
12. Identificar restricciones técnicas.
13. Identificar riesgos comerciales y técnicos.
14. Priorizar requisitos comerciales.
15. Determinar alternativas arquitectónicas.
16. Seleccionar una solución arquitectónica.
17. Crear diagrama de alcance.
18. Estimar infraestructura organizativa y recursos requeridos.
19. Crear un documento del proyecto.
20. Realizar el compromiso de logro.

El alcance es subdivido por dos conjuntos de pasos para descubrir requisitos en la etapa del alcance. Un conjunto es:

- Para descubrir requisitos iníciales.
- Para descubrir reglas y datos conjuntamente.

Los pasos para descubrir requisitos iníciales incluyen:

- Crear descripciones de casos de uso.
- Agregar panoramas bien establecidos.
- Identificar decisiones.
- Completar el Modelo Conceptual.
- Los pasos para descubrir reglas y los datos incluyen:
- Identificar fuentes de reglas.
- Seleccionar un conjunto de directrices para el descubrimiento de reglas.
- Seleccionar o confirmar estándares de reglas.
- Planificar el tiempo de descubrimiento de reglas y el propósito.
- Descubrir reglas a través del conjunto de directrices.
- Autenticar las reglas.
- Dar a las reglas un valor comercial.
- Definir términos.

- Definir hechos.
- Empezar un modelo de términos/hechos.
- Agregar escenarios concretos.

Notar que los pasos para descubrir requisitos iniciales contienen sólo un paso nuevo para proporcionar reglas y ese es un paso para revelar decisiones detrás de eventos comerciales.

Las decisiones son un framework (Marco de Trabajo) para organizar reglas detalladas. También notar que la mayor parte de los pasos para descubrir reglas y datos conjuntamente son nuevos o modificados para adecuar a una metodología de reglas de negocio.

2.1.5.2 Paso 2: Análisis

Hay tres conjuntos separados de pasos para analizar reglas, datos y procesos. La razón para esto es que hay técnicas diferentes de análisis y entregables diferentes para cada paso; pero relacionados.

Notar que todos los pasos para analizar reglas son nuevos. Algunos de los pasos para analizar datos son nuevos o revisados porque hay superposición entre comprender restricciones de integridad de datos y revelar restricciones más complicadas. Por consiguiente, se necesita coordinar bien las reuniones de reglas comerciales tradicionales acerca de los datos con las reuniones de reglas más complicadas acerca de los procesos comerciales usando los datos. Muchos de los pasos para analizar procesos están revisados tomando una metodología de reglas de negocio. Eso es porque se quiere separar los detalles de la ejecución de reglas del flujo de proceso básico. El flujo de ejecución de reglas también puede influenciar el flujo global de proceso.

Los pasos para analizar reglas:

1. Hacer cada regla atómica.
2. Comprender los patrones subyacentes de regla.
3. Remover reglas redundantes.
4. Resolver superposiciones entre las reglas.
5. Resolver incongruencias entre las reglas.
6. Garantizar la integridad entre las reglas.
7. Identificar dependencias entre las reglas.
8. Refinar el proceso basado en dependencias de las familias de reglas o las actividades de datos.

9. Optimizar las reglas para el negocio.

Los pasos para analizar datos:

1. Identificar las entidades candidatas.
2. Determinar relaciones entre entidades.
3. Identificar llaves primarias y alternas.
4. Propagar claves foráneas.
5. Determinar reglas comerciales extremadamente importantes.
6. Agregar atributos.
7. Normalizar atributos.
8. Analizar relaciones.
9. Determinar reglas detalladamente.
10. Combinar con modelos lógicos relacionados.
11. Integrar modelos de datos con perspectivas comerciales más amplias.
12. Anticipar el futuro en el modelo.
13. Identificar a las entidades creadas en las reglas.
14. Identificar relaciones creadas en las reglas.
15. Identificar atributos creados en las reglas.
16. Correlacionar reglas con el modelo lógico enriquecido en regla de datos.

Los pasos para analizar procesos:

1. Descubrir y expandir un flujo básico preliminar del proceso.
2. Asignar decisiones y reglas a la capacidad de reglas.
3. Confirmar la condición del flujo básico de proceso.
4. Considerar alternativas para el flujo básico de proceso.
5. Crear un diagrama simple de flujo de trabajo para mostrar a la máxima concurrencia para el flujo básico del proceso.
6. Finalizar el flujo básico preferido de proceso.
7. Crear un diagrama simple de flujo de trabajo para el flujo básico de proceso.
8. Revisar diagramas básicos de flujo de trabajo usando escenarios bien establecidos.
9. Establecer referencias para contexto comercial.
10. Confirmar los diagramas básicos de flujo de trabajo.
11. Estudiar importantes transiciones de estado.
12. Complementar todos los pasos.
13. Crear otro entregable del análisis de procesos para el flujo básico del proceso

14. Crear un diagrama de flujo de trabajo para el flujo de reglas, si es necesario.

2.1.5.3 Paso 3: Diseño

El siguiente es un conjunto de pasos para diseñar la automatización de reglas y todos estos pasos son nuevos. También contiene pasos para diseñar bases de datos relacionales de alta calidad importante para dar sistemas cambiables.

Los pasos para diseñar reglas incluyen:

1. Confirmar una arquitectura, pero agudizarla con una capacidad de reglas.
2. Determinar los requisitos básicos para la capacidad de reglas.
3. Determinar si se adquirirá un producto comercial de reglas.
4. Si no, determinar si se desarrollará una capacidad de reglas.
5. Determinar en cuál capa se implementará las reglas.
6. Diseñar las reglas en un producto comercial de reglas orientado a datos.
7. Diseñar las reglas en una capacidad de reglas de cosecha propia orientada a datos.
8. Diseñar las reglas en un producto comercial de reglas orientado a servicios.
9. Diseñar las reglas en una capacidad de reglas de cosecha propia orientada a servicios.
10. Afinar la base de datos.
11. Afinar las reglas.
12. Afinar las reglas alterando la ejecución de reglas a otra capa.
13. Afinar las reglas duplicando la ejecución de reglas.
14. Afinar las reglas cambiando plantillas de reglas.
15. Diseñar el resto de sistema.

Los pasos para diseñar la base de datos relacional:

1. Determinar tablas.
2. Determinar columnas.
3. Adaptar la estructura de almacenamiento al ambiente del producto.
4. Diseñar las reglas.
5. Analizar eventos comerciales proporcionados por la base de datos.
6. Considerar rutas de acceso para los eventos comerciales.
7. Afinar las opciones invisibles.
8. Afinar las estructuras visibles de almacenamiento.

2.1.6 LAS PISTAS DE LA METODOLOGÍA DE REGLAS DE NEGOIO

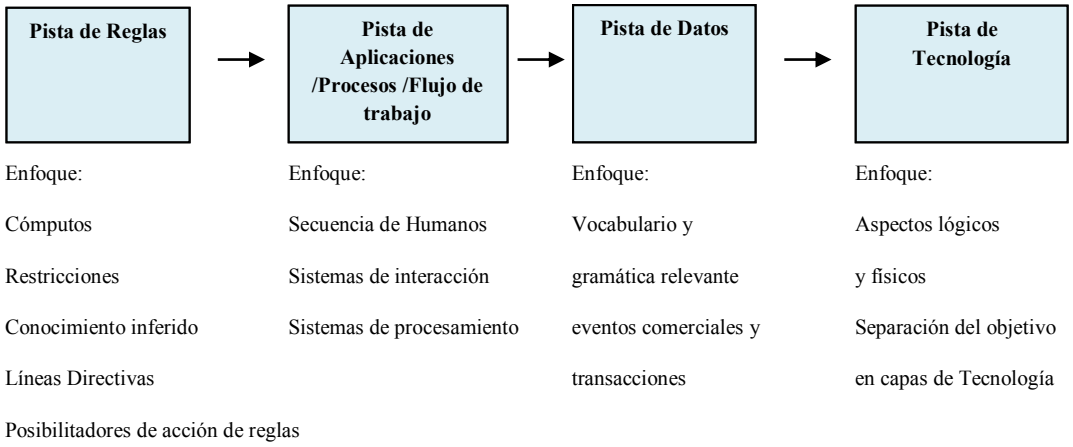


Figura 2.3: Pistas de la metodología de reglas de negocio
Fuente: Traducción de (Barbara Von Halle, 2002, pág. 68)

En la figura anterior se puede mirar las cuatro pistas o rastros de la metodología comercial de reglas.

A continuación se resume y traduce del libro de (Barbara Von Halle, 2002, págs. 68-70) las pistas o rastros que se debe tener presente en cada fase de la metodología.

2.1.6.1 La pista de tecnología

La pista de tecnología incluye pasos para seleccionar, adecuar, y dar soporte de tecnología a fin de apoyar la metodología comercial de reglas. Mientras que se puede utilizar cualquier tecnologías para implementar reglas comerciales incluyendo cualquier lenguaje de programación o archivos planos. La pista de tecnología centra su atención en tecnologías en reglas que habilitan la entrega más rápida y proporciona realizar los cambios más fácilmente en los sistemas comerciales de reglas.

En la mayoría de los casos, se supone que se utilizará una Internet o una Web Front-End, un DBMS relacional, tecnología comercial orientada en reglas, y Herramientas Case o Tecnología de repositorio.

2.1.6.2 La Pista de Proceso

La pista de proceso se enfoca en entender las interacciones del actor, las secuencias de interacciones entre los humanos y el sistema, así como también las interacciones entre el sistema y otras piezas automatizadas, como bases de datos u otros sistemas. Por consiguiente, como se sigue los pasos en la pista de proceso, se entregará una lista de todos los eventos comerciales despachados por el sistema objetivo, el proceso

comercial asociado del evento comercial, y finalmente para las decisiones que se hacen a cuenta del proceso en dar servicio al evento comercial. Tan pronto como se efectúa una transición de comprender el proceso del evento para revelar las decisiones detrás del evento, se efectúa una transición de la pista de proceso en la pista de regla. Entonces se puede imaginar que la transición de eventos comerciales para los procesos para las decisiones para las reglas es lo natural.

2.1.6.3 La Pista de Datos

La pista de datos produce los modelos de datos y base de datos eventualmente. Cuando se sigue los pasos en la pista de datos, se revela el vocabulario y gramática con la cual las reglas son expresadas y se crea una estructura estable de datos para representar ese vocabulario. Específicamente, como se revelan reglas, se asegura que cada palabra o frase es representada en el modelo de datos y base de datos. Entonces, se puede entender que la transición de la colección de reglas para las construcciones de datos es asimismo también muy natural.

Después de los pasos en la pista de datos, se organiza los términos y los hechos en una estructura estable de datos, usando análisis arquitectónico de datos y principios de diseño. Pero se da una estructura de datos con reglas mínimas. De aquí, los pasos conducen a analizar la estructura de datos para la estabilidad de largo plazo haciendo frente al cambio comercial. Por la importancia de la base de datos, la pista de datos contiene pasos para desarrollar una base de datos tan estable como sea posible, y tan anticipado como sea posible. La meta es identificar incrementos implementables de datos que permiten el crecimiento estructural de la base de datos con el paso del tiempo con interrupción mínima para el negocio. La habilidad para que la estructura de datos soporte el cambio comercial y direcciones futuras es lo suficientemente importante para intercambiar opiniones más adelante.

Los negocios hoy necesitan cambiar rápidamente, ya sea reactivamente o proactivamente. Acomodar tales cambios en sistemas existentes es usualmente muy consumidor de tiempo y caro. Históricamente, los cambios de sistemas más caros han sido aquellos que requieren un cambio en la lógica del sistema (como, en una regla de cómputo, en una regla de inferencia, o en una regla de restricción, por ejemplo) o requiera un cambio en la estructura de datos (como, los atributos de gestión, agregar atributos, cambiar llaves, agregar tablas).

Considere primero los cambios en la lógica del sistema. En el pasado, hacer cambios a la lógica de un sistema fue una tarea consumidora de tiempo porque un programador debió encontrar todo código procesal pertinente (pudo estar en muchos lugares),

cambiarlo, y probarlo. Las reglas sin embargo, situadas para el cambio a través de la tecnología de regla permiten cambios dinámicos a la lógica centralmente manejada como reglas declarativas. Por consiguiente, cuando se expresa el sistema lógico como reglas declarativas, los cambios para esas reglas requieren significativamente menos tiempo, por consiguiente cuestan menos dinero, y tienen un impacto positivo más rápido en el negocio mismo.

Ahora se considera cambios en la estructura de datos. Esto puede seguir siendo muy consumidor de tiempo y caro, aun al usar tecnología comercial de reglas. Primero, que el cambio de datos tiene que ser analizado y diseñado. Cambiar los datos estructurales principales puede requerir que los datos se bajen y se vuelvan a cargar. Además, la lógica correspondiente de sistema también puede necesitar cambiar.

Si se expresa mucho del sistema lógico como reglas declarativas, estos cambios pueden requerir menos tiempo que de la otra manera. Sin embargo, los cambios estructurales de datos son todavía tan consumidores de tiempo y caros como siempre. Por consiguiente, los cambios estructurales de datos se convierten en los cambios más significativos, causando la mayor parte de interrupción.

Es suficiente esto para decir que el desarrollo de sistemas de regla de negocio, la economía cambiaría.

Si la regla cambia es fácil la modificación; entonces los cambios de datos pueden emerger como la barrera más significativa para el cambio comercial, si el diseño de la base de datos no ocurre correctamente.

La pista de datos no solo resalta la importancia del modelo subyacente de datos, pero provee técnicas para asegurar que el modelo inicial de datos permanece estable con el paso del tiempo. La pista de datos se dirige a esto comprendiendo el alcance completo de datos desde el principio y por ahí identificando esos aspectos de su estructura de datos que se necesita diseñar con el futuro en mente. Tiene poco sentido situar las reglas para el cambio y luego la base de datos permanezca como una barrera significativa para ese cambio.

2.1.6.4 La Pista de Regla

La pista de reglas representa el conjunto de reglas detrás de las interacciones y encima de los datos, donde las reglas son manejadas como un componente lógico, se separan del flujo básico de proceso y se separan de los datos. La pista de reglas enfoca la atención en capturar, analizar, automatizar y cambiar las reglas del negocio. Estas reglas son la lógica básica que guía las decisiones y las acciones dentro de la organización comercial y ahora dentro de un sistema de reglas de negocio. Elaborando

las reglas del negocio como un entregable, las personas comerciales mantienen el control de desarrollo de sistemas, gobernando el comportamiento de sus humanos y sus sistemas. A través de la especificación de estas reglas, los expertos comerciales predeterminan las respuestas para las situaciones estándar, adelantadas y comerciales que pueden levantarse dentro de un evento comercial. Al mismo tiempo, los expertos comerciales selectivamente pueden omitir reglas para esas decisiones que son mejor dejadas para la creatividad de elección humana, en vez de las reglas predefinidas. La pista de reglas apunta a exteriorizar las reglas de operación del negocio a fin de que la comunidad comercial entera y sus sistemas automatizados puedan pensar y puedan actuar espontáneamente y más inteligentemente.

Separando las reglas de los datos y procesos, se maneja la curva de aprendizaje para las personas comerciales. Las reglas documentadas sirven de mensajes de error cuando son violados. Las reglas sirven de base para análisis futuros porque se experimenta con reglas que los líderes comerciales desafían y cambian el comportamiento comercial.

La pista de regla comienza con los pasos para descubrir reglas detrás de un evento comercial o detrás de las decisiones para ese evento. Incluye pasos para reducir esas reglas para un conjunto mínimo pero completo. Le conduce a analizar dependencias entre las reglas así como también las correlaciones de reglas con los datos y las operaciones de la base de datos. Como se puede ver, la metodología comercial de reglas le da una solución muy integrada al desarrollo de sistemas, por definición.

Eso es porque la metodología pide que la expresión de una regla siempre guarda relación con los datos o el modelo comercial de objetos. Esto fomenta una conexión irrompible entre los dos. No hay literalmente ninguna forma para desintegrar la lógica del sistema objetivo (sus reglas) de la base subyacente de datos.

La pista de regla, como todas las pistas, es prevaleciente en todas las fases de la metodología. Comenzando con la fase del alcance, por ejemplo, la pista de reglas es evidente en el contexto comercial de las reglas, como en los objetivos y las políticas. En la fase de descubrimiento, la pista de reglas incluye pasos para captar decisiones que pueden representar grupos lógicos de reglas. En la fase de análisis, la metodología incluye pasos para reducir esas reglas a un conjunto mínimo pero completo. Durante el análisis, se analiza dependencias entre las reglas así como también las correlaciones de reglas para con los datos y las operaciones de la base de datos.

Con una comprensión de las cuatro pistas, ahora se puede suministrar las pistas dentro de las fases de la metodología.

2.1.7 LAS FASES DE LA METODOLOGÍA DE REGLAS DE NEGOCIO

Una metodología de reglas de negocio para el desarrollo de sistemas, como la mayoría de metodologías de desarrollo de sistemas, tiene seis fases conceptualmente diferentes. Estos son el alcance, planificación, descubrimiento, análisis, diseño e implementación. La figura demuestra que cada fase tiene pasos y distribuciones para cada uno de las cuatro pistas.

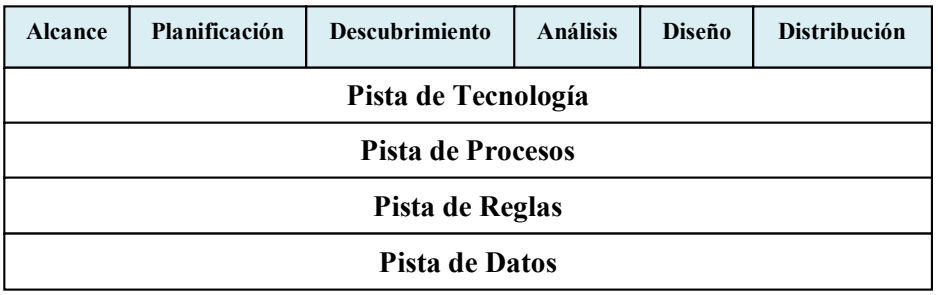


Figura 2.4: Las fases de metodología de sistema de reglas de negocio.
Fuente: Traducción de (Barbara Von Halle, 2002, pág. 71)

No se intenta sugerir una metodología de cascada pasada de moda para el desarrollo de sistemas. Sin embargo, es importante comprender las diferencias conceptuales entre las fases. Por ejemplo, tiene mucha importancia hacer un trabajo minucioso en el alcance y la planificación. Recuerde que una metodología de reglas de negocio tiene como meta entregar sistemas con una base fuertemente arquitectónica (que tome en cuenta los cambios futuros del negocio). Esto quiere decir que la entrega incremental de sistemas puede convertirse en la adición, cambio, y retiro de reglas o conjunto de reglas dentro de esa base.

Esto requiere que el primer incremento incluya los aspectos equilibrantes de la arquitectura de información, las jurisdicciones de reglas, y la administración de reglas. Mientras que la fase de descubrimiento puede ocurrir en partes incrementales, seguido por el análisis, diseño, y entrega, mientras más descubrimiento en partes incrementales, seguido por análisis, diseño, y entrega más descubrimiento ocurre en paralelo.

Fase de Alcance

El alcance es el proceso de captar requisitos comerciales de alto nivel y los límites para un sistema nuevo o existente. A este respecto, la fase del alcance para un sistema comercial de regla no es diferente de otras clases de sistemas.

La primera parte del alcance es la identificación de eventos comerciales, correlacionados que apuntan a las distribuciones del sistema. Las unidades de entrega

incrementales definidas a lo largo de los límites de resultados comerciales de eventos en entregas incrementales que son de valor comercial.

La segunda parte es el alcance de datos, con una comprensión de la extensión de uso compartido de datos a través de la organización. Comprender el alcance de datos permite planificar el análisis de datos, el diseño, e implementación que adecuará necesidades actuales y futuras del negocio y comunidades de usuario.

Desde una perspectiva de reglas de negocio, hay tres consideraciones de reglas comerciales. La primera parte es el contexto comercial para las reglas comerciales eventuales. La segunda consideración es el conjunto de entregables en la preparación para las reglas, específicamente la identificación de políticas como precursores para las reglas. La tercera parte es el propósito para manejar las reglas detrás del sistema.

2.1.7.1.1 Alcance del Contexto Comercial para las Reglas

El contexto comercial es la base comercial para guiar y dar soporte según las reglas. El contexto comercial incluye la misión de la organización, las estrategias, los objetivos, las políticas, y las métricas comerciales de funcionamiento. Estos representan las razones para el sistema y las medidas por las cuales será estimado el éxito, del mismo modo que cambie (sus reglas) con el paso del tiempo.

De mayor importancia para una metodología de reglas de negocio son las políticas que establecen el contexto directo de las reglas. En contraste a otras metodologías, el concepto de políticas comerciales emerge como un aspecto importante del contexto comercial para el sistema. Por consiguiente se buscará políticas comerciales existentes o revisadas que contradigan los fines perseguidos en el sistema, esto conducirá a crear o modificar reglas minuciosamente elaboradas pero cambiables.

Las políticas son las que darán soporte a los objetivos del sistema así como también las que minimizan riesgos del sistema.

2.1.7.1.2 El propósito para Manejar Reglas detrás de un Sistema

Para un sistema de reglas de negocio, es importante determinar, con el patrocinador y stakeholders(los interesados), la razón por la cual se desea manejar las reglas. Por ejemplo, los interesados requieren utilizar administración de reglas para lograr uno o más de lo siguiente:

- Entregar reglas compartidas a través de los límites organizativos de los stakeholders.
- Resolver incongruencias indeseables entre las reglas encontradas.
- Identificar donde es apropiado las reglas o pueden ser inconsistentes.

- Resolver incongruencias en los objetivos comerciales.
- Crear reglas para los nuevos procesos.

Otra diferencia importante durante la fase del alcance es la planificación de la administración de reglas, incluyendo metodología, repositorio, e infraestructura organizacional. Esto lleva a la fase de planificación.

2.1.7.2 La Fase de Planificación

Durante la planificación, se crea un plan de proyecto para construir el sistema comercial de reglas. Un plan de proyecto para un sistema de reglas de negocio incluye un énfasis en lo siguiente:

- Separar reglas a todo lo largo del descubrimiento, análisis, diseño, y distribución.
- Rastrear reglas desde su origen de negocio para la implementación del sistema.
- Exteriorizar reglas para todas las audiencias a través de un repositorio de reglas.
- Situar o localizar reglas para el cambio utilizando o simulando tecnología descrita en reglas.

Para lograr un énfasis en los puntos anteriores, hay que considerar al menos cinco aspectos de su plan de proyecto que es necesario para adecuar una metodología de reglas de negocio.

El primer aspecto a considerar es el conjunto de tareas para establecer estándares de reglas.

El segundo es el conjunto de tareas y conjunto de directrices específicamente para descubrir, analizar, diseñar, y entregar reglas automatizadas como un activo separadamente administrado.

La tercera parte incorpora la oportunidad para probar y destacar tecnología comercial de reglas.

El cuarto aspecto nuevo para su plan de proyecto se ocupa al menos de cuatro roles nuevos para ocuparse de las reglas.

1. Un analista de regla es responsable de capturar reglas de conversaciones de negocio, documentos, o código de programa.
2. Un diseñador de regla es responsable de determinar dónde deben las reglas ser implementadas dentro de la arquitectura de la aplicación.
3. Un implementador de regla es responsable de codificar las reglas ejecutables, aunque los desarrolladores de aplicación o los administradores de la base de datos, según donde las reglas sean implementadas, pueden jugar este papel.
4. Un integrador de reglas o un administrador analiza reglas a través de los eventos comerciales y a través de aplicaciones para asegurar reglas de alta

calidad para la organización. Este rol probablemente también selecciona y maneja el repositorio en el cual las reglas son introducidas y de las cuales las reglas son manejadas.

El quinto aspecto nuevo es el conjunto de tareas para el repositorio de reglas. Estos incluyen documentación de metadatos y los requisitos del repositorio de reglas, una meta-modelo de reglas, y la decisión de un mecanismo de almacenamiento de regla. Se necesitará un manual del usuario para el repositorio de reglas, y quizá los materiales de entrenamiento. Si su proyecto consiste en excavar reglas de sistemas existentes, se necesitará métodos en relación a la forma de hacer esto, y quizá los materiales de entrenamiento.

2.1.7.3 La Fase de Descubrimiento

La fase de descubrimiento revela requisitos detallados del sistema, pero la tecnología permanece neutral. Esta fase tiene dos partes. La primera parte es el descubrimiento de requisitos iniciales. El segundo es el descubrimiento de reglas y los datos.

Se separa la fase del descubrimiento de la fase de análisis por una razón simple: El descubrimiento refiere a revelar requisitos, no a analizarlos.

El propósito de descubrir requisitos iniciales es documentar sólo los aspectos esenciales del comportamiento de sistema porque estos conducen al descubrimiento de las reglas y datos subyacentes. Los aspectos esenciales del comportamiento del sistema incluyen cinco puntos: Las tareas o las actividades detrás de cada evento comercial, las decisiones realizadas en nombre de esas tareas o esas actividades, la información para los que se estableció referencias en la construcción de esas decisiones, el conocimiento creado o los juicios hecho por esas decisiones, y finalmente, los escenarios verdaderos o imaginarios de eventos para probar la integridad del comportamiento de sistema.

Una diferencia significativa en una metodología de reglas de negocio emerge cuando, durante la fase de descubrimiento, se alterna el descubrimiento del comportamiento de sistema con el descubrimiento de decisiones y reglas detrás de los eventos comerciales. Es decir, rápidamente se intercambia el enfoque de eventos y procesos (el aspecto de obrar) al descubrimiento y el análisis formal de toma de decisiones (el aspecto intelectual) detrás de un evento comercial.

Por consiguiente, el propósito de descubrir reglas y datos comienza y nunca se detiene captando reglas, y también solidificando la información y conocimiento. Tenga presente, luego de descubrir las reglas el proceso debería ser iterativo. En un mundo comercial de reglas, el descubrimiento de reglas, esencialmente, nunca llega al final. Después de todo, no es realmente simplemente una fase.

Si se tiene la intención de construir un sistema de información diseñado para cambiar sus reglas, agregar nuevas, y retirar a las viejas. Entonces, el descubrimiento de reglas es un diálogo continuo con la comunidad comercial y eso es una ventaja comercial.

Muchas metodologías previas de desarrollo de sistemas comienzan con comprender la secuencia de interacciones del usuario, secuencia de procesos detrás de esas interacciones, y quizá clases y objetos que aceptan la responsabilidad para cierta funcionalidad. En cambio en una metodología de reglas de negocio, sin embargo, mucho de esto es por ahora dejado en suspenso mientras se mueve rápidamente a descubrir la inteligencia organizacional detrás del evento. La razón por la que no se puede comprender la "secuencia esencial u obligatoria de la interacción del usuario" hasta que se haya revelado la colección esencial subyacente de reglas y un modelo estable correspondiente de datos. Más que eso, al usar productos comerciales de reglas, no se necesita diseñar objetos o clases para ejecutar reglas. La ejecución de regla puede ser manejada por un servicio de reglas o ambiente de desarrollo de reglas.

La fase de descubrimiento tiene un enfoque comunal comercial alto. Se puede descubrir reglas de cualesquiera de las personas comerciales o de código fuente. La meta del descubrimiento de reglas es simplemente darse cuenta que las reglas son o lo que alguien piensa que son o deben ser. Se tiene la intención de expresar las reglas en el lenguaje de la comunidad comercial así es que la audiencia comercial las comprende. Se administra y comunica las reglas en la misma forma que los analistas administran y comunican los datos en elementos existentes de datos en los sistemas o en pantallas e informes.

2.1.7.4 La Fase de Análisis

La fase de análisis aplica la disciplina a los artefactos coleccionados en cada pista. Específicamente, los pasos en la pista de reglas aplican disciplina familiar y nueva a la colección de reglas, del mismo modo que el análisis de datos le da la disciplina a una colección de elementos de datos.

Los pasos de la metodología de análisis de reglas conducen a encontrar incongruencias y redundancias en las reglas. Incluye pasos para producir cadenas de dependencia de reglas, el cuál descubre el "flujo" esencial del "pensar" que emerge del saber de las reglas.

Durante la fase del análisis, se determina cuáles decisiones y reglas básicas son y serán compartidos a través de los límites organizativos de aplicación. El concepto importante a hacer durante el descubrimiento es que los eventos comerciales siguen políticas y requieren que las decisiones sean hechas. Cuando las reglas se ejecutan,

establecen referencias a fragmentos de información y pueden crear fragmentos de información nuevos, llamados conocimiento, para llevar a cabo decisiones. Todo estos activos intelectuales (las decisiones, las reglas, la información base, y el conocimiento creado en reglas) pueden ser compartidos a través de límites organizacionales, con un análisis apropiado, cuando pueden asignarse al negocio.

La fase de análisis incluye tareas para analizar reglas dentro de un conjunto de reglas de alta calidad, crear un modelo lógico de datos enriquecido en reglas, evaluando la calidad de los recursos de datos, y extrayendo reglas del sistema recurso, si es apropiado. Se puede validar reglas a través de factorías de validación de reglas.

El análisis de reglas también tiene una orientación comercial en el paso final. Se refiere a problemas de recuperación de reglas para la audiencia comercial, depurar las reglas, y optimizarlas para el negocio. En un sistema de reglas de negocio, se puede cambiar las reglas más tarde.

Sin embargo, se quiere tener la seguridad de que las reglas dentro de cada incremento nuevo de sistema sean libres de incongruencias y no contengan reglas que parecen no servir para el propósito comercial o se desvían de los objetivos del negocio.

Los pasos de análisis de reglas le llevan de las concatenaciones de dependencia de reglas de vuelta a la interacción de flujo de trabajo y la secuencia de estas. Específicamente, las concatenaciones de dependencia de reglas pueden mejorar la secuencia inicial de interacción del usuario (el flujo de trabajo preliminar) para conservar la secuencia esencial de dependencias de reglas. De aquí, los pasos le conducen a introducir a una secuencia esencial subyacente, las secuencias alternativas para otras razones, como la función o la satisfacción aumentada del cliente. Otra vez, la transición de análisis de regla de regreso al análisis de procesos es muy natural.

Finalmente, en la fase de análisis, están pasos para construir el modelo de datos de los términos y los hechos detrás de las reglas. Otra vez, la transición de las reglas a los datos es muy natural.

Sin embargo, sobre todo, los pasos de Análisis de Datos inducen a analizar el modelo con el futuro en mente donde ese futuro puede mantener cambios en las políticas y las reglas. Hay pasos para desarrollar el modelo de datos del sistema en uno que puede servir para una perspectiva de negocio del cruce organizacional.

El balance entre análisis y diseño es tan controversial como siempre, aun en una metodología de reglas de negocio. El análisis minucioso es deseable para diseños de la base de datos de alta calidad. El diseño de procesos y diseño de reglas puede ocurrir iterativamente, algunas veces con análisis formal pequeño.

2.1.7.5 La Fase del Diseño

Dentro de la pista de reglas, la fase de diseño incluye pasos para clasificar reglas en tipos donde esos tipos pueden ser asignados a las opciones de implementación. Una regla puede ser implementada en la capa de presentación, la capa intermedia, la capa de la base de datos, o una combinación.

Los pasos del diseño de reglas guían al diseñador de reglas determinando como implementar esas reglas en esas capas. Las opciones le incluyen tecnología comercial de reglas, código de cosecha propia, DBMS (disparadores y procedimientos almacenados, etc.). Para las reglas que no serán obligadas a usar tecnología comercial de reglas, hay pasos por los cuales el diseñador los asociará a las operaciones correspondientes (insertar, actualizar, eliminar) de datos para asegurar que cada una de las reglas se dispara en cada instancia. De este modo (a diferencia de las metodologías previas de desarrollo de sistemas) la ejecución de reglas trasciende a nivel transaccional y los límites de aplicación. También, correlacionar las reglas para las operaciones de datos permite análisis.

2.1.8 VERIFICACÍON DE LAS REGLAS DE NEGOCIO

Para que las reglas de negocio de una empresa sean verdaderamente un activo estratégico en relación a otras empresas, la verificación o comprobación es parte esencial de la metodología de reglas de negocio.

En su investigación (Carlos Alberto Mejía Castelo, 2011, pág. 15) manifiesta que "La verificación es el proceso que apunta por la detección de inconsistencia, incompletitud (Lógica. Propiedad de los sistemas lógicos en los que cualquier expresión cerrada no es derivable, dentro del mismo sistema) (Larousse Editorial, 2009) o redundancia en un conjunto de reglas de negocio sin considerar el significado de las reglas, esto significa que no le importa acerca de si la regla es correcta o no".

También sugiere que tener reglas lógicamente consistentes y completas que han sido probadas, podrían dar resultados incorrectos pero lo harán en una manera consistente.

También sugiere que para que una organización pueda estar segura de que sus reglas son consistentes, completas y precisas deben estar sujetas a un ambiente cambiante y tener diferentes fuentes de procedencia.

2.1.9 VALIDACIÓN DE LAS REGLAS DE NEGOCIO

Una vez que se han verificado las reglas de negocio el siguiente paso será comprobar la validez de las reglas en la que se detectaran resultados incorrectos o comportamientos indeseados.

La validación de las reglas será realizada por las personas expertas que dominan la lógica del negocio dentro del contexto empresarial.

La validación de las reglas dentro del contexto de las tecnologías de reglas, se realizará comparando los resultados obtenidos de la aplicación realizada con dichas tecnologías con resultados elaborados por el personal responsable de administrar dichas reglas.

En su tema (Carlos Alberto Mejía Castelo, 2011, pág. 15) manifiesta que las reglas de negocio son aprobadas por miembros de la organización responsable por las reglas, tomando en cuenta los siguientes puntos que son transcritos a continuación:

- Inconsistencia: Es una condición en un conjunto de reglas de negocio que ocurre cuando 2 o más reglas llevan a comportamientos o resultados conflictivos.
- Incompletitud: Es una condición en un conjunto de reglas que ocurre cuando existe un caso de negocio que lleva a un resultado o comportamiento indefinido.
- Redundancia: Es una condición en un conjunto de reglas de negocio que ocurre cuando existe una regla de negocio que no tiene contribución significativa al posible comportamiento o posibles resultados.
- Anomalía: Es un término más general para inconsistencia, Incompletitud o redundancia.

De igual forma (Carlos Alberto Mejía Castelo, 2011, pág. 16) también propone algunos puntos para considerar cuando un conjunto de reglas tiene una alta calidad y describe las reglas de negocios formales e informales:

- Cada caso distinto de negocio lleva a solo un comportamiento o resultado posible.
- Cada posible caso de negocio lleva a algún resultado o comportamiento.
- Cada regla, aplicada en cualquier caso de negocio, tiene una contribución significativa o comportamiento.

Reglas de negocio informales: Son declaraciones en lenguaje natural.

Reglas de negocio formales: Son declaraciones en lenguaje matemático o lógico.

2.1.10 SISTEMAS DE GESTIÓN DE REGLAS DE NEGOCIO

Un Sistema de Gestión de Reglas de Negocio (BUSINESS RULES MANAGEMENT SYSTEM, BRMS) es un sistema de software que permite al usuario definir, desplegar, ejecutar, monitorear, reutilizar, depurar, separar y evaluar una compleja lógica de decisiones o reglas de negocio que son usadas por sistemas dentro de una organización o empresa; esto incluye políticas, requisitos, declaraciones condicionales dentro del contexto del negocio, ejecutando las acciones apropiadas.

Exteriorizando reglas comerciales a través de herramientas que permitan manejar reglas, un BRMS permite a los expertos comerciales definir y mantener las decisiones que guían el comportamiento de sistemas, reduciendo la cantidad de tiempo y el esfuerzo requerido para actualizar sistemas de producción, y aumentando la habilidad de la organización o empresa para responder a los cambios en un ambiente comercial competitivo.

Las soluciones BRMS automatizan políticas en aplicaciones personalizadas, complejas y comerciales. Los costos inferiores de mantenimiento de la aplicación, facilita más la implementación precisa y coherente de política de negocio a través de aplicaciones, y enriquece la colaboración entre su negocio y los departamentos de tecnología de la información.

En su tema (Carlos Alberto Mejía Castelo, 2011, pág. 19) manifiesta que "Un BRMS se compone de tres elementos básicos: un motor de reglas, encargado de ejecutar las reglas de negocio, un sistema de gestión de configuración, encargado de almacenar las reglas de negocio utilizadas y una interfaz de acceso para los usuarios que permita la gestión y edición de las reglas de negocio".

También dice que los BRMS ejecutan reglas de negocio representadas a través de una expresión de tipo "si – entonces", que son ejecutables en un motor de reglas. Lo siguiente es una transcripción de (Carlos Alberto Mejía Castelo, 2011, págs. 20-21). Los principales componentes y funcionalidades que comparten la mayoría los BRMS son los siguientes:

Un repositorio para la gestión de la configuración: Este repositorio permitirá almacenar distintas versiones de las reglas de negocio, llevando un control de los cambios que se han producido en el contenido de la regla y los usuarios que los han efectuado.

Un motor de reglas: Capaz de ejecutar y gestionar las reglas de negocio de forma eficiente y adecuada, el motor almacena una serie de hechos que codifican el conocimiento que tiene de un problema.

Sobre estos hechos aplica un conjunto de reglas expresadas como sentencias de control condicionales, a veces denominadas "si entonces", que permiten inferir nuevos hechos.

Integración con las aplicaciones de desarrollo de la organización: Es necesario integrar el proceso de identificación y desarrollo de reglas de negocio en el proceso de desarrollo software contemplado en la organización. Muchos de los BRMS disponibles actualmente proporcionan entornos de desarrollo basados en modelos y que contemplan distintos roles para los usuarios. Estos entornos pretenden reducir la cantidad de código a generar por los desarrolladores a la hora de introducir las reglas de negocio en los sistemas de información de la organización.

Simulación de las reglas: La validación o prueba del comportamiento de una regla de negocio es un paso esencial en el ciclo de desarrollo de la misma, al igual que para cualquier otro componente software.

Monitorización y análisis: Las herramientas de monitorización y análisis proporcionan estadísticas de ejecución e informes de utilización de las reglas de negocio. Este componente permite saber qué reglas se ejecutaron durante una transacción determinada, en qué momento y quién las ejecutó, cuál fue el resultado de la ejecución, con que otros sistemas se interactuó y, en general, toda aquella información útil para el seguimiento del comportamiento de las reglas de negocio.

Gestión y administración: Muchos de los BRMS disponibles en la actualidad incorporan herramientas que permiten desplegar las reglas de negocio en los sistemas de información de la organización, gestionar la seguridad en el acceso a las mismas, sus versiones y efectuar un seguimiento del rendimiento y el estado del BRMS durante su funcionamiento.

Reutilización de reglas: La posibilidad de reutilizar las reglas de negocio es una facilidad básica que deberían ofrecer todos los BRMS actuales.

2.1.11 CARACTERÍSTICAS DE UN BRMS

En su tema de (Carlos Alberto Mejía Castelo, 2011, págs. 21-22) expone características de una gran variedad de productos, tanto comerciales como de código abierto, que implementan motores de reglas, todos estos motores comparten un conjunto de características.

- Implementan un algoritmo de ejecución de reglas eficiente. La mayoría de los motores de reglas actuales incluyen una implementación del algoritmo, RETE (Quiere decir "red". (The JBoss Drools team, pág. 16), el más eficiente

conocido actualmente para la verificación de antecedentes y consecuentes de las reglas de negocio.

- Incorporan gestión de configuración. Un factor importante a tener en cuenta a la hora de trabajar con reglas de negocio es la necesidad de disponer de información de versiones existentes de una misma regla, además de conocer los usuarios que modificaron las distintas versiones y que elaboraron aquellas que, en un instante determinado, se encuentran en un entorno de producción.
- Distinguen roles de usuarios. La principal ventaja de los BRMS radica en la posibilidad de modificar la lógica de negocio sin necesidad de recompilar y reconstruir el código de la aplicación que da soporte a dicha lógica de negocio. Esto conduce a la posibilidad de que los analistas de negocio o expertos en el dominio puedan, sin conocimientos técnicos profundos, modificar el comportamiento de una aplicación por sí mismos.
- Incluyen herramientas de despliegue. Aparte de permitir la creación y definición de las reglas de negocio, es deseable disponer de facilidades para el despliegue de las reglas de negocio. Este componente de despliegue estaría encargado de situar los artefactos que conforman una regla de negocio en el entorno de ejecución adecuado. También estaría al tanto de las dependencias existentes entre el motor de reglas y el resto de componentes de software disponibles en la organización, asegurando que se encuentra en ejecución la versión adecuada para la regla y manteniendo registro del proceso de ejecución.
- Incluyen herramientas de validación. Como parte del entorno de edición de las reglas de negocio, la mayoría de los sistemas incluyen la posibilidad de depurar y validar la ejecución de las reglas de negocio creadas. Estas facilidades de depuración suelen integrarse en los entornos de edición para los desarrolladores de las reglas de negocio.
- Integración con el proceso de desarrollo. La mayoría de los BRMS suelen incorporar una interfaz basada en un navegador para los expertos de negocio y otra interfaz propia de desarrolladores integrada en algún entorno de desarrollo como puede ser Eclipse (IDE Entorno de desarrollo Integrado) en el caso de considerar Java como lenguaje de programación. Por otra parte, el desarrollo de reglas de negocio debe apoyarse en los modelos de objetos existentes en la organización, los cuales, por norma general, deben importarse en el entorno de desarrollo de reglas de negocio.

2.1.12 MOTOR DE REGLAS DE NEGOCIO

En la documentación oficial de Drools (The JBoss Drools team, pág. 10) proporciona una historia breve de Inteligencia Artificial (AI) en la que muestra el inicio de los motores de reglas de negocio.

Un motor de reglas tiene como base "La Representación de Conocimiento y el Razonamiento".

La Representación de Conocimiento y el Razonamiento (KRR) se trata como se representa el conocimiento en forma simbólica, o sea cómo se describe algo. El razonamiento es acerca de cómo se emprende el acto de pensar usando este conocimiento (The JBoss Drools team, pág. 2)

En su investigación (Castelo Carlos Alberto Mejía, 2011, pág. 23) dice que "Un motor de reglas de negocio es un sistema de información, que ejecuta reglas de negocio o monitorea actividades de negocio en términos del cumplimiento de las reglas en tiempo de ejecución.

Los motores de reglas de negocio generalmente incluyen características para el desarrollo de reglas de negocio que permiten una definición conveniente y la verificación de las reglas".

2.1.13 VENTAJAS DE UN MOTOR DE REGLAS

Las siguientes ventajas son una traducción y resumen del documento oficial de Drools (The JBoss Drools team, págs. 13-14).

2.1.13.1 Programación Declarativa:

Los motores de reglas permiten decir “qué Hacer”, y no “cómo hacerlo”. La ventaja más significativa es que se puede usar reglas para facilitar expresar soluciones a los problemas difíciles y consecuentemente verificar esas soluciones. Las reglas son mucho más fáciles que leer código fuente.

Los sistemas de reglas son capaces de solucionar muy bien, problemas muy complejos, proporcionar una explicación de cómo se logró la solución y por qué cada “decisión” a lo largo del camino fue tomada (mucho más fácil que con otros sistemas de Inteligencia Artificial como las redes neurales o el cerebro humano).

2.1.13.2 Separación de la Lógica de los Datos:

Los datos están en el dominio de los objetos, la lógica está en las reglas. Esto fundamentalmente rompe con la programación orientada a objetos donde se asocia los datos y la lógica, lo cual puede ser una ventaja o una desventaja dependiendo del punto

de vista. La conclusión es que la lógica puede ser mucho más fácil de mantener cuando hay cambios en el futuro, cuando la lógica es colocada en las reglas.

2.1.13.4 Velocidad y Escalabilidad:

El algoritmo Rete y sus descendientes como el algoritmo Rete OO (algoritmo Rete orientado a objetos) de Drools, proveen formas muy eficientes de patrones de regla que emparejan datos del dominio de los objetos.

2.1.13.5 Centralización de Conocimiento:

Usando reglas, se crea un repositorio de conocimiento (una base de conocimiento) que es ejecutable.

Esto quiere decir que es un punto único de almacenamiento de reglas, para las políticas comerciales, y las reglas son tan legibles que también pueden servir de documentación.

2.1.13.6 Integración con Herramientas:

Las herramientas como Eclipse y GUVNOR (repositorio de reglas con interfaz web para el usuario) que proveen formas para revisar, manejar reglas y obtener asistencia de retroalimentación inmediata, de validación y de contenido.

2.1.13.7 Facilidad de explicación:

Los sistemas de reglas eficazmente proveen una "facilidad de explicación" pudiendo registrar las decisiones hechas por el motor de reglas junto con él para que las decisiones fueron hechas.

2.1.13.8 Reglas Comprensibles:

Creando modelos de objetos y, optativamente, Lenguajes Específicos de Dominio que modelan el dominio problemático al que se puede establecer y escribir las reglas que están muy cerca del lenguaje natural.

2.1.14 PARTES DE UN MOTOR DE REGLAS

2.1.14.1 Editor de reglas

Es donde se podrá escribir las reglas de negocio para que el motor de reglas de producción las pueda consumir y pueden ser: un sistema de administración de reglas de negocio (BRMS) como Guvnor, IDE como Eclipse, tablas de decisiones con archivos

*.xlsx o archivos plano de texto con extensión *.drl (Leguaje de reglas Drools) o archivos *.xml.

2.1.14.2 Compilador de reglas

El compilador es un software que permite traducir las reglas de negocio escritas en un lenguaje natural hacia un formato para que el motor de reglas de producción las pueda entender.

2.1.14.3 Modelo de hechos

El modelo de hechos permite definir la información que manejará el motor de reglas de producción, por lo que tanto la descripción de un hecho como su formato deben estar claramente especificados.

2.1.14.4 Repositorio de reglas

Un repositorio de reglas permite centralizar la administración de dichas reglas en la que se podrá crear, modificar, eliminar, manejar sus versiones y su almacenamiento será seguro.

2.1.15 CONCEPTOS DE LOS MOTORES DE REGLAS

2.1.15.1 Hechos

Los hechos se definen como el contenedor en el que se transporta información desde y hacia el motor de reglas de negocio, estos pueden ser objetos planos en Java (Castelo Carlos Alberto Mejía, 2011, pág. 26)

2.1.15.2 Memoria de trabajo

En aplicaciones reales cada usuario que necesita acceder a un motor de reglas de producción se le asigna una única sesión en la memoria de trabajo la que contiene todo el conocimiento (hechos) y los cambios que se produzcan ya sea de inserción, actualización, eliminación o provocados por las reglas se verán reflejados.

2.1.15.3 Concordancia de patrones

Lo siguiente es una traducción y resumen de la documentación oficial de Drools (The JBoss Drools team, pág. 4)

El motor empareja hechos y datos en contra de las Reglas de Producción para inferir conclusiones que dan como resultado acciones.

El proceso de emparejar hechos nuevos o existentes en contra de las reglas de producción es llamado patrón de emparejamiento o concordancia, el cual es realizado por el motor de inferencia. Las acciones se ejecutan en respuesta a los cambios en los datos; Se dice que es una metodología manejada a través de datos para razonar. Las acciones mismas pueden cambiar datos, lo cual a su vez podría emparejar en contra de otras reglas causando que estas se disparen; A esto se refiere como emparejamiento hacia adelante.

Las Reglas son almacenadas en la memoria de producción y los hechos que el motor de inferencia empareja son guardados en la Memoria de Trabajo. Los hechos son afirmados en la Memoria de Trabajo donde luego pueden ser modificados o revocados. Un sistema con un gran número de reglas y hechos pueden resultar en muchas reglas siendo ciertos para la misma afirmación de un hecho; Se dice que estas reglas están en conflicto. La Agenda administra el orden de ejecución de estas reglas conflictivas usando una estrategia de Resolución de Conflicto.

2.1.15.4 Resolución de conflictos

Lo siguiente es una traducción y resumen de la documentación oficial de Drools (The JBoss Drools team, pág. 101).

La resolución de conflictos es requerido cuando hay múltiples reglas en el Agenda. Cuando se dispara una regla puede tener efectos secundarios en la Memoria de Trabajo, el motor de reglas necesita conocer en qué orden las reglas deberían ser disparadas (por ejemplo, disparar la regla A puede causar que la regla B sea removida de la agenda).

Las estrategias de resolución de conflictos predeterminadas empleadas por Drools son: Prioridad y LIFO (Last In, First Out (último en entrar, primero en salir)).

Lo más posible es la prioridad, en cuyo caso un usuario puede especificar que una cierta regla tiene una prioridad superior (dándole un número más alto) que otras reglas. En ese caso, la regla con prioridad más alta será preferida. Las prioridades LIFO se refieren al orden de carga, las reglas que se cargaron primero, son las que se ejecutaran primero.

2.1.15.5 Mantenimiento de la verdad

(Carlos Alberto Mejía Castelo, 2011, pág. 28) En su tema describe de una forma clara en que consiste el mantenimiento de la verdad.

"La memoria de trabajo está en continuo cambio, ya que las reglas en ejecución pueden cambiar los hechos sobre los cuales se está trabajando, existe la posibilidad de que nuevas reglas se conviertan en reglas relevantes para la nueva situación, además,

estas nuevas reglas relevantes que serán agregadas a la agenda y ejecutadas posteriormente, tienen la posibilidad de cambiar los hechos y crear aún más escenarios en que otras reglas de negocio se convierten en relevantes.

Este proceso conocido como mantenimiento de la verdad, garantiza que ha finalizado un tiempo de ejecución, y asumiendo que las reglas de negocio están completas y verificadas, se obtendrá un conjunto de hechos que representa la verdad, siendo esta la verdad como la entiende el motor de reglas.

Existe claramente la posibilidad de que la ejecución de reglas cree un ciclo de ejecución, causando que la misma regla o el mismo conjunto de reglas se ejecute un sin número de veces dado un conjunto de hechos, esta situación debe controlarse desde el momento en que se escriben las reglas, en primer lugar escribiendo las reglas de la forma más precisa posible, en segundo lugar aplicando atributos a las reglas que indiquen un número máximo de veces que cada regla puede ser ejecutada, o usando el flujo de reglas".

2.1.15.6 Algoritmo Rete

Lo siguiente es una traducción propia del documento oficial de Drools (The JBoss Drools team, págs. 16-22).

El algoritmo Rete fue inventado por el Dr. Charles Forgy y documentado en su tesis PhD en 1978-79. Una versión simplificada del periódico fue publicada en 1982. La palabra latina "rete" quiere decir "red".

El algoritmo Rete puede ser separado en 2 partes: La compilación de reglas y la ejecución en tiempo de ejecución.

El algoritmo de compilación describe cómo las Reglas en la Memoria de Producción son procesadas para generar una red eficiente de discriminación. En términos poco técnicos, una red de discriminación (Discriminar: Separar, distinguir, diferenciar, una cosa de otra. (Larousse, 2002, pág. 351)) Es usada para filtrar datos como se propagan a través de la red. Los nodos en lo alto de la red tendrían muchos emparejamientos, y al bajar la red, habría menos emparejamientos. En la misma raíz de la red están los nodos terminales. En el escrito el Dr. Forgy 1982, él describió 4 nodos básicos: Raíz, input 1, input 2 y terminal.

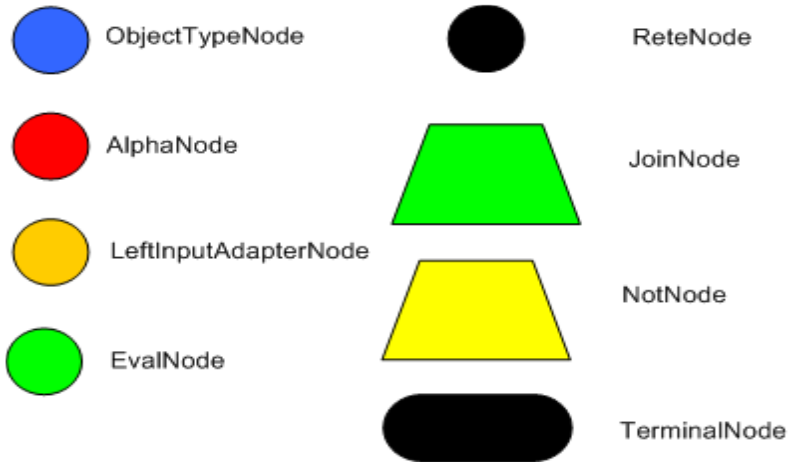


Figura 2.5: Nodos Rete.
Fuente: Tomado de (The JBoss Drools team, 2009, pág. 17)

El nodo raíz está donde todos los objetos entran en la red. De allí, inmediatamente va al ObjectTypeNode. El propósito del ObjectTypeNode es asegurar que el motor no haga más trabajo que él que necesita. Por ejemplo, se tiene 2 objetos: Cuenta y Orden. Si el motor de reglas intentase evaluar cada nodo solo en contra de cada objeto, desaprovecharía una buena cantidad de ciclos. Para hacer esto eficientemente, el motor sólo le debería pasar el objeto a los nodos que emparejen el tipo del objeto. La forma más fácil para hacer esto es crear a un ObjectTypeNode y tener todos los nodos 1 input y 2 input descendientes de este. Así, si una aplicación afirma una Cuenta nueva, no se propagará para los nodos para el objeto de Orden. En Drools cuando un objeto es afirmado este recupera una lista válida de ObjectTypesNodes por una búsqueda en un HashMap de Clase del objeto; Si esta lista no existe, escanea a todos los emparejamientos validos encontrados de ObjectTypeNodes que coloca en reserva en la lista. Esto le permite a Drools emparejar en contra de cualquier tipo Clase que empareja con una comprobación de instanceof.

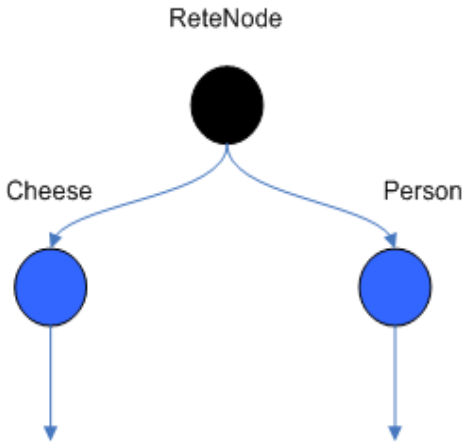


Figura 2.6: ObjectTypeNodes.
Fuente: Tomado de (The JBoss Drools team, pág. 18)

ObjectTypeNodes puede propagarse hacia AlphaNodes, LeftInputAdapterNodes y BetaNodes.

AlphaNodes se usa para evalúan condiciones literales. Por ejemplo, Account.name == "Mr Trout" es una condición literal. Cuando una regla dispone de condiciones literales múltiples para un solo tipo de objeto, son vinculadas. Esto quiere decir que si una aplicación afirma un objeto Cuenta, primero debe satisfacer la primera condición literal antes de que pueda proceder al siguiente AlphaNode. En el escrito del Dr. Forgy, él se refiere a estos como las condiciones IntraElement. El siguiente diagrama muestra al AlphaNode las combinaciones para la clase Cheese (name cheddar, strength strong):

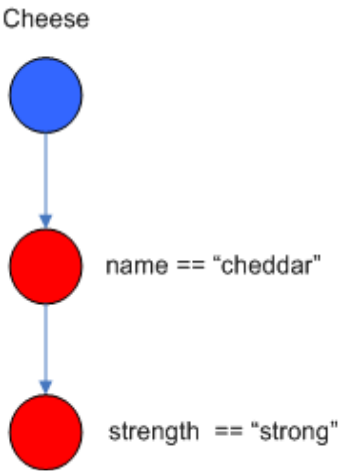


Figura 2.7: AlphaNodes.
Fuente: Tomado de (The JBoss Drools Team, pág. 18)

Drools extiende el algoritmo Rete optimizando la propagación del ObjectTypeNode hacia AlphaNode usando un Hashing (Es un método para almacenar y recuperar registros de una base de datos. Permite insertar, suprimir, y la búsqueda de registros basados en un valor clave de búsqueda.). Cada vez que un AlphaNode es añadido a un ObjectTypeNode le añade el valor literal como una clave al HashMap con el AlphaNode como el valor. Cuando una instancia nueva entra en el nodo ObjectType, en vez de propagarse para cada AlphaNode, en lugar de eso puede recuperar al AlphaNode correcto del HashMap, por consiguiente evitando innecesarios chequeos literales.

Hay dos nodos de entrada, JoinNode y NotNode, y ambos son tipos de BetaNodes. BetaNodes se usa para comparar 2 objetos, y sus campos, el uno para el otro. Los objetos pueden ser los mismos o de tipos diferentes. Por convención se refiere a los dos nodos de entrada como izquierdo y el derecho. El nodo de entrada izquierdo para un BetaNode es generalmente una lista de objetos; En Drools éste es un Tuple (Una tupla, en matemáticas, es una secuencia ordenada de objetos, esto es, una lista con un número limitado de objetos). El nodo de entrada derecho es un solo objeto solo.

Dos Nodos pueden usarse para implementar la verificación " exists". BetaNodes también tiene memoria. El nodo de entrada izquierdo es llamado Beta Memory y recuerda todos los tuples entrantes.

El nodo entrante derecho es llamado Alpha Memory y recuerdan todos los objetos entrantes. Drools extiende Rete realizando indexando en los BetaNodes. Por ejemplo, si se sabe que un BetaNode realiza una comprobación en un campo String, como cada objeto entrante se puede hacer una búsqueda hash en ese valor String. Esto quiere decir cuándo entran los hechos del lado contrario, en lugar de iterar sobre todos los hechos para encontrar junturas válidas, se hace una búsqueda devolviendo potencialmente a candidatos válidos. En cualquier punto una juntura válida es encontrada que el Tuple está unido al Objeto; Que es llamado un emparejamiento parcial; Y luego propagado para el siguiente nodo.

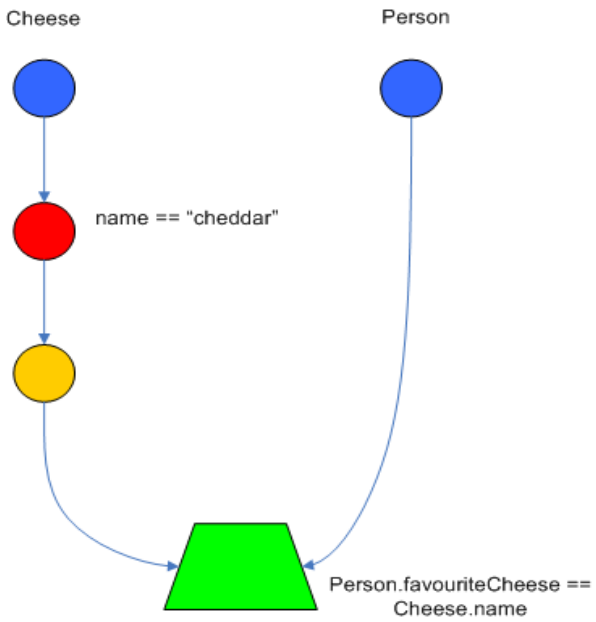


Figura 2.8: JoinNode.
Fuente: Tomado de (The JBoss Drools team, pág. 20)

Para habilitar el primer Objeto, en el anteriormente caso citado Cheese, entra en la red usa un LeftInputNodeAdapter - éste toma un Objeto como una entrada y propaga a un solo Objeto Tuple.

Los nodos terminales son usados para indicar a una sola regla a emparejado todas sus condiciones; En este punto se dice que la regla tiene un emparejamiento completo. Una regla con un "or" los resultados conectivos disyuntivos condicionales en generación de sub reglas pues lógicamente posible se ramifica; Así una regla puede tener nodos terminales múltiples.

Drools también realiza uso compartido del nodo. Muchas reglas repiten los mismos patrones, y el compartir nodos deja colapsar esos patrones a fin de que no tengan que ser reevaluados para cada instancia sola.

Las siguientes dos reglas comparten el primer patrón, pero no lo último:

```
rule
when
       Cheese( $cheddar : name == "cheddar" )
       $person : Person( favouriteCheese == $cheddar )
then
       System.out.println( $person.getName() + " likes cheddar" );
end
```

Como se puede ver abajo, la red compilada Rete demuestra que el nodo alpha es compartido, pero los nodos betas no lo son. Cada nodo beta tiene a su TerminalNode. Si el segundo patrón hubiera sido lo mismo también habría sido compartido.

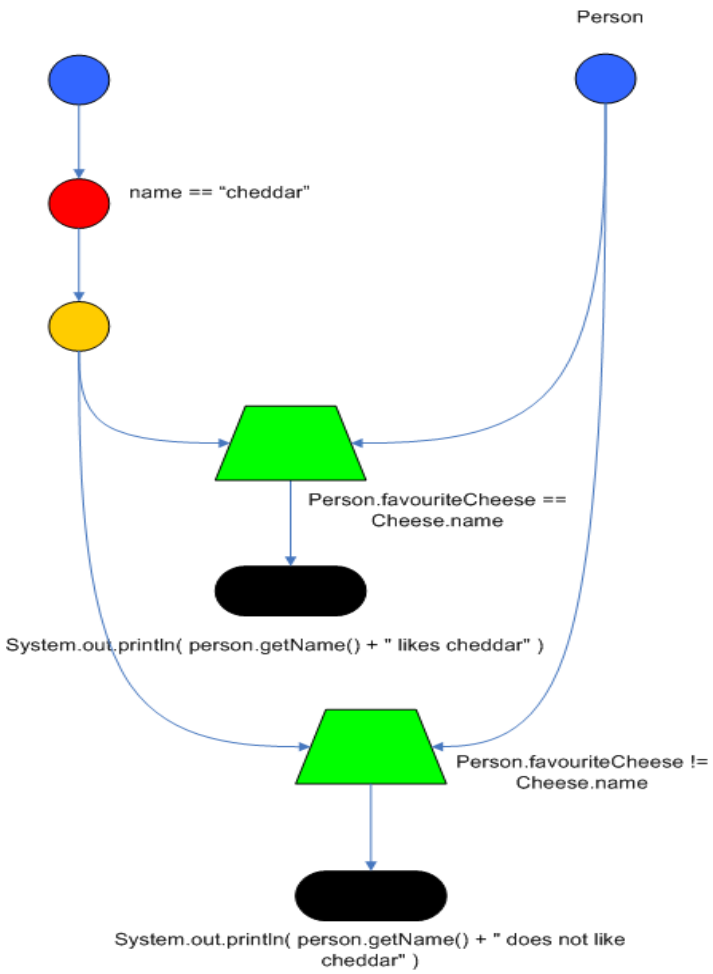


Figura 2.9: Node Sharing.
Fuente: Tomado de (The JBoss Drools team, pág. 22)

Refiérase al documento (Castelo Carlos Alberto Mejía, 2011, pág. 23) que explica sobre el algoritmo Rete.

2.1.15.7 Algoritmo de Inferencia

Un algoritmo de inferencia (algunas veces llamado "estrategia de razonamiento") es el componente en un motor de inferencia que decide que reglas deberían ser evaluadas y cuando, para resolver un problema en particular.

El motor de interferencia tratara de encontrar un camino óptimo dentro de un conjunto de reglas para resolver un problema o tratar una situación.

Existen 2 estrategias bien conocidas que pueden ser implementadas por un algoritmo de inferencia: “forward chaining” y “backward chaining”, estas estrategias son aplicadas a las reglas del tipo de reglas de producción.

2.1.15.8 Encadenamiento hacia delante (Forward chaining)

Lo siguiente es una traducción propia del documento oficial de Drools (The JBoss Drools team, pág. 5) .

Es "data-driven" conducido por datos de tal manera reaccionario, con los hechos siendo afirmados en la Memoria de Trabajo, lo cual resulta en uno o más reglas siendo concurrentemente verdadero y programado para la ejecución por la Agenda.

En resumen, se comienza con un hecho, se propaga a través de las reglas, y se termina en una conclusión.

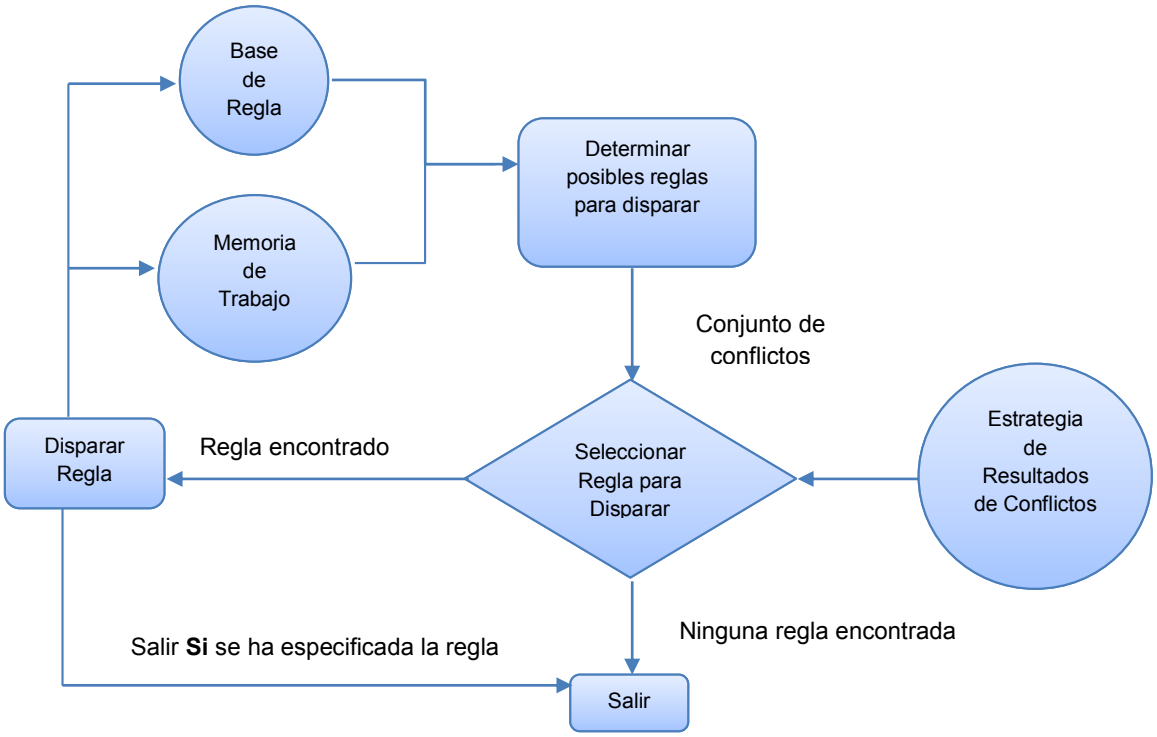


Figura 2.10: Forward Chaining.
Fuente: Tomado de (The JBoss Drools team, pág. 7)

2.1.15.9 Encadenamiento hacia tras (Backward chaining)

Lo siguiente es una traducción propia del documento oficial de Drools (The JBoss Drools team, pág. 6).

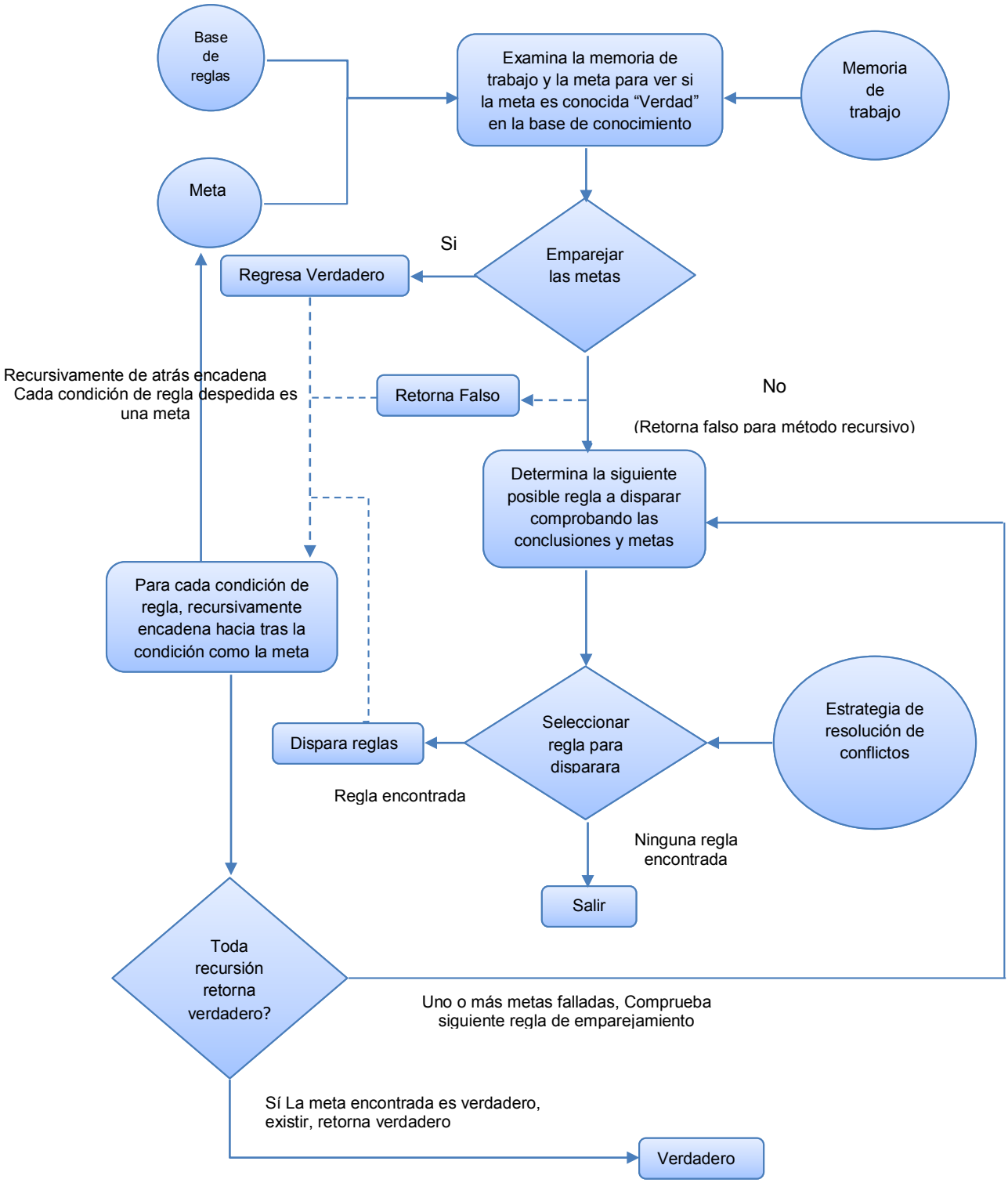


Figura 2.11: Backward chaining.
Fuente: Tomado de (The JBoss Drools team, pág. 6)

Es "goal-driven" manejado hacia la meta, quiere decir que se comienza con una conclusión que el motor intenta satisfacer. Si no puede, luego va en busca de conclusiones que puede satisfacer. Estos son conocidos como las sub metas, eso ayudará a satisfacer alguna parte desconocida de la meta actual. Continúa este proceso hasta ya sea la conclusión inicial es probado o no hay más sub metas. Prolog es un ejemplo de un motor Backward Chaining. Drools también pueden hacer a backward chaining, que se refiere a como la derivación queries o búsquedas.

2.1.16 TIENDAS VIRTUALES

Lo siguiente es un resumen del sitio web por (Thompson Ivan, 2012) para tener idea de que es una tienda virtual y lo que se debe tener en cuenta cuando ya se tiene una. Dice que el objetivo de una tienda virtual es captar más clientes, satisfacer a los actuales y aumentar las ventas, que es el objetivo que se persigue.

"Una Tienda Virtual es un espacio en internet donde se puede poner a la venta durante las 24 horas del día y los 7 días de la semana diversos tipos de productos de forma muy similar a como se los exhibiría y vendería en una tienda tradicional".

A continuación se resume algunas razones a tener presente para obtener una tienda virtual:

Una tienda virtual no está limitada por la capacidad de atención a los clientes al mismo tiempo con respecto a la presentación de productos, recepción de preguntas, opiniones, quejas, solicitudes de cotizaciones, ventas. En cambio, una tienda tradicional depende de su capacidad física para atender a sus clientes en el tamaño del local y personal de atención disponible.

Una tienda virtual puede ser accesible desde cualquier lugar donde haya una computadora conectada a internet.

Una tienda virtual puede estar disponible en cualquier parte del mundo, lo que da la posibilidad de captar clientes a nivel local, nacional, e internacional dependiendo del tipo de producto.

En una tienda virtual los productos pueden rivalizar con cualquier competidor de su sector dando una posición estratégica que permitirá atraer a los clientes de otras empresas.

Una tienda virtual con respecto al costo de equipamiento, implementación y de personal resulta mucho más económica que una tienda tradicional gracias a empresas que se han encargado de desarrollar servicios que permiten tener una Tienda Virtual a muy bajo costo.

Igualmente con respecto a la forma de cobrar en una tienda virtual, el sitio web (Thompson Ivan, 2012) dice que es muy importante las diferentes formas de pago que se debe ofrecer al cliente cuando ha decidido comprar en nuestra tienda.

Existen dos opciones para cobrar en internet: 1) On-Line (en línea) y 2) Off-Line (fuera de línea). Ambas, son necesarias para que la mayor cantidad de clientes puedan efectuar su compra en internet.

La cobranza on-line es aquella en la que se realiza el cobro en un sitio web o tienda virtual sin tener contacto directo o personal con el cliente. El elemento más solicitado para realizar este tipo de cobranza es la «Tarjeta de Crédito».

La cobranza off-line o fuera de línea es utilizada cuando los clientes no tienen Tarjeta de Crédito o tienen temor de utilizarla en internet se lo puede realizar por Western Union (http://www.westernunion.com)

De igual manera en el sitio web (Azuanet Soluciones Web, s.f.) Expone 9 razones para tener una tienda online, vender por internet.

El negocio abierto las 24 horas. ¿Cuánto costaría tener el negocio abierto las 24 horas del día durante todo un año? Teniendo una tienda online muy poco. Una de las grandes ventajas de tener su tienda online es poder competir durante todo el día con negocios similares al suyo.

Reducción de gastos. Con una tienda online los gastos serán mínimos y solo deberá de preocuparse de la buena atención al cliente y el correcto envío de los pedidos.

Estudiar mejor al cliente, mediante las tiendas online y la web analítica se puede saber las preferencias del cliente de forma estadística con el fin de poder ofrecer mejores productos al cliente indicado.

No tiene limitaciones geográficas. Mediante una tienda online se podrá vender a todo el mundo sin limitaciones de zonas como podría ocurrir en una ciudad.

Si tiene un buen producto, tendrá unas muy buenas ventas. Así es, cuando un producto gusta y la gente lo aprecia, su negocio será recomendado y podrá experimentar un crecimiento muy superior al de una tienda física. Sin límites ni fronteras.

Rapidez en los pedidos. Con su tienda online solamente le molestarán cuando hayan realizado un pedido, es el cliente el que crea su pedido y usted solo se preocupa de ver el correcto pago y posterior envío.

No es necesario tener Stock de grandes cantidades. Si se sabe elegir distribuidores de una forma correcta, no se necesita comprar stock por adelantado para servirlo en el sitio al cliente sino que se ahorros en la compra de stock hasta que el cliente no haya efectuado el pago.

Aumento de compras por internet. Cada día hay más personas que compran en tiendas online, o lo que es lo mismo: por internet. El tiempo es uno de los factores principales que hacen que el consumidor efectúe de una forma fácil e intuitiva una compra, ahorrándose desplazamientos al centro comercial.

Mientras antes empieza a vender por Internet, más le costará después ponerse a la altura de su competencia, estudiar a sus clientes y crear marca para tener una tienda de confianza.

Con respecto al comercio electrónico, tiendas virtuales el tema es muy basto que resulta demasiado profundizar y no es el objetivo de este tema. Por lo que se recomienda la lectura del libro" Diseño de sitios Web Manual de referencia" de Thomas A. Powell, Mc Graw Hill, para profundizar en los principios básicos de diseño de sitios web.

Para profundizar sobre tiendas virtuales se recomienda el libro "El libro blanco del comercio electrónico - Guía práctica de comercio electrónico para pymes, autor Asociación Española de la Economía Digital (adigital) 2012, también el libro de Comercio Electrónico de Crovetto Huerta Christian y Alarcon Herrera Erika 2005.

2.1.17 METODOLOGÍA DE DESARROLLO DE SOFTWARE

2.1.17.1 Introducción

Se denomina Metodología a un proceso de software detallado y completo. Las metodologías se basan en una combinación de los modelos de procesos genéricos como cascada, evolutivo, incremental, espiral entre otros.

Además una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, entre otras.

2.1.17.2 Metodología de Desarrollo Web

El desarrollo de aplicaciones Web involucra decisiones importantes de diseño y de implementación que influyen en todo el proceso de desarrollo; el alcance de la aplicación y el tipo de usuarios son consideraciones tan importantes como las tecnologías elegidas para realizar la implementación.

De esta forma, así como las tecnologías pueden limitar la funcionalidad de la aplicación, decisiones de diseño equivocadas también pueden reducir la satisfacción del usuario. Por lo tanto, desde el punto de vista de la ingeniería del software, es importante proveer mecanismos adecuados que influyan directamente en su construcción, para que

la realización de este tipo de aplicaciones satisfaga las necesidades tanto de los usuarios como de los clientes que contratan el desarrollo de este tipo de aplicaciones.

2.1.17.3 Metodología de Desarrollo RUP

2.1.17.3.1 Introducción

El Proceso Unificado de Racional. Es un proceso de ingeniería de software que suministra un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga la necesidad del usuario final dentro de un tiempo y presupuesto previsible. Es una metodología de desarrollo iterativo enfocada hacia los casos de uso, manejo de riesgos y el manejo de la arquitectura.

El RUP mejora la productividad del equipo ya que permite que cada miembro del grupo sin importar su responsabilidad específica acceda a la misma base de datos de conocimiento. Esto hace que todos compartan el mismo lenguaje, la misma visión y el mismo proceso acerca de cómo desarrollar software.

Comprende tres conceptos claves: es dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental, además tiene cuatro fases que son: Inicio, Elaboración, Construcción y Transición que son la parte dinámica del proceso, en cada iteración se deben realizar actividades o flujos de trabajo que dependiendo en qué fase de desarrollo en la que se encuentre el proyecto se da mayor interés a cada uno de ellos.

2.1.17.3.2 Principales Características

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software.

El RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso). (Diaz Flores, 1996)

2.1.17.3.3 Fases de la Metodología RUP

- Fase de Inicio

  Durante esta fase de inicio las iteraciones se centran con mayor énfasis en las actividades de modelamiento de la empresa y en sus requerimientos

- Fase de Elaboración

  Durante esta fase de elaboración, las iteraciones se centran al desarrollo de la base de la diseño, encierran más los flujos de trabajo de requerimientos, modelo de la organización, análisis, diseño y una parte de implementación orientada a la base de la construcción

- Fase de Construcción

  Durante esta fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones las cuales se seleccionan algunos Casos de Uso, se redefine su análisis y diseño y se procede a su implantación y pruebas. En esta fase se realiza una pequeña cascada para cada ciclo, se realizan tantas iteraciones hasta que se termine la nueva implementación del producto.

- Fase de Transición

  Durante esta fase de transición busca garantizar que se tiene un producto preparado para su entrega al usuario.

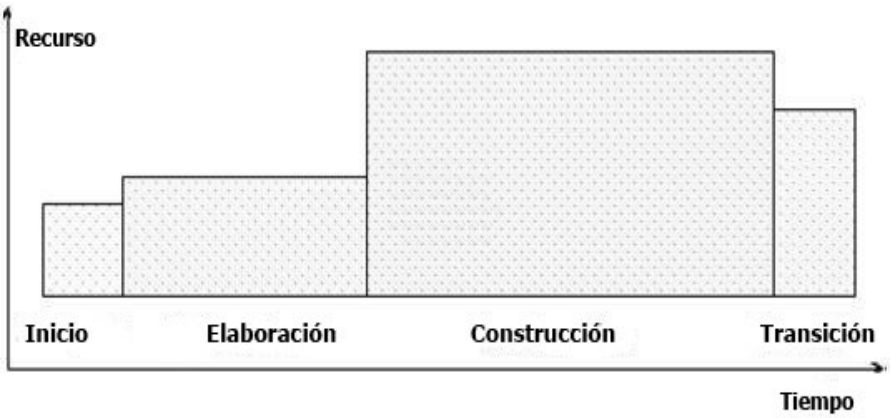


Figura 2.12: Fases del RUP

Fuente: (LLC, 2014)

2.1.17.3.4 Ciclo de Vida

El ciclo de vida RUP es una implementación del Desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades.

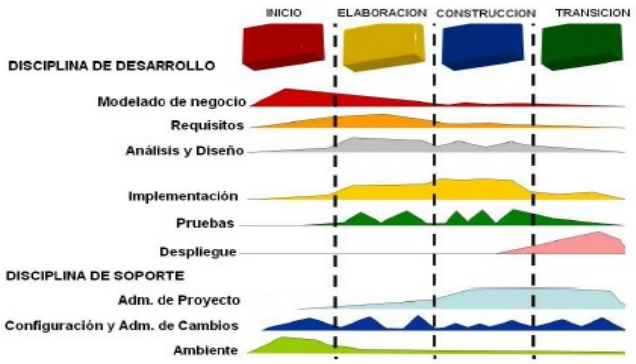


Figura 2.13: Ciclo de Vida RUP

Fuente: Tomado de **(Diaz Flores, 1996)**

2.1.17.3.5 Flujos de trabajo:

- Modelado de negocio
- Requisitos
- Análisis y diseño
- Implementación
- Pruebas

2.1.17.3.7 Artefactos

RUP en cada una de sus fases (pertenecientes a la estructura estática) realiza una serie de artefactos que sirven para comprender mejor tanto el análisis como el diseño del sistema.

Estos artefactos (entre otros) son los siguientes:
Inicio:

- Documento Visión
- Especificación de Requerimientos

Elaboración:

- Diagramas de caso de uso

Construcción:

- Documento Arquitectura que trabaja con las siguientes vistas:

Vista Lógica:

- Diagrama de clases
- Modelo E-R (Si el sistema así lo requiere)

Vista de Implementación:

- Diagrama de Secuencia
- Diagrama de estados
- Diagrama de Colaboración

Vista Conceptual

- Modelo de dominio

Vista Física

- Mapa de comportamiento a nivel de hardware.

CAPÍTULO III

3.1 ANÁLISIS DE LA IMPLEMENTACIÓN DEL SISTEMA WEB

3.1.1 ELECCIÓN DE HERRAMIENTAS DE DESARROLLO Y GESTOR DE BASE DE DATOS.

Para realizar el estudio de las herramientas de desarrollo de software, se evalúan los 5 elementos de software requeridos para la construcción de aplicaciones web:

- Plataforma Operativa.
- Servidor de Aplicaciones.
- Lenguaje de Desarrollo.
- Estándares de Desarrollo.
- Base de Datos.

Una vez cumplido el proceso anterior, se analiza las características técnicas que se tomarán en cuenta en la elaboración del sistema, comparando las alternativas del uso del software (libre o propietario) que son las siguientes:

- Requerimientos de Hardware y Software tanto para el servidor como para el cliente.
- Costos operativos de tecnología.
- Compatibilidad con la plataforma tecnológica existente o proyectada.
- Estabilidad, fiabilidad y facilidad de uso.
- Seguridad.
- Documentación.
- Existencia de implementaciones de software de éxito comprobadas en el ámbito local y nacional.
- Disponibilidad de las actualizaciones del software.
- Los costos de las herramientas a utilizar.

Se establece utilizar las siguientes tecnologías:

- JSF 2.0 para la capa de la Vista. Interfaz de usuario (Java EE 6).
- Componentes EJB 3.0 para encapsular la lógica del negocio.
- Drools 5.5 motor de reglas de producción para administrar las reglas de negocio conjuntamente con EJB 3.0.
- API Persistencia Java (JPA) para la capa de persistencia.
- Servidor de Aplicaciones JBOSS-AS 7.1. contenedor de componentes.
- Base de Datos POSTGRES 9.1 para almacenar la información.
- El sistema operativo del servidor quedará abierto entre Linux y Windows.

- Navegador Mozilla Firefox 26.0, Internet Explorer 8.0.

A continuación, se explica brevemente cada uno de los anteriores elementos.

3.1.1.1 Framework de Desarrollo JSF 2.0.

Desarrollado a través del Proceso de la Comunidad Java (Java Community Process) bajo la especificación JSR - 314, la tecnología Java Server Faces establece el estándar para crear interfaces de usuario del lado de servidor. Diseñado para ser flexible, la tecnología JSF enriquece las Interfaces de usuario existentes.

JSF es un Framework web MVC el cual se enfoca en simplificar la construcción de interfaces de usuario para aplicaciones web y aseguran rehusar interfaces de usuario fáciles para implementar (Hopkins, 2014).

El objetivo por el cual se utilizará JSF 2 es para proporcionar la separación en capas de la arquitectura de la aplicación, en la cual la lógica de negocio se llevará a cabo por la integración del motor de reglas de producción para manejar la lógica del negocio.

La arquitectura JSF

Una aplicación JSF es similar a alguna otra aplicación basada en tecnologías Java web; Corre en un contenedor Java Servlet, y contiene (tutorialspoint.com, 2014, pág. 10):

- Los componentes JavaBeans como modelos conteniendo la funcionalidad específica en la aplicación y los datos
- Una biblioteca personalizada de etiqueta para representar manipuladores de evento y los validadores
- Una biblioteca personalizada de la etiqueta para suministrar componentes de Interfaz de Usuario
- Componentes de Interfaz de Usuario representados como objetos con estado en el servidor
- Las clases Helper del lado del servidor
- Los validadores, manejadores de eventos y de navegación
- El archivo de configuración de recurso para configurar la aplicación.

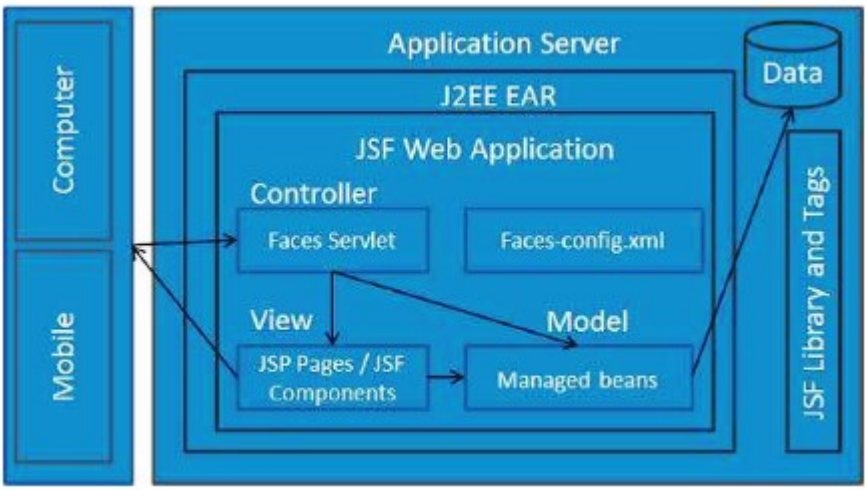


Figura 3.1: Arquitectura de JSF 2

Fuente: Tomado de **(tutorialspoint.com, 2014, pág. 10)**

3.1.1.2 Enterprise Java Beans (EJB).

Los componentes EJBs se encargan de encapsular la lógica de negocio de la aplicación, son componentes de software, y es un modelo de programación que permite construir aplicaciones Java mediante objetos ligeros (como POJO's). Por lo que son responsables de la seguridad, transaccionalidad, concurrencia, etc. El estándar EJB permite centrarse en el código de la lógica de negocio del problema que se quiere solucionar y deja el resto de responsabilidades al contenedor de aplicaciones donde se ejecutará la aplicación (Marco David, 2011). Los EJB necesitan estar dentro del contexto de un contener para ser administrados. Dicho contenedor tiene que estar dentro de un servidor web o de aplicaciones que cumple con la especificación de Java Enterprise Edition 6 que es la que se utilizará. Estos EJBs serán los encargados de la persistencia y transaccionalidad conjuntamente con JPA que se describirá a continuación.

3.1.1.3 Java Persistence API (JPA)

El JPA es una especificación de Sun Microsystems, hoy de propiedad de Oracle para la persistencia de objetos Java a cualquier base de datos relacional.

(Sánchez Juan José Meroño, 2009, pág. 71) Dice JPA es el estándar para gestionar persistencia de Java, definida como parte de EJB3 y que opera con POJO's (acrónimo de Plain Old *Java* Object). Se trata de corresponder las entidades de base de datos como objetos Java, o lo que es lo mismo, dotar de persistencia a los objetos del lenguaje Java.

En esta figura se muestra la interacción de los diversos componentes de la arquitectura de JPA:

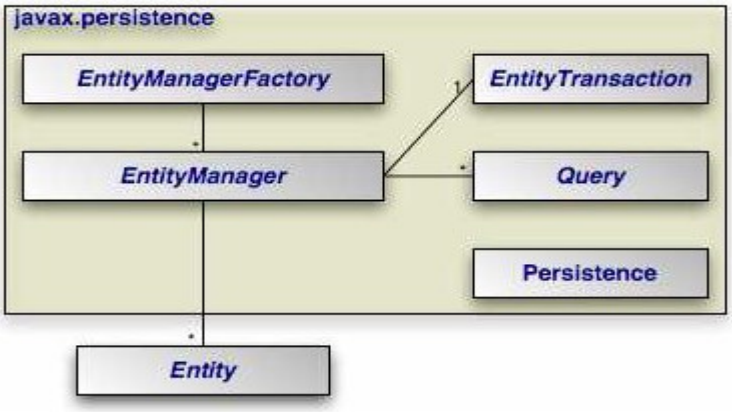


Figura 3.2: Arquitectura de JPA.
Fuente: Tomado de **(Sánchez Juan José Meroño, 2009, pág. 71)**

El principal componente de la arquitectura es el "EntityManager". Generalmente en un servidor de aplicaciones con contenedor de EJB; esta entidad será proporcionada y no es necesario recurrir al factory, pensado para entornos sin contenedor de EJB (por ejemplo en un entorno de test). La clase Persistence dispone de métodos estáticos que permiten obtener una instancia de EntityManagerFactory de forma independiente al proveedor de JPA y a partir del que se puede obtener el EntityManager necesario.

Mediante EntityManager se puede obtener objetos "Query" para cargar objetos persistidos con determinados criterios. JPA emplea el lenguaje de consulta JPQL ó SQL. También se puede realizar operaciones CRUD sobre los objetos persistentes "Entity".

Cada instancia de EntityManager tiene una relación uno a uno con una EntityTransaction, permitiendo el manejo de operaciones sobre objetos persistentes con la idea global de transacción, de forma que se ejecuten todas las operaciones o ninguna.

Por otro lado la unidad de persistencia es lo que aporta la información necesaria para enlazar con la base de datos relacional adecuada (Sánchez Juan José Meroño, 2009, pág. 71).

3.1.1.4 Servidor de Aplicaciones JBOSS.

Un servidor de aplicaciones se considera la base de un sistema distribuido ya que es una plataforma de Middleware(Es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos (Fundación Wikimedia I. , 2014)) para el desarrollo y despliegue de software basado en componentes. Proporciona servicios que soportan la ejecución y disponibilidad de las aplicaciones, tareas relacionadas con el mantenimiento de la seguridad y del estado, acceso a datos y persistencia entre otros.

JBOSS AS 7 es un proyecto de código abierto, basado en J2EE 6, e implementado 100% en Java.

Entre la características se puede mencionar la rapidez, ligero, núcleo modular, despliegue en caliente y en paralelo, administración elegante, administración de dominios entre otras; refiérase a la documentación oficial de JBoss Admin Guide (Braun Heiko, 2013)

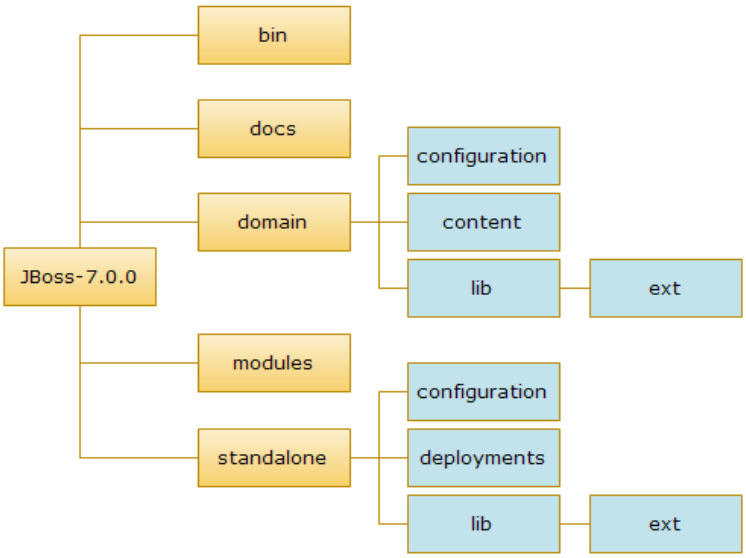


Figura 3.3: Sistema del archivo del servidor de aplicación.
Fuente: Tomado de **(Marchioni Francesco, 2011, pág. 19)**

3.1.1.5 Base de Datos PostgreSQL.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (Es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution (Fundación Wikimedia, 2013)) y con su código fuente disponible libremente. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. A continuación se tiene un gráfico que ilustra de manera general los componentes más importantes en un sistema PostgreSQL.

PostgreSQL está basado en una arquitectura cliente-servidor. El programa servidor se llama Postgres y entre los muchos programas cliente se tiene, por ejemplo, pgaccess (un cliente gráfico) y psql (un cliente en modo texto).

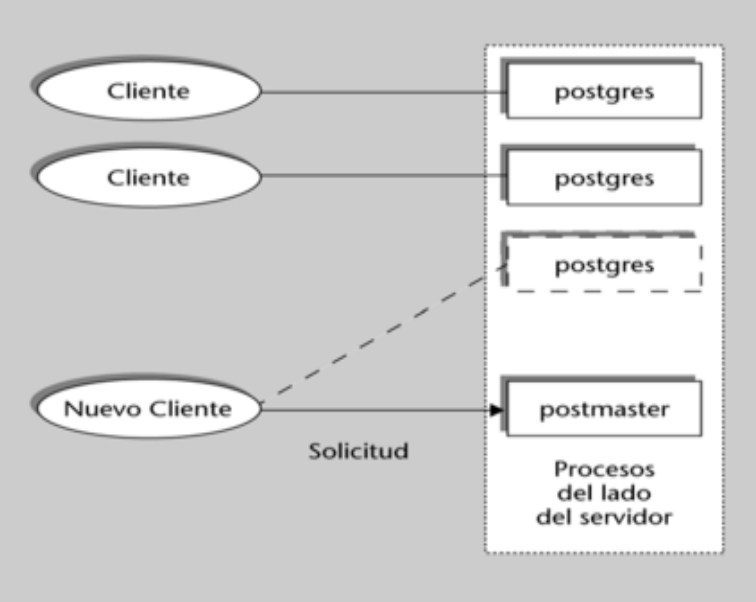


Figura 3.4: Descripción de arquitectura de PostgreSQL.
Fuente: Tomado de (Elías López, 2014)

Un proceso servidor Postgres puede atender exclusivamente a un solo cliente; es decir, hacen falta tantos procesos servidor Postgres como clientes haya. El proceso postmasteres el encargado de ejecutar un nuevo servidor para cada cliente que solicite una conexión.

Se llama sitio al equipo anfitrión (host) que almacena un conjunto de bases de datos PostgreSQL. En un sitio se ejecuta solamente un proceso postmaster y múltiples procesos Postgres. Los clientes pueden ejecutarse en el mismo sitio o en equipos remotos conectados por TCP/IP (Elías López, 2014).

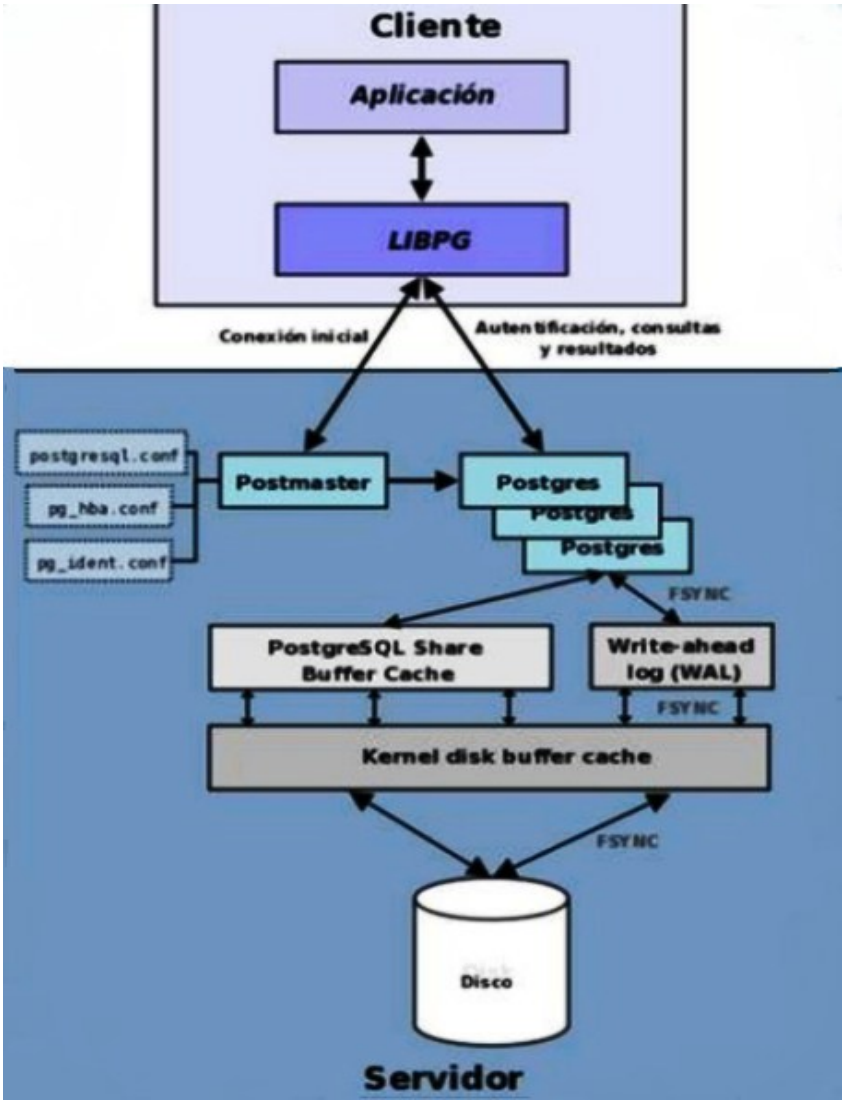


Figura 3.5: Arquitectura de PostgreSQL.
Fuente: Tomado de (Bryan Rodriguez, 2013)

3.1.1.6 El Lenguaje de Reglas DROOLs

En lo referente al lenguaje de reglas para el motor de reglas de producción Drools es una traducción propia y resumen del documento oficial de Drools 5.5 (The JBoss Drools team, págs. 125-215).

Drools tiene un lenguaje "nativo" de reglas. Este formato es muy ligero en términos de la puntuación, y soporta lenguajes naturales. Un archivo de reglas es un archivo con una extensión drl. En un archivo DRL puede tener múltiples reglas, consultas y funciones, así como también algunas declaraciones de recursos como imports, globals y atributos que son asignados y usados por las reglas y consultas. Un archivo DRL es simplemente un archivo del texto. La estructura global de un archivo de reglas es:

```
package package-name
imports
globals
functions
queries
rules
```

La orden en la cual los elementos están declarados no es importante, excepto por el nombre del paquete que, estando declarado, debe ser el primer elemento en el archivo de reglas. Todos los elementos son optativos.

3.1.1.6.1 Que hace una regla

Una regla tiene la siguiente estructura aproximada:

```
rule " Nombre "
atributos
when
       LHS (Lado Izquierdo)
then
       RHS (Lado Derecho)
end
```

Es muy simple. En su mayor parte la puntuación no es necesaria, aun las comillas dobles para "los nombres" son optativas, como son las nueva línea (newlines). Los atributos son indicios simples (siempre optativo) de cómo debería comportarse la regla.

LHS es la parte condicional de la regla, lo cual sigue una cierta sintaxis que se describirá a continuación.

RHS es básicamente un bloque que permite agregar código semántico específico dialectal (Relativo al dialecto: Se aplica a la palabra, frase o modo de expresión que es propio de un dialecto (Larousse, 2014)) para ser ejecutado. Es importante notar que el espacio blanco no es importante.

3.1.1.6.2 Palabras Clave (Keywords)

Drools 5 introduce el concepto de palabras clave (keywords) hard(duro) y soft(suave). Literalmente palabras clave duras(Hard keywords) son reservadas, no se puede usar cualquier palabra clave hard cuando se nombra a los objetos del dominio, las propiedades, los métodos, las funciones y otros elementos que se usan en el texto de la regla.

Aquí está la lista de hard keywords que se debe evitar escribir como identificadores al escribir reglas:

lock-on-active, date-effective, date-expires, no-loop, auto-focus, activation-group, agenda-group, ruleflow-group, entry-point, duration,package, import, dialect, salience, enabled, attributes, rule, extend, when, then, template, query, declare, function, global, eval, not, in, or, and, exists, forall, accumulate, collect, from, action, reverse, result, end, over, init

Se puede hacer estas palabras (hard y soft) como parte de un nombre con el método "Camel Case", es decir si se usa nombres compuestos a partir de la segunda palabra la primera letra es con mayúscula (nombreVariable). Si se necesita utilizar como identificadores en lugar de keywords, el lenguaje DRL le provee la habilidad de escapar hard keywords en texto de regla. Para escape una palabra, simplemente inclúyalo en acentos grave, como esto:

```
Persona( 'true' == "yes" )
En un método Persona.isTrue()
```

3.1.1.6.3 Comentarios

Los comentarios son secciones de texto que son ignoradas por el motor de reglas cuando son encontrados. Existen dos tipos de comentarios:

El comentario en una línea "/".

Los multi-comentarios son usados para comentar bloques de texto.

Ejemplo.

```
rule "Ejemplo de comentarios"
when
       /* Este es un comentario multi-línea
       en el lado izquierdo de la regla*/
       eval( true )
then
       // y este es un ejemplo en una sola línea
end
```

3.1.1.6.4 El paquete

Un paquete es una colección de reglas y otras construcciones relacionadas, como imports y globals. Un paquete representa a un namespace, lo cual es único para un agrupamiento dado de reglas. El nombre del paquete mismo es el namespace, y no está relacionado con archivos o carpetas dentro de cualquier forma.

Note que un paquete debe tener un namespace y debe ser declarado dentro de las convenciones estándar Java para paquete.

3.1.1.6.5 Import

Las declaraciones import trabajan como declaraciones import en Java. Se necesita especificar las rutas completamente válidas y los nombres de tipo para cualquier objeto que se quiere usar en las reglas. Drools automáticamente importa clases del paquete Java del mismo nombre, y también del paquete java.lang.

3.1.1.6.6 Global

Con una global se define variables globales. Se usan para hacer disponible objetos de aplicación en las reglas. Se usan para proveer datos o servicios que las reglas usan, especialmente los servicios de aplicación usados en las consecuencias de reglas, y devolver datos de las reglas, como Logs (Un log es un registro oficial de eventos durante un rango de tiempo en particular. También conocido como bitácora (Fundación Wikimedia, 2013)) o los valores agregados en las consecuencias de reglas, o para las reglas que interactúan con la aplicación, hacer Callbacks (Una devolución de llamada o retro llamada (Fundación Wikimedia, 2013)).

Las variables globals no son introducidas en la Memoria de Trabajo, y por consiguiente una global nunca debería ser usado para establecer condiciones en las reglas excepto cuándo tiene un valor constante inmutable. El motor no puede ser notificado acerca de los valores cambiados de globals y no rastrea sus cambios. El uso incorrecto de globals en las restricciones puede producir resultados sorprendentes - sorprendiendo de mala manera.

Si múltiples paquetes declaran globals con el mismo identificador deben ser del mismo tipo y todos ellos establecerán referencias para el mismo valor global.

Para usar globals se debe: Declarar la variable global en su archivo de reglas y usarla en las reglas. Ejemplo:

```
global java.util.List myGlobalList;
rule "Usando una variable global"
when
       eval( true )
then
       myGlobalList.add( "Hello World" );
end
```

Establecer el valor global en la Memoria de Trabajo. Es una mejor práctica anteponer todos los valores globales afirmando cualquier hecho para la Memoria de Trabajo.

```
List list = new ArrayList();
WorkingMemory wm = rulebase.newStatefulSession();
wm.setGlobal( "myGlobalList", list );
```

3.1.1.6.7 Función (function)

Las funciones son una forma de introducir código semántico en un archivo de recursos de reglas, similar a las clases helper de Java. La ventaja principal de usar funciones en una regla es que puede mantener todo la lógica en un lugar, y se puede cambiar las funciones según se necesite. Las funciones son más útiles para invocar acciones en la parte de la consecuencia (then) de una regla, especialmente si esa acción particular es usada repetidas veces, quizá con sólo diferir parámetros para cada regla.

Una declaración de función:

```
function String hello(String name) {
             return "Hello "+name+"!";
}
```

Alternativamente, se puede usar métodos estáticos en una clase helper, por ejemplo Persona.saludo(). Drools soporta el uso de function imports, así es que todo lo que se necesitaría hacer es:

import function my.package.Foo.hello

Sin distinción de la forma, la función es definida o importada, se usa una función llamándola por su nombre, en la consecuencia o dentro de un bloque semántico de código. El ejemplo:

```
rule "using a static function"
when
       eval( true )
then
       System.out.println( hello( "Bob" ) );
end
```

3.1.1.6.8 Reglas

Una regla especifica cuando (when) una colección particular de condiciones ocurre, especificadas en el Lado Izquierdo(Left Hand Side)(LHS), entonces (then) ejecuta las consultas (querys) especificadas como una lista de acciones en el Lado Derecho (Right Hand Side) (RHS).

Una regla debe tener un nombre único dentro de su paquete de reglas. Si se define una regla dos veces en el mismo DRL produce un error al cargar. Si usted agrega un DRL que incluye un nombre de regla que ya existe en el paquete, este reemplaza la regla previa. Si un nombre de regla tiene espacios, entonces necesitará estar entre comillas dobles.

Los atributos - descritos de abajo - son optativos. Son mejor escrito línea por línea.

El LHS de la regla sigue a la palabra clave when (idealmente en una línea nueva), de modo semejante el RHS sigue a la palabra clave then (otra vez, idealmente en una nueva línea). La regla es terminada por la palabra clave end. Las reglas no pueden estar anidadas.

Sintaxis general de una regla

```
rule "<nombre>"
<atributos>*
when
       <elementos condicionales>*
then
       <acciones>*
end
```

```
rule "Aprobar si no es rechazado"
salience -100
agenda-group "approval"
when
	not Rechazado()
	p : Policia(aprovado == false, policiaEstado:estado)
	exists Conductor(edad > 25)
	Proceso(estado == policiaEstado)
then
	log("Aprobado: Debido a no haber objeciones.");
	p.setApproved(true);
end
```

3.1.1.6.9 Atributos de Regla

Los atributos de regla proveen una forma declarativa para influenciar el comportamiento de la regla. Algunos son muy simples, mientras los otros son parte de subsistemas complicados como ruleflow. Para obtener el máximo provecho de Drools se debe tener una comprensión correcta de cada atributo.

- **no-loop:** Valor por defecto: False, Tipo: Boolean.
  Cuando la consecuencia de una regla modifica un hecho puede causar que la regla se active otra vez, causando un lazo infinito. Estableciendo a no-loop a verdadero se saltará la creación de otra activación para la regla con la colección actual de hechos.
- **Ruleflow-group:** Valor por defecto: N/A, Tipo: String.
  Ruleflow es una característica de Drools que le permite ejercer control sobre los disparos de reglas. Las reglas que son ensambladas por el mismo identificador de ruleflow-group se disparan sólo cuando su grupo está en actividad.
- **lock-on-active:** Valor por defecto: False, Tipo: Boolean.
  Cada vez que un ruleflow-group se activa o una agenda-group recibe el foco, cualquier regla en ese grupo que tiene lock-on-active estableció a verdadero no será activado más; Sin distinción del origen de la actualización, la activación de una regla que empareja es descartada. Ésta es una versión más fuerte de no-loop, porque el cambio ahora podría ser causado no sólo por la regla misma. Es ideal para reglas de cálculo donde se tiene un número de reglas que modifican un hecho y no se quiere que cualquier regla re-emparejada se dispare otra vez. Sólo cuando el ruleflow-group ya no es activo o la agenda-group pierde el foco esas reglas con lock-on-active colocado a verdadero se

vuelve elegible otra vez para sus activaciones ser colocados encima del agenda.

- **salience (Prioridad):** Valor por defecto: 0, Tipo: Integer. Cada regla tiene un número de prioridad por defecto es cero y puede ser negativas o positivas. Salience es una forma de prioridad donde las reglas con valores superiores están en la parte superior ordenadas en la cola de Activación.
  Drools también soportes salience dinámico donde se puede usar una expresión involucrando variables de salto.
  Prioridad dinámica (Dynamic Salience)

```
rule " Dispare en la orden de la fila 1,2,.."
salience( -$rank )
when
       Element( $rank : rank,... )
then
       ...
end
```

- **agenda-group:** Valor por defecto: MAIN, Tipo: String
  Agenda-group dejan al usuario dividir en partes a la Agenda proporcionando más control de ejecución. Sólo las reglas en el Agenda-group que ha adquirido el foco son admitidas para dispararse.
- **auto-focus:** Valor por defecto: Falso, Tipo: Boolean
  Cuando una regla es activada donde el valor auto-focus es cierto y el Agenda-group de la regla no tiene foco aún, luego recibe foco, dejando la regla potencialmente dispararse.
- **activation-group:** Valor por defecto: N/A, Tipo: String
  Las reglas que le pertenecen a este mismo activation-group, cada una identificada por este atributo valor string, sólo se disparara exclusivamente. En otras palabras, la primera regla en un activation-group a disparar cancelará las activaciones de otras reglas, impidiéndolas disparar.
- **date-effective:** Valor por defecto: N/A, Tipo: String
  Contiene una fecha y una definición de tiempo. Una regla sólo puede activarse si la fecha actual y el tiempo va tras de atributo efectivo en la fecha.
- **date- expires:** Valor por defecto: N/A, Tipo: String

Conteniendo una fecha y una definición de tiempo. Una regla no puede activarse si la fecha actual y el tiempo es después de que la fecha expira el atributo.

- **duration:** Valor por defecto: No hay valor por defecto, Tipo: Long
  La duración manda que la regla se disparará después de una duración especificada, si es todavía cierta.

3.1.1.6.10 Sintaxis del Lado Izquierdo de una regla

El Lado Izquierdo (Left Hand Side) es un nombre común para la parte condicional de la regla when (LHS). Consiste de cero o más Elementos Condicionales. Si el LHS es vacío, será considerado como un elemento de condición que es siempre cierto y serán activadas una vez, cuándo una sesión nueva de Memoria de Trabajo es creada.

Los elementos condicionales trabajan en uno o más patrones (los cuáles son descritos más abajo). El elemento condicional más común es and. Por eso está implícito cuando usted tiene patrones múltiples en el LHS de una regla que no están conectados en cualquier forma: Ejemplo de And implícito

```
rule "2 patrones desconectados"
when
       Patron1()
       Patron2()
then
       ... // actions
end
// La regla anterior es internamente reescrita como:
rule "2 patrones conectados con And"
when
       Pattern1()and Pattern2()
then
       ... // actions
end
```

3.1.1.6.11 Patrón

Un elemento del patrón es el Elemento Condicional más importante. Potencialmente puede emparejar en cada hecho que es insertado en la Memoria de Trabajo. Un patrón contiene cero o más restricciones y tiene un patrón opcional obligatorio.

En su forma más simple, sin restricciones, un patrón empareja en contra de un hecho del tipo (type) dado. El tipo no necesita ser la clase real de algún objeto hecho (fact). El

patrón puede referirse a las superclases o aun las interfaces, por consiguiente potencialmente emparejara a los hechos de muchas clases diferentes.

```
Object() // Empareja todos los objetos en la Memoria de Trabajo
```

Dentro del patrón que el paréntesis agrupa es dónde toda la acción ocurre: Define las restricciones para ese patrón. Por ejemplo, con una restricción relacionada con la edad:

```
Persona( edad == 100 )
```

Patrón vinculando permite referir al objeto emparejado, use un patrón vinculando como variable $p.

Ejemplo: Patrón con una variable vinculada

```
rule ...
when
       $p : Persona()
then
       System.out.println( "Persona " + $p );
end
```

El símbolo dólar ($) es solo una convención; Este puede ser usada en reglas complejas donde ayuda a distinguir entre variables y campos, pero no es obligatorio.

3.1.1.6.12 Restricción

Una restricción (Constraint) es parte de un patrón y es una expresión que retorna true or false. Este ejemplo tiene una restricción que afirma.

```
Persona( edad == 100 )
5 es menor que 6:
Persona( 5 < 6 ) // Como una restricción debería ser usadas en un patrón real
```

El acceso a las propiedades en Java Beans (Fundación Wikimedia, Bean, 2014) puede ser directamente. Una propiedad bean queda al descubierto utilizando un estándar Java bean getter: Un método getMyProperty () (o isMyProperty () para un boolean primitivo) que no tome argumentos y retorne algo.

Puede usarse cualquier expresión Java que devuelve un boolean como una restricción dentro de los paréntesis de un patrón. Las expresiones Java pueden ser mixtas con otras mejoras de expresiones, como el acceso de la propiedad:

```
Persona( edad == 50 )
```

Esto es posible cambiar la prioridad de evaluación usando paréntesis como en cualquier lógica y expresión matemáticas:

```
Persona( edad > 100 && ( edad % 10 == 0 ) )
```

El separador Coma (,) AND es usado para separar grupos de restricciones. Este tiene implícitamente un conectivo semántico AND.

```
// Persona mayores de 50 años y pesan más de 80 kg
Persona( edad > 50, peso > 80 )
```

El operador coma no puede ser embebido en una expresión de restricción compuesta, tal como paréntesis

```
// Persona mayores de 50 años y pesan más de 80 kg
Persona( edad > 50, peso > 80 )
Persona( ( edad > 50, peso > 80 ) || estatura > 2 ) // Esto compila error
// Use esto
Persona( ( edad > 50 && peso > 80 ) || estatura > 2 )
```

Variables vinculadas (Binding variables) Una propiedad puede ser vinculada a una variable:

```
// 2 personas de la misma edad
Persona( $primeraEdad : edad ) // vinculación
Persona( edad == $ primeraEdad) // expresión de restricción
```

El prefijo del símbolo dólar ($) es solo convención; Este puede ser de ayuda en reglas complejas que ayudan a distinguir entre variable y campos.

```
Persona( $edad : edad * 2 < 100 )
// Recomendado (separar vinculaciones y expresiones de restricción)
Persona( edad * 2 < 100, $edad : edad )
También permite colocar más que una restricción o campo usando los
conectivos de restricción && o ||.
```

Operadores especiales DRL

La coerción para el valor correcto para el evaluador y el campo será pretendida.

Los operadores son: <, <=, >, >=

Estos operadores pueden ser usados en propiedades con orden natural. Por ejemplo, para campos tipo Date, < significa menor para campos String, esto significa alfabéticamente bajo.

```
Persona( primerNombre < $otroPrimerNombre )
Persona( fechaCumpleanios < $otraFechaCumpleaños )
Solo aplica en propiedades comparables.
```

Operador de emparejamiento (matches) emparejar un campo en contra de cualquier expresión regular Java. Típicamente la expresión regular es un String literal, pero las variables que resuelva una expresión regular son también permitidas.

```
Cheese( type matches "(Buffalo)?\\S*Mozarella" )
```

Operador contener (contains) es usado para chequear si un campo de una Collection o un Array contienen un valor especifico.

```
Ejemplo: Contains con colecciones tipo Collections
Persona( nombre contains "Juan" ) // contains con literal String
Persona ( nombre contains $var ) // contains con una variable
Solo aplica en propiedades Collection.
```

Operador miembro de (memberOf) es usado para chequear si un campo es un miembro de una Collection o Array; que la collection debería ser una variable.

Ejemplo de restricción literal con una Collections.

```
Persona( edad memberOf $campoEdad )
```

Operadores in y not in (la restricción compuesta de valor)

La restricción compuesta de valor es usada donde hay más de un valor posible para emparejar.

Actualmente sólo los evaluadores in y not in soporta esto. El segundo operando de este operador debe ser una lista de valores separados por comas, encerrados entre paréntesis.

A los valores son dados como variables, literales, valores de retorno o identificadores calificados. Ambos evaluadores son realmente sintácticos se granula, internamente reescrito como una lista de restricciones múltiples utilizando a los operadores: != y ==

Operador de precedencia

Los operadores son evaluados en esta precedencia:

```
Person( $cheese : favouriteCheese )
Cheese( type in ( "stilton", "cheddar", $cheese ) )
```

Operator type	Operators	Notes
(nested) property access	.	Not normal Java semantics
List/Map access	[ ]	Not normal Java semantics
constraint binding	:	Not normal Java semantics
multiplicative	* / %	
additive	+ -	
shift	<< >> >>>	
relational	< > <= >= instanceof	
equality	== !=	Does not use normal Java (*not*) *same* semantics: uses (*not*) *equals* semantics instead.
non-short circuiting AND	&	
non-short circuiting exclusive OR	^	
non-short circuiting inclusive OR	|	

Operator type	Operators	Notes
logical AND	&&	
logical OR	||	
ternary	? :	
Comma separated AND	,	Not normal Java semantics

Figura 3.6: Operadores de Precedencia.

Fuente: Tomado de Drools (The JBoss Drools team, págs. 172-173)

3.1.1.6.13 Elementos Condicionales básicos

El elemento Condicional 'and' es usada para agrupar otros elementos condicionales en una conjunción lógica. Drools soporta ambos prefix and y infix and.

```
//infixOr
Queso( tipoQueso : tipo ) and Persona( QuesoFavorito == tipoQueso)

Agrupamiento explicito con paréntesis es también soportado:

//infixOr con agrupamiento
(Queso (tipoQueso: tipo) and ( Persona(QuesoFavorito == tipoQueso) and
                                    Person(QuesoFavorito == tipoQueso))
```

El elemento condicional 'or' es usado para agrupar otros elementos condicionales en la lógica disyuntiva.

Drools soporta ambos prefix or y infix or.

```
//infixOr
Queso( tipoQueso : tipo ) or Persona( QuesoFavorito == tipoQueso)

Agrupamiento explicito con paréntesis es también soportado:

//infixOr con agrupamiento
(Queso (tipoQueso: tipo) or ( Persona(QuesoFavorito == tipoQueso) and
                                    Person(QuesoFavorito == tipoQueso))
```

El elemento condicional 'not' es el cuantificador non-existencial de lógica de primera orden y chequea la inexistencia de algo en la Memoria de Trabajo. Piense acerca de not como querer decir "no debe haber ninguno de .."

La palabra clave not puede ser seguido por paréntesis alrededor de los elementos condicionales a los que se aplica. En el caso de un solo patrón se puede omitir los paréntesis.

```
not Persona()
```

El elemento condicional 'exists' es el cuantificador existencial de lógica de primer orden y revisa en busca de la existencia de algo en la Memoria de Trabajo. Si usa exists con un patrón, la regla sólo se activará a lo sumo una vez, a pesar de cuánto datos hay en Memoria de Trabajo que empareja la condición adentro del patrón exists. Desde entonces sólo la existencia tiene importancia, ninguna de las vinculaciones serán establecidas. La palabra clave exists debe ser seguido por paréntesis alrededor de los elementos de condición a los que se aplica. En el caso más simple de un patrón puede omitir los paréntesis.

```
exists Persona().
```

3.1.1.6.14 Elementos condicionales avanzados

El elemento condicional forall completa el soporte de Lógica de primer orden en Drools. El elemento condicional forall evalúa verdadero cuando todos los hechos que corresponden al primer patrón corresponden a todos los demás patrones.

```
rule "Todos los empleados tienen programas de atención de salud y dental"
when
       forall( $emp : Employee()
       HealthCare( employee == $emp )
       DentalCare( employee == $emp ))
then
       // Todos los empleados tienen atención de salud y dental
end
```

El elemento condicional from permite a los usuarios especificar un recurso arbitrario para los datos para ser emparejados por patrones LHS. Esto permite el motor no razonar sobre datos en la Memoria de Trabajo. El recurso de datos podría ser un sub-campo en una variable comprometida o los resultados de una llamada de método. Es una construcción poderosa que permite de la integración con otros componentes de aplicación.

```
rule "Validar código zip"
when
       Persona( $direccionPersona : direccion )
       Address(direccion == "23920W") from $direccionPersona
Then
              // zip code is ok
End
```

Se debe temer cuidado, sin embargo, al usar a from, especialmente en conjunción con el atributo de regla lock-on-active como puede producir resultados inesperados.

El elemento condicional collect permite razonar a las reglas sobre una colección de objetos obtenidos de los recursos dados o de la Memoria de Trabajo. En términos de la Lógica de primer orden este es un cualificador de cardinalidad. Un ejemplo simple:

```
import java.util.ArrayList
rule " Incremente prioridad si el sistema tiene más que 3 alarmas pendientes "
when
       $sistema : Sistema()
       $alarmas : ArrayList( tamanio >= 3 )
       from collect( Alarmas( sistema == $sistema, estado == 'pendiente' ) )
then
       // Incrementa la prioridad porque el sistema ha tenido más de 3 alarmas
       //pendientes
end
```

3.1.1.6.15 Sintaxis del Lado Derecho de una regla

El Lado Derecho (Right Hand Side) es un nombre común para la consecuencia o la acción la parte de la regla then (RHS); Esta parte debería contener una lista de acciones para ser ejecutada. Es mala práctica usar código imperativo o condicional en el RHS de una regla; Como una regla debería ser atómica en naturaleza.

La parte RHS de una regla también debería ser mantenida pequeña, así manteniéndolo declarativa y legible. Si encuentra necesario código imperativo o condicional en el RHS, entonces tal vez debiese estudiar en sus partes esa regla en las reglas múltiples. El propósito principal del RHS es insertar, retractar modificar datos en la Memoria de Trabajo. Para ayudar con eso hay algunos métodos de conveniencia que se puede usar para modificar la Memoria de Trabajo; Sin tener que primero establecer referencias para una instancia en la Memoria de Trabajo.

update(object, handle), update (object): Le dirá al motor que un objeto ha sido cambiado (uno que ha estado obligado con algo en el LHS) y las reglas pueden necesitar ser reconsiderados.

insert(new Something()): Colocara a un nuevo objeto de su creación en la Memoria de Trabajo.

insertLogical(new Something()): Es similar a insert, pero el objeto será automáticamente retractado cuando no hay más hechos para soportar la verdad de la regla automáticamente disparada regla.

retract(handle): Remueve un objeto de la Memoria de Trabajo.

La declaración modify() es una extensión de lenguaje le provee una metodología estructurada a que el hecho se actualiza. Combina la operación de actualización con un número de llamadas setter para cambiar los campos del objeto. Éste es el esquema de sintaxis para la declaración modify:

```
modify ( <fact-expression> ) {
       <expression> [ , <expression> ]*
}
```

Dentro del paréntesis la <fact-expression> debe producir una referencia del objeto hecho. La lista de expresiones en el bloque debería constar de llamadas a los métodos setter para el objeto dado, para ser escrita sin la referencia usual del objeto, lo cual es automáticamente pretendido por el compilador. El ejemplo ilustra una modificación simple de hecho.

```
Ejemplo de una declaración de modify
rule "modify stilton"
when
       $stilton : Cheese(type == "stilton")
then
       modify( $stilton ){ setPrice( 20 ),setAge( "overripe" )}
end
```

Query Es una averiguación es una forma simple para buscar hechos en la Memoria de Trabajo que corresponden a las condiciones afirmadas.

Por consiguiente, contiene sólo la estructura del LHS de una regla, de manera que no se especifica ni 'when' ni 'then'. Una averiguación tiene una colección optativa de parámetros, cada uno del cual puede ser optativamente escrito. Si el tipo no es dado, el tipo Object es asumido. El motor intentará forzar los valores según se necesite. Los nombres de averiguación son globales para la Base de Conocimiento; Así es que no le añada a las averiguaciones el mismo nombre de los paquetes para la misma Base de Reglas.

```
query "people over the age of 30"
       person : Person( age > 30 )
end

QueryResults results = ksession.getQueryResults( "people over the age of 30" );
System.out.println( "we have " + results.size() + " people over the age of 30" );
System.out.println( "These people are are over 30:" );
for ( QueryResultsRow row : results ) {
       Person person = ( Person ) row.get( "person" );
       System.out.println( person.getName() + "\n" );
}
```

3.1.2 ANÁLISIS DE SUBMÓDULOS DEL SISTEMAS

Una vez definido la tecnología de desarrollo, se analizarán detenidamente los módulos a desarrollar en la Tienda Virtual.

A continuación, se muestra un diagrama generalizado de la Tienda Virtual.

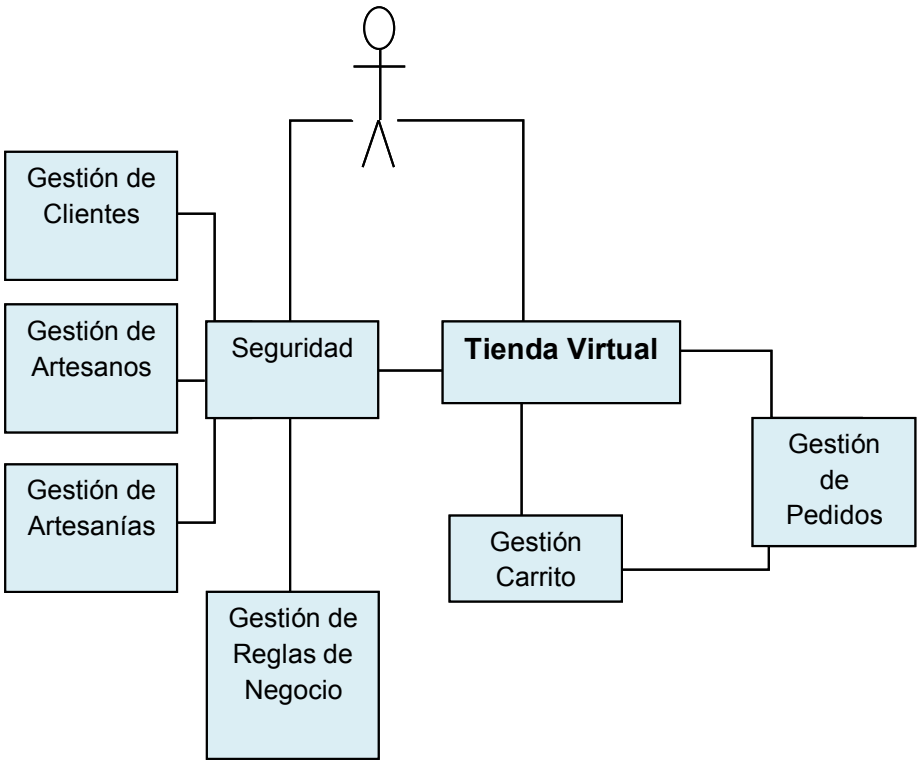


Figura 3.7: Análisis de los Módulos de la Tienda Virtual.
Fuente: Propia.

El software a construir ha de cumplir con los siguientes requisitos funcionales.

1. Permitir consultar el catálogo de artesanías.
2. Permitir buscar una artesanía en el sistema por cualquiera de los datos descriptivos de la artesanía.
3. Permitir ver el detalle de una determinada artesanía del catálogo.
4. Permitir consultar las ofertas disponibles de las artesanías.
5. Los usuarios han de estar registrados previamente en el sistema para poder realizar compras de las artesanías.
6. El usuario una vez autenticado en el sistema, se le asignará un "carrito de compra" donde se irán acumulando las compras realizadas.
7. El usuario podrá eliminar compras del carrito, incrementar la cantidad de cada artesanía a comprar.

8. Existirá un usuario especial con perfil de administrador que podrá realizar operaciones de mantenimiento del sistema.
9. Gestión de artesanías: Crear, actualizar, eliminar, cambiar el estado de la oferta de una artesanía, asignarle un artesano.
10. Gestión de artesanos: Crear, actualizar, eliminar, cambiar el estado del artesano, asignar las artesanías elaboradas por el artesano.
11. Gestión de usuarios: Crear, actualizar, dar de baja a un usuario.
12. Gestión de pedidos: Actualizar, cancelar un pedido.
13. El administrador podrá poner en oferta Artesanías, pudiendo indicar su precio de oferta y el periodo en el que permanecerá de oferta.
14. El administrador podrá consultar la lista de usuarios registrados y dispondrá de las opciones de modificar, eliminar, bloquea a usuarios y consultar las compras realizadas.
15. Un usuario que esté bloqueado no podrá acceder al sistema para realizar compras.
16. La gestión de las reglas de negocio será a través del sistema de administración de reglas de negocio GUVNOR de BRMS DROOLS.
17. Para poder poner en producción el aplicativo por la entidad interesada; Esta deberá tener la infraestructura informática necesaria.

CAPÍTULO IV

4.1 FASE DE INICIO

4.1.1 VISIÓN DEL PROYECTO

4.1.2 Propósito

Los desarrolladores de software enfrentan la necesidad de modificar las reglas de negocio de una aplicación de software ocultas en líneas de código fuente cada vez que el entorno cambia donde la aplicación es utilizada. Este tipo de problemas se da ya que el hombre se desenvuelve en un mundo dinámico y los sistemas de software tienen que adaptarse para el cambio.

El propósito de este capítulo es definir a alto nivel los requerimientos de la aplicación "IMPLEMENTAR UNA TIENDA VIRTUAL DE ARTESANÍAS PARA LA EMPRESA ENCHAPES ESPECIALES PRODUCTORES DE ACCESORIOS PARA DECORACIÓN DE MUEBLES".

Para la implementación de este aplicativo se utilizará la metodología de reglas de negocio y el motor de reglas de producción DROOLS el cual es un sistema de reglas de producción basado en el algoritmo RETE OO (Fundación Wikimedia, Algoritmo Rete, 2014), el cual cumple la implementación de la especificación JSR-94 (estándar para motor de reglas de negocio y framework (Marco de trabajo) de empresa para construcción, mantenimiento, y el refuerzo de políticas de empresa en una organización, aplicación o servicio (Fundación Wikimedia, Drools, 2014)) que define un conjunto de funciones y procedimientos que debe ofrecer un motor de reglas en Java. Dando así un modelo automatizado capaz de modificar las reglas de negocio de una empresa, facilitando la disponibilidad de las reglas para su modificación, exposición, rastreo que satisfagan los requerimientos de la empresa.

El objetivo de este capítulo es recoger, analizar y definir las necesidades de alto nivel y características de la tienda virtual para la empresa ENCHAPES ESPECIALES productores de accesorios para la decoración de muebles la cual será ejecutado en entorno web y manipulará la información de clientes, artesanías, artesanos, pedidos, y las restricciones necesarias para el correcto funcionamiento.

La tienda virtual contribuirá a la optimización de recursos tanto materiales como humanos de la empresa "ENCHAPES ESPECIALES", aumentando la productividad y las ventas de artesanías que elabora la empresa.

4.1.3 Alcance

El documento Visión contiene los módulos que forman parte de la Implementación de la Tienda Virtual de artesanías para la empresa ENCHAPES ESPECIALES productores de accesorios para decoración de muebles, así como los servicios que estarán disponibles para los usuarios.

El aplicativo Web tendrá las siguientes funcionalidades, para los diferentes usuarios:

- Usuario y clave para el ingreso al sistema
- Gestión de Clientes.
- Gestión de Artesanías.
- Gestión de Artesanos.
- Gestión del Carrito de Compras.
- Gestión de Pedidos.
- Gestión de las reglas de negocio a través del sistema de administración de reglas de negocio BRMS DROOLS.

4.1.4 Posicionamiento

Oportunidades de Negocio

Para la empresa ENCHAPES ESPECIALES tener facilidad para la venta y promoción de artesanías es muy importante por la cual se creará la tienda virtual que permitirá aumentar la producción y venta de artesanías satisfaciendo las necesidades de los propietarios y artesanos de dicha empresa.

La tienda virtual también permitirá acceder a sus utilidades a través de internet.

Definición del Problema

El problema de	**Los desarrolladores de software que enfrentan la necesidad de modificar las reglas de negocio de una aplicación de software ocultas en líneas de código fuente cada vez que el entorno cambia donde la aplicación es utilizada.**

afecta a	Todas las aplicaciones de software que su lógica de negocio cambia con demasiada frecuencia ya sea por razones estratégicas de la propia empresa o por las leyes del entorno que se desenvuelve.
El impacto asociado es	Para cuando es necesario modificar las reglas de negocio de la empresa en el aplicativo, los recursos materiales y humanos para implementación del nuevo requerimiento no satisfacen las diferentes necesidades de los involucrados.
Una solución exitosa debería	Permitir modificar las reglas de negocio sin necesidad de sacar de producción al aplicativo de software. Por lo que es necesario lograr la separación de las reglas de negocio del resto de sistema para que puedan ser rehusadas; Para poder rastrear las reglas de negocio y saber el motivo de su creación y el cambio que se ha producido en ellas. Por lo tanto las reglas de negocio serán exteriorizadas para los actores dentro del contexto empresarial. Y esto permite ubicarlas para que puedan ser cambiadas en cualquier momento. Satisfacer las necesidades del Propietario

Tabla 4.1: Definición del Problema

Fuente: Propia

Sentencia que define la posición del producto

Para	**La empresa de ENCHAPES ESPECIALES de manera especial el Propietario que es el encargado de la producción y administración de la misma, que le permitirá ingresar todas las reglas de negocio con la finalidad de vender y promocionar las artesanías en**

	madera.
Quienes	El Propietario que de acuerdo a la producción de artesanías no dispone de un aplicativo que les permita promocionar y vender. La forma tradicional de promocionar y vender las artesanías produce dificultad de hacer promociones y ofertar para fomentar las ventas y por ende la producción. Falta de software adecuado para saber que artesanías son más solicitadas por los clientes.
El nombre del Producto	Tienda Virtual "SanAntonioStore" para la empresa ENCHAPES ESPECIALES.
Que	Administra y almacena la información necesaria para obtener buenos resultados. Tiene registrada las artesanías conjuntamente con los artesanos que las realizan y sus clientes, determinando que artesanía es las más solicitadas. Podrá generar reportes para determinar que artesanías son las más vendidas, así como también que artesanos las realizan y en qué tiempo.
Debido a que	El Propietario de la empresa tiene registrada la información manualmente y en archivos Excel de una forma informal.
El producto	Permitirá administrar y gestionar la venta de artesanías a través de un proceso automático mediante las reglas de

	negocio que posee la empresa. Permitirá automatizar los diferentes procesos como son control de artesanos, artesanías, pedidos, carrito de compras.

Tabla 4.2: Definición de la posición del producto

Fuente: Propia

4.1.5 Descripción de los interesados y usuarios

Para proveer de manera efectiva los productos y servicios que se ajusten a las necesidades de los usuarios, es necesario involucrar a todos los participantes para el modelado de requerimientos. También es necesario identificar a los usuarios del sistema.

Esta sección muestra un perfil de los participantes y de los usuarios involucrados en el proyecto, así como de los problemas más importantes que estos perciben para enfocar la solución propuesta hacia ellos. No describen los requerimientos específicos ya que estos se capturan mediante otro artefacto. En lugar de estos se proporciona la justificación de porque estos requisitos son necesarios.

Resumen de los Interesados

Los interesados son todas aquellas personas directamente involucradas en la definición y alcance de este proyecto.

A continuación se presenta la lista de los interesados:

Nombre	Descripción	Responsabilidades
Franklin Germán Chamorro Chuquín	Desarrollador del proyecto de tesis.	Representa a todos los posibles usuarios del sistema. Seguimiento del desarrollo del proyecto. Responsable del análisis y diseño del proyecto. Aprueba requisitos y funcionalidades. Capacita en el manejo del aplicativo.
Lic. Carlos Chamorro Flores	Propietario de la empresa Representante funcional de la empresa y de todos los usuarios potenciales de la misma.	Administra y asigna posibles usuarios para el sistema Gestiona el correcto funcionamiento del sistema y procesa la implantación de mismo.
Ing. José Luís Rodríguez	Director del proyecto de tesis	Responsable de realizar un seguimiento del desarrollo del proyecto y aprobación de los requisitos y funcionalidades del sistema.

Tabla 4.3: Resumen de los Interesados

Fuente: Propia

Resumen de los Usuarios

Los usuarios son todas aquellas personas que proporcionan los requerimientos necesarios para desarrollar el proyecto. A continuación se presenta una lista de los usuarios:

Nombre	Descripción	Stakeholder
Administrador, Propietario.	Responsable de toda la información de la empresa: Gestión • Artesanías • Artesanos • Pedidos • Carrito de compras • Clientes Asignación • Artesanías a Artesanos Creación de reglas Reportes • Ventas de Artesanías • Ventas de Artesanías por Cliente	Lic. Carlos Chamorro Flores
Usuarios visitantes y compradores	Podrán revisar el catálogo de productos y serán potenciales compradores.	Usuarios registrados y visitantes

Tabla 4.4: Resumen de los Usuarios

Fuente: Propia

Entorno de Usuario

Los usuarios podrán acceder al sistema ya sea con una cuenta de usuario administrador validada por el sistema, como comprador o simplemente visitante. Trabajará en un entorno web que es compatible con cualquier navegador y funcionará en Internet. Cada usuario tendrá su propio entorno de trabajo según las funcionalidades asignadas al grupo de usuarios o roles que pertenezca.

Cuando se ingrese con el rol de administrador se podrá manipular y administrar el catálogo de clientes, artesanías, artesanos, pedidos. Las reglas de negocio serán gestionadas a través de GUVNOR (Es un repositorio centralizado de reglas de negocio que permite gestionarlas con una interface web (José Manuel Sánchez Suárez, 2013)) Drools; para realizar una compra el visitante necesitará registrarse con el rol de comprador.

Los usuarios no tendrán ninguna limitante para el correcto manejo del aplicativo ya que contarán con la respectiva capacitación que les permitirá manipular de manera ágil y rápida la tienda virtual.

La planificación va orientada a que después que se alimente al sistema con reglas de negocio válidas para la venta correcta el administrador podrá realizar cambios en las reglas de acuerdo a sus necesidades.

4.1.6 Perfiles de los Stakeholders

4.1.6.1 Responsable del Proyecto

Representante	**Franklin Germán Chamorro Chuquín**
Descripción	Responsable del Proyecto
Tipo	Analista de Sistemas
Responsabilidades	Gestionar el correcto desarrollo del proyecto en lo referente al diseño, construcción e implementación.
Criterios de éxito	Obtener un sistema de calidad que cumpla con los requerimientos funcionales establecidos.
Implicación	Jefe de proyecto (Project Manager)
Entregables	Documento Visión Glosario Lista de riesgos Resumen del modelo de casos de uso Especificaciones complementarias.
Comentarios	El perfeccionamiento constante con el desarrollo del sistema

Tabla 4.5: Perfil del Responsable del Proyecto

Fuente: **Propia**

4.1.6.2 Responsable Funcional

Representante	**Franklin Germán Chamorro Chuquín**
Descripción	Analista del proyecto
Tipo	Usuario
Responsabilidades	Acceso a todos los datos de la tienda virtual de usuarios funcionales.
Criterios de éxito	Sistema en funcionamiento.
Grado de participación	Activa

Tabla 4.6: Responsable Funcional del Proyecto

Fuente: **Propia**

4.1.6.3 Perfil de los usuarios

Administrador del sistema

Representante	**Lic. Carlos Chamorro Flores**
Descripción	Persona que administrará el sistema web.
Tipo	Propietario
Responsabilidades	Administrar funcionalmente el sistema
Criterio de éxito	Sistema en funcionamiento.
Grado de participación	Activa
Comentarios	Ninguno

Tabla 4.7: Perfil del Usuario: Administrador del Sistema

Fuente: **Propia**

Administrador funcional del sistema

Representante	**Franklin Germán Chamorro Chuquín**
Descripción	Desarrollador del Proyecto
Tipo	Analista de Sistemas
Responsabilidades	Responsable de tener actualizada la información de la institución, y proporcionar la información de los mismos de manera rápida. Responsable del ingreso de las reglas de negocio de la empresa.
Criterios de éxito	Sistema en funcionamiento Obtención rápida de la información cuando el visitante navega por la aplicación. Modificación de las reglas de negocio si necesidad de parar el sistema.
Grado de participación	Activa
Comentarios	Ninguno

Tabla 4.8: Perfil del Usuario: Administrador funcional del Sistema

Fuente: Propia

Usuario Normal del Sistema

Representante	**Franklin Germán Chamorro Chuquín**
Descripción	Usuario Normal
Tipo	Usuario visitante y registrado
Responsabilidades	Realiza la declaración de patente e IAT, a través del internet.
Criterio de éxito	Sistema con interfaz amigable y de fácil uso.
Grado de participación	Activa
Comentario	Ninguno

Tabla 4.9: Perfil del Usuario: Usuario normal del Sistema

Fuente: Propia

Necesidades de los interesados y usuarios

Necesidades	Prioridad	Inquietudes	Solución Actual	Solución Propuesta
Diseñar un sistema que facilite y administre el ingreso y la asignación de artesanos conjuntamente con las artesanías que elaboran.	Alta	El sistema debe registrar toda la información correctamente para facilitar la venta de artesanías	No Existe.	Diseñar, desarrollar e implementar la tienda virtual para la empresa de Enchapes Especiales
Implementar este sistema en el menor tiempo posible con el fin de ponerlo en producción con el fin de empezar a vender artesanías.	Alta	Tener el control de los datos de Artesanos, artesanías, pedidos	Actualmente la información de artesanías, artesanos se los maneja de manera manual y los datos se guardan en documentos de Microsoft office.	Registrar la información en un sistema unificado, para tener control sobre la misma.
Elaborar el sistema utilizando herramientas que faciliten el desarrollo.	Alta	Cumplir con todos los requerimientos de los usuarios.	Desarrollo del sistema con la aprobación de los involucrados.	Desarrollar con la ayuda de los usuarios.
La interfaz del sistema debe ser fácil de manejar, cumpliendo con todos los	Alta	Cumplir con todos los requerimientos de los usuarios.	Desarrollo del sistema con la aprobación de los involucrados.	Desarrollar un sistema con la ayuda de los involucrados.

requerimientos establecidos.				
Obtener reportes para consultar las ventas por artesanías, por artesanos	Alta	Poder manipular los resultados para obtener los diferentes tipos de reporte necesario para el propietario	No existe	Elaborar usando herramientas existentes para la creación de reportes.

Tabla 4.10: Necesidades de los Interesados y Usuario

Fuente: Propia

4.1.7 Vista General del Producto

El producto a desarrollarse es una tienda virtual para la promoción y venta de artesanías de empresa ENCHAPES ESPECIALES PRODUCTORES DE ACCESORIOS PARA DECORACIÓN DE MUEBLES.

4.1.7.1 Perspectiva del Proyecto

Figura 4.1: Perspectiva del Proyecto.
Fuente: Propia

4.1.7.2 Resumen de capacidades

A continuación se mostrará un resultado con los beneficios que obtendrán los usuarios a partir del producto:

Beneficios para el usuario	Características que lo soportan
El proceso de registro, promoción y venta será integral.	El sistema incluye desde el registro de artesanías y artesanos hasta la promoción y venta de artesanías a través del carrito de compras. El usuario administrador contará con una herramienta de administración de datos.
La empresa contará con una herramienta centralizada para promocionar y vender artesanías a los visitantes y usuarios registrados en la tienda virtual.	Se registrará los datos particulares de los clientes, artesanos, artesanías, pedidos.
Se tendrá alta disponibilidad.	El acceso a la información a través de la Web permitirá a los usuarios un acceso inmediato desde cualquier browser.
Facilidades para el análisis y toma de decisiones	Permitirá generar diversos tipos de reportes y funciones de consulta.

Tabla 4.11: Resumen de Capacidades

Fuente: Propia

4.1.7.3 Suposiciones y Dependencias

Se asume que el propietario de la empresa cuente con la infraestructura necesaria tanto para el acceso a la base de datos por parte de la aplicación, como para la publicación del sistema en el internet de modo que cada usuario podrá acceder al sistema a través de un navegador conectado a internet.

4.1.7.4 Costos y Precios

Detalles		Parcial (USD)	REAL (USD)
Hardware	Equipo desarrollo	1,200	1,200
	Equipo servidor Web y servidor de Base de Datos	1,800	0
	Equipos de prueba con navegador Web	800	0
Software	Data Base PostgreSQL	0	0
	Framework JSF	0	0
	Servidor de Aplicaciones Empresariales	0	0
	Suministros de oficina	400	400
	Libros, empastados	500	500
	Viáticos y Movilización	600	600
Subtotal		5300	2700
	10% Imprevistos	530	270
Total	**General**	5.830,00	2.970,00

Tabla 4.12: Costos y Precios

Fuente: Propia

4.1.7.5 Características del Producto

4.1.7.6 Facilidades de acceso y uso

El sistema será desarrollado utilizando el motor de reglas de producción Drools 5.X en la capa de negocio que será manejado por el carrito de compras a través de las reglas de negocio que se le suministrará.

4.1.7.7 Unificación de la Información

Uno de los principales objetivos del sistema es explorar el uso de una metodología de reglas de negocio en la fase de alcance y planificación de un proyecto de software.

4.1.7.8 Mejor control y validación de la información

Los usuarios del sistema contarán con validación para el ingreso de la información necesaria.

4.1.7.9 Restricciones

El sistema tiene las siguientes limitaciones:

- La Tienda Virtual solo será aplicada para la Empresa ENCHAPES ESPECIALES productores de accesorios para la decoración de muebles.
- La Tienda Virtual no permitirá administrar las reglas de negocio. La administración se la realizara con GUVNOR Drools.
- Para realizar las pruebas de Ventas on-line el propietario deberá tener la infraestructura necesaria.

4.1.7.10 Seguridad

Los usuarios del sistema deben especificar un nombre de usuario y una contraseña proporcionados por el usuario administrador para obtener acceso.

El sistema cuenta con el manejo de variables de sesión para el control de acceso a usuarios y navegación.

4.1.7.11 Otros requerimientos del sistema

4.1.7.12 Requisitos de Sistema

Para la publicación de la aplicación se necesita:

- Instalar y configurar el servidor de aplicaciones
- Validar la conexión al servidor de base de datos
- Cargada la data básica (usuario administrador y roles)

4.1.7.13 Requisitos de Calidad

El desarrollo de la Tienda Virtual para la empresa ENCHAPES ESPECIALES se ajustará a la Metodología de Desarrollo de Software RUP, con los parámetros de calidad establecidos por la metodología. La calidad del producto informático permitirá cumplir con su objetivo, validando la información de ingreso, mejorando en eficiencia con información fiable y transparente.

4.1.7.14 Requisitos a nivel intelectual y licencia.

El desarrollo del proyecto se basó en software libre tanto en lenguaje de programación como en base de datos.

4.1.7.15 Requisitos de Software

Para el flujo de información se puede precisar los equipos informáticos y el software necesario planteado en el proyecto.

Usabilidad

- Capacidad de comprensión del sitio global
- Capacidad de estética y de interfaz

Funcionalidad

- Servicio de búsquedas y facilidad de navegación entre páginas

Fiabilidad

- Proceso correcto en la recuperación de errores
- Correcta validación de información para ingreso el ingreso de la información

Eficiencia

- Rendimiento del tiempo de respuesta
- Velocidad de la generación de páginas

Capacidad de mantenimiento

- Facilidad de corrección
- Adaptabilidad
- Extensibilidad

4.1.7.16 Requisitos a Nivel de Software.

Base de Datos

- PostgreSql 9.x o superior versión estable

Browser

- Microsoft Internet Explorer (browser)
- Mozilla Firefox
- Google Chrome

Lenguaje de desarrollo

- Java
- JSF 2
- JavaScript
- AJAX
- HTML
- Drools 5.5

Servidores

- Servidor Aplicaciones: JBoss-AS 7.1
- Servidor de base de datos: Postgres 9.2
- Máquina de JAVA
- Guvnor Drools

Sistema operativo

- Linux/Windows Server, para el servidor
- Windows/Linux, para los usuarios

Lenguaje

- Back-End: Español
- Front-End: Español

4.1.7.18 Requisitos de documentación.

- Manual de Administrador.
- Manual de Usuario
- Manual Técnico.

4.2 PLAN DE DESARROLLO DEL SOFTWARE

El objetivo de la Planificación del proyecto de software es proporcionar un Marco de Trabajo (Estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software (Fundación Wikimedia, 2014)), que permita a los involucrados en el proyecto hacer estimaciones razonables de recursos, costos y planificación en el transcurso de la elaboración.

Estas estimaciones se hacen dentro de un lapso de tiempo en el que se limita el inicio, y conforme avanza el proyecto de software se actualizará regularmente hasta la culminación del mismo. Además dichas estimaciones plantean escenarios de modo que los resultados del proyecto pueden limitarse.

Para el desarrollo de la Tienda Virtual se utilizara la metodología RUP (Rational Unified Process), la misma que permitirá utilizar sus normas para definir el proyecto y de la misma manera permitirá organizar de mejor manera tanto el desarrollo como la documentación.

4.2.1 Propósito

El propósito para este proyecto es definir, planificar y controlar el desarrollo de un producto razonable con un costo mínimo y un período de tiempo específico.

Además se debe analizar detenidamente las variables del desarrollo del proyecto para que en este proceso se fortalezca las bases del sistema y no se deje de lado este importante proyecto.

4.2.2 Alcance

Utilizando la metodología de RUP se ha generado el plan de desarrollo de este proyecto, para así poder tener la documentación necesaria en el momento que así se lo requiera. Para definir bien los requisitos y objetivos de este proyecto, la empresa ENCHAPES ESPECIALES productores de accesorios para decoración de muebles ha colaborado con la información necesaria para el desarrollo del proyecto.

4.2.3 Vista general del proyecto

La información que a continuación se incluye ha sido proporcionad de ciertas reuniones que se han realizado con el gerente propietario encargado de proporcionar la información necesaria para realizar la aplicación.

La empresa "ENCHAPES ESPECIALES productores de accesorios para decoración de muebles" al inicio de cada recepción de artesanías de los respectivos artesanos lo realiza de forma rudimentaria y manualmente es por ello que considera necesario el desarrollo de la Tienda Virtual para la administración de artesanías, artesanos e incrementar las ventas de artesanías y promocionar dichos productos a nivel nacional e internacional.

El proyecto debe proporcionar una propuesta para el desarrollo de todos los subsistemas implicados en la Tienda Virtual. Estos subsistemas se pueden diferenciar en los siguientes bloques:

El proyecto se crea con la finalidad de satisfacer necesidades específicas de la empresa. En el Capítulo III se ha analizado a detalle el sistema, lo que permite distinguir los siguientes módulos:

- Gestión de Artesanías.
- Gestión de Artesanos.
- Gestión de Clientes.
- Ingreso y administración de Reglas de Negocio a través de Drools GUVNOR.
- Ingreso y administración de Carrito de Compras.
- Generación de reportes.

4.2.4 Propósito, alcance y objetivos

La información que a continuación se incluye ha sido extraída de las diferentes reuniones que se han celebrado con el stakeholder de la empresa Enchapes Especiales desde el inicio del proyecto.

El gerente propietario de la empresa Enchapes Especiales realiza el registro de las artesanías, artesanos, promociones, ventas y pedido en archivos Excel, lo que dificulta enormemente la administración de la misma ya que no se cuenta con una base de datos personalizada; es por ello que se considera necesario el desarrollo de una Tienda Virtual que permitirá la gestión de la información generada como resultado de su razón social y sobre todo potencializar la promoción y venta de artesanías desde la web.

El proyecto debe proporcionar una propuesta para el desarrollo de todos los implicados en la venta de artesanías.

Solucionar los requerimientos de los involucrados en el proceso de registro, promoción y venta de artesanías mejorara substancialmente la administración de la empresa Enchapes Especiales en lo referente a los artesanos, propietarios y clientes.

4.2.5 Administración, Asignación y Restricciones de la Tienda Virtual

- Ingreso y administración de usuarios: actualizar las contraseñas, crear y eliminar usuarios
- Ingreso y administración de artesanías.

- Ingreso y administración de artesanos
- Ingreso y administración de clientes
- Ingreso y Administración de pedidos
- Gestión del carrito de compras.
- Administración de Seguridades del Sistema
- Generación de Reportes.

4.2.6 Suposiciones y Restricciones

Las suposiciones y restricciones con respecto a la aplicación a desarrollarse, se derivan directamente de las entrevistas con el gerente propietario.

Puntos críticos:

- Seguridad en la aplicación.
- La Tienda Virtual solo será aplicada para la Empresa ENCHAPES ESPECIALES productores de accesorios para la decoración de muebles.
- Para realizar las pruebas de Ventas on-line el propietario deberá tener la infraestructura necesaria.
- La gestión de las reglas de negocio de la Tienda Virtual se lo realizará a través GUVNOR Drools.
- Integración de la aplicación.
- La empresa se encargara de adquirir la infraestructura necesaria para las pruebas y puesta en producción de la Tienda Virtual.

4.2.7 Entregables del Proyecto

A continuación, se indican y describen cada uno de los artefactos que serán generados y utilizados por el proyecto, estos constituyen los artefactos entregables. Esta lista constituye la configuración o esquema del RUP desde la perspectiva de artefactos el mismo que será utilizado para el desarrollo de este proyecto. Es preciso destacar que de acuerdo a la filosofía de RUP (todo proceso es iterativo e incremental), todos los artefactos son objeto de modificaciones a lo largo del proceso de desarrollo, con lo cual, sólo al término del proceso se podrá tener una versión definitiva y completa de cada uno de ellos. Sin embargo, el resultado de cada iteración y los hitos del proyecto están enfocados a conseguir un cierto grado de completitud (Larousse, 2009) y estabilidad de los artefactos.

Visión

Este documento define la visión del producto desde la perspectiva del usuario, especificando las necesidades y características del producto.

Plan de Desarrollo del Software

Es el presente documento.

Glosario

Se establece una descripción de los términos que se utiliza en este proyecto.

Especificaciones de Casos de Uso

Se representará mediante Diagramas de Casos de Uso las funciones del sistema y los actores que hacen uso de ellas., además para casos de uso cuyo flujo de eventos sea complejo podrá adjuntarse una representación gráfica mediante un Diagrama de Actividad.

Prototipos de interfaces de usuario.

Se trata de prototipos que permiten al usuario hacerse una idea más o menos precisa de la interfaz que proveerá la aplicación y así, conseguir retroalimentación de su parte respecto a los requisitos del proyecto. Estos prototipos se realizarán con las mismas herramientas con las que se realizará todo el proyecto obteniendo un producto en el que se pueda navegar y ejecutar la aplicación. Así mismo, este artefacto, será desechado en la fase de construcción en la medida que el resultado de las iteraciones vayan desarrollando el producto final.

Modelo de análisis y diseño.

Este modelo establece la realización de los casos de uso en clases y pasando desde una representación en términos de análisis (sin incluir aspectos de implementación) hacia una de diseño (incluyendo una orientación hacia el entorno de implementación), de acuerdo al avance del proyecto.

Modelo de Datos

Este modelo describe la representación lógica de los datos persistentes, de acuerdo con el enfoque para modelado relacional de datos. Para expresar este modelo se utiliza un Diagrama de Clases.

Diccionario de datos

Un diccionario de datos es un conjunto de metadatos que contiene las características lógicas de los datos que se van a utilizar en el sistema, incluyendo nombre, tipo y descripción.

Modelo de Implementación

Este modelo es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros ejecutables, ficheros de código fuente, y todo otro tipo de ficheros necesarios para la implantación y despliegue del sistema. Este modelo es sólo una versión preliminar al final de la fase de Elaboración, posteriormente tiene bastante refinamiento.

Lista de Riesgos

Este documento incluye una lista de los riesgos conocidos y vigentes en el proyecto, ordenados en orden decreciente de importancia y con acciones específicas de contingencia o para su mitigación.

Casos de Prueba

Las pruebas son especificadas mediante un documento que establece las condiciones de ejecución, las entradas de la prueba, y los resultados esperados. Estos casos de prueba son aplicados como pruebas de regresión en cada iteración. Cada caso de prueba llevará asociado un procedimiento de prueba con las instrucciones para realizar la prueba.

Manual de Instalación

Este documento incluye las instrucciones para realizar la instalación de la herramienta con la que se realizó el producto.

Material de Apoyo al Usuario Final

Corresponde a un conjunto de documentos y facilidades de uso del sistema, incluyendo: Manual Técnico, Manual de Operación.

Producto

La estructura del sistema Tienda Virtual será empaquetada y almacenada en un CD con los mecanismos apropiados para facilitar su instalación. El producto, a partir de la

primera iteración de la fase de construcción es desarrollado incremental e iterativamente, obteniéndose una nueva versión al final de cada iteración.

4.2.8 Organización del Proyecto

Participantes del Proyecto

Programador. Al ser el proyecto para obtener el título de Ingeniera en Sistemas Computacionales el desarrollo de este proyecto le corresponde señor Franklin Germán Chamorro Chuquín, bajo la dirección del Ing. Ing. José Luís Rodríguez, como director de Tesis.

Ingeniero de software. Al ser el proyecto para obtener el título de Ingeniera en Sistemas Computacionales el desarrollo del proyecto bajo la metodología RUP le corresponde al señor Franklin Germán Chamorro Chuquín, bajo la dirección del Ing. José Luís Rodríguez como director de Tesis.

Roles y Responsabilidades

A continuación se describen las principales responsabilidades de cada uno de los puestos en el equipo de desarrollo durante las fases de Inicio y Elaboración, de acuerdo con los roles que desempeñan en RUP.

Puesto	Responsabilidades
Analista de Sistemas	Captura, determina y valida los requisitos, interactuando con los usuarios mediante entrevistas. Elaboración del Modelo de Análisis y Diseño. Colaboración en la elaboración de las pruebas funcionales y el modelo de datos.
Programador	Construcción de prototipos. Colaboración en la elaboración de las pruebas funcionales, modelo de datos y en las validaciones con el usuario
Ingeniero de Software	Gestión de requisitos, gestión de configuración y cambios, elaboración del modelo de datos, preparación de las pruebas funcionales, elaboración de la documentación. Elaborar modelos de implementación y despliegue.

Tabla 4.13: Roles y Responsabilidades

Fuente: Propia

4.2.9 Plan del Proyecto

Se presenta la organización en fases e iteraciones y el calendario del proyecto.

4.2.10 Plan de las Fases

El desarrollo se llevará a cabo en base a fases con una o más iteraciones en cada una de ellas. La siguiente tabla muestra la distribución de tiempos y el número de iteraciones de cada fase (para las fases de Construcción y Transición es sólo una aproximación muy preliminar).

Fase	Número de Interacciones	Duración
Fase de inicio	2	4 semanas
Fase de Elaboración	1	5 semanas
Fase de Construcción	1	8 semanas
Fase de Transición	-	- semanas

Tabla 4.14: Plan de las Fases

Fuente: Propia

Los hitos que marcan cada fase se describe a continuación

Descripción	Hitos
Fase de Inicio	En esta fase desarrollará los requisitos del producto desde la perspectiva del usuario, los cuales serán establecidos en el artefacto Visión. Los principales casos de uso serán identificados y se hará un refinamiento del Plan de Desarrollo del Proyecto. La aceptación del cliente / usuario del artefacto Visión y el Plan de Desarrollo marcan el final de esta fase.
Fase de Elaboración	En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y / o críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán

	implementados en la primera versión de la fase de Construcción deben estar analizados y diseñados (en el Modelo de Análisis / Diseño). La revisión y aceptación del prototipo de la arquitectura del sistema marca el final de esta fase. En este caso particular, por no incluirse las fases siguientes, la revisión y entrega de todos los artefactos hasta este punto de desarrollo también se incluye como hito. La primera iteración tendrá como objetivo la identificación y especificación de los principales casos de uso, así como su realización preliminar en el Modelo de Análisis / Diseño, también permitirá hacer una revisión general del estado de los artefactos hasta este punto y ajustar si es necesario la planificación para asegurar el cumplimiento de los objetivos. Ambas iteraciones tendrán una duración de una semana.
Fase de Construcción	Durante la fase de construcción se terminan de analizar y diseñar todos los casos de uso, refinando el Modelo de Análisis/Diseño. El producto se construye en base a 2 iteraciones, cada una produciendo una versión a la cual se le aplican las pruebas y se valida con el cliente / usuario. Se comienza la elaboración de material de apoyo al usuario.
Fase de Transición	En esta fase se prepararán dos realces para distribución, asegurando una implantación y cambio del sistema previo de manera adecuada, incluyendo el entrenamiento de los usuarios y el empaquetamiento del producto.

Tabla 4.15: Plan de Fases: Hitos

Fuente: Propia

Calendario del Proyecto

A continuación se presenta un calendario de las tareas del proyecto incluyendo las fases de Inicio, Elaboración y Construcción. Como se ha comentado, el proceso iterativo e incremental de RUP está caracterizado por la realización en paralelo de todas las disciplinas de desarrollo a lo largo del proyecto, con lo cual la mayoría de los artefactos son generados muy tempranamente en el proyecto pero van desarrollándose en mayor o menor grado de acuerdo a la fase e iteración del proyecto. La siguiente figura ilustra este

enfoque, en ella lo ensombrecido marca el énfasis de cada flujo de trabajo en un momento determinado del desarrollo.

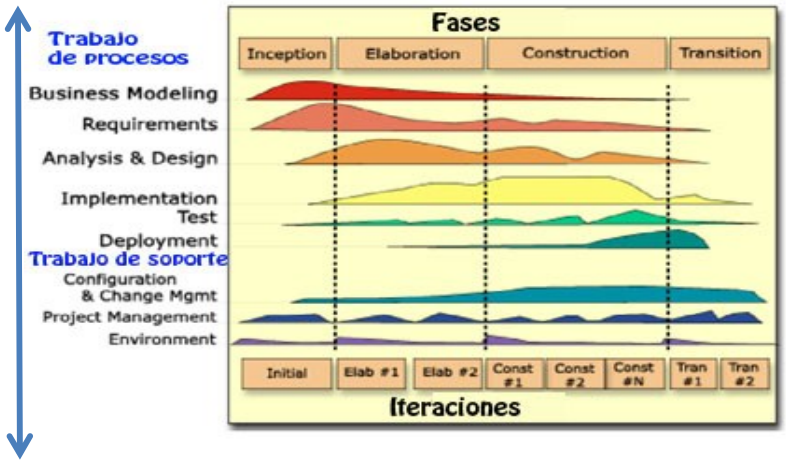


Figura 4.2: Rational Unified Process (RUP)

Fuente: (Diaz Flores, 1996)

Para este proyecto se ha establecido el siguiente calendario. La fecha de aprobación indica cuándo el artefacto en cuestión tiene un estado de completitud suficiente para someterse a revisión y aprobación, pero esto no quita la posibilidad de su posterior refinamiento y cambios.

Disciplinas / Artefactos generados o modificados durante la Fase de Inicio	**Comienzo**	**Aprobación**
Requisitos		
Visión	Semana 1	Semana 4
Modelo de Casos de Uso	Semana 2	Siguiente fase
Especificación de Casos de Uso	Semana 3	siguiente fase
Gestión del proyecto		
Plan de Desarrollo del Software en su versión 1.0	Semana 4	Revisar en cada Fase
Ambiente	Durante todo el proyecto	

Tabla 4.16: Artefactos: Fase de Inicio

Fuente: Propia

Disciplinas / Artefactos generados o modificados durante la Fase de Elaboración	Comienzo	Aprobación
Requisitos		
Visión		Aprobado
Modelo de Casos de Uso	Semana 5	Semana 7
Especificación de Casos de Uso	Semana 6	Semana 7
Análisis / Diseño		
Modelos de Datos	Semana 7	Revisar en cada iteración
Implementación		
Prototipos de Interfaces de Usuario	Semana 10	Revisar en cada iteración
Gestión del proyecto		
Plan de Desarrollo del Software en su versión 2.0 Iteraciones	Semana 6	Revisar en cada iteración
Ambiente	Durante todo el proyecto	

Tabla 4.17: Artefactos. Fase de Elaboración

Fuente: Propia

Disciplinas / Artefactos generados o modificados durante la Fase de Construcción	Comienzo	Aprobación
Análisis/Diseño		
Modelo de Datos	Fase anterior	Revisar en cada iteración
Implementación		

Prototipos de Interfaces de Usuario	Fase anterior	Revisar en cada iteración
Modelo de Objetos de Negocios	Semana 11	Revisar en cada iteración
Pruebas		
Casos de Pruebas Funcionales	Semana 12	Revisar en cada iteración
Gestión del proyecto		
Plan de Desarrollo del Software en su versión 3.0 y planes de las Iteraciones	Semana 11	Revisar en cada iteración
Casos de Uso negociados para la Iteración		
Casos de Uso escogidos	Semana 11	Semana 12
Ambiente	Durante todo el proyecto	

Tabla 4.18: Artefactos: Fase de Construcción

Fuente: Propia

Seguimiento y control del proyecto

Gestión de Requisitos

Los requisitos del sistema son especificados en la Visión del Proyecto. Cada requisito tendrá una serie de atributos tales como importancia, estado, entre otros. Estos atributos permitirán realizar un efectivo seguimiento de cada requisito.

Los cambios en los requisitos serán gestionados mediante una Solicitud de Cambio, las cuales serán evaluadas y distribuidas para asegurar la integridad del sistema y el correcto proceso de gestión de configuración y cambios.

Control de Plazos

El calendario del proyecto tendrá un seguimiento y evaluación semanal por el responsable del proyecto y demás miembros del equipo.

Control de Calidad

Los defectos detectados en las revisiones y formalizados también en una Solicitud de Cambio tendrán un seguimiento para asegurar la conformidad respecto de la solución de dichas deficiencias. Para la revisión de cada artefacto y su correspondiente garantía de calidad se utilizarán las guías de revisión incluidas en RUP.

Gestión de Riesgos

A partir de la fase de Inicio se mantendrá una lista de riesgos asociados al proyecto y de las acciones establecidas como estrategia para mitigarlos o acciones de contingencia.

4.3 ARQUITECTURA DEL SISTEMA

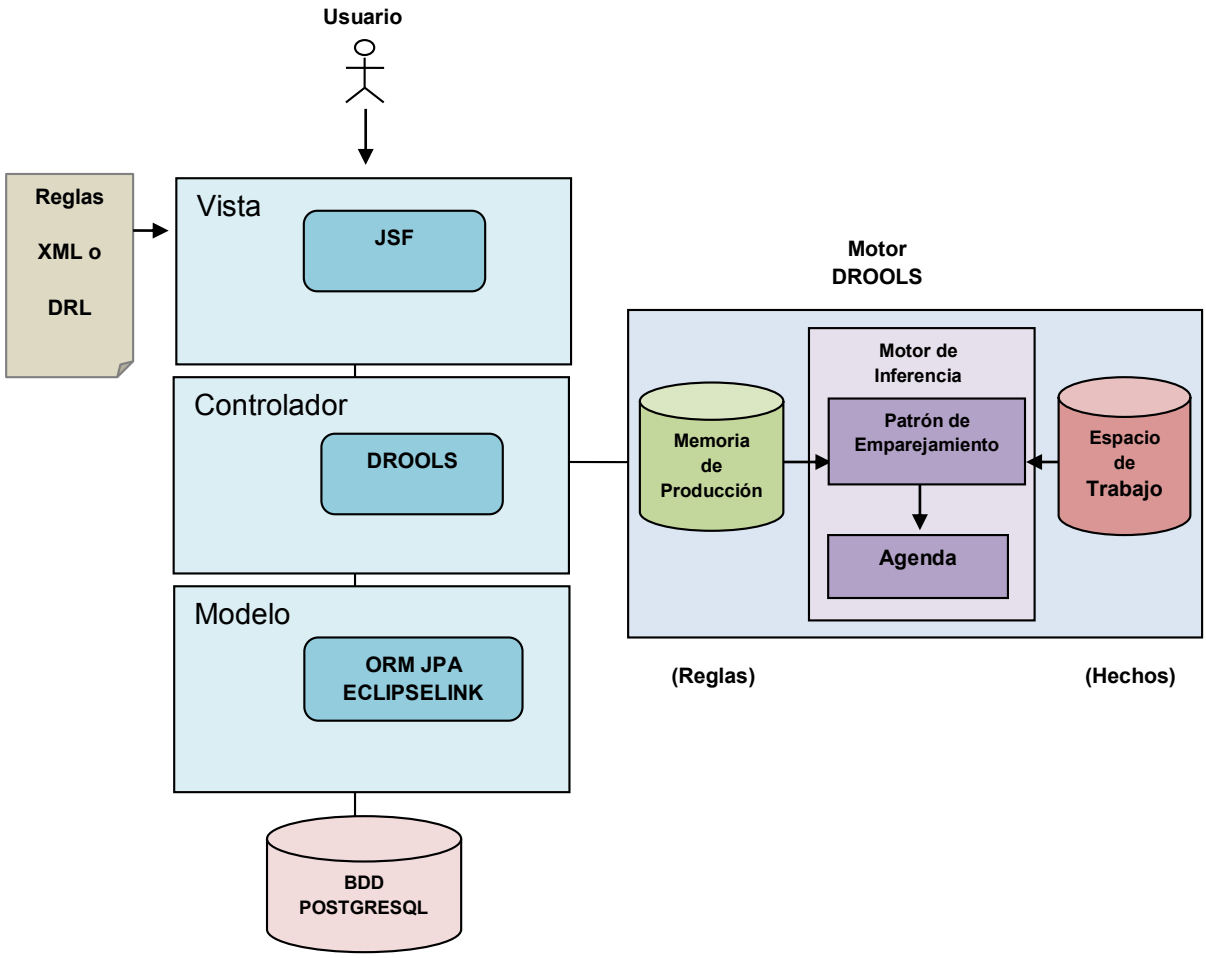


Figura 4.3: Arquitectura de la Tienda Virtual "SanAntonioStore"
Fuente: **Propia**

La arquitectura propuesta para la implementación del aplicativo se describe a continuación.

- Se tiene un **Servidor de Aplicaciones JBoss-AS** en donde está corriendo la Tienda Virtual "**SanAntonioStore**", para que el sistema funcione se tiene a JSF un marco de trabajo de interfaces de usuario del lado del servidor para aplicaciones Web basada en tecnología Java (Juan José Meroño Sánchez, 2009, pág. 48) el cuál responderá a las peticiones realizadas por el usuario a través de la web, JSF accede a la base de datos PostgreSQL a través de los Managed Beans que interactúan conjuntamente un contenedor de EJBs los cuales permiten realizar las operaciones básicas CRUD mediante la capa de persistencia JPA; además estos componentes son la base para la integración del motor de reglas de producción Drools Expert que se encuentra en la capa de negocio del patrón de diseño Modelo-Vista-Controlador.
- En la **Base de Datos PostgreSQL** se tiene la base de datos de nombre SanAntonioStore la cual está estructurada en el esquema public y tablespace SanAntonioStore.
- La administración de reglas de negocio se lo realizará mediante Guvnor Drools que es un sistema de administración de reglas de negocio con interfaces de usuario web.
- El usuario Administrador de la lógica comercial podrá gestionar las reglas de negocio a través de Guvnor mediante un Browser. De la misma forma el usuario Administrador podrá ingresar a la Tienda Virtual para la administración del modelo de datos y los usuarios clientes registrados como visitantes o anónimos tendrán acceso vía web mediante un Browser en el InterNet

CAPÍTULO V

5.1 FASE DE ELABORACIÓN

5.1.1 ESPECIFICACIONES DE CASO DE USO

A continuación los requisitos serán establecidos mediante un modelo de casos de uso que incluirá diagramas de casos de uso utilizando estereotipos que permitirá identificar de mejor manera las funciones del proyecto realizadas por los actores o usuarios del sistema. Por lo tanto los casos de uso determinan los requisitos funcionales del sistema, es decir descripción, flujo básico, flujo alternativo, pre-condiciones y post-condiciones.

El siguiente caso de uso muestra una visión general de la Tienda Virtual

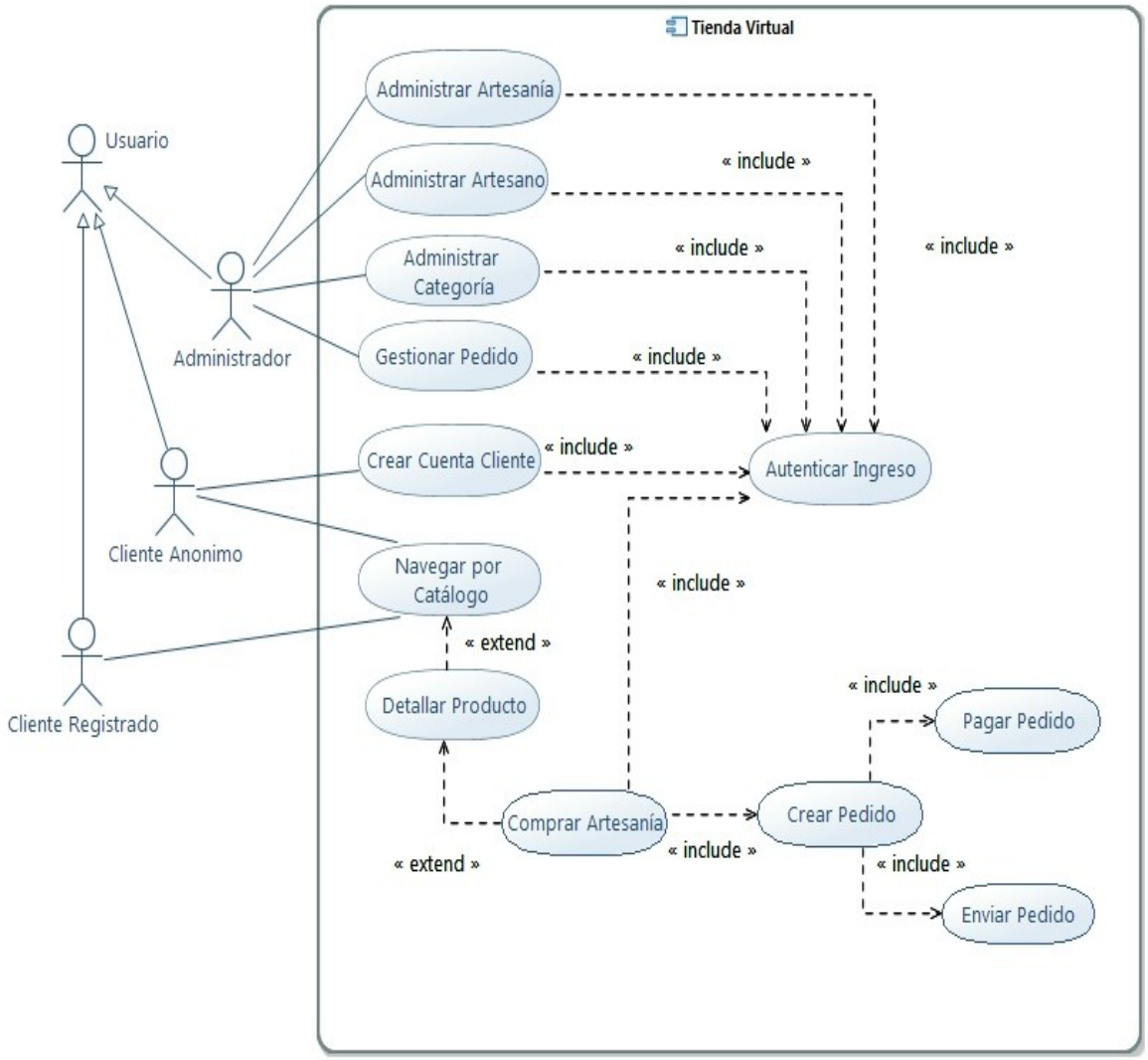


Figura 5.1: Caso de uso general

Fuente: Propia

Caso de Uso: Autenticar Ingreso a la Aplicación.

Caso de uso	Autenticar Ingreso a la Aplicación.	
Descripción	El usuario se conecta a la Tienda Virtual.	
Actor iniciador	Usuario	
Actores secundarios	-	
Resumen	El usuario inicia sesión con su usuario y contraseña a la Tienda Virtual.	
Pre condiciones	El usuario se ha registrado previamente, tiene un estado activo y dispone de una contraseña.	
Post condiciones	-	
Flujo de eventos	***Interacciones del usuario***	***Obligaciones del sistema***
	1. El usuario va a la página desde la que puede conectarse (login.xhtml). 3. El usuario introduce sus datos en el formulario. 4. El usuario pincha en el botón "Ingresar".	2. El sistema solicita los datos a través de un formulario. 5. El sistema recoge los datos del formulario. 6. El sistema comprueba que dichos datos se encuentren en la base de datos. 7. El sistema inicia sesión con el usuario en cuestión.
Extensiones síncronas	En 4 el usuario puede cancelar la operación. En 6, si el empleado introduce un usuario y contraseña incorrectos, el sistema muestra el error y vuelve a solicitar los datos.	
Extensiones asíncronas	-	

Tabla 5.1: Caso de Uso: Autenticar Ingreso a la Aplicación
Fuente: **Propia**

Caso de Uso: Agregar Artesanías al Carrito de Compras.

Caso de uso	Agregar Artesanías al Carrito de Compras	
Descripción	El usuario agrega un producto al carrito.	
Actor iniciador	Usuario	
Actores secundarios	-	
Resumen	Cualquier usuario podrá añadir artesanías del catálogo de la Tienda Virtual que desee al carrito en cualquier momento sea usuario registrado o anónimo.	
Pre condiciones	-	
Post condiciones	El sistema actualizará el resumen del carrito.	
Flujo de eventos	***Interacciones del usuario***	***Obligaciones del sistema***
	1. El usuario selecciona una artesanía en el catálogo. 3. El usuario pulsa sobre el botón "Añadir al carrito" de la artesanía deseada.	2. El sistema busca en la base de datos las características de la artesanía. 4. El sistema comprueba que la artesanía se encuentre en el carrito. 5. Si se encuentra, suma la cantidad a la artesanía existente en el carrito. 6. Si no se encuentra, se añade la artesanía nueva al carrito. 7. El sistema actualiza los totales del carrito.
Extensiones síncronas	En 3 el usuario puede cancelar la operación.	
Extensiones asíncronas	-	

Tabla 5.2 : Caso de Uso: Agregar Artesanías al Carrito de Compras.
Fuente: Propia

Caso de Uso: Eliminar Artesanía del Carrito de Compras.

Caso de uso	Eliminar Artesanía del carrito del compras	
Descripción	El usuario elimina una artesanía del carrito.	
Actor iniciador	Usuario	
Actores secundarios	-	
Resumen	Cualquier usuario podrá eliminar una artesanía del carrito en el momento que desee.	
Pre condiciones	La artesanía debe haber sido añadida al carrito anteriormente.	
Post condiciones	El sistema actualizará el resumen del carrito.	
Flujo de eventos	***Interacciones del usuario***	***Obligaciones del sistema***
	1. El usuario va a la página donde se encuentra el resumen del carrito (Ver cesta). 3. Pulsa sobre el botón “Eliminar” de la/las artesanía/s que desea eliminar del carrito.	2. El sistema muestra el resumen del carrito de compras. 4. Busca en el carrito el identificador de las artesanías marcadas. 5. Elimina del carito las artesanías seleccionadas. 6. Actualiza los totales del carrito.
Extensiones síncronas	En 3 el usuario puede eliminar todos los elementos del carrito el sistema mostrará un mensaje indicando que el carrito está vacío.	
Extensiones asíncronas	-	

Tabla 5.3: Caso de Uso: Eliminar Artesanía del carrito de compras
Fuente: Propia

Caso de Uso: Modificar cantidad del Carrito de Compras.

Caso de uso	Modificar cantidad del Carrito de Compras.	
Descripción	El usuario modifica la cantidad de una artesanía añadida al carrito.	
Actor iniciador	Usuario	
Actores secundarios	-	
Resumen	Cualquier usuario podrá modificar la cantidad de una artesanía que desee adquirir.	
Pre condiciones	La artesanía debe haber sido añadida al carrito anteriormente.	
Post condiciones	El sistema actualizará el resumen del carrito.	
Flujo de eventos	***Interacciones del usuario***	***Obligaciones del sistema***
	1. El usuario va a la página donde se encuentra el resumen del carrito. 2. Pulsa sobre el botón "Incrementar cantidad" que se encuentra en Ver cesta.	3. Modifica la cantidad de las artesanías a los que su cantidad es diferente a la anterior. 4. Si la cantidad nueva es cero, el sistema elimina la artesanía del carrito. 5. Actualiza los totales del carrito.
Extensiones síncronas	En 2 el usuario puede modificar todas cantidades a cero, el sistema mostrará un mensaje indicando que el carrito está vacío.	
Extensiones asíncronas	-	

Tabla 5.4: Caso de Uso: Modificar cantidad del Carrito de Compras
Fuente: Propia

Caso de Uso: Crear Cuenta de Usuario.

Caso de uso	Crear cuenta de Usuario.	
Descripción	Crear cuenta de usuario en la Tienda Virtual.	
Actor iniciador	Usuario	
Actores secundarios	-	
Resumen	El usuario anónimo introduce sus datos personales y el sistema lo da de alta como usuario registrado.	
Pre condiciones	El usuario no se ha dado de alta previamente.	
Post condiciones	El nuevo usuario queda registrado en el sistema.	
Flujo de eventos	***Interacciones del usuario***	***Obligaciones del sistema***
	1. El usuario va a la página desde la que puede darse de alta ("Mi Cuenta"). 2. Selecciona la opción "Registrar Nuevo usuario". 4. El usuario introduce sus datos personales. 5. El usuario pincha sobre el botón "Guardar".	3. El sistema solicita los datos personales a través de un formulario. 6. El sistema registra al nuevo usuario en la base de datos. 7. El sistema confirma que el usuario se ha dado de alta correctamente.
Extensiones síncronas	En 4 y 5 el usuario puede cancelar la operación. En 6, si el usuario no ha introducido todos los campos, el sistema informará del error y volverá a solicitarlos.	
Extensiones asíncronas	-	

Tabla 5.5: Caso de Uso: Crear cuenta de Usuario.
Fuente: Propia

Caso de Uso: Modificar Usuario.

Caso de uso	Modificar usuario.	
Descripción	Modificar los datos del usuario en la Tienda Virtual.	
Actor iniciador	Administrador	
Actores secundarios		
Resumen	El administrador modifica los datos de un usuario si es necesario en la base de datos.	
Pre condiciones	El usuario solicita al empleado la modificación de sus datos personales. El empleado está conectado a Internet.	
Post condiciones	Los datos del usuario quedan modificados en el sistema.	
Flujo de eventos	***Interacciones del usuario***	***Obligaciones del sistema***
	1. El administrador pincha en "Usuario o Cliente". 3. El administrador introduce los datos del cliente. 5. El empleado selecciona el nombre del usuario a modificar. 8. El empleado modifica los datos oportunos en el formulario. 9. El empleado pulsa el botón "Enviar datos".	2. El sistema muestra un listado con todos los usuarios o clientes. 4. El sistema muestra las coincidencias de la búsqueda. 6. El sistema busca en la base de datos el cliente seleccionado. 7. El sistema muestra en un formulario todos los datos del cliente. 10. El sistema modifica al usuario en la base de datos. 11. El sistema confirma que el usuario se ha modificado correctamente.
Extensiones síncronas	En 5 y 9 el empleado puede cancelar la operación. En 10, si el usuario no ha dejado un campo en blanco, el sistema informará del error y volverá a solicitarlo.	

Tabla 5.6: Caso de Uso: Modificar usuario.
Fuente: Propia

Caso de Uso: Cambiar estado de Pedido.

Caso de uso	Cambiar estado de Pedido.	
Descripción	Cambiar el estado de un pedido.	
Actor iniciador	Administrador	
Actores secundarios	-	
Resumen	El administrador modifica el estado de uno o varios pedidos en la base de datos.	
Pre condiciones	El pedido se ha realizado previamente. El administrador está conectado a su Internet	
Post condiciones	Los datos del pedido quedan modificados en el sistema.	
Flujo de eventos	***Interacciones del usuario***	***Obligaciones del sistema***
	1. El empleado pincha en "Cambiar el estado de un pedido" de su Intranet. 3. El empleado introduce el número del pedido a modificar. 5. El empleado cambia el estado del pedido. 6. El empleado pincha en el botón "Actualizar".	2. El sistema muestra un listado con todos los pedidos y un buscador. 4. El sistema muestra el pedido buscado. 7. El sistema modifica el pedido en la base de datos. 8. El sistema confirma que el pedido se ha modificado correctamente.
Extensiones síncronas	En 3 y 6 el empleado puede cancelar la operación. En 7, si el empleado introduce un estado incorrecto, el sistema muestra el error y vuelve a solicitar el nuevo estado.	
Extensiones asíncronas	-	

Tabla 5.7: Caso de Uso: Cambiar estado de Pedido
Fuente: Propia

Caso de Uso: Finalizar Pedido.

Caso de uso	Finalizar pedido.	
Descripción	El usuario finaliza su compra.	
Actor iniciador	Usuario	
Actores secundarios	-	
Resumen	El usuario finaliza su pedido y se almacena en la base de datos.	
Pre condiciones	El usuario se ha registrado previamente y ha añadido al carrito artesanías	
Post condiciones	El pedido queda registrado en la base de datos.	
Flujo de eventos	***Interacciones del usuario***	***Obligaciones del sistema***
	1. El usuario va a la página "pedido.xhtml" y pincha en "Realizar Pedido".	2. El sistema accede al Id del usuario. 3. El sistema actualiza la tabla pedido y detalle_pedido. 4. El sistema informa al usuario mediante un mensaje la confirmación del pedido.
Extensiones síncronas	-	
Extensiones asíncronas	-	

Tabla 5.8: Caso de Uso: Finalizar pedido
Fuente: Propia

5.1.2 ANÁLISIS Y DISEÑO

El siguiente diagrama relacional refleja el diseño de la Tienda Virtual en la base de datos.

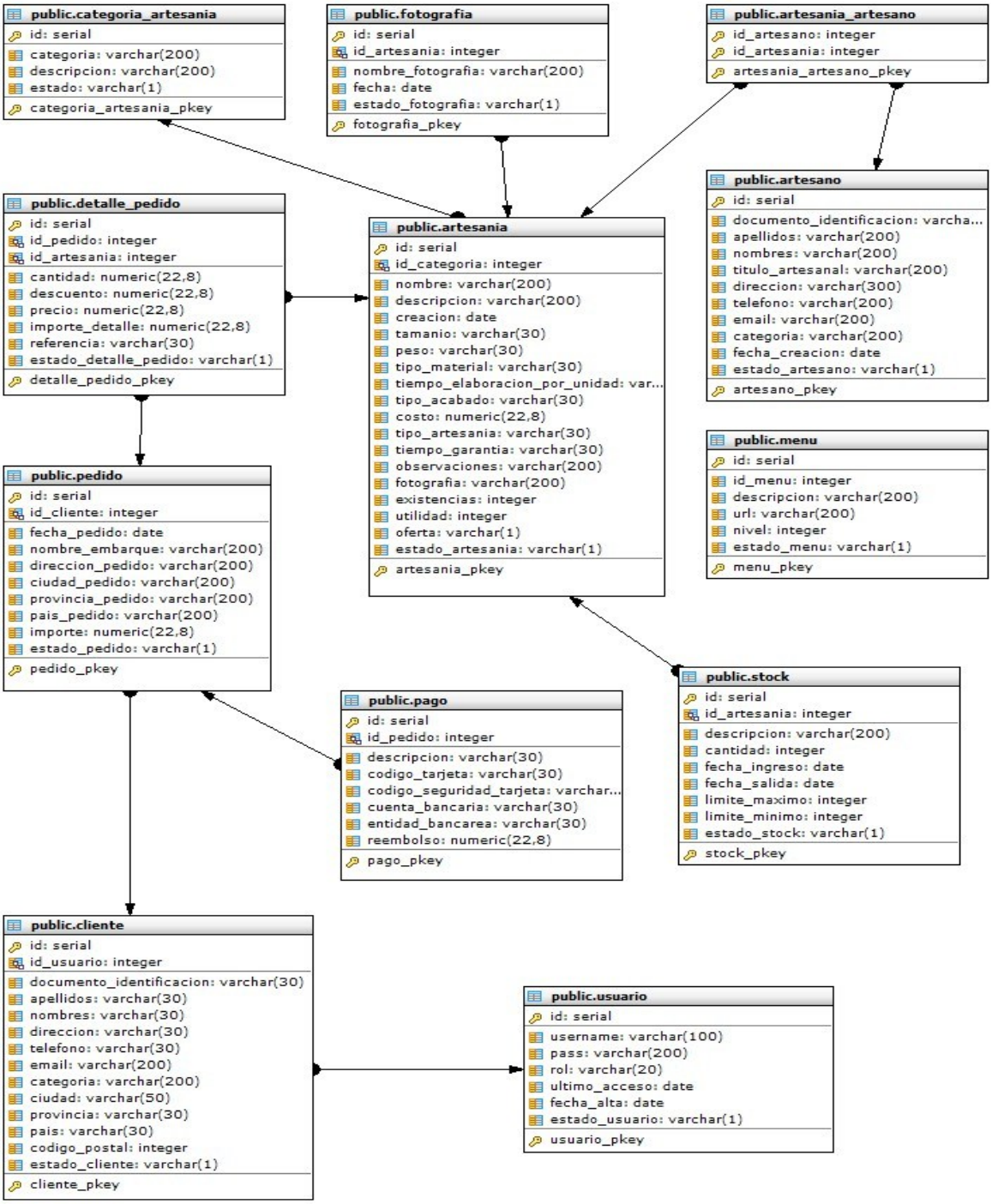


Figura 5.2: Modelos de Datos
Fuente: Propia

Diccionario de Datos

A continuación, se presenta el diccionario de datos para la Tienda Virtual.

Nombre de la tabla: artesanias

artesanías		En esta tabla se almacena los nombres de cada artesanía con sus respectivas características.	
Columnas	**Tipo de Datos**	**Descripción**	**Referencia**
id	integer	Registra un número que se incrementa mediante una secuencia	artesania_pkey
id_categoria	integer	Registra el código de la categoría de artesanía a la que pertenece.	fk_categoria _artesania
nombre	character varying(200)	Almacena el nombre descriptivo de la artesanía	-
descripcion	character varying(200)	Almacena una descripción detallada de la artesanía	-
creacion	date	Almacena la fecha de creación de la artesanía	-
tamanio	character varying(30)	Almacena el tamaño de la artesanía Alto x Ancho x Espesor	-
peso	character varying(30)	Almacena el peso como descripción	-
tipo_material	character varying(30)	Almacena el tipo de Material: Madera	-
tiempo_ elaboracion _por_unidad	character varying(30)	Almacena un valor de tiempo	-
tipo_acabado	character varying(30)	Almacena el tipo de acabado de la artesanía: Lacado, Envejecido, Pintado	-
costo	numeric(22,8)	Representa un número cuyo valor es en dólares	-
tipo_artesania	character varying(30)	Almacena el tipo de artesanía: Decorativa, Utilitaria	-
tiempo_garantia	character varying(30)	Almacena un valor que indica el tiempo en el cual se podrá hacer un reclamo	-
observaciones	character	Almacena la descripción de los	-

	varying(200)	cuidados que se debe tener con las artesanías	
fotografia	character varying(200)	Almacena la dirección física de la imagen	-
existencias	integer	Almacena el número de artesanías existentes	-
utilidad	integer	Almacena un valor que se le incrementara	-
oferta	character varying(1)	Almacena un valor 'S' o 'N' si esa en oferta o no	-
estado_artesania	character varying(1)	Almacena el estado de la artesanía si está activo o no	-

Tabla 5.9: Diccionario de Datos: Tabla artesanía

Fuente: Propia

Nombre de la tabla: artesano

artesano		En esta tabla se almacena los nombres de cada artesano y sus datos personales.	
Columnas	**Tipo de Dato**	**Descripción**	**Referencia**
id	integer	Registra un número que se incrementa mediante una secuencia	artesano_pkey
documento_ identificacion	character varying(30)	Registra documento de identificación del artesano.	-
apellidos	character varying(200)	Registra los Apellidos del artesano	-
nombres	character varying(200)	Registra los nombres del artesano	-
titulo_artesanal	character varying(200)	Registra un documento que identifica la profesión de artesano	-
direccion	character varying(300)	Registra la dirección de residencia del artesano	-
telefono	character varying(200)	Registra un número de teléfono del artesano.	-
email	character varying(200)	Registra un correo electrónico del artesano	-

categoria	character varying(200)	Registra una categoría del artesano: Mayorista o Minorista	-
fecha_creacion	date	Almacena la fecha en que fue creado el artesano	-
estado_artesano	character varying(1)	Almacena el estado del artesano: S o N	

Tabla 5.10: Diccionario de Datos: Tabla artesano

Fuente: Propia

Nombre de la tabla: categoría_artesania

categoria_artesania		En esta tabla se almacena los nombres de cada categoría de las artesanías.	
Columnas	**Tipo de Datos**	**Descripción**	**Referencia**
id	integer	Registra un número que se incrementa mediante una secuencia	categoria_artesania_pkey
categoria	character varying(200)	Describe una categoría que se asignará a la artesanía	-
descripcion	character varying(200)	Almacena una descripción de la categoría	-
estado	character varying(1)	Almacena el estado de la categoría: S o N	-

Tabla 5.11: Diccionario de Datos: Tabla categoría_artesania

Fuente: Propia

Nombre de la tabla: cliente

cliente		En esta tabla se almacena los nombres de cada cliente registrado en la Tienda Virtual.	
Columnas	**Tipo de Datos**	**Descripción**	**Referencia**
id	integer	Registra un número que se incrementa mediante una secuencia	cliente_pkey
Documento _identificacion	character varying(30)	Registra documento de identificación del cliente.	-
apellidos	character varying(30)	Registra los apellidos del cliente	-
nombres	character	Registra los nombres del cliente	-

	varying(30)		
direccion	character varying(30)	Registra la dirección del cliente	-
telefono	character varying(30)	Registra teléfono del cliente	-
email	character varying(200)	Registra un correo electrónico del cliente	-
categoria	character varying(200)	Registra una categoría de cliente: Mayorista o Minorista	-
ciudad	character varying(50)	Registra la ciudad de residencia del cliente	-
provincia	character varying(30)	Registra la provincia de residencia del cliente	-
pais	character varying(30)	Registra el país de residencia del cliente	-
codigo_postal	integer,	Registra un código postal	-
estado_cliente	character varying(1)	Registra el estado del cliente: S o N	-
id_usuario	integer	Referencia con la tabla usuario	fk_usuario_cliente

Tabla 5.12: Diccionario de Datos: Tabla cliente

Fuente: Propia

Nombre de la tabla: detalle_pedido

detalle_pedido		En esta tabla se almacena los detalles de cada pedido.	
Columnas	**Tipo de Datos**	**Descripción**	**Referencias**
id	integer	Registra un número que se incrementa mediante una secuencia	detalle_pedido_pkey
id_pedido	integer	Referencia con la tabla pedido	fk_pedido_ detallepedido
id_artesania	integer	Referencia con la tabla artesanía	fk_artesania_ detallepedido
cantidad	numeric(22,8)	Registra la cantidad comprada de cada artesanía	-
descuento	numeric(22,8)	Registra el descuento para	-

		cada artesanía	
precio	numeric(22,8)	Registra el precio de cada artesanía	-
importe_detalle	numeric(22,8)	Registra el sub total por cada ítem	-
referencia	character varying(30)	Registra una referencia u observación	-
estado_detalle_pedido	character varying(1)	Registra el estado cuando ha sido anulado el pedido	-

Tabla 5.13: Diccionario de Datos: Tabla detalle_pedido

Fuente: Propia

Nombre de la tabla: pedido

pedido		En esta tabla se almacena los pedidos que realiza cada cliente.	
Columnas	**Tipo de Datos**	**Descripción**	**Referencias**
id	integer	Registra un número que se incrementa mediante una secuencia	pedido_pkey
id_cliente	integer	Referencia con la tabla cliente	fk_cliente_pedido
fecha_pedido	date	Registra la fecha cuando se realiza el pedido	-
nombre_embarque	character varying(200)	Registra datos cuando del medio de transporte cuando la venta sea en el exterior del país	-
direccion_pedido	character varying(200)	Registra la dirección destino del pedido	-
ciudad_pedido	character varying(200)	Registra la ciudad destino del pedido	-
provincia_pedido	character varying(200)	Registra la provincia destino del pedido	-
pais_pedido	character varying(200)	Registra el país destino del pedido	-
importe	numeric(22,8)	Registra el valor total del pedido	-
estado_pedido	character varying(1)	Registra el estado del pedido	

Tabla 5.14: Diccionario de Datos: Tabla pedido

Fuente: Propia

Nombre de la tabla: usuario

usuario		En esta tabla se almacena los usuarios que se registran en la Tienda Virtual.	
Columnas	Tipo de Datos	Descripción	Restricciones
id	integer	Registra un número que se incrementa mediante una secuencia	usuario_pkey
username	character varying(100)	Registra un usuario único en la Tienda Virtual	-
pass	character varying(200)	Registra una contraseña del usuario	-
rol	character varying(20)	Registra el rol o papel al que pertenece el usuario	-
ultimo_acceso	date	Registra la última fecha de ingreso a la Tienda Virtual	-
fecha_alta	date	Registra la fecha de creación del usuario	-
estado_usuario	character varying(1)	Registra el estado del usuario: S o N	-

Tabla 5.15: Diccionario de Datos: Tabla usuario

Fuente: Propia

Nombre de la tabla: usuario

pago		En esta tabla se almacena los pagos.	
Columnas	Tipo de Datos	Descripción	Referencia
id	integer	Registra un número que se incrementa mediante una secuencia	pago_pkey
descripcion	character varying(30)	Registra una descripción de la forma de pago	-
codigo_tarjeta	character varying(30)	Registra el código secreto de la tarjeta de pago de la transacción	-
codigo_	character varying(30)	Registra código de	-

seguridad_tarjeta		seguridad de la tarjeta	
cuenta_bancaria	character varying(30)	Registra la cuenta bancaria	-
entidad_bancarea	character varying(30)	Registra el nombre de la entidad bancaria	-
reembolso	numeric(22,8),	Registra el reembolso en caso de devolución	-
id_pedido	integer	Referencia con la tabla pedido	fk_pago_pedido

Tabla 5.16: Diccionario de Datos: Tabla detalle_pedido

Fuente: Propia

Nombre de la tabla: fotografia

fotografia		En esta tabla se almacena las fotografías.	
Columnas	Tipo de Datos	Descripción	Referencia
id	integer	Registra un número que se incrementa mediante una secuencia	fotografia_pkey
id_artesania	integer	Referencia a la tabla artesanía	fk_artesania_fotografia
nombre_fotografia	character varying(200)	Registra el nombre de la fotografía de la artesanía	-
fecha	date	Registra la fecha de registro de la fotografía	-
estado_fotografia	character varying(1)	Registra el estado de la fotografía	-

Tabla 5.17: Diccionario de Datos: Tabla fotografía

Fuente: Propia

Nombre de la tabla: stock

stock		En esta tabla se almacena el stock.	
Columnas	**Tipo de Datos**	**Descripción**	**Referencias**
id	integer	Registra un número que se incrementa mediante una secuencia	stock_pkey
id_artesania	integer	Referencia con la tabla artesanía	fk_artesania_stock
descripcion	character varying(200)	Registra una descripción del stock	-
cantidad	integer	Registra la cantidad actual del stock de la artesanía	-
fecha_ingreso	date	Registra la fecha de ingreso del stock	-
fecha_salida	date	Registra la fecha de salida del stock	-
limite_maximo	integer	Registra el límite máximo del stock	-
limite_minimo	integer	Registra el límite mínimo del stock	-
estado_stock	character varying(1)	Registra el estado del stock	-

Tabla 5.18: Diccionario de Datos: Tabla detalle_pedido

Fuente: Propia

5.2 FASE DE CONSTRUCCIÓN

5.2.1 METODOLOGÍA DE REGLAS DE NEGOCIO:

En la fase de construcción se utilizará los 4 principios de la metodología de reglas: Separar, Rastrear, Exteriorizar, Situar o localizar las reglas

Para empezar se dará un identificador secuencial denominado R**N.** Dichas reglas están en lenguaje natural dentro del contexto del negocio.

R1: Si existe un cliente registrado con estado activo y existe una o más artesanías registradas con estado activo y existe en stock

Entonces registrar el pedido con su detalle o artesanías

R2: Si existe un cliente registrado con estado activo y existe una o más artesanías registradas con estado activo y existe en stock y la artesanía está en oferta
Entonces actualizar el importe total de la artesanía multiplicando por el descuento de 1 por ciento

R3: Si existe un cliente registrado con estado activo y existe una o más artesanías registradas con estado activo y existe en stock y el importe total del pedido es mayor o igual 50 dólares
Entonces actualizar el importe total del pedido multiplicando por el descuento de 10 por ciento

R4: Si existe un cliente registrado con estado activo y existe una o más artesanías registradas con estado activo y existe en stock y la cantidad total del número de artesanías es mayor o igual 5 unidades
Entonces actualizar el importe total del pedido multiplicando por el descuento de 5 por ciento

R5: Si existe un cliente registrado con estado activo y su categoría es mayorista y existe una o más artesanías registradas con estado activo y la cantidad total del número de artesanías es mayor o igual 50 unidades
Entonces actualizar el importe total del pedido multiplicando por el descuento de 1 por ciento

R6: Si existe un cliente registrado con estado activo y la fecha de compra esta entre una fecha inicial y una fecha final y existe una o más artesanías registradas con estado activo y la cantidad total del número de artesanías es mayor o igual 2 unidades
Entonces actualizar el importe total del pedido multiplicando por el descuento de 0.5 por ciento

Extracción las clases

Modelo de datos o Definición de Hechos

Cliente:	Identificador de usuario.- Número de cédula
	Nombre y Apellido.- Descripción
	Categoría.- Mayorista o minorista
	Estado.- Activo o In-activo
Categoría de Artesanía:	Identificador secuencial
	Descripción.-
	Estado.- Si o No
Artesanía:	Identificador de artesanía.- Número secuencial

	Nombre.- Descripción
	Categoría.- Utilitaria, decorativa, religiosa…
	Stock.- Cantidad existente en inventario
	Oferta.- Si o No
	Estado.- Si o No
Pedido:	Identificador.- Número secuencial
	Cliente.- Propietario del pedido
	Importe.- Valor total del pedio en dólares
	Fecha.- Fecha en que se realizó el pedido
	Estado.- Activo o In-activo
Detalle Pedido:	Identificador.- Número secuencial
	Pedido.-
	Artesanía.-
	Importe.- Sub total de la compra del artículo actual
	Cantidad.- Número de unidades compradas
	Costo.- Valor de la artesanía en dólares
	Estado.- Activo o In-activo

Generación de reglas de acuerdo a la metodología de reglas

R1: Si existe un Cliente.Estado = ‘S’ y Artesanía. Estado = ‘S’ y Artesania.Stock > 2

Entonces Pedido.Importe = Suma (DetallePedido.Cantidad * DetallePedido.Costo)

R2: Si existe un Cliente.Estado=‘S’ y Artesanía.Estado=‘S’ y Artesania.Stock > 2 y Artesania.Oferta=’S’

Entonces Pedido.importe = Suma (DetallePedido.Cantidad * DetallePedido.Costo) * 0.01

R3: Si existe un Cliente.Estado=‘S’ y Artesanía.Estado=‘S’ y Artesania.Stock > 2 y Pedido.Importe >=50

Entonces Pedido.importe = Pedido.Importe – (Pedido.Importe * 0.1)

R4: Si existe un Cliente.Estado=‘S’ y Artesanía.Estado=‘S’ y Artesania.Stock > 2 y Suma (Artesania.Cantidad) >=5

Entonces Pedido.importe = Pedido.Importe – (Pedido.Importe * 0.05)

R5: Si existe un Cliente.Estado=‘S’ y Cliente.Categoria=’Mayorista’ y Artesanía.Estado=‘S’ y Artesania.Stock > 2 y Suma (Artesania.Cantidad) >=50

Entonces Pedido.importe = Pedido.Importe – (Pedido.Importe * 0.05)

R6: Si existe un Cliente.Estado=‘S’ y Artesanía.Estado=‘S’ y Artesania.Stock > 2 y Suma (Artesania.Cantidad) >= 2 y Pedido.Fecha Entre FechaInicial y FechaFinal

Entonces Pedido.importe = Pedido.Importe – (Pedido.Importe * 0.05)

5.2.2 MODELADO DE OBJETOS DE NEGOCIOS

5.2.2.1 Diagramas de Clases

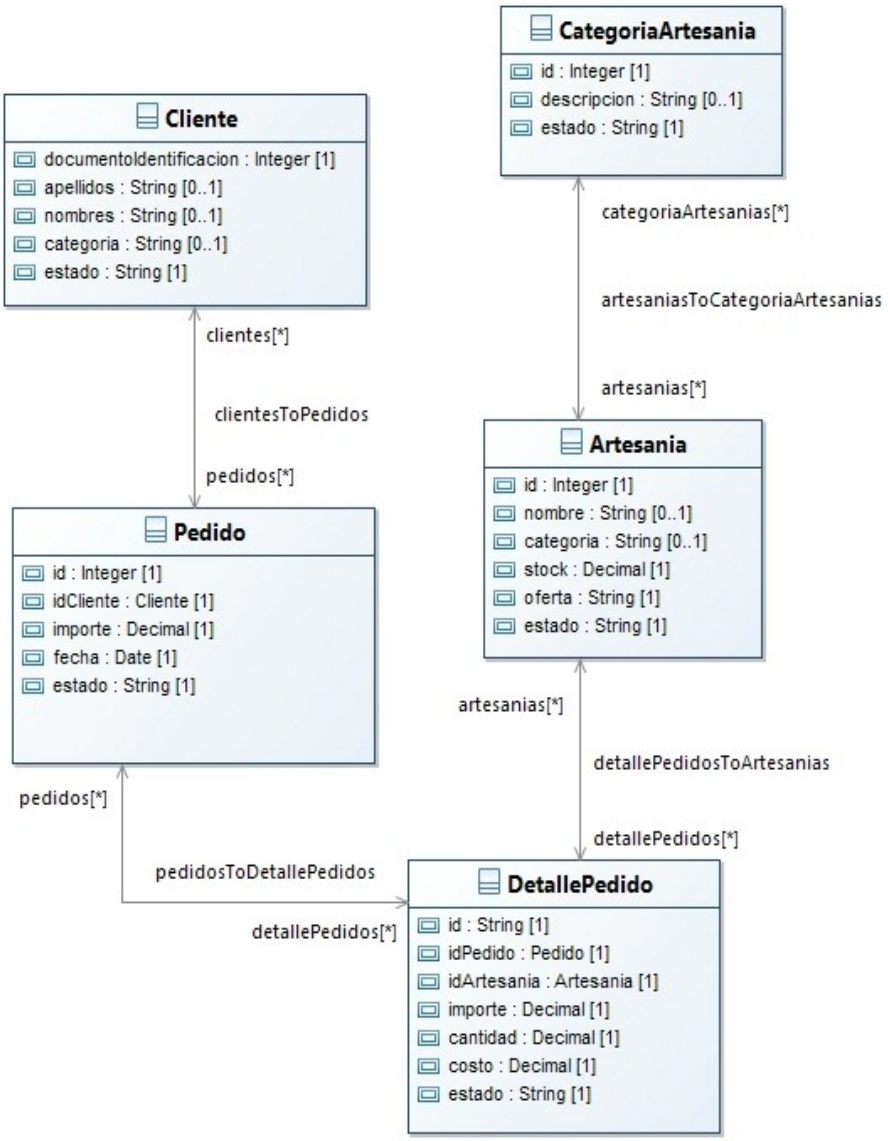


Figura 5.3: Diagrama de Clases

Fuente: Propia

5.2.2.2 Diagramas de Actividades

5.2.2.2.1 Inicio de Sesión

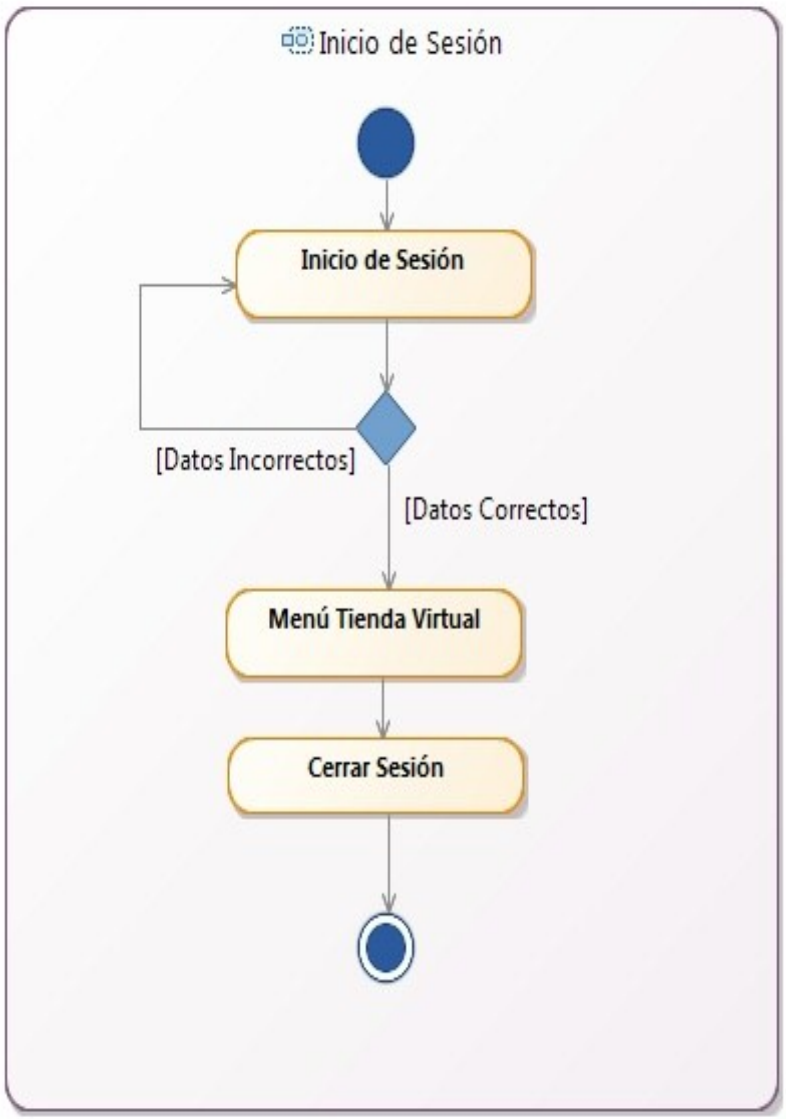


Figura 5.4: Diagrama de Actividades: Inicio de Sesión

Fuente: Propia

5.2.2.2.2 Comprar Artesanía

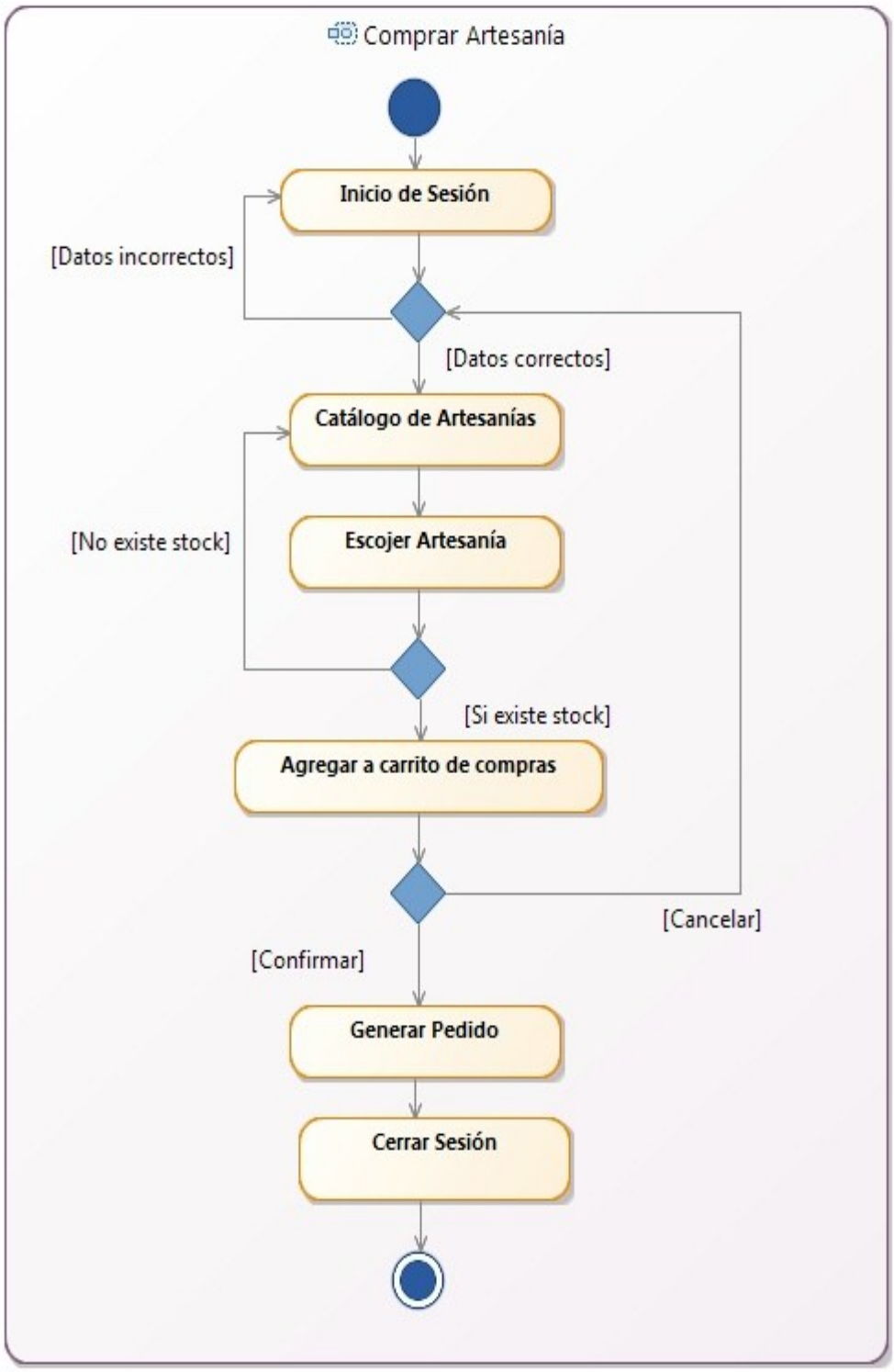


Figura 5.5: Diagrama de Actividades: Comprar Artesanía.

Fuente: Propia

5.2.2.2.3 Creación de Clientes

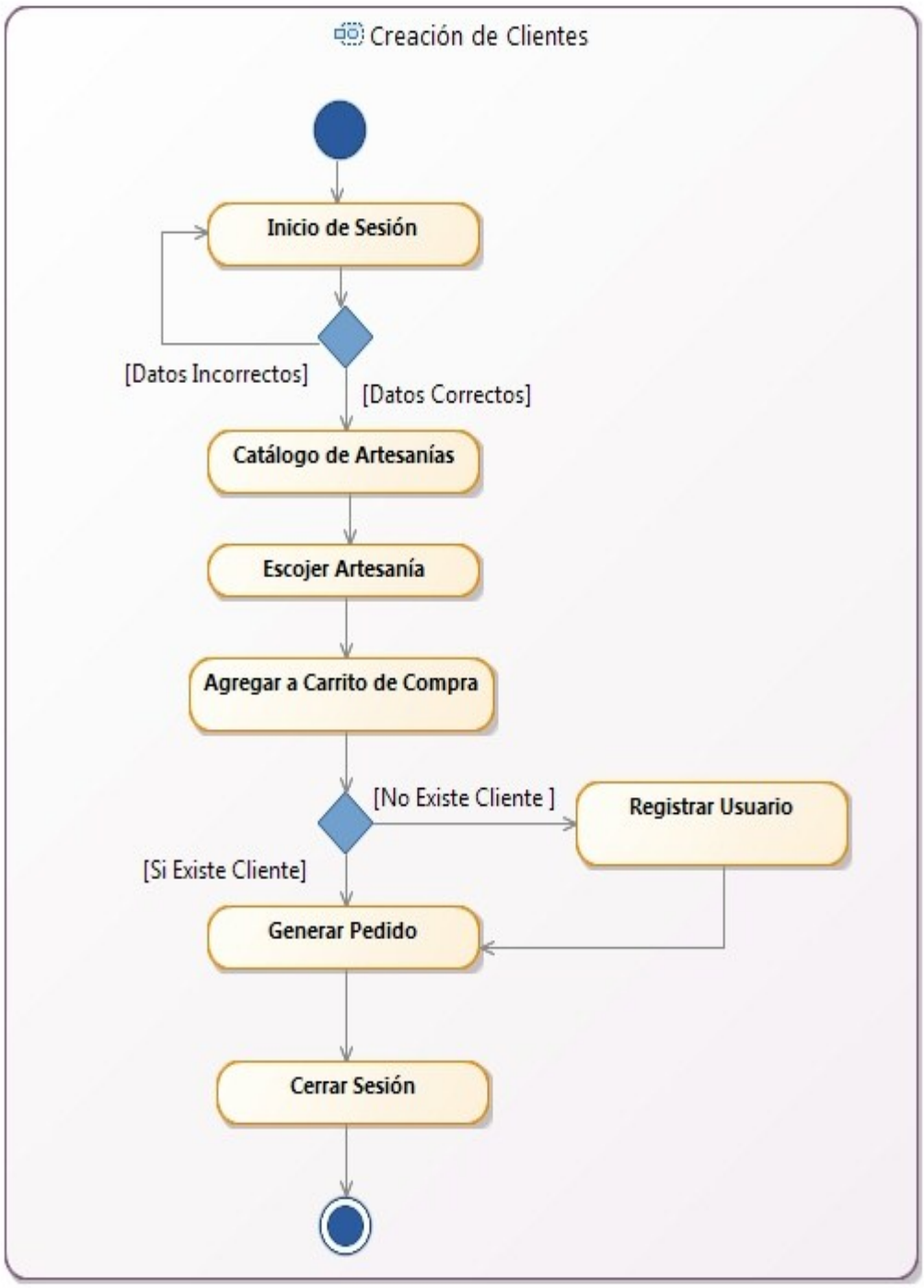


Figura 5.6: Diagrama de Actividades: Creación de Clientes.

Fuente: Propia

5.2.2.2.4 Creación de Artesanía

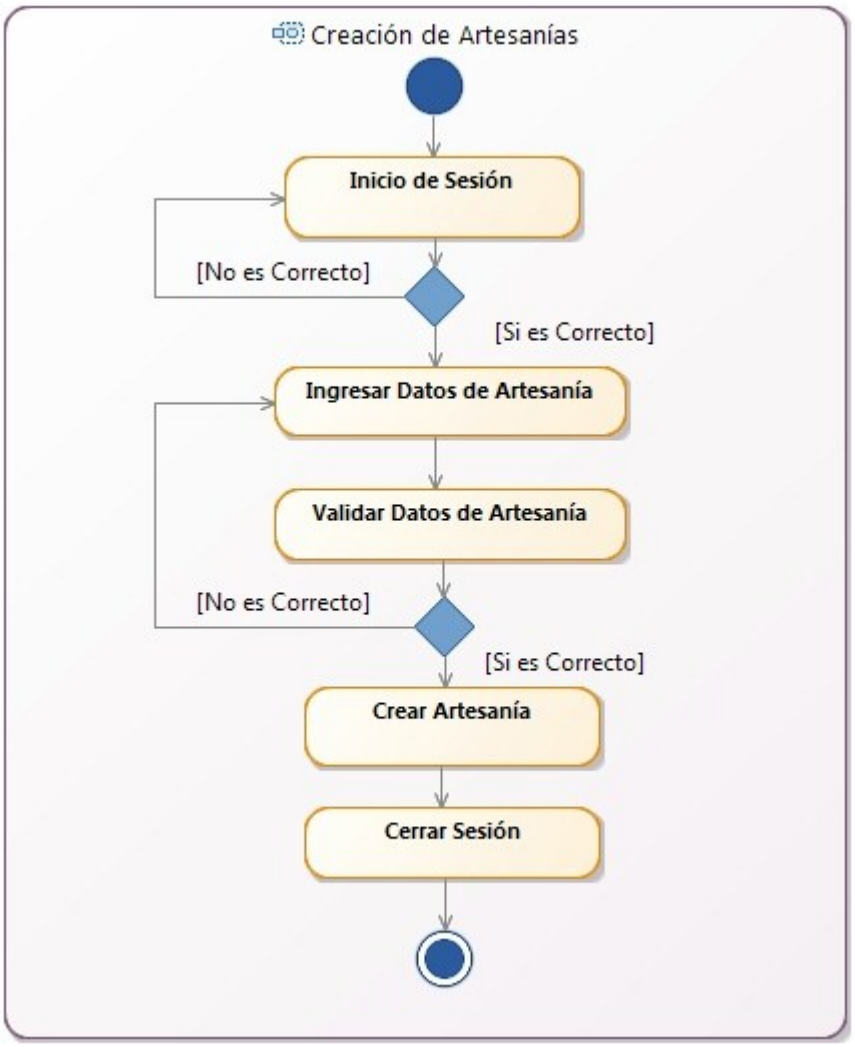


Figura 5.7: Diagrama de Actividades: Creación de Artesanía

Fuente: Propia

5.2.2.3 Diagramas de Secuencias

5.2.2.3.1 Comprar Artesanía

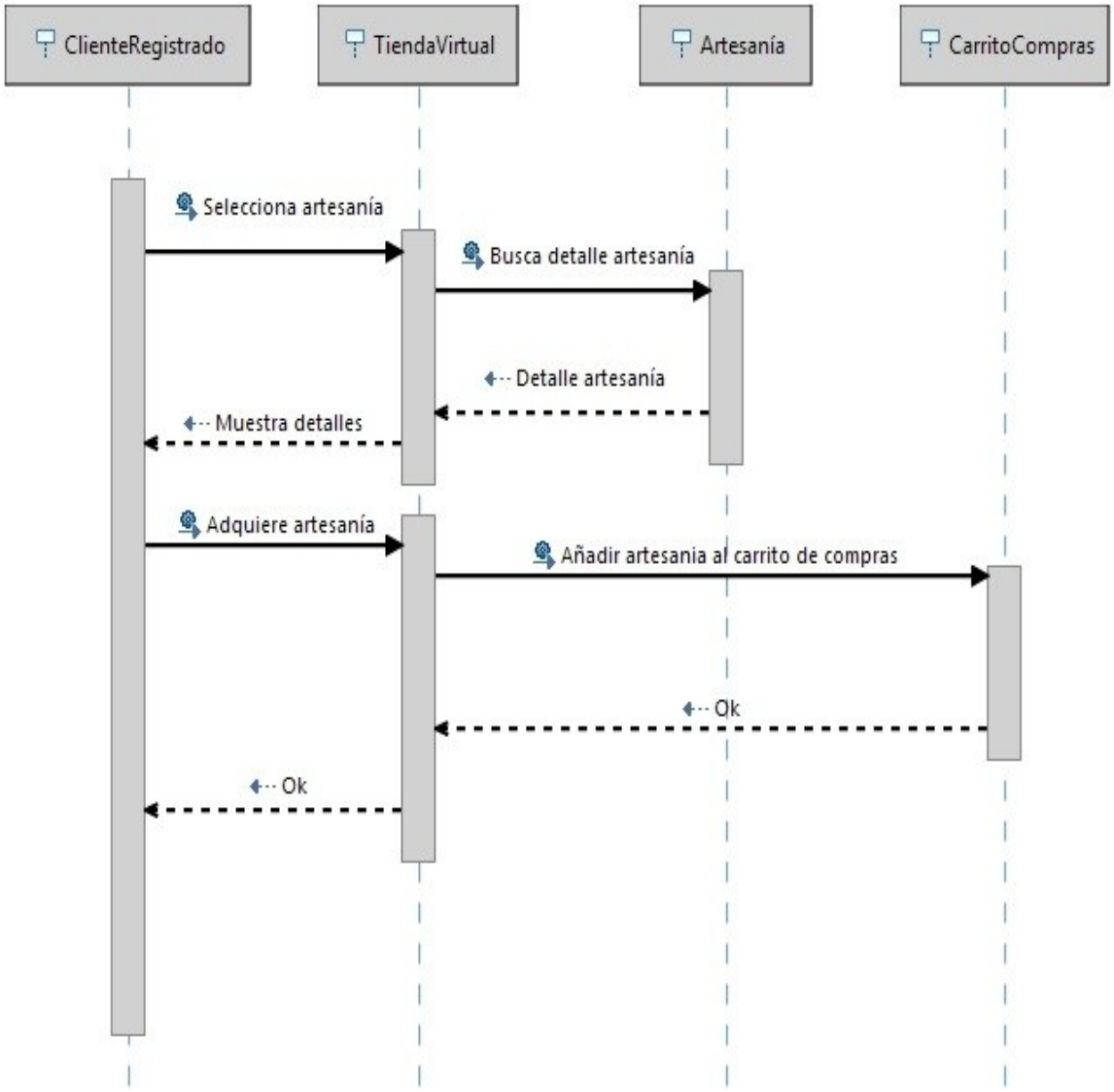


Figura 5.8: Diagrama de Secuencia: Comprar Artesanía

Fuente: **Propia**

5.2.2.3.2 Ingresar Artesanía

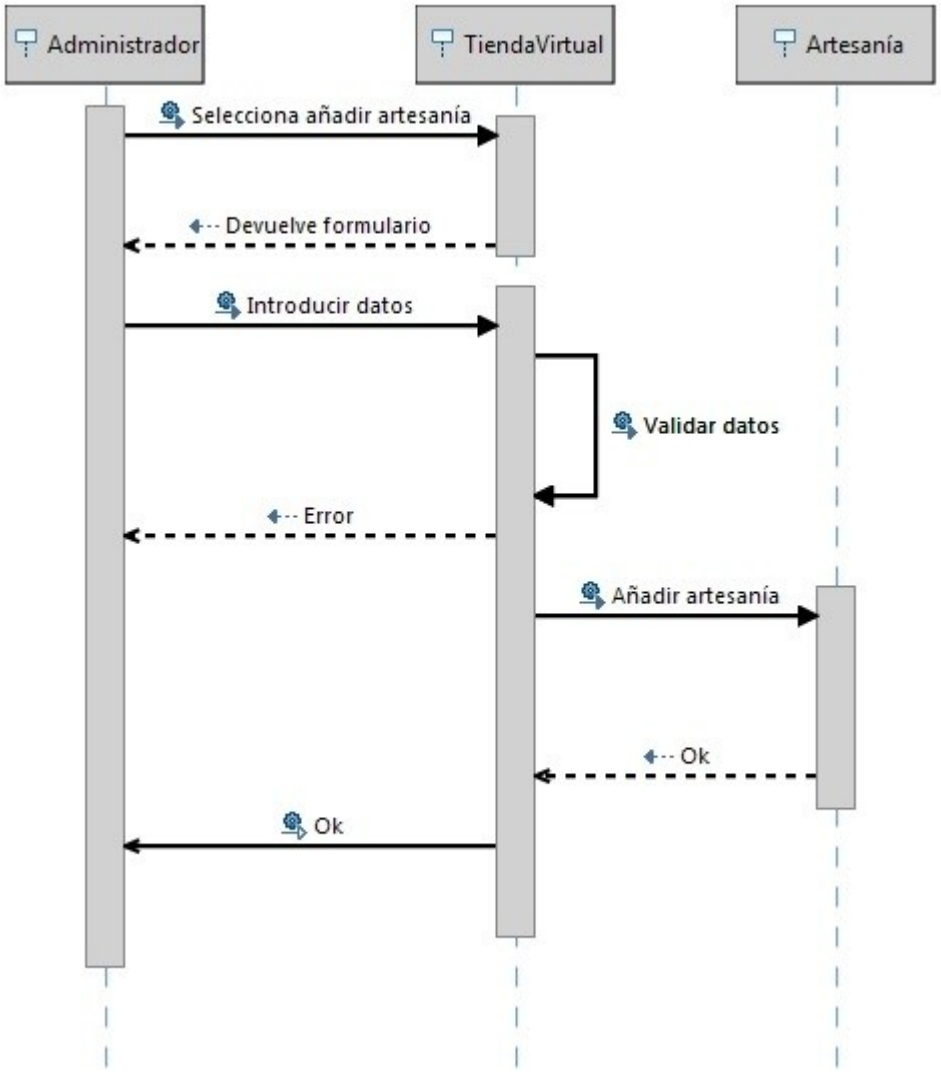


Figura 5.9: Diagrama de Secuencia: Ingresar Artesanía

Fuente: Propia

CAPÍTULO VI

6.1 FASE DE TRANSICIÓN

6.1.1 IMPLEMENTACIÓN DE LA TIENDA VIRTUAL “SANANTONIOSTORE”

6.1.1.1 INSTALACIÓN DE LA APLICACIÓN

Para el funcionamiento del sistema es necesario:

Java

Versión máquina virtual 1.6 (jdk-7u15-windows-x64.exe)

Creación de variables de entorno

JAVA_HOME: C:\java\jsdk16

PostgreSQL

Version postgresql-9.1.3-1-windows.exe

Figura 6.1: Instalación de PostgreSql

Fuente: Propia

Se puede cambiar la ruta de instalación o dejar la ruta por defecto; hacer clic en el botón siguiente.

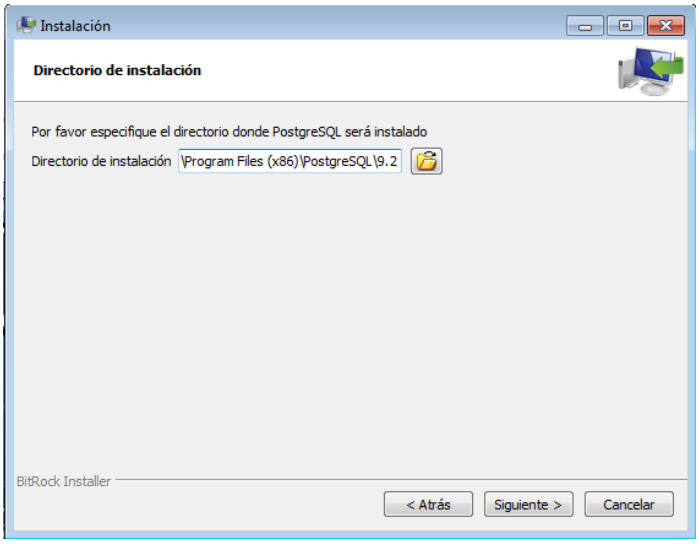


Figura 6.2: Instalación de PostgreSql, selección del directorio

Fuente: Propia

En la dirección de registro se puede darle una nueva ubicación de instalación, una vez direccionada la instalación se hace clic en el botón Siguiente.

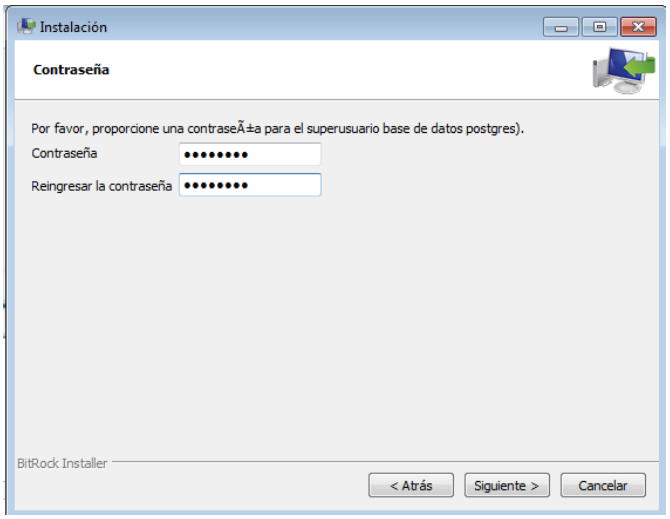


Figura 6.3: Ingreso de clave de usuario PostgreSql

Fuente: Propia

Aquí se agrega una contraseña la cual es manejada para cualquier conexión desde una aplicación; hacer clic en el botón siguiente.

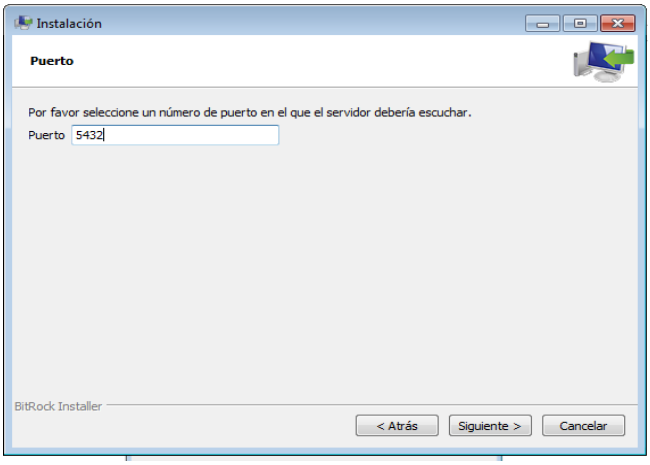


Figura 6.4: Puerto por default de PostgreSql

Fuente: **Propia**

Aquí se especifica el puerto que en este caso le dejara el mismo (5432); hacer clic en el botón siguiente para finalizar la instalación.

Driver jdbc

Versión pgJDBC V9.3-1100-1.

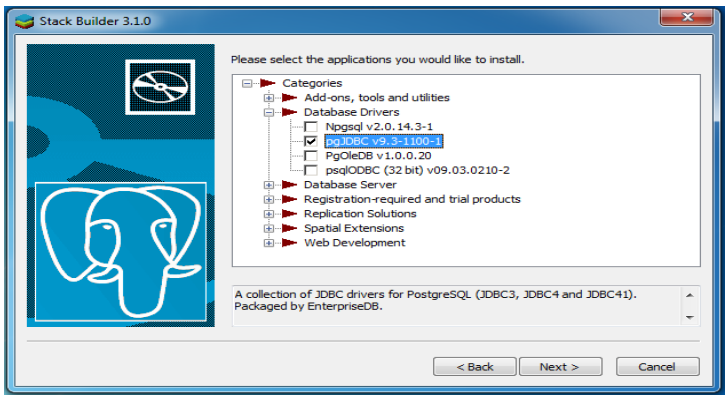


Figura 6.5: Driver JDBC de PostgreSql

Fuente: Propia

Configuración de las variables de entorno (Variables de sistema)

Comprobación: Abrir consola cmd e ingresar ***java -version***

ANT_HOME	C:\apache-ant-165
JAVA_HOME	C:\java\jsdk17
Path	;%JAVA_HOME%\bin;%ANT_HOME%\bin;
JBOSS_HOME	C:\jboss-as-711\bin
Path	;%JBOSS_HOME%\bin;

Instalación y configuración del servidor de aplicaciones JBoss-AS 7.1

Descargar el archivo empaquetado jboss-as-7.1.1.Final.zip de la página oficial.

Descomprimir y colocar la carpeta en la ruta indicada en JBOSS_HOME

Crear el usuario administrador para el servidor JBOSS 7.1

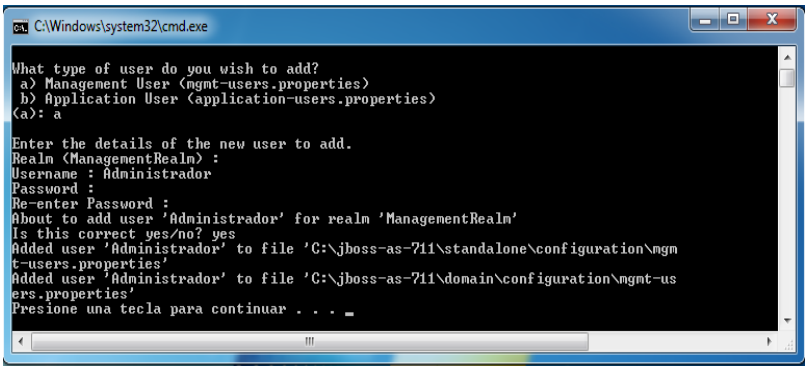


Figura 6.6: Creación Usuario JBoss

Fuente: Propia

Crear rol usuario administrador para la base de datos "SanAntonioStore".

Login Roles: Administrador

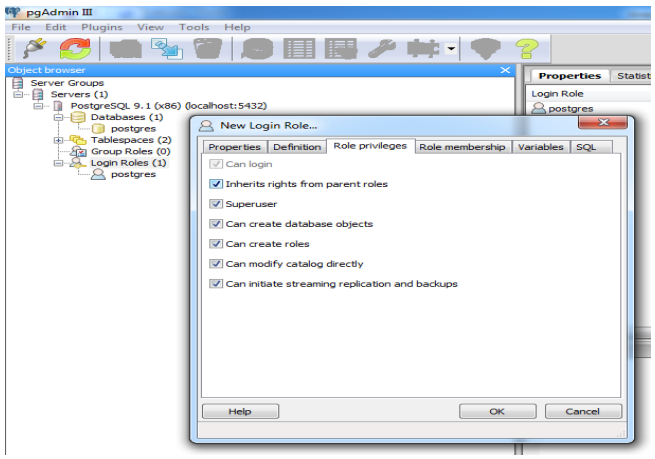


Figura 6.7: Creación de Rol Administrador en PostgreSql

Fuente: Propia

Creación Tablespace para los objetos del modelo de datos: “SanAntonioStores”

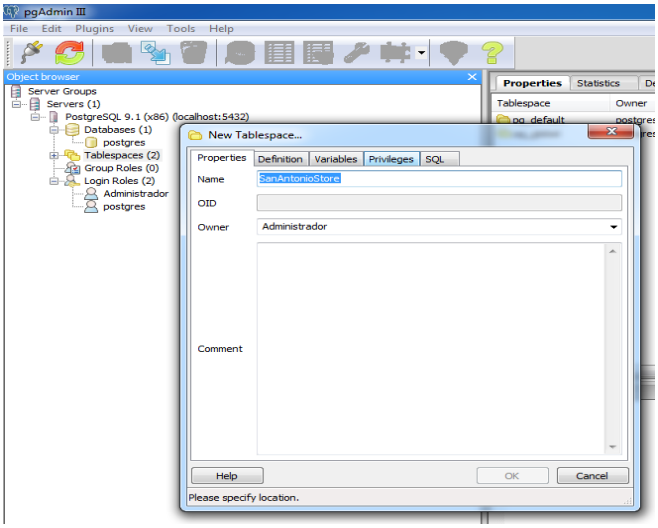


Figura 6.8: Creación de Tablespace “SanAntonioStore” en PostgreSql

Fuente: Propia

Creación de la Base de datos: “SanAntonioStore”

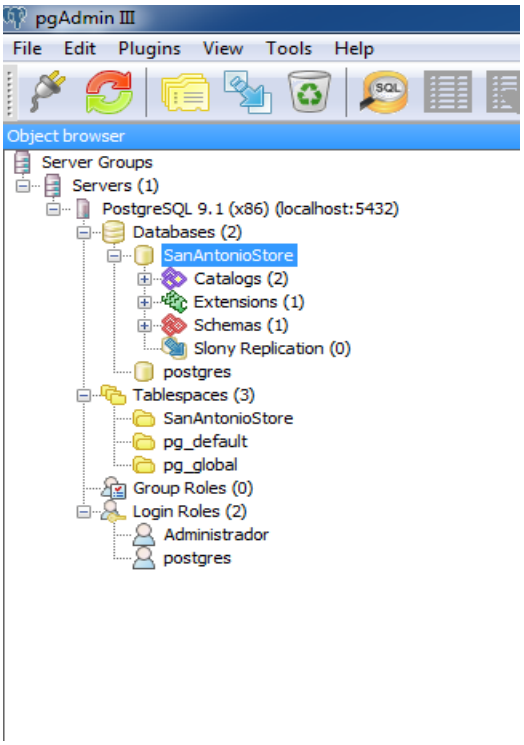


Figura 6.9: Creación de la base de datos “SanAntonioStore” en PostgreSql

Fuente: Propia

Crear el datasource de PostgreSQL y Drools en jboss 7

Crear un directorio C:\jboss-as-711\modules\org\postgresql\main y se copia el archivo postgresql-9.1-901.jdbc4.jar y luego se crea el archivo module.xml e ingresa lo siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
        <resources>
                <resource-root path="postgresql-9.1-901.jdbc4.jar" />
        </resources>
        <dependencies>
                <module name="javax.api" />
                <module name="javax.transaction.api" />
        </dependencies>
</module>
```

Figura 6.10: Creación de DataSource de PostgreSQL en JBoss 7

Fuente: **Propia**

Crear un directorio C:\jboss-as-711\modules\org\ drools\main y se copia los archivos drools-distribution-5.5.0.Final y luego se crea el archivo module.xml e ingresar lo siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="org.drools">
    <resources>
        <resource-root path="antlr-2.7.7.jar"/>
        <resource-root path="antlr-3.3.jar"/>
        <resource-root path="antlr-runtime-3.3.jar"/>
        <resource-root path="bcmail-jdk14-138.jar"/>
        <resource-root path="bcprov-jdk14-138.jar"/>
        <resource-root path="dom4j-1.6.1.jar"/>
        <resource-root path="drools-clips-5.5.0.Final.jar"/>
        <resource-root path="drools-compiler-5.5.0.Final.jar"/>
        <resource-root path="drools-core-5.5.0.Final.jar"/>
        <resource-root path="drools-decisiontables-5.5.0.Final.jar"/>
        <resource-root path="droolsjbpm-introduction-docs-5.5.0.Final.jdocbook"/>
        <resource-root path="drools-jsr94-5.5.0.Final.jar"/>
        <resource-root path="drools-persistence-jpa-5.5.0.Final.jar"/>
        <resource-root path="drools-templates-5.5.0.Final.jar"/>
        <resource-root path="drools-verifier-5.5.0.Final.jar"/>
        <resource-root path="ecj-3.5.1.jar"/>
        <resource-root path="guava-r06.jar"/>
        <resource-root path="hibernate-jpa-2.0-api-1.0.1.Final.jar"/>
        <resource-root path="itext-2.1.2.jar"/>
        <resource-root path="javassist-3.14.0-GA.jar"/>
        <resource-root path="jsr94-1.1.jar"/>
        <resource-root path="jta-1.1.jar"/>
        <resource-root path="jxl-2.6.10.jar"/>
        <resource-root path="knowledge-api-5.5.0.Final.jar"/>
        <resource-root path="knowledge-internal-api-5.5.0.Final.jar"/>
        <resource-root path="log4j-1.2.14.jar"/>
```

Figura 6.11: Creación de Drools en JBoss 7

Fuente: Propia

En el archivo standalone.xml ubicado en el directorio C:\jboss-as-711\standalone\configuration insertar el siguiente código dentro de la etiqueta <datasources>

```
<datasource jta="true" jndi-name="java:jboss/datasources/SanAntonioStoreDS" pool-name="SanAntonioStoreDS"
            enabled="true" use-java-context="true" use-ccm="true">
    <connection-url>jdbc:postgresql://localhost:5432/SanAntonioStore</connection-url>
    <driver-class>org.postgresql.Driver</driver-class>
    <driver>org.postgresql</driver>
    <security>
        <user-name>administrador</user-name>
        <password>admin</password>
    </security>
    <validation>
        <validate-on-match>false</validate-on-match>
        <background-validation>false</background-validation>
    </validation>
    <statement>
        <share-prepared-statements>false</share-prepared-statements>
    </statement>
```

Figura 6.12: Configuración DataSource de PostgreSQL en Drools en JBoss 7

Fuente: Propia

Y luego este código en la siguiente etiqueta <drivers>

```
<driver name="org.postgresql" module="org.postgresql">
    <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
</driver>
```

Figura 6.13: Configuración Driver de PostgreSQL en Drools en JBoss 7

Fuente: Propia

6.1.1.2 ESPECIFICACIONES DE CASOS DE PRUEBAS

Caso de prueba: Crear nueva Categoría para artesanías

Descripción breve

El objetivo de la prueba es que el administrador agregue una nueva categoría de artesanía.

Comprobar la creación de una nueva categoría

Ingresar al menú de Administrador en Categorías opción Crear Nueva Categoría de Artesanía; ingrese los datos de la nueva categoría.

Condiciones de ejecución

Para el ingreso de los datos de la nueva categoría de artesanía debe tener permisos de Administrador.

Entrada

- Se introduce “Madera” en el campo Categoría
- Se introduce “Madera de Cedro” en el campo Descripción
- Seleccionar “S” en el campo Estado

Resultado esperado

El sistema ingreso correctamente los datos de la nueva categoría y retorna automáticamente a la misma interfaz para volver a ingresar otra categoría de artesanías.

Evaluación de la prueba

Prueba superada con éxito.

Caso de prueba: Crear Nuevo Artesano

Descripción breve

El objetivo de la prueba es que el administrador agregue un nuevo artesano.

Comprobar la creación del Nuevo Artesano

Ingresar al menú de Administrador en Artesano opción Crear Nuevo Artesano; ingrese los datos del nuevo artesano.

Condiciones de ejecución

Para el ingreso de los datos del nuevo artesano debe tener permisos de Administrador.

Entrada

- Se introduce “1002492567” en el campo Documento Identificación
- Se introduce “Endara” en el campo Apellidos
- Se introduce “Juan” en el campo Nombres
- Se introduce “Ebanista” en el campo Título Artesanal
- Se introduce “Luís Enrique Cevallos” en el campo Dirección
- Se introduce “062550004” en el campo Teléfono
- Se introduce “jendara@hotmail.com” en el campo Email
- Se introduce “Minorista” en el campo Categoría
- Se introduce “01/10/2014” en el campo Fecha Creación
- Seleccionar “S” en el campo Estado Artesano

Resultado esperado

El sistema ingreso correctamente los datos del nuevo artesano y retorna automáticamente a la misma interfaz para volver a ingresar otro artesano.

Evaluación de la prueba

Prueba superada con éxito.

Caso de prueba: Crear Nueva Artesanía

Descripción breve

El objetivo de la prueba es que el administrador agregue una nueva Artesanía.

Comprobar la creación y acceso de la Nueva Artesanía

Ingresar al menú de Administrador en Artesanía opción Crear Nueva Artesanía; ingrese los datos de la nueva artesanía.

Condiciones de ejecución

Para el ingreso de los datos de la nueva artesanía debe tener permisos de Administrador.

Entrada

- Se introduce “Repisa” en el campo Nombre
- Se introduce “Repisa de Cedro” en el campo Descripción
- Se introduce “23/10/2014” en el campo Creación
- Se introduce “30 cm” en el campo Tamaño
- Se introduce “2 lb” en el campo Peso
- Se introduce “Madera de Cedro” en el campo Tipo Material
- Se introduce “10u x semana” en el campo Tiempo Elaboración Por Unidad
- Se introduce “Lacado al natural” en el campo Tipo Acabado
- Se introduce “30” en el campo Costo
- Se introduce “30 cm” en el campo Tamaño
- Seleccionar “Decorativa” en el campo Tipo Artesanía
- Se introduce “1 año” en el campo Tiempo Garantía
- Se introduce “No exponer al sol” en el campo Observaciones
- Se introduce “Repisa.jpg” en el campo Fotografía
- Se introduce “10” en el campo Existencias
- Se introduce “0” en el campo Utilidad
- Seleccionar “N” en el campo Oferta
- Seleccionar “S” en el campo Estado ***Categoría***
- Seleccionar “Madera” en el campo IdCategoria

Resultado esperado

El sistema ingreso correctamente los datos de la nueva artesanía y retorna automáticamente a la misma interfaz para volver a ingresar otra artesanía.

Evaluación de la prueba

Prueba superada con éxito.

Caso de prueba: Crear Nuevo Usuario

Descripción breve

El objetivo de la prueba es que el administrador agregue un nuevo Usuario.

Comprobar el Ingreso del Nuevo Usuario

Ingresar al menú de Administrador en Artesanía opción Crear Nuevo Usuario; ingrese los datos del nuevo usuario.

Condiciones de ejecución

Para el ingreso de los datos del nuevo usuario debe tener permisos de Administrador.

Entrada

- Se introduce “juanperez” en el campo Username.
- Se introduce “********” en el campo Pass.
- Seleccionar “Cliente” en el campo Rol.
- Se introduce “23/10/2014” en el campo Último Acceso
- Se introduce “23/10/2014” en el campo Fecha Alta.
- Seleccionar “S” en el campo Estado Usuario.

Resultado esperado

El sistema ingreso correctamente los datos del nuevo usuario y retorna automáticamente a la misma interfaz para volver a ingresar otro usuario.

Evaluación de la prueba

Prueba superada con éxito.

Caso de prueba: Crear nuevo Cliente

Descripción breve

El objetivo de la prueba es que el administrador agregue un nuevo Cliente.

Comprobar el Ingreso del Nuevo Cliente

Ingresar al menú de Administrador en Cliente opción Crear Nuevo Cliente; ingrese los datos del nuevo cliente.

Condiciones de ejecución

Para el ingreso de los datos del nuevo cliente debe tener permisos de Administrador.

Entrada

- Se introduce “1002492567” en el campo Documento Identificación
- Se introduce “López” en el campo Apellidos
- Seleccionar “Anita” en el campo Nombres.
- Se introduce “Av. El Inca y Los Nogales E14-135” en el campo Dirección
- Se introduce “022551382” en el campo Teléfono.
- Se introduce “alopez@hotmail.com” en el campo Email
- Se introduce “Mayorista” en el campo Categoría
- Se introduce “Quito” en el campo Ciudad
- Se introduce “Pichincha” en el campo Provincia
- Se introduce “Ecuador” en el campo País
- Se introduce “CodigoPostal” en el campo Código Postal.
- Seleccionar “S” en el campo Estado Cliente.
- Seleccionar “anitalopez” en el campo Estado IdUsuario.

Resultado esperado

El sistema ingreso e imprimió correctamente los datos y retorna automáticamente a la interfaz de formularios declarados listos para pago y actualiza el estado “declarado” al estado “pagado”, cambio de color de bandera de rojo a verde.

Evaluación de la prueba

Prueba superada con éxito.

CAPÍTULO VII

7.1 CONCLUSIONES Y RECOMENDACIONES

7.1.1 CONCLUSIONES

- El desarrollo de esta aplicación ayudará a la Empresa ENCHAPES ESPECIALES productores de accesorios para decoración de muebles a llevar de manera ordenada y sistematizada las transacciones comerciales y económicas que se llevan dentro de la misma.

- La integración de las herramientas y Framework que se utilizó en el desarrollo de la aplicación se logra gracias al patrón de diseño Modelo-Vista-Controlador para aplicaciones Web conjuntamente con la plataforma Java EE 6, que permite identificar claramente los Frameworks en cada capa.

- Se comprobó que la utilización del motor de reglas de producción DROOLS cumple con la función de desacoplar la capa de negocio de las demás capas del aplicativo, por lo que las reglas de negocio que rigen la Tienda Virtual están almacenados en un archivo de texto con exención DRL que utiliza el motor de inferencia.

- Se diseñó una aplicación de fácil manejo para el usuario para evitar cualquier tipo de error humano que pueda ocasionar pérdida de información.

- El modelamiento de las reglas de negocio utilizando la metodología de reglas propuesta, permite cumplir con el objetivo de separar, rastrear, exteriorizar, situar o localizar las reglas de negocio para cambios en el futuro sin la necesidad de compilar o desplegar la aplicación nuevamente si no a través del sistema de administración de reglas Guvnor.

- El modelamiento del aplicativo en RUP permite realizar actualizaciones en el futuro; el Sistema está diseñado para tal tarea, donde él o los administradores puedan hacer los cambios que se requieran como la implementación de otros módulos.

- Se Comprobó que el manejo de Drools 5.5 conjuntamente con Guvnor es satisfactorio al administrar la lógica de negocio, ya que se administra las reglas a través de una interfaz Web.

7.1.2 RECOMENDACIONES

- Debe ser meticuloso al momento de escoger los entornos de desarrollo integrado (IDEs) ya que esto facilita o dificulta la integración de las diferentes capas representados por los Frameworks que formaran parte de la aplicación.

- Recolectar la información de todo el contexto de la empresa para poder separar, rastrear, exteriorizar, situar o localizar el mayor número de reglas de negocio al inicio del desarrollo del sistema para evitar el rediseño de las reglas, la base de datos y del aplicativo.

- Que la Empresa auspiciante disponga de la infraestructura y conexiones necesarias para la implementación e instalación del Aplicativo, y así evitar contratiempos en la evaluación y pruebas del sistema; y en el acceso al sistema desde internet.

- Que para el inicio de la construcción de una nueva aplicación en la que se decida utilizar un motor de reglas optar por la versión Drools 6 ya que es mucho más transparente su integración.

- Es recomendable utilizar un motor de reglas de producción cuando el número de reglas de la lógica de negocio es compleja y numerosa.

- Se recomienda a las Autoridades de la Universidad y de la Facultad buscar convenios con otras instituciones para que los estudiantes y futuros profesionales apliquen sus conocimientos y al mismo tiempo tengan más oportunidades en el ambiente laboral.

- A nuestros docentes, es importante fomentar la investigación de nuevas tecnologías ya que con estas se puede llegar a soluciones más óptimas y rápidas.

GLOSARIO DE TÉRMINOS

Drools (o JBoss Rules): Es un sistema de gestión de reglas de negocio (BRMS, por las siglas en inglés de Business Rule Management System).

JSR-94: Estándar para motores de reglas de negocio en Java.

JCR: Drools usa JCR (JackRabbit) para gestionar el repositorio de reglas.

Guvnor: Es un proyecto open source de la JBoss Community. Forman parte de los productos para empresa JBoss Enterprise BRMS, gracias a que han alcanzado un grado de madurez adecuado.Componentes de la versión JBoss Community. Es un repositorio centralizado para Drools Knowledge Bases.

Inferencias: Una inferencia es una declaración completa que prueba condiciones y al encontrarlos verdadero, establece la verdad de un hecho nuevo.

Motor de reglas: Tiene como base "La Representación de Conocimiento y el Razonamiento".La Representación de Conocimiento y el Razonamiento (KRR) se trata como se representa el conocimiento en forma simbólica, o sea cómo se describe algo.

Repositorio de reglas: Un repositorio de reglas permite centralizar la administración de dichas reglas en la que se podrá crear, modificar, eliminar, manejar sus versiones y su almacenamiento será seguro.

BIBLIOGRAFÍA

ARISTASUR. (28 de Septiembre de 2012). *Sistema de Coordenadas Geográficas: Longitud y Latitud*. Obtenido de http://www.aristasur.com/contenido/sistema-de-coordenadas-geograficas-longitud-y-latitud

Azuanet Soluciones Web. (s.f.). *azuanet*. Obtenido de azuanet: http://www.azuanet.com/ampliar-287-noticias-10-razones-para-tener-una-tienda-online-vender-por-internet.html

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (pág. 32). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (pág. 12). John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (pág. 13). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (pág. 15). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (pág. 15). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (pág. 16). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (págs. 20-21). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (págs. 20-21). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (págs. 33-35). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (págs. 33-35). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (pág. 36). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (pág. 36). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (págs. 64-67). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (pág. 68). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (págs. 68-70). New York: John Wiley & Sons, Inc.

Barbara Von Halle. (2002). Business Rules Applied. En Barbara Von Halle, *Business Rules Applied* (pág. 71). New York: John Wiley & Sons, Inc.

Barbara, V. H. (2002). Business Rules Applied. En V. H. Barbara, *Business Rules Applied* (pág. 13). New York: John Wiley & Sons, Inc.

Blogger de maygua SIG. (4 de Noviembre de 2012). *Sistemas de Información Maygualida*. Obtenido de http://sigmaygualida.blogspot.com/2012/11/sistemas-de-informacion-geografica.html

Braun Heiko. (17 de Mayo de 2013). *AS71 Admin Guide*. Obtenido de https://docs.jboss.org/author/display/AS71/Admin+Guide

Bryan Rodriguez. (20 de 11 de 2013). *Arquitectura e implementación de PostgreSQL 9.3*. Obtenido de http://es.slideshare.net/bjhp90/arquitectura-e-implementacin-de-postgresql-93

Carlos Alberto Mejía Castelo. (2011). INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO. En Carlos Alberto Mejía Castelo, *INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO* (pág. 15). Santiago de Cali.

Carlos Alberto Mejía Castelo. (2011). INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO. En Carlos Alberto Mejía Castelo, *INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO* (pág. 15). Santiago de Cali.

Carlos Alberto Mejía Castelo. (2011). INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO. En Carlos Alberto Mejía Castelo, *INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO* (pág. 16). Santiago de Cali.

Carlos Alberto Mejía Castelo. (2011). INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO. En Carlos Alberto Mejía Castelo, *INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO* (pág. 19). Santiago de Cali.

Carlos Alberto Mejía Castelo. (2011). INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO. En Carlos Alberto Mejía Castelo, *INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO* (págs. 20-21). Santiago de Cali.

Carlos Alberto Mejía Castelo. (2011). INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO. En Carlos Alberto Mejía Castelo, *INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO* (págs. 21-22). Santiago de Cali.

Carlos Alberto Mejía Castelo. (2011). INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO. En Carlos Alberto Mejía Castelo, *INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO* (pág. 28). Santiago de Cali.

Castelo Carlos Alberto Mejía. (2011). INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO. En Castelo Carlos Alberto Mejía, *INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO* (pág. 23).

Castelo Carlos Alberto Mejía. (2011). INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO. En Castelo Carlos Alberto Mejía, *INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO* (pág. 26). Santiago de Cali.

Castelo Carlos Alberto Mejía. (2011). INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO. En Castelo Carlos Alberto Mejía, *INVESTIGACIÓN PARA LA ADOPCIÓN Y USO DE LOS MOTORES DE REGLAS DE NEGOCIO* (pág. 23). Santiago de Cali.

CENEC_CARCHI. (2011). *Actividades Ecnómicas Principales Carchi.* Obtenido de http://www.uasb.edu.ec/UserFiles/381/File/CENEC_CARCHI.pdf

Chubut, C. (2012). *Conceptos geodésicos básicos*. Obtenido de http://dc340.4shared.com/doc/dWuFiz_d/preview.html

COOTAD. (06 de Octubre de 2011). *Impuestos Municipales.* Obtenido de http://www.captur.travel/web2011/informacion_juridica/documentos/tributario/impuestoSegunCOOTAD.pdf

COOTAD. (s.f.). *Impuestos Municipales.* . Obtenido de http://www.captur.travel/web2011/informacion_juridica/documentos/tributario/impuestoSegunCOOTAD.pdf

Dave W., M., Allan, K., & Steven D., N. (2005). *Fundamentos de PHP 5.* Madrid: Ediciones Anaya.

Diaz Flores, M. M. (6 de Marzo de 1996). *RATIONAL UNIFIED PROCESS & EXTREME PROGRAMMING.* Obtenido de http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf

Disclaimer. (2010). *OSGeoLive.* Obtenido de http://live.osgeo.org/es/overview/postgis_overview.html

Económico, C. N. (2011). *Actividades Ecnómicas Principales Carchi*. Obtenido de http://www.uasb.edu.ec/UserFiles/381/File/CENEC_CARCHI.pdf

Edgewall , S. (2009). *Map Guide Open Source*. Obtenido de http://trac.osgeo.org/mapguide/about

Elías López. (2014). *PostgreSQL*. Obtenido de http://sabd15n1.wikispaces.com/PostgreSQL

Elías López. (2014). *PostgreSQL*. Obtenido de http://sabd15n1.wikispaces.com/PostgreSQL

Esfinge, L. (23 de Mayo de 2013). *Geoserver*. Obtenido de http://geoserver.org/display/GEOSDOC/1+GeoServer+Architecture

Esfinge, L. (23 de Mayo de 2013). *GEOSERVER*. Obtenido de http://geoserver.org/display/GEOSDOC/1+GeoServer+Architecture

ESRI. (2002). *Que es ARCGIS.* Obtenido de http://downloads.esri.com/support/whitepapers/ao_/what-is-arcgis-spanish.pdf

ESRI. (22 de Febrero de 2011). *ArcGIS Resources*. Obtenido de http://resources.arcgis.com/es/help/getting-started/articles/026n0000000s000000.htm

Esri Proprietary , R. (1995-2014). *ESRI*.

Estrada, J. (1988). *Laboratorio de la Cartografía.* Trillas S.A.

Fundación Wikimedia. (20 de Mayo de 2013). *Callback (informática).* Obtenido de http://es.wikipedia.org/wiki/Callback_%28inform%C3%A1tica%29

Fundación Wikimedia. (12 de Marzo de 2013). *Licencia BSD*. Obtenido de http://es.wikipedia.org/wiki/Licencia_BSD

Fundación Wikimedia. (17 de Julio de 2013). *Log (registro).* Obtenido de http://es.wikipedia.org/wiki/Log_%28registro%29

Fundación Wikimedia. (23 de Septiembre de 2014). *Algoritmo Rete*. Obtenido de http://es.wikipedia.org/wiki/Algoritmo_Rete

Fundación Wikimedia. (8 de Mayo de 2014). *Bean.* Obtenido de http://es.wikipedia.org/wiki/Bean

Fundación Wikimedia. (10 de Octubre de 2014). *Drools*. Obtenido de http://es.wikipedia.org/wiki/Drools

Fundación Wikimedia. (16 de Octubre de 2014). *Framework*. Obtenido de http://es.wikipedia.org/wiki/Framework

Fundación Wikimedia. (11 de Octubre de 2014). *JavaServer Faces*. Obtenido de http://es.wikipedia.org/wiki/JavaServer_Faces

Fundación Wikimedia, I. (27 de Agosto de 2014). *JBoss.* Obtenido de http://es.wikipedia.org/wiki/JBoss

Fundación Wikimedia, I. (7 de Octubre de 2014). *Middleware.* Obtenido de http://es.wikipedia.org/wiki/Middleware

Fundación Wikimedia, I. (25 de Junio de 2014). *PrimeFaces.* Obtenido de http://es.wikipedia.org/wiki/PrimeFaces

Fundación Wikimedia, I. (6 de Mayo de 2014). *QGIS*. Obtenido de http://es.wikipedia.org/wiki/Quantum_GIS

Geomática en Latinoamérica. (09 de Octubre de 2013). Obtenido de http://latingeomatica.blogspot.com/

Hax Arnoldo C. (2005). *Dirección de operaciones en la empresa.* Editorial Hispano Europea.

Hopkins, J. (2014). *JSF 2 Tutorial Series.* Obtenido de http://www.coreservlets.com/JSF-Tutorial/jsf2/

IDE, L. (2014). *CONCETOS DE GIS.* Obtenido de http://latinide.cedia.org.ec/index.php/conceptos-sig

Jimpako. (2008). *Servidores Geográficos*. Obtenido de http://servidoresgeograficos.blogspot.com/2008/07/geodatabase.html

José Manuel Sánchez Suárez. (01 de Marzo de 2013). *Introducción a Guvnor*. Obtenido de http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=introduccionGuvnor

Juan José Meroño Sánchez. (2009). *Curso Java y Tecnologías Java EE.* Murcia: Plan Formación PAS 2009.

Karen Coral Rodríguez, J. I. (2004). *Manual de Gramática del Castellano.* Lima-Perú: PROEDUCA - GTZ.

Karen Coral Rodríguez, J. I. (2004). Manual de Gramática del Castellano. En J. I. Karen Coral Rodríguez, *Manual de Gramática del Castellano* (pág. 69). Lima-Perú: PROEDUCA - GTZ.

La Verdad Multimedia, S. (2010). *Cartografía.* Obtenido de http://www.atlasdemurcia.com/index.php/secciones/3/cartografia-actual-topografica-ortofotos-satelite/2/

Larousse. (2002). Diccionario Larousse . En Larousse, *Diccionario Larousse* (pág. 351). Larousse editorial.

Larousse. (2009). *thefreedictionary.com*. Obtenido de http://es.thefreedictionary.com/completitud

Larousse. (2014). *thefreedictionary.com.* Obtenido de http://es.thefreedictionary.com/dialectales

Larousse Editorial, S. (2009). *TheFreeDictionary* . Obtenido de TheFreeDictionary : http://es.thefreedictionary.com/incompletitud

LAROUSSE, D. (2001). *Diccionario LAROUSSE.* Editorial Larousse.

LLC, T. (2014). *Procesos de Software*. Obtenido de http://procesosdesoftware.wikispaces.com/METODOLOGIA+RUP

Marchioni Francesco. (2011). JBoss AS 7 Configuration, Deployment, and Administration. En Marchioni Francesco, *JBoss AS 7 Configuration, Deployment, and Administration* (pág. 19).

Marco David. (08 de Marzo de 2011). *Introducción a EJB 3.1 (I).* Obtenido de http://www.davidmarco.es/articulo/introduccion-a-ejb-3-1-i

Martinez, L. (MAyo de 2008). *GEODATABASE.* Obtenido de http://ocw.um.es/ciencias/sistema-d

Minesota, U. o. (2014). *Map Server 6.4.1*. Obtenido de http://mapserver.org/introduction.html#introduction

Montoya , J. (2012). Obtenido de http://www.actividadeseconomicas.org/2012/05/que-son-las-actividades-economicas.html#.U2BVq6KejPo

Montoya, J. D. (2012). *Actividades Ecnómicas*. Obtenido de http://www.actividadeseconomicas.org/2012/05/que-son-las-actividades-economicas.html#.U2BVq6KejPo

nm, b. (12). *mbn m.* sf: swrw.

Pacheco Aguila , Y. (8 de Febrero de 2007). *Ajax un nuevo acercamiento a las Aplicaciones Web*. Obtenido de http://www.monografias.com/trabajos43/ajax/ajax2.shtml#ixzz31tcJJzya

Palomar Vasquez, J. (2011). *Iniciación a GvSIG*. Obtenido de http://personales.upv.es/jpalomav/cursos/gvsig/gvsig_1_11/gvsig_1_11.html?t34.html

PostgreSQL, B. d. (Noviembre de 2012). *POSTGRESQL*. Obtenido de http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html%7D

Quiñonez, E. (16 de Octubre de 2009). *PostgreSQL como Funciona una base de Datos por Dentro.* Obtenido de https://wiki.postgresql.org

Regents of the University, M. (2014). *MapServer 6.4.1 Documentation*. Obtenido de http://mapserver.org/introduction.html#introduction

Registro Oficial Nro 47. (2013). Registro Oficial Nro 47. En D. d. EL Gobierno Autónomo, *Registro Oficial* (págs. 35-40). Quito.

Sánchez Juan José Meroño. (2009). *Curso Java y Tecnologías Java EE.* Obtenido de http://u.jimdo.com/www38/o/s2e3cb89a232d3f68/download/m585343573579bc3b/1365710255/Curso+Java+y+J2EE.pdf

Sánchez Juan José Meroño. (2009). *Curso Java y Tecnologías Java EE.* Obtenido de http://u.jimdo.com/www38/o/s2e3cb89a232d3f68/download/m585343573579bc3b/1365710255/Curso+Java+y+J2EE.pdf

Sánchez Juan José Meroño. (2009). *Curso Java y Tecnologías Java EE.* Obtenido de http://u.jimdo.com/www38/o/s2e3cb89a232d3f68/download/m585343573579bc3b/1365710255/Curso+Java+y+J2EE.pdf

SIGMA. (2003). *Sistemas de Gestión.* Obtenido de http://www.ing.unlp.edu.ar/produccion/pp/sigma/hr/sig2.htm

The JBoss Drools team. (2009). *Drools Expert User Guide Version 5.5.0.Final*. Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (10 de Octubre de 2014). *Guvnor User Guide*. Obtenido de http://docs.jboss.org/drools/release/5.5.0.Final/drools-guvnor-docs/html_single/#d0e24

The JBoss Drools team. (s.f.). *Drools Expert User Guide.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de Drools Expert User Guide Version 5.5.0.Final: https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de Drools Expert User Guide Version 5.5.0.Final: https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de Drools Expert User Guide Version 5.5.0.Final: https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

The JBoss Drools Team. (s.f.). *Drools Expert User Guide Version 5.5.0.Final.* Obtenido de https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf

Thompson Ivan. (2012). *Portal de Mercadotecnia*. Obtenido de Portal de Mercadotecnia: http://www.promonegocios.net/internet-online/cobrar-internet.html

Thompson Ivan. (2012). *Promonegocios.net*. Obtenido de Promonegocios.net: http://www.promonegocios.net/venta/tiendas-virtuales.html

tutorialspoint.com. (2014). *JavaServer Faces (JSF) Tutorial.* Obtenido de http://www.tutorialspoint.com/jsf/

tutorialspoint.com. (2014). *JavaServer Faces (JSF) Tutorial.* Obtenido de http://www.tutorialspoint.com/jsf/

Universidad de Colombia, B. (2010). *Ecología del Paisaje y Modelación de Ecosistemas*. Obtenido de http://www.ciencias.unal.edu.co/unciencias/web/dependencia/?itpad=1423&niv=1&itact=1766&ti=false&itroot=1423&dep=33

Wikipedia. (2014). *Datum.* Obtenido de http://es.wikipedia.org/wiki/Datum

ANEXOS

ANEXO A: MANUAL DE ADMINISTRADOR

A.1 Introducción

Este manual está dirigido a los usuarios administradores que trabajarán en conjunto con la Tienda Virtual, y resume toda la funcionalidad del sistema para su correcto funcionamiento.

A.2 Objetivo

Proporcionar una guía de operación y manejo de las funciones de la Tienda Virtual 'SanAntonioStore'.

1. Ingreso a la Tienda Virtual "SanAntonioStore" como administrador

Para la administración del sistema se ingresa la siguiente URL.

http://localhost:8080/SanAntonioStore/ que muestra la página de inicio de la aplicación y se da un clic en el link ADMINISTRADOR en la parte inferior derecha; Este link redirección a la siguiente interfaz donde se ingresará el usuario y clave.

El usuario con rol Administrador puede acceder.

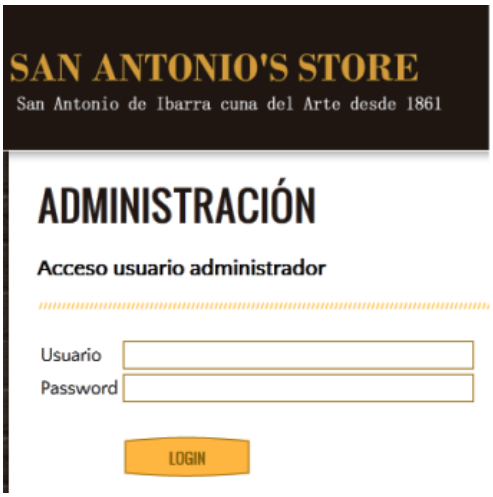


Figura 0.1: Interfaz de ingreso al Sistema para el usuario con rol Administrador
Fuente: Propia

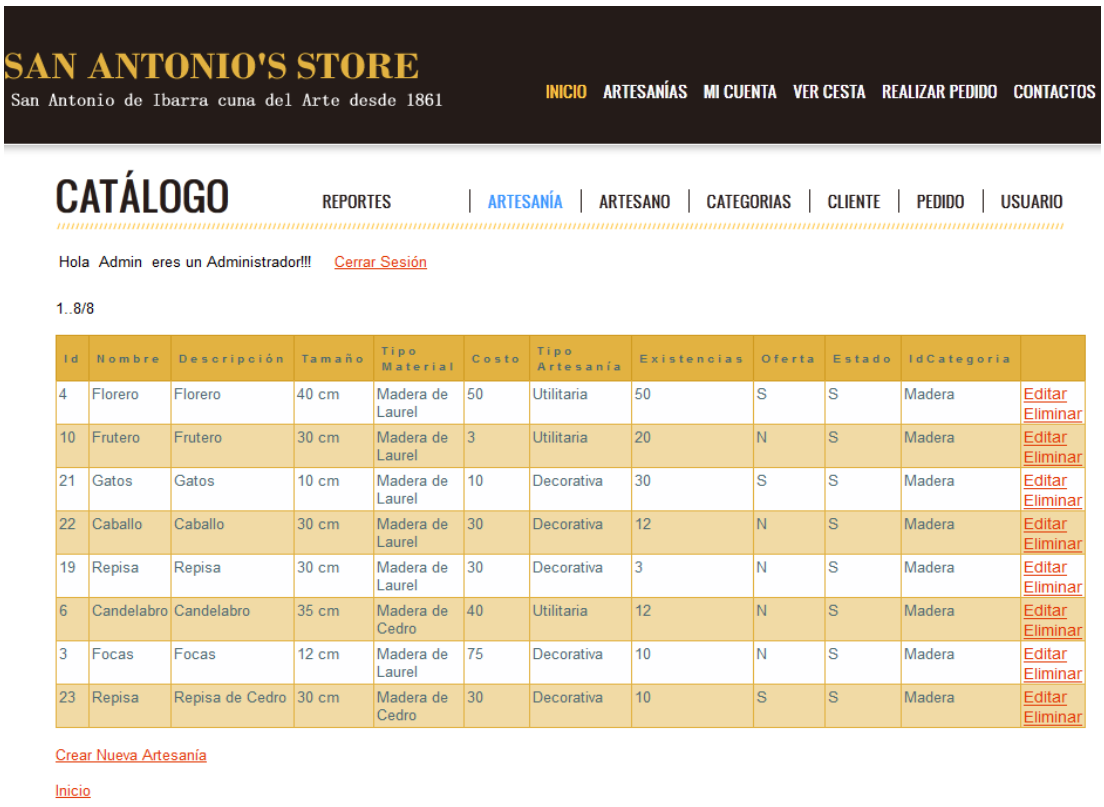
Dentro del Sistema se tiene el siguiente menú de administración.

Figura 0.2: Interfaz del menú de administrador del Sistema
Fuente: Propia

A Continuación vamos a detallar cada opción y como hacer la manipulación de datos.

1.1 Menú Principal

Es la interfaz de inicio del sistema, donde se muestra el catálogo para gestionar el modelo de datos. Se puede administrar artesanías, artesanos, categorías de artesanías, clientes, pedidos, usuarios y generar reportes.



Id	Nombre	Descripción	Tamaño	Tipo Material	Costo	Tipo Artesanía	Existencias	Oferta	Estado	IdCategoria	
4	Florero	Florero	40 cm	Madera de Laurel	50	Utilitaria	50	S	S	Madera	Editar Eliminar
10	Frutero	Frutero	30 cm	Madera de Laurel	3	Utilitaria	20	N	S	Madera	Editar Eliminar
21	Gatos	Gatos	10 cm	Madera de Laurel	10	Decorativa	30	S	S	Madera	Editar Eliminar
22	Caballo	Caballo	30 cm	Madera de Laurel	30	Decorativa	12	N	S	Madera	Editar Eliminar
19	Repisa	Repisa	30 cm	Madera de Laurel	30	Decorativa	3	N	S	Madera	Editar Eliminar
6	Candelabro	Candelabro	35 cm	Madera de Cedro	40	Utilitaria	12	N	S	Madera	Editar Eliminar
3	Focas	Focas	12 cm	Madera de Laurel	75	Decorativa	10	N	S	Madera	Editar Eliminar
23	Repisa	Repisa de Cedro	30 cm	Madera de Cedro	30	Decorativa	10	S	S	Madera	Editar Eliminar

Figura 0.3: Interfaz del Menú Principal
Fuente: Propia

Sub Menú:

Artesanía.- Se podrá crear, modificar, eliminar una artesanía con las siguientes propiedades:

- Nombre: Nombre descriptivo único de la artesanía.
- Descripción: Pequeña descripción significativa de la artesanía.
- Creación: Fecha de creación de la artesanía.
- Tamaño: Descripción de la altura, ancho, espesor
- Peso: Descripción del peso en libras
- Tipo Material: Descripción del material que ha sido elaborada la artesanía.
- Tiempo Elaboración Por Unidad: Descripción del tiempo de elaboración por unidades.
- Tipo Acabado: Tipo de acabado de la artesanía dependiendo del material ej. Lacado, pintado, dorado.
- Costo: Valor en dólares
- Tipo Artesanía: Se clasificará en 3 categorías: Decorativa, Utilitaria, Religiosa
- Tiempo Garantía: Tiempo de garantía de la artesanía en caso de problemas por el material de elaboración.
- Observaciones: Recomendaciones y cuidados de la artesanía después de la venta
- Fotografía: Imagen descriptiva de la artesanía
- Existencias: Número de unidades listas para la venta
- Utilidad: Un valor expresado en porcentaje
- Oferta: Si la artesanía tiene un descuento adicional.
- Estado: Referente a si esta en producción o no
- IdCategoria: Categoría a la q pertenece la artesanía

Artesano.- Se podrá crear, modificar, eliminar un artesano con las siguientes propiedades:

- Documento Identificación: Número de documento de identificación único para el artesano.
- Apellidos: Apellidos del artesano
- Nombres: Nombres del artesano
- Título Artesanal: Título artesanal por ej. Ebanista, escultor
- Dirección: Dirección donde se le puede localizar al artesano

- Teléfono: Número de teléfono para poder comunicar con él
- Email: Correo electrónico
- Categoría: De acuerdo al número de artesanías que elabore
- Fecha Creación: Fecha de registro del artesano
- Estado Artesano: Estado del artesano

Categoría de Artesanía.- Se podrá crear, modificar, eliminar una categoría artesanía con las siguientes propiedades:

- Categoría: Nombre único descriptivo que agrupará artesanías
- Descripción: Pequeña descripción de la categoría de artesanía
- Estado: Estado de la categoría.

Cliente.- Se podrá crear, modificar, eliminar un cliente con las siguientes propiedades:

- Documento Identificación: Número de documento de identificación único para el cliente.
- Apellidos: Apellidos del cliente
- Nombres: Nombres de cliente
- Dirección: Dirección válida para realizar la entrega del pedido
- Teléfono: Número de teléfono para poder comunicar con él
- Email: Correo electrónico
- Categoría: De acuerdo al número de artesanías que compre
- Ciudad: Ciudad donde vive el cliente
- Provincia: Provincia donde vive el cliente
- País: País donde vive el cliente
- Código Postal: Número de acuerdo al país donde vive el cliente
- Estado Cliente: Si está activo o no
- IdUsuario: Usuario registrado conjuntamente con el cliente

Pedido.- Se podrá modificar y anular un pedido con las siguientes propiedades:

- Id: Número identificador del pedido
- Fecha Pedido: Fecha cuando se realizó el pedido
- Nombre Embarque: Nombre del modo de transporte
- Dirección Pedido: Dirección del pedido a entregar
- Ciudad Pedido: Ciudad de entrega
- Provincia Pedido: Provincia de entrega

- País Pedido: País de entrega
- Importe: Valor total de la compra
- Estado Pedido: Estado si el pedido es anulado
- IdCliente: Identificador del pedido

Usuario.- Se podrá crear, modificar y anular un usuario con las siguientes propiedades:

- Username: Nombre que identifica al usuario del usuario tanto para administrar el modela de datos o realizar una compra
- Pass: Contraseña para ingresar al sistema
- Rol: Tipo de rol Administrador o Cliente
- Último Acceso: Ultima fecha de ingreso al sistema
- Fecha Alta: Fecha de registro en el sistema
- Estado Usuario: Estado del usuario si está activo o no

2. Ingreso al sistema de administración de reglas de negocio Guvnor Drools para administrar las reglas de negocio

Para ingresar al sistema se ingresa la siguiente URL.

http://localhost:8080/jboss-brms/ en la cual se podrá modificar las reglas de negocio en lenguaje nativo de Drools con extensión *.drl

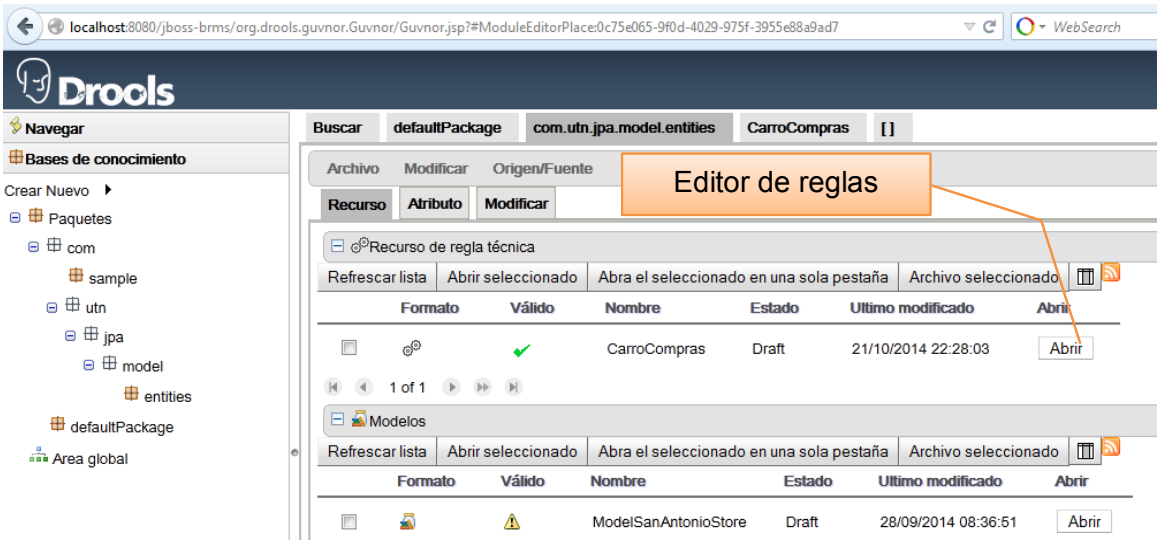


Figura 0.4: Interfaz de Guvnor para editar las reglas de negocio
Fuente: Propia

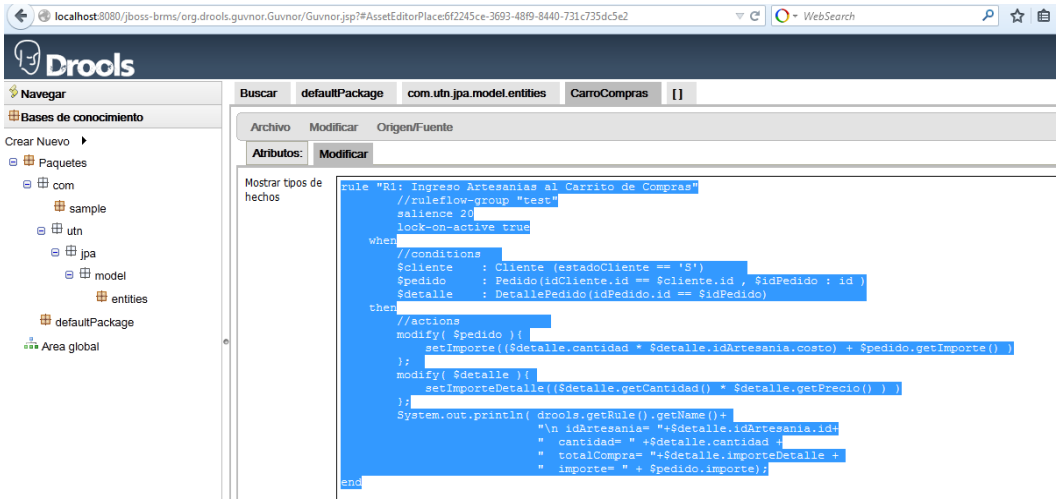


Figura 0.5: Interfaz de Guvnor mostrando una regla de negocio en lenguaje drl
Fuente: Propia

2 Reportes

Hacer clic en el menú principal en ***Reportes***, aquí podremos visualizar y filtrar por algunos campos dependiendo de las necesidades.

- Reporte de todas las compras.
- Reporte de artesanos
- Reporte de artesanías

ANEXO B: MANUAL PARA UTILIZAR EL CARRITO DE COMPRAS

B.1 Introducción

Este manual está dirigido a la o las personas que realizaran las compras en la Tienda Virtual.

B.2 Objetivo

Proporcionar una guía de manejo para realizar una compra en la Tienda Virtual "SanAntonioStore" como cliente registrado o visitante

Para ingresar al catálogo de artesanías ingresar la siguiente URL.

http://localhost:8080/SanAntonioStore/artesanias.xhtml que muestra la página de catálogo en el que se muestra las artesanías.

Figura 0.6: Interfaz del catálogo de la Tienda Virtual "SanAntonioStore"
Fuente: Propio

El usuario no registrado puede navegar por el catálogo de artesanías; también puede agregar, eliminar y aumentar ítems al carrito de compras.

Para confirmar la compra el usuario no registrado tiene que registrarse creando una sesión de compra.

Figura 0.7: Interfaz del carrito de compras (opción Ver Cesta)
Fuente: Propio

Para el usuario registrado ingresa con su usuario y contraseña

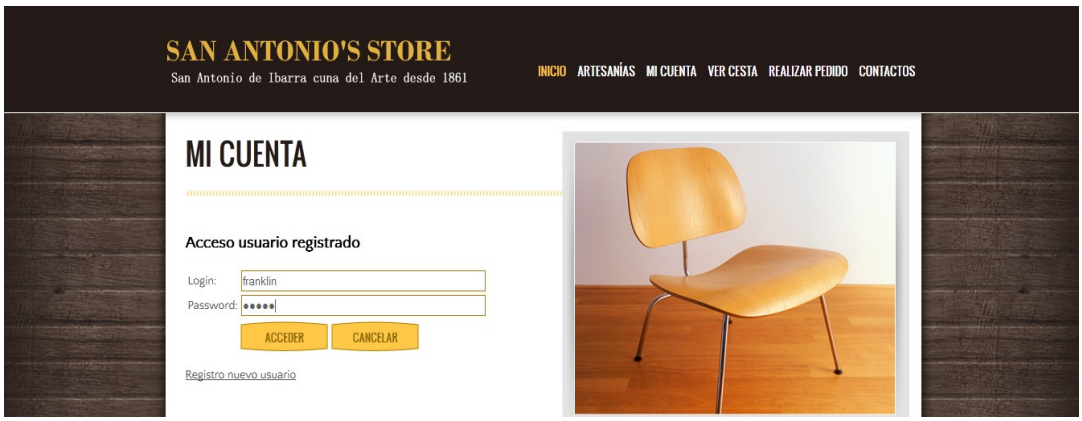


Figura 0.8: Interfaz del carrito de compras (opción Mi Cuenta)
Fuente: Propia

El usuario registrado o visitante puede ver el carrito de compras e incrementar cada uno de los ítems.

Figura 0.9: Interfaz del carrito de compras (opción Ver Cesta), incrementar ítems
Fuente: Propia

Cuando el usuario quiere confirmar el pedio lo hará a través del link REALIZAR PEDIDO

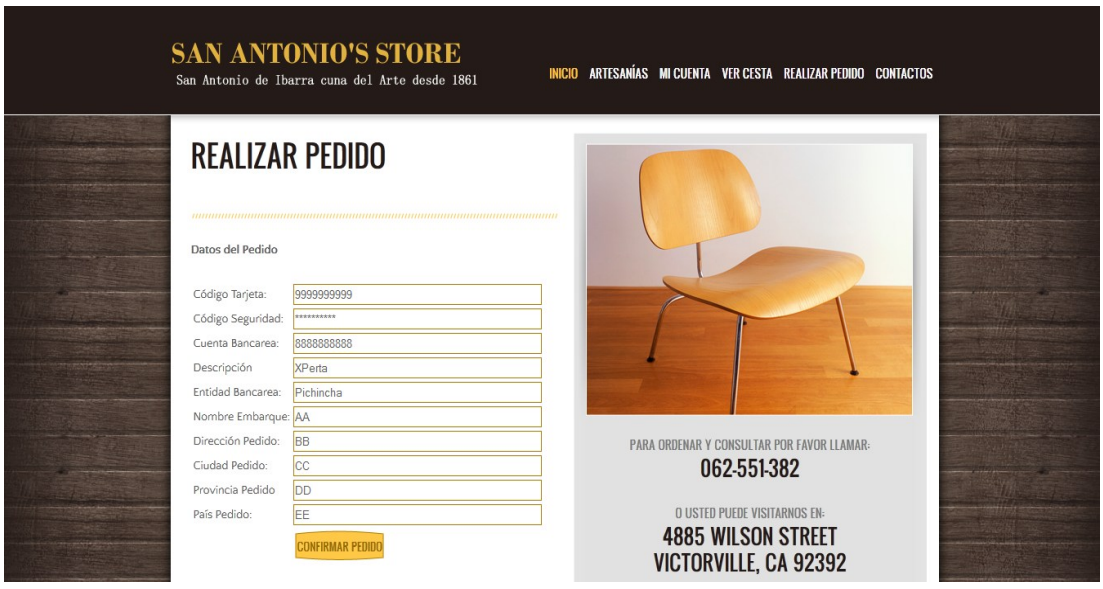


Figura 0.10: Interfaz del carrito de compras (opción Realizar Pedido)
Fuente: Propia

Pedido.- Se ingresará información sensible para confirmar el pedido en las siguientes propiedades:

- Código Tarjeta: Número valido de tarjeta de crédito
- Código Seguridad : Código sensible único y valido
- Cuenta Bancaria: Número valido de la cuenta bancaria
- Descripción: Descripción del tipo de documento
- Nombre Embarque: Nombre del tipo de transporte para entregar la compra
- Dirección Pedido: Dirección válida para realizar la entrega de la compra
- Ciudad Pedido: Ciudad de entrega de la compra
- Provincia Pedido: Provincia de entrega de la compra
- País Pedido: País de entrega de la compra

Una vez generado el pedido se muestra un resumen de la compra

Figura 0.11: Interfaz del carrito de compras (Pedido generado con Exito)
Fuente: Propia

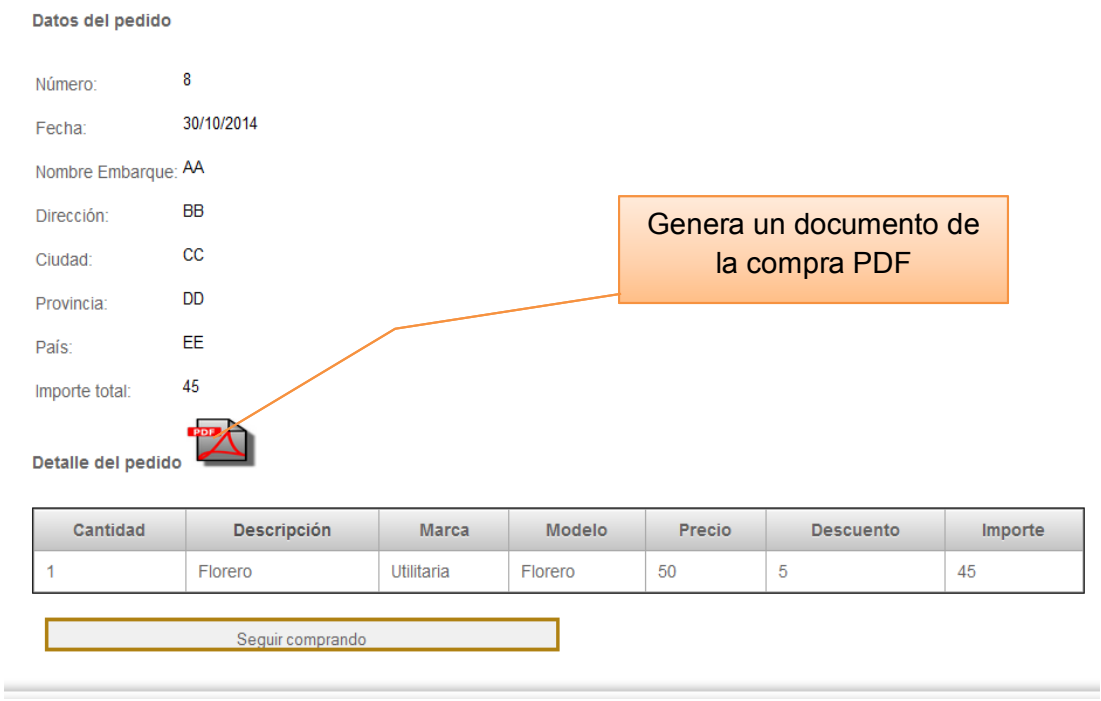


Figura 0.12: Interfaz del carrito de compras (Resumen de pedido completado)
Fuente: Propio

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.

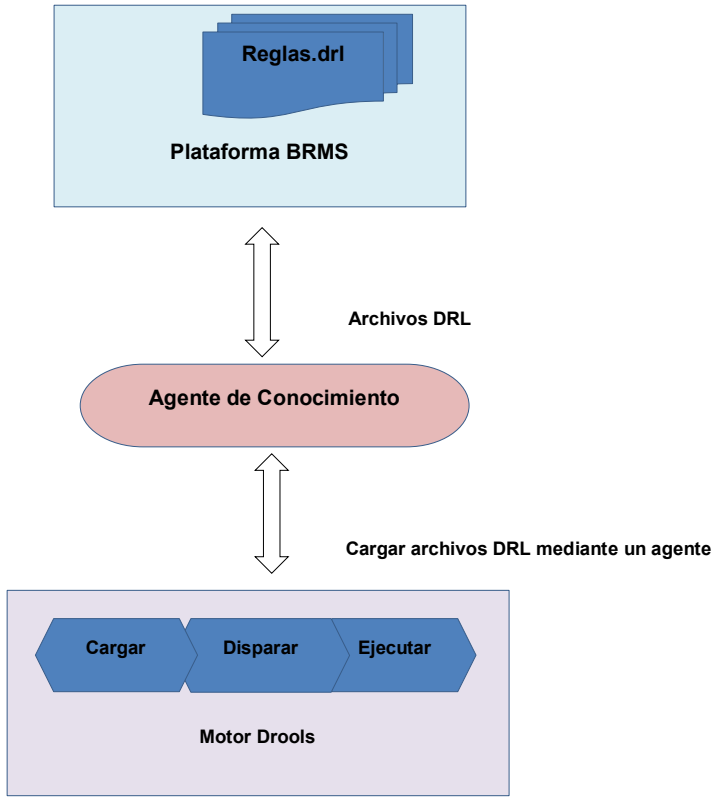


Figura 0.14: Esquema entre el motor de reglas y el BRMS
Fuente: Propia

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

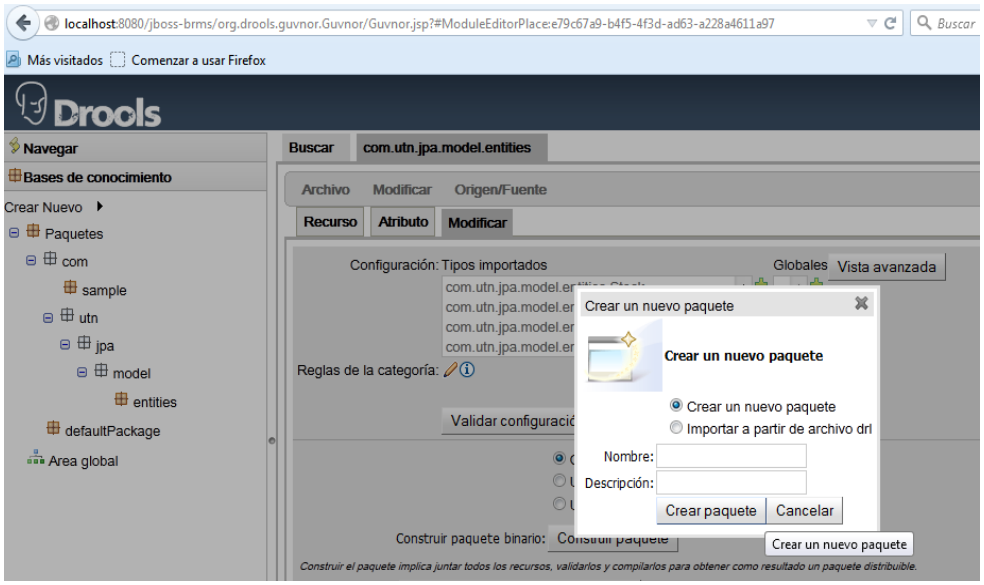


Figura 0.16: Creación de un paquete en GUVNOR
Fuente: Propia

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

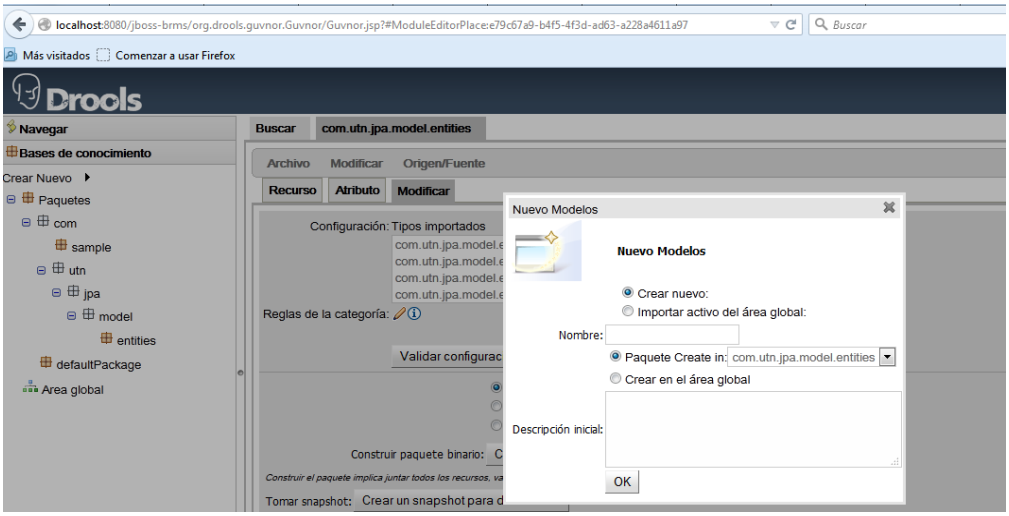


Figura 0.17: Creación de un modelo de clase en GUVNOR
Fuente: Propia

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

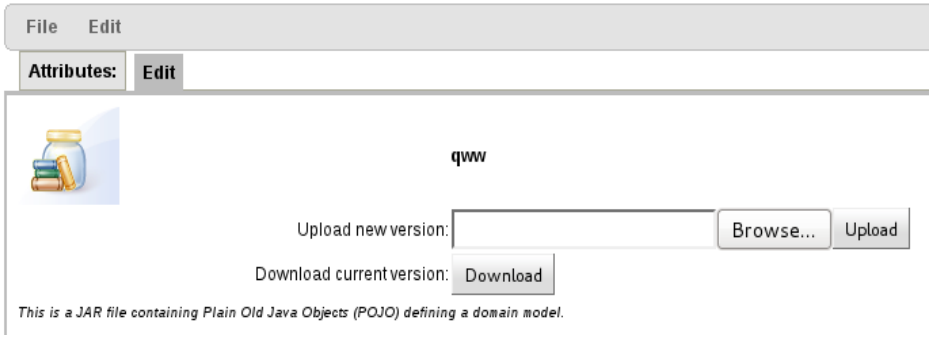


Figura 0.18: Importación del modelo de clases en GUVNOR
Fuente: Propia

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

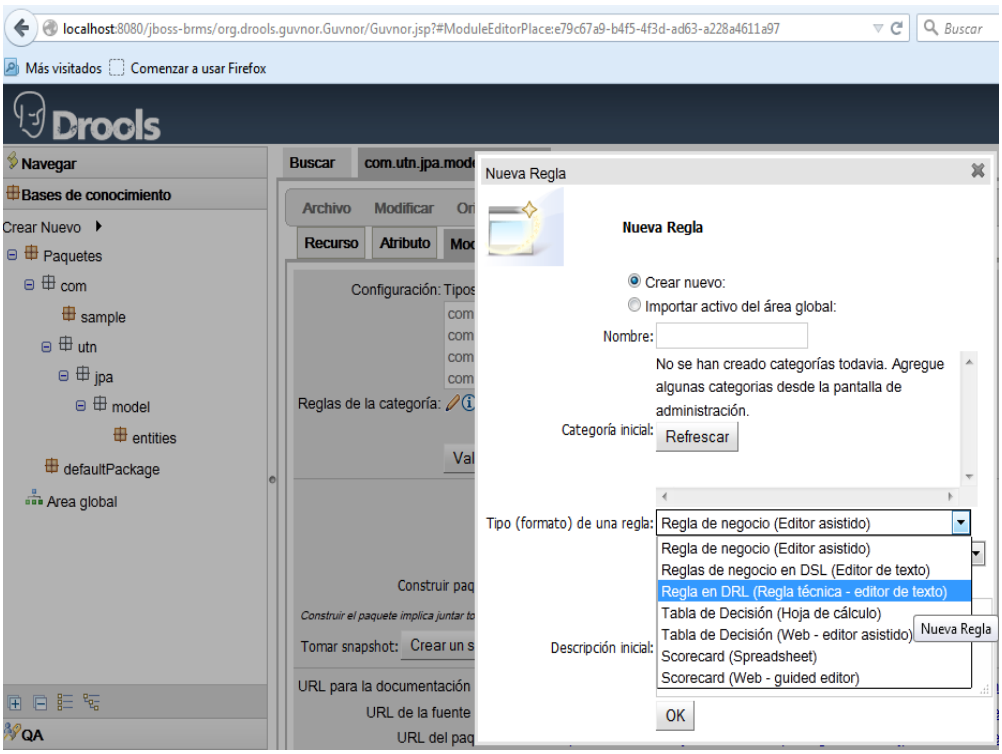


Figura 0.19: Creación de un archivo de reglas en GUVNOR
Fuente: Propia